# An Exploration of Information Loss in Transformer Embedding Spaces for Enhancing Predictive AI in Genomics

Plan B Paper by

Daniel Hintz

Advised by

Tim Robinson, Shaun Wulff, Alexandre Skiba, and Nicholas Chia

In Partial Fulfillment Of The Requirements

For

A Masters In Statistics

Department of Mathematics & Statistics

University of Wyoming

Laramie, WY USA

# Abstract

In this paper, we examine the quality of information preserved in the Genomic Neural Network Embedding algorithm, Genome-scale language model (GenSLM). We also set out to answer the question: to what degree can we retrieve information from GenSLM embeddings via intrinsic and extrinsic evaluation? By answering this question, we can provide bioinformaticians' a benchmark of how easily information can be extracted from GenSLM embeddings. In our analysis, we used intrinsic and extrinsic assessments of whole sequence and sub-sequence Covid DNA. For extrinsic evaluation, supervised learning for classification was performed as a proxy for the quality of information retrieved. For intrinsic evaluation, unsupervised learning tasks, such as redundancy, separability, and the relative preservation of the embedding space were performed to assess the quality of the data reduction. Results offer evidence that GenSLM achieves massive dimension reduction and successfully encodes genetic information into an embedding that allows for easy data retrieval. It also requires fewer resources, technology, or time compared to methods such as One-Hot encoding.

# Contents

# 1 Introduction

Neural network embeddings are widely used in machine learning for dimension reduction (Iuchi et al., 2021). Although the characteristics of neural network embedding algorithms have been extensively explored in natural language processing, their application in genomic data is less studied (Iuchi et al., 2021). Natural language inherently possesses structured and semantically meaningful elements like nouns and adjectives, which aid in interpretation, as well as evaluating the quality of embeddings representation. In contrast, DNA structures are less well-understood, which might explain why advancements in embedding algorithms initially flourished in natural language processing before their adaption to genomics.

Deoxyribonucleic acid (DNA) is a molecule that contains the genetic code that provides the instructions for building and maintaining life. The structure of DNA can be thought of as rungs on a ladder (known as base pairs) involving the pairing of four nucleotides - Adenine, Thymine, Cytosine, and Guanine. These nucleotides are commonly referenced as A's, T's, C's and G's, respectively, where A's bind with T's, and G's bind with C's. A popular representation of the DNA ladder is given in Figure 1. Genomic sequencing is a process used to decipher the genetic material found in an organism (i.e., reading the base letters that comprise the genome). Different life forms are characterized not only by different orderings of the four aforementioned nucleotides, but each lifeform sequence is of different lengths. For example, the SARS-CoV2 virus - the source pathogen[1] for the Covid pandemic - is approximately 30,0000 base pairs (bp's) in length, whereas the human genome is approximately 3 billion base pairs. Genomes considered in their entirety are referred to as whole genomes, where a whole genome is made up of the coding and non-coding region. The coding region specifically encodes proteins that are essential for all the functions necessary for life. Figure 2 depicts the whole SARS-CoV2 genome, marking structural and nonstructural proteins within the coding region along with the first 43 nucleotides for the nsp1 protein. While Figure 3 depicts how mRNA (A translation of DNA where T $\rightarrow$ U) is translated to amino acids. In this process, mRNA, which is synthesized from DNA by replacing thymine (T) with uracil (U), serves as the template for protein synthesis. Amino acids[2], which are assembled in a specific sequence dictated by the mRNA, are the fundamental building blocks of proteins.

[1] Now the the most studied DNA sequence in the world.

[2] Its important to note that many embedding algorithms embed amino acids instead of mRNA or DNA.

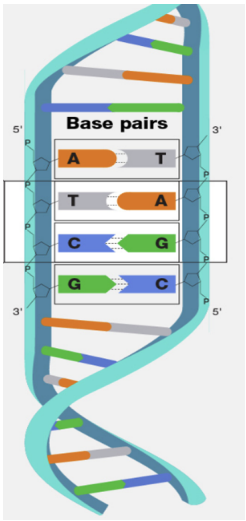


Figure 1: Base Pairs - Adapted from (National Human Genome Research Institute, 2024)

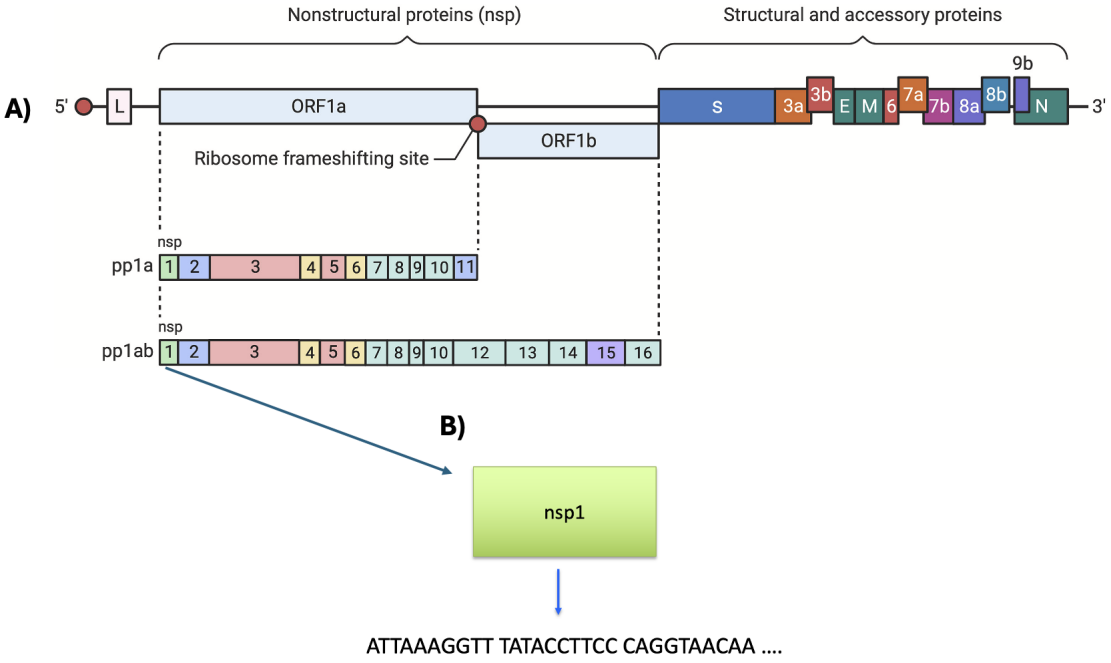


Figure 2: 2A. Whole Sars-CoV2 Genome with each Coding Region; 2B. Non-Structural Protein 1 (nsp1), and First 43 Nucleotides adapted from (Kandwal and Fayne, 2023, p. 99)

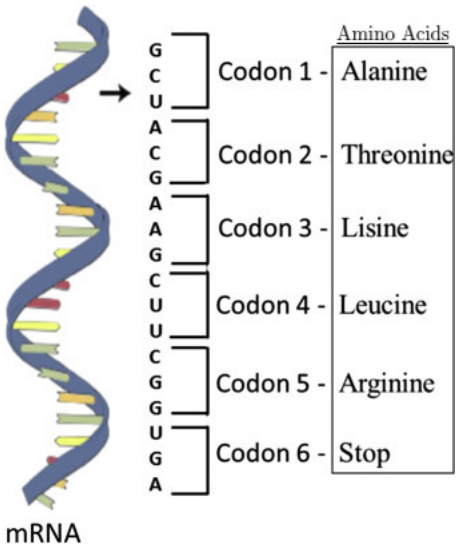


Figure 3: Sourced from (ScienceDirect, 2004)

The leap from embedding algorithms for natural language to embedding algorithms tailored to genomic data was first used with amino acids by Riis and Krogh (1996). Modern neural network embedding algorithms include DNABert, GENSLM, and HyenaDNA (Zhou et al., 2023; Zvyagin et al., 2023; Nguyen et al., 2024). Despite the use of these modern embedding algorithms, there has been little investigation into the redundancy, recoverability, and level of granularity associated with these algorithms that can be explained by the embedding space (Gharavi et al., 2021).

The effectiveness of any machine learning application[3] relies on the data representation, the objective function, and the optimization procedure (Liu and Sun, 2023). As noted by Iuchi et al. (2021), the structure of how the data is represented (e.g., sounds to images, DNA sequences to numeric matrices, DNA to binary encoding, etc.) is crucial as it impacts the performance of both Natural Language Processing (NLP) and deep learning applications (Iuchi et al., 2021, p. 3199). Therefore, it is vital for researchers working with genomic embeddings to understand the potential for information retention based on the embedding algorithms chosen for specific applications; this understanding is particularly crucial given the ongoing demand for dimension reduction to ease computational burdens in areas of transformer development[4].

This project explores the quality of embeddings through both *intrinsic* and *extrinsic* evaluations. Intrinsic evaluation focuses on the direct study of information preservation, whereas extrinsic evaluation assesses the quality of an embedding algorithm based on its performance in downstream tasks, such as regression and classification, across varying levels of task complexity.

Genomic sequences continuously change through mutations, and much of genomic research focuses on

[3] a Neural network is an example of a machine learning application

[4] such as transformer attention mechanisms. Where for example, GenSLM tokenizes sequences by codons, leading to a $10,000 \times 10,000$ attention block for Covid (30kb) and a 1B $\times$ 1B block for human DNA.

studying these variations Corso et al. (2021). To quantify the distance between two sequences, biologists use precise statistical models that calculate what is known as the edit distance (e.g., how different is the sequence, s1 = “AGTA" from the sequence, s2 = “AAGT"). The edit distance represents the smallest number of (weighted) insertions, deletions, or substitutions needed to convert one sequence, s1, into another, s2. There are a variety of edit distance metrics, such as the Hamming, Levenstein, and cosine distance.

It should be noted that edit distances are computationally expensive due to quadratic complexity and are difficult to parallelize. This makes using genomic sequences in their nucleotide (or amino acid) encodings a significant bottleneck in large-scale analyses Corso et al. (2021). Instead of comparing whole genome sequences, it would be more convenient to compare dimensionally reduced sequences, provided that the dimension maintains a large proportion of the information contained in the whole genome sequence. In response to this problem, embedding algorithms have exploded onto the scene for reducing the algorithmic complexity of working with whole genome sequences. By using embedding algorithms, the computation burden shifts to a preprocessing task and relieves computational complexity and redundancy of downstream tasks that can allow for more efficient pipelines.

