BINF2010 – Scripting assignment

"Glue" programming and scripting are important skills in bioinformatics, allowing the automation of complex and repetitive tasks when analysing large datasets. This assignment is worth 15% of the total mark for the course and requires you to write a script to automate a "pipeline" of bioinformatics tasks.

Two options for the assignment are presented. Option 1 involves writing a bash shell script while option 2 involves a scripting language such as Perl or Python. Students who are doing or have done COMP2041 Software Construction and are familiar with Perl should be doing Option 2. If you have or are taking COMP2041 and you choose Option 1 your mark will be capped at 65%.

General considerations:

- Your script should run at the command line in a CSE account, but can assume that the binftools set-up command (`source ~binftools/setup-env.sh`) has already been run. You can make use of any programs installed in binftools (type `ls ~binftools/bin` for a list) as well as remote databases and software that you can access using their RESTful API.
- Your script should not crash or hang when given incorrect user input (eg, no arguments, or an incorrect argument) and instead should quit gracefully with a suitable error message.
- Your script should be properly formatted and commented. Some marks will be given for code that can be understood easily.
- You should try to include **an additional feature** to your script in addition to what is in the spec. It could be an additional analysis step, or an improvement to the suggested user interface. Indicate what that additional feature is and how to activate it in the comments of your script. This additional feature is worth 5% of the mark for the assignment (out of 100%). For example, for option 2, an extra feature could be to have your script also output a graphical representation of your annotations.
- That aside the script will be marked on its ability to perform the required task correctly, and its compliance with the specification.
- Your script should be your own work, however it is OK to use, as part of it, open source code that has been published as long as its source is clearly acknowledged in the comments. This is especially for option 2.
- You can make use of open source libraries such as BioPerl for some tasks as long as this use is acknowledged in the comments, and the script does not require any additional special installation steps such as installing extra libraries. Any extra code should be either already installed on the CSE computers or binftools, or included in the script itself.
- Your submission should consist of only one file, either .sh or .pl or .py, and should be directly executable.
- Submission will be through a link on the course website.
- Submission deadline will be advised on the website.

- Please ask your questions on the Assignment forum on Moodle. Email questions will be redirected to the forum.
- Finally: TEST, TEST and then TEST some more. Use a variety of inputs, testing locations etc, and make sure your program has no unintended consequences. One year a student submitted a script that did what was asked but then erased everything in the current directory (except for the output file it had created). This was not a good outcome.

Option 1: Bash shell script

Write a bash shell script that retrieves all the reviewed mammalian protein sequences from UniprotKB that correspond to a specific gene name (specified as argument 1 when running the script) then creates a multiple sequence alignment out of these sequences then builds a phylogenetic tree from this multiple sequence alignment.

- Your submission should consist of one file named **`uniTree.sh`**
- The script should be invoked using the command **`./uniTree.sh`** ***`GENE`*** where *GENE* is a gene code in UniprotKB, found in the Gene name [GN] field. Example gene names include INS and EGF.

The output of the script should be a phylogenetic tree file in Newick format (nested parentheses) suitable for input into a tree visualisation program such as Dendroscope. The file should be named ***GENE*.tree** (where GENE is the gene name specified at the command line). The leaves of the tree should be labelled with only the species name as much as the program allows you to (in some cases you may get multiple proteins from the same species in which case you may need to add a number to the name in the tree). An example output for EGF is on Moodle. However, note that because the databases are constantly updated, the example output may not be exactly as your program will generate it.

- The script should perform the following suggested steps:
    - Retrieve from UniprotKB all the sequences with the specified gene name in their GN field that also are of Mammalian origin (field **OC= Mammalia [40674]**) and have been reviewed for inclusion into SwissProt (field **Reviewed** set to **Yes**). The sequences should be retrieved in one file in fasta format. You probably will need to make use of the Uniprot RESTful API (http://www.uniprot.org/help/programmatic_access) for this, together with the UNIX command **`wget`**.
    - Edit the header of the fasta sequences so that they contain only the species name. If there are multiple sequences from the same species (for example a protein with multiple isoforms) you can add a number to the species name – eg Mus musculus1 and Mus musculus2.
    - Run a multiple sequence alignment program (eg **`clustalw`**) to align the sequences and create a multiple sequence alignment
    - Run another program to build a phylogenetic tree out of the multiple sequence alignment. Clustalw with the **`-tree`** option is probably the easiest option for this as phylip programs are quite

tricky to script and require writing a "batch file" (but do not simply use the clustalw guide tree created during multiple sequence alignment (.dnd file) as the tree as it is not a true phylogenetic tree).
    - Rename the tree file if necessary.
    - Delete any intermediary files created by the script, keeping only the final tree output (and any other files that were already there when you started the program).

Option 2: Perl or Python script

Select this option if you are already familiar with a scripting language, for example if you are doing or have done COMP2041
Write a Perl or Python script that

1. takes as input a bacterial genome sequence in fasta format
2. searches it for ORFs
3. selects the 3 longest ORFs of length ≥ 150bp/50aa (if there are not 3 ORFs that long, selects only the ones over the minimum length)
4. translates them into protein
5. searches the PDB database for matching proteins with at least 40% identity using the RSCB PDB Search API (https://search.rcsb.org/index.html#search-api), retrieving only the highest-scoring hit
6. Formats the results into a table in CSV format

The table should have up to 3 data rows (for the 3 longest ORFs) plus one header row with the following headers:

- Start: the start position of the ORF on the sequence
- End: the end position of the ORF on the sequence (note: for ORFs on the reverse strand, the end position index will be smaller than the start position)
- Strand: FORWARD or REVERSE
- BLAST hit: the PDB identifier of the **top hit** matching the translated ORF in the **PDB** database. If there is no similar sequence with at least 40% sequence identity then this field should just contain "-" (no quotes)
- E-value: the e value for the top BLAST hit, provided it is equal or lower than 1. If it is >1 then the field should contain "-" (no quotes)

The lines in the CSV file should be sorted by start position. An example input and output (geobacter.fasta, geobacter.csv) are provided in Moodle. However, note that because the databases are constantly updated, the example output may not be exactly as your program will generate it.

- Your submission should be a single file named geneannot.pl or geneannot.py
- The script should be invoked at the command line using the command **`./geneannot.pl` *`filename`*`.fasta`** (or **`./geneannot.py`**

***filename*.fasta**) where ***filename*.fasta** is the name of a nucleotide sequence file in fasta format containing a segment of a bacterial genome

- Your script should work with nucleotide sequences of length of up to 20,000 nucleotides (it can work with longer sequences if you want to but you should assume a test sequence of up to 20 kbps).
- Script should quit gracefully if the input sequence is invalid or not in fasta format.
- Script output should be in CSV format and go to standard output so it can be redirected if necessary. For example:
    - **`./geneannot.pl foo.fasta`** will display the CSV output on the screen.
    - **`./geneannot.pl foo.fasta > bar.csv`** will save the output to a file named bar.csv
- Suggested steps for the script:
    - Submit the input sequence to an ORF detection program of your choice (as long as it is installed in binftools)
    - For each of the 3 longest ORFs identified longer than (or equal to) 150bp, its protein translation should be submitted using the Protein Sequence Search API at RSCB PDB.
    - Parse both the ORF detection and the RSCB search outputs to extract the required information into a table in the specified format
    - Sort the ORFs by start position
    - Delete any intermediary files created by the script, keeping only the final CSV output (and any other files that were already there when you started the program).