**Creating a Python Tool for The Purpose of Further Understanding HNH Endonuclease in Bacteriophage**

Caleb A. Collar

BIO 400Q, Carthage College

**Author Note**

Correspondence about this article should be addressed to ccollar@carthage.edu

Abstract

A framework tool written in Python for detecting the presence of HNH endonuclease genes in sequenced bacteriophage genomes was created. The process and purpose of building the HNH endonuclease detection tool is documented through this article and pertains to HNH endonuclease — terminase interactions. The general thesis for this tool’s first step in development is to: quickly and efficiently search a given bacteriophage genome for the presence of HNH endonuclease, and report on the data associated with its presence in a clear and concise manner. Future plans for the HNH endonuclease detection tool include the ability to detect terminase large subunit genes, display information about them, and additionally give relative distances between the HNH endonuclease gene(s) and the terminase gene(s) if they are present.

Keywords: HNH endonuclease, Python, Terminase, Framework

Creating a Python Tool for The Purpose of Further Understanding HNH Endonuclease in Bacteriophage

Currently, there is not much understanding around the importance and function of HNH endonuclease in bacteriophage life cycles. This tool was built in an effort to provide a basic open-source platform on which future bioinformatic tools can be developed and used for the purpose of furthering such research. The base functionality of this tool utilizes locally installed BLAST+ functions (Camacho, et al., 2009) and frequently updated Biopython libraries (Cock, et al., 2009) to achieve a simple and quick query about HNH endonuclease within a designated genome. HNH endonuclease is known to have physical interactions with terminase large subunits and play an important role in capsid head packaging (Kala, et al., 2014). Identifying the presence and location of HNH endonuclease genes in bacteriophage genomes, as well as identifying the presence of terminase large subunit genes in the same genome, may provide an avenue of research which can be exploited to further understand these proteins’ interactions.

# Methods

The production of the HNH endonuclease detection tool requires some dependencies to be implemented before the process of assembling the Python script. Once in place, these dependencies were then used by the Python script to carry out the functionality of the tool. The following procedure assumes one already has the necessary tools for interfacing with all the mentioned software and dependencies. For additional methods on how to find documentation about command line interfaces[[1]](#endnote-1) and write in a modern IDE[[2]](#endnote-2) with Python[[3]](#endnote-3) see the endnotes.

**Installing BLAST+**

To build and utilize the HNH endonuclease gene search function, BLAST+ must first be downloaded and extracted properly. Normally, BLAST+ functions are called via a command line environment – in this instance however, BLAST+ functions were used by Biopython modules. The BLAST+ source code and executables can be found at <https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/> for download. The proper version for your system(s) must be downloaded, in this instance the Linux version was used which is compressed as a ‘tar.gz’ archive. Additional breakdown on how to install BLAST+ can be found at <https://www.ncbi.nlm.nih.gov/books/NBK279671/>.

**Adding BLAST+ to the system PATH**

To allow the Python program to interface with BLAST+ functions, one must point the operating system PATH to the proper system directory where the BLAST+ bin folder is located.

This can be accomplished in UNIX systems through bash:

**$ export PATH=$PATH:$HOME/ncbi-blast-2.10.0+/bin**

**Installing Python**

One must ensure that Python 3x. is installed on the system. To find the current version of Python or check if Python is installed run the following command in bash:

**$ python -V**

If python is not installed or up to date, run:

**$ apt-get update**

**$ apt-get install python3**

**Installing Biopython[[4]](#endnote-4)**

One must also ensure that Biopython is installed so that the Python script can properly import the needed modules.

Running the following command will use the Python package management tool to quickly install Biopython on any platform:

**$ pip install biopython**

For updating an older version of Biopython, use:

**$ pip install biopython –upgrade**

**Creating a file structure**

To keep the project organized and allow for easy access to FASTA or FAA files; a file structure was created (Figure 1). This file structure consists of a folder for HNH endonuclease consensus sequences and a folder for FAA genomes for comparison. The Python program and the markdown readme file will exist in the primary directory.

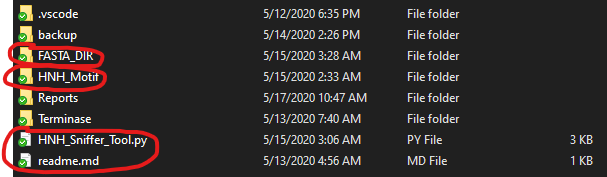


Figure 1

**Writing the Code**

The program must first import the necessary modules from the Biopython library like the ‘NcbiblastpCommandline()’ function, and other dependencies like the ‘io’ library. The imports used for this tool are seen in Figure 2.

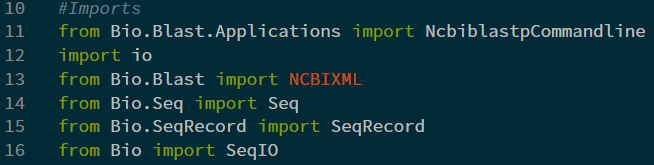


Figure 2

The next step was writing a line that allows for user input and will determine which FASTA/FAA file should be analyzed for HNH endonuclease consensus similarities (Figure 3).