The goal of this study is to provide a performance evaluation of the GenSLM embedding algorithm using extrinsic and intrinsic evaluation techniques. The extrinsic evaluation focuses on downstream tasks, while the intrinsic evaluation focuses on understanding the quality of information produced by the embedding algorithm. The extrinsic evaluation assesses downstream tasks from a range of difficulties. The extrinsic evaluation techniques are somewhat more straightforward than those techniques used in intrinsic evaluation. Applying intrinsic evaluation techniques to genetic DNA data poses more challenges compared to their application (and original use) in natural language processing (Lavrač et al., 2021; Liu and Sun, 2023; Iuchi et al., 2021). For context, the definition of intrinsic evaluation in the NLP literature is the “assessment of whether the similarities of the input entities (training examples) described in the original representation space are preserved in terms of the similarities of the transformed representations” (Lavrač et al., 2021, p. 11). Translating this to a genomics context, the distances between two sequences that are submitted to an embedding algorithm should have relativity the same distance in their embedded matrix representation. For supervised learning in the intrinsic case, classification is performed with labels protein and variant using classification and regression trees (CART), i.e., only a weak learner. For unsupervised learning in the intrinsic case, separability is assessed with Linear Discriminate Analysis (LDA), Principal Component Analysis (PCA), and t-Distributed Stochastic Neighbor Embedding (t-SNE) as well as Redundancy being assessed with PCA and Singular Value Decomposition (SVD). Notably, this paper marks the first extrinsic evaluation of GenSLM and possibly the first intrinsic evaluation of an embedding algorithm for genomic sequences.

### 1.1 Organization

This study will first expand upon the background of embedding algorithms for genomic sequences and their origins in the NLP domain. A brief background of SARS-COV-2 and GenSLM is provided to better understand the nature of tasks performed later in the study. This is followed by a summary of the data processing and exploration as a precursor to running the embedding algorithms and methods for intrinsic and extrinsic evaluation. The resulting findings from the intrinsic and extrinsic evaluation and related interpretations are then presented, followed by a discussion and conclusions.

## 2 Background

### 2.1 Embeddings and Embedding Algorithms

In natural language processing, words in the natural language are commonly represented by numeric vectors. For example, suppose $\vec{\text{queen}} = (0.3, 0.9)$, $\vec{\text{king}} = (0.5, 0.7)$, $\vec{\text{woman}} = (0.3, 0.4)$ and $\vec{\text{man}} = (0.5, 0.2)$. Note that $\vec{\text{king}} - \vec{\text{man}} + \vec{\text{woman}} = \vec{\text{queen}}$, i.e., we have the result $(0.5, 0.7) - (0.5, 0.2) + (0.3, 0.4) = (0.3, 0.9)$. Similarly, in genomics, specific nucleotides of DNA - 'A', 'G', 'T', 'C' - can be represented by numeric vectors. These vectors are often referred to as 'embeddings,' which is a term originally used in mathematics to describe the process of mapping from one space to another. However, in both computational biology and language processing, the term 'embeddings' has now more commonly come to mean the dense vectors produced by specific algorithms, such as GenSLM, hyenaDNA, or DNABERT2. To improve clarity, it is useful to differentiate between 'embedding algorithms' which refer to the methods generating these vectors, and 'embeddings'[5] themselves, which are the resultant dense vectors also known as 'Distributed Representations' or 'Latent Spaces' (Liu and Sun, 2023).

### 2.2 Natural Language

Genomic sequences resemble natural language in that characters are used to define their meaning, and that meaning depends on their location relative to other sequences (Iuchi et al., 2021). For instance, the interpretation of the word "bank" as either a financial entity or a landform beside a body of water hinges on its surrounding context (Liu and Sun, 2023, p. 47). In an analogous manner, the formation and structure of a segment of RNA is influenced by the sequences adjacent to it (Iuchi et al., 2021). Therefore, given the parallels between natural language and genomic sequences, utilizing natural language processing (NLP)

[5] Although, the term "embeddings" is sometimes more strictly applied only to dense vectors like word2vec, DNABERT2, and GenSLM, rather than sparse tf-idf or PPMI vectors (Jurafsky and Martin, 2019).

techniques can offer insight into the functional and structural information embodied in genomic sequences (Iuchi et al., 2021). An important characteristic of embeddings in NLP is that vector distance is also a measure of semantic similarity, as already demonstrated with $\vec{\text{king}} - \vec{\text{man}} + \vec{\text{woman}} = \vec{\text{queen}}$, see Figure 4.

Vector distance is equally crucial in genomic embedding algorithms and is generally determined using linear algebra techniques, including the dot product, Euclidean distance, and cosine similarity (Iuchi et al., 2021). As an illustration, consider a list containing information on organisms as well as food. Then, using an embedding algorithm, a 2 dimensional (x, y) embedding is obtained, as seen in Figure 5. In this simple example, it is expected that the vector similarity for Cows would be closer to frogs than Sushi or Pizza ,and similarly, it would be expected that the vector similarity of Sushi would be closer to Pizza than Cows or Frogs. Likewise, in genomic sequences, the $\alpha$-amino-3-hydroxy-5-methyl4-isoxazolepropionic acid receptor and the N-methyl-D-aspartate receptor(which are both ionotropic glutamate receptors), would have a close vector similarity (Iuchi et al., 2021, p. 3199). Hence, it is hoped that the embeddings and the original sequences possess similar function and structures so that there should be high vector similarity between proteins (Iuchi et al., 2021, p. 3199).

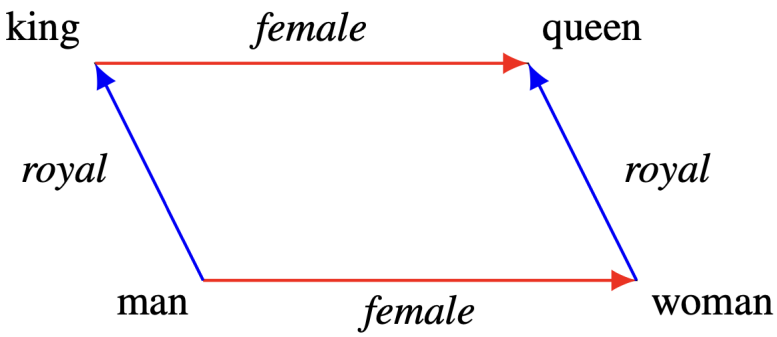


Figure 4: Sourced from (Ethayarajh et al., 2018, p. 4)

## 2.3 Sequence Representations

Authors of highly cited textbook *Representation Learning and NLP*[6], Liu and Sun (2023), provide a framework that underscores the importance of high-quality representations (embeddings) in successful machine learning models from an NLP perspective, see Equation 1 (Liu and Sun, 2023).

$$\text{Machine Learning} = \text{Representation}^{7} + \text{Objective} + \text{Optimization} \tag{1}$$

[6] A canonical source for Embedding in NLP applications.
[7] Where the representation can be seen as a feature engineering task.

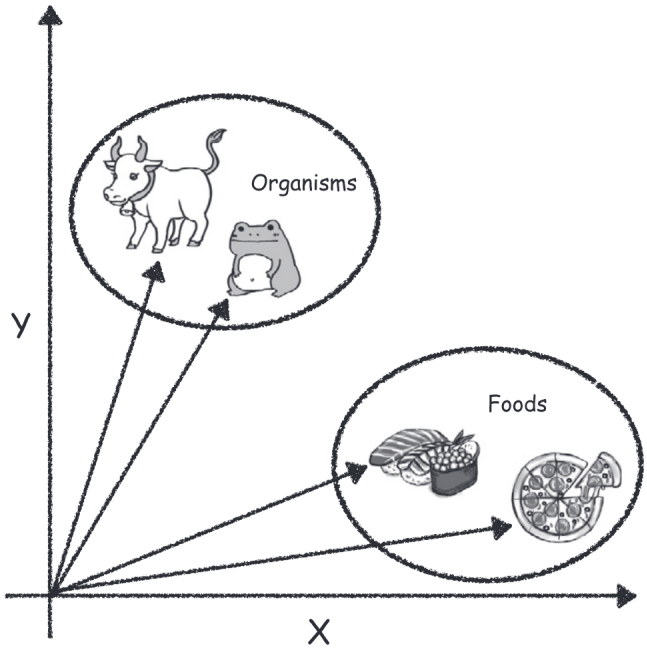


Figure 5: Sourced from (Iuchi et al., 2021, p. 3199)

It's from equation 1, we can begin to understand that an effective representation (embedding) of a genomic sequence is crucial for clustering, classification, regression, protein function identification, structural analysis, and predicting genetic disorders (Angermueller et al., 2016). In a study by Jing et al. (2019), they found differences of up to 27.51% in the classification rate of Protein Secondary Structure between different embedding algorithms with a Random Forest learner (Jing et al., 2019, p. 1927). Clearly, the representation method, or embedding algorithm, is critical for successful and accurate protein classification. For context, the margin between 1st and 20th place in Kaggles Stanford Ribonanza RNA Folding Competition with $100,000 in prize money is a difference in mean absolute error (MAE) of only 0.00598 (Kaggle, 2023).

## 2.4 SARS-COV-2

As already discussed, Coronaviruses are enveloped RNA viruses that possess a positive-sense single-stranded genome and are part of the Coronaviridae family. They are divided into four main subgroups: Alphacoronavirus, Betacoronavirus, Gammacoronavirus, and Deltacoronavirus, with SARS-CoV-2 falling under the Betacoronavirus genus. The single-stranded RNA genome of coronaviruses contains 29,891 nucleotides which translates into 9890 amino acids. The genome of SARS-CoV-2 includes several open reading frames (ORFs) that code for both structural (SP) and non-structural proteins (NSP) which play crucial roles in the virus life cycle and its pathogenic mechanisms, see Figure 2. Coronaviruses have structural proteins, such as the S (spike), E (envelope), M (membrane), and N (nucleocapsid) proteins that construct the virus's

physical structure, and sixteen non-structural proteins (nsp1-16) that are involved in the virus metabolism and interactions with the immune system of the host. Like many viruses, coronaviruses can mutate during replication. These mutations may lead to new pathogenic variants that could alter the virus's transmission, impact the severity of the disease, and influence the effectiveness of vaccines (Farhud and Mojahed, 2022).

By nature, DNA contains highly redundant information. The sequence similarity across SARS-CoV-2 genomes is generally high, typically greater than 99%, with only a small number of changes that yield distinct phenotypes (Zvyagin et al., 2023). For example, as few as 22 mutations can identify a variant as unique from the progenitor of SARS-COV-2, as in the case of Gamma Variant (Farhud and Mojahed, 2022). This is contextually crucial information as redundancy is a centerpiece for evaluating the quality of the embedding algorithm and it's associated output, see Section 4.

