



BLAST Command Line Applications User Manual

Christiam Camacho*NCBI*

camacho@ncbi.nlm.nih.gov

Thomas Madden*NCBI*

madden@ncbi.nlm.nih.gov

George Coulouris*NCBI*

coulouri@ncbi.nlm.nih.gov

Vahram Avagyan*NCBI*

avagyanv@ncbi.nlm.nih.gov

Ning Ma*NCBI*

maning@ncbi.nlm.nih.gov

Tao Tao*NCBI*

tao@ncbi.nlm.nih.gov

Richa Agarwala*NCBI*

richa@ncbi.nlm.nih.gov

1. Introduction

This manual documents the BLAST (Basic Local Alignment Search Tool) command line applications developed at the National Center for Biotechnology Information (NCBI). These applications have been revamped to provide an improved user interface, new features, and performance improvements compared to its counterparts in the NCBI C Toolkit. Hereafter we shall distinguish the C Toolkit BLAST command line applications from these command line applications by referring to the latter as the BLAST+ applications, which have been developed using the NCBI C++ Toolkit (<http://www.ncbi.nlm.nih.gov/books/bv.fcgi?rid=toolkit.TOC&depth=2>).

Please feel free to contact us with any questions, feedback, or bug reports at blast-help@ncbi.nlm.nih.gov.

2. Installation

The BLAST+ applications are distributed in executable and source code format. For the executable formats we provide installers as well as tarballs; the source code is only provided as a tarball. These are freely available at [ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/.](ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/) Please be sure to use the most recent available version; this will be indicated in the file name (for instance, in the sections below, version 2.2.18 is listed, but this should be replaced accordingly).

2.1 Windows

Download the executable installer `ncbi-blast-2.2.18+.exe` and double click on it. After accepting the license agreement, select the install location and click “Install” and then “Close”

2.2 MacOSX

For users without administrator privileges: Download the `ncbi-blast-2.2.18+-universal-macosx.tar.gz` tarball and follow the procedure described in Other Unix platforms.

For users with administrator privileges and machines MacOSX version 10.5 or higher: Download the `ncbi-blast-2.2.18+.dmg` installer and double click on it. Double click the newly mounted `ncbi-blast-2.2.18+` volume, double click on `ncbi-blast-2.2.18+.pkg` and follow the instructions in the installer. By default the BLAST+ applications are installed in `/usr/local/ncbi/blast`, overwriting its previous contents (an uninstaller is provided and it is recommended when upgrading a BLAST+ installation).

2.3 RedHat Linux

Download the appropriate `*.rpm` file for your platform and either install or upgrade the `ncbi-blast+` package as appropriate using the commands:

```
Install:
rpm -ivh ncbi-blast-2.2.18-1.x86_64.rpm
Upgrade:
rpm -Uvh ncbi-blast-2.2.18-1.x86_64.rpm
```

Note: one must have root privileges to run these commands. If you do not have root privileges, please use the procedure described in Other Unix platforms.

2.4 Other Unix platforms

Download the tarball and expand it in the location of your choice.

2.5 Source tarball

Use this approach if you would like to build the BLAST+ applications yourself. Download the tarball, expand it and in the expanded directory type the following commands:

```
cd c++
./configure --without-debug --with-mt --with-build-root=ReleaseMT
cd ReleaseMT/build
make all_r
```

The compiled executables will be found in `c++/ReleaseMT/bin`.

In Windows, extract the tarball and open the appropriate MSVC solution or project file (e.g.: `c++\compilers\msvc800_prj\static\build`), build the `-CONFIGURE-` project, click on “Reload” when prompted by the development environment, and then build the `-BUILD-ALL-` project. The compiled executables will be found in the directory corresponding to the build configuration selected (e.g.: `c++\compilers\msvc800_prj\static\bin\debugdll`).

3. Quick start

3.1 For users of NCBI C Toolkit BLAST

The easiest way to get started using these command line applications is by means of the `legacy_blast.pl` PERL script which is bundled along with the BLAST+ applications. To utilize this script, simply prefix it to the invocation of the C toolkit BLAST command line application and append the `--path` option pointing to the installation directory of the BLAST+ applications. For example, instead of using

```
blastall -i query -d nr -o blast.out
```

use

```
legacy_blast.pl blastall -i query -d nr -o blast.out
--path /opt/blast/bin
```

For more details, refer to the section titled Backwards compatibility script.

3.2 For users of Web BLAST (<http://blast.ncbi.nlm.nih.gov>)

Users of Web BLAST can take advantage of the search strategies to quickly get started using the BLAST+ applications, as these intend to allow seamless integration between the Web and command line BLAST tools. For more details, refer to the section on BLAST search strategies.

3.3 For new users of BLAST

An introduction to BLAST is outside the scope of this manual, more information on this subject can be found on http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs. Nonetheless, new users will benefit from the examples in the cookbook as well as reading the user manual.

3.4 Downloading BLAST databases

The BLAST databases are required to run BLAST locally and to support automatic resolution of sequence identifiers. Documentation about these can be found <ftp://ftp.ncbi.nlm.nih.gov/blast/db/README>. These databases may be retrieved automatically with the `update_blastdb.pl` perl script, which is included as part of this distribution.

This script will download multiple tar files for each BLAST database volume if necessary, without having to designate each volume. For example:

```
./update_blastdb.pl htgs
```

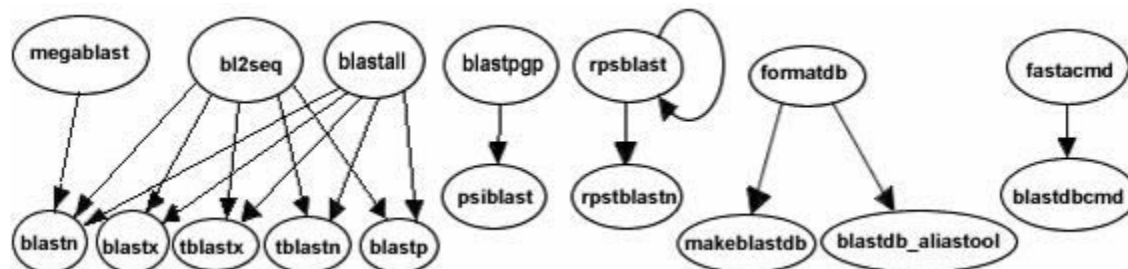
will download all the relevant HTGs tar files (`htgs.00.tar.gz`, ..., `htgs.N.tar.gz`)

The script can also compare your local copy of the database tar file(s) and only download tar files if the date stamp has changed reflecting a newer version of the database. This will allow the script run on a schedule and only download tar files when needed. Documentation for the `update_blastdb.pl` script can be obtained by running the script without any arguments (perl is required).

4. User manual

4.1 Functionality offered by BLAST+ applications

The functionality offered by the BLAST+ applications has been organized by program type, as to more closely resemble Web BLAST. The following graph depicts a correspondence between the NCBI C Toolkit BLAST command line applications and the BLAST+ applications:



As an example, to run a search of a nucleotide query (translated “on the fly” by BLAST) against a protein database one would use the blastx application instead of blastall. The blastx application will also work in “Blast2Sequences” mode (i.e.: accept FASTA sequences instead of a BLAST database as targets) and can also send BLAST searches over the network to the public NCBI server if desired.

