Unix, Perl and BioPerl

Session 3: Sequence analysis with Perl (modules including BioPerl)

Exercise 1: Parsing a file of multiple BLAST reports

Goal: Learn Bio::SearchIO, a very helpful BioPerl module to parse BLAST reports, even those concatenated from multiple queries.

See `http://jura.wi.mit.edu/bio/education/bioinfo2005/unix-perl/` for course page
See `http://bioperl.org/HOWTOs/SearchIO/index.html` for a good description of the Bio::SearchIO module.

This exercise should be completed with general Perl commands alone. The BioPerl-specific commands have already been written in the starting script. See the script for additional explanation of the script and individual lines of code.

To do:

Main tasks:

1. Run BLAST to search all of your rat ESTs against the human RefSeq database
2. Remove tags from an HTML file
3. Parse the BLAST file into tab-delimited output

- Make a directory called perl_2 and copy all files from /home/george/perl_2
  Starting script = `blastparse.pl`
  data file = `ests.fa` (the same set of 12 rat ESTs you used in the first class)
  a shell script that can remove HTML tags = `rmTags.csh`

- If needed, make `blastparse.pl` and `rmTags.csh` executable.

- Run BLASTX with the set of rat ESTs, ests.fa, against the set of human RefSeq proteins ("hs.faa"). BLAST reports in HTML are often easier to read, so set the HTML tag to true (-T T) in your command. Save the file as `ratblast.html`

```
blastall -i ests.fa -p blastx -d hs.faa -T T -o ratblast.html
```

- Look at the file `ratblast.html` in a text editor or a web browser (using the command 'netscape' with X Windows)

- Most BLAST parsers only accept text input, without any HTML tags, so either the BLAST analysis has to be originally produced as text, or one can try to remove the tags from the HTML file. To do this, use the shell script `rmTags.csh`, where the .csh identifies it as a C-shell script. If you look at it, you'll see it's quite short but depends on regular expressions

(in the sed "stream editor" tool) to identify and remove tags. Run the script by name (How do you do this?) to get the syntax. Save the output at `ratblast.txt`

- Open the Perl script `blastparse.pl` Note that near the top, it contains the line

```
use Bio::SearchIO;
```

so when the script runs, it will have access to commands described in the BioPerl module called SearchIO, the "Driver for parsing Sequence Database Searches". The script is already set up to process all the data, but it won't have any output (except warnings about variables being used only once). These warnings can be ignored in this example – but other times, the warning can indicate misspelled variables. Then the user is prompted for input.

- The lines

```
$report = new Bio::SearchIO(
          -file=>"$inFile", -format => "blast");
```

create a new SearchIO "object" that is then analyzed using Perl's object-oriented syntax. This object can be created with multiple query sequences, since a BLAST report can be made of several reports combined together (one after the other).

- Understanding the three levels of loops: The outer loop [`while($result = $report->next_result)`] is executed once for each query sequence, creating an object ($result) describing a single BLAST report in the most common use of the term. The next loop [`while($hit = $result->next_hit)`] goes through each matching sequence, creating the $hit object. Then the inner loop [`while($hsp = $hit->next_hsp)`] looks at each query-hit pair in more detail, going through each high-scoring segment pair (HSP) corresponding to one local alignment.

- **Below line "# 1"**: The first required command is to print a header line with the following terms, separated by tabs:

  QUERY NAME, HIT NUM, HIT DESCRIPTION,
  HIT-E-VALUE, FRACTION IDENTITY

- **Below line "# 2"**: Add the required set of commands to print out the top 5 hits for each query sequence, and print each line once, only for the first HSP. See comments in the script for further help with what to do. The values you can print (but add others if you want) are those listed above, so translate this pseudocode into Perl code. If you want to add other fields, modify the names of the fields printing in (1).

- Run the script, save the output as a text file (ending in ".txt") and look at it. Is it consistent with the whole HTML report? Can you identify where the data in your output comes from in ratblast.txt or ratblast.html?

- Log out of hebrides, download your output file with sftp, and look at it in Excel. What series of commands do you need to use? Hopefully you'll notice that it's much easier to sort and analyze lots of BLAST output in this format than in traditional BLAST output files.

- One possible solution (blastparse_SOLUTION.pl) will be in /home/george/solutions