Proposal: Phylogenetic Approach to Quantifying Mutation Rate across Viral Genes
(Tool name: *ViralGeneClock*)

Background & Rationale

In the context of rapidly mutating viruses, phylogeny is conducive to inferring the evolution models and predicting the potential emergence of new strains. The advancement of phylogenic models for analyzing viral evolution has led to the widespread adoption of these tools for vaccine and antiviral development [1]. While tools like TreeTime utilize sophisticated Maximum Likelihood algorithms for phylogenetic studies in viruses [2], there remains a gap in available tools that directly facilitate the comparison of mutation rates and molecular clocks across the genes among different viral strains.

The objective of my project is to build a tool where the users could potentially deposit the whole genome sequence (WGS) of different strains of a virus. The tool would then examine the evolutionary relationship of the strains, and also compare the mutation rates of the genes. The rationale for the development of such a tool is as follows:

1. *ViralGeneClock* would identify regions within the viral genome that are highly conserved across the strains. This could potentially offer vital insights for drug and vaccine development. Focusing on regions prone to rapid mutation for drug/vaccine development might lead to challenges, as the virus could simply evolve beyond the scope of the treatment over time. Therefore, leveraging information on conserved regions, alongside personalized research on the virus itself, could serve as a promising starting point for drug/vaccine design. This approach is being employed in SARS-CoV-2 research; RNA-targeting molecules, which attack conserved RNA structures and sequences, are being emphasized as potential antiviral drug candidates [3].

2. *ViralGeneClock* would assist in understanding the genetic distances and evolutionary ties between different strains of the virus.
3. *ViralGeneClock* would identify regions with rapid mutation, indicating that the region is under strong selective pressure. Recognizing these areas prone to rapid mutation, researchers could potentially anticipate the emergence of new viral variants, and assess the impact of these regions on transmissibility and virulence of a strain.

Developer Environment, Tools & Resources

ViralGeneClock is a Linux tool being developed through the Ubuntu subsystem. The tool will primarily utilize Python3 with the *Biopython* package and other modules (*numpy, pandas, matplotlib*) for data manipulation and visualization. It also leverages the pipeline of publicly available Linux tools such as *Prokka* [4] and *Muscle* [5]. *Prokka* is a command-line tool, implemented in Perl, which offers reliable annotation of genomic viral, bacterial, and archaeal sequences [4]. *Muscle* is a multiple-sequence alignment tool, which will be leveraged to form alignment files for each strain. Neighbor-joining algorithm, implemented from the *Biopython* package, will then use the alignment file for clustering the strains according to their genetic distances. Finally, if time permits, *ViralGeneClock* will be transformed into a web application using Flask, a Python web framework.

Data Sources

Theoretically, the tool will work with WGS across different strains of any virus. To assess and optimize my model, I will primarily be working with SARS-CoV-2 strains. The WGS

of SARS-CoV-2 strains, in FASTA format, is publicly available through NCBI Virus. The WGS of the specified 16 strains will be retrieved via NCBI's e-utilities for testing the tool:

Accession Number	Pango Lineage	Accession Number	Pango Lineage
1. NC_045512	B	9. PP250483	BF.10
2. OR075545	XBB.1.16	10. PP439669	JN.1
3. OQ991501	XBB.1.5	11. PP435534	HV.1
4. PP127519	EG.5.1	12. OQ437945	B.1.1.7
5. PP292788	AY.3	13. PP421053	P.1
6. PP429773	BA.2	14. PP299611	B.1.617.2
7. PP439021	BA.5.5	15. PP292591	BE.1
8. PP298667	CH.1.1	16. PP298634	DN.2

Methodology

1. Viral Annotation: *Prokka* uses an external feature prediction tool, *Prodigal*, to identify the coordinates of protein-coding CDS. *Prodigal* is a non-supervised machine learning algorithm, which detects protein-coding regions by scanning for open reading frames (ORFs), and assigning scores to each ORF based on codon usage bias, GC frame plot and length of the ORF [6]. Then, *Prokka* compares the gene code at a protein sequence level with databases of known sequences to annotate the predicted gene. This produces an .ffn FASTA file with genomic features of all predicted gene. Finally, *ViralGeneClock* arranges the .ffn files from each strain into individual FASTA format files for each gene (each FASTA file will then contain the genome sequence across all the strains for that particular gene).
2. Multiple Sequence Alignment: The gene FASTA file obtained after viral annotation will then be aligned using MUSCLE (Multiple Sequence Comparison by Log-Expectation). MUSCLE will take the homologous gene sequences as an input, and employ a progressive alignment algorithm [5]. In this algorithm, the sequences are initially pairwise aligned based on similarity scores, and then these pairwise alignments are progressively aligned into larger

alignments. In the context of the pipeline for *ViralGeneClock*, this will produce an .aln file format for each gene.