The blastn, blastp, blastx, tblastx, tblastn, psiblast, rpsblast, and rpstblastn are considered search applications, as they execute a BLAST search, whereas makeblastdb, blastdb_aliastool, and blastdbcmd are considered BLAST database applications, as they either create or examine BLAST databases.

There is also a new set of sequence filtering applications described in the section Sequence filtering applications and an application to build database indices that greatly speed up megablast in some cases (see section titled Megablast indexed searches).

Please note that the NCBI C Toolkit applications seedtop and blastclust are not available in this release.

4.2 Common options

The following is a listing of options that are common to the majority of BLAST+ applications followed by a brief description of what they do:

4.2.1 best_hit_overhang: Overhang value for Best-Hit algorithm. For more details, see the section Best-Hits filtering algorithm.

4.2.2 best_hit_score_edge: Score edge value for Best-Hit algorithm. For more details, see the section Best-Hits filtering algorithm.

4.2.3 db: File name of BLAST database to search the query against. Unless an absolute path is used, the database will be searched relative to the current working directory first, then relative to the value specified by the BLASTDB environment variable, then relative to the BLASTDB configuration value specified in the configuration file.

4.2.4 dbsize: Effective length of the database.

4.2.5 dbtype: Molecule type stored or to store in a BLAST database.

4.2.6 db_soft_mask: Filtering algorithm ID to apply to the database as soft masking for subject sequences. The algorithm IDs for a given BLAST database can be obtained by invoking `blastdbcmd` with its `-info` flag (only shown if such filtering in the BLAST database is available). For more details see the section Masking in BLAST databases.

4.2.7 culling_limit: Ensures that more than the specified number of HSPs are not aligned to the same part of the query. This option was designed for searches with a lot of repetitive matches, but if possible it is probably more efficient to mask the query to remove the repetitive sequences.

4.2.8 entrez_query: Restrict the search of the BLAST database to the results of the Entrez query provided.

4.2.9 evalue: Expectation value threshold for saving hits.

4.2.10 export_search_strategy: Name of the file where to save the search strategy (see section titled BLAST search strategies).

4.2.11 gapextend: Cost to extend a gap.

4.2.12 gapopen: Cost to open a gap.

4.2.13 gilist: File containing a list of GIs to restrict the BLAST database to search. The expect values in the BLAST results are based upon the sequences actually searched and not on the underlying database.

4.2.14 h: Displays the application's brief documentation.

4.2.15 help: Displays the application's detailed documentation.

4.2.16 html: Enables the generation of HTML output suitable for viewing in a web browser.

4.2.17 import_search_strategy: Name of the file where to read the search strategy to execute (see section titled BLAST search strategies).

4.2.18 lcase_masking: Interpret lowercase letters in query sequence(s) as masked.

4.2.19 matrix: Name of the scoring matrix to use.

4.2.20 max_target_seqs: Maximum number of aligned sequences to keep from the BLAST database.

4.2.21 negative_gilist: File containing a list of GIs to exclude from the BLAST database.

4.2.22 num_alignments: Number of alignments to show in the BLAST output.

4.2.23 num_descriptions: Number of one-line descriptions to show in the BLAST output.

4.2.24 num_threads: Number of threads to use during the search.

4.2.25 out: Name of the file to write the application's output. Defaults to `stdout`.

4.2.26 outfmt: Allows for the specification of the search application's output format. A listing of the possible format types is available via the search application's `-help` option. If a custom output format is desired, this can be specified by providing a quoted string composed of the desired output format (tabular, tabular with comments, or comma-separated value), a space,

and a space delimited list of output specifiers. The list of supported output specifiers is available via the `-help` command line option. Unsupported output specifiers will be ignored. This should be specified using double quotes if there are spaces in the output format specification (e.g.: `-outfmt "7 sseqid ssac qstart qend sstart send qseq evaluate bitscore"`).

4.2.27 parse_deflines: Parse the query and subject deflines.

4.2.28 query: Name of the file containing the query sequence(s), or '-' if these are provided on standard input.

4.2.29 query_loc: Location of the first query sequence to search (Format: start-stop).

4.2.30 remote: Instructs the application to submit the search to NCBI for remote execution.

4.2.31 searchsp: Effective length of the search space.

4.2.32 seg: Arguments to SEG filtering algorithm (use 'no' to disable).

4.2.33 show_gis: Show NCBI GIs in deflines in the BLAST output.

4.2.34 soft_masking: Apply filtering locations as soft masks (i.e.: only when finding alignment seeds).

4.2.35 subject: Name of the file containing the subject sequence(s) to search.

4.2.36 subject_loc: Location of the first subject sequence to search (Format: start-stop).

4.2.37 strand: Strand(s) of the query sequence to search.

4.2.38 threshold: Minimum word score such that the word is added to the BLAST lookup table.

4.2.39 ungapped: Perform ungapped alignments only.

4.2.40 version: Displays the application's version.

4.2.41 window_size: Size of the window for multiple hits algorithm, use 0 to specify 1-hit algorithm.

4.2.42 word_size: Word size for word finder algorithm.

4.2.43 xdrop_gap: X-dropoff value (in bits) for preliminary gapped extensions.

4.2.44 xdrop_gap_final: X-dropoff value (in bits) for final gapped alignment.

4.2.45 xdrop_ungap: X-dropoff value (in bits) for ungapped extensions.

4.3 Backwards compatibility script

The purpose of the `legacy_blast.pl` Perl script is to help users make the transition from the C Toolkit BLAST command line applications to the BLAST+ applications. This script produces its own documentation by invoking it without any arguments.

The `legacy_blast.pl` script supports two modes of operation, one in which the C Toolkit BLAST command line invocation is converted and executed on behalf of the user and another which

solely displays the BLAST+ application equivalent to what was provided, without executing the command.

The first mode of operation is achieved by specifying the C Toolkit BLAST command line application invocation and optionally providing the `--path` argument after the command line to convert if the installation path for the BLAST+ applications differs from the default (available by invoking the script without arguments). See example in the first section of the Quick start.