## 2.5 GenSLM

GenSLM is a hierarchical transformer-based model that leverages both Generative Pre-trained Transformers (GPT) for individual gene sequences and stable diffusion techniques. An important concept of GenSLM, and all embedding algorithms, is the idea of a "tokenizer". In the popular NLP algorithm BERT, words are treated as 'tokens' rather than whole units, and the tokenizer is simply the name for the function that breaks up words into sub-units. In GenSLM, the input sequence is tokenized into codons, which are chunks of 3 nucleotides at a time: "AAG", "TAG", etc. The output dimension of the embedding is n×512, where n represents the number of input sequences. Each element of the 512 elements should roughly correspond to an open reading frame (ORF) where Covid has 20 ORF; see the 20 ORFs for image A) depicting coding region in Figure 2. Hence, for a Covid sequence with a length of 30,720 base pairs (bp), there are $30,720/3 = 10,240$ codons ("tokens") from 20 open reading frames, which provides us with 512 elements (Eq. 2). However, GenSLM elements are not meaningfully interpretable, and the dimensions of GenSLM vectors are simply the product of the scaling to be specific to the Covid sequences, as shown in Equation 2 below:

$$\begin{aligned} \text{Number of GenSLM Elements} &\equiv \frac{\text{Number of Codons for a 30720 bp Covid sequence}}{\text{Number of Covid ORF's}} \\ &\equiv \frac{10240}{20} = 512 \end{aligned} \tag{2}$$

The core engine of GenSLM includes transformers and diffusion models. These two types of models are used in tandem to create an understanding of the local (codon level) and global (sequence level) context needed for genetic analyses. The first step in the GenSLM algorithm is to pass the input sequence to its

transformer encoder that converts the bp character representation into numeric vectors before recursively running the transformer encoder through a diffusion model to learn a condensed distribution of the whole sequence. The transformers in GenSLM are used to capture local interactions within a genomic sequence, whereas the diffusion model integrates information across much larger segments of the genome that can essentially capture information that could be missed at the transformer level. The integration of transformer models and diffusion models is hierarchical by using the transformer for finer details and the diffusion model for broader context. This hierarchical integration allows GenSLM to analyze and predict potential variants in the genome.

The foundation model for GenSLM used in this paper is the 25 million parameter model from Globus as specified from the GenSLM Github landing page ramanathanlab/genslm. An important caveat of the GenSLM-25M model is that it was specifically trained only on the first year of the SARS-CoV-2 data consisting of $\sim$ 85,000 SARS-CoV-2 genome sequences, as well as using sequences aligned to the now outdated reference NC_045512.1 (Zvyagin et al., 2023). Thus, the model did not have the opportunity to see any of strains beyond March 2021 (Zvyagin et al., 2023).

# 3 Data and Processing

This section is organized as follows: in Section 3.1 the source for the 581 sequences upon which this report is based is described. Section 3.2 explains the data cleaning (exclusion criteria) and pre-procesing (multiple sequence alignment). In Section 3.3 data structure is explored within the realm of whole genome sequences across multiple SARS-CoV2 variants for the purpose of comparison with their embedding equivalents, as described in Section 5. In Section 3.4, the workflow of downloading the data, to generating the embeddings is presented.

## 3.1 Data Description

All of the Covid sequences used in this paper were taken with permission from the Global Initiative on Sharing All Influenza Data (GISAID) via the API GISAIDR (Wirth and GISAIDR, 2022). No geographical or temporal restrictions were placed on the sequences extracted and instead were only filtered for sequence quality. However, extensive post-processing was performed above and beyond the GISAID's exclusion criterion for Covid sequences.

## 3.2 Data Cleaning & Pre-Processing

In this study, the process of preparing genomic data involved several meticulous steps to ensure the accuracy and utility of the data for embedding and analysis. To achieve this, automated quality filters available through the GISAID API were employed. Only sequences marked as “Complete" and having “High coverage" were included in the dataset. GISAID classifies genomes longer than 29,000 nucleotides as complete, aligning well with the reference genome for SARS-CoV-2, which is 29,674 base pairs long. The “High coverage" filter restricts sequences to those with less than 1% undefined bases (denoted as ‘N’), which indicates a high repetition in the reading of each nucleotide and thereby enhances the accuracy of the genomic sequence. Conversely, sequences with more than 5% undefined bases are considered to have low coverage, indicating poorer quality due to insufficient sequencing depth, which could lead to significant uncertainties in the genomic data. This exclusion criterion was supplemented with additional post-processing to exclude sequences with ‘N’ ambiguous nucleotides and gap length. The exclusion criteria for Gap length was executed by removing sequences with gap lengths in proteins twice as large as a mode gap length. Together, the GISAID’s exclusion criterion and additional post-processing ensured that only high-quality, reliable DNA sequences were used for subsequent analysis.

Protein level analysis focused on accurately segmenting the SARS-CoV-2 proteins before passing protein sequences to GenSLM to be embedded. This required a Multiple Sequence Alignment (MSA), for which the Clustal Omega algorithm was used from the msa package in R (Sievers et al., 2011; Bodenhofer et al., 2015). To obtain the locations of proteins in terms of base pairs, the Wuhan reference sequence (NC_045512.2), was included in the alignment for which protein locations are documented (NCBI Reference Sequence, 2023; Bai et al., 2022, p. 283). By exploiting the known locations of the reference sequences, as well as string matching unaligned reference sequences to the aligned reference sequence, the updated protein locations were obtained for later slicing of whole sequence genomes into the respective proteins. Through these careful preparatory steps involving the GISAID API for data extraction, quality control, and multiple sequence alignment, precise and meaningful data for subsequent analysis could be obtained.

## 3.3 Exploratory Data Analysis

The sequences used in all the subsequent analyses had a sequence length around what would be expected for high coverage Covid sequences ($\sim$ 29000 bp). Moreover, after performing the multiple sequence alignment, a Radial Dendrogram was constructed by first calculating the identity pairwise distances (IPD). This distance is a square root dissimilarity measure of the proportion of bases matching between two sequences (Charif and Lobry, 2024). For example, consider two DNA sequences, Sequence 1: “ATCG", and Sequence 2: “ATCA".

To calculate the identity pairwise distance between these sequences, you first determine how many positions are identical. Here, the first three nucleotides (ATC) are identical, and the fourth nucleotide differs. So, three out of four nucleotides are identical, making the proportion of bases matched 0.75. The dissimilarity is then $1.0 - 0.75 = 0.25$. The identity pairwise distance between sequences 1 and 2 is obtained as the square root of the dissimilarity: $\sqrt{0.25} = 0.5$. This value quantifies the difference based on mismatched nucleotides, considering gaps if specified. There are no gaps in this simple example. After obtaining the identity pairwise distances, an agglomerative clustering with complete linkage is performed based on the pairwise distances. Finally, the Radial Dendrogram in Figure 7 was generated with the R function `ape::plot.phylo` (Paradis et al., 2024). This dendrogram, which accounted for gaps, illustrates moderately good clustering of Covid variants, albeit with the somewhat strange placement of the Wuhan reference genome relative to the scientific census for very large samples of the Covid sequence (Nextstrain, 2022). This feature is likely not a failure on the part of agglomerate clustering algorithm but the sample of sequences used.

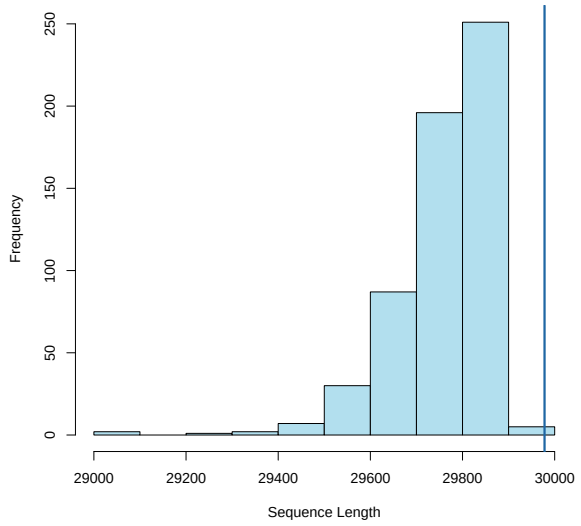


Figure 6: Sequence length across 581 subjects (Line Marks Sequence Length Post-Alignment)

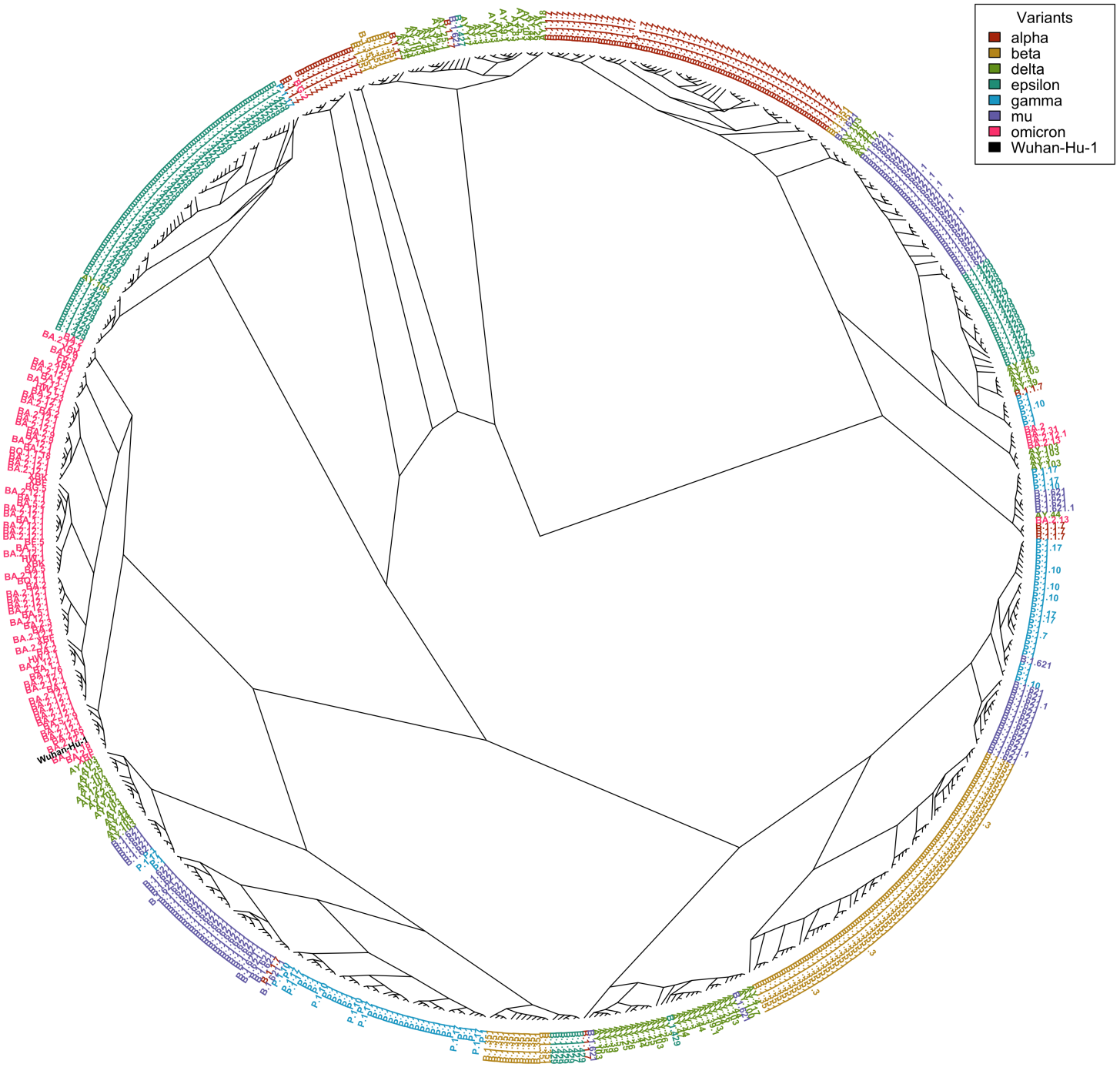


Figure 7: Radial Dendrogram for 581 Aligned Whole Genome Covid Sequences with Wuhan-Hu-1 Reference
- See Figure 12 for Comparison -

The placement of the Wuhan sequence for the dendrogram in Figure 7 is merely a product of the structure of the data, as shown in Table 1 and Figure 8. This suggests the sample of 581 Covid sequences used may not be reflective of the Covid variants overall.

