Introduction to BLAST and protein homology searches Version 3

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.qfudtnw
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+
```

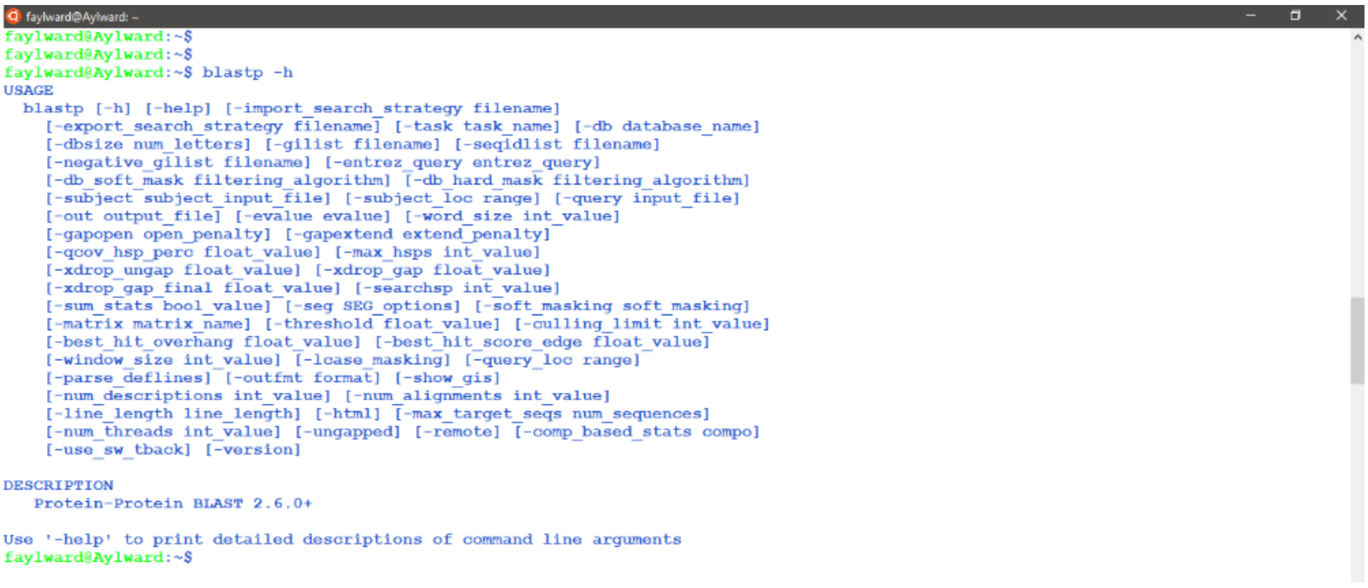
You will be asked if you wish to continue after you are told how much space it will take up. To proceed type 'Y'.

It may take a minute or two to finish installing.

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

blastp -h

And you should see short-form instructions for the general usage and different options, something like this:



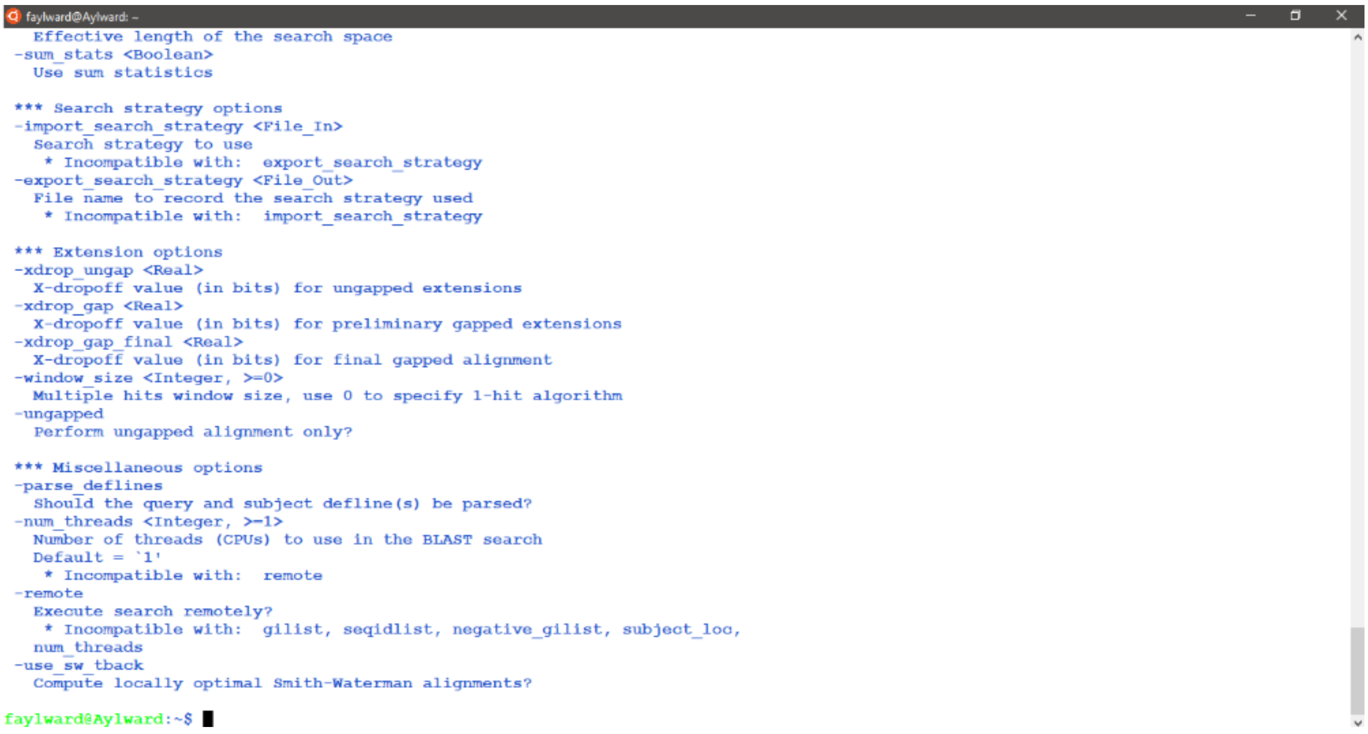
```
faylward@Aylward:~$
faylward@Aylward:~$
faylward@Aylward:~$ blastp -h
USAGE
  blastp [-h] [-help] [-import_search_strategy filename]
    [-export_search_strategy filename] [-task task_name] [-db database_name]
    [-dbsize num_letters] [-gilist filename] [-seqidlist filename]
    [-negative_gilist filename] [-entrez_query entrez_query]
    [-db_soft_mask filtering_algorithm] [-db_hard_mask filtering_algorithm]
    [-subject subject_input_file] [-subject_loc range] [-query input_file]
    [-out output_file] [-evalue evalue] [-word_size int_value]
    [-gapopen open_penalty] [-gapextend extend_penalty]
    [-qcov_hsp_perc float_value] [-max_hsps int_value]
    [-xdrop_ungap float_value] [-xdrop_gap float_value]
    [-xdrop_gap_final float_value] [-searchsp int_value]
    [-sum_stats bool_value] [-seg SEG_options] [-soft_masking soft_masking]
    [-matrix matrix_name] [-threshold float_value] [-culling_limit int_value]
    [-best_hit_overhang float_value] [-best_hit_score_edge float_value]
    [-window_size int_value] [-lcase_masking] [-query_loc range]
    [-parse_deflines] [-outfmt format] [-show_gis]
    [-num_descriptions int_value] [-num_alignments int_value]
    [-line_length line_length] [-html] [-max_target_seqs num_sequences]
    [-num_threads int_value] [-ungapped] [-remote] [-comp_based_stats compo]
    [-use_sw_tback] [-version]

