WU BLAST 2.0 TOPICS

Index

Description

Washington University BLAST (**WU BLAST**) version 2.0 is a powerful software package for gene and protein identification, using sensitive, selective and rapid similarity searches of protein and nucleotide sequence databases. The feature list for WU BLAST 2.0 is long and continues to expand. Much of this is outlined below. A complete suite of search programs (**blastp**, **blastn**, **blastx**, **tblastn** and **tblastx**) is included in the package, along with database management and support programs that include **nrdb**, **patdb**, **xdformat**, **xdget**, **seg**, **dust** and **xnu**.

WU BLAST has been built to be the most trusted database search tool in your software toolbox, doing what you tell it, doing precisely what it says it's doing, and able to handle even your biggest jobs with aplomb. WU BLAST was built from the start to offer superior performance and flexibility. Its unique combination of features, sensitivity, speed and reliability is achieved by using advanced algorithms, through painstaking software coding, the use of extensive error checks, and through a superior design that anticipates future needs. To help users keep pace with the latest technology, with every new release of WU BLAST a high degree of backward compatibility has been provided.

WU BLAST is neither a re-hashed nor “Mac-ified” version of NCBI BLAST, although WU BLAST is in many ways easier to use. WU BLAST shares essentially no code with NCBI BLAST, except for some portions that both packages copied from the public domain ungapped BLAST 1.4 (W. Gish, unpublished). For more information about the lineage and history of WU BLAST development, please go here.

Licensing

Information on licensing of WU BLAST 2.0 can be found here.

Key Features

Some key features of WU BLAST 2.0 are described below.

- Gapped alignment routines are available (and used by default) in *all* BLAST search modes: BLASTP, BLASTN, and TBLASTN (Altschul *et al.*, 1990), as well as, BLASTX (Gish and States, 1993) and TBLASTX (W. Gish, 1994, unpublished). Gaps can optionally be turned off in any mode if desired.
- Potentially *multiple* regions of similarity are identified and reported for each database sequence, thus yielding increased sensitivity and selectivity. This feature is essential for finding: all exons in a multi-exon gene sequence, not just the longest or best-matching exon; all complete or partial copies of a repetitive element in a genomic sequence, not just the best matching one; and multiple, discrete domains of similarity between sequences, not just the highest-scoring one.
- Karlin and Altschul (1993) "Sum" statistics are available (and used by default) in *all* search modes, to evaluate the joint probability of multiple regions of similarity, as described by Altschul and Gish (1996). By this technique, sets of similar regions are often found to be statistically significant that individually would be insignificant and might otherwise go unreported. The combination of well-chosen heuristics and statistics in WU BLAST is often more sensitive/selective than: the full dynamic programming approach of Smith and Waterman (1981), that finds and evaluates the significance of only the highest scoring alignment with each database sequence; and other approaches or BLAST implementations that identify multiple regions of local similarity which are then evaluated individually for statistical significance.
- Poisson statistics are available as an option to Karlin-Altschul Sum statistics in all search modes. Simpler Karlin-Altschul (1990) statistics, that do not involve joint probability calculations, are also available as an option.
- Using the postsw option, a full Smith-Waterman alignment is performed on query-subject pairs of sequences that are to be reported by BLASTP. Smith-Waterman results are combined with the initial BLAST results and redundancy is removed. This may alter the relative ranking of database matches before output. Use of this option is recommended, although it may be supplanted in the future by other option(s) or by a redefined default behavior.
- The execution of WU BLAST 2.0 has been optimized such that gapped searches typically run *faster* — sometimes many times faster — than the ungapped version 1.4 programs ever did, while increasing sensitivity. An exception to this observation is BLASTN (already quite fast) which typically runs about 10% slower with its default parameters, due to the addition of gapped alignment steps.
- The classical 1-hit BLAST algorithm has not been changed in WU BLAST 2.0 and remains the default method for finding ungapped alignments that are then used as "seeds" for finding gapped alignments. WU BLAST 2.0 thus retains sensitivity and control characteristics to which users became accustomed with previous versions of BLAST; the addition of gapped alignments in version 2.0 merely improves sensitivity. When assessed at the same sensitivity level, the optimized, classical BLAST algorithm implemented in WU BLAST 2.0 exhibited nearly the same speed as the 2-hit algorithm implemented at the NCBI (Altschul *et al.*, 1997) and uses significantly less memory. For users who desire still higher speed, an improved 2-hit algorithm is available in a higher-performance (more efficient and more sensitive) WU implementation than that from the NCBI. (See the hitdist option); Users should note that even in the WU version of the 2-hit algorithm, significantly more memory is required to achieve the same level of sensitivity as the classical 1-hit algorithm, so even though the 2-hit algorithm may be faster when its parameters are tuned to achieve the same level of sensitivity as the 1-hit algorithm, its increased memory

requirements may be prohibitive). While the classical 1-hit BLAST algorithm remains the default in all WU BLAST search modes, the 2-hit algorithm is available as an option in all search modes, as well — including BLASTN.
- Gapped alignments in the **blastn** search mode are evaluated correctly — as is indeed the case for all WU BLAST search modes — using different statistical parameter values (λ, *K* and *H*) than those used to evaluate the significance of ungapped alignments. If appropriate parameters are unavailable for the particular combination of scoring matrix and gap penalties being used, a prominent WARNING is displayed.
- Unique to WU **blastn** is support for fully-specified scoring matrices, not just simple match/mismatch scoring systems. This allows (for example) transitions to be scored differently than transversions; and permits positive G-A substitution scores for the design of siRNAs where G-U base pairing is allowed. Scoring matrices may also be tailored to improve the design of PCR primers. Scoring matrices were first supported in 1994, by the NCBI's ungapped **blastn** version 1.4 (Gish, W., unpublished; see http://blast.wustl.edu/blast-1.4). Support for nucleotide scoring matrices was dropped by the NCBI's **blastall** 2.0 program first released in 1997, but has been maintained continuously in all WU versions of the software since the migration to Washington University in 1994.
- Word lengths (re: the *W* parameter) as short as 1 have been supported continuously by WU **blastn**, as are nucleotide neighborhood words, using the neighborhood word score threshold parameter, *T*. Using neighborhood words, nucleotide sequence similarity can be detected even in the absence of any identical residues between two sequences. Users are cautioned, however, that careless use of the *T* parameter can result in vast and overwhelming amounts of memory being requested by the software; *T* should likely be used only in conjunction with very short word lengths.
- Information describing "consistent" groups of alignments (HSPs) is provided by BLAST 2.0, when the topcomboN or links options are used. This facility can help with construction of distinct gene structure(s) from a barrage of alignments.
- Multiple output formats are available — and can in fact be produced simultaneously from a single program run — including an informative tabular output and XML output that conforms to the NCBI DTD.
- The eXtended Database Format (**XDF**) provides the foundation for easier, more flexible and more accurate management of protein and nucleotide sequence databases. XDF was developed in part because the NCBI BLAST 2.0 database format could not accurately represent sequences over 16 Mbp in length, at a time when human genome contigs exceeded 25 Mbp (Hattori *et al.*, 2000) With GenBank release 149 in August 2005, the sequences total over 50 Gbp in length. Since its inception in 1999, **XDF** has allowed for accurate storage and indexed retrieval of individual sequences up to 1 Gbp (billion base-pairs). Other BLAST software limits database files to 2 gigabytes, whereas WU BLAST's XDF supports databases (and database files) of virtually unlimited size.