3. Neighbor-Joining Algorithm for Annotated Genes: Neighbor-joining (NJ) is a bottom-up clustering method for estimating genetic distances, branch lengths and creating phylogenetic trees [7]. While neighbor joining algorithm has largely been replaced by other algorithms with superior accuracy such as Maximum Likelihood and Maximum Parsimony, NJ remains the least computationally expensive algorithm among them [8]. This makes NJ appropriate for analyzing large data sets in a local device. In *ViralGeneClock*, NJ will create phylogenetic tree for the submitted viral strains, and calculate branch length and genetic distances for each annotated gene across the strains. The NJ pipeline will also involve bootstrapping to create multiple pseudo-datasets. This will increase the robustness of the model and avoid biases from sampling variation.
4. Mutation Rate Estimation: NJ algorithm will produce identity matrix for all the viral strains (for each gene), and predict branch lengths. *ViralGeneClock* will utilize this data to estimate mutation rate for each gene across the viral strains. The identity matrix provides the proportion of identical sites between sequences, and subtracting the value from 1 gives the proportion of mutation. By dividing the proportion of mutations observed between two strains by the length of the branch connecting them in the phylogenetic tree, *ViralGeneClock* will roughly estimate the mutation rate.

$$\textbf{Estimated Mutation rate} = \frac{\textbf{(1 – Identity proportion)}}{\textbf{Branch Length}}$$

While this estimate won't provide an exact mutation rate for the gene, the ratio of mutation rates estimated for different genes can give insight into their relative mutation rates.

Project Timeline & Milestones

In order to keep up with the deadline, the timeline of the project is outlined below:

Milestone	Completion State / Deadline
1. Setting up Ubuntu sub-system in Windows device using Oracle Virtual Box.	Done.
2. Installing necessary Python packages, *Prokka* and *Muscle*.	Done.
3. Manually running and verifying the results from *Prokka* and *Muscle* pipelines on SARS-CoV-2 data.	Done.
4. Writing Python scripts to manually perform neighbor joining algorithm, and produce phylogenetic tree, branch length and identity matrix on SARS-CoV-2 data.	Done.
5. Writing Python scripts to automate *Prokka*, and prepare its results for multiple sequence alignment via *Muscle*.	Ongoing, Deadline: 3/10/2024
6. Writing Python script(s) to automate *Muscle*, and prepare its results for NJ.py (python script for neighbor-joining).	Deadline: 3/24/2024
7. Writing Python script(s) to estimate mutation rate for each gene based on the data from NJ.py.	Deadline: 3/24/2024
8. Consolidating all the Python scripts together, automating the entire process, and making *ViralGeneClock* a functional Linux tool.	Deadline: 4/7/2024
9. Examining the tool with WGS data of HIV and Influenza virus strains; making necessary changes to the model to improve its accuracy.	Deadline: 4/14/2024
10. Transforming *ViralGeneClock* into a web application using Flask, a Python web framework.	Deadline: 4/28/2024

