DEVELOPING A WORKFLOW TO DETERMINE THE RATE OF MUTATION OF THE AFRICAN CASSAVA MOSAIC VIRUS

by

Grey Chibawe

Dissertation submitted to the University of Zambia in partial fulfilment of the requirements of the degree of Master of Science in Computer Science

THE UNIVERSITY OF ZAMBIA
LUSAKA

2019

COPYRIGHT

Sections of this work have been submitted as manuscripts for publication to the following peer reviewed journal article:

CHIBAWE, Grey et al. Using Open Source Tools for Analysis of the Mutation Rate of African Cassava Mosaic Virus. Zambia ICT Journal, [S.l.], v. 2, n. 1, p. 44-56, june 2018. ISSN 2616-2156. Available at: <http://ictjournal.icict.org.zm/index.php/zictjournal/article/view/50>.

Other sections have been published as part of the conference proceedings and abstracts:

Grey Chibawe, Mayumbo Nyirenda, Lillian Mzyece, and Jackson Phiri - **The Use of Open Source Tools in Research: A Case Study of the Mutation of African Cassava Mosaic Virus**, *Zambia Association of Public Universities and Colleges (ZAPUC) International Conference*, 2018.

Signature: .....

Date: .....

APPROVAL

This dissertation of Grey Chibawe has been approved as fulfilling the requirements or partial fulfilment of the requirements for the award of Master of Science in Computer Science by the University of Zambia

Examiner 1 ..... Signature: ..... Date: .....

Examiner 2 ..... Signature: ..... Date: .....

Examiner 3 ..... Signature: ..... Date: .....

Chairperson
Board of ..... Signature: ..... Date: .....
Examiners

Supervisor ..... Signature: ..... Date: .....

ABSTRACT

Cassava is one of the alternative crops to Zambia's staple crop maize. Unlike maize, cassava is drought resistant; maize has adversely been affected by drought. Unfortunately cassava has been ravaged by the African cassava mosaic virus for over one hundred years. It causes cassava mosaic disease which can result into a reduction in tuberous yield of even up to seventy percent. Biological and agricultural science researchers world over have not yet found a solution to the adverse effects of the virus partly because of the software tools used to study the virus. The tools are not comprehensive, are fragmented and have a complicated workflow. These factors cause the virus to mutate before any solution to its effects is found. The tools used are also mostly proprietary resulting in only few researchers participating in the study of the African cassava mosaic virus. A case study of the research on the African cassava mosaic virus (ACMV) mutation gives the picture of the foregoing. Researchers' efforts to curb its effects are usually outdone by its unpredictable and rapid rate of mutation. This research, therefore, used the ACMV case study to develop a workflow that would be used to determine the rate of mutation of the ACMV. The workflow brings together pre-existing fragmented tools via an XML communication workflow which results into a tool that is easy to use. The analysis of ACMV takes three to four stages: percentage identity analysis through a BLAST, multiple sequence alignment and/or pairwise sequence analysis and phylogenetic tree creation. All the stages, put together, bring out information on whether the ACMV strain being studied has mutated or not. The five pre-existing tools (Bioedit, Geneious, Mega, NCBI and SDT) are not interoperable. They use varying file formats for data input and output although they all include xml. This research proposes the use of open source biopython modules to build applications that would be ably used to study the ACMV. An easier workflow based on xml related protocols is also proposed in order to reduce total response time of the applications used to study the ACMV. The proposed solution was tested and proved to be three times faster than the pre-existing tools used by Mount Makulu research station researchers, in Zambia. A proof of concept on the development of a single page application based on html, xml and svg was done. Ultimately, the reduction

in response time of the applications used to study the ACMV would improve the rate at which mutation in the ACMV is determined and thus help come up with solutions against the virus fast enough to curb its effects and improve cassava yields. The solution to the effects of the ACMV would increase food security in Zambia and other parts of Africa that depend on cassava as staple food.

Keywords: African cassava mosaic virus; bioinformatics; biopython; computational framework; data sharing techniques; HTML; metagenomics; open source tools; software; SVG; XML; workflow

DEDICATION

This dissertation is dedicated to

God Almighty

and

my beloved family and friends.

A special dedication is given to my beloved wife Nchimunya Chibawe,

who endured the impact of a husband in school and sacrificed resources in many ways to make the schooling a success.

ACKNOWLEDGEMENTS

I am highly indebted to Dr. Mayumbo Nyirenda (lecturer at the Computer Science Department of UNZA) who unreservedly supervised this research. Without his guidance this research would have been less comprehensive.
The work of Dr. Jackson Phiri, who co-supervised the research, is greatly appreciated.

The research would never have been the same without Mr. Rabson Mpundu Mulenga who inspired the research due to his and his colleagues' work on the African Cassava Mosaic Virus at Mount Makulu Research Station in Zambia. I am greatly indebted to his contribution to the research; he provided the initial data and biological science guidance that was used in the research.

I am also indebted to my employer at Thornhill Boarding and Day School who allowed me time to study.

I value the precious time I shared with my fellow postgraduate students, particularly Ms. Lillian Mzyece; because of the drive we gave each other we pushed to the end of the programme.

Last but not the least, I am greatly indebted to my wife who gave me full support throughout the study. Plenty resources that she sacrificed in order for me to get to the end of the programme.

May God bless all the aforesaid contributors to the research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

ACRONYMS

ACMV	African Cassava Mosaic Virus
Ai	Artificial Intelligence
BLAST	Basic Local Alignment Search Tool
CGI	Information Technology Consulting Group
CMD	Cassava Mosaic Disease
CP	Coat Protein
DNA	Deoxyribonucleic Acid
DTD	Document Type Declaration
GIF	Graphical Interchange Format
GIS	Geographic Information System
HSP	High-scoring Segment Pair
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
MP	Mammalian Phenotype
MEGA	Molecular Evolutionary Genetics Analysis
MMRS	Mount Makulu Research Station
NCBI	National Centre for Biotechnology Information
NSP	Nuclear Shuttle Protein
RAM	Random Access Memory

SDT	Sequence Demarcation Tool
SGML	Standard Generalised Markup Languages
SPA	Single Page Application
SVG	Scalable Vector Graphics
TSM	Travelling Salesman
XML	eXtensible Markup Language
XSLT	eXtensible Stylesheet Language Transformations
W3W	World Wide Web Consortium

CHAPTER 1

INTRODUCTION

Chapter one is an introduction to the research. The chapter begins with the introduction to the research followed by the statement of the problem then the aim of the study and objectives followed by the research questions, significance of the study and finally the summary of the entire chapter.

1.1 Introduction To The Research

Development, world over, is driven by relevant research. Research needs necessary tools in order to be successful and yield results that are relevant to development. Most of the times these tools may be expensive and difficult to obtain for the researcher who strongly desires to use them. Many researchers now rely on computerised tools to carry out research. These tools may be used in statistical analysis of results. Most tools that bring out appreciable results are commercial and thus expensive to many researchers, especially student researchers. The average of cheapest monthly subscription fees for software tools may be about twenty-five dollars, which may be unaffordable for researchers in low income countries. Thanks to open source policies that have made available software tools at no cost to the user. Such tools may have limitations but they most of the times bring out the intended results. For this research, a case study of the African cassava mosaic virus (ACMV) mutation gives the picture of the foregoing. The research was triggered by the need at Mount Makulu research station (MMRS), in Zambia, to determine the rate of mutation of the ACMV. The focus of the research was on the software tools used by researchers at Mount Makulu research station in Zambia to analyse the ACMV. Based on

a pen and paper interview with the researchers, it was established that the tools (Bioedit, Geneious, Mega and SDT), which they use, are fragmented, are proprietary and have a complicated workflow. One of the research works they have been using the tools for is the study of the mutation rate of the ACMV. It is for this reason that this research used the ACMV case study to develop a workflow that would be used to determine the rate of mutation of the ACMV.

Due to climate change, most of southern Africa is resorting to climate adapting crops or early maturing varieties for food security. In Zambia, sorghum, millet and cassava are the alternative to the country's long standing staple food maize. Unfortunately, cassava has not been spared from disease though being a better alternative to maize. It is the diseases that have kept Zambian researchers trotting to discontinue the effects of the cassava diseases or cut the life cycles of the organisms that cause the diseases. 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. The ACMV was first reported in the Asian cassava where it had biotic competitors that kept it under check. In the biological world, an organism when in its natural and original environment has competitors that will keep it under check ecologically. Usually, when an organism is taken to an environment where such competitors are absent, and it establishes well, it can reap havoc if it is a pest or parasite. Most of these transfers of such organisms are done by human travellers. The ACMV is believed to have been brought to Africa through imports that could have innocently aimed at enhancing cassava production.

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. This research, therefore, focussed on developing an easier workflow using pre-existing software tools in order to de-fragment them. Simplifying the workflow when studying the ACMV would reduce total response time of software used and consequently help determine the rate of mutation of the ACMV before its next mutation. This would eventually allow biological and agricultural scientists in at MMRS to curb the adverse effects of the ACMV and eventually improve food security. The de-fragmenting of the pre-

existing software used by the MMRS researchers was dependent on developing protocols that would get them to communicate and automate data transfer between the tools.

1.2 Statement Of The Problem

Researchers in Zambia have ardently sought to find a solution to the dwindling cassava yield caused, to a large extent, by the ACMV. Their perceived solutions are always out done by the ACMV's unpredictable and rapid rate of mutation. Studies on the rate of mutation are carried out by performing DNA multiple sequence alignments using software such as the National Centre for Biotechnology Information (NCBI) online platform, Bioedit, Mega 6, Sequence Demarcation Tool (SDT) and Geneious. These software tools can each bring out portions of desired results. The researchers have to manually feed data from tool to tool because these tools do not communicate with each other and some do not have web service capabilities. The software tools they use generally take in different file formats, as input, which makes it difficult for them to automatically feed data into each other. They are also commercial, created by different organisations or programmers whose prime reason for creating them is money making through monthly or annual subscription and high prices if their license is once off. The foregoing factors make these software tools first expensive to acquire, worse for student researchers and individual professional researchers. Secondly, the software tools' failure to bring out all but some desired results increases the cost of research because researchers have to rely on more that one tool to get through a full process of results. Thirdly, the use of several tools that do not communicate to each other delays any final output for action of controlling the ACMV to take place. If these factors are not solved cassava yield in Zambia and Africa will keep dwindling to extents where the problem may affect communities that depend on the cassava as staple food.

Therefore, 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 and software tools which are fragmented and have a complicated workflow.

1.3 Aim Of The Study

This research sought to contribute to finding a solution to the African cassava dwindling yields which is mostly caused by the ACMV. To do so, I decided to use the case study of the ACMV mutation to propose affordable software which is de-fragmented and has a less complicated workflow. The proposed solution would help biological and agricultural science researchers work fast enough to sever the effects of the ACMV on the African cassava. I, therefore, aimed at developing a workflow that uses pre-existing tools to offer a de-fragmented tool that has a less complicated workflow to be used to determine the rate of mutation of the African cassava mosaic virus.

1.4 Objectives

My research was guided by the following three objectives, which when attained would help me succeed at achieving the aim.

1. To conduct a study of on-line and standalone software tools used by Zambian agricultural and biological science researchers to analyse the ACMV genome.

2. To propose a workflow for the prediction of the mutation of the ACMV.

3. To develop a comprehensive tool that has a less complicated workflow and uses the tools and workflow in objectives "1." and "2." respectively.

1.5 Research Questions

Through the research, I needed to answer the following questions as I traversed the objectives in order to achieve the aim of developing a workflow that uses pre-existing tools to offer a comprehensive user-friendly tool that would be used to determine the rate of mutation of the ACMV.

1. Which of the pre-existing on-line and standalone software tools used to analyse ACMV genome data are generic enough to be ported into a comprehensive tool?

2. What file exchange formats are fundamental for developing protocols for a workflow, from existing tools, in order to determine the mutation rate of the African cassava mosaic virus?

3. How feasible is a comprehensive tool that does not require a user to switch from one application to the other?

1.6 Significance of The Study

The ACMV is one of the major causes for the dwarfing of the African cassava yield. Researchers, in Zambia, have been working ardently to control the dwindling yield of the African cassava caused, to a large, extent by the ACMV. If a solution to the adverse effects of the ACMV is found, cassava would comfortably join the list of crops that sustain Zambia and Africa's food security. A break through in curbing the negative effects of the ACMV would help the researchers come up with cassava varieties which are resistant to the virus. The rate at which the ACMV mutates has been a major hindrance to a break through in controlling the adverse effects of the virus. Its mutation rate is rapid and unpredictable. A solution could have probably been found already had the software tools used in the ACMV mutation rate research been non fragmented and had an easy workflow. The software tools the biological and agricultural science researchers at MMRS currently use do not talk to each other and each mostly brings out only a single desired result out of the many desired results like DNA multiple sequence alignment, pairwise alignment, and phylogenetic trees. The current software tools and their possible outputs are explained in chapter two. Most of these software tools are commercial and quite dear in price. If they were very affordable, the number of researchers working to tackle the problem of the ACMV would increase. Students could also afford to get them and join in the fight. This would result in consented effort which definitely would yield sure and quicker results. This is the reason why the research aimed at developing a workflow that uses pre-existing tools to offer a de-fragmented tool with an easy workflow, that would be used to determine the rate of mutation of the African cassava mosaic virus.

Summary

Cassava is one of the crops which can potentially compliment Zambia's staple food, maize, in providing food security. Its potential is affected by its dwindling yields caused by the adverse effects of the ACMV. Biological and agricultural science researchers who seek to find a solution to the dwindling yield in cassava have not succeeded due to the virus' rapid and unpredictable rate of mutation compounded by the lack of affordable, de-fragmented software tools that have an easy workflow, in their reach. The research, therefore, aimed at developing a workflow that uses pre-existing tools to offer a de-fragmented tool, with an easy workflow, that would be used to determine the rate of mutation of the African cassava mosaic virus.

CHAPTER 2

PRELIMINARIES

Algorithms existed before computers but algorithms exist at the heart of computing today [3]. Algorithms have been used in many areas of computing like developing tools to analyse data in databases, clever algorithms that make it possible to manage and manipulate large volumes of Internet data, numerical algorithms and number theory for e-Commerce as examples [3]. Which algorithm is used depends on the algorithm that brings out results or carries out a task faster than the other. Analysing plenty data requires algorithms that are less tasking on random access memory (RAM). Where possible, frequently used data must be cached for faster response. Computer science uses the Big-O Notation to determine resource usage and efficiency of an algorithm. Some of the Big-O notations are presented in section 2.1. Section 2.2 goes on to describe algorithm design techniques which offer options on an algorithm to use for any given task. This chapter winds up with the biology of the ACMV. Many bioinformatics books focus on parameter setting and other details without explaining the idea behind algorithms used. It is, however better to present the principles that drive the algorithm's decision [4]. In the research I strive to present such principles before getting to use the algorithms themselves. A Bioinformatics algorithm is a combination of molecular biology and computer science, both with elaborate terminology and nomenclature. Generally, nature employs algorithmic procedure in its systems. An example is the replication of DNA, which will require a cell to make a complete copy of itself before dividing [4].

2.1 The Big-O Notation

An algorithm is a well-defined procedure for a computer to solve a problem. It is a set of step-by-step computer instructions for any computer programme. An algorithm used for any computational task determines the time taken to complete a particular task and the memory resources required. Consequently, the two attributes of an algorithm determine performance of the computer when carrying out a given task. It is, therefore, always important to consider what algorithm would be used to complete a given computational task. Most algorithms are either iterative or recursive. Which one is more costly and performs better between iterative and recursive algorithms depends on the task at hand and how it has been programmed. Computer science uses the Big-O notation for algorithms. The Big-O notation is used to represent efficiency and performance of an algorithm. Big-O notation is used to indicate how the running time and space grows with increase in input size. It is mostly used to show the worst case situation of an algorithm. Some algorithms are linear while others are quadratic, logarithmic or exponential. Linear algorithms are denoted by $O(n)$. Logarithmic ones are denoted $O(n\log n)$. Quadratic algorithms use the $O(n^2)$ Big-O notation. The notation just takes into account a part of the algorithm which is bound to take most time. Constants are usually ignored. Two algorithms of different Big-O notation may take almost the same time to process a task. For example exponential algorithms may be faster with less input and more costly as input size grows. Therefore, in determining which algorithm to use, many factors, different with each algorithm, must be considered. Input size and whether the algorithm is iterative or recursive are some of the factors computer software developers consider when selecting an algorithm to use. Table 2.1 lists most of the Big-O notations. In Table 2.1, k is an arbitrary constant. Sub sections 2.1.1 to 2.1.4 briefly explain the main Big-O notations to show which algorithms are likely to be more or less efficient than the other or which ones are likely to be more expensive. Sub section 2.1.5 shows how Big-O functions are assembled.

2.1.1 Constant Function Algorithms

A Constant function algorithm is unaffected by input size. Most of the time such algorithms output non-iterative and non-recursive lines like in the example listing 2.1.

Table 2.1: Big-O Notation and corresponding algorithm types

Big-O Notation Order	**Name**	**Example Formula**
$O(1)$	Constant	$T_n = 2$
$O(\log(n))$	Logarithmic	$T_n = \log_k n$
$O(n)$	Linear	$T_n = 2n + 1$
$O(n^2)$	Quadratic	$T_n = n^2 + 2n$ - 1
$O(k^n)$	Exponential	$T_n = k^n$

Whether input size is 10 or 1000, the time taken and space used will not change. In most algorithms, the constant functions are components like the assigning of a value to a variable, adding two variables, comparing two numbers and many more such operations. These operations are instructions that are executed a fixed number of times requiring constant time allocation. In certain cases, constant functions can be stored in cache memory in order to constantly retrieve them from there than to re-assign values each time the programme runs. An algorithm that has more of a constant function instruction saves on time but may not really save on memory allocation depending on the size of the variable.

Listing 2.1: Constant function algorithm example

```
1  void printFirstElementOfArray(int arr[])
2  {
3  printf("First element of array = %d",arr[0]);
4  }
```

2.1.2 Linear Function Algorithms

In algorithms that have linear functions, constants are only effective to the output size when the input size is small. As input size grows, constants become unnoticeable and irrelevant to the performance of the algorithm. Output of linear function algorithms grows with increase in input size. Some loops are usually linear. In listing 2.2, if the input is 10, the function will print out 10 strings, if the input is 1000, the function will print out 1000 strings. So, the output, time taken and space usage grows linearly. Linear regression, which is one of the basic functions used in Artificial Intelligence (Ai), is an

example of a linear function algorithm. The growth in Ai shows the increased usage of the linear function algorithms in our day to day operations. Stock brokers use linear regression based Ai to predict future market prices. Doctors use similar Ai to determine whether a tumour is malignant or benign. Weather prediction by meteorologists uses Ai [5]. Even Human Resource recruiters may use Ai to check whether the curriculum vitae of applicants meet the minimum prescribed requirements before being considered for a short list. This is a sign of the wide use of the linear function algorithms in our day to day life. They may, however, be integrated into other complex hybrid algorithms.

Listing 2.2: Linear function algorithm example

```
1  void printAllElementOfArray(int arr[], int size, String f)
2  {
3  for (int i = 0; i < size; i++)
4      {
5      printf("%d\n", arr[i]);//This loop grows with input size
6      }
7  }
```

2.1.3 Quadratic Function Algorithms

Algorithms with a quadratic nature result from nested loops or recursive functions. Recursive functions are functions that call themselves until a termination condition is attained. Quadratic function algorithms may even be a combination of some linear component, constant and a recursive component. Since the Big-O notation is determined by the component of the function that grows faster than the rest, the other components become negligible. If the array in listing 2.3 has n items, the outer loop runs n times and the inner loop also runs n times, hence the n^2 in the function. An example of the quadratic function algorithms is the bubble sort algorithm which steps through a list of unsorted items, compares two adjacent items, sorts them in order if they are not in order and goes back through the list again and again until all items are sorted. Most sort algorithms are usually quadratic or sub-quadratic in time expenditure. An example of an algorithm that is described by a quadratic function algorithm is the genetic sequence alignment which is given the big-O notation of the order $O(nm)$, where n and m are sequences.

Listing 2.3: Quadratic function algorithm example

```
1   void printAllPossibleOrderedPairs(int arr [], int size)
2   {
3   for (int i = 0; i < size; i++)
4       {
5       for (int j = 0; j < size; j++)
6           {
7           printf("%d = %d\n", arr[i], arr[j]);
8           }
9       }
10  }
```

2.1.4 Exponential Function Algorithms

The recursive calculation of the Fibonacci numbers (listing 2.4) is a good example of an Exponential function algorithm. The output number reduces at each iteration until the termination condition is met. Most fast algorithms run in polynomial time while slow algorithms run in exponential time. BLAST uses heuristics which is a polynomial algorithm. As much as exponential function problems are bound to be slow, improvements on them are easily visible compared to improvements on polynomial function algorithms. Polynomial function algorithms may be too fast to observe improvements. Exponential function algorithms can be improved by making them dynamic. The travelling salesman (TSM) problem using dynamic programming is another example of the exponential function time algorithm. The TSM problem looks at the shortest way to traverse through a set of cities, touching each only once, and get back to the city of origin.

Listing 2.4: Exponential function algorithm example

```
1   int fibonacci(int num)
2   {
3       if (num <= 1) return num;
4       return fibonacci(num - 2) + fibonacci(num - 1);
5   }
```

2.1.5 How to Determine The Big-O Notation

Listing 2.5 is used to give an example of how to determine the Big-O notation of an algorithm. Listing 2.5 code does nothing but can give an idea of how to determine the Big-O notation for an algorithm.

Listing 2.5: Example Quadratic algorithm

```
1   void bigONotaion (int numbers)
2   {
3   int a = 5
4   int b = 6
5   int c = 10
6   for (i in range(n))
7   {
8       for (j in range(n))
9       {
10          x = i * i
11          y = j * j
12          z = i * j
13      }
14  }
15  for (k in range(n))
16  {
17      w = a*k + 45
18      v = b*b
19  }
20  d = 33
21  }
```

When the code runs, first there are three constants (value assignment to the three variables a, b, and c). The function value of these assignments is 3. Next, there is a nested loop with three different variables. The nesting once gives n^2 and because the values are three different ones the result is $3n^2$. The third one is a non-nested iteration through the loop with 2 different variables giving the value of $2n$. Finally, there is a closing constant assignment of 33 to the variable d. All put together we have $3 + 3n^2 + 2n + 1$ which simplifies into $\mathbf{3n^2 + 2n + 4}$. It is vital to understand the *Big-O* determination when developing applications based on a chosen set of algorithms or a chosen algorithm. It will

definitely help make the application efficient at use of resources like time and memory.

2.2 Algorithm Design Techniques

This section discusses the various algorithm design techniques available to bioinformatics. The sets of algorithm design techniques are exhaustive search, greedy algorithms, dynamic programming algorithms, divide-and-conquer algorithms, graph algorithms, combinatorial pattern matching, clustering and trees, hidden Markov models and randomised algorithms [4]. Not all these algorithms are of interest to the study. Later, the research focuses on algorithms that describe what happens to obtain a BLAST, a multiple sequence alignment, a pairwise alignment and a phylogenetic tree diagram. Many algorithms share similar ideas despite their varied output. Each algorithm under a specific set may have subsets.

2.2.1 Exhaustive search algorithms

The Exhaustive search algorithms search all parts of any subject ignoring any possible leads. This type of algorithm is also called the brute force algorithm. [4]. It takes too much time but assures results because all areas of a subject are searched. It is therefore one of the worst choices of all algorithm types because it is wasteful on time. For example, if you walk into your unlit three storey house and you hear your phone ringing (Figure 2-1), you have the choice to isolate where sound is coming from or to search the whole house floor by floor and part by part ignoring the source of sound until you find it (Figure 2-2). You will take a lot of time doing so but you are sure to find the phone. The earlier algorithms for genome database search used brute force. They were efficient in terms of output because their sequence matches were always correct but response time was longer. This is why they were modified into faster heuristic and dynamic algorithms which are probabilistic.

Figure 2-1: Phone search algorithm. - “Adapted from [4]”

Figure 2-2: Brute Force Algorithm: search everywhere until you find the phone. - “Adapted from [4]”

.

2.2.2 Branch-and-Bound Algorithms

Branch-and-Bound type of algorithms are able to rule out areas of unlikelihood thereby reducing on time taken for a search. Such algorithms are more efficient but performance in terms of results may be bad. Using the example of the ringing phone, with this type of algorithms, if you hear the phone ringing from above you you will ignore searching the ground floor and concentrate your efforts on searching the top floors. There could be chances that your hearing was not right, so the phone might not be on the top floors. But if it is on the top floors your success in finding it will take less time. Your path of search may also have dangerous obstacles since the room is not lit (Figure 2-3). The branch-and-bound algorithm found its application in creation of evolutionary phylogenetic trees [6]. It is a faster algorithm than brute force algorithm.