Table 1: Mean Gap Sizes, Mutations, and Total Changes for COVID-19 Variants Aligned to Reference.

| **Variant** | **Mean Gap Size** | **Mean Mutation Count** | **Total Changes** |
|---|---|---|---|
| Alpha | 324 | 294.66 | 618.66 |
| Beta | 170 | 146.35 | 316.35 |
| Delta | 203 | 167.31 | 370.31 |
| Epsilon | 191 | 166.35 | 357.35 |
| Gamma | 178 | 136.69 | 314.69 |
| Mu | 219 | 163.12 | 382.12 |
| Omicron | 185 | 62.26 | 247.26 |

*Note: Mutation counts are based on Hamming's Distance calculations.*

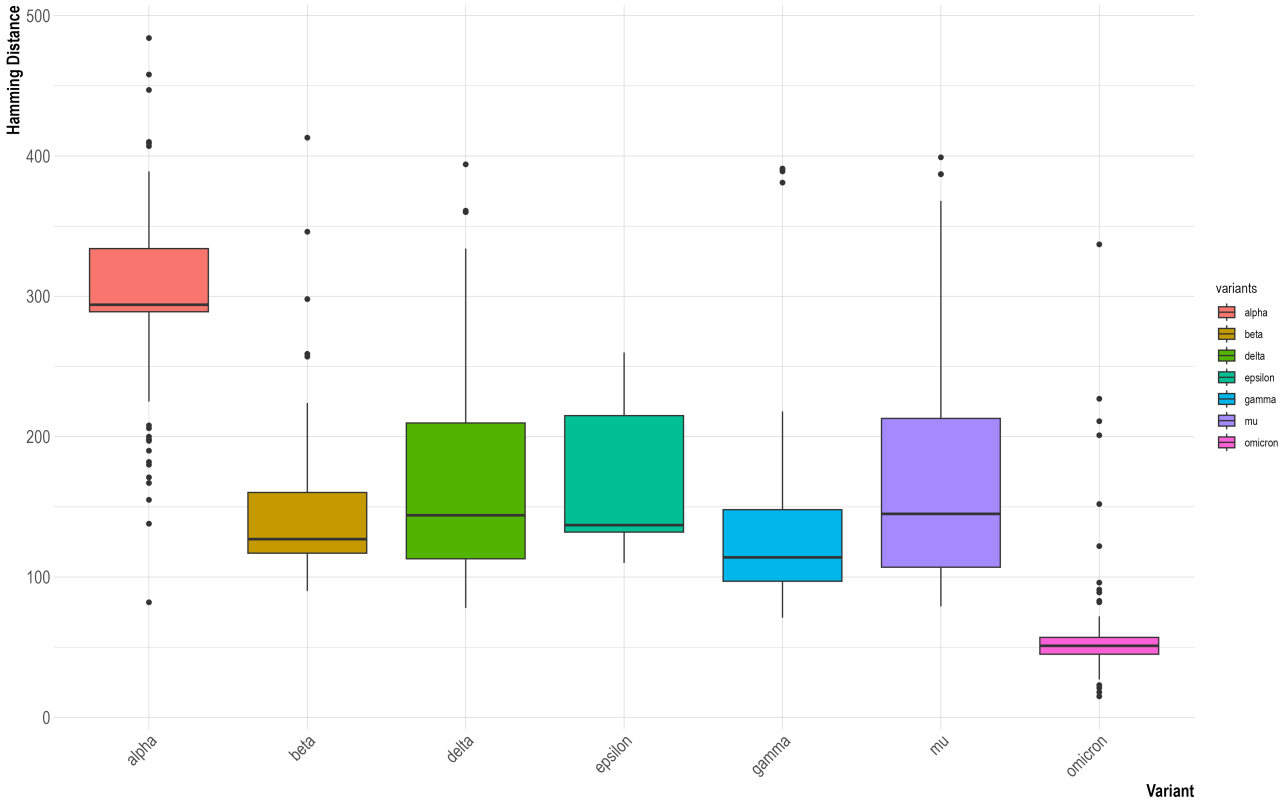


Figure 8: Whole Genome Mutation Count (Hamming Distances) from Wuhan Reference

In addition, the Hamming distance was calculated per protein sequence from the aligned Wuhan reference sequence, which indicates the number of mutations from the progenitor virus, as seen in Figure 9. It can also be noted from Figure 9, that the Spike protein (S) and M have the most mutations, colored in red to purple, while most other proteins, colored in mostly yellow, have very few mutations.

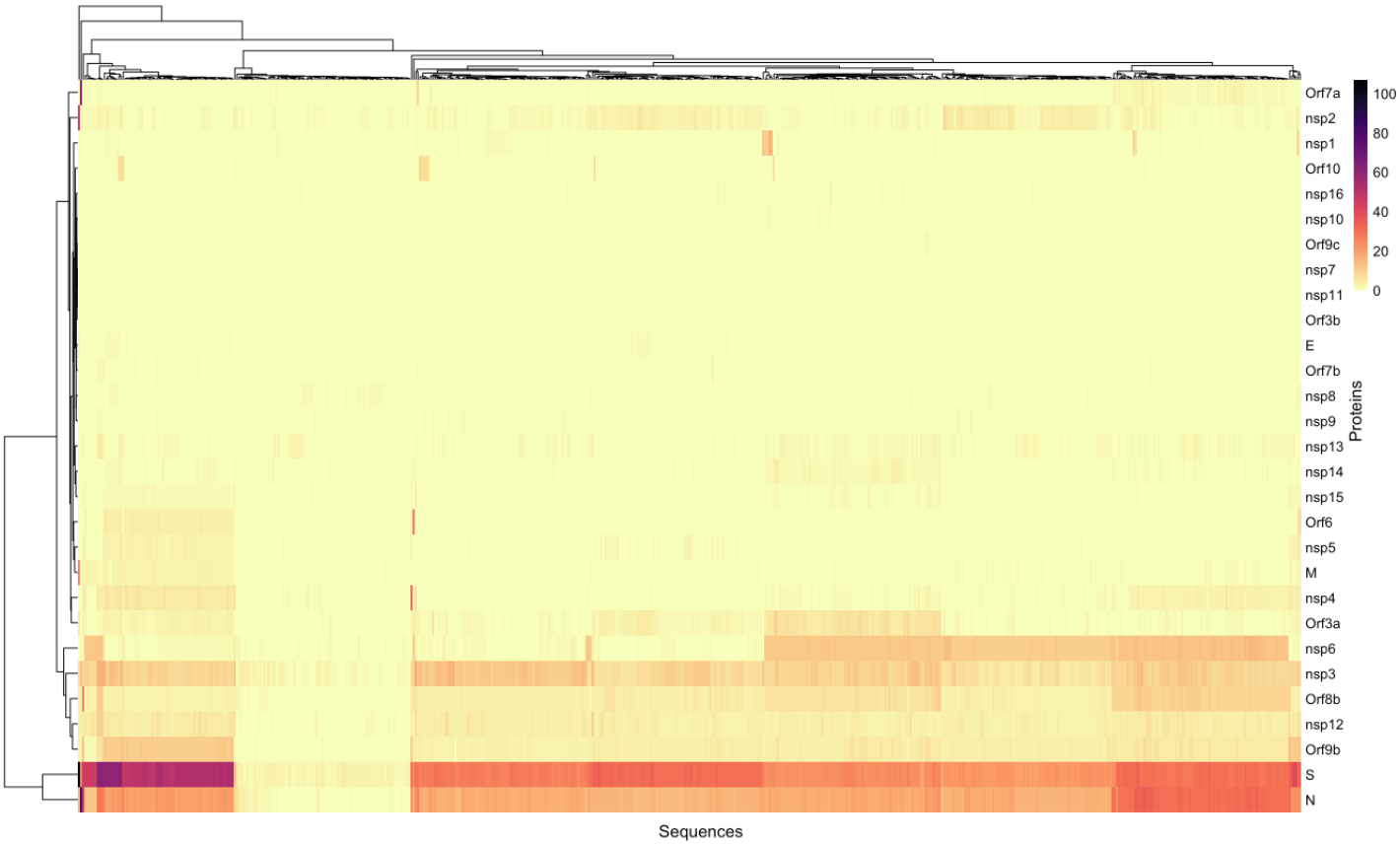


Figure 9: Hamming Distance of Aligned Proteins to Wuhan Reference Sequence

## 3.4 Generating Embeddings

For embedding the Covid DNA sequences variant by variant as well as protein by protein, a combination of bash pipelines were created and executed with SLURM Jobs Arrays on a high-performance computing (HPC) Linux environment, see Figure 25 in the appendix. For the interested reader, one can inspect the code used for executing the embedding in Scripts 1, and 2 for pre-processing, and Scripts 3, 4 and 5 in the appendix. These pipelines required computing resources greater than what is typically available for a standard-issue laptop. Initially, even in an HPC environment, computing jobs could take over twelve hours to complete and occasionally exceed 20 GB in memory utilization. By editing scrips to run as job arrays, computational times were expedited and often completed within six hours of execution. Note that the environment file for recreating the conda environment used for embedding can be found in the following

file `environment.yml`.

Passing sequences through the embedding pipeline, specifically at step 3, is part of a broader workflow depicted in Figure 10, which typically spans a week to complete. The workflow begins with downloading sequences using the GISAID API[8], spanning a half-day process for Step 1. Step 2 involves multiple sequence alignment, which could take up to a week, followed by step 3, which would take about six hours for embedding. The final step before evaluation, Step 4, aggregates the results in less than an hour. This extensive process generates numerous files: the original GISAID data in a `.feather`[9] file, a `.rds` file from the alignment, another `.feather` file after post-processing alignments, several `.h5`[10] sequence and embedding files, and a `.feather` file for the aggregated embeddings across all variants and proteins. Note that all of the code for producing the results in this paper can be found at the Github repository `DHintz137/GenSLM_Embedding`.