The second mode of operation is achieved by specifying the C Toolkit BLAST command line application invocation and appending the `--print_only` command line option as follows:

```
$ ./legacy_blast.pl megablast -i query.fsa -d nt -o mb.out --print_only
/opt/ncbi/blast/bin/blastn -query query.fsa -db "nt" -out mb.out
$
```

4.4 Exit codes

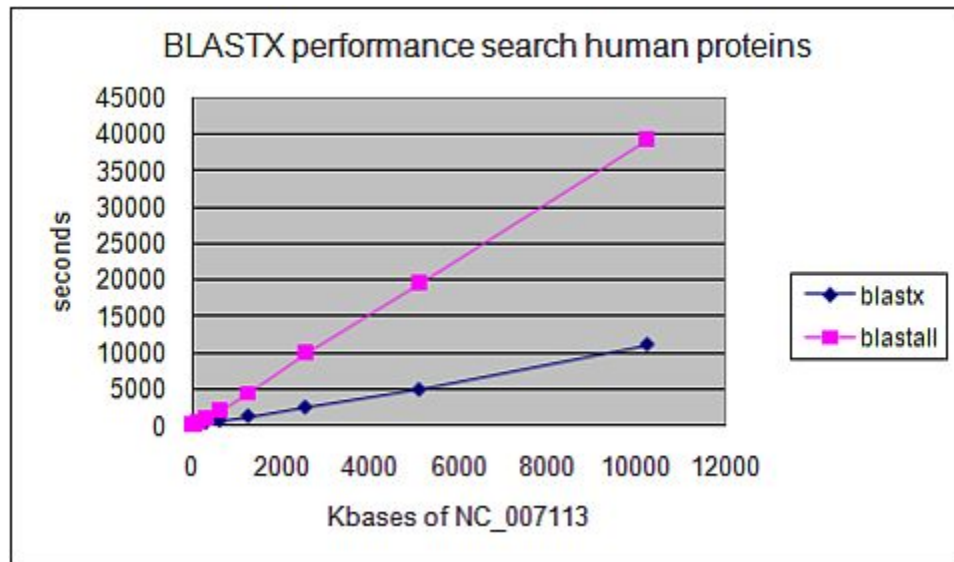
All BLAST+ applications have consistent exit codes to signify the exit status of the application. The possible exit codes along with their meaning are detailed in the table below:

Exit Code	Meaning
0	Success
1	Error in query sequence(s) or BLAST options
2	Error in BLAST database
3	Error in BLAST engine
255	Unknown error

In the case of BLAST+ database applications, the possible exit codes are 0 (indicating success) and 1 (indicating failure).

4.5 Improvements over C Toolkit BLAST command line applications

4.5.1 Query splitting—This new feature in the BLAST+ applications provides substantial performance improvements, particularly for `blastx` searches. Below is a graph comparing the runtime of `blastall` and `blastx` when searching different size excerpts of NC_007113 (varying from 10 kbases to about 10 Mbases) against the human genome database (experiments performed in July 2008):



4.5.2 Tasks— The concept of tasks has been added to support the notion of commonly performed tasks via the `-task` command line option in `blastn` and `blastp`. The following tasks are currently available:

Program	Task Name	Description
blastp	blastp	Traditional BLASTP to compare a protein query to a protein database
	blastp-short	BLASTP optimized for queries shorter than 30 residues
blastn	blastn	Traditional BLASTN requiring an exact match of 11
	blastn-short	BLASTN program optimized for sequences shorter than 50 bases
	megablast	Traditional megablast used to find very similar (e.g., intraspecies or closely related species) sequences
	dc-megablast	Discontiguous megablast used to find more distant (e.g., interspecies) sequences

4.5.3 Megablast indexed searches—Indexed searches for megablast are available and are faster than regular megablast. The application to generate the database indices is called `makemindex`, which is included in this distribution. More information about it can be found at ftp://ftp.ncbi.nlm.nih.gov/pub/agarwala/indexed_megablast/README.usage.

4.5.4 Partial sequence retrieval from BLAST databases—Improvements to the BLAST database reading module allow it to fetch only the relevant portions of the subject sequence that are needed in the gapped alignment stage, providing a substantial improvement in runtime. The following example compares 103 mouse EST sequences against the human genome shows (example run in July 2008 after the database had already been loaded into memory):

```
$ time megablast -d 9606_genomic -i est.500.in -r 2 -q -3
-F mL -m 9 -o old.out -W 11 -t 18 -G 5 -E 2
341.455u 65.242s 6:47.99 99.6% 0+0k 0+0io 0pf+0w
$ time blastn -task dc-megablast -db 9606_genomic -query
est.500.in -outfmt 7 -out new.out
218.540u 11.632s 3:50.53 99.8% 0+0k 0+0io 0pf+0w
```


Similar gains in performance should be expected in BLAST databases which contain very large sequences and many very short queries.

4.5.5 BLAST search strategies—BLAST search strategies are files which encode the inputs necessary to perform a BLAST search. The purpose of these files is to be able to seamlessly reproduce a BLAST search in various environments (Web BLAST, command line applications, etc).

4.5.5.1 Exporting search strategies on the Web BLAST

Click on "download" next to the RID/saved strategy in the "Recent Results" or "Saved Strategies" tabs.

4.5.5.2 Exporting search strategies with BLAST+ applications

Add the `-export_search_strategy` along with a file name to the command line options.

4.5.5.3 Importing search strategies on Web BLAST

Go to the "Saved Strategies" tab, click on "Browse" to select your search strategy file, then click on "View" to load it into the submission page.

4.5.5.4 Importing search strategies with BLAST+ applications

Add the `-import_search_strategy` along with a file name containing the search strategy file. Note that if provided, the `-query`, `-db`, `-use_index`, and `-index_name` command line options will override the specifications of the search strategy file provided (no other command line options will override the contents of the search strategy file).

4.5.6 Negative GI lists—Negative GI lists are available on search applications and they provide a means to exclude GIs from a BLAST database search. The expect values in the BLAST results are based upon the sequences actually searched and not on the underlying database. For an example, see the cookbook.

4.5.7 Masking in BLAST databases —It is now possible to create BLAST databases which contain filtered sequences (also known as masking information or masks). This filtering information can be used as soft masking for the subject sequences. For instructions on creating masked BLAST databases, please see the cookbook.

4.5.8 Custom output formats for BLAST searches—The BLAST+ search command line applications support custom output formats for the tabular and comma-separated value output formats. For more details see the common options as well as the cookbook.

4.5.9 Custom output formats to extract BLAST database data—`blastdbcmd` supports custom output formats to extract data from BLAST databases via the `-outfmt` command line option. For more details see the `blastdbcmd` options as well as the cookbook.

4.5.10 Improved software installation packages—The BLAST+ applications are available via Windows and MacOSX installers as well as RPMs (source and binary) and unix tarballs. For more details about these, refer to the installation section.

4.5.11 Sequence filtering applications—The BLAST+ applications include a new set of sequence filtering applications, namely `segmasker`, `dustmasker`, and `windowmasker`. `segmasker` is an application that identifies and masks low complexity regions of protein

sequences. The dustmasker and windowmasker applications provide similar functionality for nucleotide sequences (see <ftp://ftp.ncbi.nlm.nih.gov/pub/agarwala/dustmasker/README.dustmasker> and <ftp://ftp.ncbi.nlm.nih.gov/pub/agarwala/windowmasker/README.windowmasker> for more information).

4.5.12 Best-Hits filtering algorithm—The Best-Hit filtering algorithm is designed for use in applications that are searching for only the best matches for each query region reporting matches. Its `-best_hit_overhang` parameter, `H`, controls when an HSP is considered short enough to be filtered due to presence of another HSP. For each HSP A that is filtered, there exists another HSP B such that the query region of HSP A extends each end of the query region of HSP B by at most `H` times the length of the query region for B.