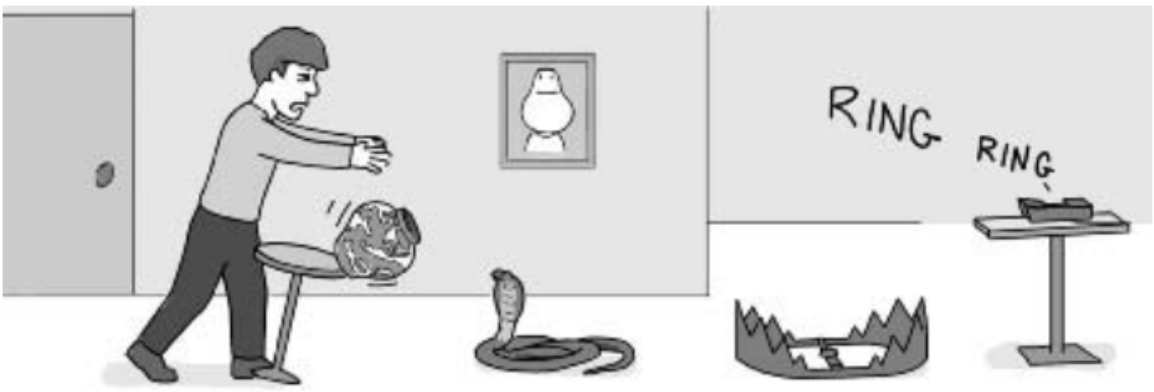


Figure 2-3: The Branch-and-Bound search algorithm can meet with obstacles. - "Adapted from [4]"

2.2.3 Greedy Algorithms

Greedy algorithms choose the most attractive part of the search as the first to start with then down the line of less attractiveness iteratively. This type of algorithms has chances of being successful and as well as being unsuccessful. Here, an example of a seller giving change to a customer is used to illustrate the algorithm (Figure 2-4). Imagine a seller needs to give out half a dollar change to a customer. The seller will focus on giving less number of coins by starting with the largest denomination coins first. For example, if the only coins that exist are 20 cents, 10 cents, 5 cents, 2 cents and 1 cent, the seller will choose to give two times 20 cents and a 10 cent coin rather than fifty times 1 cent coins. This algorithm chooses the best way to go about a solution. If the seller does not have 20

cents coins, he will have to redefine his strategy until he has the best working combination. Time may be wasted in redefining in case of no success in finding coins that fit the first combination. The greedy algorithm has been used in multiple sequence alignment [7] and phylogenetic tree generation [8]. It is a fast algorithm because of beginning a search from an area of much likelihood.

Figure 2-4: Seller giving change as example of Greedy Algorithm. - "Adapted from [4]"

2.2.4 Dynamic Programming Algorithms

Dynamic programming algorithms are simply exhaustive search algorithms but ones that avoid re-computation of already known values [9] [10]. Already computed values are stored for frequent retrieval. This is where cache comes handy to hold frequently sought data or information. This saves time which could be wasted by algorithms that break problems into small sub problems. The sub problems may become larger and therefore time wasting. A Dynamic programming algorithm keeps track of the best way to carry out a step in an activity. This set of algorithms is linear in terms of Big-O notation. Dynamic programming algorithms are growing into popular use now because they are able

to modify recursion to improve response time and efficient use of computer resources. This type of algorithms fits well with distributed systems also. Genome sequence alignment uses this type of algorithms [11]. An example Dynamic programming algorithm is the Knapsack Problem [12]. Given a homework example which has different segments A through G and each segment is valued using points and has size (time in hours) as given in Table 2.2.

Table 2.2: A homework example as example Knapsack problem

Segment	A	B	C	D	E	F	G
Value	7	9	5	12	14	6	12
Size	3	4	2	6	7	3	5

Given also that there is only 15 hours to complete the homework. It is, therefore, wise to choose segments with the best points per size in order to score most points from the homework. The choice of questions can be done recursively by trying out every possible combination to see which ones bring out the most points per size (listing 2.6). Definitely, the time taken to find the best combination is exponential.

Listing 2.6: Recursive Knapsack problem with exponential response time

```
1   // S = space left, n = # items still to choose from
2   {
3           if (n == 0) return 0;
4           if (s_n > S) result = Value(n-1,S); // nth item which can't be used
5           else result = max{v_n + Value(n-1, S-s_n), Value(n-1, S)};
6           return result;
7   }
```

But, there are only $O(nS)$ different value segments available. With dynamic programming, these can be memorised and the algorithm modified to omit any combination already stored. Each combination only passes through the array twice recursively. We, therefore, have at most $2n(S + 1)$ recursive calls with total time of $O(nS)$. The modification is shown in listing 2.7.

Listing 2.7: Dynamic programming of the Knapsack problem

```
1   Value(n,S)
2   {
3           if (n == 0) return 0;
4           if (arr[n][S] != unknown) return arr[n][S]; // <- added line of code
5           if (s_n > S) result = Value(n-1,S); // nth item which can't be used
6           else result = max{v_n + Value(n-1, S-s_n), Value(n-1, S)};
7           arr[n][S] = result; // <- another addition
8           return result;
9   }
```

2.2.5 Divide and Conquer Algorithms

This type of algorithms subdivides a problem into smaller units which may also recursively be subdivided until it is not possible to do so any more. The subunits are solved recursively (conquer) and the solutions to the subunits are recombined as a large solution for the initial big problem. Recombination may take a lot of time. Divide and Conquer saves space but may not always save time [13]. *Merge Sort* is an example of a Divide and Conquer algorithm as given in listing 2.8. DNA sequence alignment generally uses the Divide and Conquer algorithm [14] [15].

1. A sequence is first divided into two sub sequences ($n/2$).
2. The two sub sequences are sorted recursively.
3. The two sorted sub sequences are later combined to produce one sorted sequence.

This algorithm produces a linear time complexity $O(n/2 + n)$.

Listing 2.8: Dynamic programming of the Knapsack problem

```
1   arr[ ] = 1.... n
2   {
3       Sorted arrays i = K[1...n/2], \textit{j} = L[n/2+1...n]
4       Merged sorted array = 1...n/2... n
5       for \textit{t = 1 to n/2 to n}
6         {
7           if (i $\leq$ n/2 and (j $>$ n or K[i] $<$ L[j]))
8           then {M[t] = K[i], i = i + 1}
9           else {M[t] = L[j], j = j + 1}
10        }
11  }
```

2.2.6 Machine Learning Algorithms

Machine learning algorithms base their search on previous more frequent statistics. The search starts in a more frequent to a less frequent area to save on time. For example if you have been placing your missing driver's licence in the bedroom 75% of the year, in the living room 20% of the year and in the study room 5% of the year, you would save time by beginning your search for it in the bedroom followed by the living room and lastly the study room. This is because there is higher probability that it could be in the bedroom. Machine learning algorithms are probabilistic and are based on a feature having a certain value [16] [17]. Situations with higher probability of having a certain value are memorised for future use. By this, time for determining the given previous highly probable situation is saved. Machine learning algorithm has been used in genetic sequence alignment [18].

2.2.7 Randomised Algorithms

The other algorithms explained in sub sections 2.2.1 to 2.2.6 are deterministic. Randomised algorithms are non-deterministic. They are like tossing a coin or rolling a die to decide where the search may begin. They have their place in computer programming but definitely not the most efficient way to carry out a search. Coin tosses or dice rolls may affect the correctness of the result of the randomised algorithm, its running time, usage of space and other factors. The success of the randomised algorithm depends on

higher replication of the coin toss. The more tosses we have, the higher the probability of being successful. The Randomised algorithm can be used for string matching like the DNA sequence alignment [19]. Randomised algorithms are fast and usually very simple to run. They take less computer space, too. By *Big-O Notation*, randomised algorithms are generally linear (O(N)). In computer terms, a randomised algorithm receives, with the input data, random beats from which to take random choices. Randomised algorithms gather information about the spread of the input data by taking random samples of the input data.

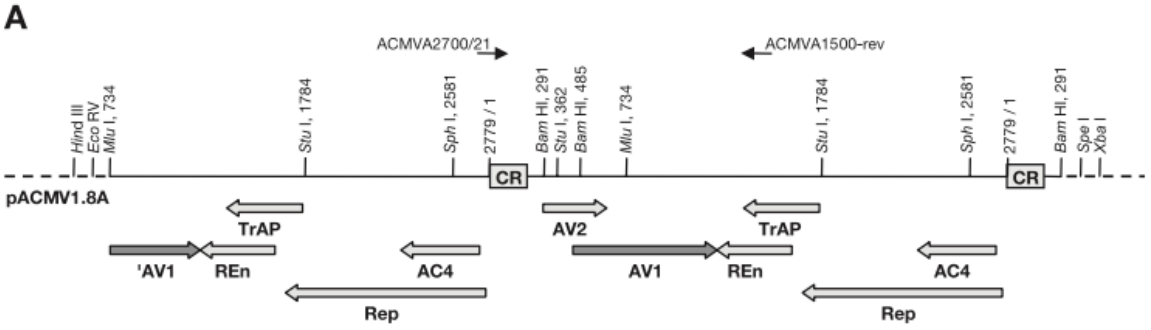
2.2.8 Heuristic Algorithms

Heuristic algorithms are a modification of deterministic algorithms like the dynamic programming, exhaustive search and greedy algorithms. The modification is to create bound on time complexity to make the algorithm very fast. One nature of heuristic algorithms is that they do not guarantee success [10], they may give you a level of success but they are very fast and use less space. They find the nearest match. One application of the heuristic algorithms is the BLAST search of matching DNA sequences against a query sequence [20].

2.3 Biology of the African Cassava Mosaic Virus

Geminiviruses compose the major group of pathogens for monocotyledonous and dicotyledonous plants in tropical, subtropical and, to a smaller extent, temperate regions [21] [22] [23]. These viruses are a large and diverse family of plant viruses. Their genome is made up of one or two circular single-stranded DNAs (ssDNA) of 2.5 - 3.0 kb with an encapsidated characteristic particles consisting of two incomplete T1 icosahedra joined together to produce a unique twinned particle structure of approximately 20 x 30 nm [24]. These viruses are capable of mutating by either deletion in the DNA or insertion. Three characteristics, genome organisation, insect vector and host range, allow for the dividing of Geminiviruses into four genera [25] [26]. One of the four genera is the genus *Begomovirus* which are transmitted by the whitefly *Bemisia tabaci* (Gennadius) to dicotyledonous plants and have either mono- or bipartite genomes (DNAs A and B). There are two key things involved in the success of viruses, their replication and quick spread. For bego-

moviruses, these two are determined by their coat protein (CP). CP and all proteins required for replication (Rep and REn) and gene regulation (TrAP) in begomoviruses are encoded by DNA A (Figure 2-5) [21]. The role of DNA B is to encode two products (NSP and MP) which are responsible for the spread of the virus and its symptom production after infection [27]. Cassava geminiviruses occur in all cassava growing areas of Africa and are considered to be the most damaging vector-borne plant pathogens [28]. 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 [29] and there is definitely need to find a solution to the ACMV problem. The abundance of the whitefly helps the ACMV spread quickly [30]. The rapid spread of the ACMV is capable of causing enormous economic losses [31] [32] [33]. The infection can result into up to seventy percent loss of tuberous yield of the African cassava [34] [35]. The mutation rate of the ACMV is rapid and unpredictable. As a result, for over one hundred years scientists who seek solutions to its effects have found it difficult to determine its mutation rate which is one of the key elements to the introduction of vaccine against ACMV into the African cassava. Despite most research on the ACMV being recent, the first record of ACMV dates as early as 1894 [36].



B

```
  1  MSKRPGDIII STPGSKVRRR LNFDSPYRNR ATAPTVHVTN RKRAWVNRPM YRKPTMYRMY   60
 61  RSPDIPRGCE GPCKVQSFEQ RDDVKHLGIC KVISDVTRGP GLTHRVGKRF CIKSIYILGK  120
121  IWLDETIKKQ NHTNNVIFYL LRDRRPYGNA PQDFGQIFNM FDNEPSTATI KNDLRDRFQV  180
181  LRKFHATVVG GPYGMKEQAL VKRFYRLNHH VTYNHQEAGK YENHTENALL LYMACTHASN  240
241  PVYATLKIRI YFYDSIGN                                               258
```

Figure 2-5: (A) Genomic map of ACMV DNA A plasmid pACMV1.8A. Solid arrows define the position, orientation, and reading frame of the viral genes. The common region (CR) for both genomic components is indicated by shaded boxes. The positions of primers and restriction sites used for cloning of DNA A constructs or characterisation of viral DNA are indicated. (B) Amino acid sequence of ACMV coat protein. Amino acid residues previously identified to be implicated in transport functions and deleted in this study are given in bold and underlined. - "Adapted from [21]"

Summary

The African cassava mosaic virus has ravaged the African cassava for over one hundred years. It causes cassava mosaic disease which reduces tuberous yield of cassava even up to 70 percent. Its mutation rate has been found to be rapid and unpredictable. As a result of the rapid rate of mutation, solutions to its adverse effects have failed for over one hundred years. Computer science can help in speeding the rate at which the mutation rate of the ACMV is determined by finding the best algorithms to use. The best algorithm required to study the rate of mutation of the ACMV is one that gives less response time and is mostly linear in terms of *Big-O Notation*. There is, therefore, need to understand algorithm performance in order to choose an algorithm that will bring results quick enough to determine the rate of mutation of the ACMV. This chapter was added to provide a background to the research by explaining concepts, tools and other subject areas used during the research. The next chapter anchors on a number of things brought out in chapter two.

CHAPTER 3

LITERATURE REVIEW

In order to understand past and related work, I took time to read what had been done before on the process that helps to sever the effects of the ACMV. The process involved the search for virus strains that were related to the ACMV through a BLAST. The results of the BLAST are aligned as a multiple sequence alignment and pairwise alignment to expose the evolution of the ACMV. The evolution can be summarised into a phylogenetic tree diagram. All this work required suitable software. This chapter outlines the said process and related software, in literature. The chapter begins with literature review on the BLAST in section 3.1, followed by pairwise and multiple alignments in sections 3.2 to 3.4, then the phylogenetic tree diagrams and their algorithms in sections 3.5 and 3.6 followed by metagenomics and its application on ACMV in sections 3.7 and 3.8, biopython in section 3.9, data sharing techniques in section 3.10, and single page applications in the last section 3.11.

3.1 BLAST

The BLAST is a very popular biological pairwise sequence alignment algorithm, the main idea of which is to search heuristically for high-scoring segment pairs (HSPs) in the alignment matrix involved between a pair of sequences (usually a query sequence and a subject sequence from a biological database) before a local extension around these HSPs takes place [37] [38]. The main heuristic (section 2.2.8) of BLAST is that there are often high-scoring segment pairs (HSPs) contained in a statistically significant alignment. BLAST is an acronym for Basic Local Alignment Search Tool. It comprises three heuristic layers,

which are seeding, extension and evaluation. In the first stage, sequences from a database are rapidly compared with a given query sequence to identify hits with a score over a particular threshold value (T). In the second stage ungapped extension is performed, between two hits in the same diagonal line, to obtain high scoring pairs (HSPs). The third stage performs gapped extension in order to insert gap characters that will make all sequences the same length. It is the most common alignment tool available. Blast is actually a set of algorithms for sequence alignment done in databases that have numerous protein and DNA/RNA sequences. In short, Blast is a set of alignment algorithms for determining whether a database contains potential homologous sequences to the newly derived sequence. These alignments exist in two types: the global and local alignments. The global alignment searches the whole length of the database sequence to determine homology with the derived sequence. This type of alignment takes a lot of time and thus expensive to run. Local alignment creates a short component of the derived sequence which matches a portion of the database sequence. This short component is called a seed. It is then extended both directions to determine homologous sections of the two sequences. The seed and its matches are usually referred to as conserved sequence segments. Whatever the change genetically, as a result of mutation or speciation, these conserved genes are carried from generation to generation. The three stages of BLAST are explained in the following sub-sections using arbitrary letters to represent a protein.

3.1.1 Stage 1

Given an example protein MKFVLL, the first stage starts with the extraction of fixed length overlapping sub-sequences (words), usually in triads for amino acid sequences and 11 for nucleic acid sequences. In this case, the extracted words are MKF, KFV, FVL, and VLL. Generally, the number of words (w) extracted from a sequence of length m (m $>$ 2) is (m-w) + 1. The extracted words are then compared with words of the same length from the query sequence one by one using a mutation scoring matrix. A match between a pair of words is considered as high scoring if the words are identical or the comparison score is over a particular threshold value (T) [37]. In order to facilitate the matching of subject sequence words with all the possible matching query sequence words, a lookup table (Figure 3-1) is created.

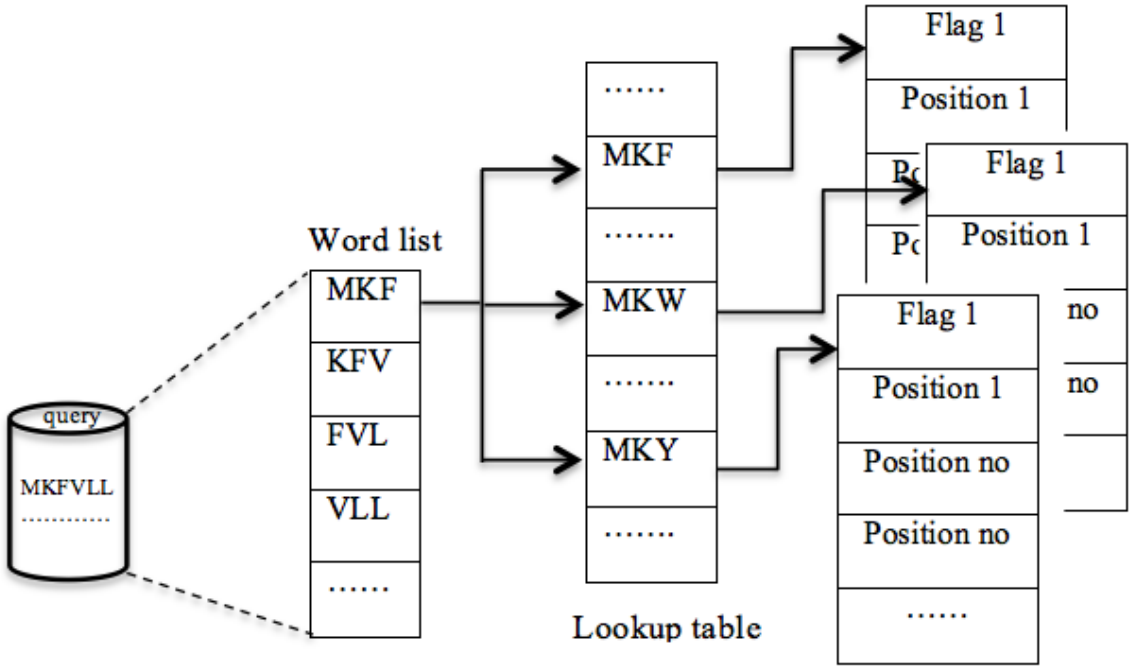


Figure 3-1: Structure of the lookup table

This lookup table contains all possible words, the number of which depends on w and the number of alphabet used. Since there are 25 continuous alphabets from A to Y in the BLOSSUM62 mutation matrix used in BLAST, with five unused alphabets (B, J, O, U, X), each amino acid can thus be coded as a five-bit binary value [37].

3.1.2 Stage 2

In the second stage a hit is extended to determine if it is contained within a segment pair whose score is greater than or equal to a threshold parameter S. The hit is actually extended on both sides of the subject sequence (sequence in the database). The extension does not stop until the accumulated total score of the HSP begins to decrease with respect to threshold parameter S. After the alignment terminates, it is trimmed back to the maximum score [37]. HSPs whose score is greater than the empirically determined cut-off score S are then output. A different example protein is used in Figure 3-2 to illustrate stage 2.

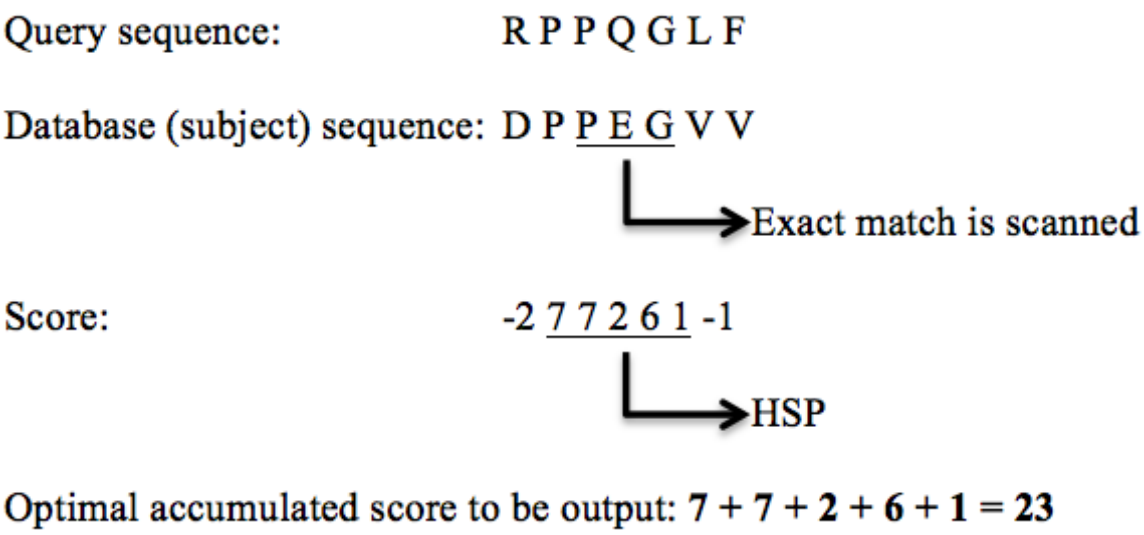


Figure 3-2: Calculation of Optimal Accumulated Score of Sequence

3.1.3 Stage 3

The third stage performs gapped extension. In order to make the subject and query sequences same length, gaps are inserted in either the subject or the query sequence. This is achieved by extending the alignment of ungapped extension both backwards and forwards from the centre of the ungapped alignment [37]. Gap characters are inserted where gaps exist.

3.1.4 Algorithm Type and Big-O Notation for BLAST

BLAST is actually a set of bioinformatics algorithms. These are BLASTn, BLASTx and TBLASTx which compare DNA query sequence to DNA sequence databases, BLASTp which compares a protein query sequence to a protein sequence database and TBLASTn which compares protein query sequence to a DNA sequence database. BLAST uses heuristic algorithm types which is explained in section 2.2.8. The Big-O notation (section 2.1) for BLAST is bound to be $O(mn)$ which is linear in the context of the search query being a constant because it creates a seed sequence from the query sequence and searches for matches based on an expected-value (e-value) through a database of n sequences, . If the query sequence was not a constant it was going to be quadratic. As already observed in chapter two, heuristic algorithms are very fast in bringing out output. At the same time they are not deterministic but probabilistic. They are, therefore, liable to bringing out inaccurate results. However, for databases that have thousands of sequences, heuristic

algorithms are the best fit for database searches. Otherwise the BLAST process would be too expensive in terms of time and other computer resource use. The accuracy of the results can be improved by setting a low expected value (e-value), usually 0.04 or less.

3.2 Pairwise and Multiple Sequence Alignments

Sequence alignment assumes that proteins share a common ancestry. Sequences that share a common ancestry are called homologous sequences. Pairwise sequence analysis is used to detect homology between different protein or DNA sequences. It is the most fundamental operation of bioinformatics [39]. It has helped biologists to detect pathogens, come up with new drugs, identify common genes and study evolution of genes and organisms [40]. There are many ways of carrying out an alignment. It can be done by sliding two sequences on two lines of a word processor, manually, it can be done with a dot plot (Figure 3-3) in windows, by rigorous Dynamic programming which is slow and optimal and it can also be done using heuristic methods like BLAST and FASTA which are fast but approximate. Pairwise analysis uses heuristic algorithms. It is used to determine either the structural or the functional relationship of two proteins. With pairwise analysis, domains or motifs which are shared between proteins are identified. BLAST searching is based on pairwise analysis. The mostly used algorithm for pairwise analysis is the classical Needleman Wunsch algorithm.

