The FASTA program package

Introduction

This documentation describes the version 36 of the FASTA program package (see W. R. Pearson and D. J. Lipman (1988), "Improved Tools for Biological Sequence Analysis", PNAS 85:2444-2448, [17] W. R. Pearson (1996) "Effective protein sequence comparison" Meth. Enzymol. 266:227-258 [15]; and Pearson et. al. (1997) Genomics 46:24-36 [18]. Version 3 of the FASTA packages contains many programs for searching DNA and protein databases and for evaluating statistical significance from randomly shuffled sequences.

This document is divided into four sections: (1) A summary overview of the programs in the FASTA3 package; (2) A guide to using the FASTA programs; (3) A guide to installing the programs and databases. Section (4) provides answers to some Frequently Asked Questions (FAQs). In addition to this document, the `changes_v36.html`, `changes_v35.html` and `changes_v34.html` files list functional changes to the programs. The `readme.v30..v36` files provide a more complete revision history of the programs, including bug fixes.

The programs are easy to use; if you are using them on a machine that is administered by someone else, you can focus on sections (1) and (2) to learn how to use the programs. If you are installing the programs on your own machine, you will need to read section (3) carefully.

FASTA and BLAST – FASTA and BLAST have the same goal: to identify statistically significant sequence similarity that can be used to infer homology. The FASTA programs offer several advantages over BLAST:

1. Rigorous algorithms unavailable in BLAST (Table I). Smith-Waterman (`ssearch36`), global: global (`ggsearch36`), and global:local (`glsearch36`) programs are available, and these programs can be used with `psiblast` PSSM profiles.

2. Better translated alignments. `fastx36`, `fasty36`, `tfastx36`, and `tfastx36` allow frame-shifts in alignments; frame-shifts are treated like gap-penalties, alignments tend to be longer in error-prone reads.

3. Better statistics. BLAST calculates very accurate statistics for protein:protein alignments, but its model-based strategy is less robust for translated-DNA:protein and DNA:DNA scores. FASTA uses an empirical estimation strategy, and now provides both search-based, and high-scoring shuffle-based statistics (`-z 21`).

4. More flexible library sequence formats. The FASTA programs can read FASTA, NCBI/ `formatdb`, and several other sequence formats, and can directly query MySQL and Postgres databases. The programs offer several strategies for specifying subsets of databases.

5. A very efficient threaded implementation. The FASTA programs are fully threaded; both similarity scores and alignments can be calculated in parallel on multi-core hardware. On multi-core machines, FASTA can be faster than BLAST while producing better alignments with more accurate statistical estimates.

6. A powerful annotation facility. The FASTA programs can incorporate functional site annotations, site variation, and domain-based sub-alignment scoring using annotations from sequence libraries. Scripts are available to download site and domain information from Uniprot,

and domain information from Pfam and CATH. Domain information can be used to sub-divide alignment scores to ensure that the aligned domain is homologous.

In addition, the FASTA programs from `fasta-36.3.4` on provide an option to produce very BLAST-like output (`-m BB`), so that analysis pipelines require minimal modification.

1 An overview of the `FASTA` programs

Although there are a large number of programs in this package, they belong to three groups: (1) Traditional similarity searching programs: `fasta36`, `fastx36`, `fasty36`, `tfastx36`, `tfasty36`, `ssearch36`, `ggsearch36`, and `glsearch36`; (2) Programs for searching with short fragments: `fasts36`, `fastf36`, `tfasts36`, `tfastf36`, and `fastm36`; (3) A program for finding non-overlapping local alignments: `lalign36`. Programs that start with `fast` search protein databases, while `tfast` programs search translated DNA databases. Table I gives a brief description of the programs.

In addition, there are several programs included. `map_db` is used to index FASTA format sequence databases for more efficient scanning. `scripts/lav2plt.pl` can plot the `.lav` files produced by `lalign -m 11` as postscript (`lav2plt.pl --dev ps`) or SVG (`lav2plt.pl --dev svg`) output.

2 Using the FASTA Package

2.1 Introduction/Overview

All the FASTA sequence comparison programs use similar command line options and arguments. The simplest command line arguments are (in order): the name of a query sequence file, a library file, and (possibly) the *ktup* parameter. If command line options are provided, they *must* precede the standard query-file and library-file arguments. Thus:

```
fasta36 -s BP62 query.file library.file
```

will compare the sequences in `query.file` with those in `library.file` using the `BLOSUM62` scoring matrix with BLASTP gap penalties (`-11/-1`).

The program can also be run by typing:

```
fasta36 -I
```

which presents the "classic" interative mode (this was the default behavior before version `36.3.4`). In interactive mode, you will be prompted for: (1) the name of the test sequence file; (2) the name of the library file; (3) whether you want ktup = 1 or 2. (1 – 6 for DNA sequences).

Current versions of the FASTA programs expect a query file and library, if you simply type "`fasta36`", you will see a short help message:

```
% ssearch36
USAGE
 ssearch36 [-options] query_file library_file [ktup]
 ssearch36 -help for a complete option list

DESCRIPTION
```

Table 1: Comparison programs in the FASTA36 package

FASTA program	BLAST equiv.	Description
`fasta36`	`blastp/` `blastn`	Compare a protein sequence to a protein sequence database or a DNA sequence to a DNA sequence database using the FASTA algorithm [15, 17]. Search speed and selectivity are controlled with the *ktup*(wordsize) parameter. For protein comparisons, *ktup* = 2 by default; *ktup* =1 is more sensitive but slower. For DNA comparisons, *ktup*=6 by default; *ktup*=3 or *ktup*=4 provides higher sensitivity.
`ssearch36`		Compare a protein sequence to a protein sequence database or a DNA sequence to a DNA sequence database using the Smith-Waterman algorithm [21]. `ssearch36` uses SSE2 acceleration, and is only 2 - 5X slower than `fasta36` [5].
`ggsearch36/` `glsearch36`		Compare a protein sequence to a protein sequence database or a DNA sequence to a DNA sequence database using an optimal global:global (`ggsearch36`) or global:local (`glsearch36`) algorithm.
`fastx36/` `fasty36`	`blastx`	Compare a DNA sequence to a protein sequence database, by comparing the translated DNA sequence in three frames and allowing gaps and frameshifts. `fastx36` uses a simpler, faster algorithm for alignments that allows frameshifts only between codons; `fasty36` is slower but can produce better alignments because frameshifts are allowed within codons [25].
`tfastx36/` `tfasty36`	`tblastn`	Compare a protein sequence to a DNA sequence database, calculating similarities with frameshifts to the forward and reverse orientations [25].
`fastf36/` `tfastf36`		Compares an ordered peptide mixture, as would be obtained by Edman degradation of a CNBr cleavage of a protein, against a protein (`fastf`) or DNA (`tfastf`) database [10].
`fasts36/` `tfasts36`		Compares set of short peptide fragments, as would be obtained from mass-spec. analysis of a protein, against a protein (`fasts`) or DNA (`tfasts`) database [10].
`lalign36`		Calculate multiple, non-intersecting alignments using the sim2 implementation of the Waterman-Eggert algorithm [22] developed by Xiaoqui Huang and Web Miller [7]. Statistical estimates are calculated from Smith-Waterman scores of shuffled sequences.

```
 FASTA searches a protein or DNA sequence data bank
 version: 36.3.8h Aug, 2019

COMMON OPTIONS (options must preceed query_file library_file)
 -s:  scoring matrix;
 -f:  gap-open penalty;
 -g:  gap-extension penalty;
 -S   filter lowercase (seg) residues;
 -b:  high scores reported (limited by -E by default);
 -d:  number of alignments shown (limited by -E by default);
 -I   interactive mode;
```

"`fasta36 -help`" (or any of the other program names in Table I) provides complete listing of the options available for the program and their default values.

The package includes several test files. To check to make certain that everything is working, you can try:

```
fasta36  ../seq/musplfm.aa ../seq/prot_test.lib
or
tfastx36 ../seq/mgstm1.aa ../seq/gst.nlib
```

2.2 Sequence files

The `fasta36` programs can read query and library files in many standard formats (Section 3.5). The default file format for query and library files – the format that will be used if no additional file format information is provided – is `FASTA` format. Like `BLAST`, version 36 can compare a query file with multiple query sequences to a sequence database, performing an independent search with each sequence in the query file.

FASTA format files consist of a description line, beginning with a '>' character, followed by the sequence itself:

```
>sequence_name1 and description
A F A S Y T .... actual sequence.
F S S       .... second line of sequence.
>sequence_name2 and description
PMILTYV ... sequence 2
```

All of the characters of the description line are read, and special characters can be used to indicate additional information about the sequence. In particular, a `'@:C 12345'` at the end of the description line indicates that the first residue of the sequence has coordinate `'12345'`, instead of starting at `'1'`. Coordinates can be negative; a DNA sequence upstream from the start of transcription could be displayed with negative coordinates.

In general, non-amino-acid/non-nucleotide sequences in the sequence lines are ignored, with the exception of `'*'`, which indicates a termination codon in a protein sequence, and can be used to indicate the match to a termination codon in protein:DNA alignments.

FASTA format files from major sequence distributors, like the NCBI and EBI, have specially formatted description lines, e.g.:

```
>np_12345| example NCBI refseq sequence
```

or

```
>sp:gstm1_human P01234 glutathione transferase GSTM1 - human
```

or `>sp|P09488|GSTM1_HUMAN glutathione transferase GSTM1 - human`

Several sample test files are included with the FASTA distribution: `seq/*.aa` and `seq/*.seq`, as well as two small sequence libraries, `seq/prot_test.lib` and `seq/gst.nlib`.

You can build your own library by concatenating several sequence files. Just be sure that each sequence is preceded by a line beginning with a '>' followed by a sequence name/description. Sequences entered with word processors should use a "text" mode, e.g. "Save as text" with MS-WORD, with end of line characters and no special formatting characters in the file. The FASTA program cannot read Microsoft Word .DOC files, or rich text (.RTF) files; query and library sequence files should contain only sequence descriptions, sequences, and end-of-line characters.

2.3 Running the programs

As mentioned earlier, the FASTA programs can be run either interactively, by typing the name of a FASTA program (and possibly command line options), followed by `-I` (`fasta36 -I`) or from the command line, entering command line options, and the query and library file names. For searches of large databases that may take several minutes (or longer), it is more convenient to run searches from the command line, e.g.:

```
fasta36 query.file library.file > output.file
```

The command line shown above could be typed in a Unix or MacOSX terminal window, or from the MS-Windows command line interface (command.exe). The command line syntax shown above works for all the FASTA programs, e.g.:

```
lalign36 mchu.aa mchu.aa > mchu.laln
fastx36 mgstm1.seq prot_test.lseg > mgstm1.fx_out
ssearch36 mgstm1.aa xurtg.aa > mgstm1_xurtg.ss
```

Command line options – The FASTA programs provide a variety of command line options that modify the default scoring matrix (`-sBL62`) and gap penalties (`-f -11`, `-g -1`), other algorithm parameters, the output options (`-E 0.1`, `-d 20`, `-m 9i`), and statistical procedures (`z -2`). A complete list of command line options is shown near the end of this document. Unlike the `BLAST` programs, all `FASTA` command line options must precede the query file name and library file name (and there are no command line options available to specify the query and library file names). Thus, you should type:

```
ssearch36 -s BL62 -f -11 -g -1 query.file library.file > output.file
```

If you include `-I` as one of the options, you can provide command line options (e.g. to change the scoring matrix or gap penalties) without a query file or library file, and the program will use the options but prompt for the necessary files .

2.4 Interpreting the results

Fig. 1 shows the output from a typical FASTA program (`ssearch36`). The output file can be viewed as four parts: (a) the initial command line and description of the query sequence used (mgstm1.aa, 218 aa) and library (PIR1, 13,143 entries); (b) a description of the search statistics, algorithm (Smith-Waterman, SSE2 accelerated), and search parameters (BLOSUM50 matrix, gap penalties: -10 to open a gap, -2 for each residue in a gap); (c) a list of high scoring library sequences, descriptions, similarity scores, and statistical significance; (d) the alignments that produced the scores.

2.4.1 Identifying homologs

In the description section (which starts: `The best scores are:`), four numbers after the description of each library sequence are shown: (i) (in parentheses) the length of the library sequence; (ii) the raw Smith-Waterman score for the alignment (`s-w`; for the `fasta36`, `[t]fast[x,y]36` programs, this column would be labeled `opt`, for the *opt*imized – banded Smith-Waterman – score), (iii) the *bit* score, and (iv) the expectation (E()), or statistical significance, of the alignment score. The E()-value depends on the size of the database searched, in this case, 13,143 sequences, so the database size is given at the top of the list.

The bit score is equivalent to a BLAST bit score; together with the length of query and library sequences, it can be used to calculate the significance of the alignment.[1] Bit scores are convenient because they provide a matrix independent score that can be compared with other searches performed with other matrices and gap penalties against other databases. However, the E()-value, or expectation, provides the most direct measure of the statistical significance of the match.