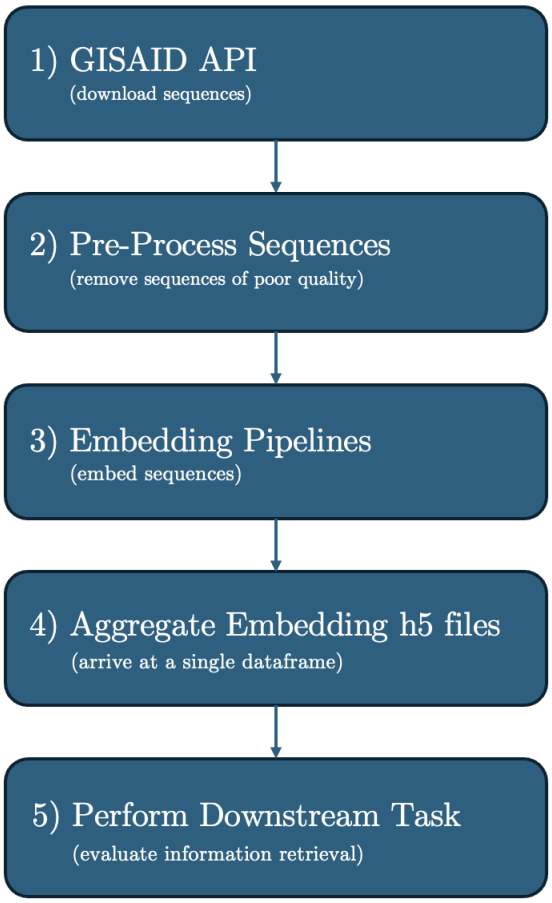


Figure 10: General Workflow

[8] Most of this time was due to the sleep times introduced to account for API rate limiting so as to avoid temporally losing access to the API.

[9] A `.feather` file is a binary columnar data storage format designed to efficiently store and access data frames.

[10] `.h5` files are a highly efficient hierarchical data format associated with HDF5 (Hierarchical Data Format version 5) and are commonly used in bioinformatics.

# 4 Methodology

This study aimed to assess the quality of the GenSLM embedding algorithms when applied to genomic analyses. The quality of an embedding is assessed based upon its information richness and the degree of non-redundancy. Information richness is greater when there is higher learner performance for downstream tasks. Non-redundancy is gauged by how much the information can still be compressed without being dimensionally redundant (Jing et al., 2019). When downstream learning tasks are evaluated, it's often referred to as extrinsic evaluation. While when the qualities of the embedding matrix itself are assessed it is often referred to as intrinsic evaluation (Lavrač et al., 2021).

There are various methods for conducting intrinsic and extrinsic evaluations of embeddings. The subsections below detail these methods. The intrinsic evaluation (Section 4.1) focuses on three attributes: redundancy, separability, and the preservation of semantic distance. Section 4.1.1 examines redundancy using Singular Value Decomposition (SVD), and also explores redundancy through a classification task on dimensionally reduced data (Section 5.2.2). The preservation of semantic distance is assessed in Section 4.1.2 using Kullback–Leibler Divergence. Separability is investigated in Section 4.1.3 through LDA, PCA, and t-SNE techniques. For extrinsic evaluation (Section 4.2), CART is employed to classify variants and proteins.

## 4.1 Intrinsic Evaluation

For intrinsic evaluation, three methods were used, each providing unique insight. Studying an embedding's redundancy reveals the efficiency of its encoding; more efficient encodings tend to perform better and use fewer computational resources. Separability gives practical insight into whether or not the embedding output can be separated into meaningful genomic groups (i.e., variants). Meanwhile, exploring the preservation of semantic distance is important for determining if the structural representation of information remains valid for subsequent tasks.

Intrinsic analysis of embeddings presents several challenges, primarily related to measuring semantic distance and validating the interpretations drawn from these measurements. For instance, distances in latent spaces—the vector spaces formed by dimension-reducing embedding algorithms—lack physical units, complicating their interpretation. Additionally, latent spaces are influenced by the architectural details of the specific neural networks (i.e., GenSLM) used to produce them (Arvanitidis et al., 2017). Moreover, the distance calculations used to determine differences between latent spaces are more complex in neural networks because the outputted latent space is non-linear (Arvanitidis et al., 2017). This fact brings significant

difficulties in selecting a distance metric that maps between both a linear and non-linear space. Inasmuch, the investigation of the semantic distance from the original representation to the embedding space may, at times, fail as a result of the metric and not the embedding algorithm chosen[11]. The following sections will provide a detailed methodology for each of these intrinsic evaluation methods.

### 4.1.1 Redundancy

In this paper, redundancy will be explored through two methodologies: Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). Detailed descriptions and results of SVD are presented in Sections 4.1.1.1 and 5.1.1.1, respectively. Similarly, PCA is discussed in Sections 4.1.3.2 and illustrated in Figure 23.

#### 4.1.1.1 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a technique used to analyze redundancy by measuring the cumulative proportion explained (CPE), where CPE is a measure of how much variance in the original data captured by the singular values, $\hat{\Sigma}_i$. Moreover, SVD decomposes any complex-valued matrix $\mathbf{X}$ of dimensions $n \times m$ into a product of three matrices, as shown below:

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^* \tag{3}$$

Here, $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices from the sets $\mathbb{C}^{n\times n}$ and $\mathbb{C}^{m\times m}$ respectively, each having orthonormal columns. The matrix $\mathbf{\Sigma}$, which lies in $\mathbb{R}^{n\times m}$, contains real, non-negative diagonal entries and zeros elsewhere. In this context, the asterisk notation ($^*$) represents the Complex Conjugate Transpose, otherwise known as the Hermitian Transpose[12].

For cases where $n \geq m$, the diagonal of $\mathbf{\Sigma}$ holds at most $m$ non-zero elements, and can be expressed as:

$$\mathbf{\Sigma} = \begin{bmatrix} \hat{\mathbf{\Sigma}} \\ \mathbf{0} \end{bmatrix} \tag{4}$$

Consequently, $\mathbf{X}$ can be precisely reconstructed using the economy SVD[13]

[11] These challenges can often easily be overcome in NLP as the embedding map to a vector for each word or sub-word unit.

[12] The Hermitian Transpose can be computed in `R` using `t(Conj(X))`.

[13] Note: $\hat{\mathbf{U}}^{\perp}$ is the orthogonal complement of the subspace spanned by $\hat{\mathbf{U}}$, i.e., it is the set of all vectors in the encompassing space (that includes $\hat{\mathbf{U}}$ and more) that are orthogonal to every vector in $\hat{\mathbf{U}}$.

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* = \begin{bmatrix} \hat{\mathbf{U}} & \hat{\mathbf{U}}^{\perp} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{\Sigma}} \\ \mathbf{0} \end{bmatrix} \mathbf{V}^* = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\mathbf{V}^* \tag{5}$$

This reduced form still completely captures the matrix $\mathbf{X}$ while utilizing a simplified structure. This simplified structure is used in calculating the cumulative proportion explained; calculated by the cumulative sum of the singular values squared, $\text{cumsum}(\text{diag}(\hat{\Sigma}_i^2))$.

### 4.1.2 Preservation of Semantic Distance

Embedding algorithms, as data transformation tools, modify the coordinate system of the output space. Although neural network embedding algorithms do not constitute invariant transformations, they are expected to preserve the semantic distances inherent in the data (Lavrač et al., 2021). For instance, it is anticipated that the cosine distance between sequences from the same variant would be smaller compared to distances between different variants. This expectation mirrors the Word Analogy task in natural language processing, as discussed by (Lavrač et al., 2021). For measuring sequence similarity in their DNA character encodings, the Discrete Kullback–Leibler (DKL) divergence is appropriate, considering that counts of nucleotide bases (“A", “G", “T" and “C") are discrete. Conversely, for embeddings, the Continuous Kullback–Leibler (CKL) divergence can be employed since embeddings exist within a continuous vector space. This paper employs both Discrete Kullback-Leibler (DKL) and Continuous Kullback-Leibler (CKL) divergence to determine if the distortions from the GenSLM transformation are minimal enough to deem the embeddings reliable for bioinformatics pipelines

#### 4.1.2.1 Continuous Kullback–Leibler Divergence

The Continuous Kullback–Leibler divergence (CKL), also known as Continuous Relative Entropy or I-divergence, is a non-symmetric measure[14] of divergence between two probability distribution functions (PDF's). Where statistical divergence is defined as the “degree of separation of two points $f$ and $g$, but it or its square root is not a distance" (Amari, 2016, p. 10). This is because it does not necessarily satisfy the symmetry condition, so that in general:

$$D_{KL}(f\|g) \neq D_{KL}(g\|f). \tag{6}$$

[14] In the sense that it is not technically a distance or metric, as it doesn't satisfy the triangular inequality.

However, a convenience of CKL is that it acts - in accordance with the geometric properties of PDF's - as the same role as squared Euclidean distance (Csiszar, 1975). Hence, the relative entropy $D(f\|g)$[15] of two PDFs, $f$ and $g$, defined as:

$$D_{KL}(f\|g) = \int_S f(x) \log \frac{f(x)}{g(x)} dx \tag{7}$$

where $S$ is the support set of $f$[16]. It should be noted that the Continuous KL Divergence was approximated using a normal distribution, based on the mean and standard deviation of the embedding vectors. This approach facilitated the integration of CKL values, where the assumption of normality is reasonably satisfied, as illustrated in Figure 24.

#### 4.1.2.2 Discrete Kullback–Leibler Divergence

Meanwhile, the Discrete Kullback–Leibler (DKL) divergence is defined as the following:

$$D_{KL}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{8}$$

where $P_i$ represents the frequency of the $i^{\text{th}}$ nucleotide in a specific genome $X$, and $Q_i$ denotes the average frequency of this nucleotide determined from all complete genomes (Akhter et al., 2017). In the discrete case, KL divergence is the statistical divergence between two DNA sequences treated as discrete PDF's.

#### 4.1.2.3 Distance Matrices

Distance matrices offer another method for evaluating the preservation of semantic distance. However, before constructing a distance matrix, an appropriate distance metric must be chosen. For DNA sequences, the identity pairwise distance is suitable due to its simplicity and interpretability. There are also efficient, open-source tools available for this calculation, such as the R function `seqinr::dist.alignment`, which was previously explained in the Section 3.3. For the embeddings, Euclidean distance is selected for its speed and simplicity despite the potential issue that the latent space may not be Euclidean. This calculation is performed using the `stats::dist` function in R.