In support of the eXtended Database Format, a new database formatting tool named **xdformat** was introduced in the WU BLAST 2.0 package in 1999. The distinct features and advantages to using XDF and the **xdformat** program include:

- fast appends of new sequence data to existing databases — no need to reformat a database just to add one or more new sequences to it;
- **xdformat** formats databases up to 4 times faster than the NCBI formatdb program, while offering a superset of features, greater reliability, and more extensive error checking by default;
- safe roll-backs of database updates when parse errors or file I/O errors arise (*e.g.*, disk full errors), leaving your database intact and ready for search and retrieval;
- regardless of the input data size, 3 and only 3 database files are *always* created by **xdformat** (or just 4 files when indexing identifiers), which simplifies the implementation and testing of robust management scripts;
- in command line shell scripts and PERL system() calls, the names of XDF database files all match

the simple regular expression `*.xp?` for peptide sequence databases and `*.xn?` for nucleotide sequence databases (or `*.x[np]?` to encompass both peptide and nucleotide databases);
- flexible indexing of *all* sequence identifiers is performed, including user-defined identifiers and identifiers defined by the NCBI;
- identifier indexes support duplicate occurrences of the same identifier — even identical "gi" identifiers that have been known to cause some indexing programs to abort — with warnings issued when duplicates do appear;
- identifier indexing is supported not only when first creating an XDF database but when appending new sequences to an existing XDF database;
- if a database was originally created without an identifier index, one can be quickly added later without having to reformat the entire database;
- identifier indexes can be quickly re-built if necessary, using different indexing policies, without having to reformat the entire database;
- intelligent retrieval of indexed sequences using a complementary program named **xdget**; **xdget** can retrieve sequences by identifier, even if not told what "name space" the identifer came from (*e.g.*, gi, accession, locus, user-defined, etc.); for more on identifier indexing, see this;
- **xdformat** and **xdget** both accept and work intelligently with identifiers that obey the International DDBJ/EBI/NCBI collaboration's Accession**.**Version identifier syntax (*e.g.*, the programs know that BAA84643.2 is a newer version of BAA84643, but will retrieve BAA84643.1 if specifically requested); and the parsing programs that come with WU BLAST for converting GenBank and EMBL database flat files into "FASTA" format report *gi* identifiers, as well as Accessions ***with Versions***;
- both the memory required and the search initiation time are reduced for databases containing large numbers of entries, which is particularly important when memory is in short supply or when multiple processors are standing by waiting for a single-threaded initialization phase to be completed;
- the ability to dump (or recover) the contents of an XDF database back into FASTA format with the original annotation and ambiguity codes intact;
- both the **X** and **N** ambiguity codes are supported in nucleotide sequences, thus permitting the use of alternative substitution scores for these letters and the use of PHRED/PHRAP sequence output "as is" as input to **xdformat**.
- Support for XDF by the BLAST search programs did not come at the expense of losing backward compatibility. WU BLAST can search databases in either XDF or the classical BLAST 1.4 database formats. For early users of NCBI or WU BLAST 1.4, migration to XDF was merely a drop-in upgrade of the version 2.0 WU software.
- When searching very large databases, virtual memory requirements are dramatically reduced in WU BLAST 2.0, eliminating program failures that occurred when system resource limits were unexpectedly reached.
- Virtual databases are supported in BLAST 2.0. Virtual databases can be specified on the command line as a white space-delimited list of component database names. Virtual databases can be comprised of components in either XDF *or* classical BLAST 1.4 format, although both formats are not supported at the same time. For example:

```
blastn "pri rod mam vrt htg" myquery.nt
```

- WU BLAST 2.0 supports *segmented query sequences*, which can be useful for representing, in a single sequence, the contigs that result from shotgun sequencing assembly or multiple short probes for a given gene. The feature is easy to employ. All of the contigs from a given clone can simply be concatenated together with a single hyphen (-) character inserted in between each contig. In contrast to other methods where long runs of ambiguity codes are inserted between contigs, segment boundaries are clearly recognizable and consume less storage. BLAST 2.0 honors segment boundaries by *guaranteeing* that no alignment — ungapped or gapped — will cross a boundary.