Additional requirements that must also be met in order to filter A on account of B are:

- i `value(A) >= value(B)`
- ii `score(A)/length(A) < (1.0 - score_edge) * score(B)/length(B)`

We consider 0.1 to 0.25 to be an acceptable range for the `-best_hit_overhang` parameter and 0.05 to 0.25 to be an acceptable range for the `-best_hit_score_edge` parameter. Increasing the value of the overhang parameter eliminates a higher number of matches, but increases the running time; increasing the `score_edge` parameter removes smaller number of hits.

4.5.13 Automatic resolution of sequence identifiers—The BLAST+ search applications support automatic resolution of query and subject sequence identifiers specified as GIs or accessions (see the cookbook section for an example). This feature enables the user to specify one or more sequence identifiers (GIs and/or accessions, one per line) in a file as the input to the `-query` and `-subject` command line options.

Upon encountering this type of input, by default the BLAST+ search applications will try to resolve these sequence identifiers in locally available BLAST databases first, then in the BLAST databases at NCBI, and finally in Genbank (the latter two data sources require a properly configured internet connection). These data sources can be configured via the `DATA_LOADERS` configuration option and the BLAST databases to search can be configured via the `BLASTDB_PROT_DATA_LOADER` and `BLASTDB_NUCL_DATA_LOADER` configuration options (see the section on Configuring BLAST).

4.6 Options by program type

4.6.1 blastp—*4.6.1.1 task*: Specify the task to execute. For more details, refer to the section on tasks.

4.6.1.2 comp_based_stats: Select the appropriate composition based statistics mode (applicable only to `blastp` and `tblastn`). Available choices and references are available by invoking the application with `-help` option.

4.6.1.3 use_sw_tback: Instead of using the X-dropoff gapped alignment algorithm, use Smith-Waterman to compute locally optimal alignments

4.6.2 blastn—*4.6.2.1 task*: Specify the task to execute. For more details, refer to the section on tasks.

4.6.2.2 penalty: Penalty for a nucleotide mismatch.

4.6.2.3 reward: Reward for a nucleotide match.

- 4.6.2.4 *use_index*: Use a megablast database index.
- 4.6.2.5 *index_name*: Name of the megablast database index.
- 4.6.2.6 *perc_identity*: Minimum percent identity of matches to report
- 4.6.2.7 *dust*: Arguments to DUST filtering algorithm (use 'no' to disable).
- 4.6.2.8 *filtering_db*: Name of BLAST database containing filtering elements (i.e.: repeats)
- 4.6.2.9 *window_masker_taxid*: Enable WindowMasker filtering using a Taxonomic ID [experimental]
- 4.6.2.10 *window_masker_db*: Enable WindowMasker filtering using this repeats database. [experimental]
- 4.6.2.11 *no_greedy*: Use non-greedy dynamic programming extension.
- 4.6.2.12 *min_raw_gapped_score*: Minimum raw gapped score to keep an alignment in the preliminary gapped and traceback stages.
- 4.6.2.13 *template_type*: Discontiguous megablast template type.
- 4.6.2.14 *template_length*: Discontiguous megablast template length.
- 4.6.3 blastx**—4.6.3.1 *query_gencode*: Genetic code to use to translate the query sequence(s).
- 4.6.3.2 *frame_shift_penalty*: Frame shift penalty for use with out-of-frame gapped alignments
- 4.6.3.3 *max_intron_length*: Length of the largest intron allowed in a translated nucleotide sequence when linking multiple distinct alignments (a negative value disables linking).
- 4.6.4 tblastx**—4.6.4.1 *db_gencode*: Genetic code to use to translate database/subjects.
- 4.6.4.2 *max_intron_length*: Identical to blastx.
- 4.6.5 tblastn**—4.6.5.1 *db_gencode*: Identical to tblastx.
- 4.6.5.2 *frame_shift_penalty*: Identical to blastx.
- 4.6.5.3 *max_intron_length*: Identical to blastx.
- 4.6.5.4 *comp_based_stats*: Identical to blastp.
- 4.6.5.5 *use_sw_tback*: Identical to blastp.
- 4.6.5.6 *in_pssm*: Checkpoint file to initiate PSI-TBLASTN (currently unimplemented).
- 4.6.6 psiblast**—4.6.6.1 *comp_based_stats*: Identical to blastp with the exception that only composition based statistics mode 1 is valid when a PSSM is the input (either when restarting from a checkpoint file or when performing multiple PSI-BLAST iterations).
- 4.6.6.2 *gap_trigger*: Number of bits to trigger gapping.
- 4.6.6.3 *use_sw_tback*: Identical to blastp.

- 4.6.6.4 *num_iterations*: Number of iterations to perform.
- 4.6.6.5 *out_pssm*: Name of the file to store checkpoint file containing a PSSM.
- 4.6.6.6 *out_ascii_pssm*: Name of the file to store ASCII version of PSSM.
- 4.6.6.7 *in_msa*: Name of the file containing multiple sequence alignment to restart PSI-BLAST.
- 4.6.6.8 *in_pssm*: Checkpoint file to re-start PSI-BLAST.
- 4.6.6.9 *pseudocount*: Pseudo-count value used when constructing the PSSM.
- 4.6.6.10 *inclusion_ethresh*: E-value inclusion threshold for pairwise alignments to be considered to build the PSSM.
- 4.6.6.11 *phi_pattern*: Name of the file containing a PHI-BLAST pattern to search.

4.6.7 rpstblastn—4.6.7.1 *query_gencode*: Identical to blastx.

4.6.8 makeblastdb—This application serves as a replacement for formatdb.

4.6.8.1 *in*: Input file or BLAST database name to use as source; the data type is automatically detected. Note that multiple input files/BLAST databases can be provided, each must be separated by white space in a string quoted with single quotation marks. Multiple input files/BLAST databases which contain white space in them should be quoted with double quotation marks inside the white space-separated, single quoted string (e.g.: `-in 'C:\My Documents\seqs.fsa' "E:\Users\Joe Smith\myfasta.fsa"`).

4.6.8.2 *title*: Title for the BLAST database to create

4.6.8.3 *parse_seqids*: Parse the Seq-id(s) in the FASTA input provided. Please note that this option should be provided consistently among the various applications involved in creating BLAST databases. For instance, the filtering applications as well as `convert2blastmask` should use this option if `makeblastdb` uses it also.

4.6.8.4 *hash_index*: Enables the creation of sequence hash values. These hash values can then be used to quickly determine if a given sequence data exists in this BLAST database.

4.6.8.5 *mask_data*: Comma-separated list of input files containing masking data to apply to the sequences being added to the BLAST database being created. For more information, see Masking in BLAST databases and the examples.

4.6.8.6 *out*: Name of the BLAST database to create.

4.6.8.7 *max_file_sz*: Maximum file size for any of the BLAST database files created.

4.6.8.8 *logfile*: Name of the file to which the program log should be redirected (stdout by default).

4.6.8.9 *taxid*: Taxonomy ID to assign to all sequences.

4.6.8.10 *taxid_map*: Name of file which provides a mapping of sequence IDs to taxonomy IDs.