In this example, the `GSTP1_RAT`, `GSTA1_RAT`, and `GSTA4_RAT` proteins share strong significant similarity (better than $E() < 6.1 \times 10^{-7}$), while the `GSTF1_MAIZE`, `GSTF3_MAIZE`, and `GSTT1_DROME` sequences do not share significant similarity ($E() < 0.001$). However, `GSTF1_MAIZE`, `GSTF3_MAIZE`, and `GSTT1_DROME` are all glutathione transferase homologs, they simply do not share statistically significant similarity with this particular `mGSTM1` query. Statistically significant sequence similarity scores *can* be used to infer *homology* (common ancestry), but non-significant scores *cannot* be used to infer *non-homology*.

While percent identity is often used to characterize the quality of an alignment and the likelihood that it reflects homology, the E()-value is a much more reliable value for homology infernence (once homology is established, the percent identity is much more useful for estimating evolutionary distance). Often sequences that share less than 30% identity will share very significant similarity (in the example above `mgstm1.aa` and `GSTA4_RAT`, with E() $<$ 6.1E-07 are 25.6% identical). The expectation value captures information about conservative replacements, identities, and alignment length to provide a *single* value that captures the significance of the alignment.

For protein searches, library sequences with E()-values $<$ 0.001 for searches of a 10,000 entry protein database are almost always homologous. Some sequences with E()-values from 1 - 10 may also be related, but unrelated sequences (1–10 per search) will have scores in this range as well.

E()-values $<$ 0.001 can reliably be used to infer homology, assuming that the statistical estimates are accurate. The two most common causes of statistical problems are low-complexity regions and amino-acid composition bias. Low-complexity regions are can be identified using the `pseg` program [23], and filtered out using the `-S` option. Composition bias rarely produces highly-signficiant E()-values, but can cause unrelated sequences to have E()-values between 0.01 and 0.001. The FASTA programs offer two shuffle-based strategies for evaluating composition bias; calculating similarity scores for random sequences with the same length and amino acid compo-

[1] $E(D) = Dmn2^{-b}$, where D is the number of sequences in the database, m, n are the lengths of the two sequences, and b is the bit score.

```
# ../bin/ssearch36 -q -w 80 ../seq/mgstm1.aa a
SSEARCH performs a Smith-Waterman search
 version 36.3.6 June, 2013(preload9)
Please cite:
 T. F. Smith and M. S. Waterman, (1981) J. Mol. Biol. 147:195-197;
 W.R. Pearson (1991) Genomics 11:635-650
Query: ../seq/mgstm1.aa
  1>>>mGSTM1 mouse glutathione transferase M1 - 218 aa
Library: PIR1 Annotated (rel. 66)
  5121825 residues in 13143 sequences

Statistics:  Expectation_n fit: rho(ln(x))= 7.4729+/-0.000484; mu= 2.0282+/- 0.027
 mean_var=56.9651+/-10.957, 0's: 9 Z-trim(119.4): 17  B-trim: 67 in 1/62
 Lambda= 0.169930
 statistics sampled from 13135 (13143) to 13135 sequences
Algorithm: Smith-Waterman (SSE2, Michael Farrar 2006) (7.2 Nov 2010)
Parameters: BL50 matrix (15:-5), open/ext: -10/-2
 Scan time:  3.820
The best scores are:                                                    s-w bits E(13143)
sp|P08010|GSTM2_RAT Glutathione S-transferase Mu 2; GST 4-4; GT  ( 218) 1248 312.0 7.7e-86
sp|P04906|GSTP1_RAT Glutathione S-transferase P; Chain 7; GST -  ( 210)  344 90.4 3.8e-19
sp|P00502|GSTA1_RAT Glutathione S-transferase alpha-1; GST 1-1   ( 222)  237 64.1 3.2e-11
sp|P14942|GSTA4_RAT Glutathione S-transferase alpha-4; GST 8-8   ( 222)  179 49.9 6.1e-07
sp|P12653|GSTF1_MAIZE Glutathione S-transferase 1; GST class-pi  ( 214)  120 35.4   0.013
sp|P04907|GSTF3_MAIZE Glutathione S-transferase 3; GST class-pi  ( 222)  115 34.2   0.032
sp|P20432|GSTT1_DROME Glutathione S-transferase 1-1; DDT-dehydr  ( 209)  100 30.5    0.38
sp|P11277|SPTB1_HUMAN Spectrin beta chain, erythrocytic; Beta-   (2137)  108 31.6     1.9
... (alignments deleted) ...
>>sp|P14942|GSTA4_RAT Glutathione S-transferase alpha-4; GST 8-8;             (222 aa)
 s-w opt: 179  Z-score: 231.0  bits: 49.9 E(13143): 6.1e-07
Smith-Waterman score: 179; 25.6% identity (54.5% similar) in 211 aa overlap (5-207:7-207)
                 10        20        30        40        50          60         70
mGSTM    MPMILGYWNVRGLTHPIRMLLEYTDSSYDEKRYTMGDAPDFDRSQWLNEKF-KLG-LDFPNLPYL-IDGSHKITQSNA
             : :.. ::  . :: ::  .   ..:         .: ...   ::. : : : : ..: . :::   .::. :
sp|P14 MEVKPKLYYFQGRGRMESIRWLLATAGVEFEE---------EFLETREQYEKLQKDGCLLFGQVPLVEIDG-MLLTQTRA
               10        20        30                 40        50        60         70
          80        90       100       110       120       130       140            150
mGSTM  ILRYLARKHHLDGETEEERIRADIVENQVMDTRMQLIMLCYNPDFEKQKPEFLKTIPEKMKLYSEF--LGK---RPWFAG
       :: ::: :..: :.  .::.: :.  . ..:  :..:   ..   ::..   : .   : . .  : .  :   . ...:
sp|P14 ILSYLAAKYNLYGKDLKERVRIDMYADGTQDLMMMIIGAPFKAPQEKEESLALAVKRAKNRYFPVFEKILKDHGEAFLVG
               80        90       100       110       120       130       140       150
              160       170       180       190       200       210
mGSTM  DKVTYVDFLAYDILDQYRMFEPKCLDAFPNLRDFLARFEGLKKISAYMKSSRYIATPIFSKMAHWSNK
       ......:.   . . . .      :. :: :. : .:. ..  :. ... .     :
sp|P14 NQLSWADIQLLEAILMVEEVSAPVLSDFPLLQAFKTRISNIPTIKKFLQPGSQRKPPPDGHYVDVVRTVLKF
              160       170       180       190       200       210       220
... (alignments deleted) ...
218 residues in 1 query   sequences
5121825 residues in 13143 library sequences
 Tcomplib [36.3.6 May, 2013(preload9)] (4 proc in memory [0G])
 start: Thu Jun  6 11:23:28 2013 done: Thu Jun  6 11:23:30 2013
 Total Scan time:  3.820 Total Display time:  0.130
Function used was SSEARCH [36.3.6 May, 2013(preload9)]
```

Figure 1: `ssearch36` results

Comparison of `seq/mgstm1.aa` against a small protein database (`pir1.lseg`). Some high-scoring sequences and all but one alignment were removed to reduce the output size.

sition (`-z 11 .. 16`),[2] and calculating statistical estimates derived from shuffles of the high-scoring sequences (`-z 21, 22, 24, 25, 26`).

When `-z 21 .. 26` shuffles are performed, the FASTA36 programs present two E()-values in the list of high scoring sequences and the alignments; the traditional one based on the library search, and a second `E2()` value, based on the shuffles of the high scoring sequences. `-z 21 .. 26` shuffles are most useful for evaluating the significance of translated-DNA:protein searches like `fastx36`. Out-of-frame translations can produce cryptic low complexity regions, which are most apparent when the high-scoring sequences are shuffled. `-z 21` shuffles are more efficient than individual sequence shuffles, because the set of high scoring sequences is shuffled 500 – 1,000 times, rather than 500 shuffles for each of 50 – 100 high scoring library sequences. It is almost as effective, because homologous sequences share similar amino-acid composition.

The statistical routines assume that the library contains a large sample of unrelated sequences. If the library contains fewer than 500 sequences (`MAX_RSTATS`), then the library sequences are shuffled to produce 500 random scores, from which lambda and K statistical parameters are estimated. If the library contains a large number of *related* sequences, then the statistical parameters should be estimated by using the `-z 11-15`, options. `-z` options greater than 10 calculate a shuffled similarity score for each library sequence, in addition to the unshuffled score, and estimate the statistical parameters from the scores of the shuffled sequences.

2.4.2 Looking at alignments

The description section described above contains the critical information for inferring homology, the `E()`-value. The alignment section shows the actual alignments that produced the similarity score and statistical estimates. In Fig. 1, the alignment display reports the percent identity, percent similarity (number of aligned residues with BLOSUM50 values ≥ 0), and the boundaries of the alignment. Note that for the `ssearch36` and `fasta36`, the alignment shown can include residues that are not part of the best local alignment (e.g. residues 1–5 and 207–218 in `mGSTM1` in Fig. 1). The amount of additional sequence context shown is the alignment line length (60 residues, set by `-w len`) divided by 2 by default, but can be adjusted with the `-W context` option.

Fig. 2 shows an example of a `fasta36` alignment produced using the `-S` option to filter out lower-case (low complexity) residues. Here, additional scores (`initn`, `init1` are shown, in addition to the `opt` score which is used to rank the sequences and calculate statistical significance. The `init1` score is the highest scoring alignment without gaps; `initn` is a score that combines consistent (non-overlapping) runs without gaps, and `opt` is the score of a banded Smith-Waterman of width 16 for *ktup=2* that is applied to sequences with `initn` scores over the optimization threshold. In Fig. 2, the `init1` score is based on the long, un-gapped region from residues 46–208 in `mGSTM1`, while the `initn` and `opt` scores include the other regions joined by gaps. The `initn` score is higher than the `opt` score, because it uses a simpler, length-independent, gap penalty.

The `init1` and `initn` scores are shown for historical reasons, and can be used to illustrate the FASTA algorithm. But the `opt` score is the most reliable and sensitive score for inferring homology; the others can be ignored.

For `fasta36` with proteins, the final alignment and score is calculated with the Smith-Waterman algorithm. For DNA sequences, a banded Smith-Waterman is used. (The `-A` option produces banded Smith-Waterman alignments for proteins, and full Smith-Waterman for DNA.) In Fig. 2, the `opt` score and `Smith-Waterman` scores are calculated on exactly the same alignment, but the `opt`

[2]Random shuffles are performed for pairwise alignments and `lalign36` by default.

```
>>GST26_SCHMA Glutathione S-transferase class-mu  (218 aa)
 initn: 422 init1: 359 opt: 407  Z-score: 836.8  bits: 162.0 E(437847): 3.7e-39
Smith-Waterman score: 451; 42.4% identity (73.4% similar) in 203 aa overlap (6-208:6-203)

               10        20        30        40        50        60        70        80
mGSTM1 MPMILGYWNVRGLTHPIRMLLEYTDSSYDEKRYTMGDAPDFDRSQWLNEKFKLGLDFPNLPYLIDGSHKITQSNAILRYL
            :::.:.::..: :.:::. ...:.:. :   :  ..:   : :.::::::.:::::: :::. :.::: ::.::.
GST26_ MAPKFGYWKVKGLVQPTRlllehleetyeeRAY---DRNEIDA--WSNDKFKLGLEFPNLPYYIDGDFKLTQSMAIIRYI
               10        20        30           40          50        60        70

               90       100       110       120       130       140       150       160
mGSTM1 ARKHHLDGETEEERIRADIVENQVMDTRMQLIMLCYNPDFEKQKPEFLKTIPEKMKLYSEFLGKRPWFAGDKVTYVDFLA
       : ::.. :   .:: . ...:. :.: :: .. ..:: ..:  : .::. .: ..:.... :... .. :. ::. ::.
GST26_ ADKHNMLGACPKERAEISMLEGAVLDIRMGVLRIAYNKEYETLKVDFLNKLPGRLKMFEDRLSNKTYLNGNCVTHPDFML
          80        90       100       110       120       130       140       150

              170       180       190       200       210
mGSTM1 YDILDQYRMFEPKCLDAFPNLRDFLARFEGLKKISAYMKSSRYIATPIFSKMAHWSNK
       :: ::    .. .::. ::.: .:   .: : .:. :..:::::  :.
GST26_ YDALDVVLYMDSQCLNEFPKLVSFKKCIEDLPQIKNYLNSSRYIKWPLQGWDATFGGGDTPPK
         160       170       180       190       200       210
```

Figure 2: Alignment with `-S` filtered sequence

score excludes the contribution from the "low-complexity" region between 19–30 in `GST26_SCHMA`.

