**Title:** Open-Reading Frame Analysis of a Bacterial Genome (*Brasscia napus*)

**Goal:** The general idea of this project was to develop a script capable of executing bioinformatic analyses on the open-reading frames of bacterial genomes from open-source biological data repositories. By leveraging the computational capabilities of integrative biological software tools through functional programming, as is possible with a biological Python library known as BioPython, one can uncover the genomes for a vast number of organisms & species through web-based interactions that can be launched within a program application. Moreover, it is possible to extend this further and discover the relevant proteins that these genetic sequences encode for. Finally, this study underscores the essence of novel bioanalytical tools in delivering extensive bioinformatic insights by demonstrating how tools like BLAST, which is used to find regions of sequence similarity between divergent protein sequence families, can also be managed and deployed in a computational environment with Python.

**Procedure:**

*Software Setup*

* This a Python (.py) program developed using Python version 3.6.

*Libraries Setup*

* Standard Python packages like NumPy and re were imported for managing different data types
* BioPython (Bio) version installment was required.
* SeqIO & Entrez functions were imported from Bio
* The ProteinAnalysis function was imported from the Bio.SeqUtils.ProtParams module
* NCBIWWW & NCBIXML functions were imported from the Bio.Blast module

*Data Setup*

* Entrez.esearch was used to browse the NCBI “nucleotide” database for the *Brasscia napus* term parameter.
* A BioPython data handle was used to fetch the sequence data from an Entrez.efetch query from the “nucleotide” database with the rettype = “gb”, for a GenBank sample with id = QGKT01000001. This returns the sequence for *Brasscia napus* that was determined via a whole-genome shotgun sequence project as a BioPython Sequence object

**Results:** The results for protein analysis for the ORFs of the selected bacterial genome were generated through a code analysis that comprised of the following functions to complete the mentioned tasks below

orf\_scan\_analyze(seq, table=11):

* this function is used to conduct a genome scan of both the template and reverse complement strands from the bacterial genome sequence that is passed into the function in the form of a BioPython Sequence object as the input.
* Each ORF Sequence object can use a callable “translate” method to convert the genetic code into an amino acid sequence using a reference translation table that is passed in as an argument to the function. Table = 11 for the purposes of this study.
* The Seq object’s “.split” method allows for quick identification of possible ORFs, as suggested in the *BioPython Tutorial and Cookbook* (J.Chang, Brad Chapman et. al)
* The ProteinAnalysis function was called on those ORF proteins that were at least 50 amino acids in length. The molecular weights, lengths, and sequence information were stored only for proteins satisfying this length threshold. Outputs are logged in the /data directory

query\_proteins(orf\_proteins,protein\_lens):

* A list of 5 query proteins of length < 1000 amino acids were selected from the results from scan\_genome to serve as input to the BLAST analysis function.

blast\_proteins(query\_proteins, e\_value\_thresh, protein\_lens):

* NCBIWWW & NCBIXML were used to run the BLAST query local from the Python environment for each protein.
* An XML file with BLAST results was generated for each query
* Txt files (BLAST\_query\_XX\_log\_results) with summarized finding from the BLAST search were generated for each query to serve as a user-interpretable output with details on the title of each alignment, the length of the alignment, the expected-value of the BLAST hits, and the HSPs of the BLAST alignment. Alignment information is *only* reported if the e-value of the hit is less than the e\_value\_thresh (set to an arbitrary high value for logging purposes). Since our query proteins are randomized, an empty log\_result file may sometimes occur.
* These outputs are logged in the log\_dir = /data directory

*Implementation (\_\_main\_\_):*

* All relevant files for the project are found in the directory “nmehtani-project” with code, docs, and data sub-directories.
* From the terminal CLI, the program can be run to stich together the entire analysis and output logging using the following commands:

1. $ cd ~/nmehtani-project/code
2. $ python main.py

* A Python notebook (main.ipynb) implmenetation is also included in the code folder for a walk-through demonstration.

**Reflection:**

Based on the results, we can determine the protein composition for different potential ORFs in the sequence. Additionally, we see reasonably high values for the molecular masses due to the high amino acid window chosen. Finally, while running BLAST for a random sample of query proteins, we see that some of the runs yield in significant similarity hits to the database search while others don’t within the specified threshold, perhaps suggesting a higher divergence from sequences that the BLAST sequence is founded upon.

Moving forward, It would be interesting to further identify and examine the functional properties of some of the significant hit results. From a computational aspect, it would be interesting to explore the other protein sequence analysis functions that enable the study of their structural and functional properties.