

**DEVELOPMENT OF AN INTERACTIVE ONLINE TOOL FOR
GENOME SEQUENCE-BASED INFLUENZA
VACCINE STRAIN SELECTION**

A Thesis presented to
the Faculty of the Graduate School
at the University of Missouri

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
SWETA PRAGYAN PRAHARAJ
Thesis Supervisor: Dr. XiuFeng Wan
May 2021

The undersigned, appointed by the Dean of the Graduate School, have examined the thesis entitled:

DEVELOPMENT OF AN INTERACTIVE ONLINE TOOL FOR GENOME SEQUENCE-BASED
INFLUENZA VACCINE STRAIN SELECTION

presented by Sweta Pragyan Praharaj, a candidate for the degree of Master of Science and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. XiuFeng Wan

Dr. Rohit Chadha

Dr. Wenjun Ma

ACKNOWLEDGMENTS

I would like to thank Dr. XiuFeng Wan for offering me this opportunity to pursue my Master's degree, guiding my research, and providing moral and emotional support during my Master's journey. I was given the wonderful opportunity by Dr. XiuFeng Wan, to work in the SystemsBio Lab and was provided the higher end lab facilities in the pursuit of my challenging research for which I am very much grateful. I would like to express my gratitude to Dr. Rohit Chadha and Dr. Wenjun Ma for their interest and consent to be part of my thesis committee. I would extend my gratitude to SystemsBio lab members, specially Cheng Gao and Cynthia Tang and Life Science IT Department specially Mattson Jeremy and Roettgen, Nathan O for helping me in setting up my project and progressing as well.

Finally, but most importantly, special thanks must go to my loving and supportive parents, Susil Kumar Praharaj and Sanju Praharaj, and my wonderful sister, Smita Praharaj, who provided endless support, understanding, and unending inspiration and always put a smile on my face. Thank you for all your love.

Sweta Pragyan Praharaj

Contents

ACKNOWLEDGMENTS	ii
LIST OF FIGURES	v
ABSTRACT	vii
1 Introduction	1
1.1 Influenza and Vaccine Strain Selection	1
1.2 Literature Review	2
1.3 Motivation	12
1.4 Problem Statement	13
1.5 Objectives	13
1.6 Thesis Organization	14
2 Web Server Development	15
2.1 Web Architecture	15
2.2 Web Visualizations and Interface	19
2.2.1 D3.js	19
2.2.2 Highcharts	22
2.2.3 Other Views and Features	24
2.3 Data Collection and Manipulation	27
2.4 Application Components	28
2.4.1 Public Input Model	28
2.4.2 User Input Model	29
2.4.3 Hybrid Input Model	31
2.4.4 Output Models	32
2.5 Implementation	35
2.5.1 Process	35

2.5.2	Flow Diagrams	36
2.5.3	Input Fasta Formats	37
2.5.4	Tools and Utilities	38
3	Application	41
3.1	Session Management	41
3.2	Job Management	42
3.3	Pre-defined Functions	42
3.4	File Uploading	44
3.5	MVC	44
3.5.1	Model	45
3.5.2	View	46
3.5.3	Controller	47
3.6	URL Helper	49
3.7	Code Repository	50
3.8	Summary	50
4	Conclusion and Future Work	51
4.1	Conclusion	51
4.2	Future Work	51

List of Figures

Figure	Page
2.1 Linux Server Details	16
2.2 Apache/HTTPD Version	16
2.3 Apache Configuration File (sysbio.conf) located at /etc/httpd/conf.d 17	
2.4 Review before Job Submission	25
2.5 Selecting viral antigens in the Antigenic Cartography	26
2.6 Viewing those nodes in the Phylogenetic Tree	27
2.7 Public Input Model	30
2.8 User Input Model	31
2.9 Hybrid Input Model	32
2.10 Phylogenetic Tree with Tooltip Information	34
2.11 2D Antigenic Map by Year	35
2.12 3D Antigenic Map by Year	35
2.13 Public Input Model Processing	36
2.14 User Input Model Processing	36
2.15 Hybrid Input Model Processing	37
2.16 Fasta Header Format having Strain name and Year	37
2.17 Fasta Header Format having Strain name, Year and Group1	37
2.18 Fasta Header Format having Strain name, Year, Group1 and Group 2	38
3.1 Unique Temp File Storage for User	43
3.2 File Upload Configuration	44

3.3	Model-View-Controller Software Design	45
3.4	Model Class	46
3.5	Codeigniter Connection to MYSQL	47
3.6	View Class	48
3.7	List of Controller Functions	49
3.8	Controller Class	49

ABSTRACT

Seasonal influenza viruses in humans infect approximately 5% to 15% of the population and cause an estimated half-million deaths worldwide per year. Among the four co-circulating seasonal influenza viruses, subtype H3N2 and H1N1 influenza A viruses have rapid mutations and frequent antigenic drift events, leading to frequent updates of vaccine strains in the seasonal influenza vaccine. Seasonal influenza vaccination is the primary option to prevent and control influenza epidemics, and the selection of an antigenic matched vaccine strain is one of the keys to the success of seasonal influenza vaccination. Thus, it is critical to have robust and rapid antigenic analyses of epidemic strains and estimates of their genetic and antigenic relationship with the vaccine strain in use. In this study, we present vaccineEvol, an interactive and user-friendly web visualization tool that allows researchers to comprehend large sequence datasets into antigenic and genetic analyses. With the integration of the genomic sequences from the public database, the tool enables the users to track and analyze both genetic and antigenic evolutionary dynamics of seasonal influenza viruses. Primarily, our application can quantify both genetic and antigenic distances among seasonal H3N2 influenza A viruses and display genetic and antigenic variants using phylogenetic tree and antigenic cartography, respectively. The users can also interactively analyze genetic and antigenic variants between the phylogenetic tree and antigenic cartography. The application performs machine learning based computations in the backend, which was previously developed in our lab, and efficient construction of trees and maps in the frontend. In summary, in this study, an interactive web server was developed for rapid antigenic and genetic analyses of seasonal influenza viruses and thus facilitate seasonal influenza vaccine strain selection.

Keywords: Antigenic Cartography, Phylogenetic Tree, Vaccine, H3N2 Influenza A Virus, vaccineEvol, web visualization, web application

Chapter 1

Introduction

The goal of this chapter is to discuss certain key biological concepts that provide an understanding of the background for developing a web interface for visualizing divergence in influenza virus strains. For any particular year, the selection of antigenically matching vaccine strain is critical for improving influenza vaccine efficiency. Hence developing tools that contribute to study influenza virus evolution is necessary. Effective visualization of large-scale molecular data is challenging. This thesis addresses three ways of selecting and entering virus strain sequences that run several back-end processes and produce antigenic cartography and phylogenetic trees as end outputs.

1.1 Influenza and Vaccine Strain Selection

Some of the factors that determine the potential burden of influenza disease include the characteristics of circulating viruses, the timing of the season, the protection level of the vaccine against illness, the number of people who receive vaccinations. Influenza season creates a huge burden on the health of people globally each year. There was an estimated death of approximately 1 million worldwide and about 100,000 in the United States. The H3N2 virus continues to circulate globally as a seasonal influenza A virus, which is associated with

severe illness and undergoes antigenic evolution. Influenza virological surveillance is essential for the early identification of novel antigenic variants of epidemiologic and clinical significance. This thesis aims to develop an application that can help researchers closely monitor the antigenic and genetic characteristics of circulating influenza viruses.

An influenza virus is a negative-stranded RNA virus that belongs to the *Orthomyxoviridae* family. There are four serotypes, A, B, C, and D. Influenza A and B viruses cause seasonal epidemics. Influenza C virus relatively affects humans mildly. The zoonosis of the influenza D virus in humans is still unclear.

The influenza A viruses have eight genomic segments (segment 1-8) with varying lengths from about 890 to 2,341 nucleotides. The surface proteins hemagglutinin (HA) is involved in virus attachment to the host cells and in cell fusion, and NA releases freshly generated virions from the host cells. Both HA and NA are the primary targets for host immune systems [11]. The subtypes of influenza A viruses are based on HA and NA. To date, 18 HA and 11 NA subtypes have been reported in influenza A viruses. Influenza B viruses have eight segments, whereas influenza C and D viruses have seven. The peak influenza season in the northern hemisphere is from October to April next year. Influenza causes more than 100,000 hospitalizations and 30,000 deaths in the United States each year [1].

1.2 Literature Review

The antigenic map is an analog of geographic cartography in comparing antigens to two or three-dimensional maps. An antigenic map effectively detects antigenic drift events. Thus it is helpful in antigenic characterization and vaccine strain selection. Influenza antigenic map is becoming a standard tool in influenza surveillance of the World Health Organization (WHO) influenza reference laboratories [10].

An extensively used resource to study and visualize the evolution of influenza

viruses is a phylogenetic tree or evolutionary tree. These diagrams more often show inferred evolutionary relationships among biological entities based on similarities and differences in their genetic characteristics. Visualizing evolutionary trees using JavaScript packages and libraries allows an extension of some essential features and construct the trees as required by the users.

There are very few web applications that integrate both antigenic and genetic characterizations and enhance the decision of vaccine strain selection. Mostly the process of antigenic analyses is isolated from that of phylogenetic analyses and used as standalone applications. Currently, a related tool that addresses both sequence and serology data is ATIVS which is a web server for analyzing serological data of all influenza viruses and hemagglutinin sequence data of human influenza H3N2 viruses to generate antigenic maps for influenza surveillance and vaccine strain selection [16]. Another closely related tool is PREDAC-H3 [18], which focuses mostly on antigenic inference and clustering algorithms and web servers. Nextstrain dashboard displays phylogeny and geographical distribution of sequences [17]. Some other phylogeny based web applications include EvolView [14] and Phylogeny.IO [15]. Though they solve the challenges in rendering visualizations, these web applications do not integrate a phylogeny with an antigenic map. Our web application addresses and solves this issue by presenting an antigenic map with a phylogenetic tree side by side.

Advances in Web Visualizations

Data visualization gives us a clear view and idea of what the data means by visual context through maps, graphs, and charts. Visualizations can make the information more intuitive for the users to comprehend and make it easier to identify trends and patterns among testing objectives. Effective data visualization is the vital final step of data analysis. Delivering extensive structured data to the end-user simply and efficiently is challenging. Complex and voluminous data requires high-end visuals for precision. In this thesis, we utilized a web-

based visualization library that features many APIs (Application Programming Interfaces) for generating dynamic and intuitive visualization content on the web called D3.js, a JavaScript library managing documents based on data. D3.js helps experience data using HTML, SVG, and CSS. D3's emphasis on web standards gives full capabilities of modern browsers without tying itself to a restrictive framework, combining powerful visualization components and a data-driven approach to DOM (Document Object Model) manipulation. Another way of overcoming static and non-interactive traditional cartographies is to use a javascript-based web visualization library called Highcharts. It provides several helpful series of charts, one of which is a scatter plot to view coordinates in two-dimensional or three-dimensional plane. This thesis describes the construction of phylogenetic trees and two-dimensional/three-dimensional antigenic maps from protein sequence data.

A phylogenetic tree is a depiction of evolutionary relationships among organisms constructed using links/branches and nodes. Closer nodes are more related than farther nodes. We can draw these trees in various equivalent styles—rectangular, circular, rooted, or unrooted trees.

Antigenic cartography or map is primarily used to analyze the virus serological data for the influenza vaccine strain selection process. Antigenic maps differ from phylogenetic trees in that they are based on serology data that reflect the antigenic properties of testing virus strains. In our application, sequence-based antigenic cartography for H3N2 influenza viruses was developed using a machine learning-based antigenic distance prediction [19]. Integrating both the tree and maps can reveal temporal trends in the antigenic space leading to an enhanced comprehension of genetic and antigenic evolution.

Advances in Web Architecture

Our application follows the two-tier client-server architecture. The two-tier architecture primarily has two parts, a client tier and a server tier. The client tier (user's browser) sends a request to the server tier (sysbio server), and the server

tier responds with the desired information (the sysbio vaccineEvol web page). Our server combines both the application and the database. For any web application to be functional over the internet, it requires a web server. Apache is the most widely used Web server on Linux systems. Web servers serve Web pages requested by client computers. Clients send a request, and the server returns the view as Web pages using Web browser applications such as Firefox, Google Chrome, or Internet Explorer. Users enter a Uniform Resource Locator (URL) or an address that points to a Web server running its Fully Qualified Domain Name (FQDN) and a path to the needed resource or software. In our case (<http://sysbio2.missouri.edu/old-site/software/VaccineEvol>). In simpler terms, a web server is a piece of software that can handle client HTTP requests for Web resources related to its configured/served websites. It sets an active connection between a server and the browsers of website visitors while delivering files between them. The Model-View-Controller (MVC) web architectural style is used to design our web application.

The MVC framework is an architectural style that divides an application into three primary logical components - Model, View, and Controller. Each architectural constituent controls specific development phases. MVC segregates the business logic and presentation layer from each other. It was traditionally designed to manage desktop Graphical User Interfaces (GUIs). Nowadays, MVC architecture has become widely used for developing web applications as well as mobile apps.