Figure 3

Once one has the ‘userIn’ variable, we must pass this string on to the ‘NcbiblastpCommandline()’ function so it blasts the correct file for comparison. After the blastp function completes, the ‘output’ variable must take the output and pass it to a handler, in this case we will use the ‘NCBIXML.read()’ function before passing it on for alignment (Figure 4).

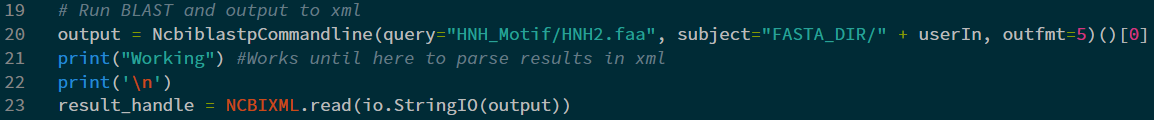


Figure 4

The handled information will then be passed on and parsed via ‘alignment’. The parsed information will then be displayed on the output console only if the alignment has an e-value less than 0.04. An e-value of 0.04 was selected as a reference cutoff for what might be considered an acceptable alignment. This variable can of course be manipulated depending on a researcher’s needs.



Figure 5

Using this code (Figure 5), the title, e-value, alignment length, position, and alignment visualization will be printed out in a clean and ordered fashion.

**Using the Tool**

Using the tool is a simple matter of invoking the python script itself:

**(directory where the project is located)~$ python HNH\_Sniffer\_Tool.py**

The program then prompts the user as seen in Figure 6.



Figure 6

Giving the program a valid input, ‘BigRedClifford.faa’ for example – will compare this file to the HNH endonuclease consensus sequence and provide an output like in Figure 7.

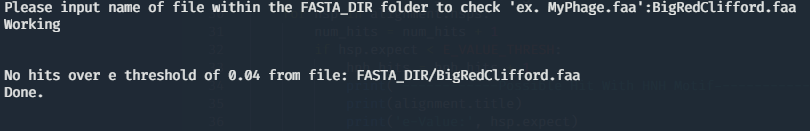


Figure 7

# Results

Results of this tool are based around the valid input (Figure 3. line 18) of an amino acid sequence — annotated or unannotated — in the FASTA/FAA format. Screenshots were taken of examples one might use in a real-world use case.

**Results of a Genome Known to Not Contain HNH Endonuclease Coding Sequences**

When testing the program with a genome file known not to contain HNH endonuclease (Figure 8), the results were as expected – running ‘BigRedClifford.faa’ through the program returned no hits over an e-value threshold of 0.04 (Figure 5. line 25).

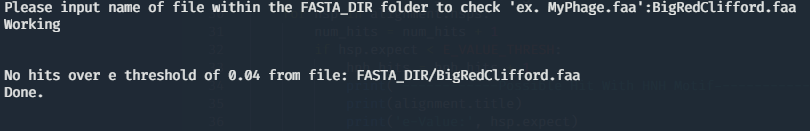


Figure 8

**Testing a Genome Known to Contain HNH Endonuclease Sequences**

Testing a file that is known and annotated to contain HNH endonuclease also yielded results as expected – running the Achromobacter phage Motura, complete genome ([QDH83728.1 accession](https://www.ncbi.nlm.nih.gov/protein/QDH83728.1)) yielded the following result in the program’s output as seen in Figure 9.

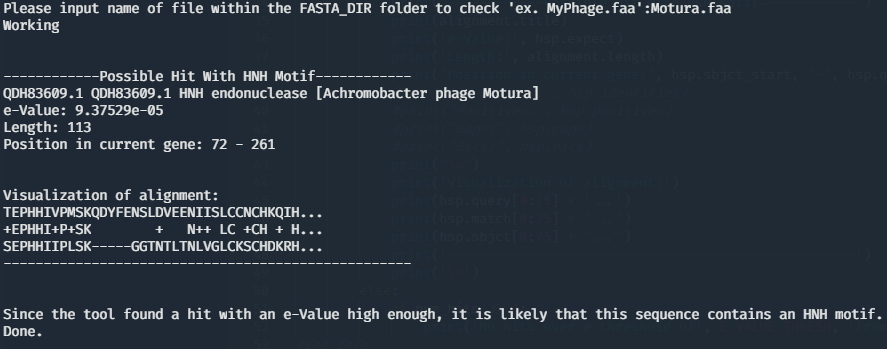


Figure 9

**Testing a Randomly Picked Bacteriophage Genome**

Testing an arbitrary file that was not known at the time to contain HNH endonuclease allowed for a quick search of the genome as a blind test. Running the Paenibacillus phage Dragolir, complete genome ([AUS03409.1 accession](https://www.ncbi.nlm.nih.gov/nuccore/MG727697.1)) yielded this result in the program output as seen in Figure 10.