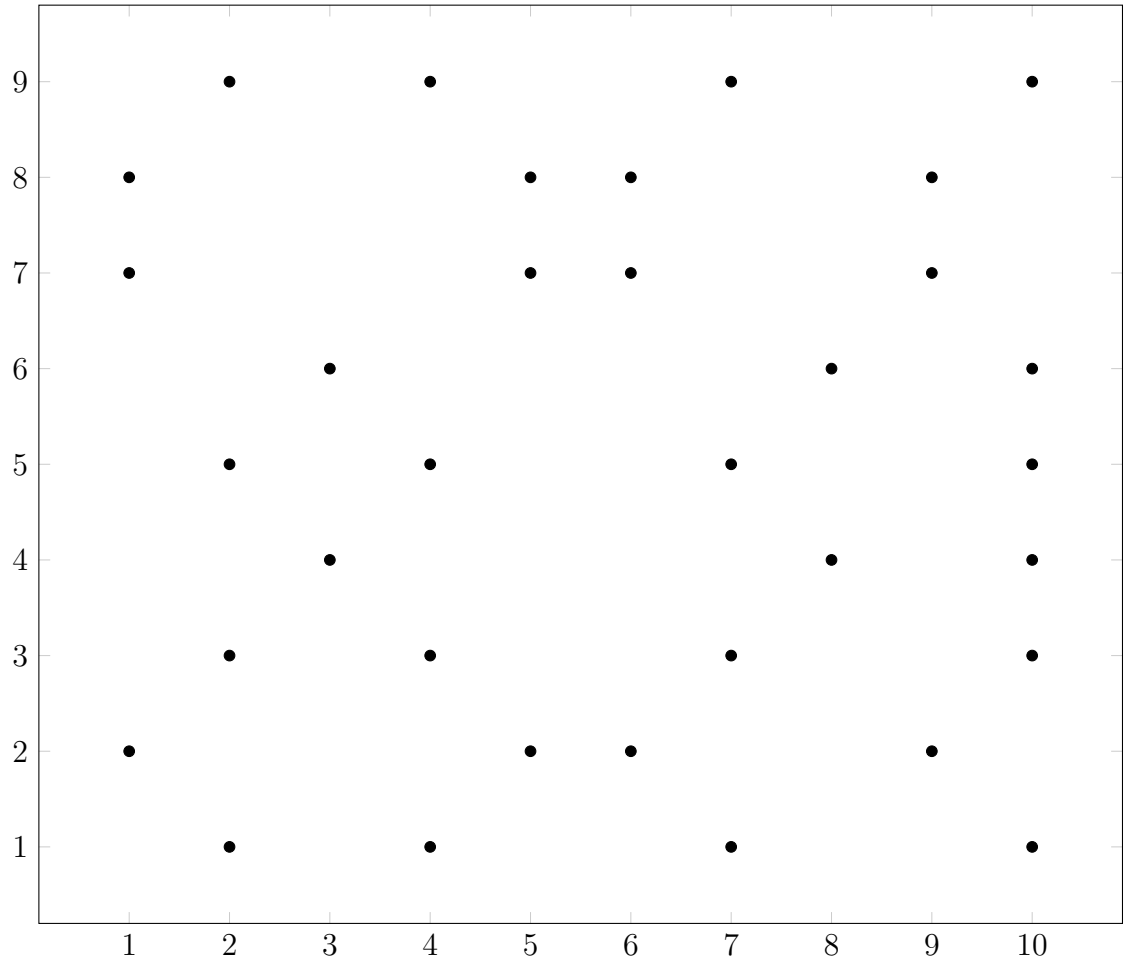


Figure 3-3: Example Dot plot

3.3 Multiple Sequence Alignment

A Multiple Sequence Alignment (MSA) is used to present homologous sequences which are sequences related by a common ancestral sequence [41]. In an alignment, a matrix of homologous characters (also called sites) of sequences is created. To ensure same length, gap characters are inserted in sequences with less characters than others. Almost all sequence analysis and analysis software tools like CLUSTALW derive their input from sequence alignments [41] [42] [43]. Multiple and pairwise sequence alignments are meant to study evolution of genome material. Evolution of genome material results in gene changes that may bring about introduction of new protein and hence new strains of the same species in organisms. Evolution of genome material results from either substitution of characters or deletion of characters or insertion of characters in a DNA sequence. Insertions and deletions can only be verified with knowledge of the ancestral sequence,

which is rarely possible. The insertions and deletions are generalised as *indels* because of the absence of the ancestral sequences in most cases [41]. Character substitutions which do not result in new protein are called synonymous mutation while those that result in new protein are called non-synonymous mutation. During a MSA, indels are represented by additional gap characters like the '-'. Figure 3-4 shows a segment of an example multiple alignment.

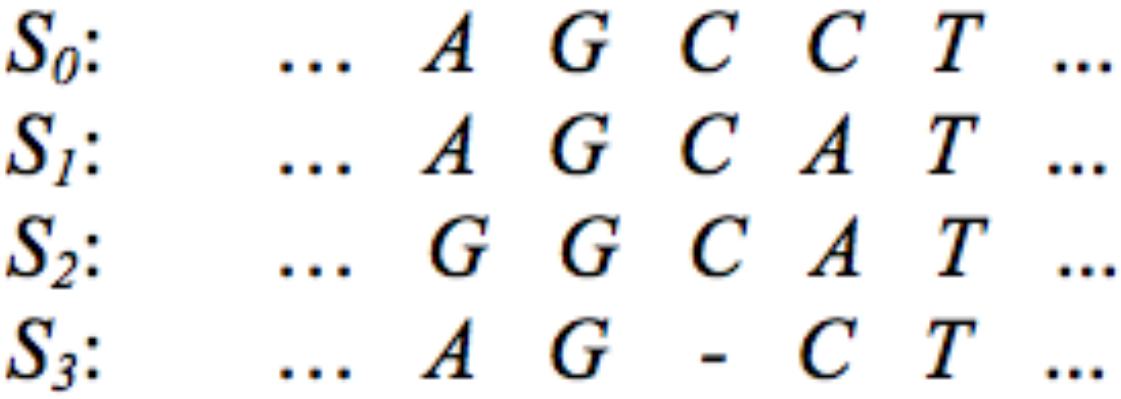


Figure 3-4: Example MSA

3.4 Algorithm Type and Big-O Notation for Sequence Alignment

The algorithm that is traditionally used for multiple sequence alignment is the dynamic programming which has a polynomial big-o notation. Dynamic programming is explained in sub-section. 2.2.4. The quadratic function algorithm explained in sub-section 2.1.3 is a member of the polynomial functions algorithms. Polynomial algorithms perform better than exponential function algorithms in terms of time and memory usage efficiency. Some scholars have worked to improve the efficiency of the multiple sequence alignment by utilising a hybrid of algorithms or using algorithms that are more efficient than the dynamic programming. Zhang in [44] used the greedy algorithm to achieve the same alignment done using the traditional dynamic algorithm ten times faster while obtaining the same data output. Ari Löytynoja and Nick Goldman in [45] observed that the traditional dynamic programming algorithm for an alignment is not distinguished between a deletion and an insertion. They improved the traditional dynamic programming algorithm to come up with their algorithm for progressive multiple alignment which had more

insertion detection than deletions in the output. Pairwise alignment is also based on traditional dynamic programming algorithm but F. Corpet in [46] proposed an improvement using hierarchical clustering.

3.5 Phylogenetic Tree

A phylogenetic tree is a diagrammatic representation of evolutionary relationships or history of objects. These objects can be anything from biological species to proteins to nucleic acids to languages [47]. The research focuses on nucleic acids. I only give a brief explanation of phylogenetic trees but detailed explanation can be found in [48] and [49]. Phylogeny simply means the ancestral relationship of species. In the same vain, phylogenetics is a research area that deals with finding the genetic relationship between species [50]. Classic phylogenetics dealt mostly with physical, or morphological features like size, colour, number of limbs or fingers [41]. These are phenotypes of organisms. Classic phylogenetics resulted from less development in biological knowledge before the genesis of genetics. The genesis of genetics brought about categorising using genotype. Modern phylogeny uses information extracted from genetic material – mainly DNA and protein sequences. [51]. Mostly, DNA or protein sites characters are used to build sequences. The most convenient way of presenting phylogenetic information is using a phylogenetic tree.

In a phylogenetic tree, species are represented by nodes. Nodes are labelled, either with species names or the values (also referred to as states) of their characters, and the edges represent the genetic connections [52]. Phylogenetic trees can be rooted or unrooted, binary or general and may show or not show edge length. In a rooted tree, one of the nodes represents a root which determines common ancestry. An unrooted tree has no hierarchy because there is no predetermined root in it. An unrooted tree just shows species relatedness. Figure 3-5 is an example of a phylogenetic tree of animal classes. In Biological studies, there are two main types of phylogenetic trees used. These are the ascii and the PhyloXML trees. A PhyloXML tree (Figure A-1) is simply an XML representation of the identity of every sequence representing viral strains including branch length involved in drawing the ascii type of tree. In an ascii tree, branching

is horizontal while the tree grows vertically. Using biopython libraries (section 3.9), branches can be distinguished by colour for easy observation.

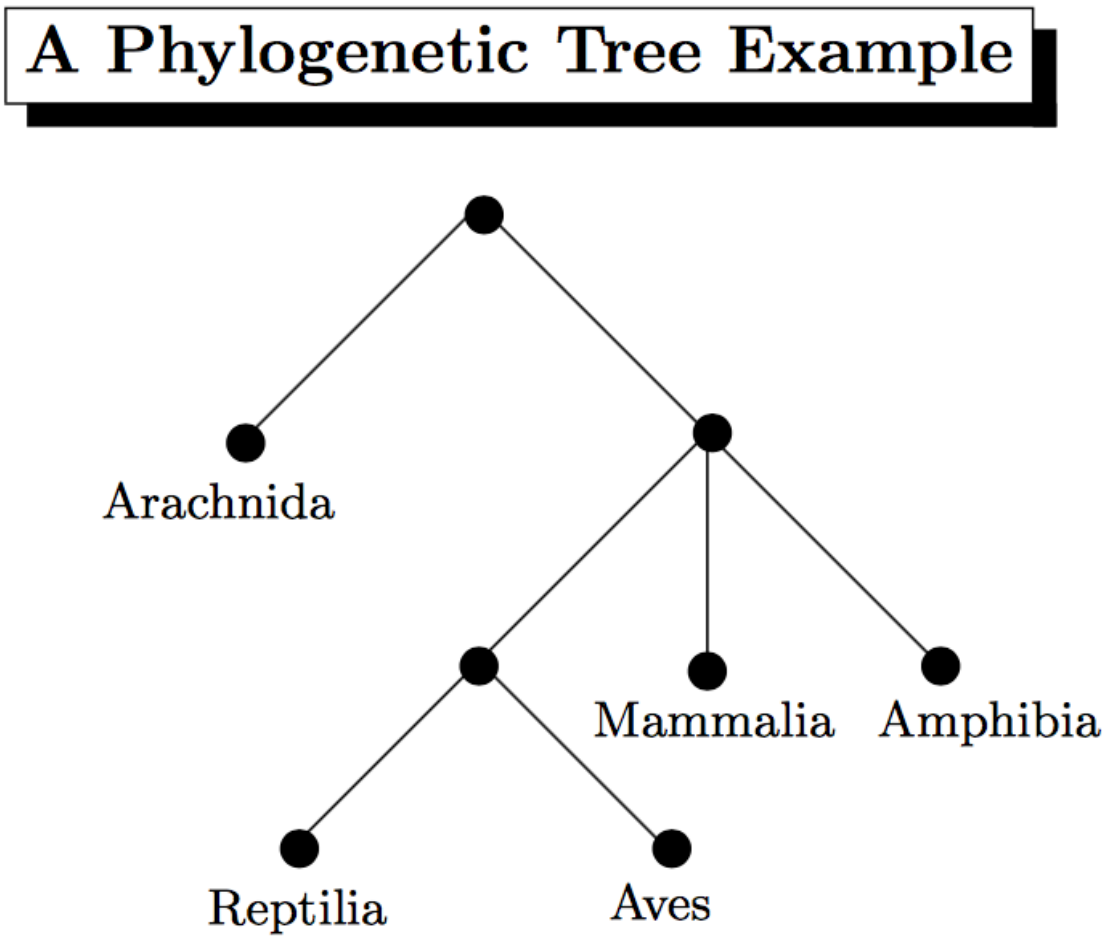


Figure 3-5: Example of a phylogenetic tree (rooted)

3.6 Algorithm For Phylogenetic Tree Diagrams

Building of phylogenetic trees generally uses the greedy algorithm by iteratively selecting the leaf of the tree that increases the phylogenetic diversity of the tree until k number of leaves has been selected [53]. The greedy algorithm design technique is explained in sub-section 2.2.3. The greedy algorithm here gives *O(n)* because it just involves setting the GREEDY STEP $S := S_0$ and applying it k times to the GREEDY STEP. This algorithm is fast because it is linear (sub-section 2.1.2). Andreas Spillner *et al* gives more information on the implementation of the *O(n)* [54]. L.-T. Nguyen in [55] discusses an improvement on the phylogenetic alignment by maximum likelihood (PAML) because maximum likelihood is heavily computational making it resource expensive. Nguyen

proposed an algorithm called IQ-Tree (important quartet tree) which was an improvement on the IQPNNI (Important Quartet Puzzling Nearest Neighbour Interchange) algorithm proposed and used by L. S. Vinh and A von Haeseler in [56]. Nguyen used the Iterated Local Search (ILS) as an underlying search framework to make the IQ-Tree two to four times faster than the IQPNNI. IQPNNI was an improvement on likelihood used in the PAML by T. Alicai et. al. in [1] and N. Goldman in [49]. Clearly. Scholars have been suggesting improvements on the traditional algorithm for the phylogenetic tree generation. What is mostly missing from the improvements is the provision of an easier workflow using pre-existing multiple software tools. This scholarly work is bound to continue because computational work is dynamic.

3.7 Metagenomics

Metagenomics is the study of microbes in the natural environment including their relationships to other organisms like their host [57] [58] [59]. Biological scientists follow microbes in their natural environment and carry out wet tests to extract DNA material for analysis. In the process of studying a particular microbe, they analyse its surrounding ecological environment also to know factors that affect its existence, its success or its dwindling numbers. The simple reason for the existence of Metagenomics is that microbes run the world [60]. They affect every other living and non-living organism. In the living organisms, they may cause disease. In the dead organisms they may cause decomposition. Microbes are, therefore, ecologically very important. Without them many life processes would not occur. Without them, the earth was going to be filled with smelly garbage all over and was going to be unbearable to live in. It, therefore, requires caution when trying to eliminate any microbe that causes any un desired effect. Biologically, control of microbes is more desirable than elimination.

3.8 Metagenomics of ACMV

Metagenomics allows scientists to study microbial diversity and dynamics without having to perform any wet tests in the artificial media [61] [62]. Metagenomics is used to study how the ACMV affects some of man's needs like vegetables. Vegetables are an important

factor of diet of the rural areas of Zambia and most of Africa as a whole. Vegetables are mostly affected by microbes. Cassava is one of the vegetables contributing to the nutrition of east, west and central Africa. Cassava has been affected by ACMV which is of the genus *Begomovirus* and family *Geminiviridae* [63]. The success of the *Geminiviruses* is also in their small DNA which is easily cloned [64] [65]. Humans have also contributed to the spread of these viruses, especially the ACMV due to the transporting of variants in the cassava from region to region trying to bring improvement to the yield. Some of the activities of humans that are meant well end drastic. The other mode of fast spread of the ACMV has been through the abundance of its vector the whitefly *Bemisia tabaci*. As much as the mutation rate of the ACMV is critical when trying to control its effects, its environment and the human factor must also be considered. Could the abundance of the cassava whitefly be a critical factor in determining the mutation rate and spread (infection) of the ACMV? Could the attempt by humans to find improved cassava variants have an adverse consequence? All these are matters for biological and agricultural scientists to consider while computer scientists provide the tools to expedite results.

3.9 Biopython

Biopython is a collection of open source bioinformatics tools written in an object-oriented scripting language called Python [66]. It is a project, which dates as far back as August 1999 [67] [68]. 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 [69] [70]. A number of modules can be integrated to make any bioinformatics project complete. It has modules for creating both online and standalone tools. It has methods for creating graphical user interfaces. Biopython has modules for the BLAST. A BLAST can be done online or from a local genome library using biopython. A local or global BLAST can be achieved with the same libraries from biopython. It has modules for multiple and pairwise sequence alignments. It also has modules for creating phylogenetic trees in both XML form and ascii form. It

is of interest to state that each stage of genome analysis, using biopython, can output an XML file which can be used as input for the next stage. These modules, which I desired to use for the purposes of this research, are summarised in Figure 4-8. There is no need to create the science behind the modules because it has already been done and made ready to use.

3.10 Data Sharing and Display Techniques

There are various data transfer and display protocols. Data sharing and display is mostly achieved using markup languages and protocols like the HyperText Markup Language (HTML), HyperText Transfer Protocol, the eXtensible Markup Language (XML) and technologies like the Scalable Vector Graphics (SVG) and Graphical Interchange Format (GIF). As part of the work in the research, I sought to find ways that different genetic analysis software tools would communicate in order to cut off manual feed of data into each one of them. I decided to work with XML which is a universal standard of data sharing technologies. I also leveraged on HTML and SVG for information display and creation of single page applications, respectively.

3.10.1 eXtensible Markup Language

The eXtensible Markup Language (XML) is a subset of the Standard Generalised Markup Languages like HTML [71] [72]. XML and HTML have the same heritage [73]. XML is actually a meta-markup language that comes with a format for describing its structured data [74]. 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. It is not a programming language but a markup language that is used to describe how information is displayed. 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 [71] [73] [75]. 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. XML is not programming language specific. This is why it can be used to link several applications or systems that are disparate in nature. It can also aid development of distributed systems. The development of a workflow of pre-existing tools to be used to determine the rate of mutation of the ACMV requires such universal technologies like the XML. It would be used to link the disparate pre-existing tools as a common data transfer protocol.
A typical XML file has three parts. The first one which is optional is the XML declaration. It is placed at the top of the XML file. Though optional, the declaration is recommended to be included in all XML files. The minimum XML declaration takes the form:

<?xml version="1.0" ?>.

It contains the version of the XML, which at the time of the research was still at version 1.0. The second part of the XML file is another optional but recommended part called the document type that refers to a DTD. Its general syntax is as follows:

<!DOCTYPE RootElement (SYSTEM | PUBLIC) External-Declarations?
[Internal-Declarations]? >

The RootElement is the name you give as the parent element of the document. You choose between the literal words SYSTEM and PUBLIC and indicate the relative path of your DTD as External-Declaration. You may have a declaration within the document, which is enclosed within square brackets as Internal-Declaration. The DTD is a document that defines how data should be displayed and categorised on any platform. The alternative to the DTD is an xml schema. One of them is the XSLT which was created by the World Wide Web consortium. XSLT is eXtensible Stylesheet Language Transformations. From this name, it is clear that a schema is used for styling XML data when presenting it on any platform or medium. An xml schema is better than a DTD because the xml schema supports namespaces. The third and last part of an XML file is the body or document instance which is a must for an XML file. It comprises elements of the XML file nested from parent element tags down to children element tags and sub children element tags. Within the element tags are the text values of the elements. Any attribute of the element is placed inside the opening element tag angle brackets after the element name. XML

attributes just provide extra information about an element in the XML file. Each element is opened and closed. Take note that the closing tag of the element has a forward slash after the opening angle bracket. The following code snippet shows an XML body.

```
<element attribute = "element definition here">

    <child element1> # nested inside parent element

        <sub child element> # nested inside child element
        </sub child element>

    </child element1>

    <child element2>
    </child element2>

</element>
```

3.10.2 Scalable Vector Graphics

One of the challenges in attempting to determine the rate of mutation of the ACMV is the time it takes to get output. This is influenced by the software tool used and the software tool's underlying algorithms. Another factor of the tools used is the time spent feeding data into different tools to obtain desired results. An application that receives data at one entry point and can output all analyses desired without data re-feed would contribute to solving this challenge. Avoiding to navigate from window to window would also contribute to the solution. Fortunately a technology like the Scalable Vector Graphics (SVG) provides such a solution. SVG is an XML based technology for graphics which has made it possible to produce high resolution two dimension graphics that we enjoy in print media to be available to the web. It is an open and platform-independent standard which is maintained by the World Wide Web Consortium (W3C) [76]. It integrates well with other web technologies like HTML, GIF and CGI to give dynamic and interactive graphics that are scalable to fit any web environment [77] [78] [79] [80]. There are three types of graphic objects permitted in SVG. The first ones are the vector graphic shapes which are mostly lines and curves. The second type are images and the third ones are text [81]. SVG makes it possible to design interactive web applications including Single

Page Applications (SPAs). SVG has been used in many applications including GIS and biomedical genetic analysis [76] [82].

3.11 Single Page Applications

SVG is the future technology of dynamic applications. Most developers and users aim at all the time improving response time when getting results from an application. One way to save time is the development of web applications that reduce on navigating from window to window. SVG helps create such interactive and dynamic single page applications. A single page application is a dynamic web application which employs reactive programming and data binding [83] [9]. SPAs allow for development of applications that respond to hovers, clicks and tapping of a single window content. With SPAs, a user does not need to navigate from window to window because content is brought to view within the same window and can be closed without losing the initial content of the window. SPAs save on time wasted in navigating from window to window. The implementation of the SPA through a proof of concept, in the research, is an added advantage to saving on time wasting and achieving success in determining the rate of mutation of the ACMV.

Summary

The study of the ACMV utilises software tools. The biological science researchers who use the tools are lay persons to information technologies. They, therefore, need comprehensive tools with an easy workflow in order to work fast enough to beat the rate of mutation of the ACMV. The ACMV is first studied from its natural environment and later genetic material is extracted for laboratory studies. The study of the virus in its natural environment is called metagenomics. There are four main stages in analysing genomic data of the ACMV; percentage relatedness through BLAST, multiple sequence alignment, pairwise sequence alignment and evolutionary relationships through phylogenetic trees. The review of literature shows that existing applications used for the analysis of genome data are not comprehensive. None of the five applications used by researchers at MMRS which were reviewed in this research brings out satisfactory output in all the four stages of ACMV genome data analysis. Faster algorithms help speed up the analysis

of ACMV genomic data which is usually huge. The analysis of algorithms was included in chapters two and three to give an understanding on selecting the best algorithms when developing software that would give shorter response time during the study of the ACMV. Most applications used for the study of genomic data are proprietary and thus not very affordable to low income earners and student researchers. Biopython is one of the cheaper solutions to expensive proprietary software because it is open source but has all modules to use for the study of the ACMV. The ability for disparate tools to communicate can enhance automatic feed of data from tool to tool. XML is the key element in linking such disparate tools. Time can further be saved if navigation from window to window or application to application is cut off. HTML, XML and SVG integration comes handy in creating SPAs that will cut off navigation from window to window or application to application.

CHAPTER 4

METHODOLOGY

To guide the research, three objectives were set out as given in section 1.4. The research begun with a study of pre-existing bioinformatics software tools which were used by biological and agricultural science researchers at Mount Makulu research station in Zambia to study the mutation of the ACMV. The results of the study lead to the proposal of a workflow that would be used to predict the mutation of the ACMV. The research ended with a proof of concept on the proposed comprehensive tool with an easy workflow to be used to study mutation in the ACMV. A purposeful sample type was used to interview a researcher at Mount Makulu research station in order to find out which software tools they used to study the ACMV and how they used them. The findings from the interview with the Mount Makulu research station researcher led to further experimentation with the software tools used by the station to determine how easy the workflow was. Later, a workflow was proposed that would integrate these tools and would be used to build tools that had easier workflow for the study of the ACMV. The research used a mixed method approach.

4.1 Mixed Method Approach

Mixed method approach to research integrates both qualitative and quantitative research. With Mixed Method, research uses more than one type of data collection and results analysis [84] [85]. The collection and analysis of both qualitative and quantitative data occurs in a mixed method research approach [86]. Mixed method research gives an advantage when tackling complex research questions [87]. Quantitative methods emphasise

objective measurements and the statistical or mathematical or numerical analysis of data collected through surveys, polls, questionnaires or pre-existing statistical data. The analysis uses computational techniques. During a quantitative approach data is statistically analysed for hidden answers to research questions; answers that have not yet been given [88]. Qualitative methods are more descriptive in nature but a descriptive approach can be applied in a quantitative study also. During qualitative methods, observations are given a detailed explanation including contexts of events and circumstances affecting the observations. A case study is an in depth exploration of an event, programme, activity, process or one or more individuals [89] [86]. Table 4.1 gives a summary of the distinction between qualitative and quantitative methods [90]. In order to learn the bioinformatics software used by the researchers at Mount Makulu research station in Zambia, the research took an exploratory approach and entered an experimental phase which lead to the development of a proof of concept that a comprehensive tool with an easy workflow can be built.

Table 4.1: Comparing qualitative and quantitative research methods.