2.4.3 Results without alignments

While sequence alignments are very informative, it is often not practical to examine all the statistically significant alignments in large-scale searches. The `-m 9` and `-m 8` options present summaries of each alignment (alignment boundaries, percent identity, and other information) in a much more compact form. `-m 9i` adds three columns to the summary report line, the fraction identical, fraction similar, and alignment length (in addition, variants are reported if they affect identity). `-m 9I` is similar to `-m 9i`, but also reports domain content if annotations are used. `-m 9c` or `-m 9C` (see options below) provide a detailed encoding of the alignment, that allows it to be reconstructed. For large-scale searches, we routinely use `-m 8` with the `-d 0` option, which sets the number of alignments shown to 0 (thus none are shown). Alternatively, the `-m 8` and `-m 8C` ouput options produce BLAST-format tabular results summaries (`-m 8C` provides commented tabular results). `-m 8CC` adds an alignment CIGAR string and annotation string to the BLAST tabular format. `-m BB` produces an output that mimics BLAST output (with alignments).

2.4.4 Alignments with annotations

The command line `-V` option (described below) causes the FASTA programs to "decorate" its sequence alignments with annotation information, such as functional sites, variants, and domain-based sub-alignment scores. For example, a comparison of the `seqs/gstm1_human.vaa` sequence with SwissProt using the `scripts/ann_feats_up_www2.pl` script:

```
     ssearch36 -m 9i -V \!../scripts/ann_feats_up_www2.pl ../seq/gstm1\_human.vaa /slib/swissprot.fa
```

Produces the following addtional output:

```
Annotation symbols:
 = : Active site
 * : Modified
 # : Substrate binding
 ^ : Metal binding
 @ : Site

The best scores are:                       s-w bits E(458668) %_id  %_sim  alen
sp|P09488.3|GSTM1_HUMAN Glutathione ( 218) 1500 375.2 2.4e-103 1.000 1.000  218 |Var: K173N;S210T;
sp|Q03013.3|GSTM4_HUMAN Glutathione ( 218) 1375 344.5 4.4e-94 0.904 0.963  218 |Var: S2P;A160V;L208V;Y209F;...
sp|Q5R8E8.3|GSTM2_PONAB Glutathione ( 218) 1310 328.5 2.9e-89 0.862 0.959  218
sp|Q9TSM5.3|GSTM1_MACFA Glutathione ( 218) 1308 328.0   4e-89 0.858 0.954  218
sp|Q9TSM4.3|GSTM2_MACFA Glutathione ( 218) 1307 327.7 4.8e-89 0.862 0.954  218
sp|P28161.2|GSTM2_HUMAN Glutathione ( 218) 1306 327.5 5.7e-89 0.853 0.959  218 |Var: S173N;
sp|P46439.3|GSTM5_HUMAN Glutathione ( 218) 1305 327.2 6.7e-89 0.876 0.954  218 |Var: L179P;
```

This summary of high scoring hits shows one of the effects of annotation—substitution of variant residues to increase the score. In this case, the query sequence `gstm1_human.vaa` is a known variant of the canonical `GSTM1_HUMAN/P09488` UniProt sequence. Without the `-V` annotation option, `gstm1_human.vaa` would be 99% identical to `GSTM1_HUMAN`, but because UniProt documents the variant residues in the feature table, the `K173N` and `S210T` substutions are made in the library (subject) sequence, producing a perfect match.

In addition to the variant substitution shown above, the alignments provide a more complete view of the annotations available on the library (subject) proteins. Below is the report for the `GSTM4_HUMAN` alignment:

```
>>sp|Q03013.3|GSTM4_HUMAN Glutathione S-trans            (218 aa)
 Variant: 2P=2P : S2P : UniProtKB FT ID: VAR_033979
 Site:@ : 7Y=7Y : Site: Glutathione binding
 Site:@ : 46W=46W : Site: Glutathione binding
 Site:@ : 59N=59N : Site: Glutathione binding
 Site:@ : 72Q=72Q : Site: Glutathione binding
 Region: 2-88:2-88 : score=599; bits=150.1; Id=0.989; Q=415.2 :  GST N-terminal :1
 Site:# : 116Y=116Y : Substrate binding: Substrate
 Variant: 160V=160V : A160V : UniProtKB FT ID: VAR_033980
 Variant: 208V=208V : L208V : UniProtKB FT ID: VAR_049487
 Region: 90-208:90-208 : score=699; bits=175.1; Id=0.833; Q=489.3 :  GST C-terminal :2
 Variant: 209F=209F : Y209F : UniProtKB FT ID: VAR_049488
 Variant: 211K=211K : R211K : UniProtKB FT ID: VAR_049489
 Variant: 212M=212M : V212M : UniProtKB FT ID: VAR_049490
 s-w opt: 1375  Z-score: 1823.2  bits: 344.5 E(458668): 4.4e-94
Smith-Waterman score: 1375; 90.4% identity (96.3% similar) in 218 aa overlap (1-218:1-218)
```

This report can be broken into three parts: (1) information on variants, described above, (2) information on annotated sites, and (3) information on annotated domains. For each annotated site, the coordinate and amino-acid resdiue in the query and library (subject) sequence is shown, as well as the conservation state (= in all these examples). For annotated domains, the overall alignment score is broken into pieces, based on the boundaries of the domains. In this case, the full alignment extends from residues 1–218, producing a raw Smith-Waterman score of 1375 and a bit score of 344.5. The GST N-terminal domain is annotated from residue 2–88 on `GSTM4_HUMAN`, and the 2–88 region of the alignment produces a score of 599 and bit score of 150.1. This region is 98.9% identical, and the probability of that similarity score is $10^{-41.52}$ (the Qvalue score is $-10log_{10}P$). The alignment associated with GST C-terminal domain is slightly less well conserved (83.3% identical), but longer, so it produces a higher Smith-Waterman score (699), bit score (175.1) and Q-value.

Sub-alignment scores can be used to identify cases of alignment over-extension [11], where an alignment extends well beyond the homologous domain. In this case, the homologous region will produce the vast majority of the score, and the non-homologous over-extension will produce very little score. For example, when SRC8_HUMAN aligns with LASP1_MOUSE, the alignment spans 200 residues, but only about 49 of those residues, an SH3 domain, are homologous:

```
>>sp|Q61792.1|LASP1_MOUSE LIM and SH3 domain protein 1;  LASP-1;              (263 aa)
 Region: 369-398:66-95 : score=20; bits=13.7; Id=0.200; Q=0.0 :  Nebulin_repeat  InterPro
 Region: 400-434:97-131 : score=-8; bits=8.7; Id=0.150; Q=0.0 :  Nebulin_repeat  InterPro
 Region: 435-499:132-203 : score=13; bits=11.4; Id=0.197; Q=0.0 :  NODOM :0
 Region: 499-547:204-261 : score=124; bits=47.8; Id=0.474; Q=92.1 :  SH3  InterPro
 s-w opt: 148  Z-score: 253.4  bits: 55.6 E(459565): 1.2e-06
Smith-Waterman score: 159; 26.3% identity (55.6% similar) in 205 aa overlap (369-547:66-261)
```

The 150 aligned residues outside the SH3 homology produce less than 20% of the alignment score, while spannign to non-homologous Nebulin repeat domains.

2.5 Program Options

Command line options are available to change the scoring parameters and output display. Unlike the NCBI BLAST programs, command line options *must* precede the query file name and library file name arguments. To see the command-line options for a program and their defaults, type `program_name -help`, e.g. `fasta36 -help` or `ssearch36 -help`. For a quick list of the most common options, just type the program name without any options (e.g. `fasta36<ret>`).

2.5.1 Command line options

`-a` (`fasta36`, `ssearch36`, `glsearch36`, `fasts36`) show both sequences in their entirety.

`-A` force Smith-Waterman alignments for `fasta36` DNA sequences. By default, only `fasta36` protein sequence comparisons use Smith-Waterman alignments. Likewise, for proteins, use band alignments (Smith-Waterman is used by default).

`-b #` Number of sequence scores to be shown on output. In the absence of this option, `fasta36` (and `ssearch36`) display all library sequences obtaining similarity scores with expectations less than the expectation (-E) threshold, 10.0 for proteins, and 2.0 for DNA:DNA and protein:translated DNA. The `-b #` option can limit the display further. There are two "submodes" of `-b`. `-b =100` will force 100 high scores to be displayed, regardless of the expectation (`-E`) threshold, and `-b >1` will show at least `1`, but is otherwise limited by `-E`. Thus, `-b 10` will show *no more than* 10 results, limited by `-E`; `-b =10` will always show *exactly* `10` results, and `-b >5` will show *at least* `5` results, but could show many more if more results have evalues $\leq$ `-E e_cut`.

`-c #,#` (`fasta36`, `[t]fast[x,y]36` only) Fraction of alignments optimized (second value is fraction of sequences joined). FASTA36 uses a statistical threshold strategy that joins and optimizes only the fraction of the alignments with an `initn` score expected `-c` times. Thus, `-c 0.05` should optimize about 5% of sequences. The actual number of sequences optimized (and joined) is displayed in the scoring parameters line. Thus:

```
Parameters: BL50 matrix (15:-5), open/ext: -10/-2
 ktup: 2, E-join: 1 (0.687), E-opt: 0.2 (0.294), width:  16
```

reports that 20% of the sequences in the database should have been band-optimized, and 29.4% were. Reducing the `-c opt` fraction improves performance, but dropping the fraction below 0.02 can reduce the accuracy of the statistical estimates.

`-c O` (letter 'O') sets the joining/optimization thresholds as they were prior to `fasta-36.3.3` (original thresholds). Positive values set the thresholds to specific score values, as was the case in older versions of `fasta`.

`-C` length of the sequence name printed at the beginning of alignment lines (default 6 characters).

`-d #` Maximum number of alignments to be displayed (must be <= to the number of descriptions, `-b #`)

`-D` Provide some debugging output. Used in conjunction with the `-e expand_script.sh`, the `link_acc_file` and `link_lib_file` are not deleted during the run; so that `expand_script.sh` scripts can be tested.

`-e expand_script.sh` Expand the set of sequences that are aligned to beyond the set of sequences searched. When the `-e expand_script.sh` option is used, the `expand_script.sh` script is run after the initial search scan but before the list of high-scoring sequences is displayed. `expand_script.sh` is given a single argument, the name of a file that contains a list of accession strings (the text between the > and the first space ('␣') character followed by the E()-value for the sequence (separated by a `<tab>` character), e.g.:

```
gi|121719|sp|P08010|GSTM2_RAT<tab>2.69e-86
gi|121746|sp|P09211|GSTP1_HUMAN<tab>1.51e-20
gi|121749|sp|P04906|GSTP1_RAT<tab>1.16e-19
gi|62822551|sp|P00502|GSTA1_RAT<tab>9.5e-12
```

The script should produce a fasta-formatted list of additional sequences printed to `stdout`. The script is run with the command:

```
expand_script.sh link_acc.tmp_file > link_lib.tmp_file
```

The sequences in `link_lib.tmp_file` (a temporary file name is actually used, and the file is deleted unless the `-D` option is used) are then compared and, if they are significant, included in the list of high scoring sequences and the alignments. The expanded set of sequences does not change the database size or statisical parameters, it simply expands the set of high-scoring sequences.

The `fasta36/misc` directory contains `expand_uniref50.pl` that uses a mySQL table based on the `uniref50` clusters. Using the script and the `uniref50` cluster information, one can search `uniref50.fasta`, but then expand the hits so that `uniprot` appears to be searched.

`-E e_cut [e_cut_r]` Limit the number of scores and alignments shown based on the expected number of scores. Used to override the expectation value of 10.0 (protein:protein; 5.0 translated-DNA:protein; 2.0 DNA:DNA) used by default. `-E 2.0` will show all library sequences with scores with an expectation value $<=$ 2.0. With `fasta-36`, a second value, `e_cut_r` is available to limit the E()-values of additional sequence alignments between the query and library sequences. If not given, the threshold is `e_cut`/10.0. If given with a value $>$ 1.0, `e_cut_r` = `e_cut` / value; for a value $<$ 1.0, `e_cut_r` = value; If `e_cut_r` $<$ 0, then the additional alignment option is disabled.

`-f #` Gap open penalty (-10 by default for proteins, -12 for DNA, -12 for `[t]fast[xy]`).

`-F #` Limit the number of scores and alignments shown based on the expected number of scores. `-E #` sets the highest E()-value shown; `-F #` sets the lowest E()-value displayed. Thus, `-F 0.0001` will not show any matches or alignments with E() $<$ 0.0001. This allows one to skip over close relationships to search for more distant relationships.

`-g #` Penalty per residue in a gap (-2 by default for proteins, -4 for DNA, -2 for `[t]fast[xy]`). A single residue gap costs `f` + `g`.