4.6.9 blastdb_aliastool—This application replaces part of the functionality offered by `formatdb`. When formatting a large input FASTA sequence file into a BLAST database,

makeblastdb breaks up the resulting database into optimal sized volumes and links the volumes into a large virtual database through an automatically created BLAST database alias file.

We can use BLAST database alias files under different scenarios to manage the collection of BLAST databases and facilitate BLAST searches. For example, we can create an alias file to combine an existing BLAST database with newly generated ones while leaving the original one undisturbed. Also, for an existing BLAST database, we can create a BLAST database alias file based on a GI list so we can search a subset of it, eliminating the need of creating a new database. For examples of how to use this application, please see the cookbook section.

This application supports three modes of operation:

1) Gi file conversion:

Converts a text file containing GIs (one per line) to a more efficient binary format. This can be provided as an argument to the `-gilist` option of the BLAST search command line binaries or to the `-gilist` option of this program to create an alias file for a BLAST database (see below).

2) Alias file creation (restricting with GI List):

Creates an alias for a BLAST database and a GI list which restricts this database. This is useful if one often searches a subset of a database (e.g., based on organism or a curated list). The alias file makes the search appear as if one were searching a regular BLAST database rather than the subset of one.

3) Alias file creation (aggregating BLAST databases):

Creates an alias for multiple BLAST databases. All databases must be of the same molecule type (no validation is done). The relevant options are `-dblist` and `-num_volumes`.

4.6.9.1 gi_file_in: Text file to convert, should contain one GI per line.

4.6.9.2 gi_file_out: File name of converted GI file

4.6.9.3 title: Title for BLAST database.

4.6.9.4 gilist: Name of the file containing the GIs to restrict the database provided in `-db`.

4.6.9.5 out: Identical to `makeblastdb`.

4.6.9.6 logfile: Identical to `makeblastdb`.

Please note that when using GI lists, the expect values in the BLAST results are based upon the sequences actually searched and not on the underlying database.

4.6.10 blastdbcmd—This application is the successor to fastacmd. The following are its supported options:

4.6.10.1 entry: A comma-delimited search string of sequence identifiers, or the keyword ‘all’ to select all sequences in the database.

4.6.10.2 entry_batch: Input file for batch processing, entries must be provided one per line. If input is provided on standard input, a ‘-’ should be used to indicate this.

4.6.10.3 pig: PIG (Protein Identity Group) to retrieve.

4.6.10.4 info: Print BLAST database information (overrides all other options).

4.6.10.5 range: Selects the range of a sequence to extract.

4.6.10.6 strand: Strand of nucleotide sequence to extract.

4.6.10.7 outfmt: Output format string. For a list of available format specifiers, invoke the application with its -help option. Note that for all format specifiers except %f, each line of output will correspond to a single sequence. This should be specified using double quotes if there are spaces in the output format specification (e.g.: -outfmt "%g %t").

4.6.10.8 target_only: The definition line of the sequence should contain target GI only.

4.6.10.9 get_dups: Retrieve duplicate accessions

4.6.10.10 line_length: Line length for output (applicable only with FASTA output format).

4.6.10.11 ctrl_a: Use Ctrl-A as the non-redundant define separator (applicable only with FASTA output format).

4.6.10.12 mask_sequence_with: Allows the specification of filtering algorithm IDs from the BLAST database to apply to the sequence data. Applicable only with FASTA and sequence data output formats (%f and %s respectively).

4.6.11 convert2blastmask—This application extracts the lower-case masks from its FASTA input and converts them to a file format suitable for specifying masking information to makeblastdb. The following are its supported options:

4.6.11.1 masking_algorithm: The name of the masking algorithm used to create the masks (e.g.: dust, seg, windowmasker, repeat).

4.6.11.2 masking_options: The options used to configure the masking algorithm.

4.6.11.3 in: Name of the input file, by default is standard input.

4.6.11.4 output: Name of the output file, by default is standard output.

4.6.11.5 outfmt: Output file format.

4.6.11.6 parse_seqids: Identical to makeblastdb.

4.6.12 blastdbcheck—This application performs tests on BLAST databases to check their integrity. The following are its supported options:

4.6.12.1 dir: Name of the directory where to look for BLAST databases.

4.6.12.2 *recursive*: Flag to specify whether to recursively search for BLAST databases in the directory specified above.

4.6.12.3 *full*: Check every database sequence.

4.6.12.4 *stride*: Check every Nth database sequence.

4.6.12.5 *samples*: Check a randomly selected set of N sequences.

4.6.12.6 *ends*: Check the beginning and ending N sequences in the database.

4.6.12.7 *isam*: Set to true to perform ISAM file checking on each of the selected sequences.

4.7 Configuring BLAST

The BLAST+ search applications can be configured by means of a configuration file named `.ncbirc` (on Unix-like platforms) or `ncbi.ini` (on Windows). This is a plain text file which contains sections and key-value pairs to specify configuration parameters. Lines starting with a semi-colon are considered comments. This file will be searched in the following order and locations:

- 1 Current working directory
- 2 User's HOME directory
- 3 Directory specified by the NCBI environment variable

The search for this file will stop at the first location where it is found and the configurations settings from that file will be applied. If the configuration file is not found, default values will apply. The following are the possible configuration parameters that impact the BLAST+ applications:

Configuration Parameter	Specifies	Default value
BLASTDB	Path to BLAST databases.	Current working directory
DATA_LOADERS	Data loaders to use for automatic sequence identifier resolution. This is a comma separated list of the following keywords: <code>blastdb</code> , <code>genbank</code> , and <code>none</code> . The <code>none</code> keyword disables this feature and takes precedence over any other keywords specified.	<code>blastdb,genbank</code>
BLASTDB_PROT_DATA_LOADER	Locally available BLAST database name to search when resolving protein sequences using BLAST databases. Ignored if <code>DATA_LOADERS</code> does not include the <code>blastdb</code> keyword.	<code>nr</code>
BLASTDB_NUCL_DATA_LOADER	Locally available BLAST database name to search when resolving nucleotide sequences using BLAST databases. Ignored if <code>DATA_LOADERS</code> does not include the <code>blastdb</code> keyword.	<code>nt</code>
GENE_INFO_PATH	Path to gene information files (NCBI only).	Current working directory
WINDOW_MASKER_PATH	Path to windowmasker files (experimental).	Current working directory

The following is an example with comments describing the available parameters for configuration:

```
; Start the section for BLAST configuration
[BLAST]
; Specifies the path where BLAST databases are installed
BLASTDB=/home/guest/blast/db
; Specifies the data sources to use for automatic resolution
; for sequence identifiers
DATA_LOADERS=blastdb
```



```

; Specifies the BLAST database to use resolve protein sequences
BLASTDB_PROT_DATA_LOADER=custom_protein_database
; Specifies the BLAST database to use resolve protein sequences
BLASTDB_NUCL_DATA_LOADER=/home/some_user/my_nucleotide_db
; Windowmasker settings (experimental)
[WINDOW_MASKER]
WINDOW_MASKER_PATH=/home/guest/blast/db/windowmasker
; end of file

```

4.7.1 Memory usage—The BLAST search programs can exhaust all memory on a machine if the input is too large or if there are too many hits to the BLAST database. If this is the case, please see your operating system documentation to limit the memory used by a program (e.g.: ulimit on Unix-like platforms).

