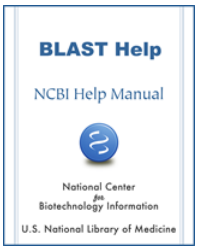
U.S. National Library of Medicine
National Center for Biotechnology Information

NLM Citation: Madden T, Camacho C. BLAST+ features. 2008 Jun 23 [Updated 2021 Mar 14]. In: BLAST® Command Line Applications User Manual [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2008-.
Bookshelf URL: https://www.ncbi.nlm.nih.gov/books/



BLAST+ features

Tom Madden[1] and Christiam Camacho[2]

Created: June 23, 2008; Updated: March 14, 2021.

Tasks

The blastn and blastp applications have a –task option. This option sets the parameters (e.g., word-size or gap values) to typical values for a specific type of search. For example, the “megablast” task is optimized for intraspecies comparison as it uses a large word-size, whereas “blastn” is better suited for interspecies comparisons with a shorter word-size. These tasks resemble the “Program Selection” section of the BLAST web pages and do not preclude the user from setting other options to override those specified by the task. See Appendix "Options for the command-line application" for documentation on parameter values for different tasks. 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

Megablast indexed searches

Indexing provides an alternative way to search for initial matches in nucleotide-nucleotide searches (blastn and megablast) by pre-indexing the N-mer locations in a special data structure, called a database index.

Using an index can improve search times significantly under certain conditions. It is most beneficial when the queries are much shorter than the database and works best for queries under 1 Mbases long. The advantage comes from the fact that the whole database does not have to be scanned during the search.

Indices can capture masking information, thereby enabling search against databases masked for repeats, low complexity, etc.

There are, however, limitations to using indexed search in blast:

- Index files are about four times larger than the blast databases. If an index does not fit into computer operating memory, then the advantage of using it is eliminated.

Author Affiliations: 1 Email: madden@ncbi.nlm.nih.gov 2 Email: camacho@ncbi.nlm.nih.gov

- Word size must be set to 16 or more in order to use an indexed search.
- Discontiguous search is not supported.

Reference: Morgulis A, Coulouris G, Raytselis Y, Madden TL, Agarwala R, Schäffer AA. Database Indexing for Production MegaBLAST Searches. Bioinformatics 2008, 24(16):1757-64. PMID:18567917

BLAST search strategies

BLAST search strategies are files that 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).

Exporting search strategies on the Web BLAST

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

Exporting search strategies with BLAST+ applications

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

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.

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).

Negative GI lists

Search applications support negative GI lists. This feature provides 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.

Masking in BLAST databases

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

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 “outfmt” in Appendix “Options for the command-line application” as well as the cookbook.

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 in Appendix "Options for the command-line application" as well as the cookbook.

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.

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 application provides a similar functionality for nucleotide sequences. Windowmasker uses a genome to identify sequences represented too often to be of interest to most users. 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.

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. evalue(A) >= evalue(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.

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).

BLAST-WindowMasker integration in BLAST+ search applications

The BLAST+ search applications support integration with the windowmasker files via the -window_masker_taxid and the WINDOW_MASKER_PATH configuration parameter (see Configuring BLAST) or via the -window_masker_db command line option.

In the first case, the WINDOW_MASKER_PATH configuration parameter should refer to a directory which contains subdirectories named after NCBI taxonomy IDs (e.g.: 9606 for human, 10090 for mouse), where the windowmasker unit counts data files should be placed with the following naming convention: wmasker.obinary (for files generated with the obinary format) and/or wmasker.oascii (for files generated with the oascii format). For an example on how to create these files, please see the Cookbook. Once these windowmasker files and the configuration file are in place, this feature can be invoked by providing the taxonomy ID to the -window_masker_taxid command line option.

Alternatively, this feature can also be invoked by providing the path to the windowmasker unit counts data file via the -window_masker_db.

Please see the Cookbook for a usage example of this feature.

DELTA-BLAST: A tool for sensitive protein sequence search

DELTA-BLAST uses RPS-BLAST to search for conserved domains matching to a query, constructs a PSSM from the sequences associated with the matching domains, and searches a sequence database. Its sensitivity is comparable to PSI-BLAST and does not require several iterations of searches against a large sequence database. See the cookbook for more information.

Concatenation of queries

BLAST works more efficiently if it scans the database once for multiple queries. This feature is known as concatenation. It speeds up MegaBLAST searches the most as they spend little time on tasks that consume CPU and most of the time streaming through the database. BLASTN and discontiguous MegaBLAST searches also run faster with concatenation, though the effect is less pronounced. BLAST+ applies concatenation on all types of searches (e.g., also BLASTP, etc.), and it can be very beneficial if the input is a large number of queries in FASTA format. BLAST+ concatenates queries by grouping them together until a specific number of letters (or "chunk size") is reached. Unfortunately, a constant chunk size for each database scan causes certain problems. For some searches the chunk size is too large, too many letters are searched at once, and the process consumes too much memory. Tests have shown that the number of successful ungapped extensions performed in the preliminary stage is a good predictor of overall memory use during a search. The BLASTN application (starting with the 2.2.28 release) takes advantage of this insight to provide an "adaptive chunk size". The application starts with a low initial chunk size of 10,000 bases and records how many successful ungapped extensions were performed during search. It adjusts the chunk size on the next database scan with a target of performing two million extensions during the search.