`-h` Short help message. Help options with `':'`, e.g. `-s:`, require an argument (`-s BP62`). Defaults are shown in square brackets, e.g.: `-s: [BL50]`.

`-help` Long help message

`-H` Show histogram.

`-i` DNA queries - search with reverse complement. For `tfastx36/y36`, search the reverse complement of the library sequence only (complement of `-3` option).

`-I` Interactive mode (the default for versions older than `fasta-36.3.4`).

`-j #` Penalty for frameshift between codons (`[t]fastx36`, `[t]fasty36`) and within a codon (`fasty36/ tfasty36` only).

`-J` (`lalign36` only) show the identity alignment (normally suppressed, `-I` in versions before `fasta-36.3.4`).

`-k #` number of shuffles for statistical estimates from shuffling.

`-l file` Location of library menu file (FASTLIBS).

`-L` Display longer library sequence description.

`-M low-high` Range of amino acid sequence lengths to be included in the search.

`-m #` Specify alignment type: 0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, B, BB, "F# out_file"

```
    -m 0        -m 1         -m 2         -m 3        -m 4
MWRTCGPPYT  MWRTCGPPYT   MWRTCGPPYT               MWRTCGPPYT
::..:: :::    xx  X      ..KS..Y...   MWKSCGYPYT  ----------
MWKSCGYPYT  MWKSCGYPYT
```

If the `-V '*@%'` annotation option has been used, annotations can be included in either the coordinate line (the default) or the middle alignment line (`-m 0M`, `-m 1M`), or both (`-m 0B`, `-m 1B`). See `-V` for more details.

`-m 5`: a combination of `-m 4` and `-m 0`. `-m 6` provides `-m 5` plus HTML formatting. In addition, independent `-m` options can be combined. Thus, one can use `-m 1 -m 6 -m 9`.

`-m 8` provides BLAST tabular format output (a tab delimited line with the query name, library name, percent identity, and other alignment information). "`-m 8C`" provides the additional information provided by the BLAST tabular format with comment lines. BLAST tabular format has been extended to include either a CIGAR string alignment encoding (`-m 8CC` with BLAST comments, `-m 8XC` without comments) and, if available, an annotation encoding matching FASTA `-m 9C` output. `-m 8CB` provides the BLAST BTOP encoding, rather than the CIGAR encoding provided by `-m 8CC`. All the `-m 9c/C/d/D` encodings are available with BLAST tabular output using `-m 8C[c/C/d/D]`. In addition to the BLAST BTOP/CIGAR encoding field added with `-m 8CB/8CC`, a second additional annotation field is added if annotations are available. The annotation field provides the same annotation boundary and score information as in standard alignment, but in a more compact form:

```
|RX:3-82:3-82:s=780;b=201.;I=0.866;Q=570.;C=C.Thio|XR:3-82:3-82:s=780;b=201.;I=0.866;Q=5
```

`|RX` annotations are based on the query sequence; `|XR` on the subject. In the example above, the boundaries of the domain in the query and subject are shown (`3-82:3-82`, as well as the raw score, bit-score, identity, Q-score, and domain name.

In the v36.3.8h release, a new option has been added to `-m 8CB`, `-m 8CBL` (or `-m 8CBl`. The `- m8CBl` option adds the lengths of the query and subject sequences after the `seqid`'s to BLAST tabular output, e.g. `qseqid qlen sseqid slen percid ...`

`-m 8CBL` does the same thing, but also adds a new field with the raw domain coordinate information for the query and subject, with `DX/XD` used to indicate the domains.

```
|DX:3-82;C=C.Thio|DX:105-201;C=C.GST_C|XD:3-82;C=C.Thio|XD:105-201;C=C.GST_C
```

`-m 9` display alignment coordinates and scores with the best score information. `-m 9i` provides alignment length, percent identity, and percent similarity only. `-m 9i` also provides variation information if it improves the score. `-m 9I` provides both identity and domain information on the summary line.

`-m 9, -m 9c` and `-m 9C` extend the normal best score information:

```
The best scores are:                                      opt bits E(14548)
XURTG4 glutathione transferase (EC 2.5.1.18) 4 -    ( 219) 1248 291.7 1.1e-79
```

to include the additional information (on the same line, separated by <tab> characters):

```
%_id  %_gid   sw  alen  an0  ax0  pn0  px0  an1  ax1 pn1 px1 gapq gapl  fs
0.771 0.771 1248  218    1  218    1  218    1  218    1  219   0    0   0
```

The first two values are fraction identical and fraction similar (score ≥ 0), followed by the Smith-Waterman alignment score (`sw`), the alignment length (`alen`), and the coordinates of the beginning and end of the alignment in the query and target (library) sequences (`an0` beginning, `ax1` end in query; `an1` beginning, `ax1` end in target/library), and the coordinate system for the beginning and end of the query and target/library sequence (`pn0` is the displayed coordinate of the first residue of the query sequence, `px0` is the displayed coordinate of the last residue, `pn1`,`px1` provide the coordinates for the target/library sequence). `gapq`, `gapl` report the number of gaps in the query and library sequence; `fs` reports the number of frameshifts.

`-m 9c` provides additional information: an encoded alignment string. For example, the alignment:

```
       10        20        30        40        50          60         70
GT8.7  NVRGLTHPIRMLLEYTDSSYDEKRYTMGDAPDFDRSQWLNEKFKL--GLDFPNLPYL-IDGSHKITQ
       :.::  . :: ::  .   .:::         : .:    ::.:   .: : ..:.. :::  :..:
XURTG  NARGRMECIRWLLAAAGVEFDEK---------FIQSPEDLEKLKKDGNLMFDQVPMVEIDG-MKLAQ
               20        30                 40        50        60
```

would be encoded: `=23+9=13-2=10-1=3+1=5` . The numbers in the alignment encoding is with repect to the beginning of the alignment, not the sequences. The beginning coordinate of the alignment is given earlier in the `-m 9c` line. `-m 9C` provides the alignment encoding in CIGAR format: `28M9D13M2I10M1I3M1D5M` .

(June, 2014) The `-m 9c/C` option has been extended to `-m 9d/D`, which encodes the positions of mismatches as well as insertions and deletions. For the example above, the `-m 9d` encoding would be: `=1x1=2x4=2x1=2x7=3-9=1x2=1x4=2x1=1x1+2x1=1x1=1x3=1x2+1=3-1x1=1x2=1` while `-m 9D` would be: `1M1X2M4X2M1X2M7X3M9D1M2X1M4X2M1X1M1X2I1X1M1X1M3X1M2X1I3M1D1X1M2X1M`

`-m 10` a parseable format for use with other programs (this option no longer reliably tested; `-m 8CBL` is easier to parse and tested more extensively).

`-m 11` Provide `lav`-like output (used by `lalign`) for graphical output.

```
lalign36 -m 11 mchu.aa mchu.aa | lav2plt.pl --dev ps > mchu_laln.ps
```

Produces a postscript plot of the local alignments. Likewise, `lav2plt.pl --dev svg` produces SVG output.

`-m BB` Format output to mimic BLAST format. `-m B` formats alignments to look like BLAST alignments (Query/Sbjct), but is FASTA output otherwise. `-mBB` imitates BLAST as much as possible, and cannot be used with other `-m` options.

`-m "F# out.file"` Send an alternate result format to `out.file`. Normally, the `-m out_fmt` option applies to the default output file, which is either `stdout`, or specified with `-O out_file` (or within the program in interactive mode). With `-m F`, an output format can be associated with a separate output file, which will contain a complete FASTA program output. Thus,

```
ssearch36 -m 9c -m "FBB blast.out" -m "F9c,10 m9c_10.out" query library
```

Sends the `-m 9c` output to `stdout`, but will also send `-m BB` output to the `blast.out` file, and `-m 9c -m 10` output to `m9_c10.out`. Consistent `-m out_fmt` commands can be set to the same file by separating them with ','. Producing alternative format alignments in different files has little additional computational cost.

Because a space (␣) is used to separate the output format (`-m`) values from the file name, the `-m F` argument must typically be surrounded by quotation marks (`"`).

One of the shortcomings of this approach is that it affects only the output format, not the other options that modify the amount of output. Thus, if you specify `-E 0.001`; that expect threshold will be used for all the output files. When a `-m` option does modify the output (e.g. `-m 8` sets `-d 0`), that modification is specific to the output file.

`-M low-high` Include library sequences with lengths between low and high.

`-n` Force the query sequence to be treated as a DNA sequence. Useful when query sequences contain a large number of ambiguous residues, e.g. transcription factor binding sites.

`-N #` break long library sequences into blocks of # residues. Useful for bacterial genomes, which have only one sequence entry. -N 2000 works well for well for bacterial genomes. (This option was required when FASTA only provided one alignment between the query and library sequence. It is not as useful, now that multiple alignments are available.)

`-o off1,off2` (Previously `-X`.) Specifies offsets for the beginning of the query and library sequence. For example, if you are comparing upstream regions for two genes, and the first sequence contains 500 nt of upstream sequence while the second contains 300 nt of upstream sequence, you might try:

```
fasta -o "-500 -300" seq1.nt seq2.nt
```

If the `-o` option is not used, FASTA assumes numbering starts with 1. (You should double check to be certain the negative numbering works properly.)

`-O` Send a copy of results to `filename`. Helpful for environments without STDOUT, but should be avoided (use `> filename` instead).

`-p` Force query to be treated as protein sequence.

`-P PSSM_file` Specify a PSI-BLAST format PSSM (Position Specific Scoring Matrix) file. `ssearch36`, `ggsearch36`, and `glsearch36` can use a PSSM file to improve the sensitivity of a search. The FASTA programs accept two PSSM file formats:

format	`blastpgp`	option
0	`blastpgp -C pssm.chk -u 0`	byte-encoded
2	`blastpgp -C pssm.asnb -u 2`	binary ASN.1

which can be specified after the file name, e.g.:

```
ssearch36 -P 'pssm.asnb 2' pssm_query.aa +sp+
```

Searches with a PSI-BLAST PSSM must still require a query sequence file, and the query sequence file must match the PSSM seed sequence. The format 0 byte-encoded PSSM is machine dependent; it must be created by `blastpgp` on the same architecture as `ssearch36`. In general, you should use the binary ASN.1 (format 2) file.

With the release of `NCBI-BLAST+`, `psiblast` replaces `blastpgp`, and `psiblast` does not produce the binary ASN.1 PSSM checkpoint data. However, the text ASN.1 PSSM checkpoint file (produced with the `psiblast` option `-out_pssm`) can be converted to a binary ASN.1 format that `ssearch36` can read using the NCBI `datatool` program (available from `ftp://ftp.ncbi.nlm.nih.gov/toolbox/ncbi_tools++/BIN/CURRENT/datatool`) together with `http://www.ncbi.nlm.nih.gov/data_specs/asn/NCBI_all.asn`. More information about `datatool` is available from `http://www.ncbi.nlm.nih.gov/data_specs/NCBI_data_conversion.html`. The NCBI BLASTP/PSI-BLAST website provides the same PSSM text ASN.1 file with the downloads link. A text ASN.1 PSSM file can be converted to a binary ASN.1 file using the command:

```
datatool -m NCBI_all.asn -v pssm.asn_txt -e pssm.asnb
```

The `pssm.asnb` can then be used with `ssearch36` with the `-P 'pssm.asnb 2'` option shown above.

`-Q,-q` Quiet - does not prompt for any input. Writes scores and alignments to the terminal or standard output file (on by default, turned off with `-I`).

`-r +n/-m` Specify match/mismatch scores for DNA comparisons. The default is `+5/-4`. `+3/-2` can perform better in some cases.

`-R file` Save a results summary line for every sequence in the sequence library. The summary line includes the sequence identifier, superfamily number (if available) position in the library, and the similarity scores calculated. This option can be used to evaluate the sensitivity and selectivity of different search strategies [14, 16].

`-s file` Specify the scoring matrix file. `fasta36` uses the same scoring matrice format as Blast. Several scoring matrix files are included in the standard distribution in the `data/` directory. For protein sequences: `codaa.mat` - based on minimum mutation matrix; `idnaa.mat` - identity matrix; `pam250.mat` - the PAM250 matrix; [4], (`-s P250`), and `pam120.mat` - a PAM120 matrix (`-s P120`). The default scoring matrix is BLOSUM50 (`-s BL50`). Other matrices include a series of modern PAM-based matrices [8]: MDM40/`-s MD40`, MDM20/`-s MD20`, and MDM10/`-s MD10`, and a selection from the BLOSUM series [6] BLOSUM50, 62, and 80/`-s BL50`, `-s BL62`, `-s BL80`. `-s BP62` sets the scoring matrix to BLOSUM62 and the gap penalties to -11/-1, identical to `BLASTP`. In addition, the VTML160 matrix (`-s VT160`) [13] and OPTIMA_5 (`-s OPT5`) [9] are available.