- Multi-sequence query files are supported BLAST 2.0, such that every sequence in the FASTA file is searched against the specified database. Each search result is separated from the next by a single ASCII form-feed character (control-L or hex 0x0c). See the qrecmin and qrecmax options.
- The format of all dates (and times) reported in WU BLAST output is controlled by the UN*X standard `CFTIME` environment variable. Dates will be reported in ISO 8601 format, if `CFTIME` is set to `'%Y-%m-%dT%H:%M:%S'`. Dates produced according to ISO 8601 are single tokens that can be compared lexicographically to immediately recognize their relative chronological order, without having to parse out and compare the individual date components (year, month, day, hour, etc.) Dates reported by the **xdformat** program are also governed by `CFTIME`. The format of many dates reported by the search programs for XDF databases is determined by the setting (if any) of `CFTIME` when the database was created or last modified by **xdformat**.
- Both *sequence filtering* and *word masking* of query sequences are supported. The terms "filter" and "mask" are sometimes used alone and interchangeably, however there are two distinct techniques people can use which deserve separate names. Lower case alphabetic letters in the query sequence can be used to inform the BLAST search program as to which residues it should either *filter* (convert to X or N) or *mask* (skip when generating neighborhood words but otherwise leave intact). See the **lcfilter** and **lcmask** options, respectively.
- Multiple `filter=<`*`filter`*`>` specifications can be requested on the BLAST command line. Each filter is executed independently and their results are OR-ed at the end.
- Whereas NCBI BLAST 2.0 uses the original external filtering technique of BLAST 1.3 (Gish, W., unpublished), which utilizes the UNIX popen() system call and temporary files, WU BLAST 2.0 avoids these problematic system interfaces.
- One or more **word masks** can be specified on the command line, using the `wordmask=<`*`mask`*`>` option, where <*mask*> may be a classical filter program such as `seg`, `xnu`, or `dust`. Whereas sequence filters convert certain letters in the query sequence into ambiguity codes (X for amino acid and N for nucleotide), word masks do not alter the sequence itself. Instead, word masks cause certain portions of the query sequence to be skipped during the neighborhood word generation step of the BLAST algorithm. This leaves the query sequence intact for generating comprehensive alignments seeded by neighborhood word hits involving more informative, unmasked regions of the sequence.
- The BLAST algorithm's word length parameter, *W*, can be set from 1 to 1024 in *all* search modes (`BLASTP`, `BLASTN`, `BLASTX`, `TBLASTN`, `TBLASTX`).
- WU BLAST 2.0 reliably supports parallel processing on a variety of SMP (symmetric multiprocessing) computing platforms. WU BLAST threads properly across multiple CPUs on dual-processor Apple PowerMacs running Mac OS X and does not require a G4 processor. POSIX threads are used under Compaq Tru64 UNIX 4.0+, Linux for X86 and Alpha processors, IBM AIX, Mac OS X, IRIX 6.5, and HP-UX 11. While POSIX threads are available under Solaris 2+ (SPARC, X86 and X64), Solaris threads are specifically used instead for slightly better performance. The IRIX m_fork() system call provides parallel processing under older versions of IRIX 4 through 6.4; and DCE threads are used under Digital UNIX 3.2.
- To illustrate just some of the flexibility available in WU BLAST 2.0, includes a PERL script named **wu-blastall** that translates an NCBI blastall command line into a *rough* equivalent WU BLAST command line and then invokes the appropriate WU BLAST search mode. The output remains in WU BLAST format, but the **wu-blastall** script may help users of the NCBI blastall program migrate to WU BLAST and start to discover its power.
  NOTE: the **wu-blastall** script is not intended to provide a literal replacement for the NCBI blastall program. Aside from being unable to mimic all aspects of the NCBI software, the script is expected to hinder user's adoption of important features that the NCBI software does not provide, it directs the software to use less sensitive parameters by default, and may be inappropriate for assessing the relative performance (sensitivity, specificity, accuracy or speed) of the NCBI and WU packages.
- The MaskerAid substitute for CrossMatch (Phil Green, unpublished) provides another example of how the unique combination of flexibility and speed of WU BLAST 2.0 has been applied to yield 30-fold faster performance of RepeatMasker in its slow mode, while maintaining sensitivity.
- Many additional command line options are available, most of which are described here.

- The WU BLAST package includes a compiled version of the nrdb program for merging identical sequences into one, as well as the newer **patdb** program (W. Gish, unpublished) for obtaining further data compression through perfect substring elimination.
- The **seg**, **xnu** and **dust** complexity filter programs are included as distinct programs that can be executed independently of the BLAST search programs. This facilitates analysis of the precise segments that are filtered and allows the filters to be utilized in other situations besides just working with BLAST. The WU BLAST search programs also support an `echofilter` option to display the filtered segments in the search output.

Please send bug reports, questions, or suggestions to BLAST E-mail

Manifest

The BLAST 2.0 package from Washington University includes the following data analysis and utility programs:

- **blasta** — the unified database search program, which provides **blastp**, **blastn**, **blastx**, **tblastn**, and **tblastx** search functionality.
  - **blastp**: compare peptide sequence queries to peptide sequence databases;
  - **blastn**: compare nucleotide sequence queries to nucleotide sequence databases;
  - **blastx**: compare nucleotide sequence queries dynamically translated in all 6 reading frames to peptide sequence databases;
  - **tblastn**: compare peptide sequence queries to nucleotide sequence databases dynamically translated in all 6 reading frames;
  - **tblastx**: compare nucleotide sequence queries dynamically translated in all 6 reading frames to nucleotide sequence databases dynamically translated in all 6 reading frames (6 x 6 = 36 pairwise combinations of reading frames).
- **xdformat** — the recommended program for rapidly converting sequences from FASTA format into the native **XDF** format read by **blasta**. The program can also append new sequences to an existing database; automatically rollback on errors; provides flexible indexing and verification services; and can dump data back into FASTA format.
- **xdget** — a flexible tool for retrieving sequences (or segments thereof) from an indexed XDF database; retrieved sequences are optionally reverse-complemented and translated in the case of nucleotide sequences. **xdformat** and **xdget** are actually one-and-the-same program, to ensure their compatibility.
- **nrdb** — a tool for rapidly removing trivial redundancy (*i.e.*, identical sequences) from one or more input files in FASTA format. A simple hash table is used, combined with data compression techniques to allow larger nucleotide sequence data sets to be manipulated in memory.
- **patdb** — on the surface, merely a poor substitute for the **nrdb** program, because it uses more memory when processing nucleotide sequences and yet accomplishes the same task by default. Hidden within its command line options, though, is the capability of identifying not just identical sequences — which is all that the **nrdb** program can do — but identifying sequences that are perfect *substrings* of others. (See the `-s` option of the program). A Patricia tree is used by the program (hence its name), automatically followed (when necessary) by one or more clean-up stages that use finite state automata. **patdb**, with its substring identification option, may be most usefully applied to protein sequences, which often differ only in their inclusion or exclusion of the initiator methionine and other post-translational modifications. When identification of perfect substrings is not desired, the **nrdb** program is more practical than **patdb** for processing nucleotide sequences, because of data compression techniques effectively used by **nrdb** that are not available in **patdb**.
- **wu-blastall** — a PERL script for converting an NCBI **blastall** command line into a *rough* equivalent **blasta** command line and then invoking **blasta**. The output is still in WU BLAST format. This is primarily intended as a technology demonstration tool but may also assist users in their migration from NCBI BLAST to the more accurate WU BLAST. For benchmarking of

BLASTs, careful tweaking of parameters may be required, but even with great care, benchmarking for speed can still be confounded by inaccuracies in NCBI BLAST.

