MCB 5472 Assignment #6: HMMER and using perl to perform repetitive tasks February 26, 2014

One note about assignment #5

- Something we didn't talk about last week in class:
  - I showed code skipping lines after the first (best) hit was found in the BLAST output
  - Really, you should have done this for both BLAST outputs
  - If you made 2 hashes, one for each BLAST output, you are probably OK
  - Else you have to make a second hash just so you can skip found lines in the second outfile

```
open (INFILE1, $ARGV[0]) or die;
open (INFILE2, $ARGV[1]) or die;
while ($line = <INFILE1>){
	@array = split "\t", $line;
	next if ($hash1{$array[0]});
	next if ($array[3]/$array[10] < 0.7);
	next if ($array[3]/$array[11] < 0.7);
	$hash1{$array[0]} = $array[1];
}
while ($line = <INFILE2>){
	@array = split "\t", $line;
	next if ($hash2{$array[0]});
	next if ($array[3]/$array[10] < 0.7);
	next if ($array[3]/$array[11] < 0.7);
	$hash2{$array[0]} = "found";
	if ($hash1{$array[1]} eq $array[0]){
		$count++;
	}
}
print "$count RBHs found";
```

This week

- You have 2 weeks to complete this assignment!
- Central concept: you can use perl to automate repetitive computations
- Will also demonstrate how to use HMMER to create and use HMMs

Perl `system` command

- `system` is a perl command that runs terminal commands from inside a perl script

e.g., `system "cat file1.faa file2.faa > both.faa";`

- Advantage: can make the filenames in these commands variables and run them inside a loop

e.g., `system "cat $file1 $file2 > both.faa";`

Input

- The key to making these types of loops is having input files containing filenames so that you can process them one after the other
  - e.g., tables with multiple columns, each containing a filename
  - e.g., a list of files generated at the terminal
    - `ls *.faa > files.list`
  - Names don't even have to match exactly, just close enough that you can make exact matches with regular expressions

Today's exercise

- For each RBH pair from last week:
  - Combine both sequences into a single file
  - Create a multiple sequence alignment using muscle
  - Create a HMM using HMMER3
  - Search the other 5 draft genomes using that model
  - Count the hits
- 4000-5000 times total! With only a few lines of code (trust me)!

Step #1

- Modify your RBH script to make a table having orthologs on the same line separated by a tab
  - allows you to extract the filenames for further analysis

Step #2

- Convert the protein multiple fasta files into individual using the terminal command seqret
  - Part of the EMBOSS package (very useful!)
  - Syntax: `seqret –auto –ossingle all.faa`
  - Output: lots of files looking like `yp_006768836.1.fasta`
    - Note: small letters
    - Note: version number maintained (in gi header)

Step #3

- Concatenate all of the protein faa sequences from the 5 E.coli draft genomes into a single file using `cat`
  - syntax: `cat *.faa > all.faa`

Everything after this is looping in a perl script

Load your RBH filename table as the input file

Use "system" to invoke terminal commands

Step #4

- Extract filenames from the input table using regular expressions
  - To match the `seqret` output:
    - Keep the version number
    - Use small letters
    - Add "`.fasta`"
  - Note on regular expressions and NCBI fasta headers:
    - The "`|`" means "or" in a regular expression
    - If you want to match this, you need to precede it with a forward slash, e.g., "`\|`"

More regular expression fun

- You can capture text out of regular expressions using round brackets

e.g., `s/^gi\|(.+)\|/ # match everything between the first two “|” characters`

  - Everything in the round brackets is kept in the special `$1` variable
    - why you can't start your variable with a number in perl
- Can do multiple times
  - `s/^gi\|(.+)\|(.+)\|/`
  - `$1` contains what was between the first brackets
  - `$2` contains what was between the second brackets

Step #5

- Join your two RBH files into a single multiple fasta file containing 2 sequences
- e.g., `system; “cat $file1 $file2 > both.faa”`

Step #6

- Align the sequences in your multiple fasta file using the program `muscle`
- Syntax: `muscle –in [both.faa] –out [both.muscle.faa]`
- Note: `muscle` outputs aligned fasta files

Step #7

- Use your sequence alignment to create a HMM
- Syntax:

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

Step #8

- Query the multiple fasta file containing all of the proteins from the 5 draft genomes with the HMM
- Syntax: `hmmscan --tblout [table.out] -o [hmmscan.out] [both.hmm] [all_drafts.faa]`

Step #9

- Parse `hmmscan` output file and count number of orthologs
- Consider length and similarity thresholds to use considering the results of assignment #4 looking for paralogs
- It is possible to have >5 orthologs (poor genome quality)
- Keep a running total of ortholog counts (hash table like assignment #4)

Important hint:

- Run through your script once placing `die` at the end of your main loop
  - make sure everything works before running through ~25,000 commands