Introduction to BLAST and protein homology searches Version 2

Frank Aylward

Abstract

This is a short tutorial on the basics of getting started with standalone BLAST+ in the Ubuntu command line.

Code is intended for use on an Ubuntu 16.04 LTS OS.

Note that a web server for BLASTP is also available: https://blast.ncbi.nlm.nih.gov/Blast.cgi?PROGRAM=blastp&PAGE_TYPE=BlastSearch&LINK_LOC=blasthome

For details on BLAST please see the NCBI webpage and release notes: https://www.ncbi.nlm.nih.gov/books/NBK131777/

And the NCBI paper on BLAST+: https://www.ncbi.nlm.nih.gov/pubmed/20003500?dopt=Citation

Citation: Frank Aylward Introduction to BLAST and protein homology searches. **protocols.io** dx.doi.org/10.17504/protocols.io.pivdke6
Published: 25 May 2018

Protocol

Make sure the right tools are installed first

Step 1.

Command to be entered into the command line are in bold

Comments are in regular typeface

We'll be using the BLASTP tool in the BLAST+ suite.

On an Ubuntu 16.06 system If you need to install this first, type:

sudo apt install ncbi-blast+

You will be asked if you wish to continue after you are told how much space it will take up. To proceed type 'Y'.

It may take a minute or two to finish installing.

Then, to view the abbreviated usage and flags, type:

blastp -h

and for the full usage and commands (there are a lot!) type:

blastp -help

We will go over some of the commonly-used commands in this tutorial, but it is always worthwhile to look at all of the available options, play around with different parameters, and see how the results can be changed.

Download the data from NCBI using the Unix wget command

Step 2.

wget -O prochlorococcus_phage_PSSM2.faa.gz ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/859/585/GCF_000859585.1_ViralProj15135/GCF_000859585.1_ViralProj15135_protein.faa.gz

wget -O prochlorococcus_phage_PSSM3.faa.gz ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/907/775/GCF_000907775.1_ViralProj209210/GCF_000907775.1_ViralProj209210_protein.faa.gz

These commands should download one .gz file each, and both will have to be uncompressed afterwards with the gunzip command

gunzip prochlorococcus_phage_PSSM2.faa.gz

gunzip prochlorococcus_phage_PSSM3.faa.gz

Get some basic stats about the files and what's inside

Step 3.

grep '^>' prochlorococcus_phage_PSSM2.faa | wc

or

grep '^>' prochlorococcus_phage_PSSM3.faa | wc

Since these are multi-FASTA files, we can (hopefully!) assume that by counting all of the lines that start with a ">" we are also counting the number of sequences in that file (since each sequence should have it's own header).

Choose one protein file to be the reference, and make the appropriate BLAST databases
Step 4.

makeblastdb -in prochlorococcus_phage_PSSM2.faa -dbtype prot

With BLAST we always need a query and a reference (or db). Which is which does not matter so much here. To identify orthologs we would want to perform a pair-wise comparison, since we would need to identify reciprocal best-BLAST hits for that. Here we can just start with a one-way BLAST.

Now use the other protein file as the query, and the reference file as the database
Step 5.

blastp -query prochlorococcus_phage_PSSM3.faa -db prochlorococcus_phage_PSSM2.faa | head -n 100

This is a pretty bare-bones command, but it will give you an idea of what the output looks like. Note that the output of BLAST went to the standard output stream (STDOUT) and that, for simplicity, we piped this into a "head" command so that only the first 100 lines would show up. We could have excluded the "head" command, but then we would have received a lot of BLAST output straight to our command line, which would be uninterpretable and not very useful.

Now let's vary the parameters a bit and see what the different results look like
Step 6.

Here is a similar command, but with a tab-delimited output and only the top 10 hits per query shown.
blastp -query prochlorococcus_phage_PSSM3.faa -db prochlorococcus_phage_PSSM2.faa -outfmt 6 | head

Now we can play around with the output parameters to ensure that only 'good hits' are reported, and only the best hit for each query protein is given.
blastp -query prochlorococcus_phage_PSSM3.faa -db prochlorococcus_phage_PSSM2.faa -outfmt 6 -max_target_seqs 1 -evalue 0.00001 -max_hsps 1 -qcov_hsp_perc 50 | head

Above, I put "good hits" in quotes since it is not always clear what constitutes a good hit. Different users will prefer different e-values and other cutoffs depending on what they are trying to do afterwards, their own comfort level, etc. As a common rule-of-thumb, e-values of 1e-3 and qe-5 are pretty common. For your own analyses you will need to use your own biological insight to decide for

yourself what you are willing to trust and whether the results make sense.

Here is a breakdown of the flags used above:
-query: this is the input file, so the file with all of the protein sequences that we want to search

-db: this is the database, so the file we just indexed with the makeblastdb command above. Note that makeblastdb creates multiple reference files and that only the root name needs to be given here (so if the database was called refdb, then refdb would be given here even though the index files are called refdb.pin, refdb.phr, etc.)

-max_target_seqs: This flat specifies that we only want the best hit for each query protein. Otherwise all hits are provided.

-outfmt: This specifies that we want the tab-delimited output format rather than the full alignment output. If you forget what the columns are you can use -outfmt 7.

-evalue: This indicates that we want to exclude all hits with evalues above this threshold. A good value is about 0.00001, or 1e-5.

max_hsps: HSPs are 'high-scoring segment pairs'. A query protein can make several separate alignments to a single reference, so this tells the program we want only the best-scoring alignment.

-qcov_hsp_perc: This is the 'query coverage high-scoring sequence pair percent', or the percent of the query protein that has to form an alignment against the reference to be retained. Higher values prevent spurious alignments of only a short portion of the query to a reference.

Now let's calculate the one-way amino acid identity (AAI) of the two genomes

Step 7.

To help with this we can install a package for simple math called 'datamash':

sudo apt install datamash

Datamash will allow for quick calculation of averages straight from the command line. Once this package is installed you can run:

blastp -query prochlorococcus_phage_PSSM3.faa -db prochlorococcus_phage_PSSM2.faa -

outfmt 6 -max_target_seqs 1 -evalue 0.00001 -max_hsps 1 -qcov_hsp_perc 50 | datamash mean 3

And the output should be a single number, which is the average of all of the % identity scores from the blast output

Now try doing the reverse and seeing how similar the results are (i.e., using PSSM2 as the query and PSSM3 as the db).

When you vary the e-value what happens to the one-way AAI? Does this make sense?

What about query coverage? How does increasing that change the one-way AAI?