SYSB 3036 W04: Homology searches

Version 3

Frank Aylward[1]

[1]Virginia Tech

dx.doi.org/10.17504/protocols.io.x8xfrxn

Frank Aylward
Virginia Tech

PROTOCOL STATUS

Working

We use this protocol in our group and it is working

1 Here we will use proteins predicted from the genomes of two *Prochlorococcus* bacteriophage genomes. We can use the wget command, which is already available as part of the base Ubuntu command line. Wget allows us to download files from a web server directly into the folder we are working in, and we need to know the URL for the file in order to do this. The National Center for Biotechnology Information (NCBI) has many genomes and genome-related datasets that it posts for researchers to use, and I have gone through and found the appropriate URLs to use here.
Here are the commands:

wget -O PSSM2.fna.gz
ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/859/585/GCF_000859585.1_ViralProj15135/GCF_000859585.1_ViralProj15135_genomic.fna.gz

wget -O PSSM3.fna.gz
ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/907/775/GCF_000907775.1_ViralProj209210/GCF_000907775.1_ViralProj209210_genomic.fna.gz

The -O flag specifies the file names that we want the downloads to be called. Without this flag wget would give the downloaded file the same names that they are given on the website, and sometimes these names can be quite long.

The above commands should download one gzip file each (extension .gz). Gzip files are commonly used for compressing data on Linux systems. Before we use them here we will have to uncompress them with the gunzip command
gunzip *.gz

After this we should have two .faa files. To check this we can use the ls command:
ls

And we should see something like this:

2 Now that we have our genomes downloaded and unzipped, we need to predict genes and proteins.

prodigal -i PSSM2.fna -a PSSM2.faa -d PSSM2.genes.fna -f gff -o PSSM2.gff

prodigal -i PSSM3.fna -a PSSM3.faa -d PSSM3.genes.fna -f gff -o PSSM3.gff

3 Now that we have the files downloaded and in the right format, we can get some basic stats about their format and content. FASTA files are formatted such that sequences are always preceded by a "header" line that starts with ">". This line contains information about the name of the sequence, and possibly other information.
Let's use "seqkit stats" to get some sequence statistics for the protein files.

seqkit stats PSSM2.faa
or
seqkit stats PSSM3.faa

4 Now that we have some basic information about the files we can begin formatting them for BLASTP.
For any search BLAST needs one FASTA file to be specified as the query, and one to be specified as the reference. Before running BLAST the reference needs to be formatted using a command called makeblastdb, which is part of the BLAST package and should have bee installed above. Makeblastdb takes a FASTA file as input and produces several files with different extensions that can be used as reference databases.

makeblastdb -in PSSM2.faa -dbtype prot

the -in flag specified the FASTA file to be formatted, and the -dbtype flag specifies the molecule type (prot for protein and nucl for nucleic acid).

If we check the folder after running the above command we should see a number of new files with the prefix PSSM2.faa and several new suffixes. Something like this:

5 Now that we have one file formatted as a reference database we can run BLASTP, using the other FASTA file as the query.

blastp -query PSSM3.faa -db PSSM2.faa | head -n 100

The output here is quite long, so we can pipe the output into a head command so we see only last 100 lines. You should see something like this (note not all 100 lines are shown below).

Note that there is a lot of information in this output. The program name and information are provided, and there is information on the query and reference databases used. The alignments that were calculated are also provided- you can see the top of one in the image above, and you can scroll to inspect it when you run it yourself. Note that this information is provided for every alignment that could be calculated for each protein in the query file, so if we had put this output into a file it would be quite large.

6 Since the output of the last step was quite extensive, we will want to find ways to simplify it.
Here is a similar command that will provide tab-delimited output (first 10 hits shown with the head command).

blastp -query PSSM3.faa -db PSSM2.faa -outfmt 6 | head

You should see something like this:

A few notes on the output here:
1) The format is tab-delimited, and the columns correspond to different statistics that were calculated for individual alignments. The columns are: query protein, reference protein, % identity, alignment length, mismatches, gap opens, query start, query end, reference start, reference end, evalue, bit score
2) Every protein in the query is compared to every protein in the reference, and all alignments are reported. So a protein in the query file could conceivable have alignments to multiple proteins in the reference (indeed, we see this in the image above). Also, multiple alignments that could be found between the same proteins are also shown (for example, if only the beginning and end of the amino acid sequences align, then two distinct alignmens will be provided).
3) Just becuase an alignment is reported does not mean it is 'real'. There are cases of 'spurious alignments' that could happend by random chance. We will need to investigate the statistics provided for each alignment to decide whether or not we think it's worth trusting.

Here are some additional parameters that will ensure that very poor alignments are not reported, and that only the best alignment for each query protein are given (for simplicity).
blastp -query PSSM3.faa -db PSSM2.faa -outfmt 6 -max_target_seqs 1 -evalue 1e-5 -max_hsps 1 -qcov_hsp_perc 50 | head

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.

7 Since we are comparing all of the proteins encoded in two viral genomes, it would be nice to get two basic statistics:
1) How many proteins in genome A are present in genome B and vice versa.
2) Of he proteins that are present in both genomes, how similar are they overall?

To answer the first question, we can simply use the command in the last step and count how many hits we find overall. Using the commands described in the previous step we need to be careful to make sure we are only counting the best hit for each query protein, and only one alignment per protein pair.

blastp -query PSSM3.faa -db PSSM2.faa -outfmt 6 -max_target_seqs 1 -evalue 0.00001 -max_hsps 1 -qcov_hsp_perc 50 | wc

Note that instead of piping the output to the 'head' command, as we did above, now we can pipe it into a 'wc' command to count how many output lines there are. You should see something like this:

So according to this analysis, there are 109 proteins in phage PSSM3 that have hits to phage PSSM2, with the parameters we used. What percent of all proteins in PSSM3 have hits to proteins in PSSM2? And vice versa?

Now as an exercise try changing the parameters a bit and see how they change the output. What do you think lowering the e-value threshold will do? What is the result of changing the query coverage percent?

Importantly, note that the results may change if we switch the query and the reference files (why would this be?), so we will want to do the reciprocal analysis too.

8 Above I mentioned two questions we would like to answer:
1) How many proteins in genome A are present in genome B and vice versa.
2) Of he proteins that are present in both genomes, how similar are they overall?

We answered the first qeustion above, and for the second we can use a package called 'datamash' that can help with simple math in the command line. Datamash will allow for quick calculation of averages straight from the command line by piping blastp directly into this command.

blastp -query PSSM3.faa -db 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. Since each line was a distinct alignment between one query protein and it's best match in the reference dataset, this gives us a nice idea of how similar the proteins are overall. Here I got 49.98, which is quite low for % amino acid identity. So it seems as though the protein sequences of these genomes are quite dissimilar.

For closely related genomes many scientists prefer using average nucleic acid identity (ANI) instead, but for distantly-related organisms this metric is less useful. Viruses evolve very quickly, so AAI is more useful here.

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?

This is an open access protocol distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited