Project Goal

The goal of the project was to make a Python tool for analyzing a bacterial genome. This analysis will include four things, all applied to both strands of the genome. First, each strand will be scanned for all potential open reading frames (ORFs) and filtered so that only frames with 50 codons or longer are considered. Second, the open reading frames will be translated from their DNA sequence into their amino acid sequence. Third, the aforementioned proteins will then have their corresponding MW calculated in kDa. Finally, the script runs a BLASTp analysis on the final proteins. These together provide a reasonable starting point for describing the potential proteome for a given prokaryote/bacteria.

Tools and Approach

The genome was downloaded as a .fasta file from BLAST. "NC_000913.3 Escherichia coli str. K-12 substr. MG1655, complete genome" is the name of the genome. This genome was read from a text file into a single Python string, representing one strand from the genome. The tool was implemented via a script called "project.py" whose primary function is called "AnalyzeBacterialGenome()." AnalyzeBacterialGenome() takes a string that represents one strand of the genome in question (5' to 3'); that is, a 1D collection of characters where each character corresponds to the nucleotide in its position in the genome in the 5' to 3' direction. The function immediately gets the reverse complement of the input strand for simultaneous analysis.

Next, the potential ORFs are found using a subroutine "GetORFs()." GetORFs() takes a string representing a DNA sequence, and returns a list containing potential ORFs. This scans the sequence one nucleotide by one nucleotide for instances of the start codon (ATG), by ranging over the entire sequence of the genome with an index i till the last 2 nucleotides of the genome. Every time a start codon is encountered, the function ranges over the sequence from where the start codon is found (index i to i+2) till a stop codon is encountered by updating an inner index j and checking if the codon (nucleotides from j to j+2) is a stop codon (if no stop codon is encountered, the sequence is considered invalid). Once a stop codon is encountered the inner loop terminates and i to j+2 is appended to a list of valid ORFs. Once every nucleotide in the genome has been considered (i has completed its range), then the list is returned. This is called for both the original strand and its reverse complement.

FilterByLength() is the next function called. This function filters a list based on some input length, checking and keeping each element greater than or equal to the input length. For this analysis, ORFs greater than or equal to 50 were considered.

After filtering for ORFs greater than 50, PerformTranslations() is called. This uses an openly sourced function "translate()" along with a library of DNA codons to translate the codons n a list of DNA sequences into a list of protein amino acid sequences. Importantly, the transcription step is skipped for brevity since the 5' to 3' coding DNA sequence is the same as the RNA transcript except T has been replaced with U.

Thirdly, GetProteinDictionary() is called to create the final outputs for AnalyzeBacterialGenome(). This is a dictionary whose keys is the protein number label (an

arbitrary label “protein i”, where i is the ith protein) and whose value is a list containing the sequence of the protein and its corresponding molecular weight in kDa. Molecular weights are calculated via a Biopython import “SeqUtils.” The method is bp.SeqUtils.molecular_weight().

Finally, the script takes the output dictionaries (one for both strands) then gets 5 of the smallest proteins and runs an NCBI BLASTp search on them. The search is limited so that the alignment and hitlist size parameters are 10 in order to make the search fast. The search results are saved to XML files “my_AntisenseBlast.xml” and “my_SenseBlast.xml.”

Description of Results

Unfortunately, the BLASTp calls in the python script are different than running the BLASTp manually on the website. I am not certain why there is a difference, but I am guessing there is some error in the parameters used for the “NCBIWWW.qblast()” call. The actual results are shown in Table 1 of the Appendix. As demonstrated in Table 1, all except the first of these proteins turned out to be a potential part of the murein transglycosylase SLT domain-containing protein. Since these proteins are all near each other in the genome (as indicated by their protein number), this may make sense. Transglycosylase is a lytic bacterial enzyme responsbile for catalyzing cleavage of the cell wall of bacteria (“Lytic Transglycosylases: Concinnity in concision of the bacterial cell wall” Dik et al. 2017). This enzyme can be used by bacteria during cell wall synthesis, degradation, and remodeling. The fact that a common protein was found from the analysis is indicative of the algorithms ability to successfully identifying potential ORFs.

Overall, 48267 potential proteins with length over 50 amino acids (ORFs of over 50 codons) were found by this method. According to “Characterization of E. coli proteome and its modifications during growth and ethanol stress” by Soufi et al (2015), they identified 2300 unique, expressed proteins to be approximately 88% of the *E. coli* proteome, which means there are approximately 2600 unique proteins in the expressed *E. coli* proteome. This means many of the 48267 potential proteins found are invalid, which is to be expected given the rudimentary method for capturing ORFs. One potential explanation for some of the disparity is that the *E. coli* in Soufi et al.’s experiments had silenced regions preventing expression of additional proteins that would be expressed by the bacteria under other conditions. Though, this would likely not completely explain the disparity. Another contributor to this is the non-coding regions of the genome were not taken into account (regulatory elements). Next steps to improve ORF capturing and other aspects of the algorithm are described below.

