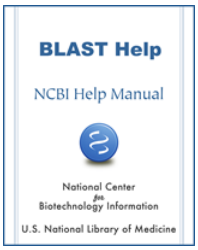
U.S. National Library of Medicine
National Center for Biotechnology Information

NLM Citation: BLAST® Command Line Applications User Manual [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2008-. Create a masked BLAST database. 2008 Jun 23 [Updated 2021 Jan 7].
Bookshelf URL: https://www.ncbi.nlm.nih.gov/books/



Create a masked BLAST database

Created: June 23, 2008; Updated: January 7, 2021.

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.

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 "Obtaining sample data for this cookbook entry", 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.

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 fasta -parse_seqids \
  -outfmt maskinfo_asn1_bin -out hs_chr_dust.asnb
```

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
```

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
```

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).

Create BLAST database with the masking information

Using the masking information data files generated in the previous 4 steps, we can create BLAST database with masking information incorporated.

Notes:

1. we should use "-parse_seqids" in a consistent manner – either use it in both steps or not use it at all.
2. Starting with the 2.10.0 release, makeblastdb produces version 5 databases by default, which uses LMDB. LMDB requires virtual memory (at least 600 GB, but 800 GB is recommended). Virtual memory is just that (virtual) and doesn't depend on the hardware in your system. In general, we recommend that BLAST users simply set the virtual memory to unlimited.

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 above, to the existing BLAST database generated in Obtaining sample data for this cookbook entry:

```
$ makeblastdb -in hs_chr –input_type blastdb -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 (-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 above to the database generated earlier by using this command line:

```
$ makeblastdb -in hs_chr –input_type blastdb -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 input and molecule type (-input_type blastdb -dbtype nucl), enable parsing of sequence ids (-parse_seqids), provide the dust masking data (-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 –input_type blastdb -dbtype nucl -parse_seqids \
  -mask_data hs_chr_dust.asnb, hs_chr_mask.asnb -out hs_chr
```

Create a protein BLAST database with masking information

We can use the masking data file generated in "Create masking information using segmasker" to create a protein BLAST database:

```
$ makeblastdb -in refseq_protein –input_type blastdb -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
```

Create a nucleotide BLAST database using the masking information extracted from lower case masked FASTA file

We use the following command line:

```
$ 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), its file type (-input_type fasta), 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 “Extract masking information from FASTA sequences with lowercase masking”.

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 BUILD38, generated by inflating and combining the hs_ref_*.fa.gz files located at:

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

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 B38"
```

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
```