If the scoring matrix is prefaced by a question mark, e.g. `?BP62`, then the scoring matrix is adjusted for each query to ensure that a 100% identical match can produce a score of at least 40 bits. This is designed for `fastx36` searches with potentially short DNA queries; A 120 nt DNA query can only produce a 40 amino-acid alignment, which, with BLOSUM62 -11/-1, cannot produce more than 23 bits of score. A scoring matrix with a higher information content is required; in the set available by default, MD40, with 2.22 bits/position, would be used. For more information about alignment length and information content, see [1].

`-S` Filter out lower-case characters in the query or library sequences for the initial score calculation (used to filter low-complexity – `seg`-ed – residues). The `pseg` program [23] can be used to lower-case mask low complexity regions in protein sequences. With the `-S` option, lower case characters in the query or database sequences are treated as `X`'s during the initial scan, but are treated as normal residues during the final alignment display. Since statistical significance is calculated from the similarity score calculated during the library search, the lower case residues do not contribute to the score. However, if a significant alignment contains low complexity regions, the residues are shown (as lower case characters, Fig. 2).

The `pseg` program can be used to produce databases (or query sequences) with lower case residues indicating low complexity regions using the command:

```
pseg ./swissprot.fasta -z 1 -q > swissprot.lseg
```

The `-S` option should always be used with `FASTX/Y` and `TFASTX/Y` because out-of-frame translations often generate low-complexity protein sequences. However, only lower case characters in the protein sequence (or protein database) are masked; lower case DNA sequences are translated into upper case protein sequences, and not treated as low complexity by the translated alignment programs. (There is an option in the `Makefile`, `-DDNALIB_LC`, to enable preserving case in DNA sequences.)

`-t #` Translation table - fastx36, tfastx36, fasty36, and tfasty3 now support the BLAST translation tables. See `http://www.ncbi.nih.gov/Taxonomy/Utils/wprintgc.cgi`.

`-t 1` also enables translation of `'TGA'` to `'U'` (seleno-cysteine) (by default, `'TGA'` is translated to `'*'`). Because of the ambiguity of the `'TGA'` codon, translated alignments of `'TGA'` with `-t 1` match `'U'` and `'*'` (termination) equally well.

`-t t` enables the addition of an implicit termination codon to a protein:translated DNA match. That is, each protein sequence implicitly ends with $*$, which matches the termination codes for the appropriate genetic code. To change the translation table and insert a termination character after each protein sequence, use `-t 1 -t t`.

`-T #` set number of threads/workers. Normally on a multi-core machine, the maximum number of processors/cores is used.

`-U` Treat the query sequence an RNA sequence. In addition to selecting a DNA/RNA alphabet, this option causes changes to the scoring matrix so that `G:A` , `T:C` or `U:C` are scored as `G:G -3`.

`-v #` Do window shuffles with the window size specified.

`-V str` Specify annotation characters that can be included (and will be ignored), in the query sequence file, but are displayed in the alignments. If a query file contains `"ACVS*ITRLFT?"`, where `"*"` and `"?"` are used to indicate phosphorylation, giving the option `-V '*?'`, the annotated characters in the query will (`S*`, `F?`) will be highlighted in the alignment (on the number line). A `fasts36` alignment of `seq/ngts.aa` compared to `seq/mgstm1.aa` with `-V '*?'` produces:

```
             *                10??
GT8.7      ILGYWN------------EYTDSSYDEKR---------------------------
           ::::::            :::::::::::
GT8.7   MPMILGYWNVRGLTHPIRMLLEYTDSSYDEKRYTMGDAPDFDRSQWLNEKFKLGLDFPNL
                10        20        30        40        50        60
```

In addition to showing the alignments of post-translationally modified sites, the `-V` option can be used to highlight active sites in library sequences. In the `-m 9c` output, the state of the annotated sites is summarized when `-V` is used.

(fasta-36.3.6 June 2012) The `-V` option has been extended to: (1) allow feature descriptions to be specified in a file, e.g. `-V =annot.defs` where `annot.defs` contains:

```
*:phosphorylation
@:active site
^:binding site
```

The annotation character is left of the ':', the definition is on the right. The `annot.defs` file can also be specified by setting the `FA_ANNOT_DEF` environment variable to the file name;

(2) to include optional annotation file, e.g. `-V '<features.annot'`, or script, e.g. `-V '!features.pl'` for library annotations and `-V 'q!features.pl'` for query annotations. (Some shells require `\!features.pl`.) Similar to the library expansion script, the `features.pl` script is run against a temporary file containing the list of high scoring sequence accessions (the text before the first space), e.g.

```
gi|121735|sp|P09488.3|GSTM1_HUMAN
gi|1170096|sp|Q03013.3|GSTM4_HUMAN
gi|67461004|sp|Q5R8E8.3|GSTM2_PONAB
...
```

The `features.pl` script then produces a file of annotations on those sequences, in the format:

```
>accession1
position label value
>accession2
...
```

For example:

```
>gi|121735|sp|P09488.3|GSTM1_HUMAN
23 *
33 *
34 *
116 ^
173 V N
210 V T
>gi|1170096|sp|Q03013.3|GSTM4_HUMAN
2 V P
116 ^
160 V V
208 V V
209 V F
211 V K
212 V M
>gi|67461004|sp|Q5R8E8.3|GSTM2_PONAB
...
```

The same format is used for the `-V '<feature.annot'` file.

The `V` label is special; it indicates that the feature is a variant residue and specifies the alternative residue in the label field. Thus, `GSTM4_HUMAN` can have a `M` at position 2. Unlike modification or active site annotations, variant residues can change the sequence of the library sequence if replacing the canonical library residue with the variant residue improves the score. Thus, without the `-V '!feature.pl'` script, the human `GSTM1B` variant with dbSNP:rs449856 would align to `GSTM1_HUMAN` (`P04988`) like this (the `-m 1` format option was used to highlight differences) :

```
The best scores are:                                                      opt bits E(1)
sp|P09488.3|GSTM1_HUMAN Glutathione S-transferase Mu 1; GST HB subuni  ( 218) 1490 335.9 3.7e-97

>>sp|P09488.3|GSTM1_HUMAN Glutathione S-transferase Mu 1; GST HB s           (218 aa)
 initn: 1490 init1: 1490 opt: 1490  Z-score: 1776.8  bits: 335.9 E(1): 3.7e-97
Smith-Waterman score: 1490; 99.1% identity (100.0% similar) in 218 aa overlap (1-218:1-218)

...
             170       180       190       200       210
gtm1_h YDVLDLHRIFEPNCLDAFPNLKDFISRFEGLEKISAYMKSSRFLPRPVFTKMAVWGNK
                   x                                    x
sp|P09 YDVLDLHRIFEPKCLDAFPNLKDFISRFEGLEKISAYMKSSRFLPRPVFSKMAVWGNK
             170       180       190       200       210
```

With a `-V '!feature.pl'` script to annotate the variants, the alignment becomes:

```
The best scores are:                                                      opt bits E(1)
sp|P09488.3|GSTM1_HUMAN Glutathione S-transferase Mu 1; GST HB s       ( 218) 1500 338.1   8e-98
```

```
>>sp|P09488.3|GSTM1_HUMAN Glutathione S-transferase Mu 1; GST HB s          (218 aa)
 Variant: K173N;S210T;
 initn: 1500 init1: 1500 opt: 1500  Z-score: 1788.7  bits: 338.1 E(1): 8e-98
Smith-Waterman score: 1500; 100.0% identity (100.0% similar) in 218 aa overlap (1-218:1-218)
...
            170       180       190       200       210
gtm1_h YDVLDLHRIFEPNCLDAFPNLKDFISRFEGLEKISAYMKSSRFLPRPVFTKMAVWGNK

sp|P09 YDVLDLHRIFEPNCLDAFPNLKDFISRFEGLEKISAYMKSSRFLPRPVFTKMAVWGNK
            170  V    180       190       200       21V
```

In addition to removing the two differences as residues 173 and 210, which produces a 100% identical alignment, inclusion of the variant library sequence also improves the raw similarity score and $E()$-value. An example script (`misc/up_feats.pl`) that extracts annotations from a mysql database of Uniprot features is provided.

If the annotation script produces lines beginning with '=', then these lines are taken as annotation definitions, similar to the `annot.defs` file described above. Thus:

```
=*:phosphorylation
=@:active site
=^:binding site
>gi|121735|sp|P09488.3|GSTM1_HUMAN
23 *
33 *
34 *
116 ^
173 V N
210 V T
```

will produce the same annotation descriptions as the `annot.defs` file.

Scripts to produce annotations are available in the `scripts/` directory as `scripts/ann_feats*.pl`. Scripts with `www` in the name, e.g. `scripts/ann_feats_up_www2.pl` and `scripts/ann_pfam_www.pl` download annotation information from Uniprot or Pfam web services, respectively. Scripts lacking `www` require require a MySQL database that associates features or domains with sequence identifiers (accessions). With fasta-36.3.8, domain annotations are allows to overlap each other (which often happens in Pfam and UniProt); FASTA 36.3.6 did not support overlapping domains. Scripts that can produce overlapping domain annotations have `_e` in their names, but will produce non-overlapping domain annotations with the `--no-over` option. Thus: `scripts/ann_pfam_www_e.pl --acc sp|P43553|ALR2_YEAST` produces:

```
>sp|P43553|ALR2_YEAST
451  -  683   PF01544 :1
667  -  799   PF01544 :1
```

While `scripts/ann_pfam_www_e.pl --acc --no-over sp|P43553|ALR2_YEAST` produces:

```
>sp|P43553|ALR2_YEAST
451  -  675  PF01544 :1
676  -  799  PF01544 :1
```

`-w #` Display width value (<200). Sets the approximate width of the high-score descriptions and the length of residue alignments. `-w 60` by default.

`-W #` context length (default is 1/2 of line width -w) for alignment, for programs like `fasta36` and `ssearch36`, that provide additional sequence context.

`-X extended_option` A number of rarely used options are now only available as extended options:

- `X1` sort output by `init1` score (for compatibility with FASTP; obsolete).
- `XB` Calculate pecent identity, percent similarity, and alignment using the BLAST model, which excludes gapped residues. This allows very high identity alignments with large gaps to look much closer, but causes the alignment length to drop by the length of the gap.
- `Xb` (Previously `-B`.) Show the z-score, rather than the bit-score in the list of best scores (rarely used, provided for backward compatibility).
- `XI` Modify rounding used in percent identity/percent similarity display to ensure that sequences that have a mismatch are not shown as 100.0% identical. Without this option, a single mismatch in a 10,000 residue alignment would be shown as 100.0% identical; with this option, it would be shown as 99.9% identical.
- `Xo` (`fasta36`, `[t]fast[x/y]36` only) (Previously `-o`.) Turn off the default `opt` score calculation and sort results by `initn` scores (reduces sensitivity and statistical accuracy, obsolete).
- `XM` The maximum amount of memory available for storing the library in multi-sequence searches. The value is specified in MBytes (`-XL16`) or GBytes (`-XL4G`) and can also be set using the `LIB_MEMK` environment variable (`LIB_MEMK=4G`). Negative values remove the memory restriction. By default (set as a compile-time option, `-DMAX_MEMK=2`), set to 2 GBytes in 32-bit environments and 12 GBytes in 64-bit environments.
- `XN/XX` Alter the treatment of N:N (DNA) or X:X (protein) alignments for counts of identities and similarities. By default the `FASTA` programs count N:N or X:X as identical, but not similar, because their alignment scores are typically negative. `-XNS`, `-XN+`, `-XXS`, and `-XX+` treat N:N and X:X alignments as “similar” , even though their alignment scores are negative, when calculating percent similarity. `-XND`, `-XN-`, `-XXD`, and `-XX-` treat N:N and X:X alignments as non-identical for calculating percent identity.
- `Xx` (Previously `-x`) Specify the penalty for a match to an `X`, and mismatch to `X`, independently of the PAM matrix. Particularly useful for `fastx3/fasty36`, where termination codons are encoded as `X`. For example, `-Xx=0,-1` scores an `X:X` match as 0, and `X:not-X` as -1.
- `Xy` (Previously `-y`.) Set the width of the band used for calculating ”optimized” scores. For proteins and ktup=2, the width is 16. For proteins with ktup=1, the width is 32 by default. For DNA the width is 16.

`-z -1,0,1,2,3,4,5,6`
`-z -1` turns off statistical calculations. `z 0` estimates the significance of the match from the mean and standard deviation of the library scores, without correcting for library sequence length. `-z 1` (the default) uses a weighted regression of average score vs library sequence length; `-z 2` uses maximum likelihood estimates of λ and K; `-z 3` uses Altschul-Gish parameters [2]; `-z 4 - 5` uses two variations on the `-z 1` strategy. `-z 1` and `-z 2` are the best methods, in general.

`-z 11,12,14,15,16`
: estimate the statistical parameters from shuffled copies of each library sequence. This allows accurate statistics to be estimated for libraries comprised of a single protein family.