- **wu-formatdb** — a PERL script for converting an NCBI **formatdb** command line into the equivalent **xdformat** command line and then invoking **xdformat**. This is primarily intended as a technology demonstration tool but may also assist users in their migration from NCBI BLAST to WU BLAST.
- **pam** — a program to compute amino acid substitution scoring matrices having arbitrary scales, using the Dayhoff PAM model.
- **pressdb.real** — the legacy **pressdb** program for users who are reliant on the NCBI BLAST 1.4 database format for nucleotide sequences.
- **setdb.real** — the legacy **setdb** program for users who are reliant on the NCBI BLAST 1.4 database format for amino acid sequences.
- **gb2fasta** — a parser to extract nucleotide sequences from GenBank flat files into FASTA format.
- **gt2fasta** — a parser to extract amino acid sequences from CDS features in GenBank flat files and output them in FASTA format.
- **sp2fasta** — a parser to extract protein or nucleotide sequences from EMBL, TrEMBL, or Swiss-Prot database files and output them in FASTA format.
- **pir2fasta** — a parser to extract protein sequences from NBRF PIR database files and output them in FASTA format.
- **seg** — a low-complexity filter for protein and nucleotide sequences (Wootton and Federhen, 1993; Wootton and Federhen, 1996). The program identifies low compositional complexity regions.
- **dust** — a low-complexity filter for nucleotide sequences (Hancock and Armstrong, 1994; Tatusov and Lipman, unpublished).
- **xnu** — a low-complexity filter for protein sequences (Claverie and States, 1993). The program identifies short-periodicity repeats.
- **sysblast.sample** — a sample configuration file that system administrators may wish to modify and install as **/etc/sysblast**. For all users on a system-wide basis, parameter settings in this file can be used to establish:
  - a limit on the number of CPUs or threads employed by each BLAST job;
  - the default number of CPUs or threads employed per BLAST job;
  - a "nice" value for BLAST processes;
  - the maximum amount of memory allocated per BLAST job.

To Fly...

If the gapped alignments are nice, but even more speed or less memory use are desired, read how to make the programs fly.

Examples

1. One timed benchmark is a BLASTN comparison of unmasked *Arabidopsis thaliana* chromosomes 2 and 4, which are respectively 19.6 Mbp and 17.5 Mbp in length. The computer used in this particular example was a quad-processor Pentium*III* Xeon system (550 MHz, 512 KB L2 cache per processor) running the Mandrake Linux 2.4.22-10mdk kernel. Using a single thread of execution (one processor), BLASTN 2.0 [29-Apr-2004] required **10 minutes 42 seconds** elapsed (wall clock) time (10 minutes 38 seconds CPU time) and approximately 850 MB of memory to search both strands at once with the command:

```
wu-blastall -p blastn -d at.chr2 -i at.chr4 -Ff
```

The same job took **3 minutes 00 seconds** elapsed time to complete on a single 2.4 GHz AMD Opteron model 250 processor running the Linux 2.6.9 kernel.

2. Using the `wu-blastall` wrapper script and a single processor thread, the 2.18 Mb genome of *Neisseria meningitis* serogroup A was compared against the 2.27 Mb genome of *Neisseria meningitis* serogroup B in less than **8 seconds** elapsed and CPU time.

3. Using the `wu-blastall` wrapper script, the entire 3.098 Gb human genome reference sequence was compared to itself using BLASTN 2.0 [15-Nov-2004] on a dual 900 MHz Itanium2 ("Merced") processor computer system with 10 GB memory, running Linux 2.4.18, in less than **13 days 2 hours** elapsed time, using two threads (both processors). On a dual 2.4 GHz AMD Opteron system, the same search was completed within **6 days 21 hours** elapsed time. In this study, the human genome sequence was masked in advance for interspersed repeats and low-complexity regions. Note that the Itanium system used here was 1-1/2 years older than the Opteron system, and so these benchmarks should not be construed as pitting two contemporary technologies against each other. Note also that the execution times reported here would be nearly halved — with no loss of information — had the whole genome cross comparison not compared the chromosomal sequences to each other twice (*e.g.*, compare chromosome 1 against chromosome 2 but not chromosome 2 against chromosome 1).

When the seed word length was increased to 14 from its default of 11, the same whole human genome cross-comparison was completed in less than **19 hours 26 minutes** elapsed time on the dual Opteron system. For this search, approximately 16 GB memory were required. On a quad-processor 1.5 GHz Itanium2 system, the job was completed in under **8 hours 54 minutes**; for this search, approximately 24 GB memory were required.

The longest sequence in the 42-record human genome sequence data set was a 246 Mb contig for chromosome 1. It was compared using the default seed word length against the entire genome (including the chromosome 1 query itself) within **1 day 2 hours** elapsed time on the quad Itanium2 system, within **12 hours 51 minutes** elapsed time on the dual Opteron system, and within **11 hours 23 minutes** on a 2.5 GHz Quad-processor PowerMac G5 (Processor Performance set to “Highest” in the Energy Saver control panel).

4. Below are some sample WU BLAST 2.0 results produced using default parameters, with the exception of the often-recommended `seg` low-complexity filter and the frequently used -postsw option of WU BLASTP 2.0. The specific exceptions to the defaults are noted in each case.
Default parameters for NCBI blastall were also used, with the exception of using -G7 -E2 to make the scoring system identical to the WU default gap penalty of 9 for the first residue in a gap and 2 for subsequent residues in the gap.

- **Example 1** `BLASTP` output produced with and without gaps; *with* gaps but without Karlin-Altschul “Sum” statistics; and *with* gaps and Karlin-Altschul “Sum” statistics but without the -postsw option.
  Example 1 NCBI blastall output.
- **Example 2** `BLASTP` output produced with and without gaps; and *with* gaps but without Karlin-Altschul “Sum” statistics.
  Example 2 NCBI blastall output.
- **Example 3** `BLASTP` output produced with and without gaps.
  Example 3 NCBI blastall output.

Command Line Options

Descriptions of the command line options and parameters available in WU BLAST 2.0 are here.

Environment Variables

As described below and elsewhere, WU BLAST 2.0 supports several environment variables to adapt its behavior to different computing environments: `BLASTDB`, `BLASTFILTER` and `BLASTMAT`. To support dual

WU/NCBI BLAST installations, WU BLAST also supports the environment variables WUBLASTDB, WUBLASTFILTER and WUBLASTMAT, with the WU versions of these variables taking precedence over the corresponding non-WU versions when both are set.

In WU BLAST 2.0, the BLASTDB (or WUBLASTDB) environment variable can be a list of one *or more* directory names in which the programs are to look for database files. In UNIX parlance, such an environment variable might be called a *path* for the database files. Directory names should be delimited from one another by a colon (":") and listed in the order that they should be searched. If the BLASTDB environment variable is not set, the programs use a default path of `".:/usr/ncbi/blast/db"`, such that the programs first look in the current working directory (".") for the requested database and then look in the `/usr/ncbi/blast/db` directory. For backward compatibility with programs that expect BLASTDB to be a single directory specification and not a path, if the user has set a value for BLASTDB but omitted the current working directory, the version 2 programs will still look for database files in the current working directory as a last resort.

The BLASTFILTER (or WUBLASTFILTER) environment variable can be set to the directory containing the filter programs, such as seg and xnu. The default directory for the filter programs is `/usr/ncbi/blast/filter`. This usage is unchanged from version 1.4.

The BLASTMAT (or WUBLASTMAT) environment variable can be set to the parent directory for all scoring matrix files. The default directory for these files is `/usr/ncbi/blast/matrix`, beneath which are nt and aa subdirectories for storing scoring matrix files appropriate for nucleotide and amino acid alphabets. This usage is unchanged from version 1.4.