[15] where $\|$ denotes $f$ is relative to $g$.
[16] Note that $D(f\|g) = 0$ if $\text{supp}(g) \notin \text{supp}(f)$.

### 4.1.3 Separability

In the context of bioinformatics, the separability of embedding outputs into meaningful genomic groups, such as variants, is crucial for effective analysis and interpretation. The ability to distinguish these groups within the embedding space indicates that the transformation algorithms have maintained critical biological distinctions. This aspect of data analysis ensures that subsequent analyses, such as clustering or classification, are grounded in biologically relevant differences rather than artifacts of the data transformation process. In this section, we explore various statistical and machine learning methods to evaluate the separability of genomic groups in the embedding space, providing a check for robust downstream analyses.

#### 4.1.3.1 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a statistical technique used for classification and discrimination of observations into predefined groups based on multivariate data. The primary goal of LDA is to find rules that best separate known groups (denoted as $G_1, G_2, \ldots, G_g$) and optimally allocate observations to these groups. This involves two main components: Discriminant Analysis (DA), which develops rules that effectively distinguish between different groups, and Classification Analysis (CA), which assigns new observations to one of the predefined groups based on these rules.

The decision to assign an observation $y_i$ to a group $k$ is based on either minimizing the distance between the observation and the group centroid or maximizing the likelihood that the observation belongs to group $k$. Assuming that the data for each group follows a multivariate normal distribution, the probability density function (PDF) for an observation $y_i$ belonging to group $k$ is given by:

$$f_k(y_i) = (2\pi)^{-\frac{p}{2}} |\Sigma_k|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(y_i - \mu_k)^T \Sigma_k^{-1} (y_i - \mu_k)\right] \tag{9}$$

If we assume equal covariance matrices across groups ($\Sigma_1 = \cdots = \Sigma_g = \Sigma$), then the equation simplifies, and we replace $\Sigma_k$ with a pooled estimate $\widehat{\Sigma} = S_p$. The probability of group $k$ given observation $y_i$ (posterior probability) is calculated using Bayes' rule:

$$P(G_k \mid y_i) = \frac{\widehat{f}_k(y_i) P(G_k)}{\sum_{h=1}^{g} \widehat{f}_h(y_i) P(G_h)} \tag{10}$$

where $P(G_k)$ is the prior probability of group $k$. The observation is then assigned to the group that maximizes this posterior probability. Under the assumptions of multivariate normality, equal covariance matrices, and equal priors, the optimal allocation rule becomes linear, known as Fisher's Linear Discriminant Function (LDF). Fisher's LDF seeks a linear combination of variables that maximizes the ratio of the between-

group variance to the within-group variance, thereby maximizing the separation between groups. This is achieved by finding coefficients $\underline{a}_{(j)}$ that satisfy:

$$E^{-1}H\underline{a}_{(j)} = \lambda_{(j)}\underline{a}_{(j)} \tag{11}$$

where $E$ is the within-group sum of squares and cross-product matrix, and $H$ is the between-group equivalent. LDA is being utilized in this paper for its utility in forcing a separation of groups, as illustrated by Figure 14.

#### 4.1.3.2 Principal Component Analysis

Principal Component Analysis (PCA) is a dimension reduction technique used to simplify a dataset with many variables into fewer principal components that still capture the essential variability in the data. These components are linear combinations of the original variables, formulated to maximize the captured variance sequentially, ensuring that each component is uncorrelated with the others. Mathematically, this involves solving for components that maximize the variance under orthogonality and normalization constraints, represented in the eigen decomposition of the covariance or correlation matrix: $\underline{a}'_{(j)}S\underline{a}_{(j)}$ subject to $\underline{a}'_{(j)}\underline{a}_{(j)} = 1$ and $\underline{a}'_{(j)}\underline{a}_{(m)} = 0$, see equation 13. Each principal component score $c_{i(j)}$ is then calculated as $c_{i(j)} = \underline{a}'_{i(j)}\underline{z}_i$, where $\underline{z}_i$ is the standardized data vector. This process is sensitive to the scaling of the original data, making the choice between using the covariance matrix $S$ (see equation 14) and the correlation matrix $R$ crucial, depending on the nature of the data. Applying PCA to a covariance matrix focuses on maximizing variance among highly correlated variables, whereas PCA for a correlation matrix standardizes the scale of variables to treat them equally. As already indicated, this choice can have large ramifications for the analysis, especially when original variable scales differ significantly. In the mathematical formulation, each component maximizes a specific variance function under constraints that ensure orthogonality and unit scaling.

$$\text{Linear Combination } l_{(j)} = \underline{a}'_{(j)}y = a_{(j)1}y_1 + a_{(j)2}y_2 + \ldots + a_{(j)p}y_p \tag{12}$$

$$\underline{a}_{(j)} \text{ maximizes } \underline{a}'_{(j)}S\underline{a}_{(j)} \text{ subject to } \begin{cases} \underline{a}'_{(j)}\underline{a}_{(j)} = 1 & \text{(scaled)} \\ \underline{a}'_{(j)}\underline{a}_{(m)} = 0 & \text{(orthogonal)} \end{cases} \tag{13}$$

$$S = A\Lambda A' = \left[\underline{a}_{(1)}\underline{a}_{(2)}\cdots\underline{a}_{(p)}\right]\begin{bmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \cdots & 0\\ \vdots & \vdots & \cdots & \vdots\\ 0 & 0 & \cdots & \lambda_p\end{bmatrix}\begin{bmatrix}\underline{a}'_{(1)}\\ \underline{a}'_{(2)}\\ \vdots\\ \underline{a}'_{(p)}\end{bmatrix} \tag{14}$$

In practical terms, PCA is used to derive scores, or principal components, that can effectively summarize the data. By plotting these components, we can visually assess if the grouping variable, such as a specific genomic classification, distinctly separates. This visualization helps determine if the embedding matrix possesses the quality of separability, indicating how well it maintains meaningful distinctions within the data.

#### 4.1.3.3 T-Distributed Stochastic Neighbor Embedding

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a statistical method for visualizing high-dimensional data by reducing its dimensions to two or three, facilitating the exploration of data structure. Initially, t-SNE constructs a probability distribution in the high-dimensional space such that similar data points are assigned higher probabilities while dissimilar points are assigned lower probabilities. The similarity of data point $x_j$ to $x_i$ is modeled as a conditional probability $p_{j|i}$, computed using a Gaussian distribution centered at $x_i$. These probabilities are symmetrized to form a joint probability distribution $p_{ij}$ in the high-dimensional space.

In the low-dimensional space, t-SNE defines a similar probability distribution using a Student t-distribution, denoted as $q_{ij}$. The choice of the t-distribution, characterized by heavier tails than a Gaussian distribution, helps in effectively modeling the joint probabilities of dissimilar data points, allowing them to be positioned further apart on the map.

The main objective of t-SNE is to minimize the Kullback-Leibler divergence between the two probability distributions $p_{ij}$ and $q_{ij}$ over all pairs of data points. This minimization is typically achieved through gradient descent techniques. The Kullback-Leibler divergence acts as a cost function that measures the fidelity with which distances in the high-dimensional space are preserved in the low-dimensional embedding.

The effectiveness of t-SNE lies in its ability to capture the local structure of the data while revealing clusters and patterns that are difficult to discern in high-dimensional spaces. However, the method is sensitive to the choice of parameters like perplexity and learning rate, and its computational cost can be substantial for large datasets. Results are seen in Figures 18 and 19.

## 4.2 Extrinsic Evaluation

For extrinsic evaluation, the potential range of benchmark tasks is vast, necessitating the selection of a meaningful subset that categorizes tasks by difficulty. This subset includes tasks such as classifying proteins (see Section 4.2.2) and classifying variants (see Section 4.2.1). These classification tasks serve as practical benchmarks to assess the performance of GenSLM embeddings across varying levels of task complexity. The variations in task difficulty are discussed in Sections 4.2.1 and 4.2.1, with additional details on the chosen learning method provided in Section 4.2.3.

### 4.2.1 Variant Classification

Variant Classification is deemed a harder task given, as already mentioned, the high sequence similarity across SARS-CoV-2 genomes (Zvyagin et al., 2023) and the longer sequence length. In addition, because variant classification is whole genome, the amount of data given to the learner is the same as the number of sequences, while for protein classification, the number of rows of the dataset is the number of proteins times the number of patients, making the task of classifying variants more data constrained than classifying proteins. For performing the classification, a classification and regression tree (CART) learner is utilized with no tuning for classifying the variants Alpha, Beta, Gamma, Delta, Epsilon, Omicron, and Mu in Section 5.2.1. This method achieved notable accuracy, as shown in Section 5.2.1.

In addition classification on variants was also performed with the a One-Hot-encoding representation of the aligned 581 DNA sequences. This serves as one comparison for classification performance for alternative representations of DNA.

### 4.2.2 Protein Classification

Because Proteins have large differences in sequence lengths and functions, classifying proteins should be an easier task than classifying Covid variants. Moreover, we would expect with a shorter sequence (i.e., less dimension reduction), the hierarchical diffusion model would have fewer local interactions to stitch together. By extension, we expected that GenSLM would produce a higher quality embedding and a higher classification performance than variants. Classification was performed by CART using protein labels and 582 embedding vectors as the feature matrix. Classification was also performed on a 28 PCA competent, dimensionally reduced transformation of the GenSLM embedding matrix. This task is a hybrid of both extrinsic and intrinsic evaluation as it indicates whether further dimension reduction can be performed on the embedding without losing any predictive performance.

### 4.2.3 Classification and Regression Trees

Classification and Regression Tree's (CART) was the chosen learner for performing extrinsic evaluation based on its simplicity and popularity for supervised learning tasks in the machine learning community. It was important that a simple and interpretable learner was chosen as information retrieval from the embedding space should be minimally intensive in order for the embedding algorithm to add value to bioinformatic workflows. Hence, all training was done with 20/80 test train splits with no tuning of hyperparameters.

As a background, Classification and Regression Trees (CART) are a type of predictive model that is highly valued for their ability to produce intuitive, interpretable models resembling a flowchart-like structure. This technique involves segmenting the predictor space (the set of possible values for input variables) into distinct and non-overlapping regions for which simple predictions are made. Given the dependent variable for this study is categorical, the prediction is often the mode of the category. The primary appeal of CART lies in the simplicity of its results; producing a sequence of "if-then" decision rules that are straightforward to understand and implement, unlike more complex models like neural networks or ensemble methods.

The process of building a decision tree using CART methodology involves repeatedly splitting the data into further subsets based on feature values. This splitting process, called binary recursive partitioning, starts at the root of the tree and branches down to the leaves. At each node, the tree algorithm selects the best split based on a specific criterion that aims to maximize the homogeneity of the resultant subgroups. Optimality criteria such as the Gini index (as used in this paper) for classification tasks or variance reduction for regression guide these splits. However, allowing the tree to grow unrestrictedly until each leaf is pure leads to models that are highly complex and overfitted to the training data.