`-z 21,22,24,25,26`
: estimate the statistical parameters from shuffled copies of the highest scoring sequences reported in the search. library sequence. This shuffling strategy is much more like `prss`, since the sequences shuffled share compositional similarity to the query.

`-Z db_size` sets the apparent size of the database to be used when calculating expectation E()-values. If you searched a database with 1,000 sequences, but would like to have the E()-values calculated in the context of a 100,000 sequence database, use `-Z 100000`.

`-3` translate only three forward frames or search with only the forward strand (complement of `-i`).

Thus, to tell `fasta36` to align `seq1.aa` with `seq2.aa` showing the entirety of both sequences, with 80 characters per line, one would type:

```
fasta36 -w 80 -s BP62 -a seq1.aa seq2.aa
```

The `-w 80` and `-a` options must precede the file names. If you just enter the options on the command line followed by `-I`, the program will prompt for the file names.

In addition, the FASTA programs can accept query sequence data from `STDIN`. To specify that stdin be used as the query or library file, the file name should be specified as `@`. Thus:

```
    cat query.aa | fasta36 @:25-75 /slib/swissprot
```

would take residues 25-75 from `query.aa` and search the `/slib/swissprot`.

2.5.2 Environment variables

FASTA allows virtually every option to be set on the command line (except the *ktup*, which must be set as the third command line argument), but it is often convenient to set the `FASTLIBS` environment variable to specify the location of the `fastlibs` database description file.

`FASTLIBS` – `FASTLIBS` specifies the location of the file that contains the list of library descriptions, locations, and library types (see section on finding library files).

`LIB_MEMK` – Set the maximum amount of memory (MBytes) to be available for library buffering (equivalent to `-XM#`, see above). By default, `2GB` is available on 32-bit systems (`LIB_MEMK=2G`); `8GB` on 64-bit systems.

`REF_URL`, `SRCH_URL` and `SRCH_URL1` – These environment variables are used in HTML mode (`-m 6`) to provide links from the sequence alignment (see the links at `http://fasta.bioch.virginia.edu/fasta_www2/`). `REF_URL` is associated with the `Entrez Lookup` link; `SRCH_URL` with the `General re-search` link, and `SRCH_URL1` with the `Pairwise alignment` link. In each case, the text corresponds to a HTML URL, but with positions containing the `%s` or `%ld` (for numbers) part of a 'C' `sprintf()` call for specific variables. `REF_URL` uses the database (`protein` or `nucleotide`), together with a query term (typically the `gi` number). `SRCH_URL` and `SRCH_URL1` use `db`, `query` (`gi`, `pgm` (`fa`, `ss`, `fx`, etc.), and `start`, `stop`, and `n1` (library sequence length), where `start` and `stop` are the boundaries of the alignment, for sub-sequence searches. The values of these environment variables are used with `sprintf` to build a new URL that is linked in the output.

TMP_DIR – Location (if defined) of the temporary files used by the -e expand_script.sh option.

In addition, environment variables can be used inside both the fastlibs file and in the @db.nam files of file names. The fasta36/conf/fast_libs_e.www file, included with the distribution, shows an example, as do the descriptions of file of file names files shown below. Whenever a word of the form ${WORD} is found in fastlibs or a file of file names, the ${WORD} environment variable is expanded and inserted in the string. Thus, if <${SLIB}/blast_dbs/ describes where a list of files will be found and ${SLIB} is "/seqdata", then the resulting substitution yields: </seqdata/blast_dbs/.

3 Installing FASTA and the sequence databases

3.1 Obtaining/preparing the sequence libraries

The FASTA program package does not include any protein or DNA sequence libraries. Protein and DNA sequence databases are available via anonymous FTP from the NCBI (ftp://ftp.ncbi.nih.gov/blast/db, ftp://ftp.ncbi.nih.gov/blast/db), UniProt (ftp://ftp.uniprot.org/pub/databases/uniprot), and the EBI (ftp.ebi.ac.uk/pub/databases).

Protein Sequence Databases – Protein sequence databases are available from the NCBI, UniProt, and the EBI. The NCBI provides a "raw" database, nr, and a well-curated, less redundant database, refseq_protein, and a copy of the very well annotated swissprot database. Protein sequence databases can also be downloaded from UniProt and the EBI; both sites provide the same UniProt [3] database.

Protein libraries, particularly those used for translated-DNA:protein comparisons with fastx36 or fasty36, show be scanned to remove low-complexity regions. Matches between low complexity regions can violate the composition assumptions used by the FASTA statistical estimates. The pseg program ([23], ftp://ftp.ncbi.nih.gov/pub/seg/pseg) can be used to lower-case low complexity regions, which then can be ignored during the initial database search by using the -S option. To lower-case low complexity regions, run the pseg program against the protein sequence database:

```
pseg /seqdata/swissprot.fa -z 1 -q > /seqdata/swissprot.lseg
```

And then you can run most FASTA programs with -S:

```
ssearch36 -S mgstm1.aa /seqdata/swissprot.lseg
```

Fig. 2 shows the effect of including the -S option with lower-cased low-complexity sequences. The opt score (407), which is used to sort the results and calculate statistics, is lower than the Smith-Waterman score (451), even though exactly the same residues are aligned for each score. The opt score excludes residues 19-30, because they were marked as low-complexity by pseg; thus they are shown as lower-case. The Smith-Waterman score includes the contribution from that part of the alignment.

Out-of-frame translated DNA sequences often produce low-complexity regions [18], so it is particularly important to avoid low-complexity alignments when using fastx36 and fasty36

3.2 Searching taxonomic subsets

Because increasing database size reduces search sensitivity (an alignment with an $E()$-value of 0.001 in a search of a 100,000 entry database will have an $E()$-value of 0.1, not significant, if found in a database of 10,000,000 sequences), it is much more effective to search smaller, less redundant databases (you can always search the larger database later). Thus, the `refseq_protein` database from the NCBI is preferred over `nr`; even better are databases that reflect a limited phylogenetic range (e.g. `refseq_human` for vertebrate sequences).

While the NCBI provides organism-specific `refseq` subsets on their FTP site, they can be difficult to find. Alternatively, you can use the NCBI `Entrez` web site to download a list of `gi` numbers specific to a particular organism or taxonomic range. The FASTA programs can search a subset of a large sequence database that is specified by a list of `gi` numbers by using library format 10. For example, given a list of `gi` numbers for the human proteins in `swissprot.lseg`, the file `sp_human.db`, with the content:

```
<${SLIB}/swissprot.lseg 0:2 4|
3121763
51701705
7404340
205831112
74735515
...
```

could be used to search the human subset of `swissprot.lseg`. The `gi` numbers for the Swiss-Prot entries begin with the second line. The first line specifies the location of the file where the sequences containing the `gi` numbers can be found (`${SLIB}/swissprot.lseg`, the `libtype` of that file (`0:fasta`), the character offset to the beginning of the sequence identifier in that file (`2`), the identifier type (`4`), and the character that separates the fields in the FASTA descriptor (`|`). The identifier type can take four formats:

1	ordered accession strings (letters or numbers)
2	ordered numbers (digits only)
3	un-ordered accession strings
4	un-ordered numbers

(Ordered accession strings/numbers are ordered in both the library and the subset file.)

Thus, given the `0:2 4|` specification above, the line:

```
>gi|3121763|sp|O15143.3|ARC1B_HUMAN Actin-related protein 2/3 ...
```

would be parsed, looking for an number starting at column 4 (the first column is numbered 0), and ending with `|`. The order of sequences in the library do not have to correspond to the order in the `sp_human.db` file (un-ordered). Given a the `sp_human.db` file, a file `swissprot.lseg` in the directory specified by the environment variable `${SLIB}`, and a command of the form:

```
fasta36 -S mgstm1.aa 'sp_human.db 10'
```

Would use the `sp_human.db` file to search the subset of `swissprot.lseg` that contained the specified `gi` numbers.

3.3 DNA sequence libraries

Because of the large size of DNA databases, you will probably want to keep DNA databases in only one format. The FASTA3 programs that search DNA databases — `fasta36`, `fastm36`, and `tfastx/y36` — can read DNA databases in Genbank flatfile (not ASN.1), FASTA, and BLAST2.0 (`formatdb`) formats, as well as EMBL format. BLAST2.0 format is preferred for DNA sequence libraries, because the files are considerably more compact than GenBank format. The NCBI does not provide software for converting from Genbank flat files to Blast2.0 DNA databases, but you can use the Blast `formatdb` program to convert ASN.1 formatted Genbank files, which are available from the NCBI `ftp` site.

The NCBI also provides the comprehensive `nt` DNA database, and several EST databases in Blast2.0/`formatdb` format from `ftp://ncbi.nih.gov/blast/db`.

3.4 Finding the library files

All the FASTA programs comparison programs have the command line syntax:

```
fasta36 query.file /seqdata/library
```

However, in addition to simply specifying the location of the database to be searched (`/seqdata/library`), the FASTA programs provide several methods for referring to sequence databases without specifying a specific file. These methods can be used to provide abbreviations for sequence libraries, e.g.:

`fasta36 query.file s` or `fasta36 query.file +sp+`

To use abbreviations like `'s'` or `'+sp+'` to reference a sequence database, a `FASTLIBS` file must be used, see section 3.5.

Large DNA and protein databases are often distributed across several files. For example, the NCBI `nr` protein database is found in 5 files, `nr.00` ... `nr.04`. To search databases in multiple files, the names of the files are specified in a file of filenames, `nr.nam`:

```
<${SLIB}/blast_dbs/
nr.00 12
nr.01 12
nr.02 12
nr.03 12
nr.04 12
```

In this file, the first line `<${SLIB2}/blast_dbs/`, beginning with <, specifies the location and format (Blast2.0 `formatdb`) the data files. Text of the form `${SLIB}` refers to Unix/MacOSX/Windows environment variables; the value of `${SLIB}` is set by a Unix/MacOSX shell environment command. Thus, if the value of `${SLIB}` is `/seqdata`, then the first sequence library file to be read will be `/seqdata/blast_dbs/nr.00`, in format 12 (Blast2.0 `formatdb`).

To refer to the `nr.nam` file as a file of file names, it must be prefixed by a `@` character, e.g.

```
fasta36 query.file @nr.nam
```

Files of file names can contain references to other files of file names:

```
<${SLIB}/fasta_dbs/
@pdb.nam
@swissprot.nam
```

The FASTA file of file names is similar to the NCBI `prot_db.pal` and `dna_db.nal`, files, but unfortunately they are different, and currently FASTA cannot read NCBI `.pal` or `.nal` files that contain a `DBLIST` line. FASTA can read NCBI `.pal` or `.nal` files that do not contain a `DBLIST` line.

FASTA version `fasta-36.3.6` provides an alternative way to generate a database to be searched: the `!script.sh` file. Like the `-e expand_file.sh` script, a shell script or program can be used to produce a database to a temporary file, which is then seached. For example, if the file `cat_db.sh` contains the command `echo /seqdb/swissprot.lseg`, the command:

```
fasta36 query.aa \!@cat_db.sh
```

will cause `cat_db.sh` to produce a temporary file with the line `swissprot.lseg`, which is interpreted as an indirect file of filenames; thus, because of the `@`, the file will be interpreted as an indirect file, and the `swissprot.lseg` file will be searched. Note that on Unix systems, the `'!'` must be preceeded by a `'\'` so that it is not interpreted by the shell, as shown above.

3.5 FASTLIBS

All the search programs in the FASTA3 package can use the environment variable `FASTLIBS` to find the protein and DNA sequence libraries. (Alternatively, you can specify the `FASTLIBS` file with the `-l fastlibs.file` option.) The `FASTLIBS` variable contains the name of a file that has the actual filenames of the libraries. The `fastlibs` file included with the distribution is an example of a file that can be referred to by FASTLIBS. To use the `fastlibs` file, type:

`setenv FASTLIBS /seqdata/info/fastgbs` (csh/tcsh)
or
`export FASTLIBS=/seqdata/info/fastgbs` (bash/ksh)

Then edit the `fastlibs` file to indicate the location of the protein and DNA sequence libraries. If the protein sequence library is kept in the file `/seqdata/aa/swissprot.lseg` and your Genbank DNA sequence library is kept in the directory: `/seqdata/genbank`, then the `fastlibs` file might contain:

```
SwissProt$0P/seqdata/aa/swissprot.lseg 0
UniProt$0+uniprot+@/seqdata/aa/uniprot.nam
GB Primate$1P@/seqdata/genbank/gpri.nam
GB Rodent$1R@/seqdata/genbank/grod.nam
GB Mammal$1M@/seqdata/genbank/gmammal.nam
^   1    ^^^^       4               ^ ^
          23                         (5)
```

The first line of this file says that there is a copy of the SwissProt sequence database (a protein database) that can be selected by typing ”P” on the command line or when the database menu is presented in interactive mode.