5. Cookbook

5.1 Query a BLAST database with a GI, but exclude that GI from the results

```

Extract a GI from the ecoli database:
$ blastdbcmd -entry all -db ecoli -dbtype nucl -outfmt %g | head -1 | \
tee exclude_me
1786181
Run the restricted database search, which shows there are no self-hits:
$ blastn -db ecoli -negative_gilist exclude_me -show_gis -num_alignments 0 \
-query exclude_me | grep `cat exclude_me`
Query= gi|1786181|gb|AE000111.1|AE000111
$

```

5.2 Create a masked BLAST database

Creating a masked BLAST database is a two step process:

- a Generate the masking data using a sequence filtering utility like windowmasker or dustmasker
- b Generate the actual BLAST database using makeblastdb

For both steps, the input file can be a text file containing sequences in FASTA format, or an existing BLAST database created using makeblastdb. We will provide examples for both scenarios.

5.2.1 Collect mask information files—For nucleotide sequence data in FASTA files or BLAST database format, we can generate the mask information files using windowmasker or dustmasker. windowmasker masks the over-represented sequence data and it can also mask the low complexity sequence data using the built-in dust algorithm (through the -dust option). To mask low-complexity sequences only, we will need to use dustmasker.

For protein sequence data in FASTA files or BLAST database format, we need to use segmasker to generate the mask information file.

The following examples assume that BLAST databases, listed in 5.2.3, are available in the current working directory. Note that you should use the sequence id parsing consistently. In all our examples, we enable this function by including the “-parse_seqids” in the command line arguments.

5.2.1.1 Create masking information using dustmasker: We can generate the masking information with dustmasker using a single command line:

```
$ dustmasker -in hs_chr -infmt blastdb -parse_seqids \
  -outfmt maskinfo_asn1_bin -out hs_chr_dust.asnb
```

Here we specify the input is a BLAST database named `hs_chr` (`-in hs_chr -infmt blastdb`), enable the sequence id parsing (`-parse_seqids`), request the mask data in binary `asn.1` format (`-outfmt maskinfo_asn1_bin`), and name the output file as `hs_chr_dust.asnb` (`-out hs_chr_dust.asnb`).

If the input format is the original FASTA file, `hs_chr.fa`, we need to change input to `-in` and `-infmt` options as follows:

```
$ dustmasker -in hs_chr.fa -infmt blastdb -parse_seqids \
  -outfmt maskinfo_asn1_bin -out hs_chr_dust.asnb
```

5.2.1.2 Create masking information using windowmasker: To generate the masking information using windowmasker from the BLAST database `hs_chr`, we first need to generate a counts file:

```
$ windowmasker -in hs_chr -infmt blastdb -mk_counts \
  -parse_seqids -out hs_chr_mask.counts
```

Here we specify the input BLAST database (`-in hs_chr -infmt blastdb`), request it to generate the counts (`-mk_counts`) with sequence id parsing (`-parse_seqids`), and save the output to a file named `hs_chr_mask.counts` (`-out hs_chr_mask.counts`).

To use the FASTA file `hs_chr.fa` to generate the counts, we need to change the input file name and format:

```
$ windowmasker -in hs_chr.fa -infmt fasta -mk_counts \
  -parse_seqids -out hs_chr_mask.counts
```

With the counts file we can then proceed to create the file containing the masking information as follows:

```
$ windowmasker -in hs_chr -infmt blastdb -ustat hs_chr_mask.count \
  -outfmt maskinfo_asn1_bin -parse_seqids -out hs_chr_mask.asnb
```

Here we need to use the same input (`-in hs_chr -infmt blastdb`) and the output of step 1 (`-ustat hs_chr_mask.counts`). We set the mask file format to binary `asn.1` (`-outfmt maskinfo_asn1_bin`), enable the sequence ids parsing (`-parse_seqids`), and save the masking data to `hs_chr_mask.asnb` (`-out hs_chr_mask.asnb`).

To use the FASTA file `hs_chr.fa`, we change the input file name and file type:

```
$ windowmasker -in hs_chr.fa -infmt fasta -ustat hs_chr.counts \
  -outfmt maskinfo_asn1_bin -parse_seqids -out hs_chr_mask.asnb
```

5.2.1.3 Create masking information using segmasker: We can generate the masking information with segmasker using a single command line:

```
$ segmasker -in refseq_protein -infmt blastdb -parse_seqids \
  -outfmt maskinfo_asn1_bin -out refseq_seg.asnb
```

Here we specify the refseq_protein BLAST database (-in refseq_protein -infmt blastdb), enable sequence ids parsing (-parse_seqids), request the mask data in binary asn.1 format (-outfmt maskinfo_asn1_bin), and name the out file as refseq_seg.asnb (-out refseq_seg.asnb).

If the input format is the FASTA file, we need to change the command line to specify the input format:

```
$ segmasker -in refseq_protein.fa -infmt fasta -parse_seqids \
  -outfmt maskinfo_asn1_bin -out refseq_seg.asnb
```

5.2.1.4 Extract masking information from FASTA sequences with lowercase masking:

We can also extract the masking information from a FASTA sequence file with lowercase masking (generated by various means) using convert2blastmask utility. An example command line follows:

```
$ convert2blastmask -in hs_chr.mfa -parse_seqids -masking_algorithm repeat \
  -masking_options "repeatmasker, default" -outfmt maskinfo_asn1_bin \
  -out hs_chr_mfa.asnb
```

Here the input is hs_chr.mfa (-in hs_chr.mfa), enable parsing of sequence ids, specify the masking algorithm name (-masking_algorithm repeat) and its parameter (-masking_options “repeatmasker, default”), and ask for asn.1 output (-outfmt maskinfo_asn1_bin) to be saved in specified file (-out hs_chr_mfa.asnb).

5.2.2 Create BLAST database with the masking information—Using the masking information data files generated in steps 5.2.1.1, 5.2.1.2, 5.2.1.3, and 5.2.1.4, we can create BLAST database with masking information incorporated.

Note: we should use “-parse_seqids” in a consistent manner – either use it in both steps or not use it at all.

5.2.2.1 Create BLAST database with masking information using an existing BLAST database or FASTA sequence file as input: For example, we can use the following command line to apply the masking information, created in step 5.2.1.2, to the existing BLAST database generated in 5.2.3:

```
$ makeblastdb -in hs_chr -dbtype nucl -parse_seqids \
  -mask_data hs_chr_mask.asnb -out hs_chr -title \
  "Human Chromosome, Ref B37.1"
```

Here, we use the existing BLAST database as input file (-in hs_chr), specify its type (-dbtype nucl), enable parsing of sequence ids (-parse_seqids), provide the masking data from step 5.2.1.2 (-mask_data hs_chr_mask.asnb), and name the output database with the same base name (-out hs_chr) overwriting the existing one.

To use the original FASTA sequence file (hs_chr.fa) as the input, we need to use “-in hs_chr.fa” to instruct makeblastdb to use that FASTA file instead.