Dimension	**Qualitative**	**Quantitative**
Focus	Quality or meaning of experience	Quantity, frequency, magnitude
Philosophical roots	Constructivism, interpretivism	Positivism
Goals	Understand, describe, discover	Predict, control, confirm, test
Design characteristics	Flexible, evolving, emergent	Structured, predetermined
Data collection	Researcher as instrument	External instruments: tests, surveys, etc.
Question types	Open ended	Closed ended

4.2 Exploratory Design

An exploratory design usually precedes an experimental design in a mixed methods type of research or study [91]. It is meant to bring out knowledge about a subject before a researcher goes into experimenting with the subject. In an exploratory design, a researcher may read literature on the subject, may use pen and paper or phone interview as long as in the end there is concrete knowledge of the subject. An exploratory design does not bring about any final and conclusive solutions to research questions, it just explores them. No conclusive evidence is provided during an exploratory design. It just helps to better understand a problem. Unlike conclusive studies that provide specific and final information that guides the solution to a problem, exploratory studies arrive at a range of causes and options to arriving at a solution to a problem. With the foregoing explanation about an exploratory design, the research integrated the exploratory design in order to understand the use of the bioinformatics software by biological and agricultural scientists at Mount Makulu research station in Zambia.

4.3 Exploration of Pr-existing Bioinformatics Software Tools Used In Zambia

The research was triggered by the need at Mount Makulu research station, to determine the rate of mutation of the ACMV. In the initial stage of the research I obtained information from researchers there on what software they used and how they used them to study the ACMV. After knowing the software tools used, the Mount Makulu Research Station researchers and I tested the use of the software tools to observe response time. Observation of response time excluded human interaction of feeding data between the four stages of analysis of the ACMV involved. The four stages are percentage identity analysis, multiple sequence analysis, pairwise sequence analysis and evolutionary relatedness analysis. The computer used during the analysis was an i7, quad-core processor laptop with a sixteen gigabytes random access memory. After the study of the software used by researchers at Mount Makulu research station the research turned into elaborate experimentation with the software tools to have deeper understanding of what they could and could not do. The software tools studied were Bioedit, Geneious, Mega, NCBI and SDT. These are the software used by Mount Makulu research station researchers to study the ACMV. The study of the software took three weeks. Initial data that was used for experimentation was obtained from the Mt. Makulu research station researchers; but it is open source data downloaded from the NCBI. Fortunately, more data could be obtained freely from the NCBI. Each software tool is presented in sub sections 4.3.1 to 4.3.5.

4.3.1 Bioedit

Bioedit was used by researchers at Mt. Makulu research station only to export the sequences to a fasta format file, which was then imported into SDT for percentage identity analysis [66]. But Bioedit could do more than they comprehended. Bioedit is a tool for biologists who have no interest in the science of the software but what it can do [92]. It is a menu driven application with a graphical user interface. It is generally used for sequence manipulation and editing. It can also be used for sequence alignment of DNA, RNA nucleic acids and protein sequences. In short, Bioedit has been in molecular studies of organisms such as virus genome, bacterial genome, plant genome and animal genome [93]

[94] [95] [96] [97] [98]. Bioedit can import data from clipboard and can read various file formats like .txt, .fas, .fasta, .fst, .xml, and .meg as long as they are in fasta data format. It has NCBI web service capability while output can be viewed in many forms including colour coded sequence alignments (Figure 4-1) [66]. A multiple sequence alignment trial with Bioedit did not finish processing until it was just discontinued. Bioedit was never used by Mt. Makulu research station (MMRS) researchers to obtain phylogenetic trees.

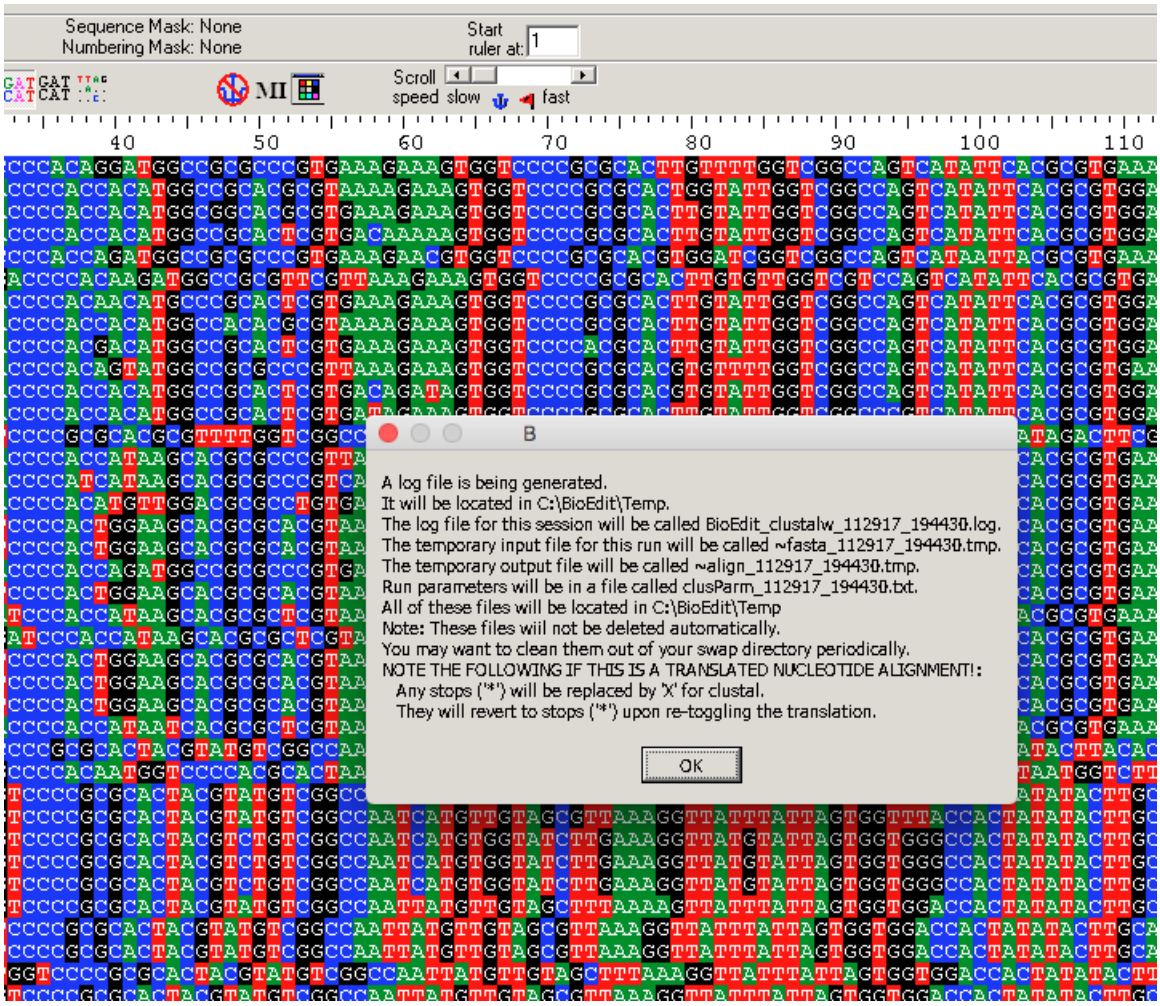


Figure 4-1: Colour coded output of multiple sequences in Bioedit, before an alignment

4.3.2 Geneious

Geneious is a desktop software application for the organisation and analysis of biological data [99] [100] [101]. This is a proprietary software tool which the researchers at MMRS have only been using for its nice output of phylogenetic trees (Figure 4-2) but have not loved its look and feel and an unpleasant presentation of multiple sequence alignment(Figure 4-3) when analysing multiple sequences [66]. Geneious can, however process

all required analyses from sequence alignment to phylogenetic tree production. Geneious has NCBI web service capability and can even create a local database of sequences. It supports multi file formats for output such as .txt, .geneious and .csv.

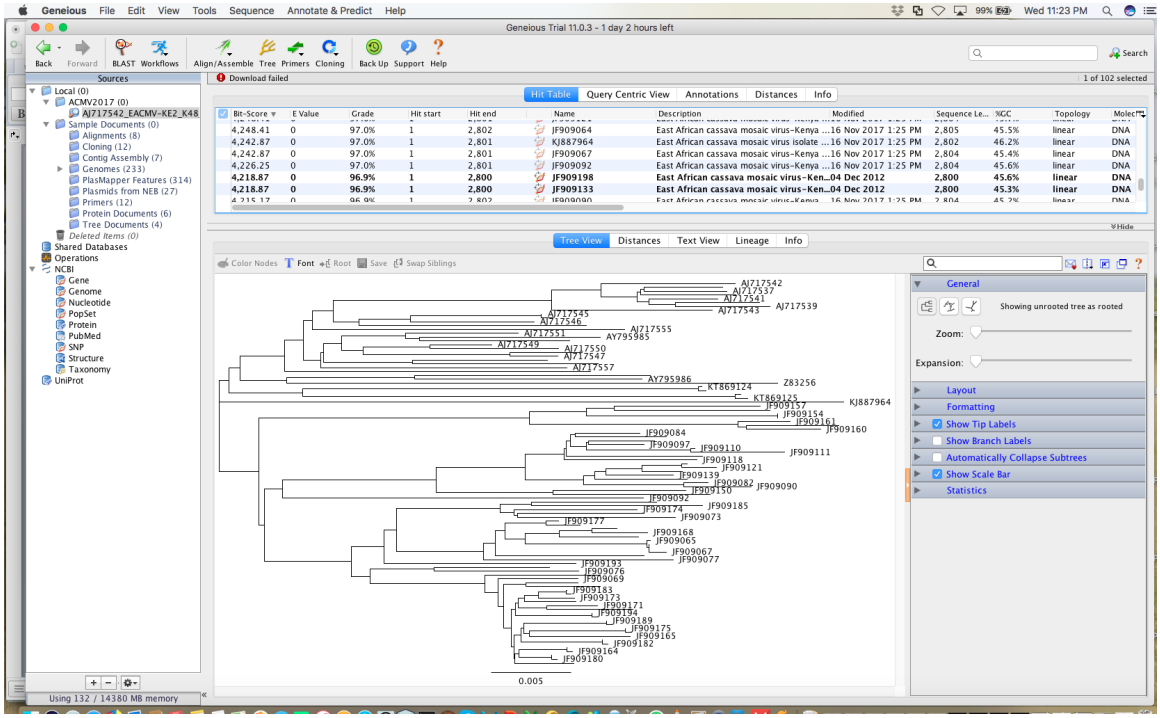


Figure 4-2: Geneious output of a phylogenetic tree

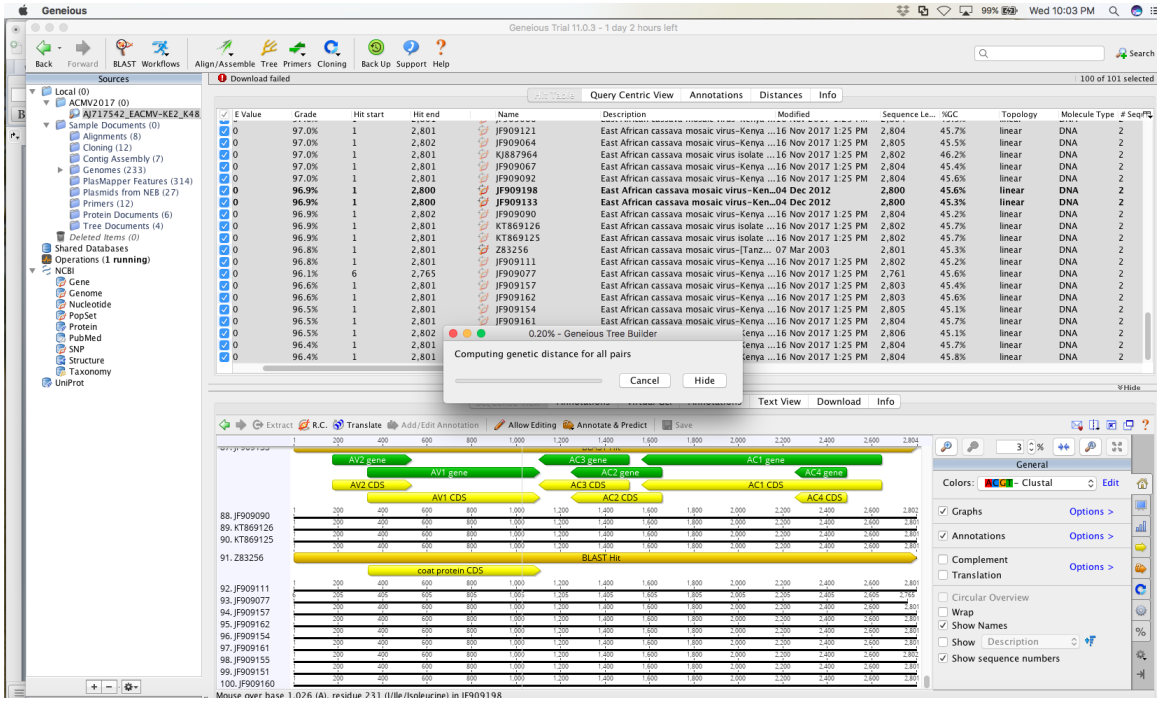


Figure 4-3: Presentation of a multiple sequence alignment in Geneious

4.3.3 Mega

At the time of the research, MMRS researchers used Mega 6 (Molecular evolutionary genetics analysis 6) but Mega 7 existed. I first tested the use of Mega 6 while working with the MMRS researchers before trying out Mega 7 also, on my own, to learn what Mega could and could not do. Mega 6 could build sequence alignments, infer phylogenetic histories, and conduct molecular evolutionary analysis [102] [103]. The focus of its earlier releases like Mega 6 which had web service capabilities was on facilitating the exploration and analysis of the DNA and protein sequence variation from an evolutionary perspective [104] [105]. The MMRS researchers used Mega 6 only for multiple sequence alignment (Figure 4-4) and phylogenetic tree production (Figure 4-5). The plain text (.txt) fasta document exported from Bioedit was imported into Mega with belief that Mega could only accept .txt files. However, the study and trial of the more advanced Mega 7 showed that Mega could read more file formats than just .txt. The other file formats were .xml and .csv. During the research the test with Mega failed to obtain data directly from NCBI web service, so, data had to be exported to acceptable file formats before use with Mega 6 and 7 [66].

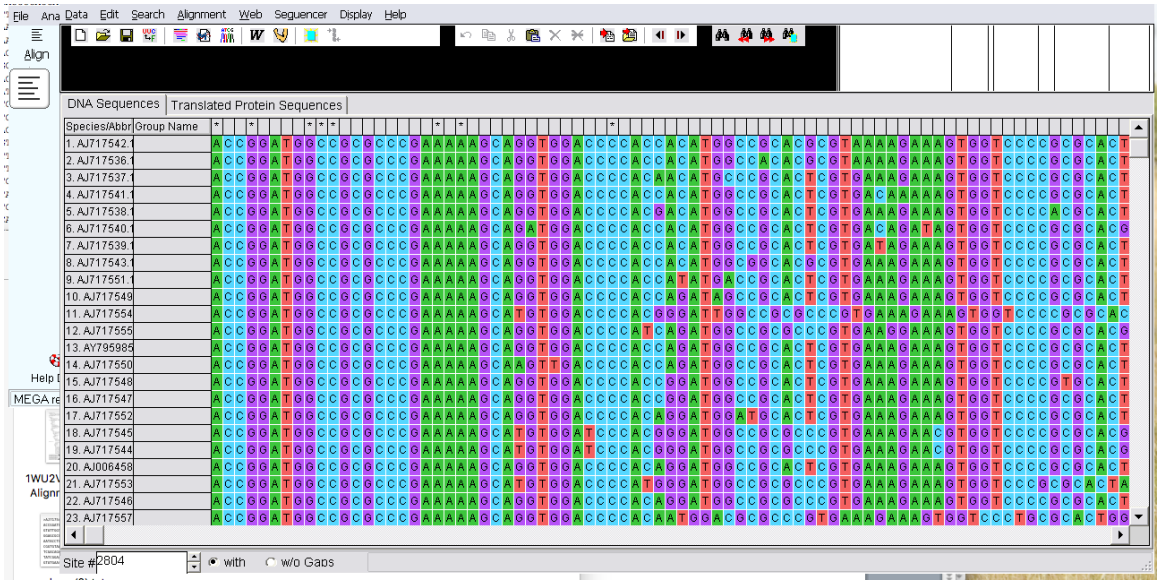


Figure 4-4: Neat colour coded output of a multiple sequence alignment in Mega 7

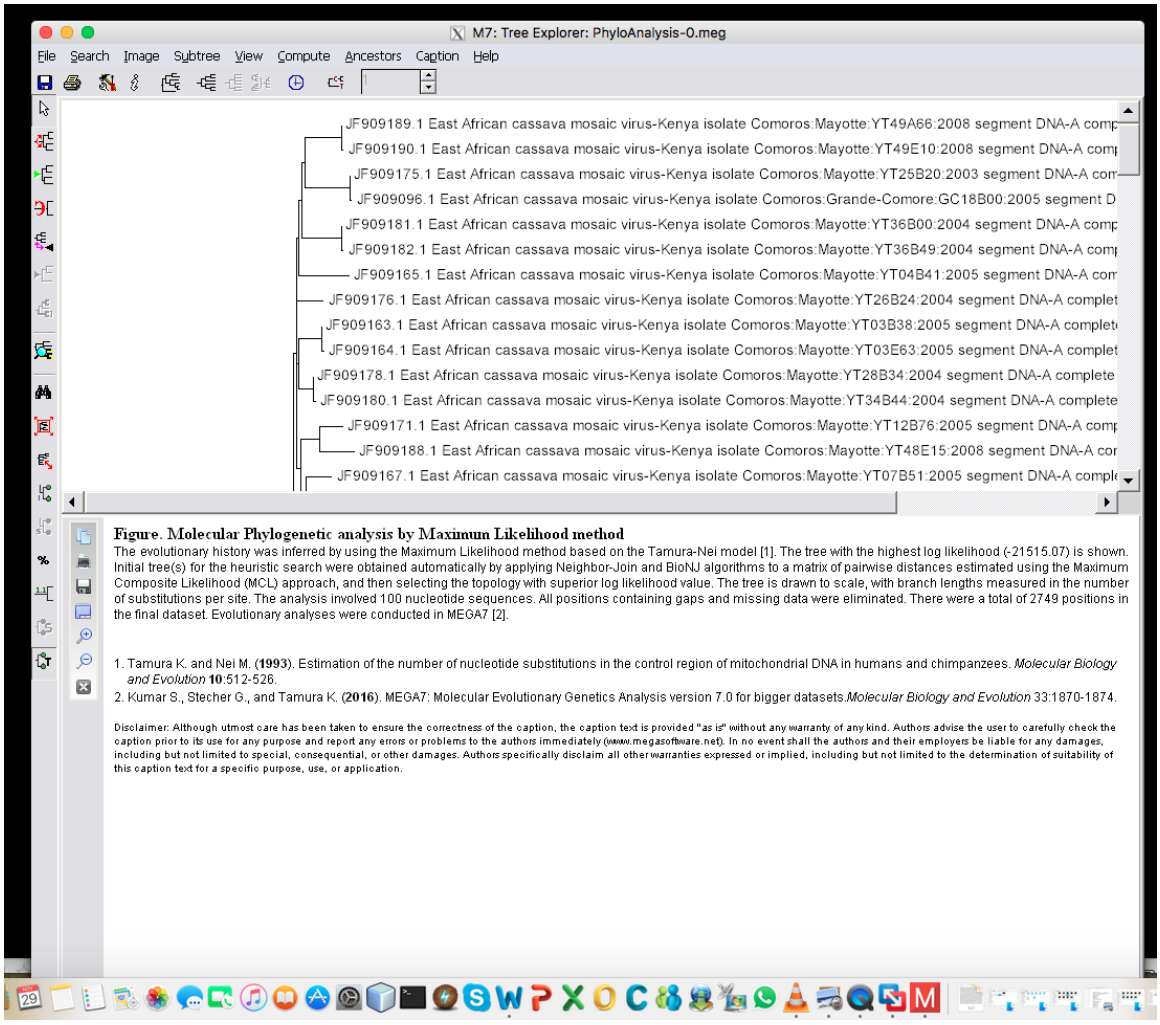


Figure 4-5: Phylogenetic tree output in Mega 7

4.3.4 National Center for Biotechnology Information

The National Center for Biotechnology Information (NCBI) advances science and health by providing access to biomedical and genomic information [106]. NCBI provides large online database resources of genome data and information. The resource also provides for tools that allow extraction of desired genome data from gene banks like GenBank [107]. One such tool is Entrez [108]. NCBI contains genome data, transcripts and protein sequences of various organisms [109]. The scientists at MMRS use the NCBI only for checking relatedness of their query sequence to other organisms whose data has already been fed in the NCBI databases [66] [110]. NCBI gives a sequence-by-sequence relatedness of organisms like viruses. NCBI can also output sequence alignment and its data can be exported to file formats such as .txt, .fas, .csv, .asn, .json, and .xml (Figure A-3).

4.3.5 Sequence Demarcation Tool

Sequence Demarcation Tool (SDT) is a pairwise genetic calculation based computer programme used to classify nucleotide or amino acid sequences. It is robust and producible. It generates high quality presentation and publication pairwise identity plots [111]. The pairwise multiple alignments are presented in form of a colour coded grid matrix (Figure 4-6 and Figure A-2) which is easy for a layperson to use during presentations. Computer science laypersons from MMRS still found the matrix presentation of alignments non user-friendly. The colour coded matrix presentation was the major attraction of the MMRS researchers to SDT [110]. During the study of the SDT, the additional information I learned is that SDT could read many file formats including .txt, .meg, and .fas.

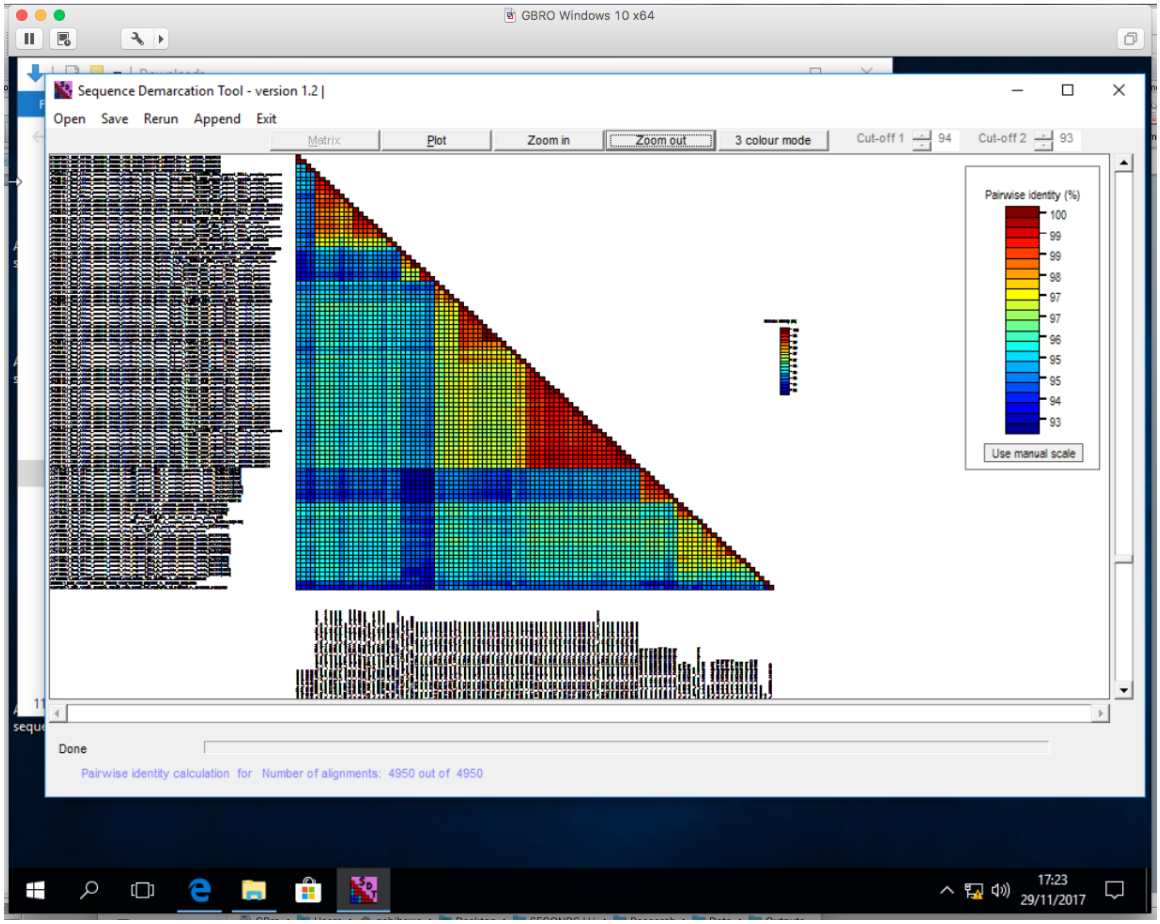


Figure 4-6: Colour coded matrix output of multiple pairwise alignment by SDT

4.3.6 Summary Of Software Usage Scenario By MMRS Researchers

Subsections 4.3.2 to 4.3.5 present what the software tools used by MMRS researchers could do. Despite all that they could do, researchers at MMRS only used some capabilities in each software tool. Their work begun with percentage identity relatedness analysis to sequence alignment, pairwise analysis and ended with evolutionary analysis through phylogenetic trees. A summary of this scenario is presented in Figure 4-7. One critical thing is that these pre-existing software tools could not communicate to each other, making their use fragmented.

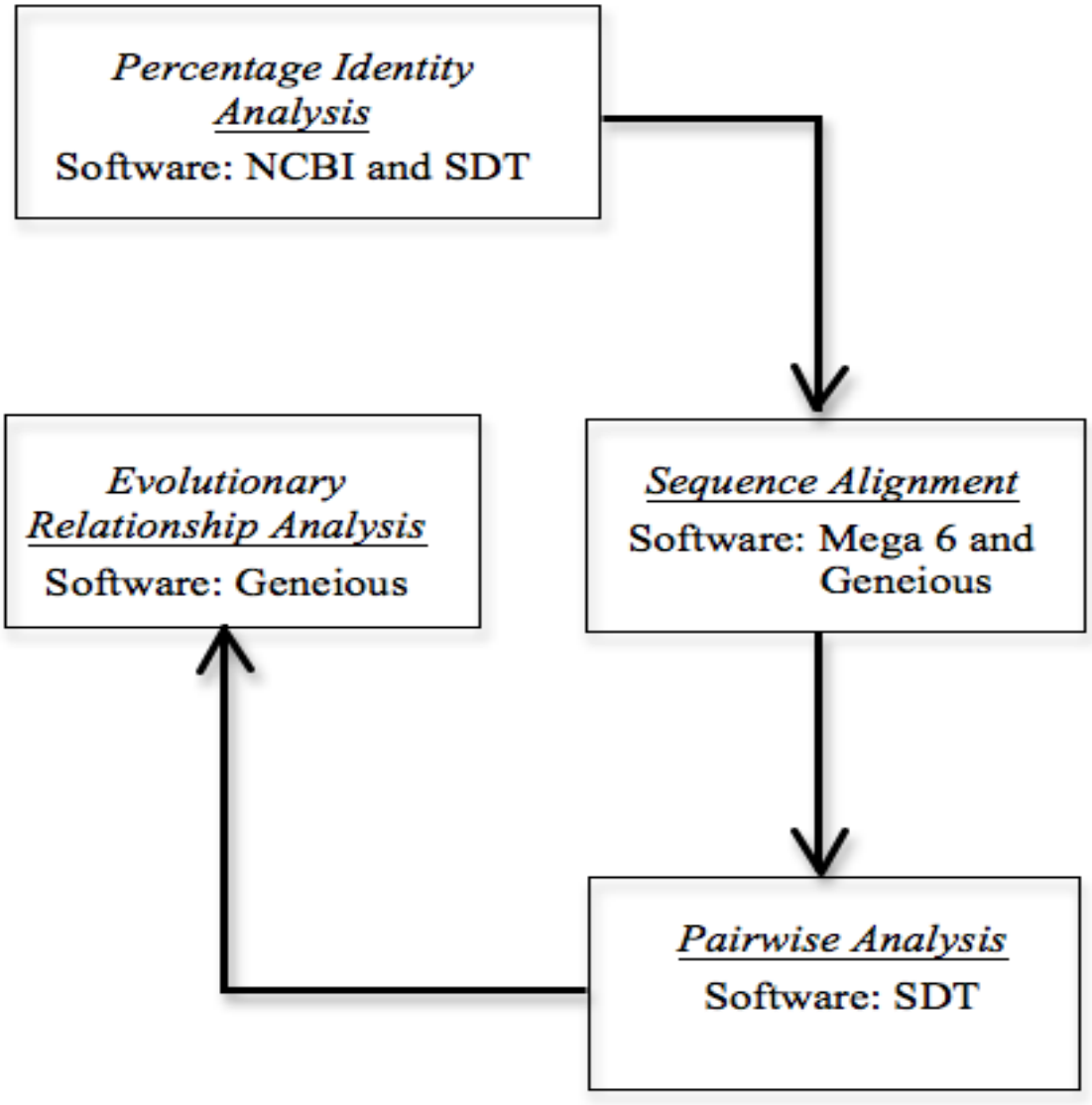


Figure 4-7: Scenario of software usage by researchers from MMRS. - "Adapted from [66]"

4.4 Ethical Consideration

The data used during this study was obtained from an open source library, NCBI, which did not demand ethical clearance before use. The use of the said data did not infringe on any human participant because there was no human factor in the interpretation and use of the data. Nevertheless, the study was based on the Directorate of Research and Graduate Studies (DRGS) Ethical committee guidelines. Ethical clearance was obtained from the University of Zambia Postgraduate studies Ethical committee and the reference number is **NASREC 2018 DEC-004**.

4.5 Proposed Workflow To Study ACMV Mutation

The review of the pre-existing bioinformatics software used to study mutation in the ACMV led to the proposal of a workflow to be used to determine the rate of mutation in the ACMV. This was based on the second objective of the research which was answering the question "What file exchange formats are fundamental for developing protocols for a workflow, from existing tools, in order to determine the mutation rate of the African cassava mosaic virus?". This entailed the determination of relevant protocols that would help link the pre-existing software tools so that they could communicate. This would avoid the manual feeding of data from software to software or the conversion of data files from format to format before input into a software tool. To achieve this, extensive reading was required, which led to the determination that the whole process of analysing the ACMV genome data could be done using open source libraries in biopython. The libraries and modules that could be used to analyse the ACMV genome data are summarised in Figure 4-8 as the proposed workflow to be used to study mutation in the ACMV. A brief background to biopython is given in section 4.5. The modules and protocols in the proposed workflow to be used to study the ACMV are explained in the results chapter in section 5.5.

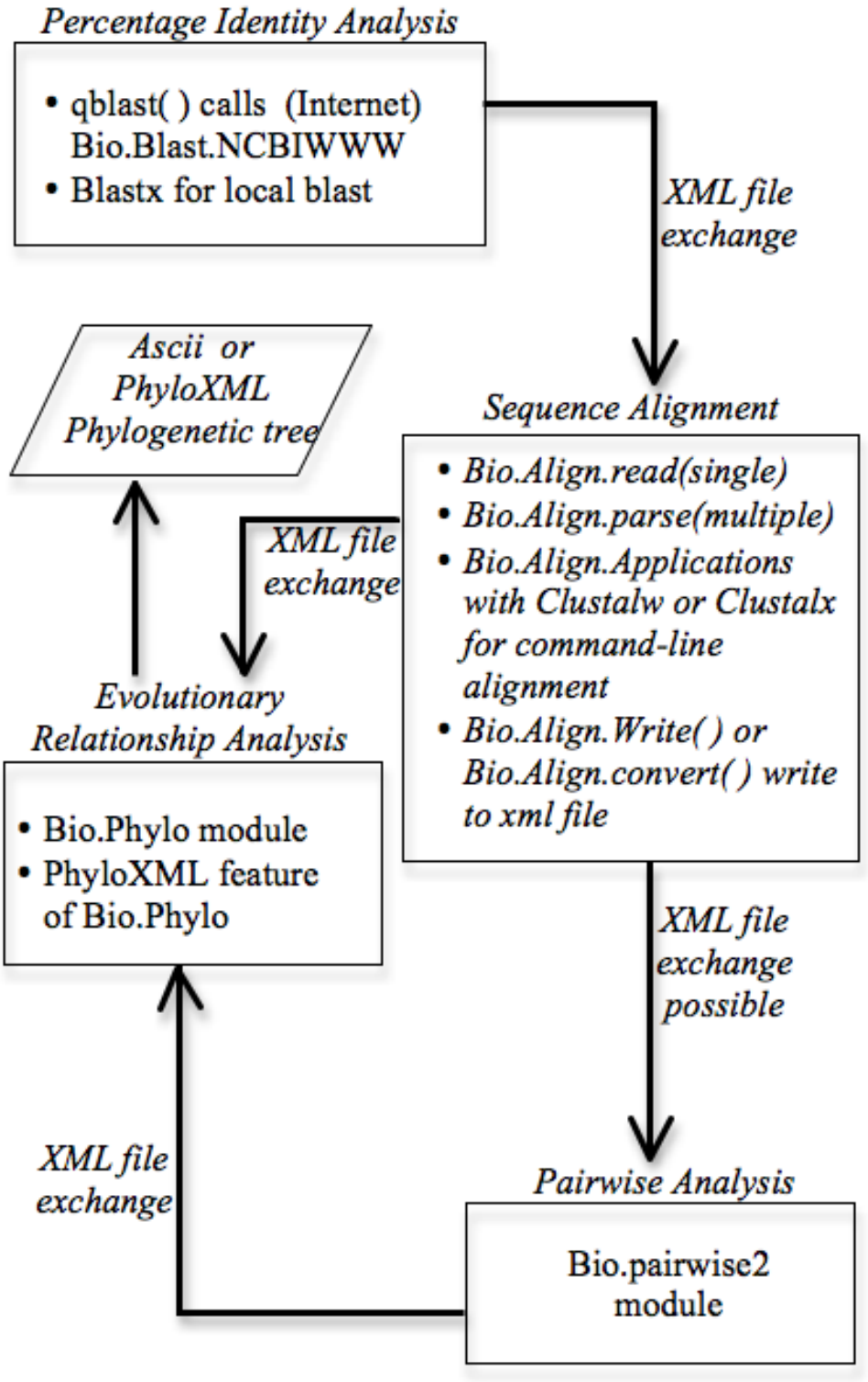


Figure 4-8: Proposed workflow to determine ACMV rate of mutation. - "Adapted from [66]"

4.6 Biopython

Biopython is explained in section 3.7 of chapter three. While studying to understand the software tools used by researchers at Mount Makulu research station I dedicated time to find other cheap alternative tools to use for the analysis of the ACMV genome data. Fortunately, biopython has modules that can ably be used for percentage identity of organisms, modules for multiple and pairwise sequence alignment and modules for evolutionary relatedness analysis. Biopython is open source and hence can allow researchers with a meagre budget to participate in the study of the ACMV. Percentage relatedness could be done through a local or global BLAST. Modules that searched and retrieved data from the NCBI exist. I tried them out and they gave output. The study of biopython was both exploratory and experimental because I had to first know what capabilities existed with biopython and later started applying the knowledge by comprehensive trials on the modules and methods that could achieve the four desired stages of studying the ACMV. The detailed explanation on the biopython modules that can be used to analyse ACMV genome data is given in section 5.5 of chapter five. The results of the experimentation using the biopython libraries are given in sub-sections 5.5.1 to 5.5.4 and section 5.6.

4.7 Single Page Application

The ultimate aim of the research was to attain comprehensiveness, user-friendliness and a fast response time in software tools to help determine the rate of mutation of the ACMV. Determination of mutation rate of the ACMV would only be achieved if overheads in response time were excluded in order to make the bioinformatics software tools fast enough to beat its mutation rate. Using biopython open source libraries, a single page application (SPA) that integrated html, XML and SVG could be developed. This research could only go up to a proof of concept on the development of the SPA. A fully fledged SPA would be developed in future works. A SPA is mostly a web application that is reactive and uses one window for all output with a dynamic data flow [83].

Summary

The research was based on three objections that contributed to attaining the aim (chapter one). The ultimate goal of the research was to improve performance in a selected set of bioinformatics software tools. Total response time of the software tools through the workflow needed to be improved. One of the overheads was failure by the existing tools to communicate because they did not have a common protocol. I, therefore, needed to find a way of integrating XML in the data exchange between the tools. Fortunately, biopython provided an even better alternative to just linking the existing tools because it has modules that can bring out all stages of the ACMV genome data analysis with XML output and input between the stages of analysis. Another overhead that could be cut off is the navigation from window to window when using a single tool. This could be cut off using a SPA. An easy workflow was the answer to the quest for getting results of the ACMV study fast enough to beat its mutation rate. Table 4.2 summarises the methodology in relation to the objectives and research questions.

Table 4.2: Summary table of objectives, research questions and methodology

Objective	**Research Question**	**Methodology**
To conduct a study of on-line and standalone software tools used by Zambian agricultural and biological researchers to analyse the ACMV genome.	Which of the pre-existing on-line and standalone software tools used to analyse ACMV genome data are generic enough to be ported into a comprehensive tool?	Reviewed pre-existing software tools used by MMRS researchers to study the ACMV to know what the tools could do. top
To propose a workflow for the prediction of the mutation of the ACMV.	What file exchange formats are fundamental for developing protocols for a workflow, from existing tools, in order to determine the mutation rate of the African cassava mosaic virus?	While reviewing the pre-existing tools and experimenting with biopython I took interest in the file formats of the outputs and the possible input file formats for the next process. The file formats are included in sections 4.3 and 5.2.
To develop a comprehensive tool that has a less complicated workflow and uses the tools and workflow in objectives "1." and "2." respectively.	How feasible is a comprehensive tool that does not require a user to switch from one application to the other?	I tested the biopython libraries for their capability to do the ACMV data analysis done by the pre-existing software tools and for their ability to output and input XML files. The results are discussed in section 5.5.

CHAPTER 5

RESULTS

In this chapter I first present results derived from the review of pre-existing software tools that are used by researchers at MMRS to study the ACMV. I then discuss the open source biopython modules, discuss the proposed protocols and also discuss the proposed workflow, all to be used to determine the mutation rate of the ACMV faster than it has been done before this research. The proposed solution is meant to improve performance in any software tool that would be built based on the proposed workflow. The performance is in terms of total response time of the bioinformatics software. When determining total response time, the human interaction with computers is excluded; only the computer's response time when processing data and outputting it is considered. In this vain, the computer's processing power is also important. The best performing software tool from the pre-existing ones (Mega), is compared against the proposed solution, which is divided into the online based solution and the offline solution.

5.1 Pre-existing Software Tools

Bioinformatics software have emerged to alleviate the trouble of the biological scientist who works with plenty genomic data. In the past these scientists would use manual methods to extract DNA/RNA and protein sequences and analyse them. Now, they no longer use manual methods to analyse the genomic data as long as they are stored as soft copies. MMRS scientists use the selection of software tools given in Tables 5.1 to 5.3. Columns 2, 3 and 4 in the table show what the software tools can do, what MMRS uses them for and their perceived limitation. As presented in the three tables, they read

and output different file formats during their processing. Each of the tools has some limitations with regards the analysis of the ACMV genome data. An example is Mega 6 that can not read XML file formats. XML may be common among the other four but they do not have a communication mechanism especially that some like Geneious are proprietary. The information in Tables 5.1 to 5.3 also brings out the fact that each tool is not comprehensive on its own owing to some of their output not being appreciated by lay users. The information of the usage and capabilities of the software tools was first obtained by interviewing a researcher at MMRS. After obtaining the information I experimented with the five tools using the initial data obtained from the MMRS researcher. The data is open source and can be obtained from the NCBI website. The information on the file formats that these tools could read and output was important in determining the protocols to integrate their usage.

5.2 The Pre-existing Software Do Not Communicate

The five software tools listed in Tables 5.1 to 5.3 do not talk to each other. Four are desktop standalone applications while NCBI is a browser based online web application. Scientists at MMRS have been collecting virus strain samples, preparing their DNA sequences in a wet environment and feeding the sequences in the NCBI interface to collect any related sequences for percentage relatedness and evolutionary analysis. From the NCBI, sequences that match the query sequence with an at least 80% relatedness are considered related to the new virus strain. They could either be the same virus strain or mutant strains of the same species. A percentage relatedness less than 40% is considered insignificant. An e-value (expected value) is used because the search (BLAST) uses heuristics which are approximations. The lower the e-value, the greater the reliability of the hits from the NCBI online genomic library of sequences. So, the researchers at MMRS essentially use NCBI for database search of related sequences. For those applications that do not have NCBI web service capability, data output from NCBI has to be exported to files before using it with them. Mega is one such. Although most of them communicate with the NCBI, because they are standalone applications with no protocols to communicate, they do not communicate with each other. Data is moved manually from application to application for any desired step of the genome data analysis.

Table 5.1: Software tools used by MMRS scientists, their capabilities, current use and limitations - Bioedit

Software	**Capabilities**	**Use by MMRS**	**Limitations**
BIOEDIT	1. Database search for related sequences by percentage. 2. Multiple sequence alignment. 3. Pairwise sequence alignment. 4. Has NCBI web service capability. 5. Can import from clipboard. 6. Can read a number of formats including .txt, .fas, .fasta, .fst, .xml, .meg (from Mega) as along as they are fasta formats.	Only used by MMRS scientists to export multiple sequence data to fasta .txt files; believed that only exports from BIOEDIT could be read by the other software tools.	1. Poor response time for multiple sequence alignment. 2. Poor response time for creating phylogenetic trees.

Table 5.2: Software tools used by MMRS scientists, their capabilities, current use and limitations - Geneious and Mega 6

Software	**Capabilities**	**Use by MMRS**	**Limitations**
GENEIOUS	1. Has NCBI web service capability. 2. Multiple and Pairwise sequence alignment. 3. Can export in multi formats including .txt, .geneious, .csv. 4. Produces nice phylogenetic trees.	1. Used mostly for phylogenetic tree production. 2. Once in a while uses Geneious for all four analyses	1. Poor look and feel of output window. 2. Re-feeding data manually for the two stages (alignment and phylogenetic tree production). 3. Proprietary software (cost of obtaining it).
MEGA 6	1. Multiple sequence alignment. 2. Phylogenetic tree production. 3. Can read many file formats.	1. Multiple sequence alignment. 2. Generation of phylogenetic trees.	1. Can not read .txt, .xml, .cvs. 2. Does not import from MS Word files.

Table 5.3: Software tools used by MMRS scientists, their capabilities, current use and limitations - NCBI and SDT

Software	**Capabilities**	**Use by MMRS**	**Limitations**
NCBI	1. It is a free genomic data library search for related sequences. 2. Provides tools for genomic data analysis. 3. Exports to many formats including .txt, .fas, .cvs, .asn, .json, .xml 4. Can align multiple sequences.	MMRS scientists only use NCBI for related sequences search.	1. Search results presentation not pleasing to computer science laypersons. 2. Will not display but export phylogenetic trees and alignments.
SDT	1. Multiple sequence alignment. 2. Presentation of pairwise alignments in a colour coded grid matrix. 3. SDT can read many file formats including .txt, .meg, .fas.	MMRS scientists use SDT for its colour coded grid matrix presentation of sequence-by-sequence comparisons.	Has no NCBI web service capability; text to be exported to files before use in SDT

5.3 The Pre-existing Software Are Not Very Comprehensive

A comprehensive software tool should have capabilities for all the four main stages of genomic data analysis. The four stages listed in Figure 4-7 are percentage identity analysis (BLAST search), multiple sequence analysis, pairwise sequence analysis and evolutionary relationship analysis (using phylogenetic trees). From Tables 5.1 to 5.3, each application lacks in at least one desired output. If it has capabilities for all, at least one of them is not pleasant to the computer science lay person's eye. Bioedit lacks in response time; Geneious lacks in good look and feel as well as automatic data feed into the different stages of analysis (alignment to phylogenetic tree); Mega will not read .txt, .xml, .csv and .docx files; NCBI itself will not display an alignment in its window, instead it will export it to a file; SDT did not communicate with NCBI. For Geneious, I had to input data for an alignment to occur and input the same data again to produce a phylogenetic tree. Sequence alignment and phylogenetic tree production would each start from the beginning to process instead of the phylogenetic tree generation continuing from the alignment. The sequence alignment output, as seen in Figure 4-3 would not easily be interpreted by a layperson in computer science or information technology. These varying capabilities are the reason why the researchers at MMRS used several tools to just analyse one set of data. Apart from some of these software tools not being appreciated all round in all the four genome analysis stages, some like Geneious have a complicated workflow on their own. In Geneious, data has to be fed in for sequence alignment and the user has to feed in data to obtain a phylogenetic tree diagram. All these overheads in the workflow are time wasting.

5.4 Performance Of The Pre-existing Software Tools - Total Response Time

In order to beat the mutation rate of the ACMV, tools used need to give a quick response time. This would result from them being able to communicate with each other in such a way that there would be automatic feed of data into each one of them if they were installed on one computer. The other reduction in response time would result from having a single comprehensive tool that would cut off data transfer from software tool to software tool. The third way to reduce response time is having software tools that have very fast algorithms and having computers with high performance computing. The fourth way reduction in response time would be achieved would be the software tool's ability to download genomic libraries to local storage to cut off the unforeseen factors that come with retrieval of data from online libraries. These factors could be inconsistent connectivity and low bandwidth of the connectivity. The review of the five bioinformatics software used by MMRS researchers in Zambia to study the ACMV was first done using the MMRS researcher's computer which was a core i5 processor laptop with 8 gigabytes random access memory. The total response time to analyse 92 sequences of the ACMV through all the four stages was nineteen (19) hours on a 3G internet network. This was tried only once before resorting to using an i7, quad core processor laptop with sixteen (16) gigabytes random access memory for the rest of the trials. The response time for each software tool from the later laptop is what was recorded in Figure 5-1. The average total response time from five trials with Mega 7 was an average of 80 minutes to output a multiple sequence alignment. For the same task and same number of runs, SDT took an average of 206 minutes and Geneious took an average of 86 minutes. Mega out performed them. Human interaction with the computer was ignored in the consideration of average total response time. SDT was the slowest. The 92 sequences used during the experimentation which gave the response time recorded in Figure 5-1 was downloaded from the NCBI after obtaining the accession from the initial data obtained from the MMRS researcher. An accession number is a unique identifier of every genomic sequence in the online library of the NCBI. In the absence of an accession number, a full DNA or RNA sequence can be used to obtain multiple sequences from the NCBI library using the NCBI website.

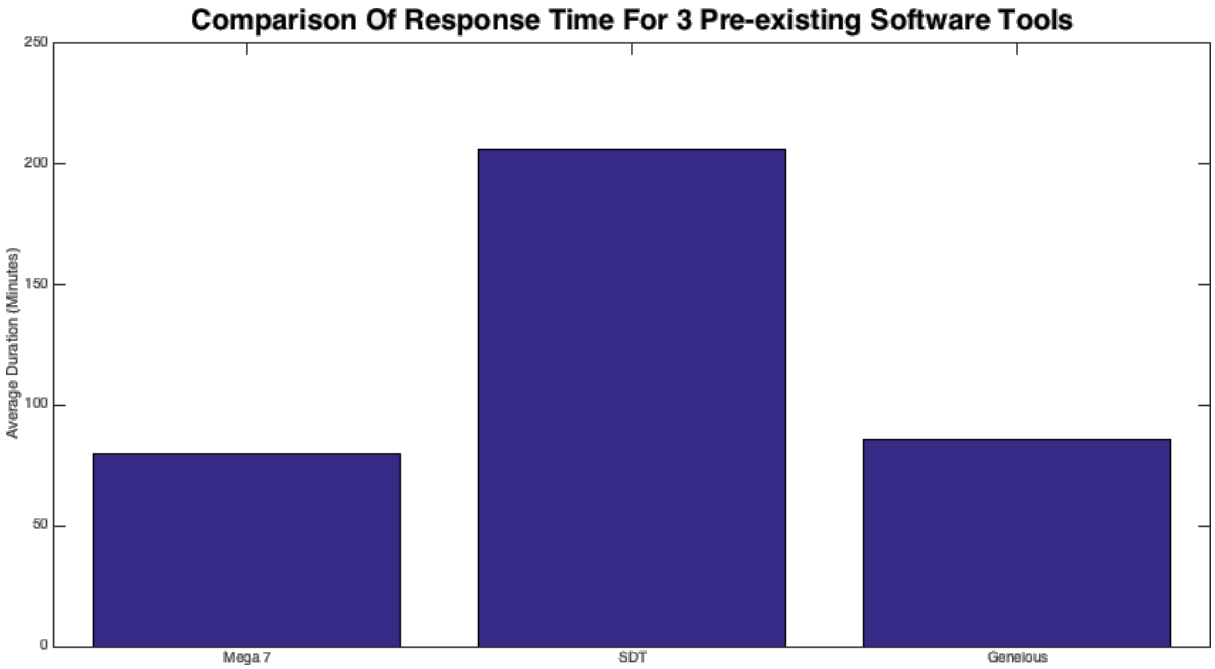


Figure 5-1: Comparison of response time in three pre-existing software tools popular for sequence alignment.