Note that there are 4 (or 5) fields in the lines in the `fastlibs` file. The first field describes library and is displayed by FASTA program; it ends with the '$'. The second field (1 character), is a 0 if the library is a protein library and 1 if it is a DNA library. The third field can either be a single character (`P`) or a word surrounded by the + symbol (`+uniprot+`), and can be used to specify the library on the command line or in interactive mode.

The fourth field is the name of the library file. In the example above, the `/seqdata/aa/swissprot.lseg` file contains the entire protein sequence library. Alternatively, `/seqdata/aa/uniprot.nam` is a file of file names, which contains a list of one or more library files. Likewise, the DNA library files are files of file names.

In addition, an optional fifth field can be used to specify the format of the library file. Alternatively, you can specify the library format in a file of file names. This field must be separated from the file name by a space character (' ') from the filename. FASTA can read the libraries in the following formats:

0	FASTA (`>SEQID` - comment/sequence)
1	Uncompressed Genbank (LOCUS/DEFINITION/ORIGIN)
2	NBRF CODATA (ENTRY/SEQUENCE) (obsolete)
3	EMBL/SWISS-PROT (ID/DE/SQ)
4	Intelligenetics (;comment/SEQID/sequence) (obsolete)
5	NBRF/PIR VMS (`>P1;SEQID`/comment/sequence) (obsolete)
6	GCG (version 8.0) Unix Protein and DNA (compressed)
7	FASTQ (sequence only, quality ignored)
9	a script that is executed to produce FASTA format sequences
10	subset format (¡/slib2/swissprot.lseg 0:2 4—)
11	NCBI Blast1.3.2 format (unix only) (obsolete)
12	NCBI Blast2.0 format
16	MySQL (requires special compilation)
17	Postgres (requires special compilation)

Today, the most popular formats are `FASTA`, type `'0'`, the default, and the NCBI Blast2.0 `formatdb` formats (type `'12'`). The FASTA programs cannot read NCBI ASN.1 formatted databases. If a library format is not specified, for example, because you are just comparing two sequences, FASTA (format 0) is used by default. To specify a library type on the command line, add it to the library filename and surround the filename and library type in quotes:

```
fasta36 query.file "/seqdb/genbank/gbmam 12"
```

NCBI `formatdb` databases are built from multiple files, e.g. `gbmam.nsq`, `gbmam.nhr`, `gbmam.nin`; to refer to the complete set of files, simply use name before the suffixes, e.g. `gbmam`. When NCBI databases distributed across several files, e.g. `gbbct.00`, `gbbct.01`, etc, those files must be included in a `gbbct.nam` file of file names.

FASTA subset format (`10`) allows users to search a subset of a sequence database, by specifying a list of `gi` numbers or accessions in a larger database. The format begins with a line naming the file sequence file followed by information about how to extract the `gi` number or accession. Thus, the line.

```
<library_file lib_fmt:id_fmt id_loc
```

where `lib_fmt` is the library format (0), `id_fmt` is the format of the sequence identifier (:1, :2 - ordered strings or numbers; :3, :4 - unordered strings or numbers), and `id_loc` is the location of the sequence identifier. For example,

```
</slib2/blast/swissprot.lseg 0:2 4|
3121763
51701705
7404340
74735515
...
```

specifies the file containing all the sequences and the file is in FASTA format ('0:'), the sequence identifier is a number (':2'), and the identifier starts at character 4 and ends with the `'|'` symbol.

The major problem that most new users of the FASTA package have is in setting up the program to find the databases and their library type. In general, if you cannot get `fasta36` to read a sequence database, there is probably something wrong with the `FASTLIBS` file. A common problem is that the database file is found, but either no sequences are read, or an incorrect number of entries is read. This is almost always because the library format (`libtype`) is incorrect.

Test the setup by running FASTA. Enter the sequence file '`mgstm1.aa`' when the program requests it (this file is included with the programs). The program should then ask you to select a protein sequence library. Alternatively, if you run the `tfastx36 -I` program and use the mgstm1.aa query sequence, the program should show you a selection of DNA sequence libraries. Once the `fastlibs` file has been set up correctly, you can set FASTLIBS=fastgbs in your AUTOEXEC.BAT file, and you will not need to remember where the libraries are kept or how they are named.

4 Frequently Asked Questions (FAQs)

Where can I get FASTA? –

The most current version of the FASTA source code is available from `http://github.com/wrpearson/fasta36`. In addition, you can get the programs from `http://faculty.virginia.edu/wrpearson/fasta`, but sometimes there is a lag between the latest release on GITHUB and the compiled versions at `faculty.virginia.edu`. This document describes `fasta-36.3.8`, which is available from `http://faculty.virginia.edu/wrpearson/fasta/fasta3.tar.gz`. In addition, pre-compiled versions of the programs are available for MacOSX and Windows.

Which program should I use? – See Table I, also:

Query	Library	FASTA pgm.	BLAST pgm.	
Prot.	Prot.	`fasta36`	`blastp`	heuristic local similarity
		`ssearch36`		optimal local sim.
		`ggearch36`		global:global sim.
		`glearch36`		global:local sim.
DNA	DNA	`fasta36`*	`blastn`	
Prot.	Prot.	`lalign36`		multiple non-intersecting
DNA	DNA			alignments
DNA	Prot.	`fastx36`	`blastx`	trans. DNA:protein sim.
		`fasty36`		
Prot.	DNA	`tfastx36`	`blastn`	protein:trans. DNA
		`tfasty36`		
Prot.	Prot.	`fasts36`		Unordered peptides
Prot.	DNA	`tfasts36`		Unordered peptides
DNA	DNA	`fasts36`		Unordered oligonucleotides
Prot.	Prot.	`fastm36`		Ordered peptides
DNA	DNA	`fastm36`		Ordered oligos

*`ssearch36` can also be used for DNA:DNA, but is much slower and no more sensitive.

How do I make FASTA act/look like BLAST? –

```
fasta36 -s BP62 -m BB query.file library.file
```

`-s BP62` sets the same scoring matrix (BLOSUM62) and gap-penalties (-11/-1) as BLAST (FASTA uses BLOSUM50 by default). `-m BB` produces very BLAST-like output.

In addition, the `-m 8` and `-m 8C` options provide BLAST tabular output, optionally with comments (`-m 8C`). This compact output is effective for analysis pipelines. In addtion, `-m 8XC` (no comments) or `-m 8CC` provides two additional blast-tabular fields, a CIGAR alignment string and, if available, an annotation string.

When I search Genbank - the program reports: `0 residues in 0 sequences`? This typically happens because the program does not know that you are searching a Genbank flatfile database and is looking for a FASTA format database. Be certain to specify the library type (”1” for Genbank flatfile) with the database name.

The search seemed to work, but I do not see any results. – In command line mode (the default), all the FASTA programs limit the number of high scoring sequences shown using an expectation value cutoff ($E() < 10$ for proteins; $E() < 2$ for DNA). Sometimes, a search will complete successfully (you see the message `XXXX residues in YYY sequences`) but the message: `!! No sequences with E() < 10` instead of `The best scores are:`. Typically, this happens because of a problem with the statistical estimation process; in particular, if the library contains only related sequences and `-z 11` was not used, none of the hits may be "significant". To trouble shoot this problem, you can search with `-z -1`, which turns off all the statistical estimation procedures, and will show the 20 highest scoring sequences (`-b #` sets the default number of sequences shown).

What is the difference between `fastx3` and `fasty3`? (or `tfastx3` and `tfasty3`)? – `[t]fastx3` uses a simpler codon based model for alignments that does not allow frameshifts in some codon positions (see ref. [25]). `fastx3` is about 30% faster, but `fasty3` can produce higher quality alignments in some cases.

What is ktup? – All of the programs with `fast` in their name use a computer science method called a lookup table to speed the search. For proteins with *ktup*=2, this means that the program does not look at any sequence alignment that does not involve matching two identical residues in both sequences. Likewise with DNA and *ktup* = 6, the initial alignment of the sequences looks for 6 identical adjacent nucleotides in both sequences. Because it is less likely that two identical amino-acids will line up by chance in two unrelated proteins, this speeds up the comparison. But very distantly related sequences may never have two identical residues in a row but will have single aligned identities. In this case, *ktup* = 1 may find alignments that *ktup*=2 misses.

How do I turn off statistics? – The FASTA programs are designed to identify homologs based on statistically significant similarity; to infer homology you need accurate statistical estimates. Sometimes, however, you know the sequences are related, and searching against libraries of related sequences can confuse FASTA if you do not use `-z 11`. If all you want are scores and alignments, use `-z -1` to turn off statistical estimates.

Where are `prss` **and** `prfx`? – Earlier FASTA3 releases included `prss3` and `prfx3`. With FASTA version 35 and 36, these programs have been incorporated into `ssearch36` and `fastx36`. FASTA version 35 and 36 programs now automatically estimate statistical parameters by shuffling - the function of `prss` and `prfx`, when searching for libraries with fewer than 500 members.

Where is `tfasta`? – Although it is possible to make `tfasta36`, it is not compiled by default. `tfastx36` and `tfasty36` allow frame-shifts to be joined into a single alignment; `tfasta` did not. `tfastx36` produces better alignments with better statistics.

Can I run the FASTA programs on a cluster? – With version 36.3.4, almost all of the FASTA programs can be run on clusters of computers using MPI (Message Passaging Interface). The programs can be compiled using `make -f ../make/Makefile.mpi_sse2` from the `fasta36/src` directory. Except for `lalign36`, all the programs in Table I are available as `fasta36_mpi`, `ssearch36_mpi`, etc.

Unfortunately, the current MPI implementation involves substantially more communications overhead than the threaded versions. The FASTA programs are very efficient on threaded machines; if the preload option is used (edit `make/Makefile36m.common` to use `comp_lib8.c`), the FASTA programs can obtain more than 40-fold speedup on a 48-core machine (the largest I have tested).

Sometimes, in the list of best scores, the same sequence is shown twice with exactly the same score. Sometimes, the sequence is there twice, but the scores are slightly different? – When any of the FASTA programs searches a long sequence, it breaks the sequence up into *overlapping* pieces. If the highest scoring alignment is at the end of one piece, it will be scored again at the beginning of the next piece. If the alignment is not be completely included in the overlap region, one of the pieces will give a higher score than the other. These duplications can be detected by looking at the coordinates of the alignment. If either the beginning or end coordinate is identical in two alignments, the alignments are at least partially duplicates.

As always, please inform me of bugs as soon as possible.

William R. Pearson
Department of Biochemistry
Pinn Hall Box 800733
U. of Virginia
Charlottesville, VA
wrp@virginia.EDU

References

[1] S. F. Altschul. Amino acid substitution matrices from an information theoretic perspective. *J. Mol. Biol.*, 219:555–65, 1991.

[2] S. F. Altschul and W. Gish. Local alignment statistics. *Methods Enzymol.*, 266:460–480, 1996.

[3] UniProt Consortium. Ongoing and future developments at the universal protein resource. *Nucleic Acids Res*, 39:D214–D219, 2011.

[4] M. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. In M. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, volume 5, supplement 3, pages 345–352. National Biomedical Research Foundation, Silver Spring, MD, 1978.

[5] M. Farrar. Striped Smith-Waterman speeds database searches six times over other simd implementations. *Bioinformatics*, 23:156–161, 2007.

[6] S. Henikoff and J. G. Henikoff. Amino acid substitutions matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, 89:10915–10919, 1992.

[7] X. Huang, R. C. Hardison, and W. Miller. A space-efficient algorithm for local similarities. *Comp. Appl. Biosci.*, 6:373–381, 1990.

[8] D. T. Jones, W. R. Taylor, and J. M. Thornton. The rapid generation of mutation data matrices from protein sequences. *Comp. Appl. Biosci.*, 8:275–282, 1992.

[9] Maricel G Kann and Richard A Goldstein. Performance evaluation of a new algorithm for the detection of remote homologs with sequence comparison. *Proteins*, 48:367–76, Aug 2002.

[10] A. J. Mackey, T. A. J. Haystead, and W. R. Pearson. Getting more from less: Algorithms for rapid protein identification with multiple short peptide sequences. *Mol. Cell. Proteomics*, 1:139–147, 2002.

[11] L. J. Mills and W. R. Pearson. Adjusting scoring matrices to correct overextended alignments. *Bioinformatics*, 29:3007–3013, 2013.

[12] R. Mott. Maximum-likelihood estimation of the statistical distribution of smith-waterman local sequence similarity scores. *Bull. Math. Biol.*, 54:59–75, 1992.

[13] Tobias Muller, Rainer Spang, and Martin Vingron. Estimating amino acid substitution models: a comparison of dayhoff's estimator, the resolvent approach and a maximum likelihood method. *Mol Biol Evol*, 19:8–13, 2002.