User Experience Mock-up

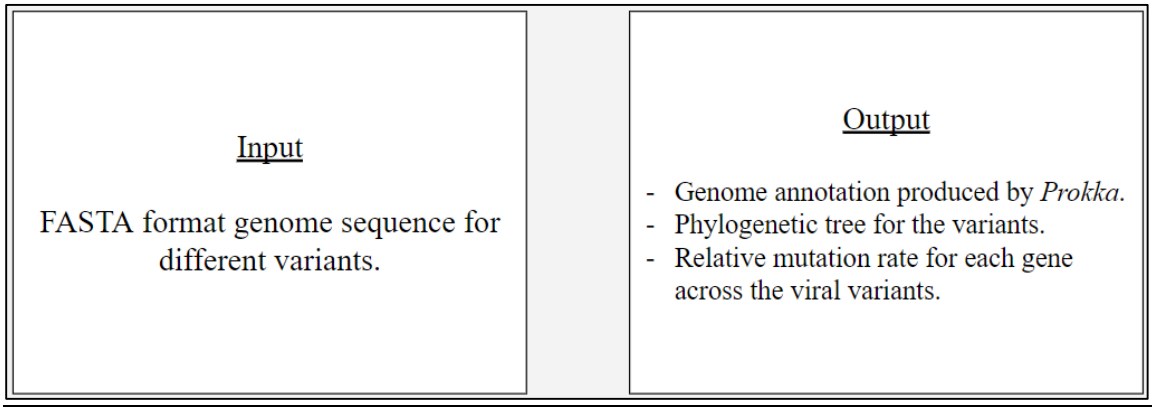

Figure 1: Input and Output for *ViralGeneClock*.

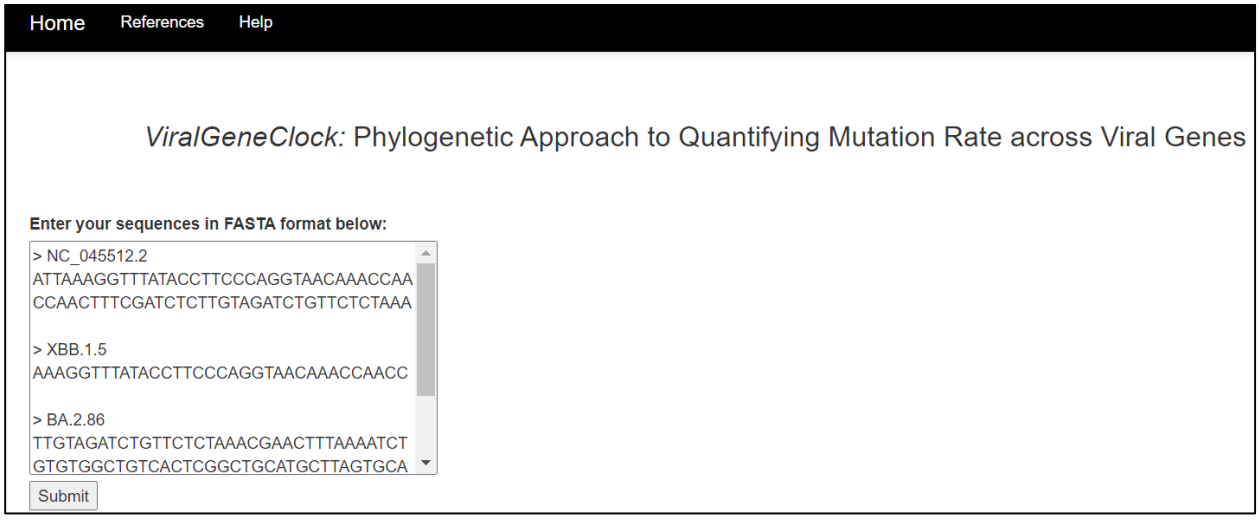

Figure 2: Sample Frontend for Receiving FASTA Input in *ViralGeneClock* Web App.

	Standard	Standard	Standard	Standard	Standard	Standard	Standard
1	locus_tag	ftype	length_bp	gene	EC_number	COG	product
2	KIEGKNJJ_00001	CDS	13209	1a			Replicase polyprotein 1a
3	KIEGKNJJ_00002	CDS	7788	rep			Replicase polyprotein 1ab
4	KIEGKNJJ_00003	CDS	3810	S			Spike glycoprotein
5	KIEGKNJJ_00004	CDS	828	3a			Protein 3a
6	KIEGKNJJ_00005	CDS	669	M			Membrane protein
7	KIEGKNJJ_00006	CDS	186				hypothetical protein
8	KIEGKNJJ_00007	CDS	366	7a			Protein 7a
9	KIEGKNJJ_00008	CDS	366				hypothetical protein
10	KIEGKNJJ_00009	CDS	1251	N			Nucleoprotein

Figure 3: Sample Output of *ViralGeneClock* (genome annotation of a SARS-CoV-2 strain produced by *Prokka*).

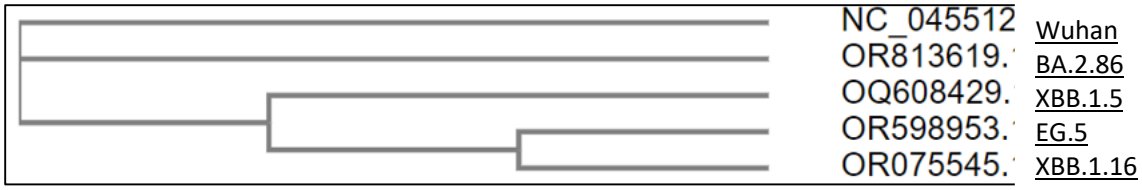

Figure 4: Sample Output of *ViralGeneClock* (phylogenetic tree of 5 SARS-CoV-2 strains produced by NJ).

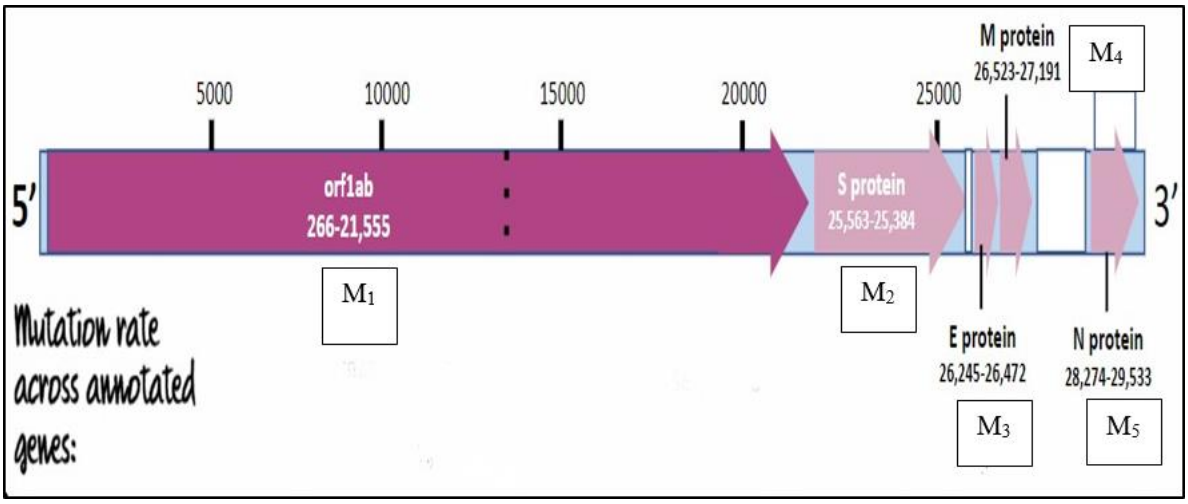

Figure 5: Sample Output of *ViralGeneClock* (estimated mutation rates across all the annotated genes).

References

1) Cardona-Ospina, J. A., Rojas-Gallardo, D. M., Garzón-Castaño, S. C., Jiménez-Posada, E. V., & Rodríguez-Morales, A. J. (2021). Phylodynamic analysis in the understanding of the current COVID-19 pandemic and its utility in vaccine and antiviral design and assessment. *Human Vaccines & Immunotherapeutics*, *17*(8), 2437–2444. https://doi.org/10.1080/21645515.2021.1880254

2) Sagulenko, P., Puller, V., & Neher, R. A. (2018). TreeTime: Maximum-likelihood phylodynamic analysis. *Virus Evolution*, *4*(1). https://doi.org/10.1093/ve/vex042

3) Hegde, S., Tang, Z., Zhao, J., & Wang, J. (2021). Inhibition of SARS-CoV-2 by targeting conserved viral RNA structures and sequences. *Frontiers in Chemistry*, *9*. https://doi.org/10.3389/fchem.2021.802766

4) Seemann, T. (2014). Prokka: rapid prokaryotic genome annotation. Bioinformatics (Oxford, England), 30(14), 2068–2069. https://doi.org/10.1093/bioinformatics/btu153

5) Edgar, R. C. (2004). MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, *32*(5), 1792–1797. https://doi.org/10.1093/nar/gkh340

6) Hyatt, D., Chen, G.-L., LoCascio, P. F., Land, M. L., Larimer, F. W., & Hauser, L. J. (2010). Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC Bioinformatics*, *11*(1). https://doi.org/10.1186/1471-2105-11-119

7) Saitou, N., & Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, *4*(4), 406–425. https://doi.org/10.1093/oxfordjournals.molbev.a040454

8) Kuhner, M. K., & Felsenstein, J. (1994). A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Molecular Biology and Evolution*, *11*(3), 459–468. https://doi.org/10.1093/oxfordjournals.molbev.a040126