ZAMBIA INFORMATION COMMUNICATION TECHNOLOGY (ICT) JOURNAL

Volume 2 (Issue 1) (2018) Pages 44-56

Using Open Source Tools for Analysis of the Mutation Rate of African Cassava Mosaic Virus

Grey Chibawe
M.Sc. Computer Science student: University of Zambia
Department of Computer Science
P.O. Box 32375-Lusaka, Zambia
gchibawe@gmail.com[1]

Mayumbo Nyirenda
Lecturer: University of Zambia
Department of Computer Science
P.O. Box 32375-Lusaka, Zambia
mayumbo@gmail.com[2]

Lillian Mzyece
M.Sc. Computer Science student: University of Zambia
Department of Computer Science
P.O. Box 32375-Lusaka, Zambia
mamzyece@gmail.com[3]

Jackson Phiri
Lecturer: University of Zambia
Department of Computer Science
P.O. Box 32375-Lusaka, Zambia
jackson.phiri@gmail.com[4]

Popular tools used in studies in life sciences are often costly. This often pauses challenges to researchers in spite of the fact that research continues to be a key to the successful systematic development of new knowledge and a fundamental aspect to the usefulness of all higher education. Particularly, higher education also aims to advance, create and disseminate knowledge through research. Such critical studies like mutation studies therefore require affordable and fast results yielding software. In such research, open source software tools become handy in place of expensive proprietary tools. In order to provide alternative software tools for research, we decided to use a case study of the mutation of the African Cassava Mosaic Virus (ACMV) done by researchers in Zambia. The study of ACMV mutation is hampered by fragmented and non-user-friendly tools, which are currently available. A number of the tools used also depend on network connection, especially the Internet, to access and analyze data. To help alleviate this problem this research proposes the use of open source libraries in biopython to generate cost efficient and user-friendly solutions. Additionally, we propose the use of an open standard using XML as a standard protocol to share data between applications or stages in genomic data analysis of the ACMV. In our strife to provide open source solutions we analysed various tools and noted that biopython is quite popular. During our study of biopython our initial results show that it's possible to use free tools to analyze data in the life sciences and consequently reduce the time and cost required to analyze ACMV. Based on this case study we propose the adoption of such open source libraries in order to make research much more affordable for scientists in the life sciences for researches that operate within a constrained budget.

Keywords—Bioinformatics; Biopython; Metagenomics; Open Source Tools; Software

I. INTRODUCTION

Cassava Mosaic Virus (*Begomovirus*) is one of the major causes of the farmers' loss of cassava yield in Central and East Africa. Cassava is generally cultivated in Latin America, Africa and India. However, species of cassava-infecting geminiviruses have only been recorded in Africa (East African Cassava Mosaic virus and South African Cassava Mosaic virus) and India and its neighboring islands (Indian Cassava Mosaic virus, ICMV). Attacks from the African Cassava Mosaic Virus (ACMV) often lead to a decline in cassava yield.

Finding a solution to the declining yield in cassava caused by the ACMV has been, to a large extent, hindered by the ACMV's quick and unpredictable mutation. Currently, the tools used to predict mutation of the ACMV are fragmented and non-user friendly. Such tools are usually also expensive. Furthermore, a number of the tools used depend on network connection, especially Internet, to access and analyse data. As such, the process of accessing genome databases and that of analysing the downloaded data on data files is somewhat long. This often hinders the scientists' and researchers' ability to intervene. It is also important to note that most of the analysis is done using several software tools because one tool alone does not give all the desirable output. Therefore, researchers have to manually move data from one tool to the next and sometimes some output from some tools may not be compatible with the file format required as input in the next tool. Life science researchers will often have to use some intermediate tool to convert the output into a format the next tool will accept as input. This often requires more expertise and knowledge than the life science research is equipped with.

Consequently, even though agricultural and biological science researchers, in Zambia, have been working to come up with ways to eliminate or reduce the virus and its effect, the rapid mutation has always out done them. This is due to the major challenges the researchers face, which are related to

fragmentation, non-user-friendliness and cost of software tools used. This makes the researchers fail to meet the output rate required to help come up with solutions quick enough to beat the virus' rate of mutation.

In order to overcome the challenge of software tools, this research aims at developing a computational framework, which can be used to determine the rate of mutation of the African Cassava Mosaic Virus (ACMV) in order to provide a tool for the life scientists who seek solutions to the mutation problem. We further propose the use of open source libraries, which can also be used to develop other tools that can be used to solve other similar problems in the life sciences. For sharing of data and information we propose the use of XML based protocols. The rest of our paper is outlined as follows; in section II we discuss the African Cassava Mosaic Virus, in section III we present various tools which are currently used in the analysis of the mutation rate of ACMV, then in section IV we discuss XML technology, in section V we discuss Biopython, a project of open source libraries for bioinformatics, in section VI we discuss our proposed solution, in section VII we bring out our experiments, in section VIII we bring out preliminary results and finally make a conclusion in section IX.

II. AFRICAN CASSAVA MOSAIC VIRUS

A. Yield Dwarfing Cassava Viruses

One of the major yield dwarfing factors of the African cassava is the ACMV. Its prime vector is the African Cassava Whitefly, *Bemisia tabaci* [1] [2]. Studies done between the year 2000 and 2016 indicate that the rate and pattern of mutation in ACMV are unpredictable, thereby always outdoing the farmers and researchers. Since the year 2000 there has been a lot of research on ACMV. Nonetheless, ACMV was first reported in East Africa in 1894 [3]. Vincent N. Fondong and Kegui Chen in [4] said that Cassava geminiviruses occur in all cassava growing areas of Africa and are considered to be the most damaging vector-borne plant pathogens. J. P. Legg et al. also said the rapid geographical expansion of the cassava mosaic disease (CMD) pandemic, caused by cassava mosaic geminiviruses, has devastated cassava crops in 12 countries of East and Central Africa since the late 1980s [5] and that there is definitely need to find a solution to the ACMV problem. Basavaprabhu L. Patil et al. suggested that the use of RNA interference (RNAi) is an important strategy for the control of ACMV [6]. The challenge farmers and researchers face with the ACMV is its rate of mutation that has been unpredictable so far. This makes it difficult to find reliable ways to eliminate it. R. C. Aloyce et al. developed a single tube duplex and a multiplex PCR for the simultaneous detection of African cassava mosaic virus (ACMV), East African cassava mosaic Cameroon virus (EACMCV), East African cassava mosaic Malawi virus (EACMMV) and East African cassava mosaic Zanzibar virus (EACMZV), four cassava mosaic begomoviruses (CMBs) affecting cassava in sub-Saharan Africa [7]. The unprecedented rate of mutation and transmission of the ACMV has been aided by the super-abundant population of the whitefly vector [8].

B. Metagenomics of ACMV

Metagenomics, which is the study of genomic sequences in order to understand relatedness of organisms, has gained ground in the past several years. Metagenomics allows scientists to study microbial diversity and dynamics without having to perform any wet tests in the artificial media [9] [10] and is often used to study mutation of viruses. Often researchers that study this mutation are not specialists in information technology and as such relevant tools used in metagenomics must be efficient and user-friendly.

Most tools used by researchers in Zambia to study plant crop viruses are Internet based. Tools such as NCBI will depend on factors such as bandwidth and connectivity for output. Alicai et al. in [11] reported the use of a package called Phylogenetic Analysis by Maximum Likelihood (PAML), which Ziheng Yang in [12] said, was not the best for phylogenetic tree making because you have to manually modify tree files. With the foregoing, it implies that researchers have to move data from software to software in order to get all components of their desired results.

III. RELATED WORKS

From the information given by interaction with scientists from Mount Makulu Research Station, in Zambia and our own study of the currently used tools, we outline commonly used tools and what they are generally used for in this section and later on in section VI scenario of their uses.

A. Bioedit

Bioedit is an old software tool that is mostly used by scientists for percentage identity analysis and the study of phylogenetic relatedness of different DNA sequences. Virus DNA sequences are compared against other sequences in e-libraries. If the percentage is less than 80 percent the virus being analysed is considered to be unrelated to the other similar viruses in the database. When used in the study of ACMV Bioedit is often used before using the Sequence Demarcation Tool to prepare the data for export into the other software tools used later on in the analysis. Bioedit can import data from a clipboard as long as the data is in a well-defined format such as fasta files. Apart from clipboard imports, Bioedit can also read a number of formats including ".txt", ".fas", ".fasta", ".fst", ".xml", ".meg" (from Mega) as long as they are fasta file formats. Bioedit also has a NCBI web service capability and it allows for sequences to be viewed in many forms including colour shades on alignments.

B. Mega 6 or 7

Mega 6 is a sequence alignment tool. It aligns gene sequences for comparison of related positions. It is also used for making of phylogenetic trees. Mega 7 can read from many file formats but ".txt", ".xml", ".csv" formats are not supported. Though Mega 7 has many things it can do, it does not communicate directly with NCBI. Therefore, data must first be exported to a file before it is imported into Mega 7. Furthermore, to achieve basic alignment using Mega tools requires the use of a basic specific plugin.

C. Sequence Demarcation Tool (SDT)

SDT has all the features contained in Bioedit and Mega 6 but is visually more pleasant. SDT is also used mostly for virus percentage identity comparisons. Any percentage above 80 percent may be interpreted as a prediction of relatedness of one or more viruses being compared with a given virus. Such a percentage may also mean the viruses being compared are the same species and strain. A percentage less than 40 will mean a lack of relatedness of the viruses under comparison by DNA or RNA. Apart from being able to align the sequences, SDT also presents pairwise comparison using colour codes. This makes it easy for laymen to understand the identity comparisons. However, the matrix presentation of SDT is not that easy to understand for laymen. SDT can read from many file formats including ".txt", ".meg", and ".fas" but does not have a NCBI web service capability and so data has to first be exported to text files before importing into SDT.

D. Geneious

Geneious is a commercial genome analysis software tool. It has web service support for genome data. It has support for multi file formats for output such as ".txt", ".geneious" and ".csv". It can do all what the previously discussed tools can do from alignments to phylogenetic trees. It however lacks in the look and feel of outputs when compared to the other tools such as that of Mega. For example, the presentation of the alignment output does not have a pleasant look and feel but the phylogenetic tree does. Geneious also has a NCBI web service capability and can also create a local library for offline use.

E. National Center for Biotechnology Information

The National Center for Biotechnology Information (NCBI) advances science and health by providing access to biomedical and genomic information [13]. In the study of the mutation of ACMV, NCBI is used just for percentage identity of microorganisms by comparison with existing organisms already fed into the NCBI genome library. NCBI gives a sequence-by-sequence relatedness of microorganisms like virus strains. Data from NCBI can also be used for DNA/RNA sequence alignment. Data from NCBI can be exported to many file formats including ".txt", ".fas", ".csv", ".asn", ".json", and ".xml". NCBI provides both data for molecular biology as well as tools to analyse and study this data.

IV. eXTENSIBLE MARKUP LANGUAGE.

eXtensible Markup Language (XML) is essentially a markup language that defines and outlines a set of rules for encoding data files in a format that is both human and machine-readable. XML was designed to be both human- and machine-readable. For this reason, XML is increasingly becoming important norm and standard for the exchange of a wide variety of data on the Web and distributed applications [14] [15] [16] [17] [18]. It has also found use in many applications because it is not programming language and machine specific [19] [20] [21]. Using XML, disparate systems can communicate with each other by exchanging XML messages. Furthermore, XML can also be used to store the data persistently. XML protocols have been developed which can be used to develop solutions that allow two or more applications to communicate in a distributed environment, using XML as the language of encapsulation, storage and transportation. Thus, XML can be used for both storage and transportation of such data.

V. BIOPYTHON.

Biopython is a collection of open source bioinformatics tools written in an object-oriented scripting language called Python. It is a project, which dates as far back as August 1999 [22] [23]. Biopython makes available libraries to people doing computational Python for biological data. It can be downloaded for free from http://www.biopython.org. It is a project where many people have contributed and are still contributing making it a library rich project. Most, if not all, of the analysis tools required for bioinformatics can be built using Biopython libraries of the Python language [24] [25]. A number of modules can be integrated to make any bioinformatics project complete. It has modules for creating both online and standalone tools. The modules, which we desired to use for the purposes of this research, are summarised in figure 5. There is no need to create the science behind the modules because it has already been done and made ready to use.

VI. MATERIALS AND METHODS

The main aim of this research is to develop a computational framework, which can be used to determine the rate of mutation of the African Cassava Mosaic Virus (ACMV) in order to provide a tool for the life scientists who seek solutions to the mutation problem. To achieve this, the first thing we did was to conduct a study of online and standalone software tools used by Zambian agricultural and biological researchers to analyse the ACMV genome. Based on this study we then propose a computational framework for the prediction of the mutation of the ACMV and use the framework and open source libraries to start the development of a comprehensive user-friendly tool.

To better understand the usage scenarios, we had audience with researchers from Mount Makulu Research Station in Zambia to find out the procedure used when analyzing genome data for the prediction of the ACMV mutation rate. We will briefly describe the procedure used.

A. CASE OF MT. MAKULU RESEARCH STATION

The researchers from Mount Makulu Research Station took the following steps in analyzing ACMV genome data:

1. Percentage Identity Analysis

Percentage identity is the quantitative similarity between at least two DNA, amino acids and other genetic sequences. There are several genome libraries, globally, which include the NCBI [13]. Researchers compare genome sequences of organisms, which they are studying, with organisms whose genome data has already been stored on genome libraries. This is done to check whether their organism has already been studied or it is a newly discovered one. This is also done to

collect data of similar organisms for possible mutation studies. The retrieval of genome data from genome libraries is known as *blast.* In the same vain, the Mount Makulu researchers were always interested in first knowing whether the organism they were studying had already been recorded somewhere or it was a newly discovered organism altogether. In order to do this, they fed their genetic sequence in the *ncbi* library for percentage identity. *Bioedit* is another software tool (standalone), which they could use for percentage identity of multiple sequences, which have been downloaded from *ncbi* [26] [27]. However, they only used *Bioedit* to export the sequences to a fasta format file, which was then imported into *SDT* for percentage identity analysis. The researchers leveraged on *ncbi* only for downloads of similar sequences and not for identity analysis. Using *SDT*, if their sequence got at least 80 percent identity they deduced that the organism under study was either related to those in the library, which had at least 80 percent hit or it was exactly the same as one of the existing organisms. If the relatedness was less than 40 percent, they supposed that their organism could be a new discovery. The multiple sequences of interest identified from the *ncbi* were saved on a file in fasta format for input in Mega 6 or Geneious, for sequence alignment. *Although* these researchers used only *ncbi*, it is not the only genome data library; others like *Swis-Prot* and *ExPASy* exist [22].

2. Sequence Alignment

After the identity analysis, the Mount Makulu researchers proceeded to have their multiple genetic sequences aligned in a manner that would allow them to identify pattern consistency or inconsistency which would lead to conclusions on whether mutation occurred or not. The software tools they generally used for sequence alignment are Mega 6 and Geneious.

3. Pairwise Analysis

Following the sequence alignment is the analysis of sequences in pairs for relatedness. SDT is used for this purpose because it displays the results of pairwise analysis in nice colour density grids, which the Mount Makulu researchers find easy to interpret because they are laymen in computer programming. The other reason for their love of SDT is its visual results, which are easy to display for explanation to other computer science laymen.

4. Evolutionary Relationship Analysis

Once the genetic relatedness of sequences has been established and the multiple sequences have been aligned for consistency check and pairwise analysis, the final stage by the Mount Makulu researchers was to show evolutionary relationship of the many strains represented by the individual sequences. This was achieved by creating a phylogenetic tree using Geneious.

The four stages of genome data analysis have been summarised using the flowchart in figure 1. Each process shape contains a stage of the genome data analysis and the software tools used by Mount Makulu Research Station researchers to achieve their desired output. Some of these software tools can do more than what the researchers use them for.

B. Proposed solution: OPEN SOURCE SOFTWARE (Bio-pyhon libraries based tool)

We observed that some of the challenges in arriving at a timely solution to the ACMV are the rate at which the virus mutates against the software tools that give output slowly and the availability of affordable fast output software tools. In striving to provide a solution to the challenges, we delved into the study of Biopython libraries to find cheaper and quicker ways of carrying out the same analyses described in sub-section A of section VI. We discovered that the analyses could be done using some Biopython modules, methods, functions and few other integrated open source tools as described in sub-sections 1 to 4 that follow.

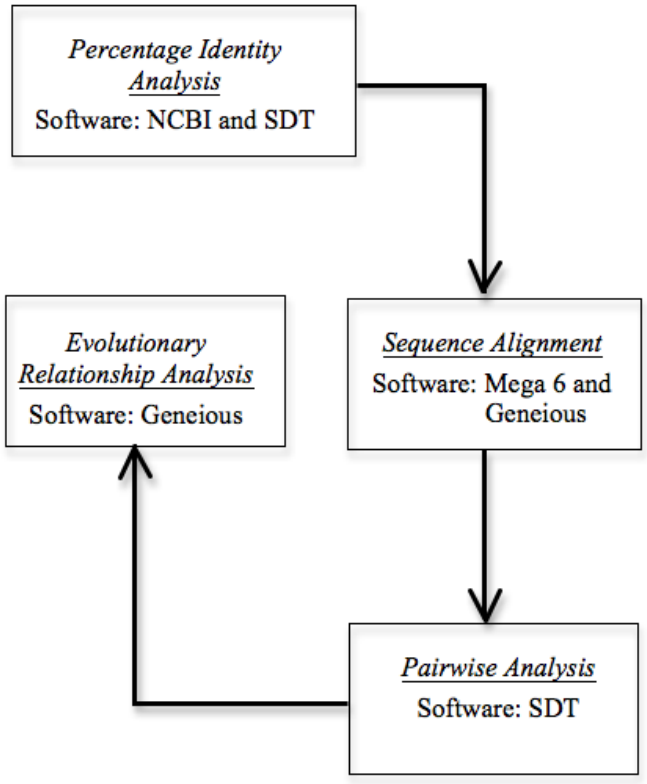

Figure 1: Stages of Genome data analysis carried out by Mt. Makulu Research Station researchers. Research software tools included

1. Percentage Identity Analysis

Biopython has a function *qblast( )* that calls the NCBI Internet blast module *Bio.Blast.NCBIWW*. This function can successfully be used to create a web service to be used for searching the NCBI library for related sequences of any microbe being studied. The NCBI genome library can be downloaded onto a local storage using Biopython. Once this is done, a *blastx* Biopython wrapper *Bio.Blast.Applications* module can be used to create command-line strings for blast purposes (identity searches from the local genome library). Blast is faster if the genome data sits on a local storage. The blast output using Biopython libraries include XML. This

makes it possible to come up with a framework that helps various software tools to communicate, knowing that XML is universal standard. The blast output can successfully be parsed using Biopython XML parsers when carrying out analyses.

2. Sequence Alignment

Using Biopython libraries, sequence alignment (from few sequences to many) can be achieved by utilising *Bio.AlignIO.read( )* function (for very few sequences) and *Bio.AlignIO.parse( )* function (for multiple sequences). The output can be written to a file of any fasta format including XML using *Bio.Align.Write( )* or *Bio.Align.convert( )* functions. *Bio.Entrez* is a parser for parsing the XML alignment files. Biopython has command-line tools for multiple sequence alignment; *Clustalw* and *Clustalx* are the most popular. They must be installed on a local pc first before an alignment is done. Bio.Align.Applications module in Biopython has a wrapper for the tools. When we tested alignment using Mega 6 and Geneious, it took over an hour to have the sequences aligned. Using *Clustalx* command-line tools took only slightly over 30 minutes, during our comparison test.

3. Pairwise Analysis

In order to carry out pairwise analysis using Biopython, the *Bio.pairwise2* module does the work. The input is mostly a fasta file but the module allows programmers to create their own functions. This is a plus in that during the creation of our own functions we could use XML files which is our proposed file share standard. Pairwise analysis is just an alternative to the sequence alignment described in sub-section 2 of section VI (B).

4. Evolutionary Relationship Analysis

Evolutionary relationships are easily presented using phylogenetic trees. Biopython has *Bio.Phylo* module for phylogenetic tree creation. PhyloXML feature of the same module helps to create XML phylogenetic trees. Phylo module can read from .dnd and .xml files.

VII. EXPERIMENTS

A. Data

We decided to run experiments, ourselves, with the software tools used by Mount Makulu Research Station researchers and went on to run more experiments but this time using biopython libraries to compare rate of output. During our experimentation we used ninety-two ACMV sequences, which we obtained from NCBI through an online nucleotide blast of a virus with accession number "AJ717542.1". A researcher from Mount Makulu Research Station gave the accession number and its genetic sequence to us. For any reference, the same data can still be obtained through a nucleotide blast on the NCBI site by either entering the accession number "AJ717542.1" or the sequence string in appendix 5. After the nucleotide blast we saved the data in a fasta format text file. It is this file that we ran alignment experiments with Mega 7, SDT and Geneious.

B. Experiments

When running a sequence alignment SDT and Geneious were able to read straight from the ".txt" file that we had saved. For Mega 7, we had to use a ".fas" file. We experimented with producing a phylogenetic tree using Geneious because the Mount Makulu researchers told us it was the only one they used for production of the phylogenetic trees. At this stage of our research we focused much on sequence alignment, comparing between that of the three preexisting software tools and that of the biopython libraries, which we tumbled on during our reading.

We implemented both the non-XML file output and the XML-file output of a blast and sequence alignment using biopython libraries. The code is presented in appendices 1 and 4. During our implementation we used PyCharm and Anaconda-Navigator Integrated Development Environments (IDE). We created a Python script to carry out online blast and parsed the handled result into aligned sequences. Our biopython code first searched NCBI for sequences that match that of the ACMV with accession number AJ717542.1 at expectation value threshold less than 0.04. The matched sequences were then aligned against our initial AJ717542.1 ACMV sequence.

After experimenting with online sequence blast and alignment we went on to experiment with carrying out sequence alignments using a file of already downloaded sequences using command line tools. We did this to compare rate of output between that of using online sources and that of using locally stored data.

The results of all experimentations are presented in section VIII (Preliminary Results).

VIII. PRELIMINARY RESULTS

This section presents results, at this stage of our research, from the experiments carried out and described in section VII. After experimenting with three preexisting software tools that are used by local researchers, in Zambia, to study genome data of the ACMV we found that the total time taken to go through all the desired steps was at least nine (9) hours; one process ran infinitely. Sequence alignment alone took an average of 2 hours 4 minutes with each software. When we used libraries from the proposed biopython we ran through all the steps within three (3) hours, which is at most one-third the time it took while using the preexisting software tools. Sequence alignment took an average of 38 minutes. At this stage of our research we implemented the nucleotide blast, multiple sequence alignment and production of phylogenetic trees (evolutionary analysis). Figures 2 to 4 summarise results from our experimentation. Figure 2 is a bar chart showing average time it took to align 92 sequences using the three preexisting software tools (Mega 7, SDT and Geneious). Mega 7 had the best average time of all the three. We then compared Mega 7 results with those of online blast/alignment and alignment from locally saved data. Figure 3 shows that the online blast/alignment had the best duration with an average time of 3 minutes 20 seconds. The alignment of locally saved data gave us an average of 38 minutes 20 seconds.

Generally, we observed that it is possible for us to use the XML standard for data transfer between modules and functions of Biopython. This would make data sharing universal because XML is universal. Figure 6 presents a summary of the proposed solution and computational framework that uses Biopython libraries (modules and functions). File sharing will be done through XML files. One XML output of one module or function will serve as input into the next module or function and this will go on until the final desired output or visual representation. Appendix 4 shows one of the XML output, after a blast. Listing 1 is a XML schema for a PhyloXML evolutionary comparison tree. By the end of this research the final product will be a software tool with graphical user interfaces for easy use by computer science laypersons.

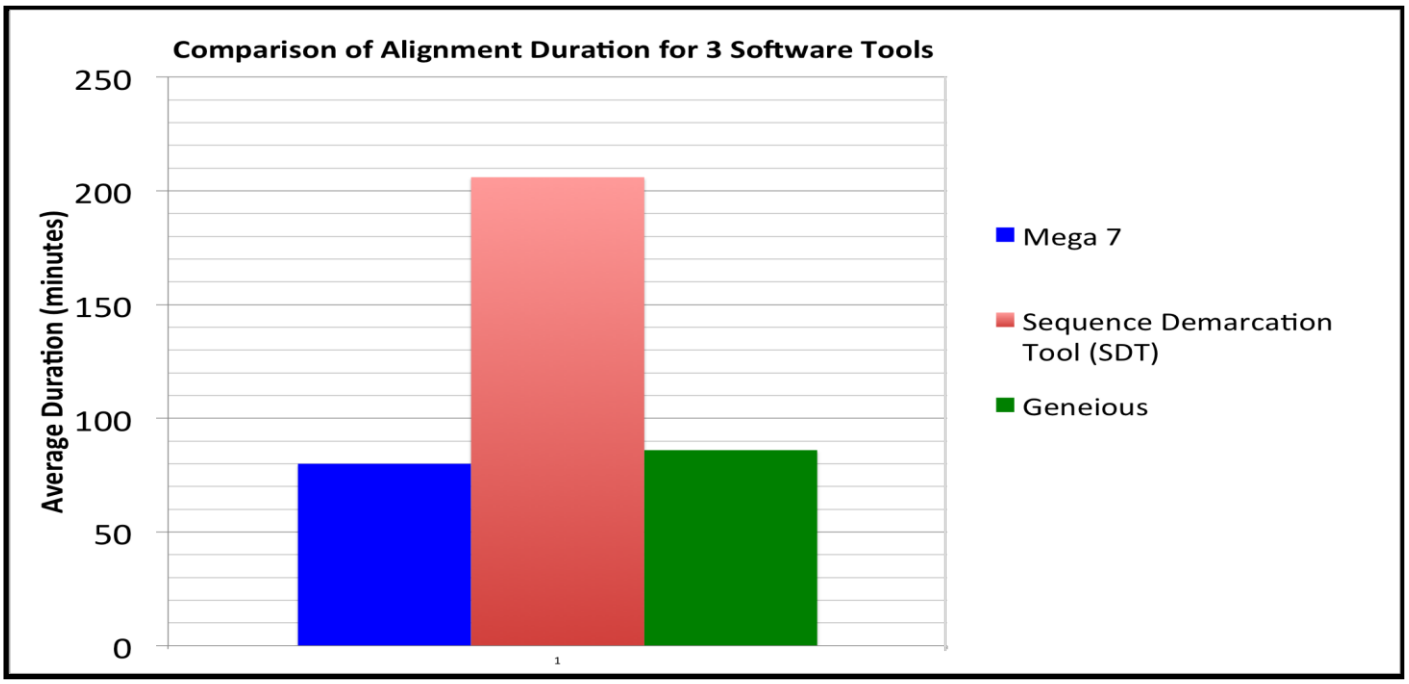

Figure 2: Comparison of average sequence alignment duration for the three preexisting software tools used by Mount Makulu Research Station researchers.

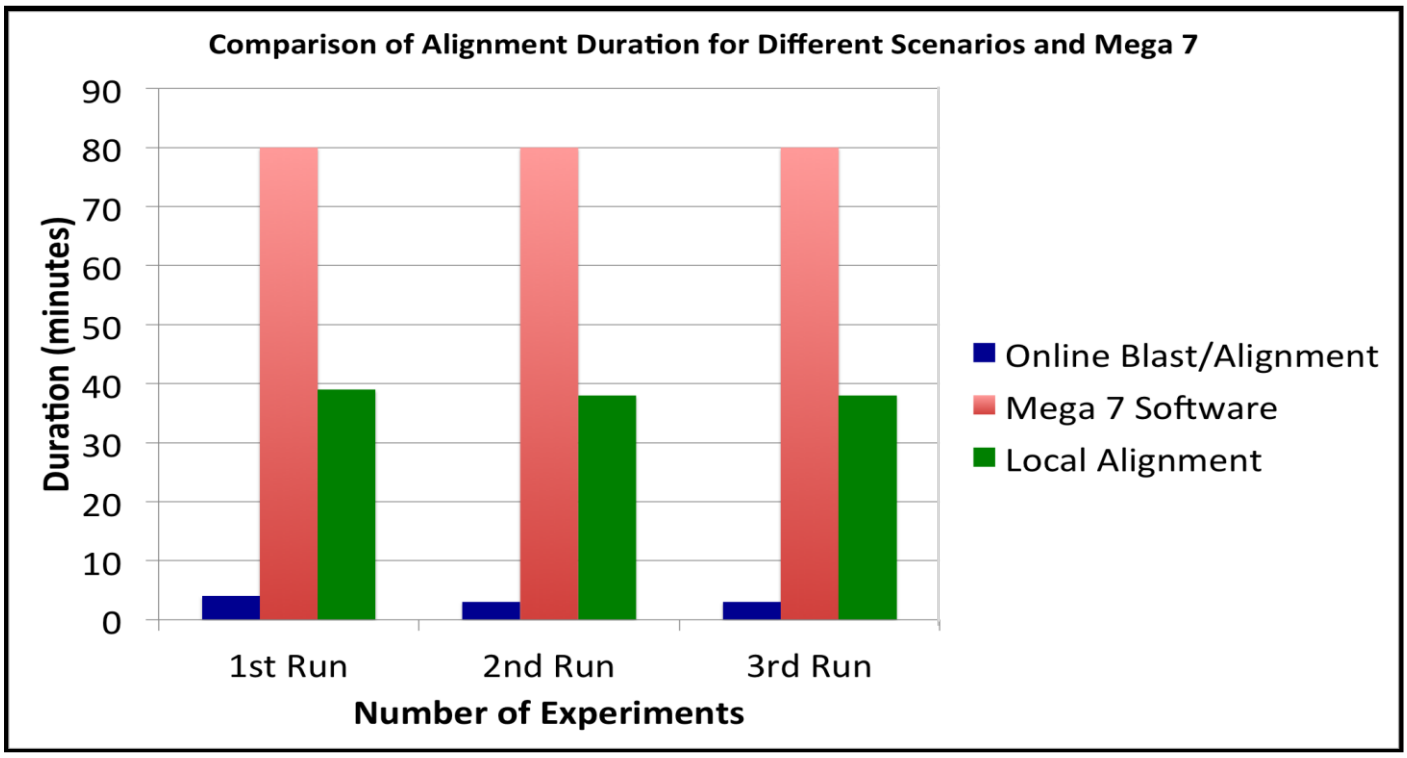

Figure 3: Comparison of sequence alignment duration of Mega 7 against the online blast/alignment and the alignment of locally saved data

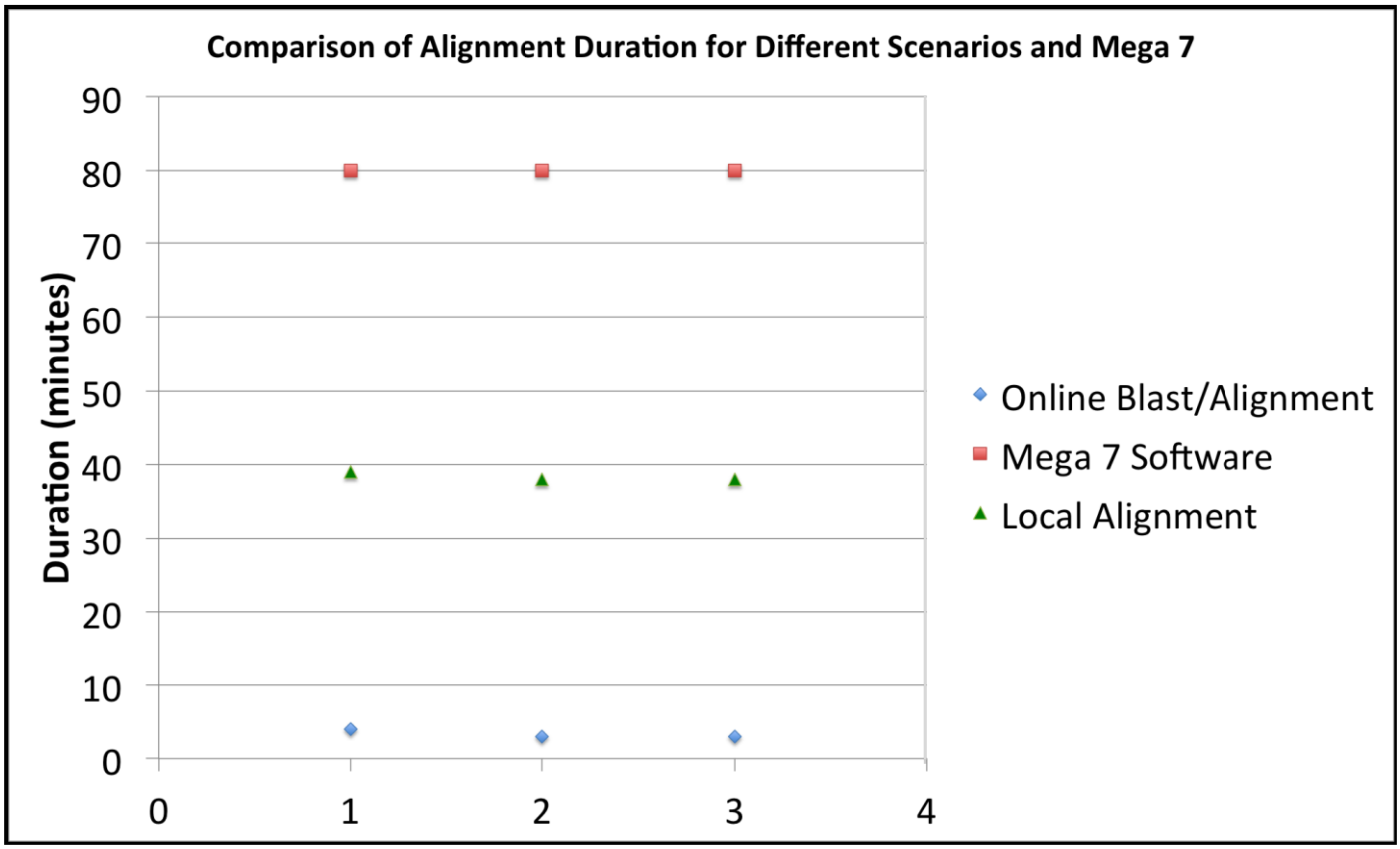

Figure 4: Scatter plot for the data in figure 3, shows stability in output for each scenario

Listing 1: XML Schema for PhyloXML Tree

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsi:schema xmlns="http://www.phyloxml.org" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.phyloxml.org http://www.phyloxml.org/1.10/phyloxml.xsd">
        <xsi:element name = "phyloxml " type = "schema namespace string">
                <xsi:element name = "phylogeny rooted = 'false'" type = "xsi:string">
                        <xsi:element name "clade" type = "xsi:string">
                                <xsi:element name = "clade" type = "xsi:string">
                                        <xsi:complexType>
                                                <xsi:sequence>
                                                        <xsi:element name = "name" type = "xsi:string" />
                                                        <xsi:element name = "branch_length" type = "xsi:string" />
                                                </xsi:sequence>
                                        </xsi:complexType>
                                </xsi:element>
                        </xsi:element>
                </xsi:element>
        </xsi:element>
</xsi:schema>
```

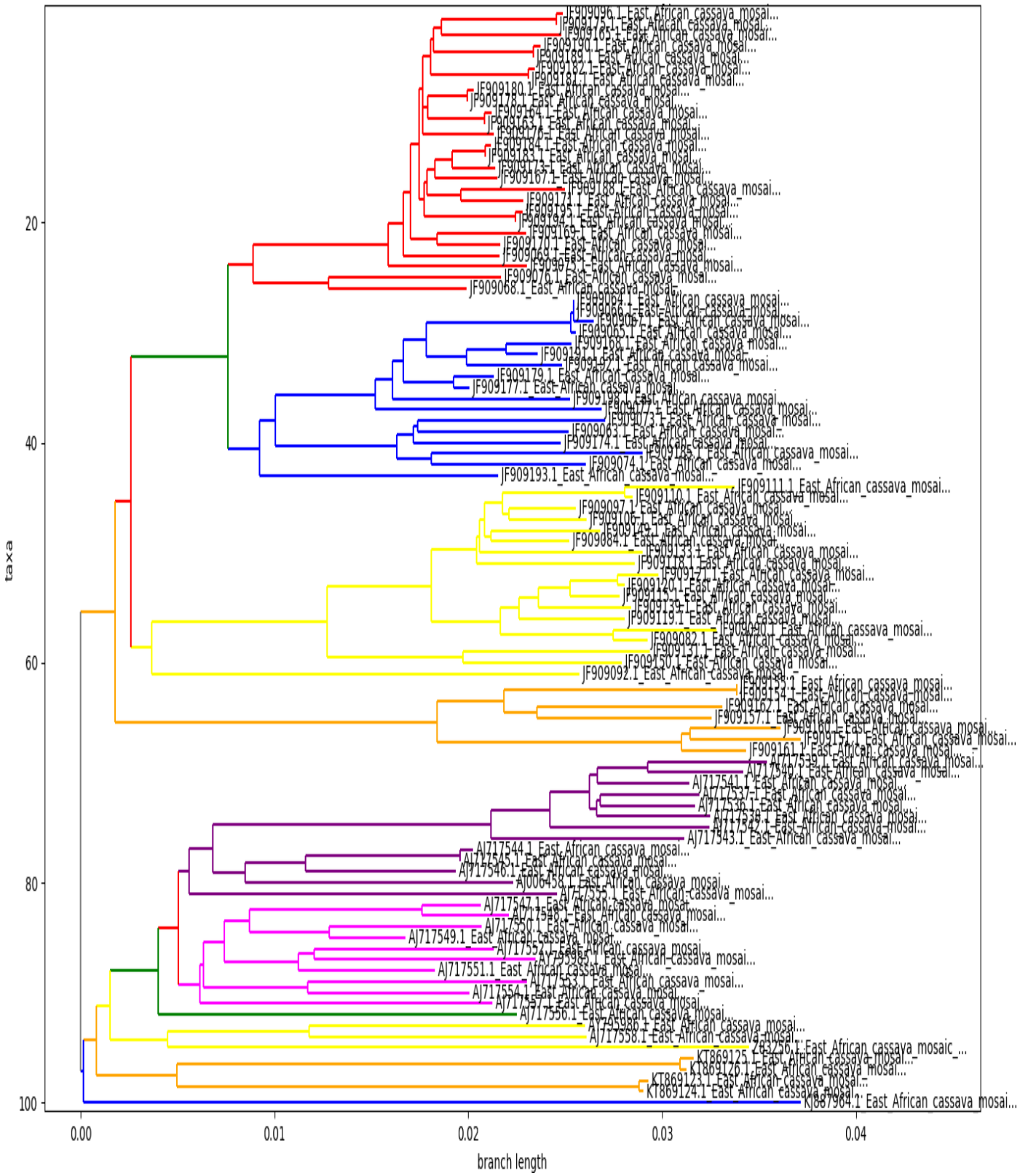

Figure 5: Output of Biopython ascii phylogenetic tree

The implementation of XML blast and saving the output to a file is a step in our desired direction of using XML for file sharing between applications in our framework or between steps when we create a comprehensive and user-friendly tool that will be used to study the mutation of the ACMV. Appendix 4 has the sample code for saving to XML after a blast.

IX. CONCLUSION

In this research we propose a computational framework that uses preexisting tools to offer a comprehensive user-friendly tool that will be used to determine the rate of mutation of the African Cassava Mosaic Virus. After analysis of the steps and procedures involved in the analysis of the mutation of ACMV we further propose the use of Biopython, which has libraries that have many capabilities from sequence alignment to pairwise analysis of genome data and phylogenetic tree production. We observed that downloading the genome libraries to local servers would make genome analysis faster. Nonetheless, the combination of online nucleotide blast and sequence alignment when using Biopython libraries proved to be faster than carrying out sequence alignment using a local genome data library. The use of a local genome library with biopython to carry out an alignment was faster than using the preexisting software tools, which are used by Mount Makulu Research Station researchers. We take note that genome libraries like NCBI have computers optimized for high-speed search of data unlike our usage of our simple RAM strained laptops during our experimentation. We, therefore, conclude that if we create a local optimized scenario we will get results faster than we did during the online blast/alignment. We also observed that the use of XML files would help with the various components of the software tools communicating with each other in form of output to input relationship. Fortunately, Biopython has methods for downloading genome data and can read from a XML file as initial input and at any level of analysis. This is a score to meet our aim of developing a computational framework that uses preexisting tools to offer a comprehensive user-friendly tool. This tool will be used to determine the rate of mutation of the African Cassava Mosaic Virus.

Biological science and agricultural science researchers in Zambia at the time of this research used manual feeding of data into each software tool used to analyse the genome data of the ACMV for its mutation. Developing a comprehensive user-friendly tool that will be used in the study of the mutation of ACMV should improve the rate at which solutions against the virus are provided. This will help in enhancing resistance to the virus, in the cassava, and thus improve cassava yields.

For future work we plan to implement data sharing using an XML based protocol for sharing information between the different stages of the genome data analysis. The blast and alignment output in XML was tested successfully. Using the said protocol, we can implement a distributed solution that takes advantage of high performance architectures and hence enhance performance of the integrated tool. We plan to create local optimized high performance architecture to provide faster output of sequence alignments. We also plan to do a survey to get feedback from the life science researchers on how to better the comprehensive customized tool. In our next report we will also discuss the science behind the sequence analysis steps in terms of the algorithms used to get our code work.

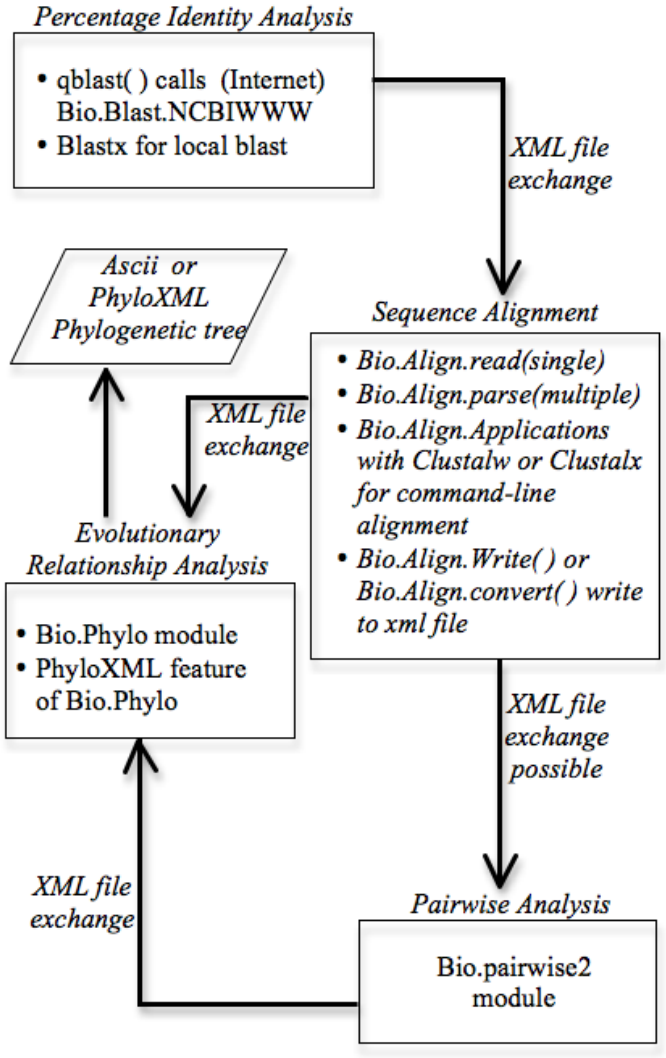

Figure 6: Proposed solution and computation framework of open source tools using Biopython libraries

REFERENCES

[1] C. A. Omongo, R. Kawuki, A. C. Bellotti, T. Alicai, Y. Baguma, M. Maruthi, A. Bua, and J. Colvin, "African cassava whitefly, bemisia tabaci, resistance in african and south american cassava genotypes," *Journal of integrative agriculture*, vol. 11, no. 2, pp. 327–336, 2012.

[2] I. Y. Rabbi, M. T. Hamblin, P. L. Kumar, M. A. Gedil, A. S. Ikpan, J.-L. Jannink, and P. A. Kulakow, "High-resolution mapping of resis- tance to cassava mosaic geminiviruses in cassava using genotyping-by-sequencing and its implications for breeding," *Virus research*, vol. 186, pp. 87–96, 2014.

[3] J. Legg and J. Thresh, "Cassava mosaic virus disease in east africa: a dynamic disease in a changing environment," *Virus research*, vol. 71, no. 1-2, pp. 135–149, 2000.

[4] V. N. Fondong and K. Chen, "Genetic variability of east african cassava mosaic cameroon virus under field and controlled environment conditions," *Virology*, vol. 413, no. 2, pp. 275–282, 2011.

[5] J. Legg, S. Jeremiah, H. Obiero, M. Maruthi, I. Ndyetabula, G. Okao- Okuja, H. Bouwmeester, S. Bigirimana, W. Tata-Hangy, G. Gashaka *et al.*, "Comparing the regional epidemiology of the cassava mosaic and cassava brown streak virus pandemics in africa," *Virus research*, vol. 159, no. 2, pp. 161–170, 2011.

[6] B.L.Patil,B.Bagewadi,J.S.Yadav,andC.M.Fauquet,"Mappi ngand identification of cassava mosaic geminivirus dna-a and dna-b genome sequences for efficient sirna expression and rnai based virus resistance by transient agro-infiltration studies," *Virus research*, vol. 213, pp. 109– 115, 2016.

[7] J. P. Legg, P. Sseruwagi, S. Boniface, G. Okao-Okuja, R. Shirima, S. Bigirimana, G. Gashaka, H.-W. Herrmann, S. Jeremiah, H. Obiero *et al.*, "Spatio-temporal patterns of genetic change amongst populations of cassava bemisia tabaci whiteflies driving virus pandemics in east and central africa," *Virus Research*, vol. 186, pp. 61–75, 2014.

[8] R. Aloyce, F. Tairo, P. Sseruwagi, M. Rey, and J. Ndunguru, "A single- tube duplex and multiplex pcr for simultaneous detection of four cassava mosaic begomovirus species in cassava plants," *Journal of virological methods*, vol. 189, no. 1, pp. 148–156, 2013.

[9] F.Valenzuela-Gonza´lez,M.Mart´ınez-Porchas,E.Villalpando-Canchola, and F. Vargas-Albores, "Studying long 16s rdna sequences with ultrafast-metagenomic sequence classification using exact alignments (kraken)," *Journal of microbiological methods*, vol. 122, pp. 38–42, 2016.

[10] K. Hipp, P. Rau, B. Scha¨fer, J. Pfannstiel, and H. Jeske, "Translation, modification and cellular distribution of two ac4 variants of african cassava mosaic virus in yeast and their pathogenic potential in plants," *Virology*, vol. 498, pp. 136–148, 2016.

[11] T. Alicai, J. Ndunguru, P. Sseruwagi, F. Tairo, G. Okao-Okuja, R. Nan- vubya, L. Kiiza, L. Kubatko, M. A. Kehoe, and L. M. Boykin, "Cassava brown streak virus has a rapidly evolving genome: implications for virus speciation, variability, diagnosis and host resistance," *Scientific reports*, vol. 6, p. 36164, 2016.

[12] Z. Yang, "Phylogenetic analysis by maximum likelihood (paml)," 2000.

[13] F. Pina-Martins and O. Paulo, "Ncbi mass sequence downloader–large dataset downloading made easy," *SoftwareX*, vol. 5, pp. 80–83, 2016.

[14] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (xml) 1.0," 2008.

[15] S. Sprenkle, "extensible markup language (xml)," 2006.

[16] A. Rooney, "Extensible markup language (xml)," *Foundations of Java for ABAP Programmers*, pp. 145–164, 2006.

[17] A. Hoekman, "Journal publishing technologies: Xml," 1999.

[18] E.Pardede, J.W.Rahayu, and D.Taniar, "Xmldataupdatemanagement in xml-enabled database," *Journal of Computer and System Sciences*, vol. 74, no. 2, pp. 170–195, 2008.

[19] S. Abiteboul, G. Gottlob, and M. Manna, "Distributed xml design," *Journal of Computer and System Sciences*, vol. 77, no. 6, pp. 936–964, 2011.

[20] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Computing*, vol. 30, no. 7, pp. 817–840, 2004.

[21] M. Gertz and J.-M. Bremer, "Distributed xml repositories: Top-down design and transparent query processing," *Department of Computer Science*, 2003.

[22] J. Chang, B. Chapman, I. Friedberg, T. Hamelryck, M. De Hoon, P. Cock, T. Antao, and E. Talevich, "Biopython tutorial and cookbook," 2010.

[23] S. Bassi, *Python for bioinformatics*. Chapman and Hall/CRC, 2017.

[24] H. C. Jubb, A. P. Higueruelo, B. Ochoa-Montan˜o, W. R. Pitt, D. B. Ascher, and T. L. Blundell, "Arpeggio: a web server for calculating and visualising interatomic interactions in protein structures," *Journal of molecular biology*, vol. 429, no. 3, pp. 365–371, 2017.

[25] A. P. Hutchins, R. Jauch, M. Dyla, and D. Miranda-Saavedra, "glbase: a framework for combining, analyzing and displaying heterogeneous genomic and high-throughput sequencing data," *Cell Regeneration*, vol. 3, no. 1, p. 1, 2014.

[26] C. Sundaravadivelan, E. Murugesh, M. Preethy, and P. Sivaprasath, "Ariadne merione ecdysone receptor (amecr) protein: An in silico approach for comparison of agonist and antagonist compounds," *Egyptian Journal of Basic and Applied Sciences*, vol. 4, no. 4, pp. 288–296, 2017.

[27] M. Najafi, G. R. Mianji, and Z. A. Pirsaraie, "Cloning and comparative analysis of gene structure in promoter site of alpha-s1 casein gene in naeinian goat and sheep," *Meta gene*, vol. 2, pp. 854–861, 2014.

APPENDICES

Appendix 1: Python code for biopython blast and alignment

```
# coding=utf-8

#Import GUI class
from tkinter import *

#Import color class
from colorama import Fore

from colorama import Style

#Import NCBI online blast module
from Bio.Blast import NCBIWWW

#Import NCBI XML blast module
from Bio.Blast import NCBIXML

#Initialise method to create GUI
blastAlign = Tk()


#Define the blast function
def blast_align():
    """

    :return:
    """
    print("Blasting....")

    #Define the function to retrieve the accession number from the GUI input variable
    def retrieve_input():
        """

        :return:
        """

        accessionNumber = textBox.get("1.0", "end-1c")
        return accessionNumber

    #Retrieve the accession number from the GUI input variable
    retrieve_input()
    #create a data handle for the blast result
    result_handle = NCBIWWW.qblast("blastn", "nr",
                                   retrieve_input(),
                                   word_size=7,
                                   gapcosts='5 2',
                                   nucl_reward=1,
                                   nucl_penalty='-3',
                                   expect=1000)
    blast_records = NCBIXML.read(result_handle)

#Can use the Expectation Value Threshold to limit number of output; here it is commented out below
    # E_VALUE_THRESH = 0.04

    #carry out the alignment after data has been appended to a handle after blast
    for alignment in blast_records.alignments:
        for hsp in alignment.hsps:
            # if hsp.expect < E_VALUE_THRESH:
print(f'{Fore.RED}****Alignment****{Style.RESET_ALL}')
            print('sequence: ', alignment.title)
            print('length: ', alignment.length)
            print('Identity:', hsp.identities)
            print(round(hsp.identities / alignment.length * 100, 2), '% identity')
            print("\033[1;41m" + hsp.query + "\033[1;m")
            print(hsp.match)
            print("\033[36m" + hsp.sbjct + "\033[0m")
    result_handle.close()
    print("")
    print("")

print("**************************************************")
    print("DONE!")

textBox = Text(blastAlign, height=1, width=20)
textBox.pack()
buttonCommit = Button(blastAlign, height=1, width=10,
text="Blast", fg="blue", command=lambda: blast_align())
buttonCommit.pack()
mainloop()
```

Appendix 2: Biopython code for an ascii phylogenetic tree

```
# coding=utf-8

from Bio import Phylo

from Bio.Phylo.PhyloXML import Phylogeny


handle = "Path_to_file/acmv2018.xml"

acmv_tree = Phylo.read(handle, 'phyloxml')

#Phylo.draw(acmv_tree)

acmv_tree.root_at_midpoint()

acmv_tree.ladderize(reverse=True)
acmv_tree.clade[0, 0, 0, 0].color = "red"
acmv_tree.clade[0, 0, 0, 0].width = 1

acmv_tree.clade[0, 0, 0, 1].color = "blue"
acmv_tree.clade[0, 0, 0, 1].width = 1

acmv_tree.clade[0, 0, 0].color = "green"
acmv_tree.clade[0, 0, 0].width = 1

acmv_tree.clade[0, 0, 1].color = "yellow"
acmv_tree.clade[0, 0, 1].width = 1

acmv_tree.clade[0, 0].color = "red"
acmv_tree.clade[0, 0].width = 1
```

```
acmv_tree.clade[0].color = "orange"
acmv_tree.clade[0].width = 1

acmv_tree.clade[1].color = "blue"
acmv_tree.clade[1].width = 1

acmv_tree.clade[1, 0, 0, 0, 0, 0].color = "purple"
acmv_tree.clade[1, 0, 0, 0, 0, 0].width = 1

acmv_tree.clade[1, 0, 0, 0, 0, 1].color = "fuchsia"
acmv_tree.clade[1, 0, 0, 0, 0, 1].width = 1

acmv_tree.clade[1, 0, 0, 0, 0].color = "red"
acmv_tree.clade[1, 0, 0, 0, 0].width = 1

acmv_tree.clade[1, 0, 0, 0].color = "green"

acmv_tree.clade[1, 0, 0, 0].width = 1

acmv_tree.clade[1, 0, 0].color = "yellow"
acmv_tree.clade[1, 0, 0].width = 1

acmv_tree.clade[1, 0].color = "orange"
acmv_tree.clade[1, 0].width = 1

acmv_tree.root.color = "gray"