Next Steps and Improvements

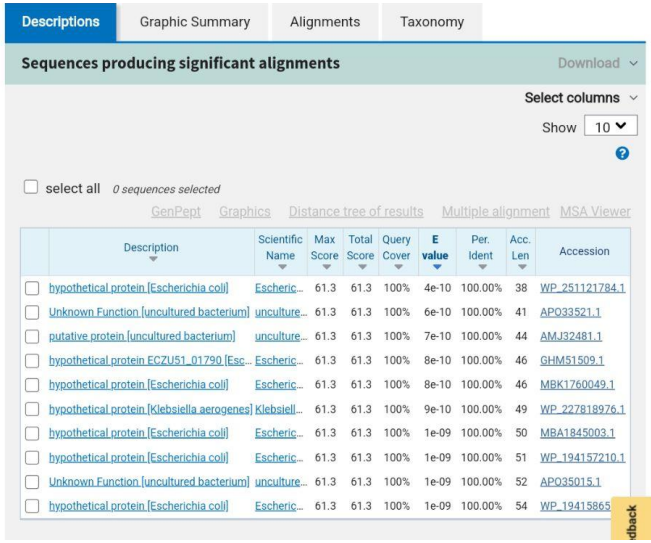
An obvious improvement is to make the BLASTp work within the Python script. This is supposed to be faster and less hands on than running BLASTp calls individually on the web, and would compile the many results into one easy xml file for both strands.

A next step would be to take into account the shine-dalgarno sequence along with common promoter proximal element(s) and other regulatory elements to eliminate even more ORFs, and narrow down to the actual coding sequences.

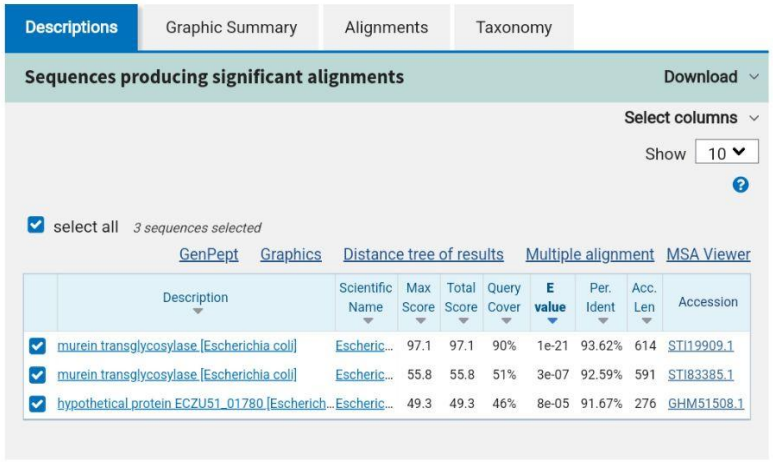
Eventually, modifying the tool or adding additional tools that would allow the user to look for specific proteins of given length or specific number of BLAST hits would be a good next step. For instance, right now the smallest proteins are found so if the user wanted to search the largest proteins or medium sized proteins the script would need to be edited manually to accomplish that. Instead, we could make the program more dynamic, automatically capturing the smallest and largest protein sizes and then prompt the user for the range of protein sizes to look for. Or perform a BLASTp search on all the proteins and return which ones have the highest alignment in other organisms. An additional tool the user could use to perform the same type of specific search on the output of AnalyzeBacterialGenome() would probably be better than adjusting the fnuction itself, since it works quickly and to maintain modularity.

Appendix

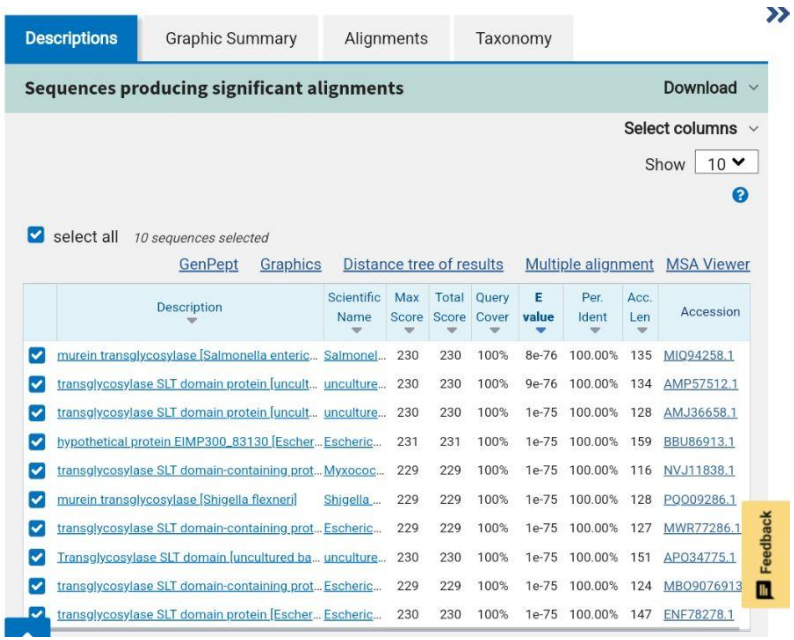
Table 1. BLASTp results for 5 Sense Proteins from AnalyzeBacterialGenome() applied to the genome of NC_000913.3 Escherichia coli str. K-12 substr. MG1655. BLASTp result manually retrieved from NCBI's BLASTp tool. (Table is split over pages 4-6)



