## ASSIGNMENT #3 – Perl – BIOL550 (35 PTS)

### Background

Analyzing data in bioinformatics often requires feeding it through a series of different tools. However, because the output from a given tool may not be directly useable as input for another, we sometimes must reformat it accordingly. This can be done with programming languages, and parsing/reformatting text is one of the core strengths of Perl.

File formats in bioinformatics are often tab-, comma- and/or semicolon-delimited. For example, the output of sequence homology (*e.g.* BLAST, Diamond) and structure homology (*e.g.* Foldseek) search tools commonly adopt a column-based output delimited by one of these characters. Likewise, genetic data stored in VCF (Variant Calling Format) files is first separated by tabs, then the data present in each column further subdivided by semicolons whenever appropriate.

Being able to write simple scripts that parse these files to extract the desired information can be incredibly useful (imagine having to go through a few thousand files manually!). In this assignment, we will practice how to parse data from multiple files automatically.

### About newline formats

The files provided with the assignment are in UNIX/Linux format. When working with files, to prevent problems, you may want to check the newline format first. This format can vary depending on the operating system on which the files were generated:

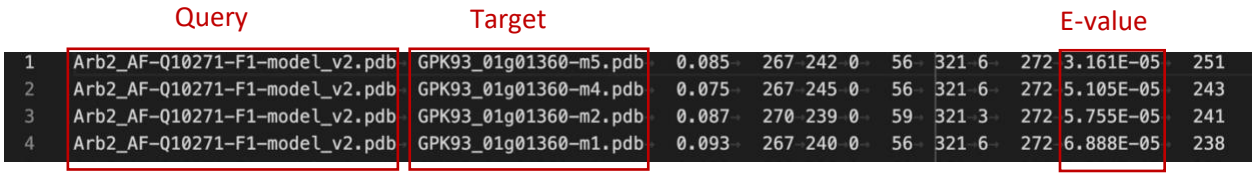
| | | | | |
|---|---|---|---|---|
| \r | => | CR (Carriage Return) | => | MacOS <= 9; really old, rarely seen nowadays |
| \n | => | LF (Line Feed) | => | Linux/UNIX and MacOS >= 10 |
| \r\n | => | CR +LF | => | MS Windows / MS DOS |

Different formats could lead to errors. For example, the following error **/usr/bin/perl^M: bad interpreter: No such file or directory** results from running a Perl script written on MS Windows (and using \r\n) as an executable on a UNIX-like system expecting \n. To fix this, you can convert the newline format of the Perl script with the Linux command line utility dos2unix, *e.g.* dos2unix *.pl.

### What to do:

Write a single Perl script named foldseek_parser.pl that does everything described below:

1) Opens (writes to) an output file named hits_per_protein.txt
   ## This is where we will store our results
2) Takes multiple file names from the @ARGV: *e.g.* foldseek_parser.pl FSEEK/*.fseek
   ## Those are input files that we will parse
3) Iterates through each .fseek file one after the other
   ## Hint; think about loops
4) Opens (reads from) the current .fseek file and iterates though its content line by line
   ## Be sure to use chomp to remove any tailing new line (to prevent potential problems)
5) Uses either a ***regular expression*** and/or the function ***split()***, and grab the following columns: query, target and e-value (columns # 1, 2 and 11, respectively) ## See description on next page




```
## The .fseek files are tab-delimited (12 columns); split() is really useful when working with files
## that are delimited by a repeated character. We have not seen it in class yet, but it is really
## easy to use. You can watch the pre-recorded lecture Perl_15_BacktricksSprintSplit_Part3 on
## blackboard if you want to learn how to use it. You can also ask us about it!
```

6) If the E-value is smaller or equal to 1e-20, prints the query, target and e-value (separated by semi-colons) to the hits_per_protein.txt output file.
```
## See A3_F22_desired_output.txt from the A3_F22_files.tar.gz TAR archive for the expected
## output
```

Test your script on the .fseek files provided in A3_F22_files.tar.gz. You can use diff -s to compare your hits_per_protein.txt file with A3_F22_desired_output.txt to ensure that it works as intended, *e.g.*

```
diff -s hits_per_protein.txt A3_F22_desired_output.txt
```

**Submit your Perl script on Blackboard**