[14] W. R. Pearson. Comparison of methods for searching protein sequence databases. *Prot. Sci.*, 4:1145–1160, 1995.

[15] W. R. Pearson. Effective protein sequence comparison. *Methods Enzymol.*, 266:227–258, 1996.

[16] W. R. Pearson. Empirical statistical estimates for sequence similarity searches. *J. Mol. Biol.*, 276:71–84, 1998.

[17] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85:2444–2448, 1988.

[18] W. R. Pearson, T. C. Wood, Z. Zhang, and W. Miller. Comparison of DNA sequences with protein sequences. *Genomics*, 46:24–36, 1997.

[19] J. T. Reese and W. R. Pearson. Empirical determination of effective gap penalties for sequence comparison. *Bioinformatics*, 18:1500–1507, 2002.

[20] T. Rognes and E. Seeberg. Six-fold speed-up of smith-waterman sequence database searches using parallel processing on common microprocessors. *Bioinformatics*, 16:699–706, 2000.

[21] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

[22] M. S. Waterman and M. Eggert. A new algorithm for best subsequences alignment with application to tRNA-rRNA comparisons. *J. Mol. Biol.*, 197:723–728, 1987.

[23] J. C. Wootton and S. Federhen. Statistics of local complexity in amino acid sequences and sequence databases. *Comput. Chem.*, 17:149–163, 1993.

[24] A. Wozniak. Using video-oriented instructions to speed up sequence comparison. *Comput Appl Biosci*, 13:145–150, 1997.

[25] Z. Zhang, W. R. Pearson, and W. Miller. Aligning a DNA sequence with a protein sequence. *J. Computational Biology*, 4:339–349, 1997.

Appendix

A FASTA Makefile compile time options

The `fasta-36/make` directory includes `Makefile`s appropriate for a broad range of environments, including Linux/Unix, BSD, MacOSX, and Windows. Makefiles are regularly tested against MacOSX, Linux, and Windows. Table 2 summarizes the Makefile options that can be modified.

As distributed, the `Makefiles` in `fasta36/make`, build a version of the FASTA programs that is optimized for single searches against arbitrary sized databases, using bit scores, efficient sampled statistics, and gap-open/extend penalties. The default compilation configuration can be changed either by changing the compile time defines (Table 2) in the main `Makefile`, e.g. `make/Makefile.linux64_sse2`, or by editing `make/Makefile36m.common`.

Table 2: FASTA `Makefile` compile time `#defines`

`#define`	Status*	Target file(s)	Function
`ALLOCNO`	obs	`dropnfa.c`, `dropfx.c`, `dropfz2.c`	allows FASTA algorithm to use memory $\sim$ query length (n0), not query $+$ library (n0+n1).
`DNALIB_LC`	undef	`initfa.c`	enable lower case masking for DNA libraries
`HTML_HEAD`	undef	`comp_lib5e.c`, `comp_lib8.c`	wrap `-m 6` HTML output with `<html> <body> </body> </html>`
`M10_CONS`	def	`c_dispn.c`	show consensus line (`:.  `) with `-m 10` output.
`OLD_FASTA_GAP`	undef	`drop*.c`	use first-residue/additional residue penalties, not open/extend.
`PGM_DOC`	def	`comp_lib5e.c`, `comp_lib8.c`	provide `#pgm_name -opt1 -opt2 query file` copy of command line
`PROGRESS`	def	`comp_lib5e.c`, `comp_lib8.c`	provide progress symbols in interactive mode
`SAMP_STATS`	def	`comp_lib5e.c`, `comp_lib8.c`	scores are sampled for statistical estimates
`SAMP_STATS_LESS`	def	`compacc.c`	a slower sampling strategy is used
`SHOW_ALIGN_SCORE`	undef	`wm_align.c`	print score, cummulative score, during alignment (for teaching)
`SHOW_HELP`	def	`comp_lib5e.c`, `comp_lib8.c`, `initfa.c`, `doinit.c`	print out help information with '-help', or no arguments given. Undef `SHOW_HELP` reverts to pre-`fasta-35.4.4`.
`SHOW_HIST`	undef	`doinit.c`	inverts current meaning of `-H` (shows by default for non-PCOMPLIB (MPI) programs).
`SHOWSIM`	def	`mshowbest.c` `mshowalign2.c`	display percent similarity
`USE_LNSTATS`	obs	`scaleswn.c`	use $ln()$-scaling for scores, removed in `fasta2.0`.

*Status: def: #defined in standard `Makefiles`; undef: undefined; obs: obsolete, provided backwards compatibility with FASTA2.0 or earlier.

High-performance searches with many queries – By default, the `comp_lib5e.c` program specified in `Makefile36m.common` builds FASTA programs that re-read the library sequence database for every query sequence. This has the advantage that sequence comparison begins almost immediately, but if thousands of searches are being performed, the database is re-read thousands of times. `Makefile36m.common` can be edited to use `comp_lib8.c` in place of `comp_lib5e.c` and the database is read only once, then held in memory for additional searches. Of course, if `comp_lib8.c` is used, the computer must have enough memory to store the complete database. Keeping the database in memory allows the FASTA programs to very efficiently used large, multi-core computers.

Parallel searches with MPI – By default under Unix/Linux/MacOSX, the FASTA programs are threaded; they will spawn as many threads as CPU cores are available (this can be limited with the `-t n-threads` option). Using `comp_lib8.c`, we see almost 48-fold speedup on a 48-core machine. The FASTA programs can also be run in parallel in the MPI environment on clusters of computers. To build the MPI versions of the programs, use `make ../make/Makefile.mpi_sse2 ssearch36_mpi`, `fastx36_mpi`, etc. The MPI programs currently substantially more communications overhead than the threaded versions, so they may not scale as well to large clusters.

FASTA version history

FASTA v33, Oct, 1999 – Dec, 2000	
Oct 1999	Add support for NCBI Blast2.0 formatted libraries, and memory mapped databases. FASTA now reads both BLAST1.4 and BLAST2.0 formatted databases. (version 3.2t08) Include Maximum Likelihood Estimates for Lambda and K (-z 2) Include a new strategy for searching with low complexity regions. The pseg program can produce libraries with low complexity regions as lower case characters, which can be ignored during the initial FASTA/SSEARCH scan, but are considered when producing the final alignments. (3.3t01) Change output to report bit scores, which are also used by BLAST.
Mar 2000	Another new statistics option, -z 6, uses Mott's approach [12] for calculating a composition dependent Lambda for each sequence. (3.3t05)
Dec 2000	Automatically change the gap penalties when alternate (known) scoring matrices are used using Reese and Pearson gap penalties [19]. First implementation to read from MySQL databases.
May 2001	change all FASTA gap penalties from first-residue, additional residue to the gap-open, gap-extend values used by BLAST.

FASTA v34, Jan, 2001 – Jan, 2007	
Jun 2002	Modify statistical estimation strategy to sample all the sequences in the database, not just the first 60,000. (3.4t11)
Jan 2003	Implementation of vector-accelerated (Altivec) code for Smith-Waterman (SSEARCH) and banded Smith-Waterman (FASTA) using the Rognes and Seebug [20] algorithm. This code was removed in Sept, 2003, because of possible conflict with a patent application, but was restored using a different algorithm in Nov. 2004.
Jun 2003	Provide PSI-SEARCH — an implementation of SSEARCH that can search with PSI-BLAST PSSM profile files. PSI-SEARCH estimates statistical significance from the distribution of actual alignment scores; thus the estimates are much more reliable than PSI-BLAST estimates. Also, change the similarity display to work with profiles. (3.4t22)
July 2003	Provide ASN.1 definition line parsing for BLAST formatdb v.4 libraries. Restructure the programs to use a table-driven approach to parameter setting. Two tables now define the algorithm, query sequence type, library type, scoring matrix, and gap penalties for all programs.
Sept 2003	A new option -V for annotating alignments provided. Designed for highlighting post-translational modifications with fasts, it can also be used to highlight active sites and other conserved residues. (3.4t23)
Dec 2003	Addition of -U option for RNA sequence comparison. G:A matches score like G:G matches to account for G:U basepairs. Change default *ktup* for short query sequences. Increase band-width for DNA banded final alignments.

FASTA version history (cont.)

July 2004	Allow searching of `Postgres`, as well as `MySQL` database queries.
Nov 2004	(`fa34t24`) Incorporation of Erik Lindahl "anti-diagonal" Altivec implementation of [24] for Smith-Waterman only. Altivec `ssearch34` is now faster than `fasta34` for query sequences $<$ 250 amino acids.
Jan 2005	Change `FASTS` to accommodate very large numbers of peptides ($>$100) for full coverage on long proteins
Jun. 2006	(`fa34t26`) Incorporation of Smith-Waterman algorithm for the SSE2 vector instructions written by Michael Farrar [5]. The SSE code speeds up Smith-Waterman 8 – 16-fold.

`FASTA` v35, March, 2007 – March, 2010

Mar. 2007	fasta v35 – Accurate shuffle-based $E()$-values for all searches and alignments; statistics from searches against small libraries are supplemented with shuffled alignments. More efficient threading strategies on multi-core computers, for 12X speedup on 16-core machines. Inclusion of `lalign` (`SIM`) local domain alignments. `lalign` alignments now have accurate shuffle-based $E()$-values.
Apr. 2007	Introduction of `ggsearch`, for global alignment searches, and `glsearch`, for searches with scores that are global in the query and local in the library. `ggsearch` and `glsearch` calculate $E()$-values using the normal distribution. Both programs can search with `PSI-BLAST` PSSMs.
Dec. 2007	Efficient strategy for searching subsets of databases (lists of GI or accession numbers)
Feb. 2008	Annotations in either query or library sequences can be highlighted in the alignment, and the state of annotated residues is compactly summarized with `-m 9c`.
Oct. 2008	Modification `lsim4.c` (`lalign35`) provided by Xiaoqui Huang to ensure that self-alignments do not cross the identity diagonal.

`FASTA` v36, March, 2010 –

Mar. 2010	`FASTA` v36 displays all significant alignments between query and library sequence. BLAST has always displayed multiple high-scoring alignments (HSPs) between the query and library sequence; previous versions of the FASTA programs displayed only the best alignment, even when other high-scoring alignments were present. New statistical options, `-z 21, 22, 26`, provide a second $E2()$-value estimate based on shuffles of the highest scoring sequences.

FASTA version history (cont.)

	Improved performance using statistics-based thresholds for gap-joining and band-optimization in the heuristic FASTA local alignment programs, increasing speed 2 - 3X.
	Greater flexibility in specifying combinations of library files and subsets of libraries. `FASTA` v36 programs can include indirect files of library names inside of indirect files of library names.
	`FASTA` 36 programs are fully threaded, both for searches, and for alignments. The programs routinely run 12 - 15X faster on 8-core machines with ”hyper-threading” (effectively 16 cores).
	`-z 21` .. `26` E2() statistical estimates from shuffled best scores.
Sep. 2010	`-m 8`, `-m 8C` BLAST tabular output.
Nov, 2010	Variable scoring matrices (`-m ?BP62`).
Dec, 2010	(`fasta-36.3.1`) SSE2 vectorized `ggsearch36`, `glsearch36` (Michael Farrar).
Jan, 2011	(`fasta-36.3.2`) MPI versions implemented and tested.
Feb, 2011	Introduce `-m B`, `-m BB` BLAST-like output.
Mar, 2011	(`fasta-36.3.4`) Program is no longer interactive by default. `fasta36 -h` and `fasta36 -help` provide common/complete options, with many defaults. `doc/fasta_guide.pdf` available.
May, 2011	(`fasta-36.3.5`) Introduce (1) `-e expand.sh` scripts to extend the effective size of the database searched, based on significant hits; (2) `-m "F# output.file"` to send different output formats to different files; and (3) `-X` expanded options, `-o` replaces the old `-X` and `-Xo` replaces `-o`.
Jan, 2012	Include `.fastq` files as library type 7
May, 2012	allow reverse-complement alignments with `ggsearch` and `glsearch`
Jun, 2012	Introduce `-V !script.pl` driven alignments, and variant scoring.
Aug, 2012	Introduce `-V !ann_feats.pl` sub-alignment (region-based) scoring.
Apr, 2013	Extend `ENV` options to introduce a domain-plotting option for FASTA web sites.
Nov, 2014	(`fasta-36.3.7`) Allow overlapping domains in annotation scripts.
Nov, 2015	(`fasta-36.3.8`) Improvements in overlapping domain code. Introduction of `scripts/annot_blast_btop.pl` to provide annotations and subalignment scoring to `blast` alignments. Provide annotations in `-m 8CB` BLAST tabular output.
May, 2016	Implement `psisearch2_msa.pl` and `psisearch2_msa.py`.
Feb, 2018	Introduce `-X B`, which causes the `FASTA` programs ignore gaps to calculate `BLASTP` percent identities.