We can check the “re-created” database to find out if the masking information was added properly, using blastdbcmd with the following command line:

```
$ blastdbcmd -db hs_chr -info
```

This command prints out a summary of the target database:

```
Database: human chromosomes, Ref B37.1
         24 sequences; 3,095,677,412 total bases
Date: Aug 13, 2009  3:02 PM      Longest sequence: 249,250,621 bases
Available filtering algorithms applied to database sequences:
Algorithm ID  Algorithm name      Algorithm options
         30      windowmasker
Volumes:
         /export/home/tao/blast_test/hs_chr
```

Extra lines under the “Available filtering algorithms ...” describe the masking algorithms available. The “Algorithm ID” field, 30 in our case, is what we need to use if we want to invoke database soft masking during an actual search through the “-db_soft_mask” parameter.

We can apply additional masking data to an existing BLAST database with one type of masking information already added. For example, we can apply the dust masking, generated in step 5.2.1.1, to the database generated in step 5.2.2.1, we can use this command line:

```
$ makeblastdb -in hs_chr -dbtype nucl -parse_seqids -mask_data \
  hs_chr_dust.asnb -out hs_chr -title "Human Chromosome, Ref B37.1"
```

Here, we use the existing database as input file (-in hs_chr), specify its type (-dbtype nucl), enable parsing of sequence ids (-parse_seqids), provide the masking data from step 5.2.1.1 (-mask_data hs_chr_dust.asnb), naming the database with the same based name (-out hs_chr) overwriting the existing one.

Checking the “re-generated” database with blastdbcmd:

```
$ blastdbcmd -db hs_chr -info
```

we can see that both sets of masking information are available:

```
Database: Human Chromosome, Ref B37.1
         24 sequences; 3,095,677,412 total bases
Date: Aug 25, 2009  4:43 PM      Longest sequence: 249,250,621 bases
Available filtering algorithms applied to database sequences:
Algorithm ID  Algorithm name      Algorithm options
         11      dust              window=64; level=20; linker=1
         30      windowmasker
Volumes:
         /net/gizmo4/export/home/tao/blast_test/hs_chr
```

A more straightforward approach to apply multiple sets of masking information in a single makeblastdb run by providing multiple set of masking data files in a comma delimited list:

```
$ makeblastdb -in hs_chr -dbtype nucl -parse_seqids \
  -mask_data hs_chr_dust.asnb, hs_chr_mask.asnb -out hs_chr
```

5.2.2.2 Create a protein BLAST database with masking information : We can use the masking data file generated in step 5.2.1.3 to create a protein BLAST database:

```
$ makeblastdb -in refseq_protein -dbtype prot -parse_seqids \
  -mask_data refseq_seg.asnb -out refseq_protein -title \
  "RefSeq Protein Database"
```

Using `blastdbcmd`, we can check the database thus generated:

```
$ blastdbcmd -db refseq_protein -info
```

This produces the following summary, which includes the masking information:

```
Database: RefSeq Protein Database
          7,044,477 sequences; 2,469,203,411 total residues
Date: Sep 1, 2009 10:50 AM      Longest sequence: 36,805 residues
Available filtering algorithms applied to database sequences:
Algorithm ID  Algorithm name      Algorithm options
           21          seg          window=12; locut=2.2; hicut=2.5
Volumes:
          /export/home/tao/blast_test/refseq_protein2.00
          /export/home/tao/blast_test/refseq_protein2.01
          /export/home/tao/blast_test/refseq_protein2.02
```

5.2.2.3 Create a nucleotide BLAST database using the masking information extracted from lower case masked FASTA file: We use the following command line, which is very similar to that given in 5.2.2.1.

```
$ makeblastdb -in hs_chr.mfa -dbtype nucl -parse_seqids -mask_data \
  hs_chr_mfa.asnb -out hs_chr_mfa -title "Human chromosomes (mfa)"
```

Here we use the lowercase masked FASTA sequence file as input (`-in hs_chr.mfa`), specify the database as nucleotide (`-dbtype nucl`), enable parsing of sequence ids (`-parse_seqids`), provide the masking data (`-mask_data hs_chr_mfa.asnb`), and name the resulting database as `hs_chr_mfa` (`-out hs_chr_mfa`).

Checking the database thus generated using `blastdbcmd`, we have:

```
Database: Human chromosomes (mfa)
          24 sequences; 3,095,677,412 total bases
Date: Aug 26, 2009 11:41 AM    Longest sequence: 249,250,621 bases
Available filtering algorithms applied to database sequences:
Algorithm ID  Algorithm name      Algorithm options
           40          repeat          repeatmasker lowercase
Volumes:
          /export/home/tao/hs_chr_mfa
```

The algorithm name and algorithm options are the values we provided in step 5.2.1.4.

5.2.3 Obtaining Sample data for this cookbook entry—For input nucleotide sequences, we use the BLAST database generated from a FASTA input file `hs_chr.fa`,

containing complete human chromosomes from BUILD37.1, generated by inflating and combining the `hs_ref_*.fa.gz` files located at:

```
ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/Assembled_chromosomes/
```

We use this command line to create the BLAST database from the input nucleotide sequences:

```
$ makeblastdb -in hs_chr.fa -dbtype nucl -parse_seqids -out hs_chr \
  -title "Human chromosomes, Ref B37.1"
```

For input nucleotide sequences with lowercase masking, we use the FASTA file `hs_chr.mfa`, containing the complete human chromosomes from BUILD37.1, generated by inflating and combining the `hs_ref_*.mfa.gz` files located in the same ftp directory.

For input protein sequences, we use the preformatted `refseq_protein` database from the NCBI `blast/db/ftp` directory:

```
ftp.ncbi.nlm.nih.gov/blast/db/refseq_protein.00.tar.gz
```

```
ftp.ncbi.nlm.nih.gov/blast/db/refseq_protein.01.tar.gz
```

```
ftp.ncbi.nlm.nih.gov/blast/db/refseq_protein.02.tar.gz
```

5.3 Search the database with database soft masking information

To enable the database masking during a BLAST search, we need to get the Algorithm ID using the `-info` parameter of `blastdbcmd`. For the database generated in step 5.2.2.2, we can use the following command line to activate one of the database soft masking created by `windowmasker`:

```
$ blastn -query HTT_gene -task megablast -db hs_chr -db_soft_mask 30 \
  -outfmt 7 -out HTT_megablast_mask.out -num_threads 4
```

Here, we use the `blastn` program to search a nucleotide query `HTT_gene*` (`-query HTT_gene`) with `megablast` algorithm (`-task megablast`) against the database created in step 5.2.2.1 (`-db hs_chr`). We invoke the soft database masking (`-db_soft_mask 30`), set the result format to tabular output (`-outfmt 7`), and save the result to a file named `HTT_megablast_mask.tab` (`-out HTT_megablast_mask.tab`). We also activated the multi-thread feature of `blastn` to speed up the search by using 4 CPUs^{\$} (`-num_threads 4`).

*This is a genomic fragment containing the HTT gene from human, including 5 kb up- and down-stream of the transcribed region. It is represented by `NG_009378`.