For more information about environment variables, see the Installation instructions.

Filters and Masks

WU BLAST provides highly flexible means for applying both “hard” and “soft” masks to a query sequence; supports alternative, user-defined filter programs; and allows the use of non-standard parameters to the standard filters. The `filter` option (for hard masking) and the `wordmask` option (for soft masking) provide the basic interface. Multiple specifications of each type are acceptable on the BLAST command line; and individual filter and wordmask specifications may consist of entire pipelines of commands.

For example, three filters are used in succession by this pipeline:

```
filter="myfilter1 | myfilter2 | myfilter3 -x5 -"
```

The first two filters in this case are expecting to read their input from UN*X standard input (also known as *stdin*), whereas *myfilter3* apparently needs to be told (with the usual "-" or hyphen argument) to read data from stdin. The standard output (*stdout*) from *myfilter1* will be read via stdin by *myfilter2*, which in turn processes the query before handing its results to *myfilter3*; finally, *myfilter3* reports its results to stdout, which the BLAST program itself reads to obtain the fully masked sequence. The final output from the filter pipeline is expected by the BLAST program to be in FASTA format.

Instead of running all 3 filters in the above example as part of one pipeline, they could instead be specified as separate filter options like this:

```
filter=myfilter1  filter=myfilter2  filter="myfilter3 -x5 -"
```

The same choice of running as a pipeline or running separately is available for wordmasks, too. And of course the two approaches can be combined on the same command line. An advantage to using the pipeline approach is that all 3 filters in the example above may complete a little bit faster, because much

of the I/O is avoided. Furthermore, when used in the pipeline, there's no requirement that the output from *myfilter1* and *myfilter2* actually be in FASTA format. Those two programs could potentially pass any information between themselves and to *myfilter3*. The only absolute requirement is that *myfilter1* must read FASTA data from stdin and *myfilter3* must output FASTA data (of the same length as the query!) to stdout.

It should be noted that with some filter programs, passing the query sequence sequentially through a pipeline of filters may yield a different result than processing the query independently with each filter and OR-ing the results. The script **seg+xnu** included in the `filter/` directory provides an example with which to test this. Specifying *filter=seg+xnu* on the BLAST command line invokes a seg and xnu pipeline that is built-in to the search programs; whereas specifying `filter="seg+xnu -"` causes the **seg+xnu** script to be invoked on the query, which independently executes **seg** and **xnu**, then ORs the separate results with **pmerge**. (The `echofilter` option can be used to see the results of filtering displayed in search program output). While the built-in seg+xnu pipeline is historically the way these two filters have been implemented, the latter interpretation, as illustrated by the seg+xnu script with pmerge, may be more desirable.

Bugs

Although WU BLAST is certainly not bug free, historically bugs have been fixed typically within 24 hours of their being reported. No outstanding issues exist with the *current* version. If you believe you might be experiencing the effects of a bug, please file a BLAST Bug Report

The software does have some characteristics worth mentioning, however, that could trip up or confuse even the most knowledgeable of BLAST users. Any *unexpected* behavior might rightfully be construed as being a bug, so the following information is provided to help avoid the unexpected. If you should encounter problems or confusing areas other than those described below, or if you have questions or suggestions, please send them to BLAST Questions & Suggestions.

- Mac OS X 10.4 “Tiger” has not proven itself to be a reliable platform for 64-bit computing when performing searches using multiple threads combined with a need for “large” memory (more than about 2.5 GB). For example, in dozens of attempts to compare the human genome to itself on all Tiger releases through version 10.4.6, searches *never* advanced more than one-third of the way to completion before a system panic arose that required a hard reset — but only when multiple CPUs or threads were employed. Sometimes a panic was induced after mere minutes had elapsed, and in other cases several hours or a few days were required. Running in “safe mode” (booting while holding down the shift key) might alleviate the problem somewhat but certainly does not eliminate it. (No such issues have been observed with non-Mac platforms. A “user-land” application such as BLAST should not be capable of causing a panic). In contrast, the same large-memory jobs when confined to a single CPU have not been observed to induce a panic, even after nearly 3 weeks of continuous execution. In response to this persistent problem, the behavior of WU BLAST has been modified specifically in the case of the 64-bit binaries distributed in `blast2.macosx-p64.tar.Z`. When these 64-bit search programs see that more than about 2 GB of memory will be needed, they automatically and unconditionally limit themselves to using a single CPU, regardless of how many CPUs are present in the computer or how many CPUs the user requests. Only if the memory requirements are seen to be less than 2 GB can additional CPUs be employed for a single search. For example, all four processors in a Quad G5 system may be employed if the memory requirements are below the threshold. It is worth noting, however, that for jobs requiring less than 2 GB memory, 64-bit virtual addressing is unnecessary and speedier execution is usually obtained from the 32-bit, G5-optimized binaries found in the `blast2.macosx-g5.tar.Z` distribution.
- When BLAST is executed on early 32-bit computing platforms (*e.g.,* Solaris 2.5 and earlier, IRIX 5 and earlier, and Linux 2.2 or earlier), due to a 2 GB file size limit inherent to these operating systems, users will be unable to search nucleotide sequence databases larger than about 8 billion

nucleotides or 2 billion amino acids. Migrating to a more contemporary 32-bit operating system — or to a 64-bit computing platform — that provides "largefile" support is sufficient to break through the "2 GB barrier".

- The statistical significance of *gapped* alignment scores is computed using values for λ, *K* and *H* that are looked up in precomputed tables. (The values for λ, *K* and *H* used in assessing the significance of *ungapped* alignment scores are computed at run time, which is practical). Values are chosen from these tables based on the scoring matrix and gap penalties used. Precomputed values are not available for all scoring matrix and gap penalty combinations, however; and the precomputed values may not be well-suited to an unusual residue composition of the query sequence. When precomputed values are unavailable, the programs issue a WARNING and proceed to evaluate gapped alignment scores using instead the values of λ, *K* and *H* computed for ungapped alignments, which are almost certainly inappropriate. In such cases, the reported significance estimates may be *highly* inaccurate and biased towards over-estimating the significance (under-estimating the probability of chance occurrence). If the user knows more accurate values for their situation, the **gapK**, **gapL** and **gapH** command line options should be used to set them.
- Precomputed values for λ, *K* and *H* are available for BLASTN searches with the following match,mismatch (M,N) scoring systems, using gap penalties {Q,R}:

```
"+1,-3", {3,3} {3,2} {3,1}
"+1,-2", {2,2} {2,1} {1,1}
"+3,-5", {10,5} {6,3} {5,5}
"+4,-5", {10,5}
"+1,-1", {3,1} {2,1}
"+5,-4", {20,10} {10,10}
"+5,-11", {22,22} {22,11} {12,2} {11,11}
```

and for the Purine-Pyrimidine scoring matrix named "pupy":

```
pupy =
        { 20, 10}
        { 10, 10}
```

