Sara Smith

Spring 2022

**140L Final Project: Ortholog Finder in Python**

**Objective:**

The objective of this project was to automate the process of finding orthologs, as we did by hand in the team lab section. This involves running BLAST with the original sequence, downloading the sequence of hits, checking for internal bsaI sites, and creating orthologs to amplify the ortholog out of the genome.

**Description:**

The github repo found here: https://github.com/2sarajade/ortholog\_finder.git contains ortholog\_finder.py, a script that can be run from the command line, as well as a few files to be used as example input/output. It takes in a fasta file with the protein sequence to find orthologs for and a list of organisms to search (by taxonomy ID). It returns the sequence of an ortholog, the 200 base pairs before, 200 base pairs after, warnings about the sequences, and optionally a suggestion for pcr primers that can amplify the 200-pre and ortholog region from the genome.

The program starts by parsing arguments provided by the user. It checks for the name of the input file, the name of the file containing organisms, if it should generate oligos, the directory to save the results in, and if the user asked for the help menu. If an unexpected argument is received, it will warn the user but continue the program. If no sequence file is specified, it will warn the user and quit the program.

If the user asks for help, it will print the help menu and quit. Otherwise, it will start the main loop. The program loops through the organisms provided in the file to do each of the following steps. Results for each organism are saved in a separate sub-directory.

First, it runs a blast search similar to how this would be done by hand. The script uses Biopython’s NCBIWWW package to connect to the online NCBI databases and Biopython’s NCBIXML package to parse the results. It pulls out the accession number, start and end of hits to pass to the next step, downloading the sequences. I used the Enterez package from Biopython to extract the sequences. It then checks if the sequence starts with a start codon. If not, it will switch to the reverse compliment and check again. If it still does not start with a start codon, it will switch back to the original orientation and warn the user. It then optionally generates oligos with bsaI sites, checks the sequences for unwanted bsaI sites, and writes all generated sequences to a fasta file called ‘result\_sequences.fasta’ warnings are sent to ‘warnings.txt’

**Usage Examples:**

Here is the output from passing the ‘-h’ (help) parameter to the function. It lists arguments to the function.

**Text

Description automatically generated**

Here is an example of how to call the program:



Which prints this to the terminal when operating normally:

Text

Description automatically generated

And generates sub-folders for each organism searched with results

Graphical user interface, application

Description automatically generated

These results and also another results folder (from using the default parameters) are included in the github.

**Warnings:**

The program will give several warnings to the user about issues with the sequences that are not severe enough to terminate the program. The program checks the sequences for the restriction site of the bsaI enzyme, and will write to ‘warnings.txt’ if it finds them. It will also add to the warnings file if the sequence does not start with a start codon.

**Issues:**

This program has two issues that could interfere with the usefulness of this project.

Issue 1: Apple Security

The first is Apple’s security protocols that make it difficult for Python to submit URL requests on Mac computers or MacBooks. Apple has been focused on increasing security on their devices in recent years, which has been breaking some older programs. The Biopython module is no exception. Unless the user installs the package ‘certifi’ and runs an executable from it, the part of ortholog\_finder.py that sends a request to the NCBI server will throw urllib.error.URLError: [SSL: CERTIFICATE\_VERIFY\_FAILED].

Mitigation:

I found a workaround for this which is included in the code, which is just to replace the default https context with an unverified one. This makes it work, but it is less secure.

Issue 2: It’s Slow

Ortholog\_finder.py is very slow, sometimes taking more than 10 minutes to find each ortholog. This is because it submits a query to the NCBI web server through Biopython’s NCBIWWW package. It checks to see if the result has been generated after 20 seconds, then every minute after that. There is no parameter to make it check more often. The time per ortholog can vary a lot depending on how busy the NCBI servers are, but I found it to generally be pretty slow when I was testing the program. It usually ran in 5-10 minutes per sample but did occasionally go up to 20 minutes.

Mitigation:

There is not much I can do about the speed of queries to the NCBI server. To help the user know that the program was still running and did not crash, I added status updates that print to the terminal before each step such as “Starting blast on [species].....,” “Checking 563 for restriction sites.....,” etc.