DESCRIPTION
   Protein-Protein BLAST 2.6.0+

Use '-help' to print detailed descriptions of command line arguments
faylward@Aylward:~$
```

For long form instructions you can type the following:

blastp -help

And you will see a very long list of all the options and what they mean (this will take up more than one screen full, so you will need to scroll up after the text finishes printing to the screen to see everything).

The bottom of this long usage guide should look something like this:



```
faylward@Aylward: ~
   Effective length of the search space
 -sum_stats <Boolean>
   Use sum statistics

 *** Search strategy options
 -import_search_strategy <File_In>
   Search strategy to use
    * Incompatible with:  export_search_strategy
 -export_search_strategy <File_Out>
   File name to record the search strategy used
    * Incompatible with:  import_search_strategy

 *** Extension options
 -xdrop_ungap <Real>
   X-dropoff value (in bits) for ungapped extensions
 -xdrop_gap <Real>
   X-dropoff value (in bits) for preliminary gapped extensions
 -xdrop_gap_final <Real>
   X-dropoff value (in bits) for final gapped alignment
 -window_size <Integer, >=0>
   Multiple hits window size, use 0 to specify 1-hit algorithm
 -ungapped
   Perform ungapped alignment only?

 *** Miscellaneous options
 -parse_deflines
   Should the query and subject defline(s) be parsed?
 -num_threads <Integer, >=1>
   Number of threads (CPUs) to use in the BLAST search
   Default = `1'
    * Incompatible with:  remote
 -remote
   Execute search remotely?
    * Incompatible with:  gilist, seqidlist, negative_gilist, subject_loc,
   num_threads
 -use_sw_tback
   Compute locally optimal Smith-Waterman alignments?

faylward@Aylward:~$ 
```

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.

Step 2.

Now that we have BLAST installed we need to download some data to start analyzing. 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 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

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

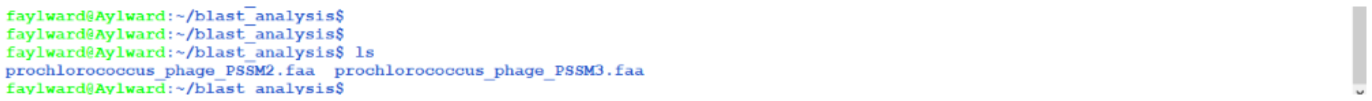
gunzip prochlorococcus_phage_PSSM2.faa.gz

gunzip prochlorococcus_phage_PSSM3.faa.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:



```
faylward@Aylward:~/blast_analysis$
faylward@Aylward:~/blast_analysis$
faylward@Aylward:~/blast_analysis$ ls
prochlorococcus_phage_PSSM2.faa  prochlorococcus_phage_PSSM3.faa
faylward@Aylward:~/blast_analysis$
```

Step 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.

Because every sequence starts with a ">" character, we can count how many protein sequences are in a file by counting the number of lines that start with ">". The Unix command to do this is grep, and the commands look like this:

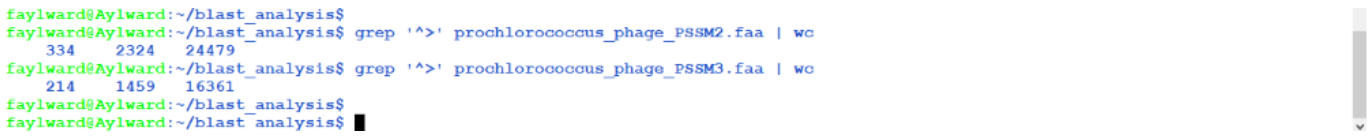
grep '^>' prochlorococcus_phage_PSSM2.faa | wc

or

grep '^>' prochlorococcus_phage_PSSM3.faa | wc

The '^' symbol that precedes the ">" tells grep that we are only looking for ">" symbols at the start of the line. At the end, the "| wc" tells the command line to pipe the output of grep to a wc command.

You can play around with these commands and remove the pipe at the end- you should find that grep returns each header line that starts wtih ">". For purposes here we don't want to see each line individually, we just want to count how many there are. The wc command returns the number of lines, characters, and bytes are in the input, so here this would equal the number of headers in a the FASTA files. The output should look something like this:



```
faylward@Aylward:~/blast_analysis$
faylward@Aylward:~/blast_analysis$ grep '^>' prochlorococcus_phage_PSSM2.faa | wc
    334    2324   24479
faylward@Aylward:~/blast_analysis$ grep '^>' prochlorococcus_phage_PSSM3.faa | wc
    214    1459   16361
faylward@Aylward:~/blast_analysis$
faylward@Aylward:~/blast_analysis$ █
```

Step 4.

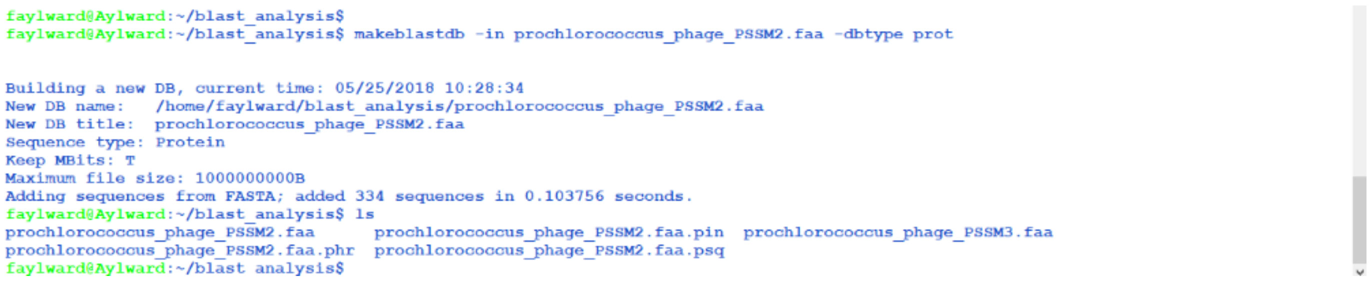
Now that we have some basic information about the files we can begin formatting them for BLASTP.

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 prochlorococcus_phage_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 prochlorococcus_phage_PSSM2.faa and several new suffixes. Something like this:



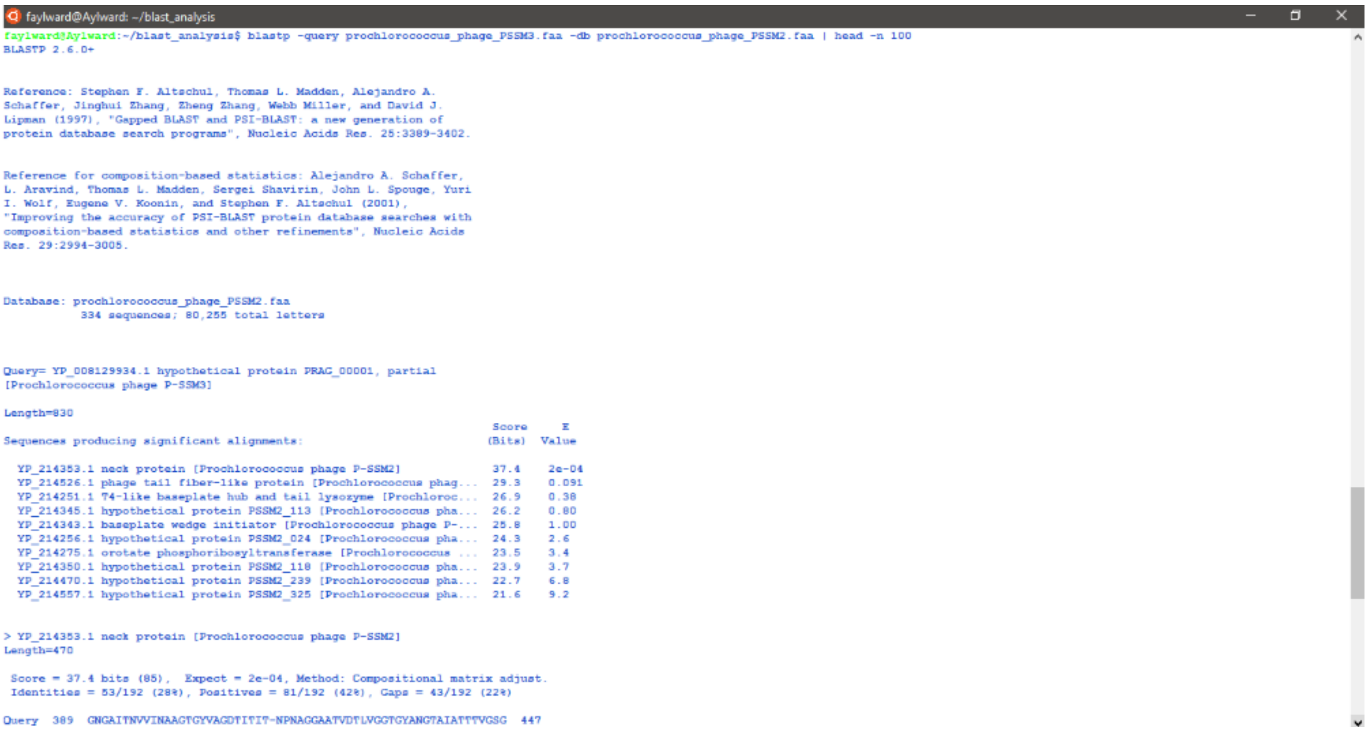
```
faylward@Aylward:~/blast_analysis$
faylward@Aylward:~/blast_analysis$ makeblastdb -in prochlorococcus_phage_PSSM2.faa -dbtype prot