To mitigate overfitting, CART incorporates a strategy known as pruning, which involves trimming down the tree after it has been grown. Pruning reduces the size of the tree by removing sections that provide little power in predicting the target variable, thus simplifying the model and potentially improving its generalizability to new data. By balancing model complexity and fit, pruning helps maintain a robust model that performs well not only on the training dataset but also on external validation sets. The Python function `DecisionTreeClassifier` was used from the `sklearn.tree` module for performing all classifications produced in Figures 20 , 21, 22, and 23.

# 5 Results

Our analysis heavily utilizes data visualizations to communicate the results of our intrinsic and extrinsic evaluations. The results of this study are detailed in the following subsection and demonstrate the utility of

GenSLM in bioinformatic workflows.

## 5.1 Intrinsic Evaluation

### 5.1.1 Redundancy

#### 5.1.1.1 Singular Value Decomposition

Using a SVD, as seen in Figure 33, the embedding matrix is still highly redundant, with only seven components explaining 99.68% of the variance. The high cumulative proportion explained indicates the chosen output dimensions of 512 elements in the GenSLM algorithm could possibly have been revised to have fewer dimensions without sacrificing information. For example, in the paper by Jing et al. (2019), 16 embedding algorithms were tested, and only two algorithms had 20 dimensions or higher (Jing et al., 2019).

### 5.1.2 Relative Preservation of Information

#### 5.1.2.1 Distance Matrices

Appearing in the order A) *top left*, B) *top right*, C) *bottom left*, D) *bottom right*, as shown in Figure 11, are A) the sequence distance matrix, B) the embedding distance matrix, C) the absolute difference of the sequence and the embedding matrix, and D) the regular difference of the sequence and the embedding matrix. First, focusing on the sequence distance matrix (plot A), it should be noted that there are seven faint outlines of blue squares running down the diagonal. Moving from left to right, these squares correspond to variants in the following order: Alpha, Beta, Delta, Epsilon, Gamma, Mu, and Omicron. Hence, these squares act like variant column names for all of the distance matrices in Figure 11. Thus, the blue squares indicate what we would expect that all sequences are most closely related to other sequences of the same variant. Moreover, in Plot A, we also observe that Alpha variants have larger within-variant variation (i.e., $\alpha_j$ to $\alpha_{j'}$), as indicated by the L-shaped red striations. What can also be noticed is that there is a lot of heterogeneity between sequences; this heterogeneity is represented by a large variation of color from blue to red.

Comparing the Sequence Distance Matrix Plot (Plot A) and the Embedding Distance Matrix Plot (Plot B), we see that the blue squares in Plot B are more visible, indicating that there are smaller distances between sequences of the same variant than Plot A. The high visibility of the diagonal in plot B indicates that the embedding clearly distinguishes sequences of different variants, and presumably, that classifying variants would be easier with the embedded representation. However, the off-diagonals for plot B indicate that sequence-to-sequence distances of the embeddings are far more homogeneous than the pairwise distances

of the DNA sequences in plot A. This finding is shared by the lower left and right plots, which show how very fine information for the pairwise distances of sequences has been lost in the embedding matrix produced by GenSLM.

Meanwhile, looking at the radial Dendrogram of the embeddings produced in Figure 12, we see that all variants now cluster perfectly, whereas in the Exploratory Data Analysis, the radial Dendrogram of the raw sequences in Figure 7 did not cluster perfectly. This confirms the results from the distance matrices, which show that that the embedding helps force the separation of variants. However, it can shown that some information was distorted in the embedding process, as indicated by the differing locations of the Wuhan reference between Figure 7 and Figure 12. The shift in the location of the Wuhan sequence indicates that the genetic distance from the original sequence data was not preserved in the transformation process of the GenSLM Embedding.

#### 5.1.2.2 Kullback–Leibler Divergence

Both the discrete and continuous KL Divergence in the left and middle panels of Figure 13 measure the divergence of individual sequences from the reference. In the Discrete case (in the left panel), we see that the positioning of variants roughly matches the results from the raw sequences in the Exploratory Data Analysis Section. For example, in Figure 13, the Alpha variant (in red) has the largest divergence from the Wuhan reference sequence compared to any other variant; this relationship is also seen in Figure 8.

Comparing the discrete and continuous KL Divergences in Figure 13, we see that Discrete KL Divergence (left panel) does not show very good clustering of variants, while in comparison, the continuous KL Divergence (middle panel) shows a very clear separation of variants. However, variants Delta and Mu have a considerably higher divergence from the reference in the continuous case (middle panel) than they do in the discrete case (left panel). There is clearly a distortion in the divergence of Delta and Mu in the discrete case versus the continuous; this distortion is also seen in the rearrangement of variants in the radial dendrograms from Figure 7 to Figure 12. Such distortion of information is concerning, as it could lead practitioners using the embeddings to overestimate the genetic distance of the Delta and Mu variants from the reference strain, potentially resulting in serious complications for subsequent analyses. Finally, in the third panel, we see that a linear relationship does not emerge when plotting the discrete versus continuous KL divergence. This means there is a fundamental change in the representation of data before and after the GenSLM embedding.

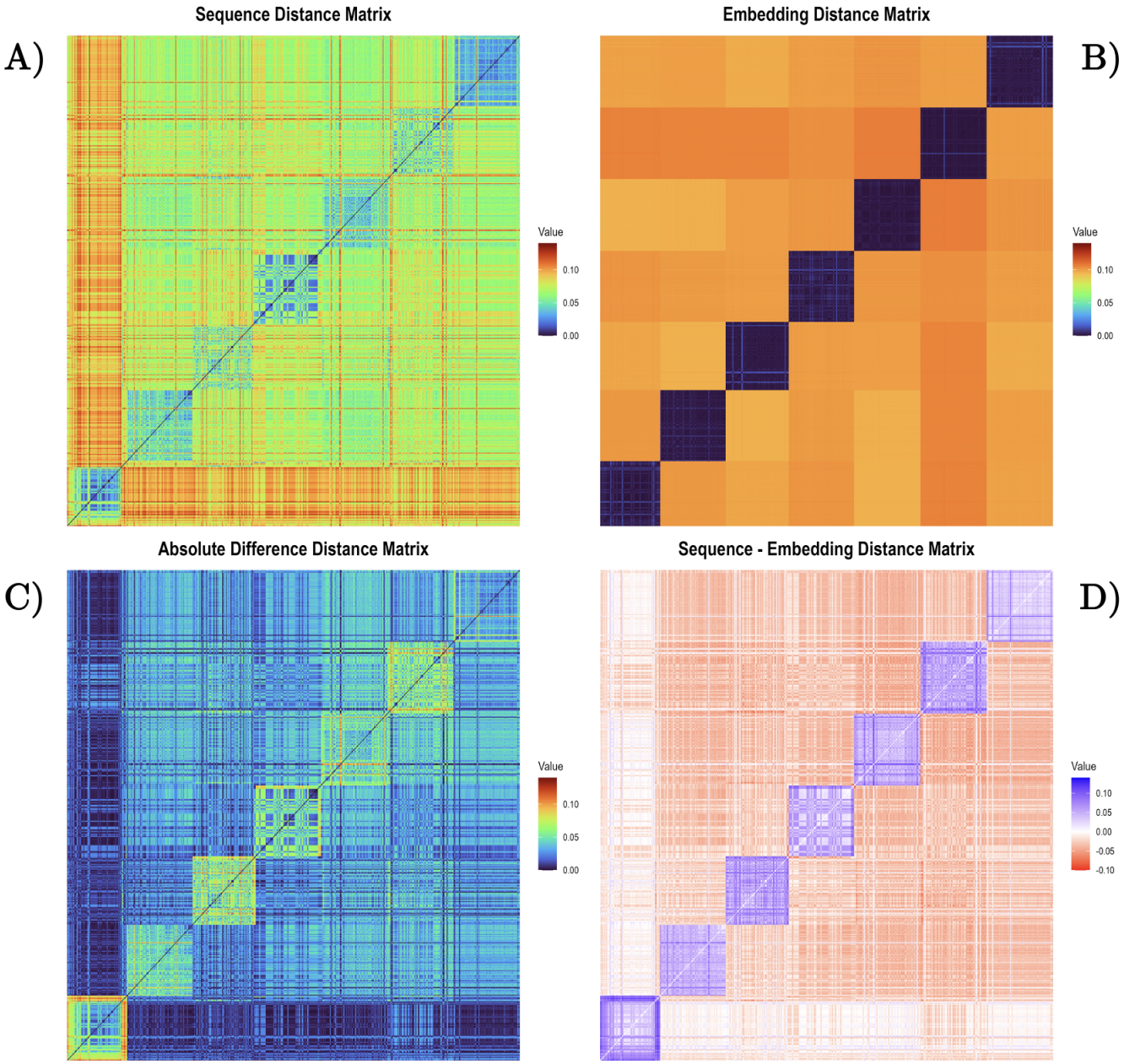


Figure 11: Pairwise Sequence Distance Matrices Versus Euclidean Distance Embedding Matrices Across Seven Variants