Our application is designed and structured to decouple the core from infrastructure (like the framework, the database, and local computational software). We adapted a design pattern that helped us to establish a clean separation between core and infrastructure code.

Multiple Sequence Alignment (MSA)

The first step in studying and forming relationships between strains is to per-

form alignment between its protein sequences. This first step is to align multiple similar sequences to obtain maximum matching of the sequences. Firstly, related sequences are identified. As the method generates multiple similar sequence pairs, it needs to be converted into a single alignment, which arranges pairwise sequence alignments so that evolutionarily equivalent locations in all sequences are matched. Multiple sequence alignment or, in short, MSA is a methodology used to study similar sequences to determine the evolutionary associations among them. It reflects shared patterns among functionally or structurally related genes. It is the alignment of three or more sequences (protein or nucleic acid). The sequence alignment then unveils conserved positions from the precursor sequence. Hence from the output, homology can be inferred, and the evolutionary relationships between the sequences studied.

MSA reveals more biological information than pairwise alignments. For example, it shows the conserved sequence patterns among testing sequences, which are subtly detected by pairwise sequence alignment. Protein multiple alignments allow us to identify conserved and functionally critical amino acid residues.

Our application utilizes MUSCLE to perform protein sequence alignment. MUSCLE stands for Multiple Sequence Comparison by Log-Expectation. MUSCLE is claimed to achieve both better average accuracy, and better speed than its counterparts ClustalW2 or T-Coffee [13]. Given a sequence file in Fasta format, MUSCLE performs MSA and generates an aligned Fasta file.

Phylogenetic Analyses

Phylogenetic analysis is an illustration of the evolutionary pathways of all protein sequences. A phylogenetic tree (also phylogeny or evolutionary tree) is a branching diagram showing the evolutionary ties among various biological species. The outer leaf nodes in the tree represent a biological entity. The branches signify a relationship between two neighboring entities and can use the length of a branch to indicate the evolutionary gap between species. There are various ways of in-

ferring a phylogenetic tree from an aligned sequence - Distance-based methods include UPGMA (unweighted pair group method with arithmetic mean) and NJ (Neighbor-Joining) methods, Character-based methods include Maximum Likelihood and Maximum Parsimony.

VaccineEvol uses the distance matrix-based NJ approach to build phylogenetic trees. We decided to choose the NJ approach as it is fast and suitable for large sequence datasets. In the NJ tree, pairs of sequences with the least number of substitutions are joined as neighbors. NJ is an iterative clustering analysis based on the minimum-evolution criterion; the topology with the least total branch length is preferred at each step. NJ algorithm is computationally fast. Hence we can use it for massive data sets. It does not assume all lineages evolve at the same pace, i.e., molecular clock assumption. If the distance matrix is ultrametric or additive, the NJ guarantees a correct tree estimation, but the NJ algorithm does not require the matrix to be ultrametric and additive. We used NINJA [20] to generate a genetic distance matrix and then the NJ approach to output a Newick tree.

H3N2 and H1N1 are the two most evolving influenza A subtypes causing seasonal influenza outbreaks. Each year, the trivalent or quadrivalent vaccine used worldwide contains influenza A strains from H1N1 and H3N2, along with one or two influenza B strains.

Influenza viruses continually change genetically and antigenically over time. This change sometimes can be substantial and sudden or gradual and slight. When the change is gradual, point mutations occur in the genetic material of the virus. Over time, this series of small mutations accumulate and modify the HA and/or NA structure. These point mutations frequently occur because the virus's replication machinery does not have a proofreading mechanism. As we already know, the antigenicity of the virus strain is more predominant at certain sites; when this point mutations occur and cause these antigenic mutations at those sites, the binding properties of this HA or NA antigen change. These produce differ-

ent strains of a virus that are antigenically different from the parent virus. The vaccines ultimately do not antigenically match the new variant. Hence, the more recent strain can have the capability of escaping the host immune responses and causing infection or illness. This event is referred to as antigenic drift. Antigenic drift is one of the major reasons causing vaccine mismatch. Effective vaccination is then possible only if the vaccine strain is antigenically similar to the current circulating strains. Hence vaccines produced need to be periodically updated to protect the population from emerging influenza antigenic variants [1].

Antigenic Drift and Shift

Prediction of antigenic relations (similar or variant) among influenza viruses helps experts design or update the existing influenza vaccine's composition. It also provides critical insights into the evolutionary mechanisms and helps to understand the epidemiology of these pathogens.

Antigens are molecules on the exterior part of a virus's structure identified by the immune system to induce an immune response by antibody production. When an influenza virus infects a person through infection or vaccination, their immune system makes distinct antibodies against the antigens on that specific infecting virus. As described by CDC, *The term "antigenic characteristics" of a virus strain describes the antibody or immune response stimulated by the antigens on a particular virus. "Antigenic characterization refers to the analysis of a virus's antigenic characteristics to help examine how related it is to another virus."*

"CDC antigenically characterizes about 2,000 influenza viruses every year to compare how similar currently circulating influenza viruses are to those included in the influenza vaccine and monitor for changes in circulating influenza viruses". Antigenic characterization can indicate the influenza vaccine's ability to produce an immune response against the influenza viruses spreading among people". This information plays a vital role in deciding the viruses to be included in the upcoming season's vaccination program.

Replication errors or random mutations in the genes cause this antigenic drift. These minor variations from antigenic drift generate viruses closely linked to one another, demonstrated by their position close together on a phylogenetic tree. Influenza viruses that are nearly related to each other typically have similar antigenic characteristics. This also signifies that antibodies our immune system produces against one influenza virus will likely recognize and respond to antigenically similar influenza viruses (called "cross-protection").

However, the small transformations or substitutions associated with antigenic drift can gather over time and result in antigenically different viruses (far away on a phylogenetic tree). It is also likely that a single or small substitution in a specific significant section on the HA can result in antigenic drift. When this occurs, the body's immune system may not identify and prevent sickness caused by the recent influenza viruses. Hence, a person becomes sensitive to influenza again, as antigenic drift has changed the antigen's structure such adequately that a person's subsisting antibodies would not neutralize the newer influenza viruses.

Antigenic drift hence is the main reason people can get influenza more than once, and it is also a significant reason why the influenza vaccine composition must be reviewed and renewed each year to follow up with evolving influenza viruses.

Antigenic shift is the abrupt and substantial transformation in an influenza virus that produces a new virus subtype. The new influenza A subtype is so different from previous subtypes that most people do not have immunity to fight it. This poses a more considerable danger to the health of the masses as it can lead to a worldwide pandemic if the virus is transmitted uncontrollably from person to person. While the antigenic shift is known to cause pandemics, antigenic drift is known to cause epidemics between these pandemics.

Serological Data based Influenza Antigenic Analyses

We followed a similar approach of constructing antigenic analyses as developed for HI datasets. The computational framework developed using serological data

suggested constructing an influenza antigenic cartography manifesting its utility in antigenic characterization. This framework has two combined processes: (1) Matrix completion algorithm that constructs influenza antigenic distance matrices; (2) Multidimensional Scaling or MDS (temporal model) in which influenza antigens are projected to two-dimensional or three-dimensional cartography. This paper discusses the difficulty through a biologically motivated temporal evolution model which is then applied to the MDS algorithm [12].

Sequence Data based Influenza Antigenic Analyses

The antigenic distance prediction was made using the H3N2 machine learning approach developed in the SystemsBio lab. This method applied an Antigen-Bridges algorithm, an antigenic distance prediction using sequence data by machine learning. This technique used the ridge regression to lessen the difference between the genetic distance matrix and the antigenic distance matrix. This method allotted weights to the residues and selected residues that are most likely to cause antigenicity (larger weight implies larger influence on antigenicity) after training the model. By combining these weights and considering the biophysical attributes of the selected antigenic sites, a sequence-based antigenicity scoring function was formulated to calculate the antigenic distance between any two HA1 sequences. [19], and [12].

$$y = w_0 + \sum_{i=0}^p w_i x_1$$

y = Antigenic distance between two strains.

w_0 = Global weight calculated by taking the average of the weight matrix.

w_i = Local weight for a specific year.

x_1 = Distance between two strains generated by PIMA function.

This method will be useful in influenza vaccine strain selection by significantly decreasing the human efforts for serological characterization and will improve the probability of suitable influenza vaccine candidate selection.

Integrating Genetic and Antigenic Analyses

As we know, viruses keep changing over time. When we sequence the genomes of viruses, we can see all the genes that make up the virus. By doing this, we can study genetic changes in viruses. Phylogeny helps us compare genetic sequences to determine how closely related or similar the virus strains or vaccine strains are to each other. We can also monitor how the viruses are evolving and what viruses are circulating in animals that might evolve to infect humans. The phylogenetic tree helps us visualize these differences. It is like a family tree for people. Viruses that are closer together on the tree are more similar. The more genetically different, the further they are from each other. This can also help in choosing what the next vaccine should be. Antigenic cartography is creating maps of antigenically variable pathogens, which shows the antigenic properties by plotting each virus/vaccine strain in a two-dimensional or three-dimensional coordinate plane. Knowing the changes in binding properties of the HA helps us understand antigenic evolution.

As opposed to genetic analyses, Antigenic cartographies are based on data that indicate the antigenic characteristics of a virus strain. Although there exists a close connection between genetic and antigenic evolution for the human influenza A (H3N2) virus, genetic distance can't be an accurate indicator of antigenic distance. For example, a mutation in an amino acid might result in a huge disproportionate difference in the binding traits of a virus strain. Antigenic maps can exhibit large alterations in the antigenic plane due to minimum amino acid substitutions. Antigenic cartography offers the probability of an enhanced interpretation of genetic and antigenic evolution. Studying and analyzing both the antigenic and genetic changes helps in making a better selection of vaccines.

As discussed earlier, the antigenic drift and antigenic shift create changes in the virus to escape our immune system, and experts require a tool to find close antigenic and genetic matches between the infecting virus and the vaccine strains to renew the vaccine composition. Phylogenetic trees and antigenic cartography can help experts figure out how different new viruses are from what they have seen

earlier and how they are evolving, which would also help them determine what components to focus on when developing a vaccine.

1.3 Motivation

Up-to-date monitoring of the antigenic dynamics of the influenza virus is crucial for the accurate selection of vaccine strains, which is important for the effective prevention of influenza spread and infection. Modest visualizations of massive and complex data require dynamic interaction between user and viewer programs. So, the application must provide an immediate response to user control. A few web-based applications and tools implement these features for a user to interact with the visualizations representing virus strains' antigenic and genetic characteristics. It is straightforward and convenient to collect influenza virus sequence data to estimate antigenic relationships between strains. The existence of easily accessible and free databases containing sequence data for hundreds of thousands of influenza strains makes sequence-based antigenic estimates an attractive proposition to researchers as an antigenic mismatch between circulating strain and vaccine strain decreases the vaccine effectiveness. Besides, antigenic relatedness within the vaccine strain and the infecting strains can influence the cross-reactivity of the antibody response. Consequently, understanding the antigenic relationships between influenza viruses circulating is vital to both virologists and immunologists. This drove us to develop an application that can address both the purposes of displaying genetic and antigenic divergence of vaccine strains. Our application is among very few that directly uses sequence-based data, integrates several computational resources in the back-end, and simplifies large sequence datasets to generate antigenic maps and phylogenetic trees.

1.4 Problem Statement

While the impact of influenza varies, it creates a considerable hardship on people worldwide each year. Yearly vaccination is the primary and most effective way to prevent influenza and influenza-associated complications, especially for high-risk groups. Vaccine mismatches can cause vaccine failure, epidemic outbreaks, and substantial economic difficulties bringing havoc and mayhem in the world; thus, developing additional agile and robust processes to distinguish antigenic variants of the influenza virus stays a major public health endeavor. Experimental methods to ascertain influenza virus antigenic properties are time-consuming and mid-throughput and require live viruses. Our application integrated and used an experimentally validated machine-learning approach developed in our lab to determine influenza virus antigenicity based on the hemagglutinin sequence. We used MUSCLE to align sequences and Ninja for calculating the genetic distance matrix. We then used the NJ approach to construct a phylogenetic tree.

1.5 Objectives

Visualizing the divergence and antigenic relationship from high-dimensional immunological data is an open challenge for advanced visualizations. We present vaccineEvol, an interactive and user-friendly web visualization tool that allows researchers to comprehend large sequence datasets into understandable low-dimensional data. The tool tracks and analyzes the evolution and antigenicity of influenza viruses for the primary purpose of vaccine strain selection and to improve the strain selection process through a better understanding of virus evolution and antigenicity. Primarily, our application predicts and displays antigenic variants of influenza H3N2 viruses infecting humans using a two or three-dimensional cartography. Enhancing the analysis of vaccine seed selection, the application shows evolutionary trees alongside antigenic cartography. The application performs high-throughput computations in the back-end and efficient construction of trees and

cartography in the front-end. The computational methods involved in deducing antigenic variants fully rely on sequence data. Hence, this thesis describes integrating low-cost, powerful computing approaches and visualization capabilities to develop a web server to facilitate fast selections of vaccine strains.