Building a new DB, current time: 05/25/2018 10:28:34
New DB name:   /home/faylward/blast_analysis/prochlorococcus_phage_PSSM2.faa
New DB title:  prochlorococcus_phage_PSSM2.faa
Sequence type: Protein
Keep MBits: T
Maximum file size: 1000000000B
Adding sequences from FASTA; added 334 sequences in 0.103756 seconds.
faylward@Aylward:~/blast_analysis$ ls
prochlorococcus_phage_PSSM2.faa      prochlorococcus_phage_PSSM2.faa.pin  prochlorococcus_phage_PSSM3.faa
prochlorococcus_phage_PSSM2.faa.phr  prochlorococcus_phage_PSSM2.faa.psq
faylward@Aylward:~/blast_analysis$
```

Step 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 prochlorococcus_phage_PSSM3.faa -db prochlorococcus_phage_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).



```
faylward@Aylward: ~/blast_analysis
faylward@Aylward:~/blast_analysis$ blastp -query prochlorococcus_phage_PSSM3.faa -db prochlorococcus_phage_PSSM2.faa | head -n 100
BLASTP 2.6.0+


Reference: Stephen F. Altschul, Thomas L. Madden, Alejandro A.
Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J.
Lipman (1997), "Gapped BLAST and PSI-BLAST: a new generation of
protein database search programs", Nucleic Acids Res. 25:3389-3402.


Reference for composition-based statistics: Alejandro A. Schaffer,
L. Aravind, Thomas L. Madden, Sergei Shavirin, John L. Spouge, Yuri
I. Wolf, Eugene V. Koonin, and Stephen F. Altschul (2001),
"Improving the accuracy of PSI-BLAST protein database searches with
composition-based statistics and other refinements", Nucleic Acids
Res. 29:2994-3005.



Database: prochlorococcus_phage_PSSM2.faa
           334 sequences; 80,255 total letters



Query= YP_008129934.1 hypothetical protein PRAG_00001, partial
[Prochlorococcus phage P-SSM3]

Length=830
                                                                      Score     E
Sequences producing significant alignments:                          (Bits)  Value

  YP_214353.1 neck protein [Prochlorococcus phage P-SSM2]             37.4    2e-04
  YP_214526.1 phage tail fiber-like protein [Prochlorococcus phag...  29.3    0.091
  YP_214251.1 T4-like baseplate hub and tail lysozyme [Prochloroc...  26.9    0.38
  YP_214345.1 hypothetical protein PSSM2_113 [Prochlorococcus pha...  26.2    0.80
  YP_214343.1 baseplate wedge initiator [Prochlorococcus phage P-...  25.8    1.00
  YP_214256.1 hypothetical protein PSSM2_024 [Prochlorococcus pha...  24.3    2.6
  YP_214275.1 orotate phosphoribosyltransferase [Prochlorococcus ...  23.5    3.4
  YP_214350.1 hypothetical protein PSSM2_118 [Prochlorococcus pha...  23.9    3.7
  YP_214470.1 hypothetical protein PSSM2_239 [Prochlorococcus pha...  22.7    6.8
  YP_214557.1 hypothetical protein PSSM2_325 [Prochlorococcus pha...  21.6    9.2


> YP_214353.1 neck protein [Prochlorococcus phage P-SSM2]
Length=470

 Score = 37.4 bits (85),  Expect = 2e-04, Method: Compositional matrix adjust.
 Identities = 53/192 (28%), Positives = 81/192 (42%), Gaps = 43/192 (22%)