- Precomputed values for λ, *K* and *H* are available for protein-level searches with the following scoring matrix and gap penalty combinations (or gap penalty ranges for R) {Q, R}:

```
blosum50 =
    { 16,  1-4}
    { 15,  1-4,6,8}
    { 14,  1-5,8}
    { 13,  1-5,8}
    { 12,  2-5,7}
    { 11,  2-4,6,8}
    { 10,  2-6,8}
    { 9,  3-5,7}
    { 8,  4-8}
    { 7,  6,7}

blosum55 =
    { 16,  1-4}
    { 15,  1-4,5,6,8}
    { 14,  1-5,7}
    { 13,  2-5,8}
    { 12,  2-5,8}
    { 11,  2-6,8}
```

```
    { 10,  3-6,9}
    { 9,  3-5,7}
    { 8,  4-8}
    { 7,  7}

blosum62 =
    { 12,  1-3}
    { 11,  1-3}
    { 10,  1-4}
    { 9,  1-5}
    { 8,  2-7}
    { 7,  2-6}
    { 6,  3-5}
    { 5,  5}

blosum80 =
    { 12, 2-12}
    { 11, 2-11}
    { 10, 2-10}
    {  9,  3-9}
    {  8,  4-8}
    {  7,  5-7}

pam40 =
    { 12,  1,2,6}
    { 11,  1,2,7}
    { 10,  1-3,7}
    { 9,  1-3,6}
    { 8,  1-4}
    { 7,  1-4}
    { 6,  2-5}
    { 5,  2-5}
    { 4,  3,4}

pam120 =
    { 12,  1,2,4}
    { 11,  1-3}
    { 10,  1-3,5}
    { 9,  1-3,5}
    { 8,  1-4,6}
    { 7,  2-4,6}
    { 6,  2-5}
    { 5,  3-5}

pam250 =
    { 16,  1-4}
    { 15,  1-5}
    { 14,  1-6}
    { 13,  1-6}
    { 12,  2-7}
    { 11,  2-7}
    { 10,  3-8}
    { 9,  3-7}
    { 8,  5-7}
    { 7,  7}
```

- Selecting an alternative scoring matrix does *not* alter the gap penalties (**Q** and **R**) from their default values. This can not only result in alignments with undesirable gap characteristics, but depending on the scoring matrix chosen, this can unwittingly create a situation in which the

programs do not have precomputed values for λ, *K* and *H*. As described earlier, a WARNING message will be displayed when precomputed values are not available; nevertheless, the statistics will be unreliable.
- The hspsepQmax and hspsepSmax parameters are measures of distance in residues along the sequences in the specific form in which they are being compared. For instance, in a BLASTX search, where the query sequence is conceptually translated, the hspsepQmax distance is in units of amino acid residues, not the underlying nucleotides of the query.
- The gap penalty parameters **Q** and **R** of WU BLAST have similar but important differences in interpretation from the parameters **G** and **E** of NCBI Gapped BLAST. While the two extension penalties **R** (WU BLAST) and **E** (NCBI BLAST) are analogous, **Q** (WU BLAST) is analogous to the *sum* of **G** and **E** with NCBI BLAST. In other words, where **Q** is the total penalty for a gap of length 1, NCBI Gapped BLAST computes this penalty as **G + E**.

Supported Platforms

The computing platforms currently supported by BLAST 2.0 include the following:

- Apple Mac OS X 10.1 through 10.4 for PowerPC G3 and G4; Mac OS X 10.3 and 10.4 for PowerPC G5, including 64-bit virtual addressing (Mac OS X 10.4 only); Mac OS X 10.4 for Intel X86 processors; "Universal" binaries for Mac OS X 10.4
- Compaq Tru64 UNIX 5.0A for generic Alpha and Alpha EV5, EV56, EV6, and EV67
- FreeBSD 4.x, 5.x and 6.x for Intel i686 (PentiumPro/II/*III*) and X64*
- Hewlett-Packard HP-UX 11 for Intel IA-64 (Itanium)
- IBM AIX 5.2 for Power3, Power4 and Power5
- Linux kernel version 2.2 for i586 (original Pentium) and i686 (PentiumPro/II/*III*)
- Linux kernel version 2.4 with Linux threads for 32-bit i686 (PentiumPro/II/*III*) and i786 (Pentium4); and 64-bit Intel IA-64 (Itanium) and X64*
- Linux kernel version 2.4 with Native POSIX Threads (NPTL) support for 32-bit and 64-bit PowerPC, PPC970, Power3, Power4 and Power5
- Linux kernel version 2.6 with Native POSIX Threads (NPTL) support for 32-bit i686 and i786; and 64-bit Intel IA-64 (Itanium) and X64*
- SGI IRIX 6.5 for MIPS R5000, R10000, R12000, R14000 and R16000
- Sun Solaris 8 for SPARC and UltraSPARC
- Sun Solaris 9 and higher for Intel i686 (PentiumPro/II/*III*)
- Sun Solaris 10 for 64-bit X64*

***X64** is shorthand for the AMD "AMD64" and Intel "EM64T" microprocessor architectures, which support 64-bit virtual addressing. X64 binaries often provide significantly better performance than their 32-bit counterparts built for the legacy 32-bit architecture known as X86. The X64 architecture is sometimes referred to by another vendor-agnostic name, X86_64.

The list of supported platforms is subject to change without notice. It was last updated 28-Jan-2006**.**

Multiple processors (multithreading or parallel processing) are effectively and efficiently supported by WU BLAST on all of the above platforms. The software also supports large files (files greater than 2 GB in size) when the underlying operating system and file system support large files (which is typically the case these days).

WU BLAST was the only BLAST available for Mac OS X when Mac OS X became publicly available — and for months thereafter. Under Mac OS X, WU BLAST has been observed to be the only BLAST that runs faster on multiple G4 processors, conditions in which some implementations actually run *slower* and are unstable. There is no evidence that vector processing instructions (such those provided

by Velocity Engine) increase the speed of BLAST searching, but these instructions certainly can restrict the software to running only on certain computers. WU BLAST obtains its superior speed through painstaking optimizations, without using specialized instructions, so users can run WU BLAST even on a G3. Because the software employs a command line interface, it can also be run on the freely available OpenDarwin operating system.