1.6 Thesis Organization

The thesis is divided into three parts. The first part discusses the overall architecture, front-end and back-end components, and interactivity between the output visualizations. The second part consists of the framework features and bioinformatics tools we used to build our application. Lastly, we concluded and explained how our thesis could be extended for enhanced performance.

Chapter 2

Web Server Development

The web application can be divided into two main components: the front-end and the back-end. This chapter aims to discuss certain fundamental concepts that provide an understanding of web application's front-end and back-end components. The application's user interface is designed using HTML5, CSS3, and Bootstrap v4. For building the phylogenetic tree, we used D3.js. The back-end end is built using a PHP framework called CodeIgniter, which is then connected to the MySQL database. The database holds the virus sequence data and job Ids for users. CodeIgniter is an Application Development Framework that helped me create this web application using PHP. It helped me build the application efficiently using its rich collection of libraries, helpers, and in-built classes. CodeIgniter allowed me to creatively focus on my project design and plan by reducing the amount of code required for a given task. We wrote several MATLAB scripts to manipulate the data.

2.1 Web Architecture

Server

The server that hosts our web application is a part of University of Missouri Linux Clusters - sysbio2.missouri.edu. It houses a CentOS based Linux 7 platform. Fig-

Figure 2.1 shows the operating system details for the server:

```
Static hostname: sysbio2.missouri.edu
Icon name: computer-server
Chassis: server
Machine ID: e2c88174daa34900aaa5cbf3e9f8874a
Boot ID: 780ca7232e4845cfb04b064a40278d89
Operating System: CentOS Linux 7 (Core)
CPE OS Name: cpe:/o:centos:centos:7
Kernel: Linux 3.10.0-1160.21.1.el7.x86_64
Architecture: x86_64
```

Figure 2.1: Linux Server Details

Web Server

A web server is a piece of software that uses HTTP (Hypertext Transfer Protocol) and other popular protocols to reply to client requests made over the World Wide Web. The principal role of a web server is to return website content through storing, processing, and delivering webpages to users. A Web server also supports SMTP (Simple Mail Transfer Protocol) and FTP (File Transfer Protocol), used for email, file transfer, and storage. Apache HTTP web server is a widely used open-source web server that delivers web pages to users through the internet. After development, as described in its official page, "Apache quickly became the most popular HTTP client on the web. Apache is a cross-platform software; therefore, it works on both Unix and Windows servers. We used Apache version 2.4.6 on our server.

```
Server version: Apache/2.4.6 (CentOS)
Server built:   Nov 16 2020 16:18:20
```

Figure 2.2: Apache/HTTPD Version

Our web application's apache configuration file is located at `/etc/httpd/conf.d`. We created a file `sysbio.conf` to overwrite the default configuration settings. In the root directory and mod rewriting settings, certificates were configured as shown

in the figure 2.3.

```
###
<virtualHost *:80>
    ServerName sysbio2.missouri.edu
    ServerAlias sysbio.missouri.edu
    DocumentRoot /var/www/html
    #Redirect permanent / https://sysbio2.missouri.edu/
</VirtualHost>
###

<VirtualHost _default_:443>
    ServerName sysbio2.missouri.edu
    ServerAlias sysbio.missouri.edu
    DocumentRoot /var/www/html
    #SSLEngine on
    #SSLCertificateFile /etc/pki/tls/certs/sysbio_missouri_edu_cert-x509CO.cer
    #SSLCertificateKeyFile /etc/pki/tls/private/sysbio.missouri.edu-SAN.key
    #SSLCertificateChainFile /etc/pki/tls/certs/sysbio_missouri_edu_interm-x509I
OR.cer
    #SSLCertificateFile /etc/pki/tls/certs/sysbio2_missouri_edu_cert.cer
    #SSLCertificateKeyFile /etc/pki/tls/private/sysbio2.missouri.edu.key
    #SSLCertificateChainFile /etc/pki/tls/certs/sysbio2_missouri_edu_interm.cer
    <Directory "/var/www/html">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
~
```

Figure 2.3: Apache Configuration File (sysbio.conf) located at /etc/httpd/conf.d

Frameworks

- *CodeIgniter (PHP Framework)*

The VaccineEvol application is developed using CodeIgniter Framework as the back-end. CodeIgniter is a PHP Model-View-Controller-based back-end or server-side framework used for developing web applications rapidly. It provides advanced libraries for connecting to the database and performing multiple operations like sending emails, uploading files, managing sessions, etc. URLs formed by CodeIgniter are search-engine friendly and clean. The URL follows a segment-based approach rather than a standard query-based approach. The built-in features of this framework have been created to allow developers to work independently without relying too much on other components. The framework uses the Model-View-Controller architectural

pattern. When a user requests a resource or enters an address in this framework, the controller returns first. The controller then loads the views and models.

CodeIgniter is developed in a way that has kept it as adaptable and usable as possible, allowing a developer to work in its path. The framework's core components are so flexible that they can be partly extended or entirely replaced as per the developer's choice and convenience. Thus, it is a malleable framework that attempts to provide the required tools while staying out of the way. Several essential features append to its sense of easy use, lightweight, and modularity. We used CodeIgniter version 3.1.11 on our server. Listed below are some beneficial features and library classes that proved to be most beneficial in building the application.

1. Model-View-Controller Based System
2. Extremely Light Weight
3. Full Featured database classes with support for several platforms.
4. Query Builder Database Support
5. Form and Data Validation
6. Security for Cross-site scripting Filtering and Cross-site Request Forgery
7. Session Management
8. Email Sending Class
9. File Uploading Class

The framework is thoroughly documented, has online tutorials and resources and a reliable community forum to address issues and bugs. CodeIgniter is easy to learn for anyone already accustomed to PHP. To sum up, CodeIgniter's excellent capabilities allowed us to create our application in a managed and modular manner.

- *Bootstrap (CSS Framework)*

The front-end component that comprises of User Interface is the View of the application. The User Interface is mostly designed using Bootstrap's HTML elements and CSS classes. Bootstrap is a CSS framework that allows us to design and customize responsive web applications quickly. It is one of the most popular front-end open-source toolkits, featuring Sass variables and mixins, a responsive grid system, extensive prebuilt components, and powerful JavaScript plugins. We used Bootstrap version 4.1.1 to design web pages.

- *MySQL (Database)*

We decided to use a relational, open-source database for our application. This is because CodeIgniter has good support for relational database management systems like MySQL. We installed MySQL to store all the application-based data and Virus strain data downloaded from public databases. We used the already existing Sysbio database and created a new table. The SQL table schema was created using the query:

```
CREATE TABLE SEQUENCES(id INT NOT NULL, name TEXT NOT NULL, year INT NOT NULL, location varchar(30) NOT NULL, sequence TEXT NOT NULL, PRIMARY KEY (id));
```

2.2 Web Visualizations and Interface

2.2.1 D3.js

D3.js is a JavaScript library for handling Document Object Model based on data. D3 brings data to life using HTML, SVG, and CSS. D3's emphasis on web standards delivers the full capacities of modern web browsers without binding to a restrictive framework, merging powerful visualization elements and a data-driven approach to DOM manipulation. All the interactive features and events are writ-

ten using jquery functionalities. I used the Underscore js package to minimize writing codes for grouping the data as and when required. D3 is not among those solid frameworks that try to accommodate every credible feature. Alternatively, D3 resolves the crux of the problem - efficient manipulation of DOM based on data. This shuns propriety representation and yields extraordinary extensibility, exposing the full capabilities of web standards such as HTML, SVG, and CSS. This library helps a developer to code extremely fast with minimal overhead, supporting large datasets and dynamic behaviors for intercommunication of DOMs and animation. D3's functional technique supports code reuse through a diverse array of official and community-developed modules.

Constructing Trees

Using the D3.js JavaScript library, our tree is implemented as a flexible and lightweight tool allowing for the display of interactive and complex phylogenetic trees in a web-based environment without security-based limitations and the need for external plugins. The implementation is fast and responsive to user interaction, with the tree elements' display parameters being easily changed through user-friendly access controls.

As described earlier, a phylogenetic tree is a design that represents evolutionary relationships among organisms. The branching pattern in a phylogenetic tree exhibits how species or other groups evolved from a series of common ancestors. We can draw phylogenetic trees in various equivalent styles. Rotating a tree about its branch points doesn't change the information it carries.

Using web-based languages coupled with the SVG graphical format allows our program to display thousands of taxa trees while remaining responsive. The Phylogenetic tree visualization built using D3 offers many important features concerning tree nodes and branches. These trees can reflect basic information strain names, branch length, and meta-data like year and location. We can display labels next to the respective nodes with detailed information. The information display is flexible

according to the users' preferences (e.g., strain names with different meta-data or branch lengths).

Rendering the tree starts with the function `init()` which we called in the body tag of your HTML page using the onload event handler : `<body onload="init(parameters);">`. The onload function takes three parameters:

- String of the Newick tree used to build the phylogram.
- String identifier for the div id into which to generate the phylogram, note that it must contain the '#' in it. For example, if we want to render our tree in a div with the id 'phylogram' you'd pass 'phylogram' to the function and you'd need the following in your HTML: `<div id='phylogram'></div>`.
- An optional object with tree parameters should be passed as a javascript object with the following optional keys' mapping-dat': (obj) list of objects where the object keys are metadata and the object values are the metadata values. This is used to color the tree nodes.
 1. `treeType` : (str) either rectangular or radial [default: rectangular]
 2. `hideRuler` : (bool) if True, the background ruler will be hidden [default: show ruler]
 3. `skipBranchLengthScaling` : (bool) if True, a cladogram will be displayed instead of a phylogram [default: phylogram]
 4. `skipLabels` : (bool) if True, all node labels will be hidden [default: show labels]

The mapping-dat is the additional metadata. The dependencies used in the building of phylogenetic trees are

- Twitter Bootstrap [9]
- jQuery [5]
- Colorbrewer.js [2]

- d3.js [3]
- D3-tip [8]
- noUiSlider [7]

Coloring Nodes by Group

The phylogenetic tree nodes were colored according to groups. The grouping of the strains was according to the meta-data in the Fasta headers of the virus sequence strains. When the phylogenetic tree loads for the first time, all the tree's leaf nodes are colored blue and inner nodes grey. Users can optionally select a group from the dropdown list provided in the control panel to color the nodes according to groups. The grouping can be done by years, location (public input model), and other meta-data (user and hybrid input models). Clustering the virus strains into groups would provide deeper insights into the vaccine seed selection.

Tooltips and Labels

In the phylogenetic tree, hovering over the tree's leaf nodes shows a modal box with defined styling displaying strain name, year, and other meta-data as per the chosen input model and input Fasta headers. In the phylogenetic tree, the labels are shown beside the nodes. Users can optionally select a group from the dropdown list provided in the control panel to change the labels according to groups.

2.2.2 Highcharts

Few web servers are offering antigenic map visualization tools for users. Our application provides this tooling with efficient interactive features. Highcharts is an absolute JavaScript-based chart and graph library meant to improve web application's views by adding interactivity, responsiveness, and clarity visualizations. Highcharts presents an extensive family of charts, diagrams, and maps that includes a modern SVG-based, multi-platform charting library. It makes it simple to add interactive charts to web and mobile projects. As stated in the official website,

"Highcharts has been in active development since 2009 and remains developer's favorite due to its robust feature set, ease of use and thorough documentation."

Scatter Plot

A two-dimensional or three-dimensional Scatter Plot is a geometrical layout, the most primary version of two/three-dimensional plotting used to display data properties as two/ three variables of a dataset using the cartesian coordinates. In our application, we use the two-dimensional/three-dimensional scatter plot as an antigenic map or cartography to denote the points in the plane as virus strains. The plotted points in the cartography revealed how antigenically near or far virus strains are from each other. Each point has an x, y, and z coordinate value. Highcharts provides an Application Programming Interface (API) for demonstrating a scatter plot and its features.

Coloring Points by Groups

When a user selects a group from the dropdown box present in the control Panel, the highcharts scatter plot loads the data grouped by the selected option and colors the points according to the group. The public model has two categories - year and location. The user model has three categories - year, group1, and group2 (user-defined groups). The year data is required, and group1 and group2 data are optional. The hybrid model has availability for three categories: year, country/region, and a user-defined group. We have years spanning from 1968 to 2016, summing to 48 years, and a total of 73 countries in the public sequences collected.

Tooltips and Labels

When a point is hovered over the antigenic map and the tree respectively, a modal box defined styling pops up and shows the virus strain group, year and name. Tooltips are a better replacement for labels in cartography. Showing labels as strain names in the cartography can hide some points. Hence using tooltips are

easier to distinguish between points. In cartography, hovering on the points shows a modal box showing the strain and strain name group. Additionally, all the points in the group are highlighted with the all other points dimmed in the chart.