5.5 Proposed Biopython Solution

Studying biopython led me to the extraction of libraries that could carry out the same four stages of the ACMV genomic data analysis that were done by the five software tools listed in Tables 5.1 to 5.3. The libraries are presented in Figure 4-8 for each stage of analysis. One thing to take note of from Figure 4-8 is that XML output at each stage of ACMV analysis was possible when I tested the proposed biopython libraries. The modules used for every stage of the ACMV genome data analysis are presented in subsections 5.5.1 to 5.5.4. Section 5.6 outlines the proposed protocols to be used for an easier workflow that links the four genome data analysis to provide faster total response time.

5.5.1 BLAST Biopython libraries

Figure 4-8 gives the methods for carrying out each stage of the ACMV data analysis. BLAST could be done by the method *qblast()* for internet BLAST and module *Blastx* for BLAST of data stored in a local database. Data output from the BLAST could be exported to a XML file for the next stage of analysis. Listing A.1 shows the sample code for a BLAST and multiple sequence alignment using the proposed libraries in biopython.

The possibility of XML output from the BLAST algorithms fits in the interest to propose a protocol that is based on universal standard technologies. BLAST is used on the NCBI libraries. During BLAST, either an accession number or a full sequence is used in the *qblast()* method to retrieve multiple sequences with relatedness from NCBI. Using the *qblast()* method, a specific database must be stated, in this case the nucleotide database. The "blastn" algorithm is used to retrieve data from the nucleotide database. So, the method specifies the algorithm used, the database to retrieve from (like the "nr") and the accession number or sequence. The code that saves the file specifies the file format. Listing 5.1 writes an XML file as output.

5.5.2 Multiple Sequence Alignment Biopython Libraries

Figure 4-8 presents methods and modules which are able to produce a multiple sequence alignment. The python methods in the biopython libraries which can be used for sequence alignment and output, as proposed, are *Bio.Align.read()* which reads sequences from a results handle after alignment, *Bio.Align.parse()* which parses multiple sequences for alignment, *Bio.Align.Write()* for writing the read sequences to a file and *Bio.Align.convert()* which will convert the output to a XML file. The conversion to XML was tested using sample code in listing 5.2. The sample code for alignment is within listings 5.2 and A.1. Again, the ability to convert alignment output to XML fits with the proposed workflow to be used to determine ACMV mutation rate.

Listing 5.1: Sample code for outputting XML after a blast

```
1  # coding=utf-8
2
3  from Bio.Blast import NCBIWWW
4
5  result_handle = NCBIWWW.qblast("blastn", "nr", "AJ717542.1",
       format_type="XML")
6  save_file = open("Pathto/file.xml", "w")
7  save_file.write(result_handle.read())
8  save_file.close()
9  result_handle.close()
```

5.5.3 Pairwise Sequence Alignment Biopython Libraries

The third stage of analysis of ACMV genomic sequences is the pairwise alignment. Figure 4-8 shows the biopython module that can be used to carry out a pairwise sequence alignment. Pairwise sequence alignment is necessary to identify deletions and insertions. The query sequence was analysed against each of the BLAST output sequences for signs of mutation; signs of indels. The biopython module used for pairwise alignment is *Bio.pairwise2*. Pairwise sequence alignment can also be achieved by just parsing the BLAST output to arrange the output sequences in any way desired. For pairwise alignment, the sequences can be arranged in pairs of the query sequence against each sequence retrieved from the genome library. Listing A.1 also integrates code for parsing BLAST output into pairwise alignments.

5.5.4 Biopython Libraries for Phylogenetic Tree Creation

The fourth stage of the ACMV genome data analysis is the creation of phylogenetic trees to analyse the evolution of the ACMV through mutation. Listing B.2 presents the sample code which was used to obtain an ascii phylogenetic tree (Figures 5-2 and 5.3). Listing 5.2 is sample code which was used to convert the ascii tree to a PhyloXML tree. The biopython *Bio.Phylo* module and *Phylo.XML* feature of the *Bio.Phylo* module can be used to create the ascii and the phyloxml types of phylogenetic trees respectively. A .dnd file is used to create an ascii tree. The .dnd file is one of the files produced when carrying out an offline alignment; .aln and .dnd files are produced at the end of the process. The .aln file can be converted to a .xml file which fits well in the proposed workflow.

Listing 5.2: Biopython code for outputting a PhyloXML tree

```
1
2       # coding=utf-8
3
4       from Bio import Phylo
5
6       from Bio.Phylo.PhyloXML import Phylogeny
7
8       dnd_file_input = "Pathto/file.dnd"
9
10      xml_file_output = "Pathto/file.xml"
11
12      PhyXMLTree = Phylo.read(dnd_file_input, 'newick')
13
14      Phylo.convert(dnd_file_input, 'newick', xml_file_output, 'phyloxml')
15
16      phyTreeFromXML = Phylogeny.from_tree(PhyXMLTree)
17
18      print(PhyXMLTree)
```

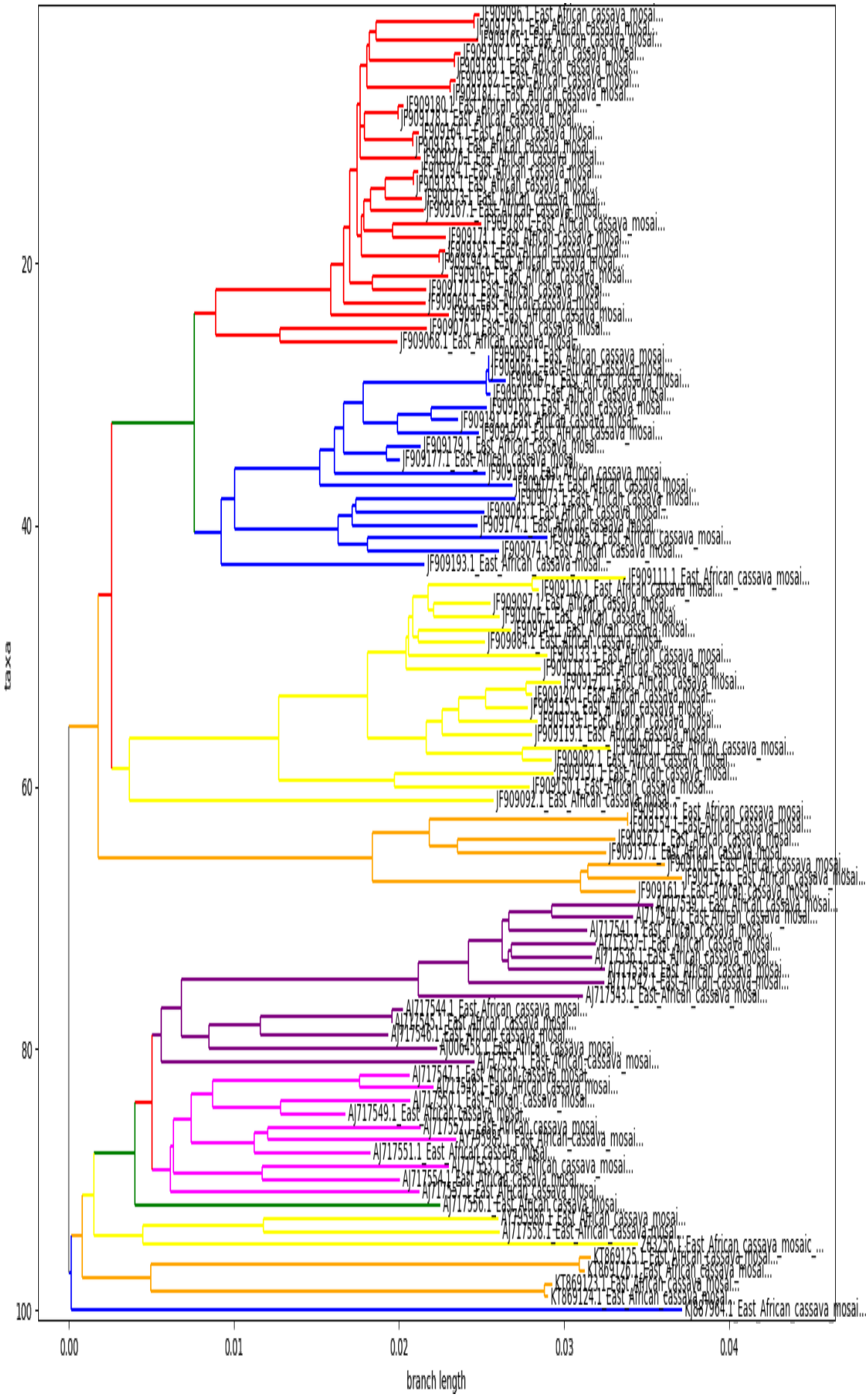


Figure 5-2: Output of Biopython ascii phylogenetic tree code

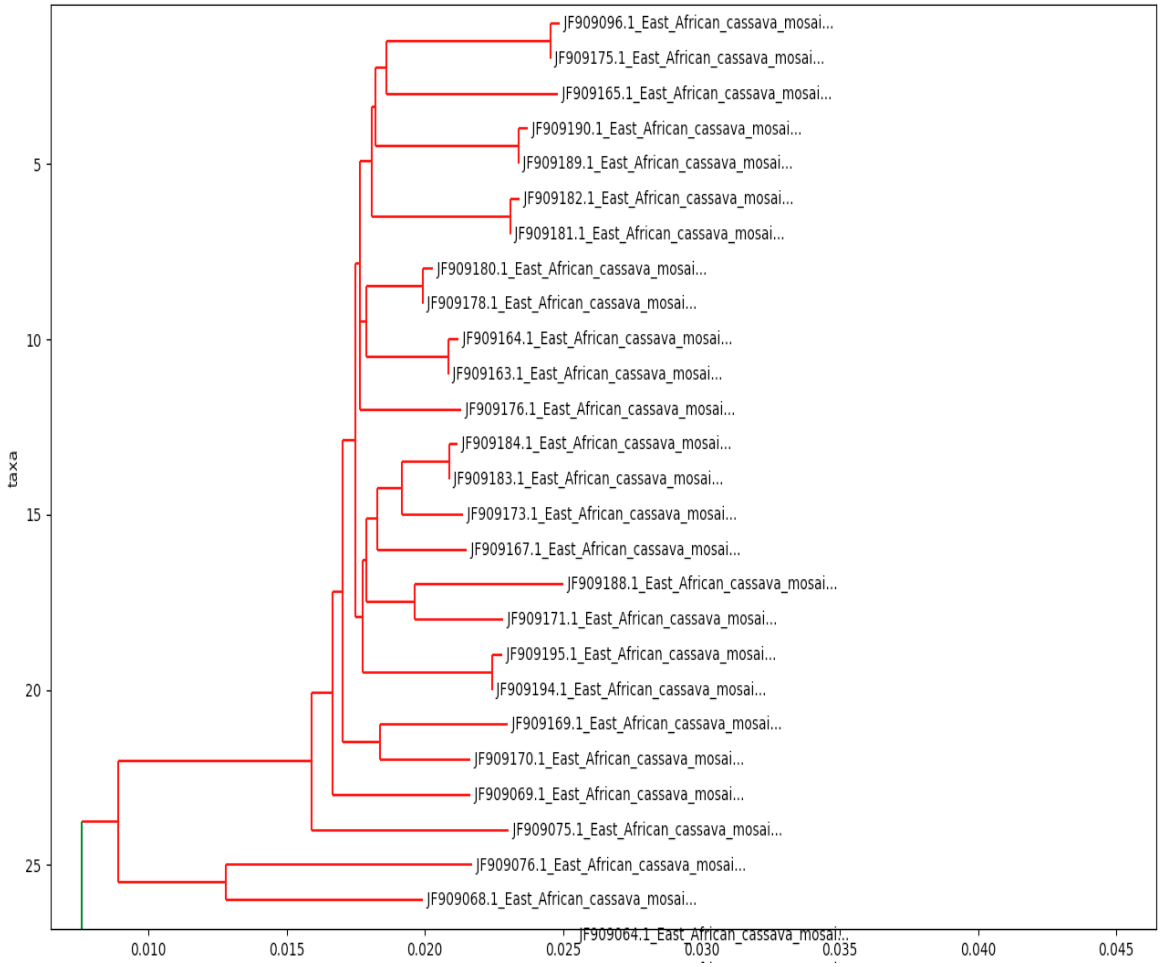


Figure 5-3: Close in of Biopython ascii phylogenetic tree output

5.6 Response Time for Proposed Solution

Having extracted biopython libraries which could be used to carry out ACMV genome data analysis, I decided to test performance of the proposed solution by response time. The proposed solution was in two options, offline solution and online solution. Offline meant creating a local genome database before analysis. The online option used data immediately as it was retrieved from NCBI and parsed it. It was compared against the best of the pre-existing software tools reviewed and summarised in Figure 5-4. The computer that was used to compare total response time between the proposed solution and Mega (the best of the pre-existing tools) was the same laptop computer used for the review of the pre-existing software tools. It was an i7, quad core processor laptop computer with sixteen (16) gigabytes random access memory. Both the offline and the online biopython methods were run five times to match the number of runs for Mega.

The average total response time for each was calculated by basic average calculation in Microsoft Office Excel. A stop watch from a smart phone was used to take the start and end time during both the pre-existing software comparison and the comparison between the proposed solution and Mega. From the results presented in Figure 5-4, it is clear that the use of the proposed libraries from biopython out performed the pre-existing software tool (Mega), which had the best performance during the comparison of the pre-existing tools. To go through all the four stages of the ACMV analysis took only a third of the time it took with Mega. The proposed solution had two options; starting with online BLAST or starting with local storage BLAST before the rest of the analyses. The offline solution took slightly longer than the online option.

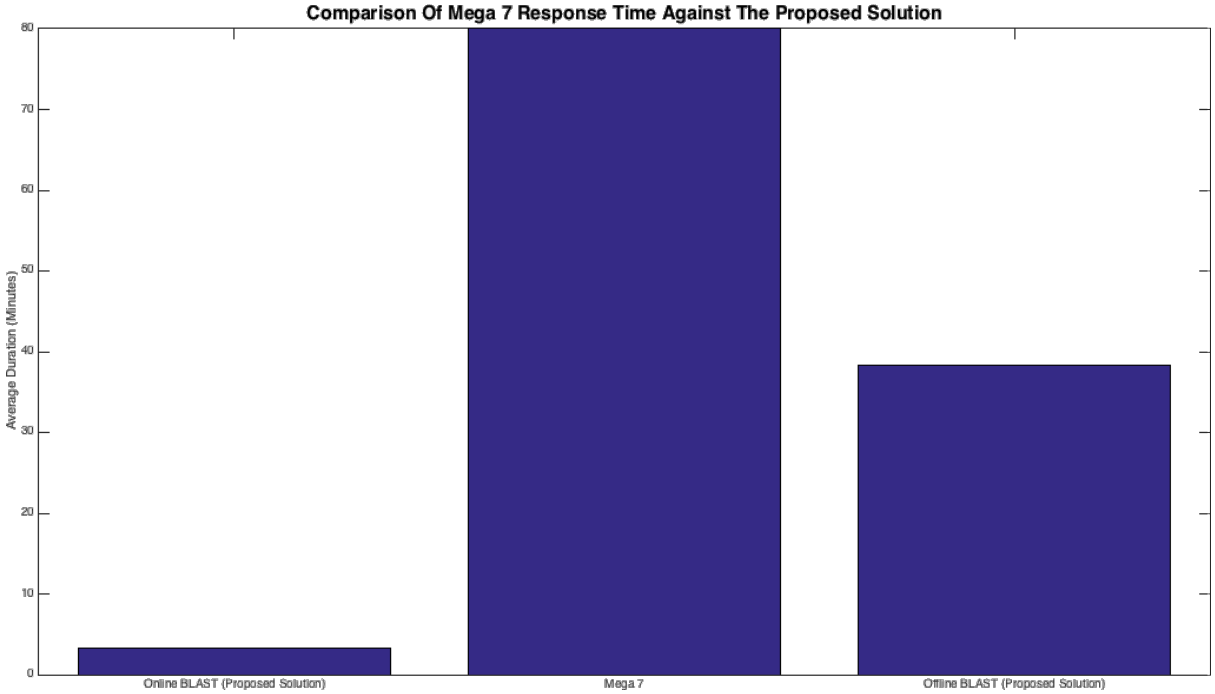


Figure 5-4: Mega 7's response time was compared against that of the proposed solution (using both online and offline BLAST)

5.7 Proposed Protocols

Three protocols have been proposed to be used for an easier workflow for the study of the ACMV genome data. The protocols are the percentage identity protocol, the blast to alignment protocol and the alignment to evolutionary relatedness protocol. Sections 5.7.1 to 5.7.3 outline these protocols.

5.7.1 Percentage Identity Protocol

The percentage identity protocol involves the following steps.

1. Provide either an accession number of a virus strain or a query sequence of the virus strain under comparison.

2. Retrieve database sequences from genomic libraries using a method that integrates accession number or query sequence provided. If there is no output, there are no matches, so input new accession number or sequence.

3. If there is output, check file format. If the retrieval method does not obtain an XML file, convert output to XML in fasta format of the sequences. Fasta format has every sequence identity starting with the ">" symbol.

4. Use an algorithm which in linear bound or polynomial for a worst case situation. Heuristic algorithm and its derivative is preferred though not as efficient as exhaustive search algorithms. The aim is to just study mutation and ascertain whether the query sequence is that of an existing or new virus.

5.7.2 Blast To Alignment Protocol

The blast to alignment protocol involves the following steps.

1. Automate the reading of the saved XML file for alignment to take place.

2. Parse the data on the XML file and apply the multiple and pairwise alignment methods to produce an alignment.

3. If the output of the alignment is any file format other than XML, convert the file to XML file to be used in the phylogenetic tree generation but keep a .dnd file also.

4. Use an algorithm which is linear bound or polynomial for a worst case situation.

5.7.3 Alignment To Evolutionary Relatedness Protocol

The alignment to evolutionary relatedness protocol involves the following steps.

1. Automate the reading of the saved XML or .dnd file for use with a method to generate a tree diagram.

2. If only the XML file exists, use it with a method to create a phyloxml tree or convert it to .dnd or other formats that can be used to produce an ascii tree.

3. If both XML and .dnd files were written using the blast to alignment protocol, use the .dnd to produce the ascii tree or the xml file to produce a phyloxml tree.

4. Use an algorithm which in linear bound or polynomial for a worst case situation. Machine learning algorithm is recommended.

5.8 Enhanced Workflow - Single Page Application

The proposed easier workflow can be used to build a single application that runs as a single page application. The single page appication would be based on the capabilities of html, xml and SVG. The proposed application would run in a browser to access both online and local resources while sticking to one window in operation. A user starts by inserting a query accession number or sequence to perform a BLAST. If there are hits, the output is kept in both primary and secondary memory for use with other modules of the application. At the click of a button, within the same window, a multiple sequence alignment results. When a user clicks on one of the retrieved sequences a pairwise analysis of the query sequence and the retrieved sequence is given within the same window. Hitting the escape button of the computer takes the user back to the multiple sequence alignment. Clicking the top left corner of the multiple sequence alignment window brings to view the multiple sequence analysis by colour coded grid matrix. A click on any square of the grid matrix brings to view the pairwise analysis of the two sequences that converge at the clicked square with additional source information like the virus identity by name, accession number, where the virus was obtained from, the researcher who extracted the DNA sequence and any other identity information. Clicking the top right corner of the full grid matrix brings to view the phylogenetic tree of the multiple sequences. Areas of the tree can be expanded by clicking them or the branches of the tree. The escape button of the computer helps to close any pop up window in view to get back to either the grid matrix or the multiple sequence alignment. Figure 5-5 is a summary of the workflow for the proposed SPA and Figure 5-6 shows a window of the proposed SPA with a phylogenetic tree brought to view above the grid matrix multiple sequence analysis. The modules build so far of the SPA are a proof of concept

that the SPA can be build to improve workflow in the analysis of the ACMV genome data.

The underlying programming language for developing the proposed SPA is python, using biopython libraries. All the methods and modules for the BLAST, pairwise and multiple sequence alignments and phylogenetic tree generation are in python as given in Figure 4-8. PHP is used to create html dynamically for the display of the retrieved data as well as the information window where the accession number or query sequence is entered and retrieval parameters are set. As proposed, SVG is used to make the output window interactive and dynamic. XML is the data transfer or exchange format used to persistently store data as it is being processed from stage to stage of the ACMV genome data analysis.

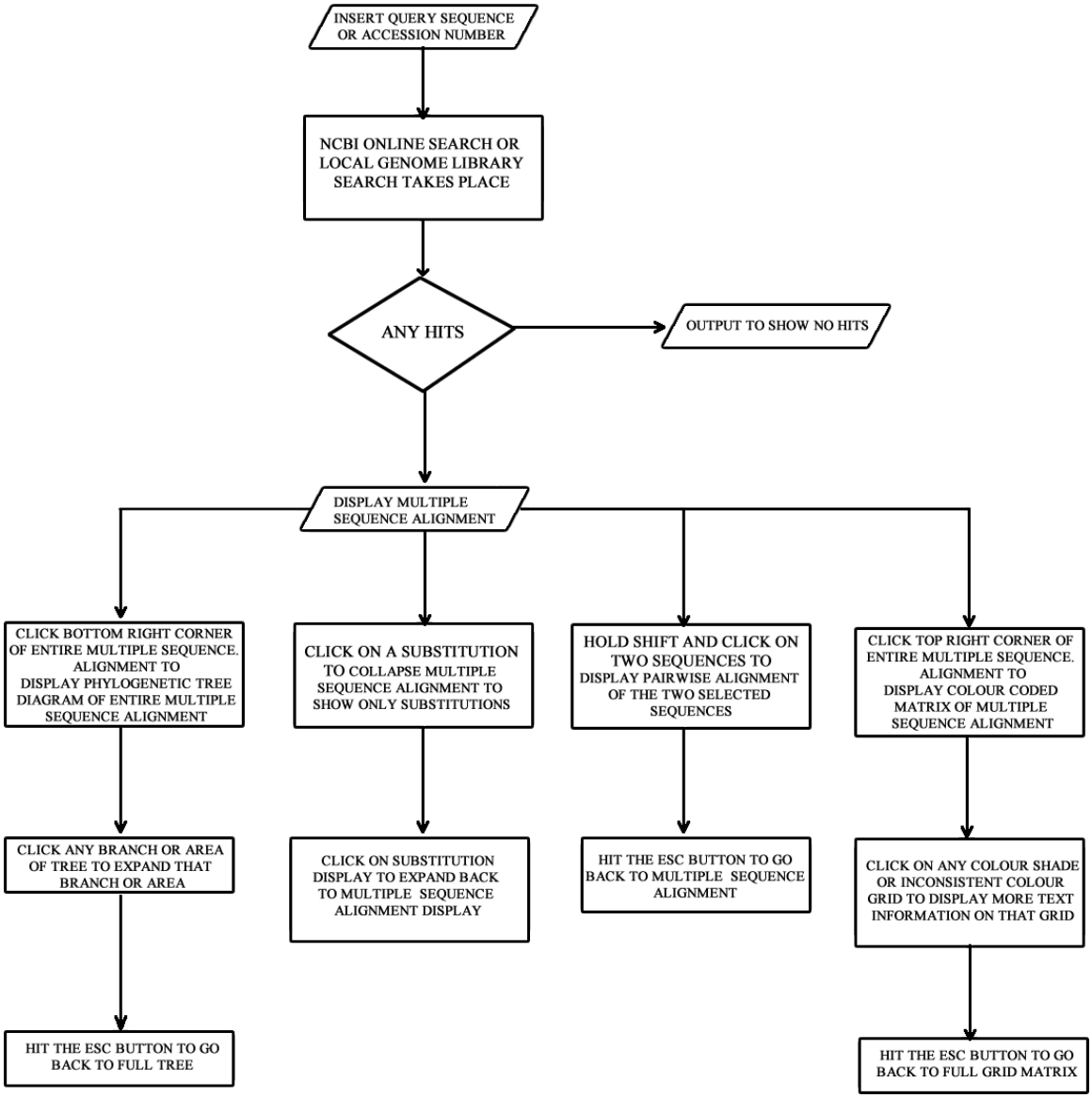


Figure 5-5: Enhanced workflow of the proposed solution for the analysis of the ACMV, using a SPA

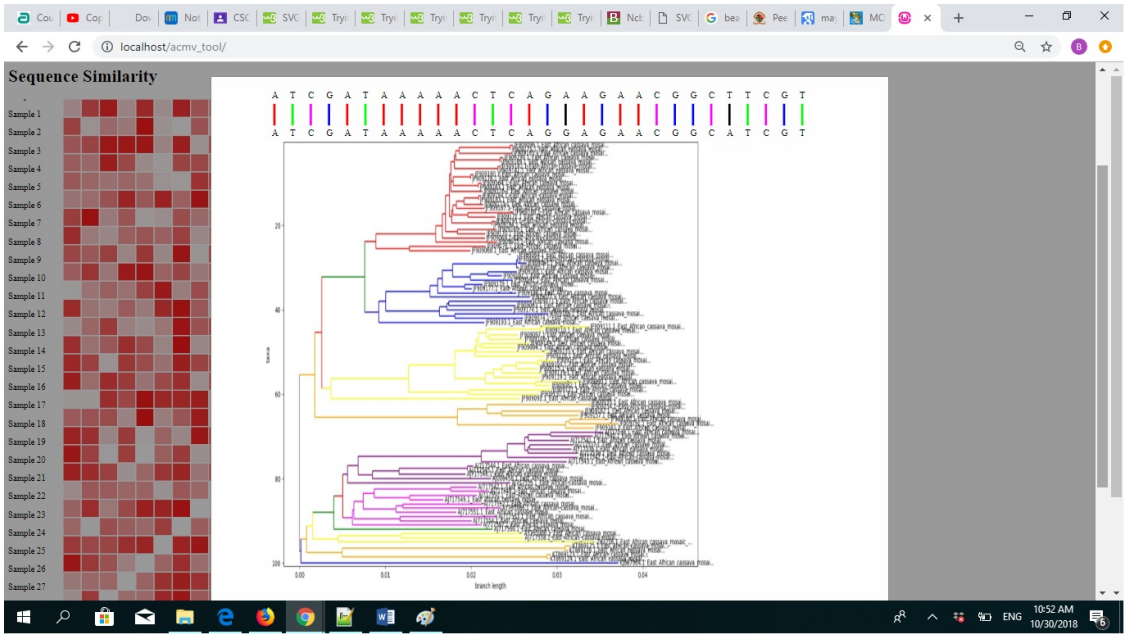


Figure 5-6: A window of the SPA showing the phylogenetic tree in view above the multiple sequence alignment grid matrix; a proof of concept

.