Query  389  GNGAITNVVINAAGTGYVAGDTITIT-NPNAGGAATVDTLVGGTGYANGTAIATTTVGSG  447
```

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.

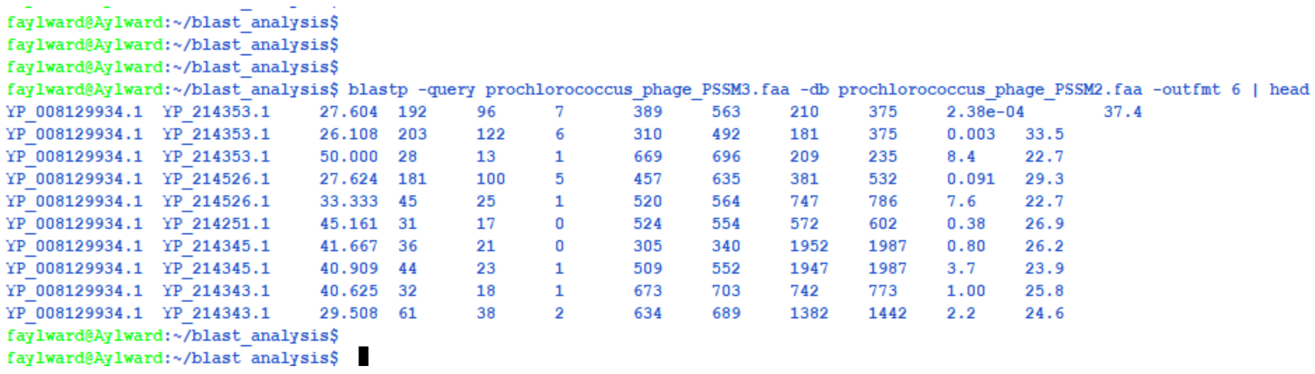
Step 6.

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 prochlorococcus_phage_PSSM3.faa -db prochlorococcus_phage_PSSM2.faa -outfmt 6 | head

You should see something like this:



```
faylward@Aylward:~/blast_analysis$
faylward@Aylward:~/blast_analysis$
faylward@Aylward:~/blast_analysis$
faylward@Aylward:~/blast_analysis$ blastp -query prochlorococcus_phage_PSSM3.faa -db prochlorococcus_phage_PSSM2.faa -outfmt 6 | head
YP_008129934.1  YP_214353.1     27.604  192     96      7       389     563     210     375     2.38e-04        37.4
YP_008129934.1  YP_214353.1     26.108  203     122     6       310     492     181     375     0.003   33.5
YP_008129934.1  YP_214353.1     50.000  28      13      1       669     696     209     235     8.4     22.7
YP_008129934.1  YP_214526.1     27.624  181     100     5       457     635     381     532     0.091   29.3
YP_008129934.1  YP_214526.1     33.333  45      25      1       520     564     747     786     7.6     22.7
YP_008129934.1  YP_214251.1     45.161  31      17      0       524     554     572     602     0.38    26.9
YP_008129934.1  YP_214345.1     41.667  36      21      0       305     340     1952    1987    0.80    26.2
YP_008129934.1  YP_214345.1     40.909  44      23      1       509     552     1947    1987    3.7     23.9
YP_008129934.1  YP_214343.1     40.625  32      18      1       673     703     742     773     1.00    25.8
YP_008129934.1  YP_214343.1     29.508  61      38      2       634     689     1382    1442    2.2     24.6
faylward@Aylward:~/blast_analysis$
faylward@Aylward:~/blast_analysis$
```

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 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

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.

Step 7.

Since we are comparing all of the proteins encoded in two viral genomes, it would be nice to get two basic statistics:

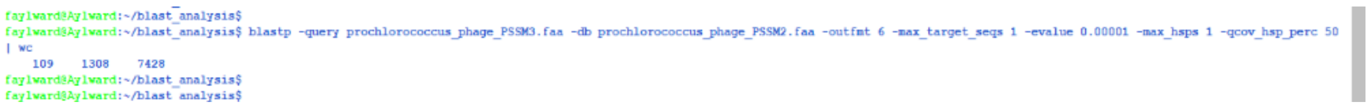
1) How many proteins in genome A are present in genome B and vice versa.

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 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 | 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:



```
faylward@Aylward:~/blast_analysis$
faylward@Aylward:~/blast_analysis$ 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
| wc
    109    1308    7428
faylward@Aylward:~/blast_analysis$
faylward@Aylward:~/blast_analysis$
```

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.

Step 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 will need to install a package called 'datamash' that can help with simple math in the command line.

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 the following command:

```
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. 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?