2.2.3 Other Views and Features

Legends

The legend for the tree is generated by passing the meta-data from JSON and after mapping and scaling colors to values. The legend is positioned by calculating the margin and width of the parent div. The legend for the antigenic map is placed below the chart, and the user can hover over any legend item to see the points belonging to the group highlighted.

Bootstrap Components

We used several Bootstrap components in displaying our view pages. We used select components and wrote functionalities to select/deselect options. We used custom radio buttons, text boxes, buttons, etc., in the `input_sequence.php` view page. In the review page, we used the bootstrap table component. The first column is a collapsible element that a user can click to see more data in a row. There are checkboxes in all the rows, which a user can select anywhere in the row space. There are pagination modules to render 25 rows per page. Users can filter the reviews by selecting/deselecting the column attributes. Users can select/deselect all the rows in all the paginations or on a single page. There are alert boxes in all the view pages that show validation errors and status messages. In the output view page, bootstrap panels are used to design a control panel.

Reviewing Input

A user has to perform actions like selections of radio buttons, checkboxes, file upload, and entering text in the first view page. After a user selects an input model, enters input data in Fasta format as required, it clicks on the submit

button. The data is then passed and handled by the controller that performs various operations on the data and passes the manipulated data to load another view page where the user can review its input in the form of a table. This page is basically a tabular data structure presentation of the Fasta file uploaded by the user or retrieved from the MySQL database. This view allows a user to select/deselect virus strains to produce the output charts. If the datasets are large, a pagination module divides the dataset into divisions of 25 rows (virus strain data) per page. The user can either select/deselect a single page or all the pages or some virus strains as per need. The datasets from the public model are presented in ascending order of year and location. The datasets from the user model are maintained and presented as entered by the user. The snapshot of the review page is shown in figure 2.4.

	#	Virus Name	Year	Country/Region
<input type="checkbox"/>	1	A/Northern_Territory/60/1968	1968	Australia
<input type="checkbox"/>	2	A/Northern_Territories/60_Y2/1968	1968	Australia
<input type="checkbox"/>	3	A/Hong_Kong/01/1968	1968	China
<input type="checkbox"/>	4	A/Beijing/1/1968	1968	China
<input type="checkbox"/>	5	A/Hong_Kong/I_12_MA21_3/1968	1968	Hong Kong
<input type="checkbox"/>	6	A/Hong_Kong/I_9_MA21_3/1968	1968	Hong Kong
<input type="checkbox"/>	7	A/Hong_Kong/I_8_MA21_3/1968	1968	Hong Kong
<input type="checkbox"/>	8	A/Hong_Kong/I_68	1968	Hong Kong
<input type="checkbox"/>	9	A/Hong_Kong/I_11_MA21_3/1968	1968	Hong Kong

Figure 2.4: Review before Job Submission

Interactivity between Phylogenetic Tree and Antigenic Cartography

The user has to select enable selection in either of the visualizations to select nodes/points. Enabling selection in one of the two charts would stop all browser selections in the other chart. For selecting multiple points in cartography, a user has to first select a point and then press the ctrl or command key to select more points. The user has the option of dragging the three-dimensional scatter plot and rotate to find the point in the case of large datasets. Users can also enable and disable zooming by dragging the mouse and selecting points to view them in a widened chart. When a user selects a node/point in one and then clicks on

viewpoints, the same strain data points or nodes are highlighted in the other chart. This is an important feature as it would let researchers find how close or far the antigenicity of the same clade group strains is from each other. Figures 2.5 and 2.6 shows the selected points and nodes in a cartography and tree respectively.

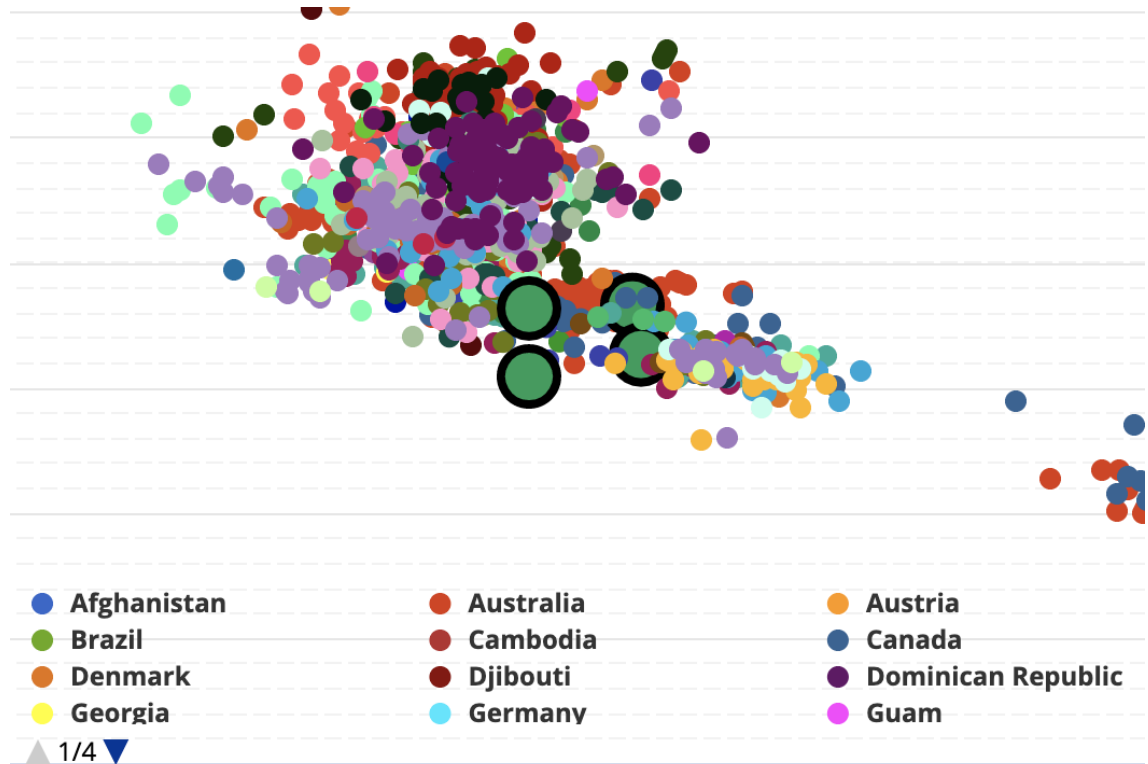


Figure 2.5: Selecting viral antigens in the Antigenic Cartography

Control Panel

The control panel provides the following list of features for controlling the phylogenetic tree

- Generate a rectangular layout
- Generate a circular layout
- Reset View
- Toggle distance labels
- Toggle leaf labels
- Scale by distance

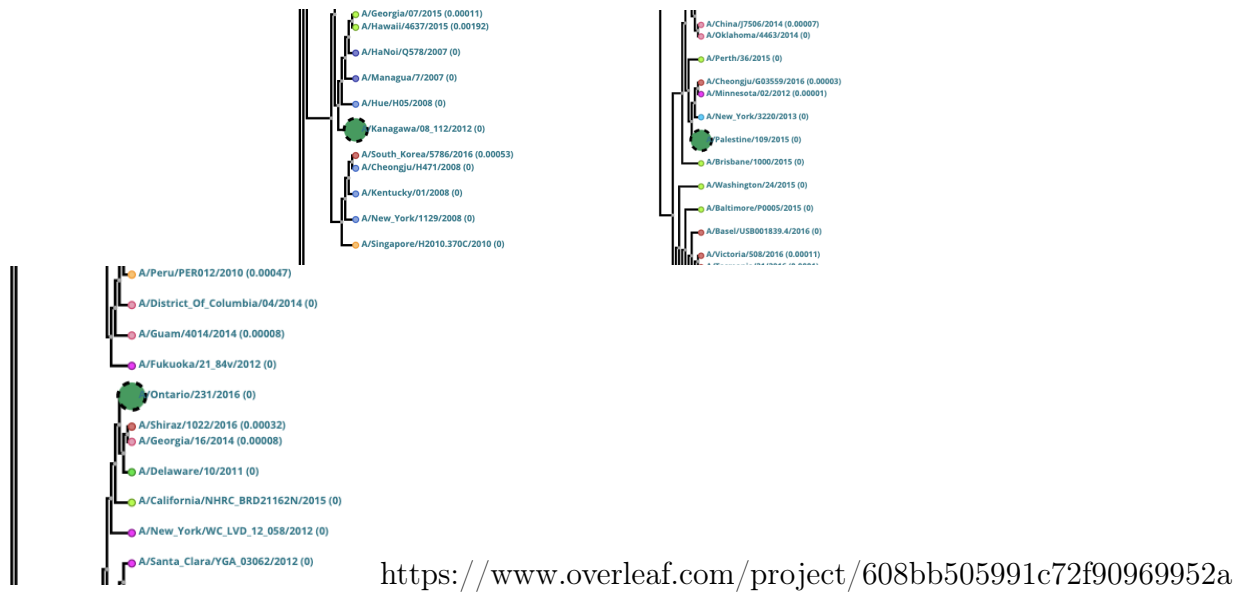


Figure 2.6: Viewing those nodes in the Phylogenetic Tree

- Change Vertical scale
- Change leaf radius
- Rotation a circular layout
- Node Coloring
- Label Grouping

Exporting Files

The user can export both the genetic and antigenic distance matrices, aligned sequences, and Newick tree string. Users can print the antigenic cartography and download the MDS coordinates.

2.3 Data Collection and Manipulation

Sequence data were sourced from Influenza Research Database (IRD)[4] and National Center for Biology Information (NCBI)[6]. All the sources had provisions for downloading data in Fasta format. Hence all the cleaning processes written in PHP scripts were for Fasta. The cleaning process consists of several steps, starting

with validations and ending with format conversion to CSV. An advantage of converting data to CSV is easy and faster insertion into MySQL database. In the first step, we checked if the data format was Fasta (by checking the number of lines between the ">" string to be consistent) and checking for invalid headers and sequences. Then we chopped the Fasta reads into an array of keys as headers and values as sequences. We further chopped the headers and extracted the strain name, year, and origin. The antigenicity prediction algorithm required unique sequences to predict the distance matrix. Hence, our database consists of only unique sequences. The input sequences are also required to be full reads (only 566 lengths long). The data was then uploaded into the MySQL database. Each row consisted of a unique index and name. The data is retrieved on-demand by the calling functions in the controller class. The data retrieval function definitions are documented in the MvaccineEvol model class.

2.4 Application Components

2.4.1 Public Input Model

We named the Public input model so because it contains data fetched from public databases. The user can choose years ranging from 1968 to 2016 and country of origin as query parameters to pull data sequences from the MySQL database. This model consists of pre-computed distance matrices (genetic and antigenic) stored in the server. When the user chooses parameters and submits the query, the program retrieves data from the database with a unique id for each strain which is already mapped to the pre-computed distance matrix for all the virus strains. The distances between the selected virus strains are then fetched by iterating only over those virus strain's unique ids that return a sub-matrix. The sub-matrix is also a distance matrix as only those rows and columns are selected. The distance matrix is then reduced into two or three-dimensional data by per-

forming Classical multidimensional scaling. MDS allows us to visualize how near points are to each other for dissimilarity matrices and can represent our data in a small number of dimensions. A $n \times n$ scale matrix is reduced to $n \times 3$ matrix. The $n \times 3$ matrix is then inferred as coordinates and visualized in a 3-dimensional scatter plot with grouping by year and location. A phylogenetic tree is constructed by converting the genetic distance matrix into a Neighbor-Joining (NJ) Newick string. The NJ algorithm performs fast calculation compared to UPGMA (unweighted pair group method with arithmetic mean) and is more reliable for large datasets. Storing pre-computed genetic and antigenic distance matrices in the server reduces a lot of time and load. The processing time is rampantly reduced. The loading time is only affected by calculating Multidimensional Scaling 3D or 2D coordinates from an antigenic distance matrix, Neighbour-Joining Newick string from a genetic distance matrix, and construction of phylogenetic tree and antigenic map visualization front-end browser. The meta-data is grouped and sorted using underscore.js (`_groupBy` and `_sortBy`) functions in javascript, which means data grouping is handled on the browser side. Figure 2.7 shows a snapshot of the Input Public Model in our application, and figure 2.13 describes the process flow.

2.4.2 User Input Model

The user model is named so because this model allows a user to enter any number of virus sequences to analyze their divergence and similarity. There are choices to input a Fasta file or paste Fasta contents in the textbox. The sequence data is checked against various validations. If the data does not follow the given format, an error message is returned. The error message would then help the user to reset and modify the input sequences. The validation process is as follows:

- Upload file size check.
- Fasta format check
- Sequences and header checks.