Installation

Please refer to the `README.html` *file that comes bundled with the software for more detailed and specific installation instructions.*

Low-complexity sequence filters or masking programs — *e.g.,* seg, xnu and dust — are bundled in WU BLAST software packages. Whatever directory you install the filter programs in, the `BLASTFILTER` environment variable should be set to point there. In the absence of this environment variable being set, the programs look for masking programs in `/usr/ncbi/blast/filter`. NOTE: unlike the NCBI search programs, WU BLAST does *not* employ sequence filtering by default.

Databases can be downloaded from any of the many sources available on the Internet. After downloading, the database files are typically uncompressed and processed into FASTA format, then into a BLAST-able database format. Included with WU BLAST software are several utility programs for converting text-based database files into FASTA format:

- **gb2fasta** converts the nucleotide sequences in GenBank flat files into FASTA format.
- **gt2fasta** converts the CDS translations (peptide sequences) in GenBank flat files into FASTA format.
- **sp2fasta** converts EMBL or SWISS-PROT flat files into FASTA format.

The NCBI software Toolbox also contains parsers, including one named **asn2fast** that can convert both nucleotide and peptide sequences in GenBank ASN.1 format into FASTA format files.

All of the above parsers can read from standard input (sometimes signified by a dash, "-"), so their input files can be maintained on disk in compressed format and dynamically zcat-ed or gunzip-ed directly into the parsers, thus saving the time and storage required for the uncompressed data. To specify standard input for a required input filename argument, some of these programs require that a double-dash (--) precede the single-dash. This double-dash signifies the end of the command line options and the start of the required arguments.

Once databases are in FASTA format, the **xdformat** program is used to convert them into a blastable e**X**tended **D**atabase **F**ormat. Terse usage instructions for this program can be obtained by invoking it without command line arguments. When producing a blastable database, **xdformat** creates 3 or 4 output files whose names by default are derived from the name of the input FASTA-format file. The output files are given distinct filename extensions and together comprise the blastable database. More information about blastable database file formats is available here.

The blastable database files can be placed anywhere, but the **BLASTDB** environment variable should point to their directory location. If the **BLASTDB** environment variable is not set, the programs look for databases in `/usr/ncbi/blast/db` and in the current working directory. On systems where NCBI BLAST will not be used, databases can be maintained in *multiple* directories listed in the **BLASTDB** environment variable, delimiting the directory names with colons, just as directory names are delimited in the PATH environment variable used by UNIX command shells.

On multi-processor computer systems, the search programs will by default employ as many CPUs as are installed (or up to 4 CPUs in the case of `BLASTN`). Using too many processors — sometimes even two processors — can be inefficient or lead to prohibitive memory requirements. Depending on how

many processors and how much memory are installed in your computer, you may want to wrap the search programs in a shell script that sets a lower number of CPUs via the **cpus=#** command line option. Another approach to changing the default number of CPUs follows below, for BLAST managers brandishing “root” or “SuperUser” privileges.

A sample file named `sysblast.sample` is bundled with the software, to help in establishing system-wide configuration parameters governing the behavior of BLAST processes. When installed under the name `/etc/sysblast`,

- the default number of CPUs employed can be altered;
- a hard limit can be imposed on the number of CPUs employable by any single BLAST process;
- and the “nice” (execution priority) value of BLAST processes can be set.

The file `/etc/sysblast` resides in a directory that is local to any given computer, so parameter values can be configured differently for different computers, even if the software itself is accessed from a shared disk partition. `sysblast` is only effective if installed in the `/etc` directory; and the `/etc` directory should only be writable by “root”. See the comments included in `sysblast.sample` file for further details. Unlike the shell script wrapper approach described earlier, limits set in `/etc/sysblast` can not be easily or unwittingly circumvented.

Citing WU-BLAST

Citations or acknowledgements of WU-BLAST usage are greatly appreciated, as are any personal accounts of how the software is being used that you might wish to share. When URLs are acceptable, please cite with:

```
Gish, W. (1996-2004) http://blast.wustl.edu
```

When URLs are not acceptable, please use:

```
Gish, W., personal communication.
```

The WU-BLAST unified search program may also be referred to by the name **BLASTA**.

In scientific communications, it is important to report the program name, as well as the specific version(s) used. In the case of WU-BLAST or BLASTA, the version is a combination of the "2.0" moniker *and* the release date. The release date can be found on the first line of output, and it is the first date displayed. For example, consider this introductory line of output:

```
BLASTN 2.0MP-WashU [02-Apr-2002] [sol8-ultra-ILP32F64 2002-04-03T01:25:46]
```

In the above, the software release date is April 2, 2002, whereas the build date of the Solaris 8 UltraSPARC binary executable was April 3rd at 1:25 AM.

Historical Notes

WU BLAST 2.0 is the original gapped BLAST with statistics. It builds upon BLAST 1.4 written by Warren Gish in 1994, while a fellow at the NCBI. (See http://blast.wustl.edu/blast-1.4; Altschul *et al.*, 1990; Gish and States, 1993). Both NCBI BLAST and WU BLAST 1.4 (but not 2.0) are in the public domain.

Development of BLAST version 2 with gapped alignments was begun by W. Gish as an independently funded research and development effort at Washington University in late 1994, where it continues as such today. WU BLAST 2.0 was initially released as free, copyrighted software in May 1996, before

the NCBI expressed an interest in pursuing this area and 16 months ahead of the NCBI releasing its public domain gapped alignment tools in September 1997. Beginning in October 1997, in response to the NCBI release, more advanced versions of WU BLAST were made available only under the covenant of a license agreement. The last freely available (and now long obsolete) version of WU BLAST is 2.0a19, posted in February 1998.

Historical notes and additional citation information for some earlier versions of NCBI and WU BLAST include:

- The first description of the classical ungapped BLAST algorithm was published by Altschul *et al.* (1990). The paper touts the flexibility of the pattern matching algorithm and describes the implementation of BLASTP and BLASTN. Development of BLASTX is alluded to, but TBLASTN was actually the third search mode implemented. A version of BLAST is also mentioned that uses a complex, dynamic programming scheme to extend hits and allow gaps in alignments, but the approach was far too slow. In addition, selectivity was compromised because satisfactory statistics were unavailable for evaluating the gapped alignment scores.
- Some of the novel methodologies contributed to the BLAST project by W. Gish include the use of: 4-to-1 compression of nucleotide sequence databases to reduce I/O costs; searching nucleotide databases natively in their compressed form for higher speed; memory-resident databases to reduce I/O bottlenecks; parallel processing for faster response to individual queries; deterministic finite state automata (DFA), to find seed words faster while often using less memory than a simple lookup table; placement of sentinel bytes at the beginning and end of sequences to facilitate faster word hit extensions; transparent execution of external “filter” programs to pre-process query sequences prior to a search (*e.g.,* to mask low-complexity regions).
- The NCBI *Experimental BLAST Network Service* (W. Gish, unpublished) was opened to the public in December 1989, providing Internet access to the latest versions of the BLAST programs and sequence databases updated on a daily basis. Searches were executed individually in parallel across the 8-processors of a Silicon Graphics Power IRIS server. Around the same time, the "nr" (quasi-non-redundant) databases were established (W. Gish, unpublished). At the request of NCBI management, a description of the experimental service was never published. Nevertheless, awareness of the service spread quickly by word-of-mouth, as is the case with WU BLAST. The experimental form of the service was ultimately discontinued more than a decade later, in March 2000.
- BLASTX first appeared in BLAST 1.1 in July 1990, and was later described and characterized by Gish and States (1993). The BLAST3 program (Altschul and Lipman, 1990) was also folded into the 1.1 release and parallelized. The use of Poisson statistics, as suggested by Karlin and Altschul (1990) to evaluate the joint probability of multiple HSPs, was also first featured in BLAST 1.1.
- BLASTC, a version of BLASTX that considered codon usage information in addition to sequence similarity (States and Gish, 1994), only appeared in the BLAST 1.3 distribution. The BLAST 1.3 distribution was also the last to include the BLAST3 program.
- The first version of BLAST to use Karlin and Altschul (1993) “Sum” statistics to evaluate the joint probability of multiple HSPs was BLAST 1.4 (W. Gish, unpublished).
- TBLASTX first appeared in BLAST 1.4 and remains attributable to W. Gish (unpublished).
- The first release of **WU BLAST** was version 1.4, which was virtually identical to NCBI BLAST 1.4, save for a few bug fixes. The **WU BLAST Archives** (*http://blast.wustl.edu*) first appeared on the Internet in 1995, to provide continued support for the work begun at the NCBI, as well as to provide a central location where BLAST-related software, information, and earlier software versions could be obtained.
- Upon my invitation, Stephen Altschul entered into a collaboration with me in late 1994, to provide support for my conjecture that pre-computed values for λ, K and H — along with Sum statistics for joint probability calculations on multiple HSP scores — could be usefully applied to the evaluation of locally optimal *gapped* alignment scores. Results from this work eventually appeared in Altschul and Gish (1996) and provide much of the foundation for today's WU BLAST 2.0 (and NCBI blastall).

- The original gapped BLAST package (BLASTP, BLASTN, BLASTX, TBLASTN and TBLASTX) with statistical significance estimates was publicly released as **WU-BLAST 2.0a**, just in time for presentation at the Cold Spring Harbor Genome Mapping and Sequencing conference in May of 1996. The software provided the full complement of BLAST search modes (BLASTP, BLASTN, BLASTX, TBLASTN and TBLASTX) and could evaluate the significance of alignment scores using standard Karlin-Altschul statistics, Poisson statistics or Sum statistics as described by Altschul and Gish (1996). The enhanced software package was a virtual drop-in upgrade for users of the previous ungapped BLAST 1.4.
- In addition to the new statistics, WU-BLAST 2.0 utilized novel ideas for making gapped alignments both practical (from a performance perspective) and effective (from a sensitivity/selectivity perspective) for high-throughput systems. The ungapped HSPs from a classical BLAST search were utilized as seeds for finding gapped alignments much as FASTA did (Pearson and Lipman, 1988). However, unlike FASTA, BLAST applied the gapped alignment step to *all* HSPs, not just a select one, and it used a secondary drop-off score (X) in the gapped alignment phase, rather than necessarily continuing expensive dynamic programming steps all the way to the ends of the sequences. By "gapping" all HSPs, computing joint probability calculations on consistent sets of alignment scores, and reporting all alignments that satisfy a statistical significance threshold, BLAST gained an advantage in sensitivity over FASTA and even the Smith-Waterman algorithm, which report only the single highest scoring alignment.
- The NCBI published its version 2 of BLAST, or **Gapped BLAST**, including a description of the **2-hit** BLAST for protein searches and **PSI-BLAST** algorithms, in Altschul *et al.* (1997), in September 1997. Almost immediately a faster, more sensitive variation of the NCBI 2-hit BLAST algorithm was implemented uniformly and provided as an option in WU-BLAST across all search modes (including BLASTN).
- The NCBI published a description of **PHI-BLAST** in Zhang *et al.* 1998.

References

Altschul, SF, and W Gish (1996). Local alignment statistics. ed. R. Doolittle. Methods in Enzymology **266**:460-80.

Altschul, SF, and DJ Lipman (1990). Protein database searches for multiple alignments. Proc. Natl. Acad. Sci. USA **87**:5509-13.

Altschul, SF, Gish, W, Miller, W, Myers, EW, and DJ Lipman (1990). Basic local alignment search tool. J. of Mol. Biol. **215**:403-10.

Altschul, SF, Madden, TL, Schaffer, AA, Zhang, J, Zhang, Z, Miller, W, and DJ Lipman (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. **25**(17):3389-402.

Claverie, JM, and DJ States (1993). Information enhancement methods for large scale sequence analysis. Computers in Chemistry **17**:191-201.

Gish, W, and DJ States (1993). Identification of protein coding regions by database similarity search. Nature Genetics **3**:266-72.

Hancock, JM, and JS Armstrong (1994). SIMPLE34: an improved and enhanced implementation for VAX and Sun computers of the SIMPLE algorithm for analysis of clustered repetitive motifs in nucleotide sequences. Comput. Appl. Biosci. **10**:67-70.

Karlin, S, and SF Altschul (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. Proc. Natl. Acad. Sci. USA **87**(6):2264-8.

Karlin, S, and SF Altschul (1993). Applications and statistics for multiple high-scoring segments in molecular sequences. Proc. Natl. Acad. Sci. **90**:5873-7.

Pearson, WR, and DJ Lipman (1988). Improved tools for biological sequence comparison. Proc. Natl. Acad. Sci. **85**(8):2444-8.

Smith, TF, and MS Waterman (1981). Identification of common molecular subsequences. J. Mol. Biol. **147**:195-7.

States, DJ, and W Gish (1994). Combined use of sequence similarity and codon bias for coding region identification. J. Comp. Biol. **1**:39-50.

Wootton, JC, and S Federhen (1993). Statistics of local complexity in amino acid sequences and sequence databases. Computers in Chemistry **17**:149-63.

Wootton, JC, and S Federhen (1996). Analysis of compositionally biased regions in sequence databases. ed. R. Doolittle. Methods in Enzymology **266**:554-71.

Zhang, Z, Schaffer, AA, Miller, W, Madden, TL, Lipman, DJ, Koonin, EV, and SF Altschul (1998). Protein sequence similarity searches using patterns as seeds. Nucleic Acids Res. **26**:3986-90.

Return to the *WU BLAST Archives* home page