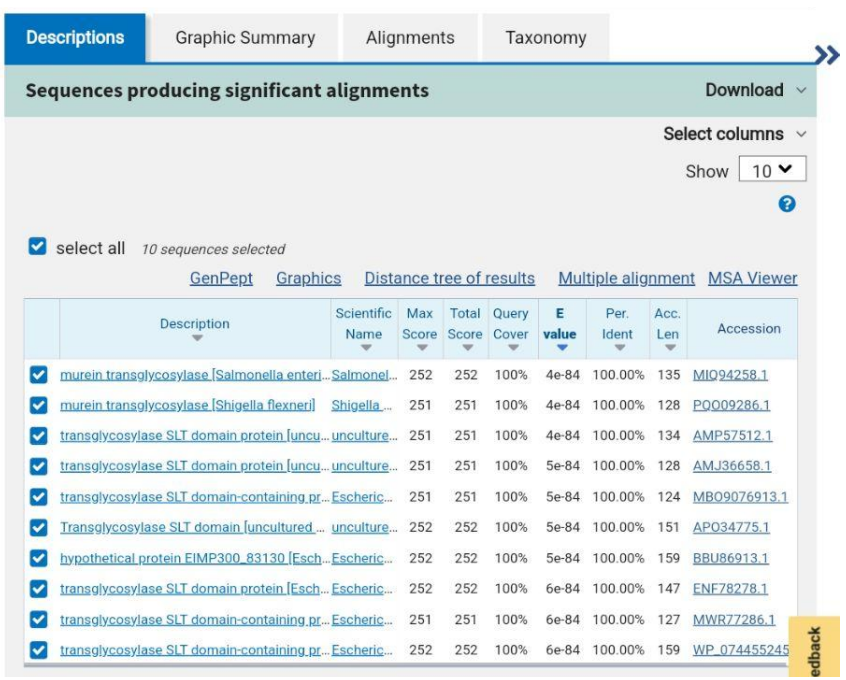
(1st Result) Sense protein 48172, length 17 amino acids, sequence: MGDKPTLMSATEWGRRY



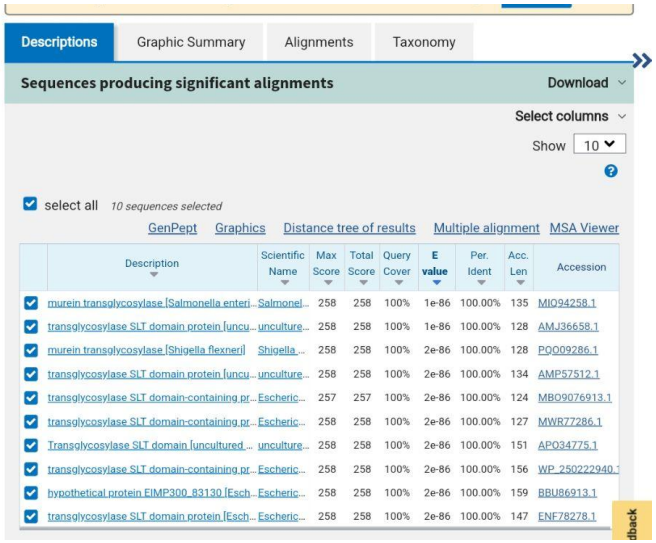
(2nd Result) Sense protein 48171, length 52 amino acids, sequence: MFISSLAIIVFSPQQLITPDQGGCEPGLATAPGVSTQWHLSRV FHSPRRAVM



(3rd Result) Sense protein 48170, length 111 amino acids, sequence:
MFSIPGYSSPGQLLDPETNINIGTSYLQYVYQQFGNNRIF
SSAAYNAGPGRVRTWLGNSAGRIDAVAFVESIPFSETRG
YVKNVLAYDAYYRYFMGDKPTLMSATEWGRRY



(4th Result) Sense protein 48169, length 121 amino acids, sequence:
MPGTATHTVKMFSIPGYSSPGQLLDPETNINIGTSYLQYVYQ
QFGNNRIFSSAAYNAGPGRVRTWLGNSAGRIDAVAFVESIPF
SETRGYVKNVLAYDAYYRYFMGDKPTLMSATEWGRRY



(5th Result) Sense protein 48168, length 124 amino acids, sequence:
MQIMPGTATHTVKMFSIPGYSSPGQLLDPETNINIGTSYLQYVYQQFGNNRIFSSAAYNAGPGRVRTWLGNSAGRIDAVAFV
ESIPFSETRGYVKNVLAYDAYYRYFMGDKPTLMSATEWGRRY