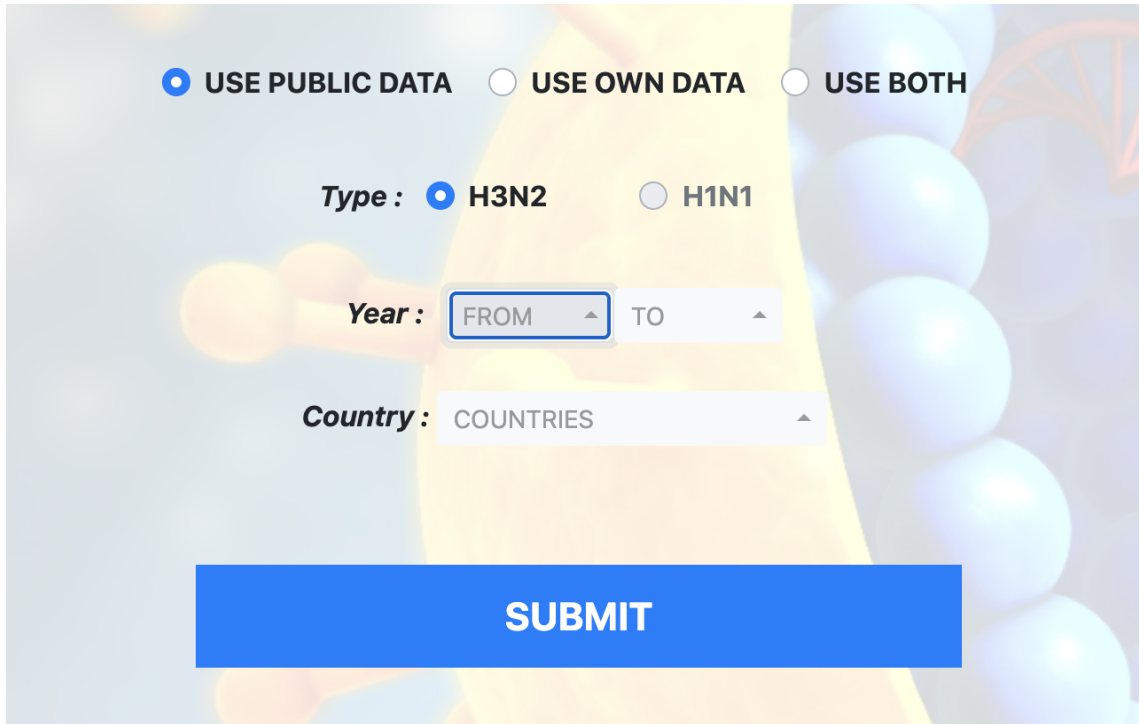
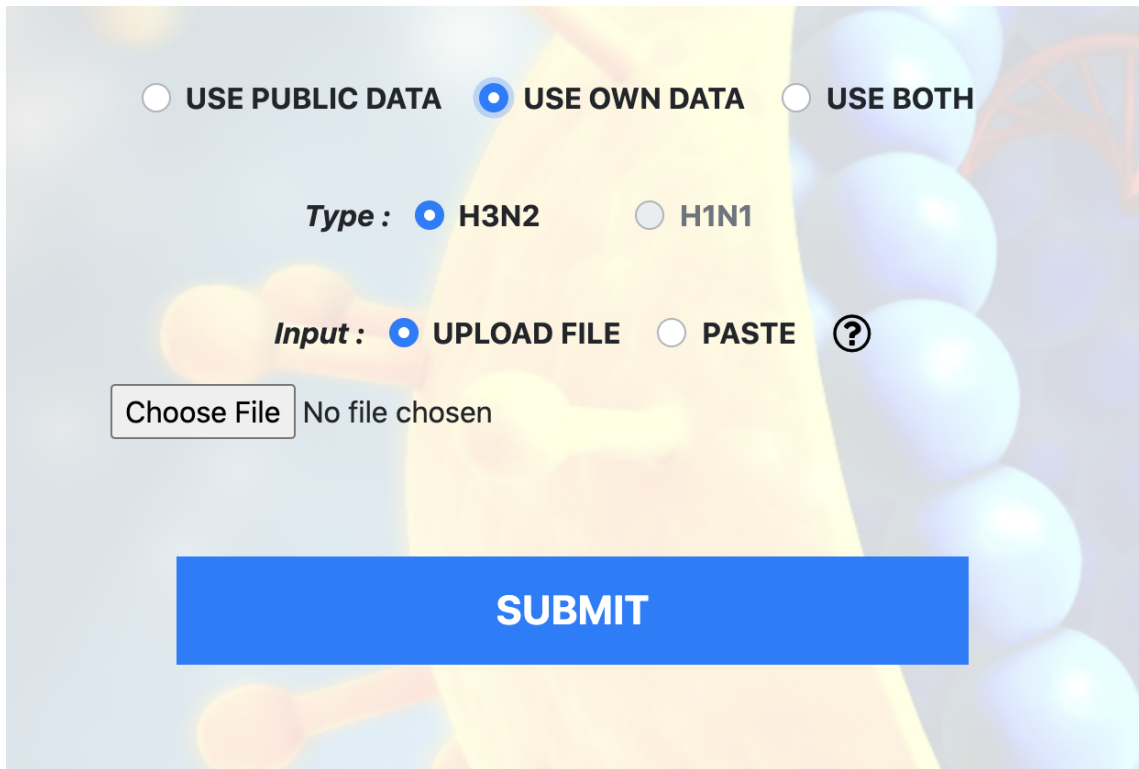
The image shows a web form titled "Public Input Model". At the top, there are three radio buttons: "USE PUBLIC DATA" (selected), "USE OWN DATA", and "USE BOTH". Below this, there are two radio buttons for "Type": "H3N2" (selected) and "H1N1". The "Year" field consists of two dropdown menus labeled "FROM" and "TO". The "Country" field is a dropdown menu currently showing "COUNTRIES". At the bottom of the form is a large blue button labeled "SUBMIT". The background of the form is a light blue and yellow gradient with a faint image of a virus particle.

Figure 2.7: Public Input Model

- Unique sequences check
- Full length HA sequence length (566) check
- Year check (1968 to 2016)
- Metadata data-type check
- Minimum number of sequences requirement check

If the input data passes through all the validations successfully, then the user can see the input sequences in a table format. It can review to select or deselect the virus strains to analyze. The input selected user sequences then go through Multiple Sequence Alignment, and the aligned file is called by two functions: Antigenic Distance Prediction described in the [2], which is the calling function located in base-site directory predict_distance.m which outputs antigenic distance matrix and the other method is Genetic distance matrix using NINJA tool. Once the distance matrices are generated, the antigenic distances are reduced to 3-dimensional data points, and the genetic distances are converted to Neighbour Joining Newick

string. The meta-data is structured as required by the phylogram and cartography visualizations in view component (output.php) written in HTML, javascript, jQuery, and Php. Figure 2.8 shows a snapshot of the Input User Model in our application and figure 2.14 describes the process flow.



USE PUBLIC DATA USE OWN DATA USE BOTH

Type : H3N2 H1N1

Input : UPLOAD FILE PASTE

Choose File No file chosen

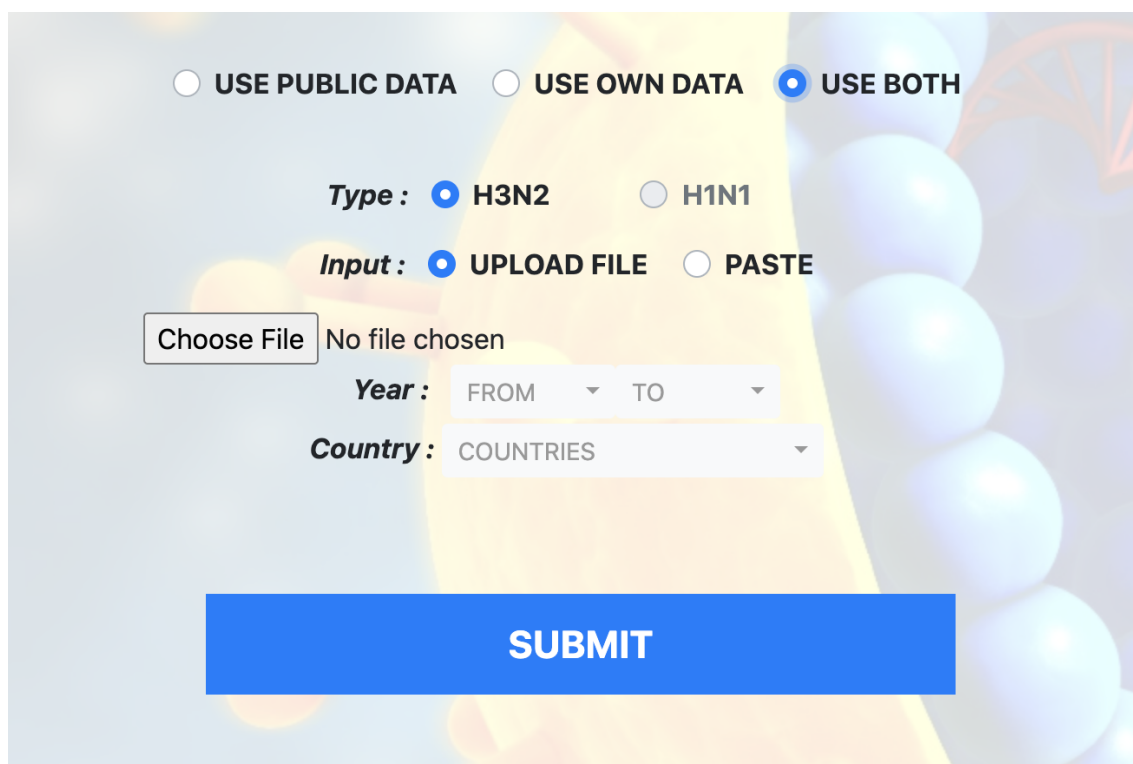
SUBMIT

Figure 2.8: User Input Model

2.4.3 Hybrid Input Model

The hybrid model is a combination of the public model and user model. It is crucial for researchers to analyze and compare their tested sequences against publicly available sequences to derive significant results. This input model can serve the purpose of entering sequences and selecting publicly available sequences to extract similarities and divergence between them. The hybrid input model has the same validations, requirements, and user interface as the user model and public model. The user sequences and public sequences retrieved from the database are combined and go through similar functionalities as the user input model. Figure 2.9 shows a snapshot of the Input Public Model in our application, and figure 2.15

describes the process flow.



USE PUBLIC DATA USE OWN DATA USE BOTH

Type : H3N2 H1N1

Input : UPLOAD FILE PASTE

Choose File No file chosen

Year : FROM ▾ TO ▾

Country : COUNTRIES ▾

SUBMIT

Figure 2.9: Hybrid Input Model

2.4.4 Output Models

Phylogenetic Tree

- *About*

The development, visualization, and analysis of phylogenetic trees contribute to biological interpretation in multiple fields, including evolutionary biology, genetics, and comparative genomics.

- *Objective*

This project aimed to develop a web-based tool that would allow users to visualize high-end phylogenetic trees and antigenic cartography to analyze the genetic and antigenic properties of strains. Several visualization tools

cater to this goal, but most of them need to be downloaded as a stand-alone system. Using web languages to render trees and cartographies has numerous benefits:

1. The frontend construction of trees is performed on the client-side (browser), which reduces the load on the server.
2. It allows interactivity between the tree and cartography.
3. Extending visual features or any other functionalities is easier using web languages than server-side languages.

All these advantages allow a user to control the displays easily without much learning.

- *Development*

The virus strain sequences in Fasta format are first aligned using MUSCLE, and the aligned Fasta sequence file is fed to NINJA to calculate the genetic distance matrix. The genetic matrix is then converted to Neighbour Joining Newick String again using NINJA. The Newick string is then parsed to equivalent JSON (Javascript Object Notation) representation, with every object having a name (strain name) and length (branch length). The object is then consumed by `d3.layout.hierarchy` to create links and nodes and forms a mid-rooted phylogenetic tree. A `d3.hierarchy` is a nested data structure describing a tree: each node has one parent node represented as `node.parent`, except for the root; Similarly, each node has one or more child nodes represented as `node.children`, except for the leaves. Additionally, each node has associated data represented as `node.data` to store additional data. A `d3.hierarchy` is an abstract data structure for working with hierarchical data. This implementation revises the depth of interior nodes in a cluster layout to show branch lengths.