Summary

At the time of the research, the MMRS scientists used Bioedit, Geneious, Mega 6, NCBI and SDT predominantly for the analysis of the ACMV. Of the five software tools, only NCBI was web based, the rest were standalone desktop applications. Three of them had NCBI web service capability except SDT. The four standalone applications did not communicate with each other, data had to be fed manually from application to application. Each tool could only satisfy output requirements of some (not all) stages of the ACMV genome data analysis. Response time of three pre-existing applications which are popular for sequence alignment (at MMRS) was compared and Mega, the one that stood best was compared against the proposed solution. Mega, which had the best total response time of the pre-existing software tools took an average of 80 minutes to give output. The proposed solution had the online BLAST and offline BLAST options. The online BLAST option performed better than the offline solution on my laptop computer with computing power far less than that used by NCBI. The online option of the proposed solution took less than five minutes on average to process and give output against thirty-eight minutes taken by the offline solution on a laptop with low computing power. To further improve

workflow, aiming at reducing total response time, a SPA was proposed to be developed. The SPA could be developed using PHP for the graphical user interface, biopython for the data processing and html and SVG for data display and interactivity. XML would be the data transfer standard.

CHAPTER 6

DISCUSSION AND CONCLUSION

This chapter discusses the results presented in chapter 5 and ends with an academic conclusion of the research. A recommendation is included in this chapter. The discussion is on the proposed solution to the fragmented software tools, which have a complicated workflow, that are currently used for the study of the ACMV. The solution to the fragmented tools is in the proposed protocols to integrate the pre-existing tools and the easy workflow which can be used to build comprehensive software tools to be used to determine the rate of mutation of the ACMV. Both options of the solution, whether use of the protocols for an easier workflow with existing tools or building a new SPA works at improving total response time so that the mutation rate of the ACMV can be beaten.

6.1 Non Comprehensive Software With Complicated Workflow

There are four main stages of the ACMV genome data analysis, namely percentage identity analysis (using BLAST), multiple sequence analysis, pairwise sequence analysis and evolutionary relationship analysis (using phylogenetic trees). Researchers from MMRS use Bioedit, Geneious, Mega, NCBI and SDT to analyse the ACMV genome data. Often researchers that use these tools are not specialists in information technology and as such the tools used must be efficient and should have an easy workflow. Some of these tools are standalone desktop applications and still others like the NCBI are internet based. Tools such as NCBI will depend on factors such as bandwidth and connectivity for output. Not every tool can perform all the four stages required for the analysis of the ACMV

genome data to the satisfaction of the computer science lay person researcher. For example, Titus Alicai et. al. in [113] reported the use of a package called Phylogenetic Analysis by Maximum Likelihood (PAML) which Ziheng Yang in [114] said was not good for phylogenetic tree making. Another example is Mega which can not read xml file types. The NCBI itself will not display a multiple sequence alignment in a colour coded grid that shows indels. Geneious is only loved for its multiple sequence alignment and phylogenetic tree diagrams while SDT's multiple pairwise presentation by a grid matrix is appreciated. Bioedit has also not been impressive in phylogenetic trees output as far as the researchers at MMRS are concerned. In their view, Mega is very good for multiple sequence alignment and production of phylogenetic trees. One thing that comes out clearly from the results is that, as much as most of these software can communicate with NCBI they do not communicate with each other. This calls for manual feeding of data from software tool to software tool. In simple terms, these tools are fragmented with a complicated workflow because they do not perform all four stages of the ACMV analysis with satisfaction to the computer science lay person researcher. So, the researcher has to move data from tool to tool to get satisfaction with look and feel output for at least a stage of the analysis. This causes loss of time and delay in determining the rate of mutation of the ACMV. Some of the tools like Geneious have non user-friendly output of the multiple sequence alignment. Its output can not be used for a presentation to lay persons. Indels (deletions and substitutions) are difficult to spot when using Geneious. The non-friendliness of certain output of some software contributes to being non comprehensive calling for use of more than one software tool to go through the four stages of genome data analysis. This gets back to the same thing of time loss and delay in output. To circumvent this problem, I proposed the use of XML to develop a workflow of the existing tools that would be used to determine the rate of mutation in the ACMV. XML is a universal data transfer protocol which would allow communication between many disparate software tools as long as they are developed to transfer data via XML output and input. This type of data transfer would make data available to all tools sitting on the same machine and even in a distributed environment in order to turn manual feed of data into an automated one. Time is saved by so doing.

6.2 Proposed Biopython Solution and Protocols for The Easier Workflow

The research set out to find ways of integrating the fragmented pre-existing tools. It ended with a proposal to use biopython libraries to build applications that would be used to study the ACMV. Biopython is open source, as such, tools that would be built would be cheaper for many researchers, including student researchers, to use. This would increase the number of researchers working to determine the mutation rate of the ACMV so that it could be determined faster than it mutates. The biopython modules also proved to bring out results faster than the pre-existing tools as evidenced by results in Figures 5-1 and 5-4. Section 5.7 outlines the proposed protocols for an easier workflow to determine the rate of mutation of the ACMV. The protocols are based on xml, a universal data transfer standard. Other technologies exist, like json and yaml which could be used for data transfer between the tools but I chose xml because the proposed solution is based on python not java, for example. The proposed protocols ensure that data is stored in xml format and is read from xml files at each stage of the ACMV genome data analysis. The xml files will ensure interoperability and make possible data sharing between disparate systems and even in a distributed system.

6.3 Performance Of The Pre-existing Software Tools

The experimentation with both the pre-existing software tools and the proposed biopython modules was done on an i7, quad core processor laptop computer which had sixteen (16) gigabytes random access memory and MacOS operating system. Each software tool and each biopython proposed solution (offline and online options) was tested five times and the average total response time calculated using basic average calculation in MS Excel. A smart phone stop watch was used to take response time. When I tested the pre-existing software tools with 92 ACMV DNA sequences Mega 7 took an average of 80 minutes to output a multiple sequence alignment. For the same task, SDT took an average of 206 minutes and Geneious took an average of 86 minutes. Mega out performed them. Although SDT has some impressive output for the MMRS researchers it took over three hours to give response using few sequences like 92. The biological science

and agricultural science researchers at MMRS many times work with thousands of DNA sequences. Such slow applications would take a time too long to beat the mutation rate of the ACMV. The rate at which the ACMV mutates has not yet been determined. It is possible that the virus may mutate several times while the biological science researcher is waiting at least three hours to have first output for analysis. After that output, the researcher would still have to transfer data to another software tool which will take at least 1 hour 20 minutes to give output for few sequences. By the time the whole process is done the researcher would have waited for 9 consecutive hours to get output for few sequences. This response time would allow the ACMV to mutate before results are analysed or by the time analysis is done. Some users of these applications may have computers with smaller RAM storage; they will take longer than 9 hours to go through the analysis process. This was at least experienced when working with the MMRS researcher's i5 processor laptop which had 8 gigabytes RAM. It took nineteen hours to give response. This factor of response time during the analysis stages led me to read further in order to find a solution that would help reduce response time further. I settled on the use of biopython, an open source set of libraries under the python programming language. As seen in Figure 4-8, libraries exist in biopython, which could perform the task done by the pre-existing tools. Output and input between the analysis stages would be done using XML files. I tested the libraries and they worked. I tested the XML output and input, it worked. Figures A-3 to A-6 in the appendices are sample outputs (screen shots) during the tests. The process from BLAST to phylogenetic tree production, using the biopython libraries, took only an average of three (3) hours. This processing and output time of three hours was a third of the time it took to use the pre-existing tools on the same 92 sequences. Time saved like this would help beat the mutation rate of the ACMV. The use of the biopython libraries with online BLAST took less time than with local storage BLAST. It must be known that the NCBI has computers with high computing power. I strongly suggest that the use of the biopython libraries with a local storage as suggested by NCBI [115] would be faster than the use of online BLAST as long as computers with high computing power are used for local databases and processing of the four analytical stages.

6.4 Cost Of Doing Research

Most of the pre-existing tools used for the study of genomic data are proprietary. Some of the very good ones have prohibitive prices to researchers with an average income, especially student researchers. Open source software come in handy to help researchers with a meagre income to participate in research. The proposed use of open source biopython libraries will help more researchers to come on board and pool computer resources towards the ACMV study.

6.5 Proposed Workflow and The Single Page Application

The use of biopython libraries and XML was a drive in the right direction to reduce processing time and increase the rate at which output is obtained so that researchers could beat the mutation rate of the ACMV. There was, however, still one problem to tackle, that of fully automated data feed into every stage of the ACMV genome data analysis. In short, the workflow involved was complicated and time wasting. The full automation of data feed would save more time than had already been saved. I, therefore, decided to achieve it by the use of html, xml and SVG which are the major constituent of a SPA. The SPA is explained in section 5.8. The SPA would enhance workflow and cut off the time of transferring data from stage to stage, whether automatically or manually. The SPAs are the future of applications; SVG is the future of SPAs. With the SPA application that I proposed to develop, whose proof of concept was done, the user does not need to move from application to application or navigate from window to window. They remain in the same window but just bring to view what they need to see. The SPA can easily be used for projected presentations instead of transferring information to a PowerPoint presentation. I called the SPA used in the proof of concept (Figure 5-6), the svgenome. The full svgenome application would be completed as future works and will be documented in future publications. It is at that stage that researchers at MMRS would be availed the application to help determine whether it is comprehensive and has an easier workflow.

6.6 Conclusion

The aim of the research was to develop a workflow that uses pre-existing tools to offer a de-fragmented tool with an easy workflow that would be used to determine the rate of mutation of the African cassava mosaic virus. The research to provide a solution to the effects of the ACMV has been a nuisance to scientists for a long time. I took note of the factors that have prevented success for a long time as fragmentation of software tools used and complicated workflow of the tools causing delayed output, costly tools that prohibit the increase in scientists doing research on the ACMV, and longer response time of the software tools. I, therefore, proposed the use of open source biopython libraries to build a comprehensive tool with an easy workflow, the use of XML for data transfer so that data transfer would be automated and the integration of html, XML and SVG to develop a SPA which would be comprehensive. I went on to develop the proof of concept on the possibility of building the SPA. Research time allowed me to go up to the proof of concept not a full fledged SPA. The SPA would be completed later and its comprehensiveness and easier workflow put to test. The proposed easier workflow and the SPA to be build would greatly reduce response time and help biological and agricultural science researchers find solutions to the adverse effects of the ACMV before its next mutation. The break through in determining the rate of mutation of the ACMV would help provide food security to Zambia knowing that cassava is one of the drought resistant alternatives to the staple food (maize) which is now highly affected by climate change.

6.7 Recommendation

I recommend the use of open source modules to build open source tools that would be cheaper to use in order to encourage many participants in the study to bring solutions to the ACMV which has ravaged cassava for over one hundred years. The recommendation is based on the confirmed presence of open source biopython libraries which can ably be used for analysing genome data.

6.8 Future Work

A proof of concept that a SPA could be built was done. There is need to complete the building of the full fledged SPA and avail it to biological and agricultural science researchers at MMRS to use for their study of the ACMV.

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 resistance to cassava mosaic geminiviruses in cassava using genotyping-by-sequencing and its implications for breeding," *Virus research*, vol. 186, pp. 87–96, 2014.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 3rd ed. MIT press, 2009.

[4] N. C. Jones, P. A. Pevzner, and P. Pevzner, *An introduction to bioinformatics algorithms*, 1st ed. MIT press, 2004.

[5] L. Mzyece, M. Nyirenda, M. K. Kabemba, and G. Chibawe, "Forecasting seasonal rainfall in zambia–an artificial neural network approach," *Zambia ICT Journal*, vol. 2, no. 1, pp. 16–24, 2018.

[6] M. D. Hendy and D. Penny, "Branch and bound algorithms to determine minimal evolutionary trees," *Mathematical Biosciences*, vol. 59, no. 2, pp. 277–290, 1982.

[7] A. R. Subramanian, M. Kaufmann, and B. Morgenstern, "Dialign-tx: greedy and progressive approaches for segment-based multiple sequence alignment," *Algorithms for Molecular Biology*, vol. 3, no. 1, p. 6, 2008.

[8] M. Steel, "Phylogenetic diversity and the greedy algorithm," *Systematic Biology*, vol. 54, no. 4, pp. 527–529, 2005.

[9] G. S. Frandsen, T. Husfeldt, P. B. Miltersen, T. Rauhe, and S. Skyum, "Dynamic algorithms for the dyck languages," in *Workshop on Algorithms and Data Structures*. Springer, 1995, pp. 98–108.

[10] N. Kokash, "An introduction to heuristic algorithms," *Department of Informatics and Telecommunications*, pp. 1–8, 2005.

[11] D. J. Lipman, S. F. Altschul, and J. D. Kececioglu, "A tool for multiple sequence alignment," *Proceedings of the National Academy of Sciences*, vol. 86, no. 12, pp. 4412–4415, 1989.

[12] R. Bird and O. De Moor, "The algebra of programming," in *NATO ASI DPD*, 1996, pp. 167–203.

[13] M. L. Fredman, "The complexity of maintaining an array and computing its partial sums," *Journal of the ACM (JACM)*, vol. 29, no. 1, pp. 250–260, 1982.

[14] S.-M. Chen, C.-H. Lin, S.-J. Chen *et al.*, "Multiple dna sequence alignment based on genetic algorithms and divide-and-conquer techniques," *International Journal of Applied Science and Engineering*, vol. 3, no. 2, pp. 89–100, 2005.

[15] R. E. Korf and W. Zhang, "Divide-and-conquer frontier search applied to optimal sequence alignment," in *AAAI/IAAI*, 2000, pp. 910–916.

[16] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006.

[17] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1995, pp. 338–345.

[18] R. Kaundal, A. S. Kapoor, and G. P. Raghava, "Machine learning techniques in disease forecasting: a case study on rice blast prediction," *BMC bioinformatics*, vol. 7, no. 1, p. 485, 2006.

[19] R. M. Karp and M. O. Rabin, "Efficient randomized pattern-matching algorithms," *IBM journal of research and development*, vol. 31, no. 2, pp. 249–260, 1987.

[20] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped blast and psi-blast: a new generation of protein database search programs," *Nucleic acids research*, vol. 25, no. 17, pp. 3389–3402, 1997.

[21] S. Unseld, T. Frischmuth, and H. Jeske, "Short deletions in nuclear targeting sequences of african cassava mosaic virus coat protein prevent geminivirus twinned particle formation," *Virology*, vol. 318, no. 1, pp. 90–101, 2004.

[22] J. Brown *et al.*, "Current status of bemisia tabaci as a plant pest and virus vector in agroecosystems worldwide." *FAO Plant Protection Bulletin*, vol. 42, no. 1/2, pp. 3–32, 1994.

[23] J. K. Brown, "The molecular epidemiology of begomoviruses," *Trends in plant virology*, pp. 279–316, 2001.

[24] W. Zhang, N. H. Olson, T. S. Baker, L. Faulkner, M. Agbandje-McKenna, M. I. Boulton, J. W. Davies, and R. McKenna, "Structure of the maize streak virus geminate particle," *Virology*, vol. 279, no. 2, pp. 471–477, 2001.

[25] C. M. Fauquet, D. Bisaro, R. Briddon, J. Brown, B. Harrison, E. Rybicki, D. Stenger, and J. Stanley, "Virology division news: revision of taxonomic criteria for species demarcation in the family geminiviridae, and an updated list of begomovirus species," *Archives of virology*, vol. 148, no. 2, pp. 405–421, 2003.

[26] C. M. Fauquet and J. Stanley, "Geminivirus classification and nomenclature: progress and problems," *Annals of applied biology*, vol. 142, no. 2, pp. 165–189, 2003.

[27] T. Frischmuth, "Genome of dna viruses," in *Molecular biology of plant viruses*. Springer, 1999, pp. 29–46.

[28] 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.

[29] 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.

[30] 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.

[31] B. L. Patil, B. Bagewadi, J. S. Yadav, and C. M. Fauquet, "Mapping and 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.

[32] S. Mansoor, R. W. Briddon, Y. Zafar, and J. Stanley, "Geminivirus disease complexes: an emerging threat," *Trends in plant science*, vol. 8, no. 3, pp. 128–134, 2003.

[33] J. Navas-Castillo, E. Fiallo-Olivé, and S. Sánchez-Campos, "Emerging virus diseases transmitted by whiteflies," *Annual review of phytopathology*, vol. 49, pp. 219–248, 2011.

[34] J. Legg, B. Owor, P. Sseruwagi, and J. Ndunguru, "Cassava mosaic virus disease in east and central africa: epidemiology and management of a regional pandemic," *Advances in virus research*, vol. 67, pp. 355–418, 2006.

[35] G. Otim-Nape, A. Bua, J. Thresh, Y. Baguma, S. Ogwal, G. Ssemakula, G. Acola, B. Byabakama *et al.*, "The current pandemic of cassava mosaic virus disease in east africa and its control." *The current pandemic of cassava mosaic virus disease in East Africa and its control.*, 2000.

[36] 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.

[37] K. B. Cheng Ling, "Design and implementation of a cuda-compatible gpu-based core for gapped blast algorithm," *Procedia Computer Science*, vol. 1, pp. 495–504, 2012.

[38] M. Y. I. Korf and J. Bedell, *An Essential Guide to the Basic Local Alignment Search Tool*. O'Reilly & Associates, 2003.

[39] J. Pevsner, "Pairwise sequence alignment," *Bioinformatics and functional genomics, 2nd edition. Hoboken: John Wiley & Sons*, pp. 47–97, 2009.

[40] W. Haque, A. Aravind, and B. Reddy, "Pairwise sequence alignment algorithms: a survey," in *Proceedings of the 2009 conference on Information Science, Technology and Applications*. ACM, 2009, pp. 96–103.

[41] H. A. Schmidt, "Phylogenetic trees from large datasets," Ph.D. dissertation, 2003.

[42] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic acids research*, vol. 22, no. 22, pp. 4673–4680, 1994.

[43] B. Morgenstern, "Dialign 2: improvement of the segment-to-segment approach to multiple sequence alignment." *Bioinformatics (Oxford, England)*, vol. 15, no. 3, pp. 211–218, 1999.

[44] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller, "A greedy algorithm for aligning dna sequences," *Journal of Computational biology*, vol. 7, no. 1-2, pp. 203–214, 2000.

[45] A. Löytynoja and N. Goldman, "An algorithm for progressive multiple alignment of sequences with insertions," *Proceedings of the National Academy of Sciences*, vol. 102, no. 30, pp. 10 557–10 562, 2005.

[46] F. Corpet, "Multiple sequence alignment with hierarchical clustering," *Nucleic acids research*, vol. 16, no. 22, pp. 10 881–10 890, 1988.

[47] M. Vingron, J. Stoye, H. Luz, and S. Böcker, "Algorithms for phylogenetic reconstructions," *Lecture, Notes and Exercises*, 2003.

[48] J. Felsenstein, "Evolutionary trees from dna sequences: a maximum likelihood approach," *Journal of molecular evolution*, vol. 17, no. 6, pp. 368–376, 1981.

[49] N. Goldman, "Maximum likelihood inference of phylogenetic trees, with special reference to a poisson process model of dna substitution and to parsimony analyses," *Systematic Zoology*, vol. 39, no. 4, pp. 345–361, 1990.

[50] T. Pupko, I. Pe, R. Shamir, and D. Graur, "A fast algorithm for joint reconstruction of ancestral amino acid sequences," *Molecular biology and evolution*, vol. 17, no. 6, pp. 890–896, 2000.

[51] Paleobio.org. (2003, June) Phylogeny in a nutshell. [Online]. Available: http://www.paleobio.org/education/nutshell.html

[52] J. Felsenstein, "Phylogenies from molecular sequences: inference and reliability," *Annual review of genetics*, vol. 22, no. 1, pp. 521–565, 1988.

[53] F. Pardi, "Algorithms on phylogenetic trees," Ph.D. dissertation, University of Cambridge, 2009.

[54] A. Spillner, B. T. Nguyen, and V. Moulton, "Computing phylogenetic diversity for split systems," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 5, no. 2, pp. 235–244, 2008.

[55] L.-T. Nguyen, *An efficient algorithm for phylogeny reconstruction by maximum likelihood*. na, 2011.

[56] L. S. Vinh and A. von Haeseler, "Iqpnni: moving fast through tree space and stopping in time," *Molecular biology and evolution*, vol. 21, no. 8, pp. 1565–1571, 2004.

[57] J. Handelsman, "Metagenomics: application of genomics to uncultured microorganisms," *Microbiology and molecular biology reviews*, vol. 68, no. 4, pp. 669–685, 2004.

[58] S. G. Tringe and E. M. Rubin, "Metagenomics: Dna sequencing of environmental samples," *Nature reviews genetics*, vol. 6, no. 11, p. 805, 2005.

[59] A. Konopka, "What is microbial community ecology?" *The ISME journal*, vol. 3, no. 11, p. 1223, 2009.

[60] N. R. Council *et al.*, *The new science of metagenomics: revealing the secrets of our microbial planet*. National Academies Press, 2007.

[61] F. Valenzuela-González, M. Martínez-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.

[62] K. Hipp, P. Rau, B. Schäfer, 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.

[63] W. N. Leke, D. B. Mignouna, J. K. Brown, and A. Kvarnheden, "Begomovirus disease complex: emerging threat to vegetable production systems of west and central africa," *Agriculture & Food Security*, vol. 4, no. 1, p. 1, 2015.

[64] J. K. Brown, F. M. Zerbini, J. Navas-Castillo, E. Moriones, R. Ramos-Sobrinho, J. C. Silva, E. Fiallo-Olivé, R. W. Briddon, C. Hernández-Zepeda, A. Idris *et al.*, "Revision of begomovirus taxonomy based on pairwise sequence comparisons," *Archives of Virology*, vol. 160, no. 6, pp. 1593–1619, 2015.

[65] M. S. Nawaz-ul Rehman and C. M. Fauquet, "Evolution of geminiviruses and their satellites," *FEBS letters*, vol. 583, no. 12, pp. 1825–1832, 2009.

[66] G. Chibawe, L. Mzyece, M. Nyirenda, and J. Phiri, "The use of open source tools in research: A case study of the mutation of african cassava mosaic virus," in *Zambia Association of Public Universities and Colleges*, May 2018.

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

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

[69] H. C. Jubb, A. P. Higueruelo, B. Ochoa-Montaño, 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.

[70] 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.