Query concatenation also means that BLAST will produce no output until the first set of concatenated queries have been processed. Some users find this disconcerting, but it is not a problem.

BLAST+ remote service

The BLAST+ applications can also send a search to the servers at the NCBI. In this case, the BLAST+ application is acting as a client and there is no need to install a database or provide more than minimal computing power. The BLAST+ remote service uses the same servers used by the NCBI BLAST website. The BLAST server can

return a Request ID (RID) as part of the results, and that RID can be used to reformat the results with the blast_formatter or on the NCBI website. In general, the servers keep the results for an RID for 36 hours. The BLAST+ applications will use the remote service if the –remote flag is added to the command line. The BLAST+ remote service uses a shared resource (the computers at the NCBI), so only one BLAST+ application should run remote searches at a time. An example in the cookbook section demonstrates a remote search.

BLAST database metadata

Starting from BLAST+ 2.13.0, makeblastdb generates an additional file with the file extension .njs (for nucleotide databases) or .pjs (for protein databases) which contains BLAST database metadata in JSON format. This file can be easily read by many tools and makes the BLAST database more Findable in the FAIR sense. Here is an example:


```
{
  "version": "1.2",
  "dbname": "protein-ecoli",
  "dbtype": "Protein",
  "db-version": 5,
  "description": "Escherichia coli protein sequences",
  "number-of-letters": 1358990,
  "number-of-sequences": 4289,
  "last-updated": "2022-03-09T13:39:00",
  "number-of-volumes": 1,
  "bytes-total": 2412774,
  "bytes-to-cache": 1397688,
  "files": [
    "protein-ecoli.pdb",
    "protein-ecoli.phr",
    "protein-ecoli.pin",
    "protein-ecoli.pnd",
    "protein-ecoli.pni",
    "protein-ecoli.pog",
    "protein-ecoli.pos",
    "protein-ecoli.pot",
    "protein-ecoli.psq",
    "protein-ecoli.ptf",
    "protein-ecoli.pto"
  ]
}
```


The fields represent the following:

Field	Description
version	Version of the BLASTDB metadata format
dbname	BLAST database base name
dbtype	Molecule type of the BLAST database
db-version	BLAST database version
description	Description of the BLAST database contents (i.e.: BLAST database title)
number-of-letters	Number of bases/residues in the BLAST database
number-of-sequences	Number of sequences in the BLAST database
last-updated	Date when this BLAST database was created
number-of-volumes	Number of BLAST database volumes that makes up this BLAST database

Table continued from previous page.

Field	Description
bytes-total	Number of bytes that comprise this BLAST database
bytes-to-cache	Number of bytes required to cache this BLAST database in main memory (RAM) for optimal performance
files	List of file names that comprise this BLAST database

Taxonomic filtering for BLAST databases

A popular feature of BLAST is the ability to filter the search of a BLAST database by taxonomy. The BLAST+ command line applications provide this functionality via several command line options:

- taxids
- negative_taxids
- taxidlist
- negative_taxidlist

These options take as input NCBI taxonomy IDs (taxIDs), which are stable, unique numerical identifiers for NCBI Taxonomy entries or TaxNodes (see Data model in the NCBI taxonomy handbook for details).

Here is an example:

BLASTP search of the nr BLAST database limited to Bacteria (taxID 2)

```
1 blastp -db nr -taxids 2 -query ...
```

Starting with BLAST+ 2.15.0, the BLAST+ command line applications support a new feature: they accept non-leaf taxIDs (i.e., those above an organism level, such as the one for primates). This improvement obviates the need to invoke separate tools or have network connectivity to limit non-leaf taxIDs. To support this feature, the NCBI distributes a standalone, file-based database called taxonomy4blast.sqlite3 . This additional database allows efficient taxonomic filtering for BLAST databases. For convenience, this database file is distributed alongside all BLAST databases distributed by the NCBI.

If you are using your own BLAST database(s) and would like to take advantage of this feature, you must set the taxonomy IDs in your database(s) and can get the taxonomy4blast.sqlite3 database by downloading https://ftp.ncbi.nlm.nih.gov/blast/db/taxdb.tar.gz , decompressing it and installing it alongside your other BLAST database(s).

Note for blastdbcmd users.

blastdbcmd supports filtering by taxID as well, but in the case of non-redundant databases (e.g.: protein nr) - where identical sequences are merged into the same entry in the BLAST database, regarless of its taxonomy - it may be advantageous to use the -target_only command line option when using -taxids . This will restrict blastdbcmd 's output to taxids matching those in the blastdbcmd invocation and their descendants (unless the -no_taxid_expansion option is used).

Note for developers and those who compile BLAST source code.

SQLite version 3.34 or more recent is required for this feature to work, as support for compound statements in recursive CTEs (Common Table Expressions) is needed.

Pre-compiled executables from the NCBI do not require that SQLite be installed.