Antigenic Cartography

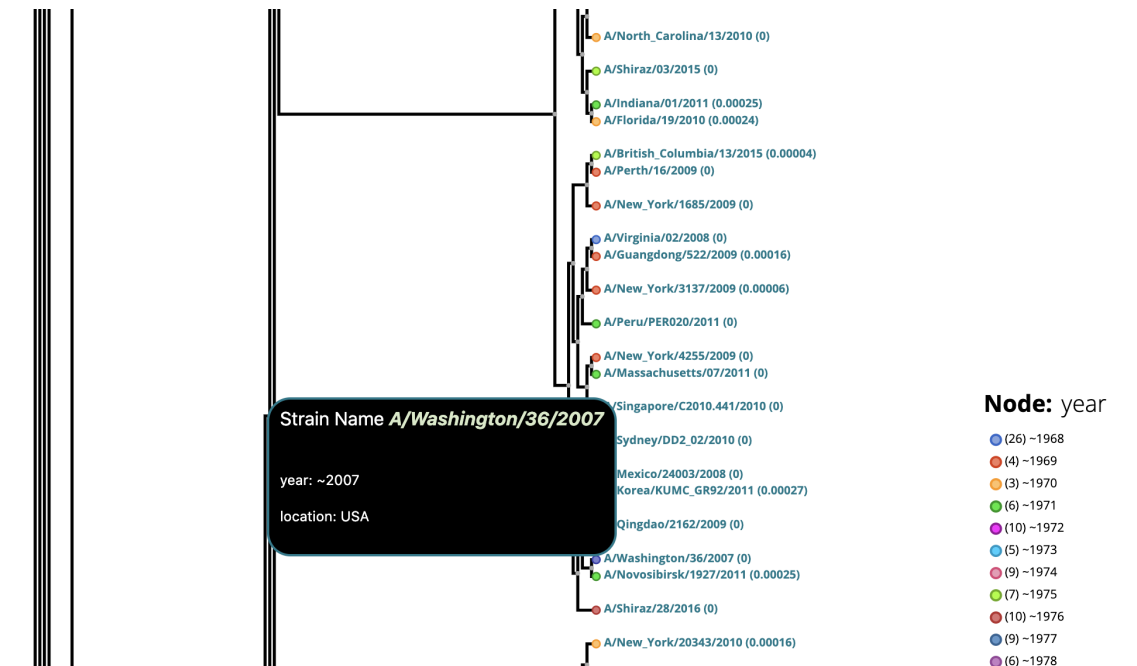


Figure 2.10: Phylogenetic Tree with Tooltip Information

- *About*

The antigenic drift of influenza viruses over a period of time can be visualized straightforwardly. Antigenic cartography is beneficial in the influenza virus evolution study and influenza viruses surveillance. Most importantly, antigenic cartography plays a significant role in vaccine updating decisions. By calculating the antigenic distances between a vaccine strain and circulating strains, researchers can make an informed decision on whether the distances are large enough to warrant a vaccine update not.

- *Objective*

Our goal was to map the antigenic divergence among viruses by years and other metadata in a simplified visual to see how any strain is antigenically different from others. We used the highcharts two-dimensional/three-dimensional scatter plot to visualize the coordinates. It provides many useful options like zooming, panning, dragging to see the points better.

- *Development*

We used the H3N2 antigenic prediction function described in 1.2 under Se-

quence based Influenza Antigenic Analyses. The algorithm runs in the background to construct an antigenic distance matrix. Then the distance matrix is projected to two/three dimensions using multidimensional scaling. The coordinates and meta-data array structure are combined and are first grouped and sorted as series data for 2D/3D scatter plots.

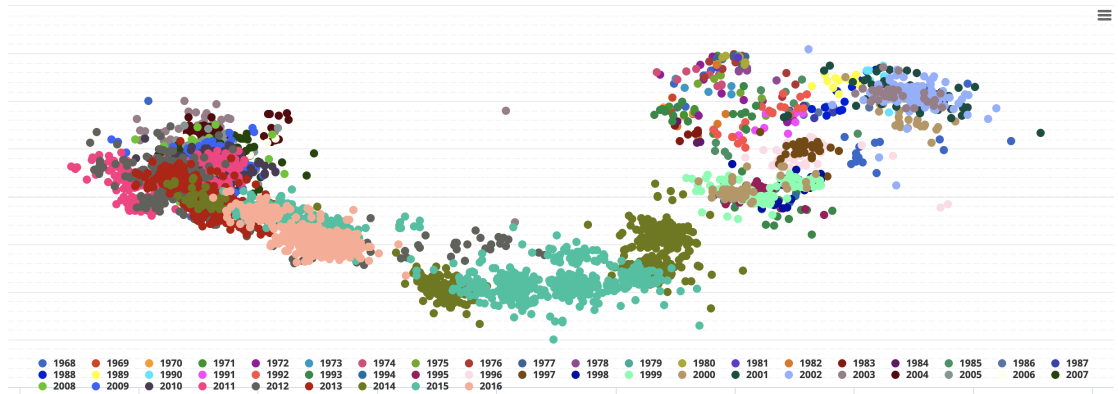


Figure 2.11: 2D Antigenic Map by Year

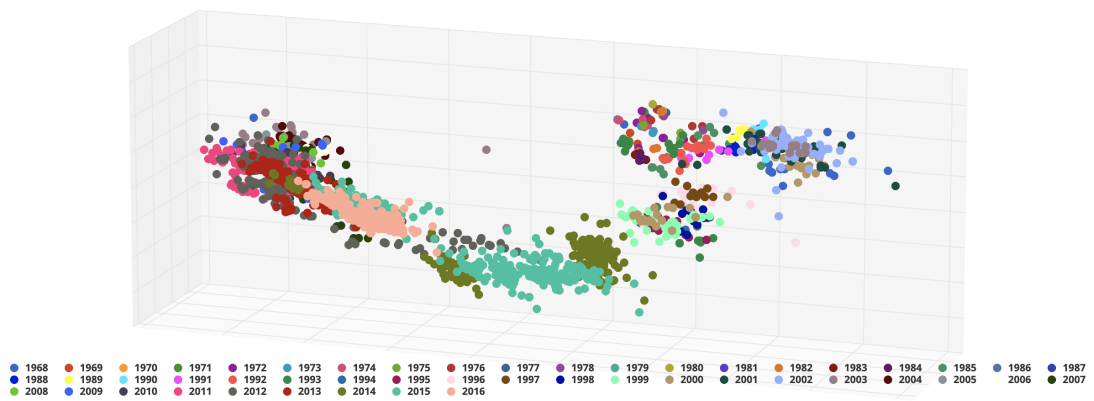


Figure 2.12: 3D Antigenic Map by Year

2.5 Implementation

2.5.1 Process

The workflow goes through several back-end and browser-based (front-end) functions to finally form meaningful visualizations for precision analysis. The web

application provides three methods of input models, namely Public, User, and Hybrid. The server can handle input data up to 2 GB. We downloaded H3N2 virus protein sequences from the open-source Influenza Research Database (IRD) and National Center for Biology Information (NCBI) database for fetching the Public data. We collected a total of 16216 virus strains from IRD and 16201 virus strains from NCBI. The data was downloaded in Fasta format, and it was made sure that the header consisted of unique virus strain name, location, and year. Only unique virus strain sequences having full HA1 sequence length were included in the public database. The H3N2 Machine Learning algorithm predicts correct sequences for years spanning from 1968 to 2016. The Fasta format requires to be in a particular format.

2.5.2 Flow Diagrams

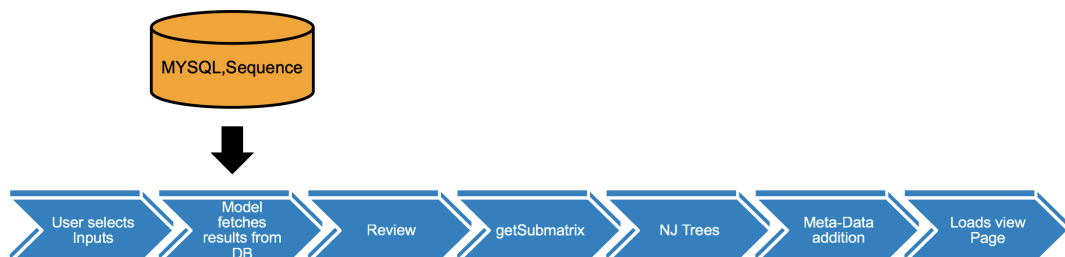


Figure 2.13: Public Input Model Processing



Figure 2.14: User Input Model Processing

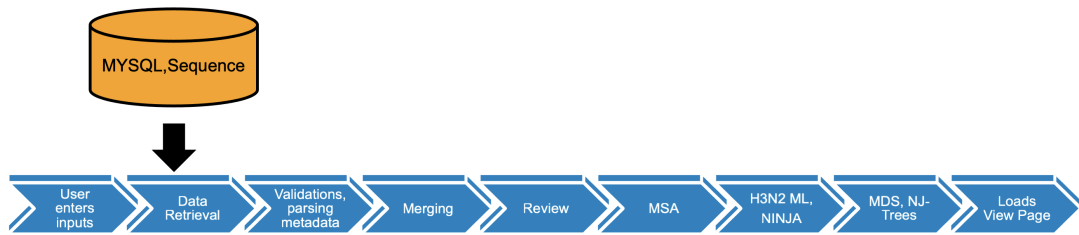


Figure 2.15: Hybrid Input Model Processing

2.5.3 Input Fasta Formats

The user has to input a Fasta file sequence in Fasta format with below given header formats:

- Format 1 : > Strain name / Year

```
>A/Hong_Kong/33/1973
MKTIIALSYIFCLVLGQDFPQNDNSTATLCLGHHAVPNGTLVKTITNDQIEVTNATELVQ
SSSTGKICNNPHRILDGIDCTLDALLGDPHCDGFQNETWDLFVERSKAFSNCYPYDVPD
YASLRSLVASSGTLEFINEGFTWTGVTQNGGSNACKRGPDSGFFSRLNWLKSGSTYPVL
NVTMPNNDNFDKLYIWGVHHPSTDQEOTSLYVQASGRVTVSTKRSQQTIIIPNIGSRPWVR
GLSSRSISYWTIVKPGDILIIINSNGNLIAPRGYFKMRTGKSSIMRSDAPIGTCISECITP
NGSIPNDKPFQNVNKITYGACPKYVKQNTLKLATGMRNVPEKQTRGIFGAIAGFIENGWE
GMIDGWYFRHQNSEGTGQAADLKSTQAAIDQINGKLNRIIEKTNEKFHQIEKEFSEVEG
RIQDLEKYVEDTKIDLWSYNAELLVALENQHTIDLTDSEMKNLFEKTRRQLRENAEDMGN
GCFKIYHKCDNACIGSIRNGTYDHDVYRDEALNNRFQIKGVELKSGYKDWILWISFAISC
FLLCVLLGFIMWACQKGNIRCICI
```

Figure 2.16: Fasta Header Format having Strain name and Year

- Format 2 : > Strain name / Year | Group 1

```
>A/Hong_Kong/33/1973 | Hong Kong|
MKTIIALSYIFCLVLGQDFPQNDNSTATLCLGHHAVPNGTLVKTITNDQIEVTNATELVQ
SSSTGKICNNPHRILDGIDCTLDALLGDPHCDGFQNETWDLFVERSKAFSNCYPYDVPD
YASLRSLVASSGTLEFINEGFTWTGVTQNGGSNACKRGPDSGFFSRLNWLKSGSTYPVL
NVTMPNNDNFDKLYIWGVHHPSTDQEOTSLYVQASGRVTVSTKRSQQTIIIPNIGSRPWVR
GLSSRSISYWTIVKPGDILIIINSNGNLIAPRGYFKMRTGKSSIMRSDAPIGTCISECITP
NGSIPNDKPFQNVNKITYGACPKYVKQNTLKLATGMRNVPEKQTRGIFGAIAGFIENGWE
GMIDGWYFRHQNSEGTGQAADLKSTQAAIDQINGKLNRIIEKTNEKFHQIEKEFSEVEG
RIQDLEKYVEDTKIDLWSYNAELLVALENQHTIDLTDSEMKNLFEKTRRQLRENAEDMGN
GCFKIYHKCDNACIGSIRNGTYDHDVYRDEALNNRFQIKGVELKSGYKDWILWISFAISC
FLLCVLLGFIMWACQKGNIRCICI
```

Figure 2.17: Fasta Header Format having Strain name, Year and Group1

- Format 3 : > Strain name / Year | Group 1 | Group 2

We wrote several functions for transforming formats between Fasta, JSON, CSV, Tab delimited Files, Newick etc.

```

>A/Hong_Kong/33/1973 | Hong Kong | Group A
MKTIIALSIFYCLVLGQDFPGNDNSTATLCLGHAVPNGTLVKTITNDQIEVTNATELVQ
SSSTGKICNNPHRILGDIDCTLIDALLGDPHCDGFQNETWDLFVERSKAFSNICYPYDVPD
YASLRSLVASSGTLEFINEGFTWTGVTQNGGNSACKRGPDSGFFSRNLNWLKSGSTYPVL
NVTMPNNDNFDKLYIWGVHHPSTDQEOTSLYVQASGRVTVSTKRSQQTIIIPNIGSRPWVR
GLSSRISYWTIIVKPGDILLIINSNGLIAPRGYFKMRTGKSSIMRSDAPIGTCSI ECITP
NGSIPNDKPFQNVNKITYGACPKYVKQNTLKLATGMRNVPKQTRGIFGAIAGFIENGWE
GMIDGWYGRHQNSEGTGQAADLKSTQAAIDQINGKLNRIIEKTNEKFHQIEKEFSEVGE
RIQDLEKYVEDTKIDLWSYNAELLVALENQHTIDLTDSEMKNLFEKTRRQLRENAEDMGN
GCFKIYHKCDNACIGSIRNGTYDHDVYRDEALNNRFQIKGVELKSGYKDWILWISFAISC
FLLCVLLGFIMWACQKGNIRCNICI

```

Figure 2.18: Fasta Header Format having Strain name, Year, Group1 and Group 2

2.5.4 Tools and Utilities

Muscle

According to announced benchmark tests, MUSCLE is a multiple sequence alignment software that is one of the best-performing alignment programs. MUSCLE stands for Multiple Sequence Comparison by Log- Expectation. The precision and pace are consistently better than other alignment software like CLUSTALW, T-COFFEE. MUSCLE can align hundreds of sequences in seconds. Only a handful of command-line options are needed to perform standard alignment tasks. Many MSA algorithms have been proposed. MUSCLE is reliable when it comes to protein sequences. Hence we chose MUSCLE to perform sequence alignment. Sequence alignment software must exhibit two primary features; it should give biological precision and lessen the time and memory requirements. We found MUSCLE to have a good amount of balance of these two features [13]. We used the following command to input a Fasta file and got the aligned sequences for large datasets:

```
muscle -in seqs.fa -out seqs.afa -maxiters 2.
```

Ninja

We used Ninja to calculate the genetic distance matrix from the aligned sequences generated by MUSCLE. The genetic distance matrix gives pair-wise distances between sequences based on the number of substitutions. The distance matrix is then converted to a tree using the Neighbor-Joining (NJ) method. NJ is a well-organized bottom-up hierarchical clustering method for inducing phylogenies. It

starts with observed pair-wise distances between sequences, and clustering order depends on a metric associated with those distances. The official algorithm has time complexity $O(n^3)$ and space complexity $O(n^2)$ for n sequences. Recent algorithmic improvements have considerably expedited NJ for inputs of thousands of sequences but are restricted to fewer than 13,000 sequences on a system with 4GB RAM. NINJA is an NJ software that uses an algorithm that promotes the speed of the NJ by reducing the number of distance values seen in each iteration of the clustering method while still calculating a suitable NJ tree. This algorithm can balance inputs larger than 100,000 sequences because of external-memory-efficient data structures. The Ninja NJ algorithm's run time is the same as the canonical algorithm, but it is more reliable for large datasets [20].