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

[72] W. W. W. Consortium *et al.* (2006, September) Extensible markup language (xml) 1.1. [Online]. Available: http://travesia.mcu.es/portalnb/jspui/?locale=en

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

[74] S. S. B. Lal. (2000) Extensible markup language (xml). [Online]. Available: http://iasri.res.in/ebook/expertsystem/XML.pdf

[75] A. Hoekman, "Journal publishing technologies: Xml," *Journal of International Business Studies*, vol. 19, pp. 125–127, 2003.

[76] R. Kamadjeu and H. Tolentino, "Web-based public health geographic information systems for resources-constrained environment using scalable vector graphics technology: a proof of concept applied to the expanded program on immunization data," *International Journal of Health Geographics*, vol. 5, no. 1, p. 24, 2006.

[77] J. Ferraiolo, F. Jun, and D. Jackson. (2000) Scalable vector graphics (svg) 1.0 specification. [Online]. Available: https://www.w3.org/TR/2001/REC-SVG-20010904/REC-SVG-20010904.pdf

[78] C. Peng, "Scalable vector graphics (svg)," in *Research Seminar on Interactive Digital Media*, ser. 590, Tik-111.590, Ed., vol. 111, Department of Computer Science and Engineering Helsinki University of Technology pcy@tml.hut.fi, Fall 2000.

[79] A. Neumann, "Scalable vector graphics (svg)," in *Encyclopedia of GIS*. Springer, 2008, pp. 1013–1021.

[80] Z.-R. Peng and C. Zhang, "The roles of geography markup language (gml), scalable vector graphics (svg), and web feature service (wfs) specifications in the development of internet geographic information systems (gis)," *Journal of Geographical Systems*, vol. 6, no. 2, pp. 95–116, 2004.

[81] J. Bowler, C. Brown, M. Capsimalis, R. Cohn, L. D. Cole *et al.*, "Scalable vector graphics (svg) 1.0 specification," *World Wide Web Consortium*, 2001.

[82] H.-J. Chung, C. H. Park, M. R. Han, S. Lee, J. H. Ohn, J. Kim, J. Kim, and J. H. Kim, "Arrayxpath ii: mapping and visualizing microarray gene-expression data with biomedical ontologies and integrated biological pathway resources using scalable vector graphics," *Nucleic acids research*, vol. 33, no. suppl_2, pp. W621–W626, 2005.

[83] S. Fowler, L. Denuzière, and A. Granicz, "Reactive single-page applications with dynamic dataflow," in *International Symposium on Practical Aspects of Declarative Languages*. Springer, 2015, pp. 58–73.

[84] R. B. Johnson, A. J. Onwuegbuzie, and L. A. Turner, "Toward a definition of mixed methods research," *Journal of mixed methods research*, vol. 1, no. 2, pp. 112–133, 2007.

[85] C. Mulundano, "Digital financial services model for higher education institutions in zambia," Master's thesis, University of Zambia, P.O. Box 32379 Lusaka, Zambia, November 2017.

[86] J. W. Creswell and J. D. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2017.

[87] D. L. Driscoll, A. Appiah-Yeboah, P. Salib, and D. J. Rupert, "Merging qualitative and quantitative data in mixed methods research: How to and why not," 2007.

[88] C. R. Kothari, *Research methodology: Methods and techniques*, 1st ed. New Age International, 2004.

[89] S. B. Merriam, *Qualitative Research and Case Study Applications in Education. Revised and Expanded from" Case Study Research in Education."*., 1st ed. Jossey-Bass Publishers, 350 Sansome St, San Francisco, CA 94104, 1998.

[90] C. for Research Quality., "Overview of qualitative research methods [video file]." August 2015. [Online]. Available: https://www.youtube.com/watch?v=IsAUNs-IoSQ

[91] C. Teddlie and A. Tashakkori, "A general typology of research designs featuring mixed methods," *Research in the Schools*, vol. 13, no. 1, pp. 12–28, 2006.

[92] T. Hall, "Bioedit version 5.0. 6," *North Carolina State University, Department of Microbiology, Raleigh, North Carolina*, vol. 192, 2001.

[93] T. Hall, I. Biosciences, and C. Carlsbad, "Bioedit: an important software for molecular biology," *GERF Bull Biosci*, vol. 2, no. 1, pp. 60–61, 2011.

[94] R. A. Fouchier, V. Munster, A. Wallensten, T. M. Bestebroer, S. Herfst, D. Smith, G. F. Rimmelzwaan, B. Olsen, and A. D. Osterhaus, "Characterization of a novel influenza a virus hemagglutinin subtype (h16) obtained from black-headed gulls," *Journal of virology*, vol. 79, no. 5, pp. 2814–2822, 2005.

[95] H. Chen, G. Smith, K. Li, J. Wang, X. Fan, J. Rayner, D. Vijaykrishna, J. Zhang, L. Zhang, C. Guo *et al.*, "Establishment of multiple sublineages of h5n1 influenza virus in asia: implications for pandemic control," *Proceedings of the National Academy of Sciences*, vol. 103, no. 8, pp. 2845–2850, 2006.

[96] R. T. Anderson, H. A. Vrionis, I. Ortiz-Bernad, C. T. Resch, P. E. Long, R. Dayvault, K. Karp, S. Marutzky, D. R. Metzler, A. Peacock *et al.*, "Stimulating the in situ activity of geobacter species to remove uranium from the groundwater of a uranium-contaminated aquifer," *Applied and environmental microbiology*, vol. 69, no. 10, pp. 5884–5891, 2003.

[97] K. Vandepoele, J. Raes, L. De Veylder, P. Rouzé, S. Rombauts, and D. Inzé, "Genome-wide analysis of core cell cycle genes in arabidopsis," *The Plant Cell*, vol. 14, no. 4, pp. 903–916, 2002.

[98] X. Hong, Y. Zhang, Y. Chu, H. Gao, Z. Jiang, and S. Xiong, "Complete sequence determination and phylogenetic analysis of fkn among seven higher primates including homonids and old world monkeys," *Yi chuan= Hereditas*, vol. 30, no. 5, pp. 595–601, 2008.

[99] M. Kearse, R. Moir, A. Wilson, S. Stones-Havas, M. Cheung, S. Sturrock, S. Buxton, A. Cooper, S. Markowitz, C. Duran *et al.*, "Geneious basic: an integrated and extendable desktop software platform for the organization and analysis of sequence data," *Bioinformatics*, vol. 28, no. 12, pp. 1647–1649, 2012.

[100] M. Farhadi, A. Fazaeli, and A. Haniloo, "Genetic characterization of livestock and human hydatid cyst isolates from northwest iran, using the mitochondrial cox1 gene sequence," *Parasitology research*, vol. 114, no. 12, pp. 4363–4370, 2015.

[101] M. Mendoza, L. Güiza, X. Martinez, X. Caraballo, J. Rojas, L. F. Aranguren, and M. Salazar, "A novel agent (endozoicomonas elysicola) responsible for epitheliocystis in cobia rachycentrum canadum larvae," *Diseases of aquatic organisms*, vol. 106, no. 1, pp. 31–37, 2013.

[102] K. Tamura, G. Stecher, D. Peterson, A. Filipski, and S. Kumar, "Mega6: molecular evolutionary genetics analysis version 6.0," *Molecular biology and evolution*, vol. 30, no. 12, pp. 2725–2729, 2013.

[103] S. Kumar, G. Stecher, D. Peterson, and K. Tamura, "Mega-cc: computing core of molecular evolutionary genetics analysis program for automated and iterative data analysis," *Bioinformatics*, vol. 28, no. 20, pp. 2685–2686, 2012.

[104] S. Kumar, K. Tamura, and M. Nei, "Mega3: integrated software for molecular evolutionary genetics analysis and sequence alignment," *Briefings in bioinformatics*, vol. 5, no. 2, pp. 150–163, 2004.

[105] S. Kumar, M. Nei, J. Dudley, and K. Tamura, "Mega: a biologist-centric software for evolutionary analysis of dna and protein sequences," *Briefings in bioinformatics*, vol. 9, no. 4, pp. 299–306, 2008.

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

[107] N. R. Coordinators, A. Acland, R. Agarwala, T. Barrett, J. Beck, D. A. Benson, C. Bollin, E. Bolton, S. H. Bryant, K. Canese *et al.*, "Database resources of the national center for biotechnology information," *Nucleic acids research*, vol. 42, no. Database issue, p. D7, 2014.

[108] N. R. Coordinators, "Database resources of the national center for biotechnology information," *Nucleic acids research*, vol. 45, no. Database issue, p. D12, 2017.

[109] K. D. Pruitt, T. Tatusova, and D. R. Maglott, "Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins," *Nucleic acids research*, vol. 35, no. suppl_1, pp. D61–D65, 2006.

[110] G. Chibawe, L. Mzyece, M. Nyirenda, and J. Phiri, "Using open source tools for analysis of the mutation rate of african cassava mosaic virus," *Zambia ICT Journal*, vol. 2, no. 1, pp. 44–56, 2018.

[111] B. M. Muhire, A. Varsani, and D. P. Martin, "Sdt: a virus classification tool based on pairwise sequence alignment and identity calculation," *PloS one*, vol. 9, no. 9, p. e108277, 2014.

[112] L. A. Palinkas, S. M. Horwitz, C. A. Green, J. P. Wisdom, N. Duan, and K. Hoagwood, "Purposeful sampling for qualitative data collection and analysis in mixed method implementation research," *Administration and Policy in Mental Health and Mental Health Services Research*, vol. 42, no. 5, pp. 533–544, 2015.

[113] T. Alicai, J. Ndunguru, P. Sseruwagi, F. Tairo, G. Okao-Okuja, R. Nanvubya, 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.

[114] Z. Yang, *Phylogenetic analysis by maximum likelihood (PAML)*, 3rd ed., Ziheng Yang Department of Biology University College London Gower Street, London WC1E 6BT, 2000.

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

APPENDICES

A.1: CODE LISTINGS

Listing A.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()

```

```
20
21 #Define the blast function
22 def blast_align():
23 """
24
25 :return:
26 """
27 print("Blasting....")
28
29 #Define the function to retrieve the accession number from the GUI input
       variable
30 def retrieve_input():
31 """
32
33 :return:
34 """
35
36 accessionNumber = textBox.get("1.0", "end-1c")
37 return accessionNumber
38
39 #Retrieve the accession number from the GUI input variable
40 retrieve_input()
41 #create a data handle for the blast result
42 result_handle = NCBIWWW.qblast("blastn", "nr",
43 retrieve_input(),
44 word_size=7,
45 gapcosts='5 2',
46 nucl_reward=1,
47 nucl_penalty='-3',
48 expect=1000)
49 blast_records = NCBIXML.read(result_handle)
50
51 #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()
```

Listing A.2: Biopython phylogenetic ascii tree code

```
# coding=utf-8
from Bio import Phylo
from Bio.Phylo.PhyloXML import Phylogeny

handle = "Pathto/file.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)
```

A.2: FIGURES

Figure A-1: Screen shot of a PhyloXML output from one of the Biopython modules

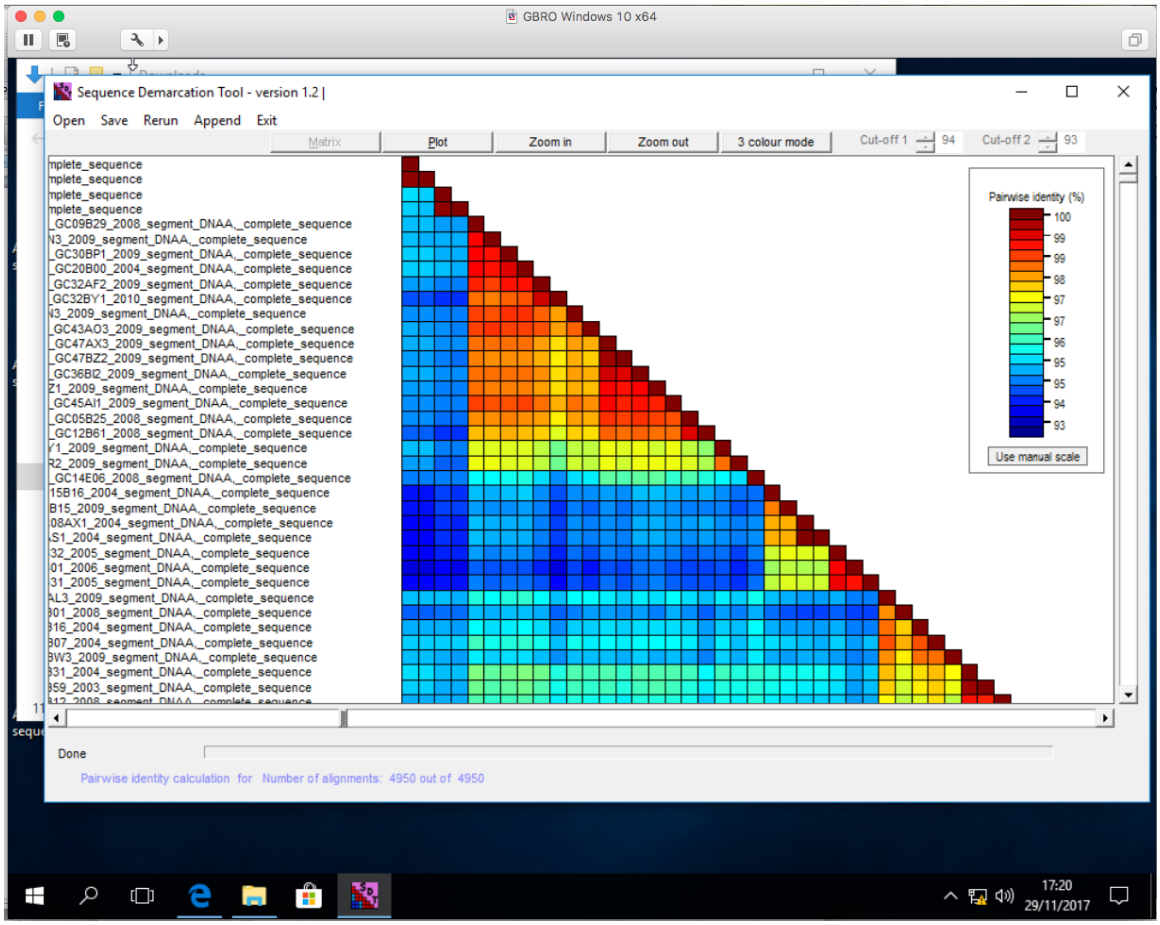


Figure A-2: Scale up of colour coded matrix output of multiple pairwise alignment by SDT

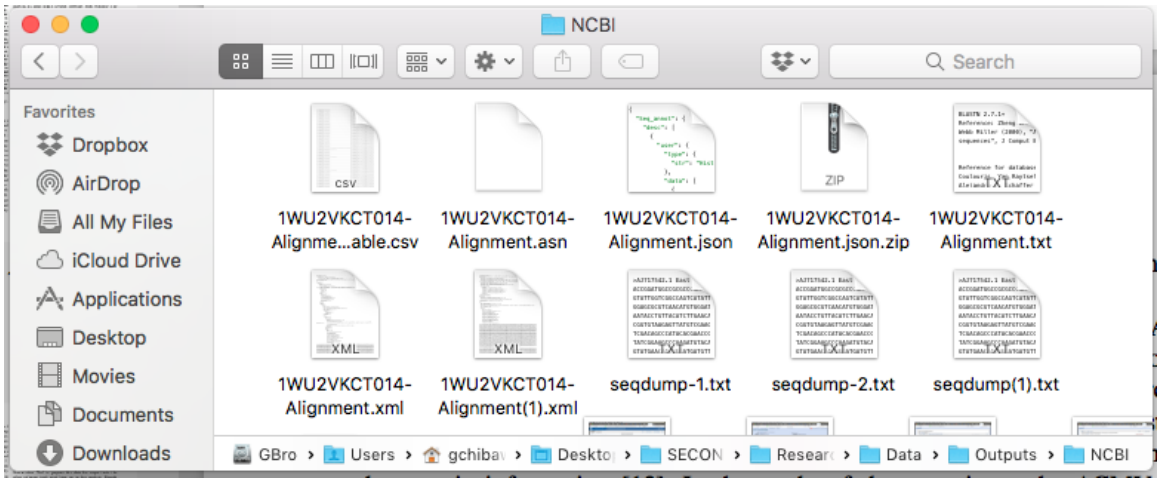


Figure A-3: Various formats outputs from NCBI stored on a local computer

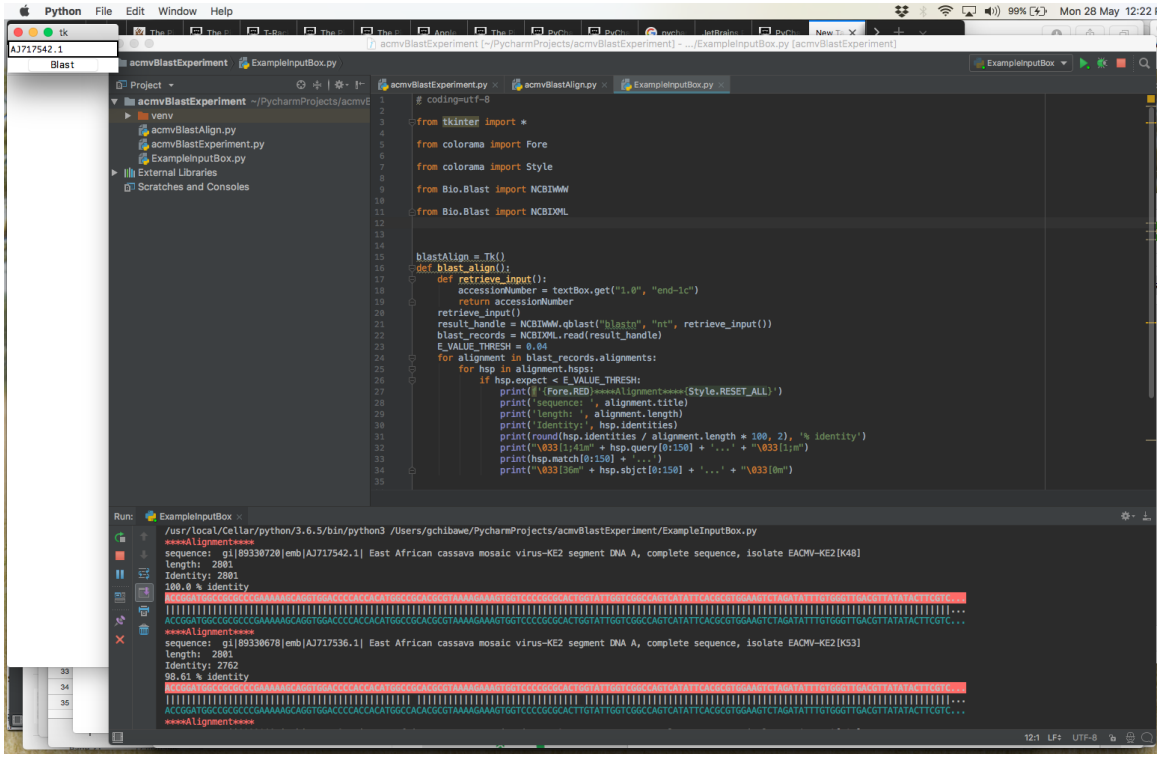


Figure A-4: Screen shot of BLAST and alignment output using PyCharm IDE

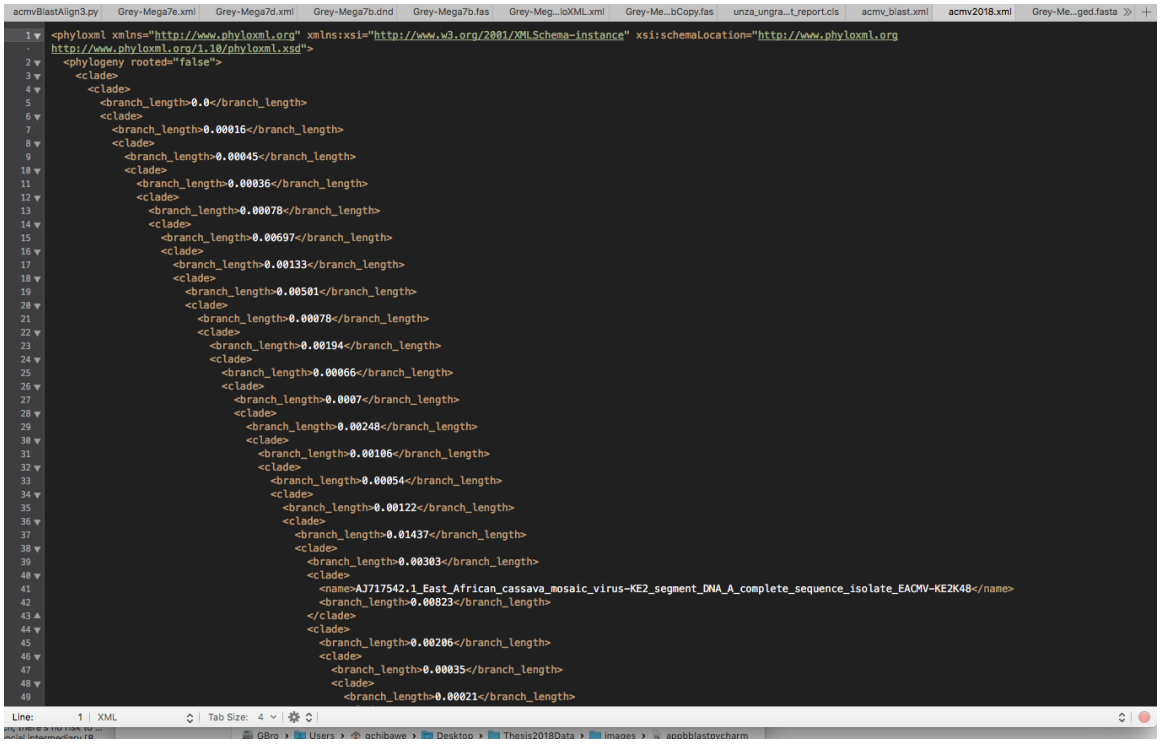


Figure A-5: Screen shot of top most portion of a PhyloXML tree output using Biopython libraries

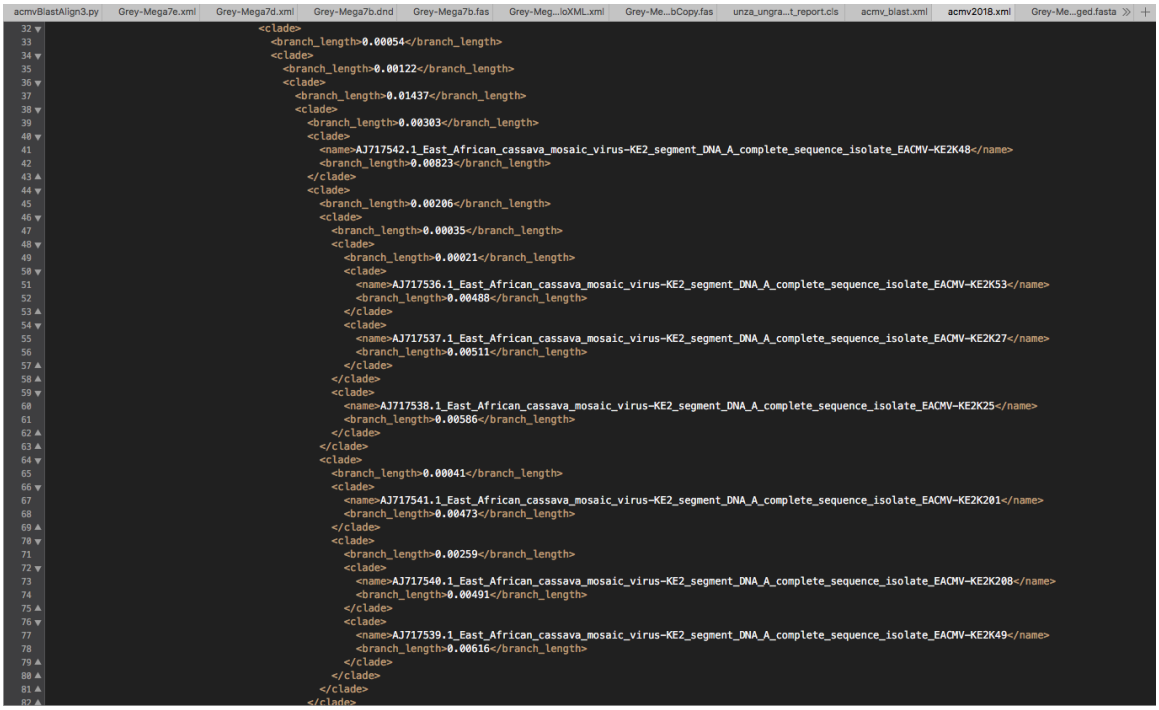


Figure A-6: Screen shot of PhyloXML tree output portion that shows the sequences