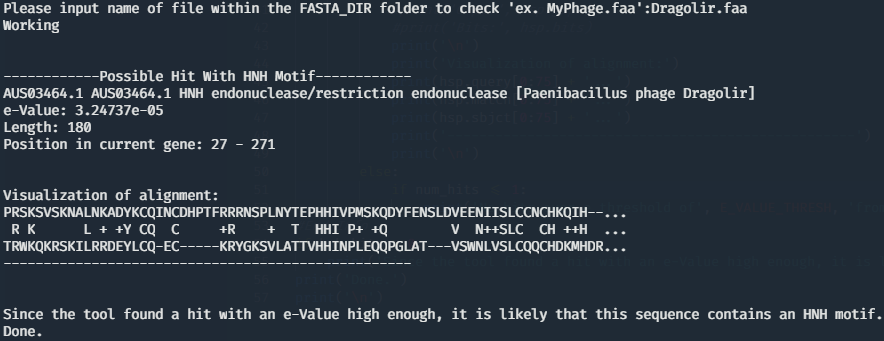


Figure 10

This output in Figure 10. lends itself to indicating the presence of an HNH endonuclease ([AUS03464.1](https://www.ncbi.nlm.nih.gov/protein/AUS03464.1)) sequence within the Dragolir genome. The e-value falls within the 0.04 threshold and is clearly annotated within the file as, “HNH endonuclease/restriction endonuclease”. With this output, we can be confident in concluding that HNH endonuclease exists within this phage’s genome and where.

# Discussion

The results of the HNH endonuclease detection tool are consistent with what one would experience should they take the time to run NCBI’s BLAST tool by manually entering comparison genomes. Accordingly, these results accomplish the thesis of this tool’s general framework purpose. That is; to quickly and efficiently search a given bacteriophage genome for the presence of HNH endonuclease and report on the data associated with its presence in a clear and concise manner.

**Notes on the Developed Tool State and the Future of the Tool**

Any amino acid sequence can be input through the HNH endonuclease detection tool. The tool has shown to be reliable at quickly identifying HNH endonuclease within bacteriophage genomes and is ready to be developed into a more mature tool for enhancing the study and research of HNH endonuclease as a whole. Without a doubt, for this tool to begin to show its worth in the avenue of HNH endonuclease research, terminase must also be identified within scanned genomes. The relationship of Terminase\_1 family terminase large subunits is known to have a unique and often juxtapositioned relationship with HNH endonucleases in long-tailed phages (Kala, et al., 2014). Depending on the use-case, this tool benefits from not requiring an application programming interface (API) with NCBI BLAST and therefore does not require a stable internet connection to utilize; however, the benefits of an API based tool are also many —especially in the current apparent web-connected era.

**Using Biopython for This Project**

Benefits of using Biopython are clear, as discussed and noted in Cock, et al., 2009, Biopython interfaces nicely with common bioinformatic tools and it is constantly updated and maintained:

“The Biopython project is a mature open source international collaboration of volunteer developers, providing Python libraries for a wide range of bioinformatics problems. Biopython includes modules for reading and writing different sequence file formats and multiple sequence alignments, dealing with 3D macro molecular structures, interacting with common tools such as BLAST, ClustalW and EMBOSS, accessing key online databases, as well as providing numerical methods for statistical learning.”

Once learned, Biopython is a foundational and powerful tool in the arsenal of a researcher using bioinformatics to tackle real-world problems and to discover new solutions. Biopython focuses on efficiency, reusability, automation, and expansion of unique methodologies.

References

Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., & Madden, T. L. (2009). BLAST+: architecture and applications. *BMC bioinformatics*, *10*, 421. <https://doi.org/10.1186/1471-2105-10-421>

Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., & de Hoon, M. J. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics (Oxford, England)*, *25*(11), 1422–1423. <https://doi.org/10.1093/bioinformatics/btp163>

Kala, S., Cumby, N., Sadowski, P. D., Hyder, B. Z., Kanelis, V., Davidson, A. R., & Maxwell, K. L. (2014). HNH proteins are a widespread component of phage DNA packaging machines. *Proceedings of the National Academy of Sciences of the United States of America*, *111*(16), 6022–6027. <https://doi.org/10.1073/pnas.1320952111>

1. Interfacing with software development tools at a basic level is necessary. The use of [Terminus α](https://eugeny.github.io/terminus/), a clean and modern command line interface (CLI) suitable for users at any skill level, is recommended. [↑](#endnote-ref-1)
2. Using an integrated development environment (IDE) is necessary in the modern world of building code of any kind. The use of [Microsoft Visual Studio Code](https://code.visualstudio.com/) (VS Code), a free and lightweight IDE with extremely powerful plugins designed for any programming language from C++ to Python, is recommended. [↑](#endnote-ref-2)
3. Using the correct extension for VS Code is necessary to acquire the correct linting and syntax. The [official Microsoft Python](https://marketplace.visualstudio.com/items?itemName=ms-python.python) plugin found in the Visual Studio marketplace, is recommended. [↑](#endnote-ref-3)
4. Biopython is freely available, with documentation and source code at [www.biopython.org](http://www.biopython.org) under the Biopython license (Cock, et al., 2009). [↑](#endnote-ref-4)