The command for distance matrix:

```
./ninja -out_type d alignment.Fasta > distance_file
```

The Command for neighbour-joining newick file:

```
./ninja -in_type d distance_file > tree.newick
```

Multidimensional Scaling

One of the many essential purposes in conceiving data is to see how close or distant points are from each other. Frequently, we visualize this with a scatter plot. A Scatter plot is a means of seeing points in a two or three-dimensional coordinate plane. A distance matrix can be large enough for which a scatter plot would fail to display the distances. Hence, it is challenging to reflect distances with many variables unless we can represent the data in a small number of dimensions. Some dimension reduction is usually needed. Multidimensional scaling (MDS) is a collection of techniques that address all these queries. MDS allows us to visualize how near points are to each other for many kinds of distance or dissimilarity metrics and can generate a description of our data in a small number of dimensions. MDS requires a matrix of pair-wise distances or dissimilarities. $Y = cmdscale(D)$ takes an n -by- n distance matrix D and returns an n -by- p configuration matrix Y .

Rows of Y are the coordinates of n points in p -dimensional space for some $p < n$. p is the dimension of the smallest space in which the n points whose inter-point distances are given by D can be embedded.

Regex

A regular expression (shortly, Regex or regexp, also known as rational expression) is a series of a pattern of characters that designates a search pattern. Ordinarily, such patterns are applied by string-searching algorithms during "find" or "find and replace" methods on strings or for input validation. It is a procedure developed in theoretical computer science and formal language theory.

We used regular expressions in our application at various steps for extracting and isolating types of data. Regexes were used in MATLAB and PHP scripts. The expressions and sequences used to write Regex vary from one language to another. We used Regex in PHP scripts for filtering out meta-data and sequences in the user and hybrid input models. Furthermore, Regex was used to differentiate and extract virus strain name, year, and start from the meta-data for further data processing. Several functions used Regex to check sequences validations and emptiness also. In MATLAB scripts, for e.g., in `mds.m` (that performs classical multidimensional scaling), any distance matrix format can be passed to extract the euclidean distance matrix. It removes any other headers or meta-data in any row or column. `Mds.m` hence is an independent script that can convert any distance matrix format to 3-dimensional coordinates. One of the most potent usages of Regex is Regular expressions with backreferences. Hence, using Regex can save much coding and make our work easier in fewer lines of code with faster implementation.

Chapter 3

Application

This chapter aims to introduce several self-defined and pre-defined functions of CodeIgniter and the bioinformatics tools used in developing the application.

3.1 Session Management

When developing web apps, we regularly need to trace and store the state of a user's activity, and hence, we have to use sessions. CodeIgniter consists of a Session class for this objective. The system can automatically load the session or initiate it manually in a controller's constructor, as the session operates as a global environment variable. The Session class will run ignored in the framework, so simply initializing the class will provoke it to read, create, and update sessions when required.

Session data is just an array associated with a particular session ID. CodeIgniter access to its session data is similar to the session handlers' mechanism provided by PHP. Handling session data is as easy as managing (read, set, and unset values) in the `$_SESSION` array. In addition, CodeIgniter also provides two more particular types of session data that flashdata and tempdata.

Our application structured the session array to store the user data, absolute paths of input and output files, and some other important data like input models,

status to be used in the views. The session array being global can be accessed anywhere in the application. Hence we made sure the array is not large by storing file paths instead of the data itself. The `_reset` function clears all the session data when the user clicks the back button.

3.2 Job Management

The job function written in the controller class is used to create unique URLs for process/job management. When a user runs a job (submits validated input sequences), a new job ID is created in the Mysql using `mjob` model class. As such, the URL becomes `baseurl.softwares/VaccineEvol/jobs/Id`. This Id is unique and is assigned to every user after successful output generation. The job table holds the ID along with the output file path. When the user later clicks on that particular URL, it can see the visualizations without any waiting time. This approach helps in reducing the load on the server and use the historical jobs already ran by the users. The output files are created and named in a way that ensures it is unique and is easily retrieved from the database. The output and input files are stored in `baseurl.tmp/VaccineEvol` directory. A unique directory is created for a particular user session by combining the user's IP and a random number returned by PHP's `mt_rand()` function. Below is a snap of how we created unique names for the user files storage. The folder and files are given all the read, write and execute permissions.

3.3 Pre-defined Functions

Constructors are methods that are called when the class loads. They can be used to perform variable declarations and calling functions by default, but they can't return a value. In our application, the `_construct()` ... initializes the parent constructor method and loads the URL and form helper classes. It loads the `mjob`

```

$this->load->model('PhyloMap');
// now we can separate session_id from tmp folder name
if (!isset($_SESSION['curFolder']))
{
    $folder = '';
    while (strlen($folder) < 32) $folder .= mt_rand(0, mt_getrandmax());
    $folder .= $_SERVER['REMOTE_ADDR']; // combine it with the user's IP
    $folder = md5(uniqid($folder, true)); // Turn it into a hash
    $_SESSION['curFolder'] = $folder;
    $_SESSION['tmpFolder'] = $folder;
}

$folder = $_SESSION['curFolder'];
// directory info for third-party resources
$this->dataBase = FCPATH . 'tmp/PhyloMap/';
$this->dataPath = $this->dataBase . $folder . '/';

$this->codePath = FCPATH . 'code/PhyloMap/';
$this->muscleCmd = FCPATH . 'code/PhyloMap/muscle';
$this->matlabCmd = FCPATH . 'code/PhyloMap/matlab';
$this->ninjaCmd = FCPATH . 'code/PhyloMap/ninja/ninja';
// make sure tmp folder exists
if (!is_dir($this->dataPath))
{
    //echo 'preparing to mkdir '.$this->dataPath;
    //mkdir('/var/www/html/test');
    @mkdir($this->dataPath);
    @chmod($this->dataPath, 0777);
}

```

Figure 3.1: Unique Temp File Storage for User

and mVaccineEvol Model classes. This way, we don't have to load it in every other function in the same class. We also declared some class-based global variables to be used across functions.

The Index() function is called first after the class is instantiated. In our application, the index function is the first entry point that consists of view page switching.

3.4 File Uploading

File management is imperative to maximum web applications. CodeIgniter's File Uploading class makes it straightforward to upload files of several mime types. The path of the file upload directory must have permissions to write or else users would not be able to upload files. Apache user recursively owns all the files. The tmp/VaccineEvol folder consisting of all the input uploads and outputs has file permissions of 777. Figure 3.2 shows the file uploading configuration in the controller class.

```
$config['upload_path'] = $this->dataPath;
$config['allowed_types'] = '*';
$config['max_size'] = 0;
$config['remove_spaces'] = true;
$config['file_name'] = 'userFile' . $time_ext;
$this->load->library('upload', $config);
if (!$this->upload->do_upload('seq_file_user'))
{
    $rtn = array('errors' => $this->upload->display_errors(), 'info' => '');
    return $rtn;
}
```

Figure 3.2: File Upload Configuration

3.5 MVC

CodeIgniter is a Model-View-Controller framework. MVC is a software technique that divides application logic from presentation. In practice, it allows the web pages to include minimum scripting since the presentation is separate from the business logic and business logic is separate from the Model. Figure 3.3 shows

the MVC architectural pattern used in our application.

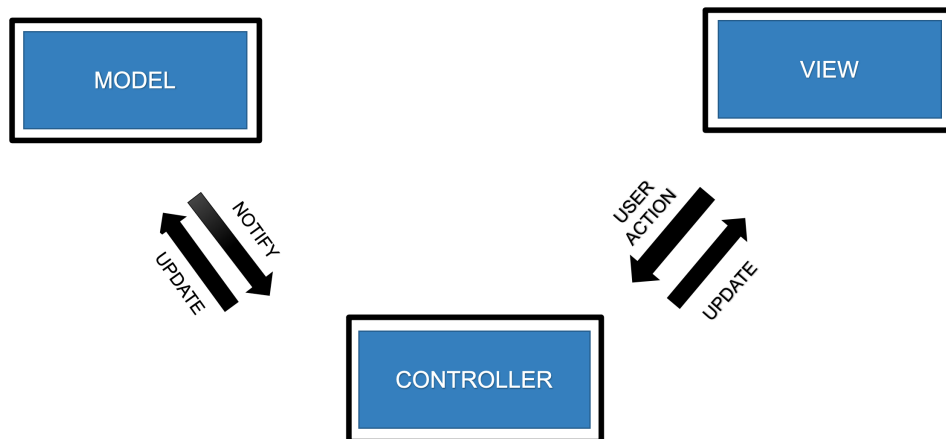


Figure 3.3: Model-View-Controller Software Design

3.5.1 Model

The Model represents the application's data objects from the database. It is the component that has direct communication with the database. Its primary function is to pass the data to and from the application's Controller to the database. So we can call it a connector to the SQL database in our case. To sum up, Typically, our model classes will contain functions that help us retrieve, insert, and update information in our database. The Model specifies what data the application should include. When this data changes, the Model will generally inform the Controller, which reloads the view to update itself. Figure 3.4 shows the Model structure.

In CodeIgniter, Models are PHP classes that work with data in our database. They allow us to interact with a specific table in the database. They consist of helper methods (query-builders) needed to communicate with a database table, including finding records, updating records, deleting records, and more. The path for Model classes is in the application/models folder. This class can be nested and have subfolders within the parent model folder if a hierarchical arrangement is needed.

Controller class typically would load model classes and call them to receive

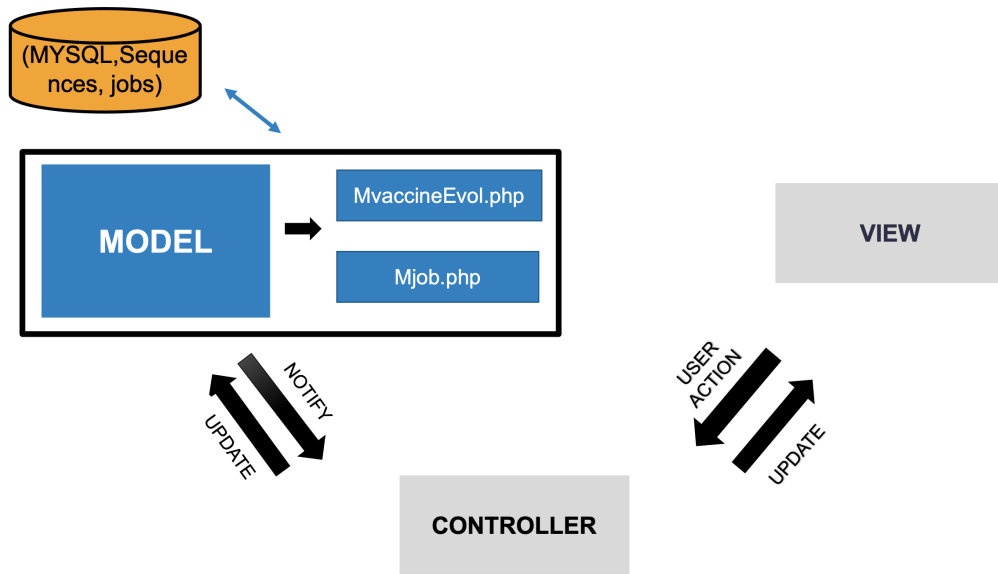


Figure 3.4: Model Class

data from database. A model can be loaded using:

```
$this->load->model('model_name');
```

After loading a model, it can't connect to the database(MySQL) automatically. We manually passed database connectivity settings via the database.php in the config folder.

