MCB 5472 Assignment #4: Introduction to command line BLAST

February 12, 2014

Today’s assignment builds on last week’s assignment using the set of 7 E.coli genomes of various qualities, and also introduces command line BLAST to answer some interesting evolutionary questions. We will be using BLAST+, which is already preinstalled on the Biotechnology Center Server. I therefore recommend that you switch to using this machine, using `qlogin` to access a compute node. You are also free to install the BLAST+ package on your own machine, but you will have to troubleshoot this yourselves. For the few of you using Biolinux, BLAST+ is preinstalled and so you can work on your local machine if you so choose.

Summary of important BLAST+ commands:

(1) Every BLAST search requires a database, and you will be using custom databases for this assignment that you have to make yourself. The BLAST+ command to do so is:

```
[jlklassen@bbcsrv3 ~]$ makeblastdb -in [name of input file] -dbtype [either
‘nucl’ or ‘prot’]
```

The output of this will be three files sharing the same name as the input file that you specified plus “`.nhr`” .”`.nin`” and “`.nsq`” for nucleotide databases and “`.phr`”, “`.pin`” and “`.psq`” for protein databases. The “database name” is the name of the input file that you specified. See `makeblastdb –help` for more information.

(2) The BLAST command syntax is the same for all basic blast flavors.

```
[jlklassen@bbcsrv3 ~]$ blastn –query [query file name] –db [database name]
```

Here are some other useful flags for this command:

```
-help
–evalue [maximum evalue threshold]
–out [output file name]
–outfmt [0 for normal alignment format; 7 for easy to parse table format]
```

You can run `blastp`, `blastx`, `tblastn` and `tblastx` by replacing `blastn` in the above command with one of these.

Before the start of next week's class: Write a short answer to each of the following questions, including: (1) exact copies of representative terminal commands that you used to answer the question, i.e., perl and/or BLAST+ (running BLAST+ from inside a perl script is ok too), sufficient for me to be able to reproduce your results; (2) a short summary of your results and how you interpret them. Include tables or figures if this helps. Answers should only be ~1-2 paragraphs long. Send these answers to me (jonathan.klassen@uconn.edu) along with any scripts used in a zipped folder. You do not need to include your input data (because we are all using the same data now) or pseudocode (although you are welcome to for comments).

Question 1: [10 marks] The two complete genome sequences from last week included several plasmids. Whether or not these plasmids were the same in each complete genome strain or present in the draft genomes is unclear. Use BLASTn to determine which plasmids in the two genomes are homologous and if these plasmids are present in the draft genomes. Defend your answers with data from your BLASTn results. YOU ARE NOT OBLIGED TO PARSE THESE BLAST RESULTS WITH PERL SCRIPTS, although feel free to!

Question 2: [10 marks] One way to identify paralogs is to BLAST sequences from a genome against itself. Identify the number of paralogous genes and proteins in the complete *Escherichia coli* O104:H4 str. 2009EL-2050 genome and its plasmids (this will require your downloading the genes, either through the web interface or as .ffn files from the FTP server). Only consider query-reference matches that can be aligned over 70% of both of their lengths. Compare the number of paralogous genes and proteins that you obtain using an evalue of 1e-5. Finally, use perl to roughly estimate the evolutionary age of your paralogs by rounding each hit's percent identify to the nearest 10% and counting how many paralogs exist in each bin. Remember that each gene can have multiple paralogs.

Note about finding lengths: The -outfmt 7 flag does not natively output query and reference sequence lengths! However, you can make it do so by copying the following syntax:

```
-outfmt "7 qseqid sseqid pident length mismatch gapopen qstart qend sstart
send qlen slen"
```

This adds the query and subject lengths as additional columns in the BLAST summary table. The ability to customize this table is the main reason I switched from blastall to BLAST+!