^{\$} The number to use under in your run will depend on the number of CPUs your system has.

In a test run under a 64-bits Linux machine, the above search takes 9.828 seconds real time, while the same run without database soft masking invoked takes 31 minutes 44.651 seconds.

5.4 Extract all human sequences from the nr database

Although one cannot select GIs by taxonomy from a database, a combination of unix command line tools will accomplish this:

```
$ blastdbcmd -db nr -entry all -outfmt "%g %T" | \
  awk ' { if ($2 == 9606) { print $1 } } ' | \
  blastdbcmd -db nr -entry_batch - -out human_sequences.txt
```

The first `blastdbcmd` invocation produces 2 entries per sequence (GI and taxonomy ID), the `awk` command selects from the output of that command those sequences which have a taxonomy ID of 9606 (human) and prints its GIs, and finally the second `blastdbcmd` invocation uses those GIs to print the sequence data for the human sequences in the `nr` database.

5.5 Custom data extraction and formatting from a BLAST database

The following examples show how to extract selected information from a BLAST database and how to format it:

```
Extract the accession, sequence length,
and masked locations for GI 71022837:
$ blastdbcmd -entry 71022837 -db Test/mask-data-db -outfmt "%a %l %m"
XP_761648.1 1292 119-139;140-144;147-152;154-160;161-216;
Extract the masked FASTA for GI 71022837:
$ blastdbcmd -entry 71022837 -db Test/mask-data-db \
-mask_sequence_with 20,40 -target_only
>gi|71022837|ref|XP_761648.1| hypothetical protein UM05501.1
MPPSARHSAHPSHHPHAGGRDLHHAAGGPPPGGGPGMPPGPGNGPMHHPHSSYAQSMPPPGGLPPHAMNGINGPP
PSTHGGPPPRMVMADGPGGAGGPPPPPHIPRSSSAQSRIMEAaggpagpppagppastspavqklslaNEaaw
vsiGsaetmedydralsayeaalrhnpysvpalsaiagvhrtdlnfekavdyfqrvinivpengdtWGSMDGHCY
LMMDDLQRAYTAYQQALYHLPNPKPEKLPWYIGIGILYDRYGSLEHAEEAFASVVRMDPNYEKANEIYFRLGIIYKQ
QNKFPASLECFRYILDNPPRPLTEIDIWFQIGHVYEQKKEFNAAKEAYERVLAENPNHAKVLQQLGWLYHLSNAG
FNNQERAIQFLTKSLESNDPAQSWYLLGRAYMAGQYNKAYEAYQQAVYRDGKNPTFWCSIGVLYYQINQYRDA
LDAYSRAIRLNPYISEVWFDLGSLYEACNNQISDAIHAYERAADLDPDNPQIQQLLRLNAEAKGGELPEAPVP
QDVHPTAYANNNGMAPGPPTQIGGGPGPSYPPPLVGPQLAGNGGGRDLSDRDLPGPGHLLGSSHSPPPFRGPPGT
DDRGARGPPHGLALAPMVGPGGPEPLGRGGFHSRGPSPGPPRMDPYGRRLGSPRRSPPPLRSVDVHDGHPGPP
HVGQGHGQGHGQGHGQGHGQGHGQGHGQGHGSHGSHGGFERGPPPLAAAGPGGPPPLDHYGRPMGGMSEREMEWER
EREREREREQAARGYPASGRITPKNEPGYARSQHGGSNAPSPAFAFRPPVYGRDEGRDYNNSHPGSGPGGPRGGY
ERGGPAPHAPAPGMRHDERGPPAPFEHERGPPPPHQAGDLRYDSYSDGRDGPFRGPPGLGRPTPDWERTRAGE
YGPPLSLHDAEGRNAGGSASKSRRGPKAKDELEAAPPPSPVPSSAGKKGKTTSSRAGSPWSAKGGVAAPGKNGK
ASTFFGTGVGAPVAAAGVGGGVGSKKGAASLRFQEDQFDSRPGSPQSRDASPASSDGSNEPLAARAPSSRMVD
EDYDEGAADALMGLAGAASASSASVATAAPAPVSPVATSDRASSAEKRAESSLGKRPYAEERAVDEPEDSYKRA
KSGSAAEIEADATSGGRLNGVSVSAKPEATAAEGTEQPKEKTRTETPLAVAQATSPEAINGKAESSEAVQPMDVD
GREPSKAPSESATAMKDSPTANPVVAAKASEPSPTAAPPATSMATSEAQPAKADSCEKNNNDEDEREEEGQIH
EDPIDAPAKRADEDGAK
```

5.6 Display BLAST search results with custom output format

The following example shows how to display the results of a BLAST search using a custom output format, in this case, the tabular output format with comments is requested along with the query accession, subject accession, evalue, query start, query stop, subject start, and subject stop. For brevity, only the first 10 lines of output are shown:

```
$ echo 1786181 | ./blastn -db ecoli -outfmt "7 qacc sacc evalue
qstart qend sstart send"
# BLASTN 2.2.18+
# Query: gi|1786181|gb|AE000111.1|AE000111
# Database: ecoli
```

```

# Fields: query acc., subject acc., evaluate, q. start, q. end, s.
      start, s. end
# 85 hits found
AE000111      AE000111      0.0      1      10596      1      10596
AE000111      AE000174      8e-30     5565     5671      6928     6821
AE000111      AE000394      1e-27     5587     5671      135      219
AE000111      AE000425      6e-26     5587     5671      8552     8468
AE000111      AE000171      3e-24     5587     5671      2214     2130
$

```

5.7 Use `blastdb_aliastool` to manage the BLAST databases

Often we need to search multiple databases together or wish to search a specific subset of sequences within an existing database. At the BLAST search level, we can provide multiple database names to the “-db” parameter, or to provide a GI file specifying the desired subset to the “-glist” parameter. However for these types of searches, a more convenient way to conduct them is by creating virtual BLAST databases for these. Note: When combining BLAST databases, all the databases must be of the same molecule type. The following examples assume that the two databases as well as the GI file are in the current working directory.

5.7.1 Aggregate existing BLAST databases

To combine the two nematode nucleotide databases, named “nematode_mrna” and “nematode_genomic”, we use the following command line:

```

$ blastdb_aliastool -dblist "nematode_mrna nematode_genomic" -dbtype nucl \
  -out nematode_all -title "Nematode RefSeq mRNA + Genomic"

```

5.7.2 Create a subset of a BLAST database

The nematode_mrna database contains RefSeq mRNAs for several species of round worms. The best subset is from *C. elegance*. In most cases, we want to search this subset instead of the complete collection. Since the database entries are from NCBI nucleotide databases and the database is formatted with “-parse_seqids”, we can use the “-glist c_elegance_mrna.gi” parameter/value pair to limit the search to the subset of interest, alternatively, we can create a subset of the nematode_mrna database as follows:

```

$ blastdb_aliastool -db nematode_mrna -glist c_elegance_mrna.gi -dbtype \
  nucl -out c_elegance_mrna -title "C. elegans refseq mRNA entries"

```

Note: one can also specify multiple databases using the -db parameter of `blastdb_aliastool`.