MCB 5472 Assignment #6: HMMER3 and scripting repetitive jobs

February 26, 2014

So…I lied last week and we're going to use that E.coli dataset once again to save you the grunt work of downloading your own data to accomplish this week's task. Our conceptual goal this week is to demonstrate how to combine perl and terminal commands to do a whole lot of work with very minimal effort. There are many ways to do this and I will only be showing you one, but the principle remains the same. The key concept is that you can use simple perl scripts to identify input files of interest and then perform some operation on those input files as a batch. So let's go!

Question: This week's question comes with several steps to help you focus on the logic.

(1) Modify your RBH script from last week to produce an output table listing the hit from genome #1 in one column and the hit from genome #2 in the other, separating the columns by tabs.

(2) Convert the multiple fasta files containing all of the protein sequences from our two test complete genomes (E.coli O104:H4 2009EL 2050 and E.coli O104:H4 2009EL 2071) into individual fasta files using the `seqret` command in the terminal.

NOTE: `seqret` *is a part of the* `EMBOSS` *software package, a remarkably useful suit that contains all sorts of small programs to manipulate sequence data. If you can think of something interesting to do, there is probably an* `EMBOSS` *command for that. Frankly, you could have used this to accomplish many of the earlier assignments in this course. But that wouldn't have taught you much perl, would it?*

Syntax for splitting a multiple fasta file:

```
seqret –auto –ossingle [input multiple fasta file]
```

One eccentricity of `seqret` *is that it uses all lower case letters for its output file names, even if the accession numbers were capitalized in the original NCBI multiple fasta file. Note that it also includes the version number, e.g.,* `yp_006768836.1.fasta`.

NOTE: More generally, `seqret` *is useful to convert between sequence formats and to trim sequences at specific positions. This works on both individual and multiple sequences, including sequence alignments.*

(3) Concatenate all of the 5 E.coli draft genome .faa files from Assignment #3 into a single multiple fasta file using cat:

```
cat *.faa > lots_of_faas.faa
```

(4) Write a perl script that does the following:

- Inputs the list of RBH hits from the complete E.coli genomes. Note that this list corresponds to all of the filenames that you just created using seqret, using a few helpful regular expressions.
- For each RBH pair, combine the sequences into a single multiple fasta file (containing only 2 sequences)
  NOTE: you can use cat for this too, e.g., `cat [fasta1] [fasta2] > mult.faa`
- Align these two sequences using MUSCLE (more on this program next week) according to this syntax:

  ```
  muscle –in [mult.faa] –out [mult.muscle.faa] (use your own names)
  ```

  NOTE: muscle outputs aligned multiple fasta files, hence my example output filename having a .faa ending

- Create a HMM using this alignment and HMMER3 according to this syntax:

```
hmmbuild [mult.hmm] [mult.muscle.faa]
hmmpress [mult.hmm]
```

- Use hmmsearch to find putative homologs in the draft E.coli genomes matching this HMM according to this syntax:
  ```
  hmmscan –o [hmmscan.out] [mult.hmm] [all_query_draft_genomes.faa]
  ```
- Parse the hmmsearch output file to estimate how many orthologs exist among the draft E.coli genomes. Use your output from Assignment #4 question #2 as a guide to setting similarity thresholds that minimize counting paralogs.

NOTE: The key perl command for this exercise is system, which tells perl to execute a system command. What makes this so powerful is the ability to use variables in the system command. For example, assuming you have obtained the fasta files from your input files:

```
system “cat $faa1 $faa2 > mult.faa”;

system “muscle –in mult.faa –out mult.muscle.faa”;

system “hmmbuild mult.hmm mult.muscle.faa”;

etc.
```

This strategy allows you to loop through a list of input files and perform the same analysis on all of them without any manual intervention. (So say, you can go home and sleep instead of babysitting your computer all night!)

NOTE: Before you run the entire analysis, you should ALWAYS run it once placing a `die` *command at the end of your loop through the list of inputs. Chances are always good one of the sub-processes will go wrong the first time and there’s no sense finding this out after you’ve run a whole bunch of useless commands.*

Before the start of class TWO WEEKS FROM NOW (Mar 12/14): Count the number of orthologs present in each draft genome and present this as a table like you did for paralogs in Exercise #3. Send this table and your scripts to me (jonathan.klassen@uconn.edu) along with a 1-2 sentence explanation for chosen threshold for defining orthologs.

NOTE: You will likely have some “ortholog” families with >5 members (the number of queried genomes). This is because there is a trade-off between sensitivity (finding all orthologs) and specificity (not counting paralogs) that is unavoidable using these types of threshold-based standards. You may also detect both halves of a fragmented gene; these would be true orthologs that falsely appear as multiple sequences due to insufficiencies during genome assembly or annotation.