Phylo.draw(acmv_tree)
```

Appendix 3: Biopython code for outputting a PhyloXML tree

```
# coding=utf-8

from Bio import Phylo

from Bio.Phylo.PhyloXML import Phylogeny

dnd_file_input = " Path_to_file /Grey-Mega7b.dnd"

xml_file_output = " Path_to_file /Grey-
Mega7bPhyloXML.xml"

PhyXMLTree = Phylo.read(dnd_file_input, 'newick')

Phylo.convert(dnd_file_input, 'newick', xml_file_output,
'phyloxml')

phyTreeFromXML = Phylogeny.from_tree(PhyXMLTree)

print(PhyXMLTree)
```

Appendix 4: Sample code for outputting XML after a blast

```
# coding=utf-8

from Bio.Blast import NCBIWWW

result_handle = NCBIWWW.qblast("blastn", "nr",
"AJ717542.1", format_type="XML")
save_file = open("Path_to_file /acmv_blast.xml", "w")
save_file.write(result_handle.read())
save_file.close()
result_handle.close()
```

Appendix 5: Genetic sequence for ACMV with accession number “AJ717542.1”

```
AATGTATCGAAGCCCAGATGTTCCTAAGGGCTGT
GAAGGCCCATGTAAGGTTCAGTCGTATGAACAG
GGGGATGATGTTAAGCACACTGGTATGGTTCGA
TGTGTCAGTGATGTTACGCGTGGGCCAGGCATTA
CCCATAGAGTCGGGAAGAGGTTTTGTGTGAAGT
CCATATATATATTGGGCAAGATCTGGATGGATG
AGAATATCAAGAAGCAAAATCATACGAACCATG
TTATGTTCTTCCTCGTGCGAGATAGAAGGCCTTA
TGGGCCGAGCCCACAAGATTTTGGACAAGTGTT
CAACATGTTTGATAATGAGCCTACTACGGCAACT
GTGAAGAATGATCTTAGGGACCGGTATCAGGTG
TTACGTAAATTCTATGCGACTGTTGTTGGTGGAC
CCTCTGGGATGAAGGAACAAGCTCTGGTTAAGA
GGTTTTTTAGGATCAATAATCATGTAGTGTATAA
TCATCAGGAACAGGCCAAGTATGAGAATCATAC
TGAGAATGCGTTGTTATTGTATATGGCATGTACA
CATGCCTCAAATCCTGTGTATGCTACGCTGAAAA
TACGCATCTATTTCTATGATGCAGTGACAAATTA
ATAAAGGTTGAATTTTATTGCATGTTGCTCCGTA
ACTTGGAGCGTGTTTAGTAATACATCGTACAGAA
CATGATCAACAGATTGAAGTACAGTGTTAATGG
AAATAACGCCTATCATATCTAAATACTTGAGCAC
TTGAGATCTAAATACTCTTAAGAAAAGACCAGT
CTGAGGCCGTAAGGTCGTCCAGACCTTGAAGTT
GAGAAAACACTTGTGAATCCCCAATGCCTTCCG
GATGTTGTGGTTGAACCGTATCTGGATTGTGATG
ATGTCGTGGTTCATGTTCCCTGGCCTCTTGTCGT
GGTTGGTGATTGCGAAATAGAGGGGATTTGTTAT
TTCCCAGGTAAAAACGCCATTCGTTGCTTGAGGC
GCAGTGATGAGTTCCCCTGTGCGAGAATCCATG
GTTGATGCAGTCGATGTGGAGATAGAACGAGCA
GCCGCATTCGAGGTCTACCCGCCTACGTCTGATG
GCCCTGGTCTTCGCTGTGCGGTGTTGGACTTTGA
TGGGCACTTGAGAACAATGGCTCGTGGAGGGTG
ACGAAGGTGGCATTCTTTAAAGCCCAGGCTTTAA
GGGACTGGTTCTTTTCCTCATCCAGAAACTCTTT
ATATGATGATGTTGGTCCTGGATTGCAGAGGAA
GATAGTGGGAATGCCGCCTTTAATTTGAATCGGC
TTTCCGTACTTTGTATTGCTTTGCCAGTCCCTTTG
```

GGCCCCCATGAATTCTTTGAAGTGTTTGAGGTAG
TGGGGGTCGACGTCATCAATGACGTTGTACCAG
GCGTCGTTGCTGTAGACCTTTGGACTGAGATCCA
GGTGTCCACATAAATAATTATGTGGTCCCAATGA
CCTGGCCCACATGGTCTTCCCTGTACGACTATCA
CCTTCTAGAACAATACTGTTGGGTCTCCAAGGCC
GCGCAGCGGAACCCATCACGTTCTCGGAAACCC
AGACTTCAAGTTCCTCAGGAACGTTAGTAAAAG
AGGATGATAAGAACGGACTAACGTAAGTTTGGG
GCGGAGCCTGGAAGATTCGATCTGCGTTAGCAG
ATATGTTATGGAACTGTAAAAAAAAGGACTTGG
GATCTTTTTCTTTGATAATTTGAAGAGCTTCGGA
TTTCGAAGAAGCATTCAACGCGTCTGCATAGACC
TGAGCTAAATGCTGGCCCTCCCCCCTGGCACTTC
GGGCATCGACTTGGAAAATTCCATCGTCAAGAA
ATTCCCCTCCCTTTTCAATGTAAGCCTTGACATC
ACCGGATGGCCGCGCCCGAAAAAGCAGGTGGAC
CCCACCACATGGCCGCACGCGTAAAAGAAAGTG
GTCCCCGCGCACTGGTATTGGTCGGCCAGTCATA
TTCACGCGTGGAAGTCTAGATATTTGTGGGTTGA
CGTTATATACTTCGTCGCGAAGTAGTGGAGCGCG
TCAACATGTGGGATCCATTGTTGAACGATTTTCC
CGAAACCGTTCACGGTTTCCGTTCTATGCTTGCT
GTTAAATACCTGTTACATCTTGAACAGGAATACG
ATCGCGGTACTGTCGGGGCTGAGTATATACGGG
ATCTAATAGGGGTGCTACGGTGTAAGAGTTATGT
CGAAGCGACCAGGAGATATAATAATCTCAACAC
CCGTATCCAAGGTGCGGAGGAGGCTGAACTTCG
ACAGCCCATACACGAACCGTGTTGTTGCCCCCAC
TGTCCGCGTCACCAGAAGCAAAATATGGGCCAA
CAGGCCCATGTATCGGAAGCCCAAGATGTACAG
GGACGATGATTTAGCGCCCTGAATGTTCGGATG
GAAATGTGTTGATCTGGATGGGGAAATGAGATC
GAAGAATCTGGGGTTGGTACATTGGAACTTGCCT
TCGAATTGGATGAGAACATGGAGATGAGGCACC
CCATCCTGATGTAGTTCTCTGCAAACCCTAACGA
ATTTGATATTCGTCGGATAAGCAAAAGCTTTTAA
TTGGGAAAGAGCCTCTTCCTTTGTTAATGAGCAG
CGGGGATAGGTGATGAAATAATTTTTGGCATTTA
TTTGAAAACGACCGGCTCTTGGCATATTTGCTGT
CGTTTTGGATCGGGGGACACTCAAAACTCCAGG
AGAACGGTGGAATGGGGGGCATTATATAGGATG
TCCCCCAATGGCATATGTGTAAATAGGTAGAAG
TCCATTCAAAATTTGAATTGCGAATATTGGCGGC
CATCCGATTAATATT---