Our application uses two models. The main Model that has all the querying functions is the MVaccineEvol.php. The functions in MVaccineEvol queries the MySQL database. When VaccineEvol loads in the server, the MVaccineEvol.php is loaded in the constructor that calls The other Model that handles all the jobs and sessions for a particular is the Mjob.php.

3.5.2 View

The View component comprises the User Interface and is primarily is what a user sees when the web page loads in the browser. It consists of all functionalities that manage how users interact with the app. The view is made with HTML, CSS, JavaScript, JQuery, D3.js, Highcharts, and Underscore.js templates. CodeIgniter allows Views to be embedded within other views. A view

```

$active_group = 'default';
$query_builder = TRUE;

$db['default'] = array(
    'dsn' => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => 'cdD,wxmN(1-j2',
    'database' => 'Sysbio',
    'dbdriver' => 'mysqli',
    'dbprefix' => '',
    'pconnect' => FALSE,
    'db_debug' => FALSE,
    'cache_on' => FALSE,
    'cachedir' => '',
    'char_set' => 'utf8',
    'dbcollat' => 'utf8_general_ci',
    'swap_pre' => '',
    'encrypt' => FALSE,
    'compress' => FALSE,
    'stricton' => FALSE,
    'failover' => array(),
    'save_queries' => TRUE
);

```

Figure 3.5: Codeigniter Connection to MYSQL

can consist of other views. Usually we load the view pages like header, footer, navigation panes inside our main view page. Views are never called directly; a controller must load them. The Controller functions as the traffic controller, responsible for fetching a particular view. In our application, the view directory is present in `/application/views/VaccineEvol`. VaccineEvol consists of three view files. `input_sequence.php`, `review.php`, `output.php`. The default page is `input_sequence.php`. It consists of CodeIgniter PHP form elements and HTML5 form validations. The three input models are presented to the user here. If a user does not enter a required field, it can not go further to submit. Figure 3.6 shows the view structure.

3.5.3 Controller

The Controller is the mediator between View and Model. It is responsible for controlling the data transference between the Model and the view. It creates a

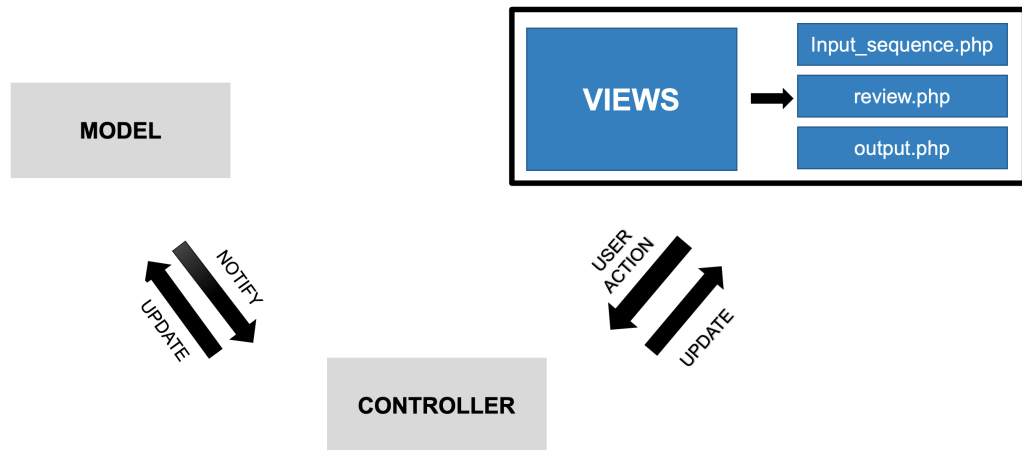


Figure 3.6: View Class

tunnel to map the user actions into model updates. The controller layer is helpful to select the most appropriate view and delivers it to the user. The Controller is the decision-maker and the adhesive between the Model and view. The Controller updates the view when the Model changes. It also adds event listeners to the view and updates the Model when the user manipulates the view.

In our application, the Controller is the most crucial component that contains most business logic. It receives data from the view and calls the Model's functions that pass those to the Model to perform some actions on those data or insert/update into the database. Also it receives data from the Model and performs some operations to passes those to the view component. Since a Controller is a class file's naming is done in a way that can be linked with a URI. Class names should begin with an uppercase letter. E.g., the file should be named as 'VaccineEvol.php,' with an uppercase 'V.' Our controller class is under the software directory in controllers directory. The "index" method is the main entry point and is always loads first if the second segment in the URI (after the controller class) is empty. In the constructor class, we have loaded our two models (mjob.php and MvaccineEvol.php). All the global variables required throughout the class are declared in the constructor. Figure 3.7 shows the list of functions in our controller class and figure 3.8 shows the controller structure.

```

/**
 * Class and Function List:
 * Function list:
 * - __construct()
 * - index()
 * - process_output()
 * - preparePublicTreeData()
 * - highchartsArray()
 * - mapIds()
 * - runMSA()
 * - runPREDICTION()
 * - runMDS()
 * - runGET_SUBMATRIX()
 * - runNINJA()
 * - arrayToCSV()
 * - csvToArray()
 * - runCCV()
 * - filePutContents()
 * - getSelections()
 * - getCount()
 * - jsonToFasta()
 * - arrayToFasta()
 * - getFileNames()
 * - process_file()
 * - _validate_input()
 * - run_validations()
 * - fas_check()
 * - get_seq()
 * - fas_get()
 * - seq_validations()
 * - fastaToArray()
 * - _reset()
 * Classes list:
 * - VaccineEvol extends CI_Controller
 */

```

Figure 3.7: List of Controller Functions

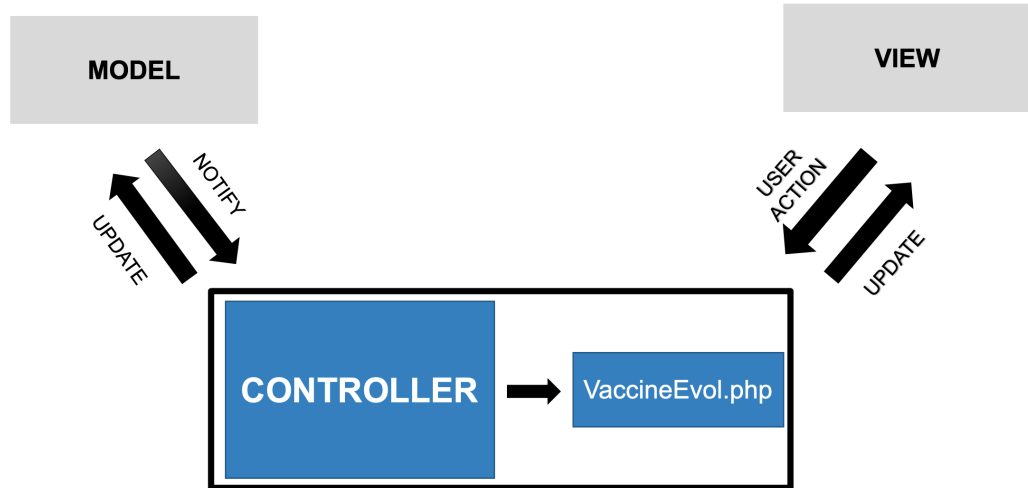


Figure 3.8: Controller Class

3.6 URL Helper

The application uses the .htaccess file located in the application path. The .htaccess file is a dominant website file that manages the high-level configuration of our website. On servers that run Apache as the webserver, the .htaccess file enables

us to adjust our website's configuration without editing server configuration files. Apache is configured so that all files named `.htaccess` are hidden because the files have important configuration settings and can be used to compromise your server. By default, URLs in CodeIgniter are search-engine friendly. Instead of using the standard "query string" approach to URLs, CodeIgniter uses a segment-based approach : `sysbio2.missouri.edu/softwares/VaccineEvol` The first segment is the controller class that should be invoked. Here the Controller `VaccineEvol.php` is located inside the directory named as 'softwares'.

The second segment is the class function or method that should be called. Here the default class that is invoked is the index function. All the segments from the third segment represent the arguments passed to those methods/functions.

3.7 Code Repository

The code repository consists of all the required resources and computational tools, precalculated matrices, softwares, and Matlab scripts that produce the output. Our application is hosted in the Sysbio2 server administered under the university of Missouri IT Department. The domain name is `sysbio2.missouri.edu`. The VaccineEvol software can be found as a list option under the softwares.

3.8 Summary

Our web application VaccineEvol is a full-fledged data-rich resource for influenza vaccine seed selection based on antigenic analyses built with PHP CodeIgniter Framework connected to MySQL database and several usable javascript libraries for efficient construction of phylogenetic trees (D3.js) and antigenic maps (Highcharts) by integrating bioinformatics tools and machine learning derived influenza sequence based antigenicity prediction algorithm developed in SystemsBio Lab.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

This thesis introduced VaccineEvol, a web application for visualizing, interacting, and managing antigenic map alongside a phylogenetic tree for vaccine strain selection using sequence-based user and public data. The thesis addresses how using browser-based visualization makes interactivity and reading of data more accessible and understandable to the end-users. We hope that we would meet the developing needs of our broad range of users.

4.2 Future Work

We aim to add several other visual features enhancing user interactivity and usage. In the future, we aim to keep VaccineEvol up-to-date with enhanced features by supporting a new subtype of influenza A H1N1 to our database and integrating new Machine Learning antigenicity prediction in the backend. We also aim to add more functionalities to improve the hybrid model by pre-calculated matrices mapping with the user sequences, reducing output generation time and computation load.

Bibliography

- [1] Center for disease control and prevention. <https://www.cdc.gov/flu/about/index.html>.
- [2] Colorbrewer. <https://gist.github.com/frankrowe/9007567>.
- [3] D3.js. <https://d3js.org/>.
- [4] Influenza research database. <https://www.fludb.org/brc/home.spg?decorator=influenza>.
- [5] JQuery. <https://jquery.com/>.
- [6] National center for biology information. <https://www.ncbi.nlm.nih.gov/genomes/FLU/Database/nph-select.cgi?go=database>.
- [7] nouislider. <https://refreshless.com/nouislider/>.
- [8] Tooltip. <https://github.com/caged/d3-tip>.
- [9] Twitter bootstrap. <https://getbootstrap.com/2.0.2/>.
- [10] J Lamar Barnett, Jialiang Yang, Zhipeng Cai, Tong Zhang, and Xiu-Feng Wan. Antigenmap 3d: an online antigenic cartography resource. *Bioinformatics*, 28(9):1292–1293, 2012.
- [11] Nicole M Bouvier and Peter Palese. The biology of influenza viruses. *Vaccine*, 26:D49–D53, 2008.

- [12] Zhipeng Cai, Tong Zhang, and Xiu-Feng Wan. A computational framework for influenza antigenic cartography. *PLoS Comput Biol*, 6(10):e1000949, 2010.
- [13] Robert C Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–1797, 2004.
- [14] Zilong He, Huangkai Zhang, Shenghan Gao, Martin J Lercher, Wei-Hua Chen, and Songnian Hu. Evolview v2: an online visualization and management tool for customized and annotated phylogenetic trees. *Nucleic acids research*, 44(W1):W236–W241, 2016.
- [15] Nikola Jovanovic and Alexander S Mikheyev. Interactive web-based visualization and sharing of phylogenetic trees using phylogeny. io. *Nucleic acids research*, 47(W1):W266–W269, 2019.
- [16] Yu-Chieh Liao, Chin-Yu Ko, Ming-Hsin Tsai, Min-Shi Lee, and Chao A Hsiung. Ativs: analytical tool for influenza virus surveillance. *Nucleic acids research*, 37(suppl_2):W643–W646, 2009.
- [17] Richard A Neher and Trevor Bedford. Nextflu: real-time tracking of seasonal influenza virus evolution in humans. *Bioinformatics*, 31(21):3546–3548, 2015.
- [18] Yousong Peng, Lei Yang, Honglei Li, Yuanqiang Zou, Lizong Deng, Aiping Wu, Xiangjun Du, Dayan Wang, Yuelong Shu, and Taijiao Jiang. Predac-h3: a user-friendly platform for antigenic surveillance of human influenza a (h3n2) virus based on hemagglutinin sequences. *Bioinformatics*, 32(16):2526–2527, 2016.
- [19] Hailiang Sun, Jialiang Yang, Tong Zhang, Li-Ping Long, Kun Jia, Guohua Yang, Richard J Webby, and Xiu-Feng Wan. Using sequence data to infer the antigenicity of influenza virus. *MBio*, 4(4), 2013.
- [20] Travis J Wheeler. Large-scale neighbor-joining with ninja. In *International Workshop on Algorithms in Bioinformatics*, pages 375–389. Springer, 2009.