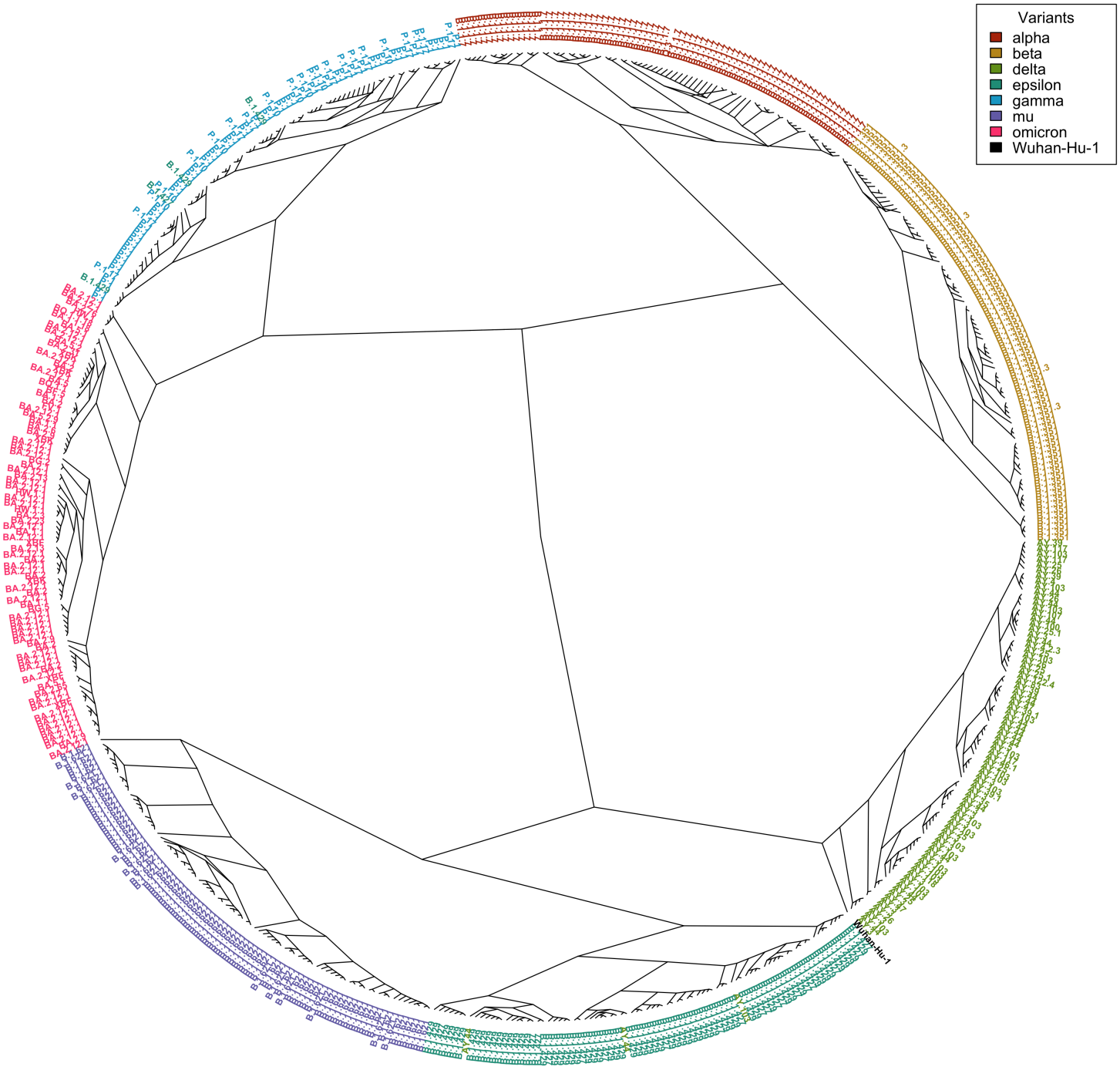


Figure 12: Radial Dendrogram for 581 Aligned Covid Sequence Embeddings with Wuhan-Hu-1 Reference
- See Figure 7 for Comparison -

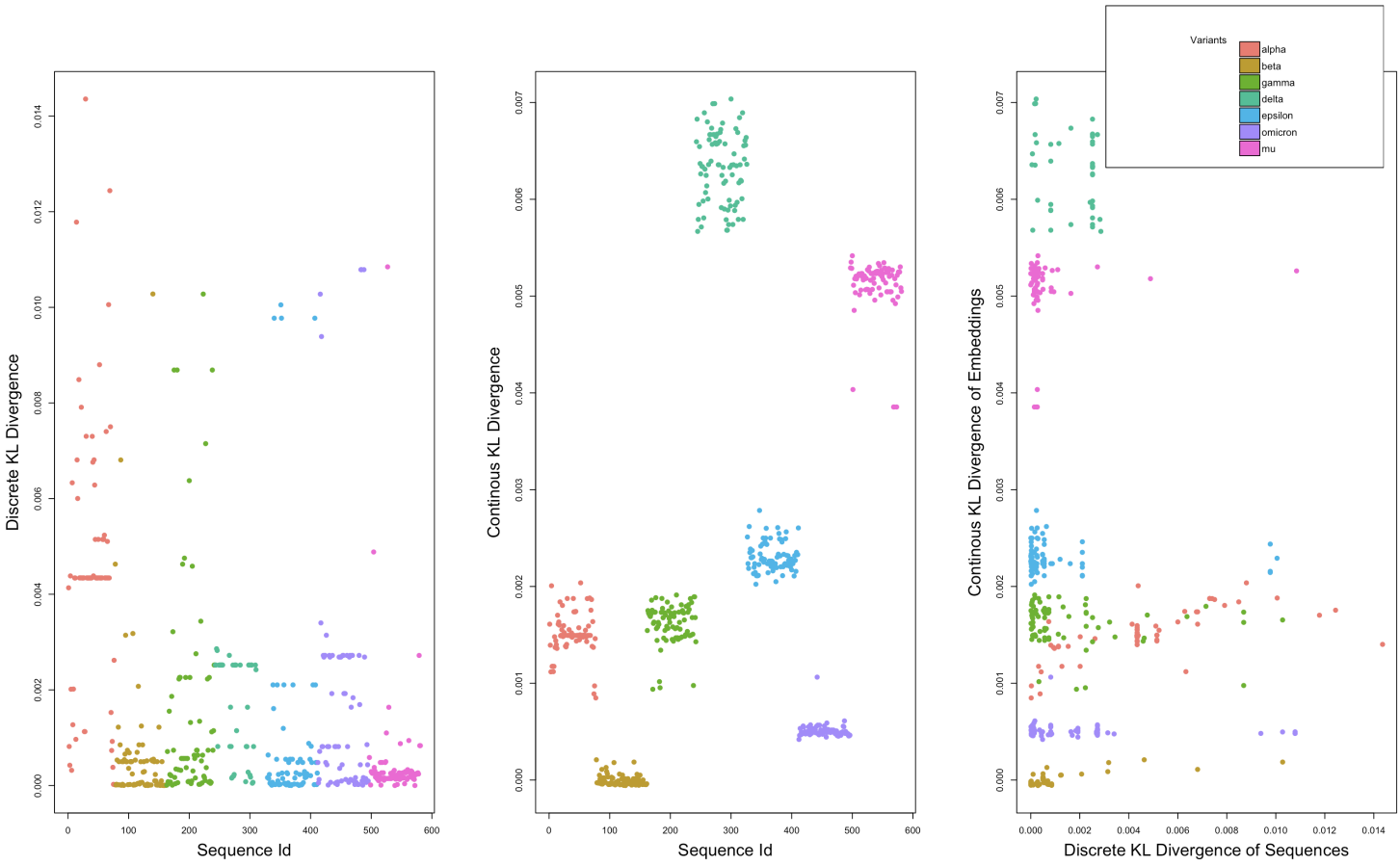


Figure 13: Discrete (DNA Sequences) Versus Continuous (Numeric Embeddings) KL Divergence Relative to the Wuhan Reference Sequence

### 5.1.3 Separability

#### 5.1.3.1 Linear Discriminant Analysis

For the LDA of proteins, as shown in Figure 14, the model perfectly separates all proteins. The zoomed-in subplot in Figure 14 shows that each dot in the zoomed-out plot is a cluster of proteins from the sequence embeddings. This indicates that there is good separability of data from the GenSLM algorithm. On the other hand, GenSLM does not have good separability when applied to variants; for example, LDA was not able to separate variants Delta and Epsilon. For the LDA of proteins and variants, we learned that separability is harder to achieve for highly similar sequences such as variants.

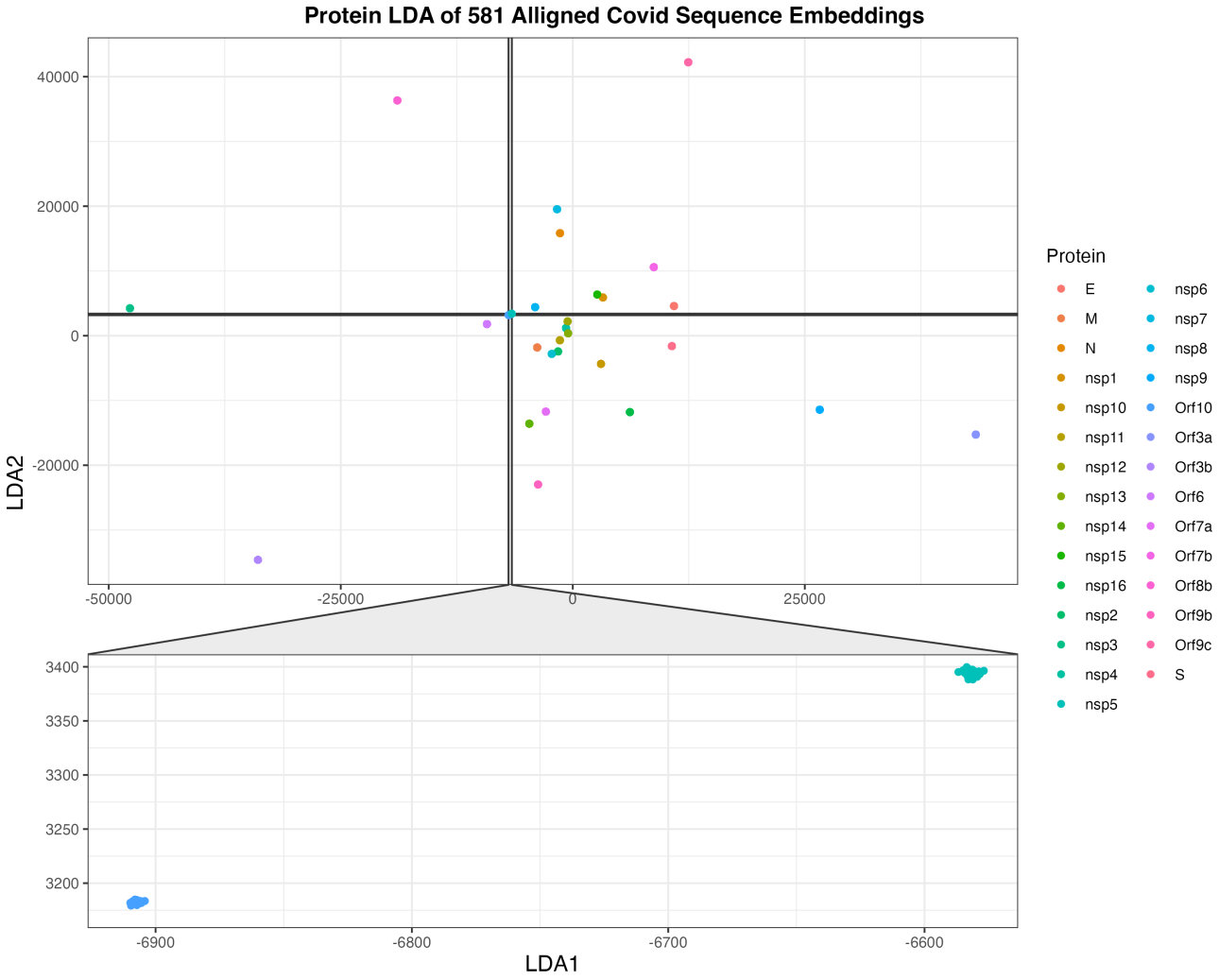


Figure 14: Protein LDA Plot

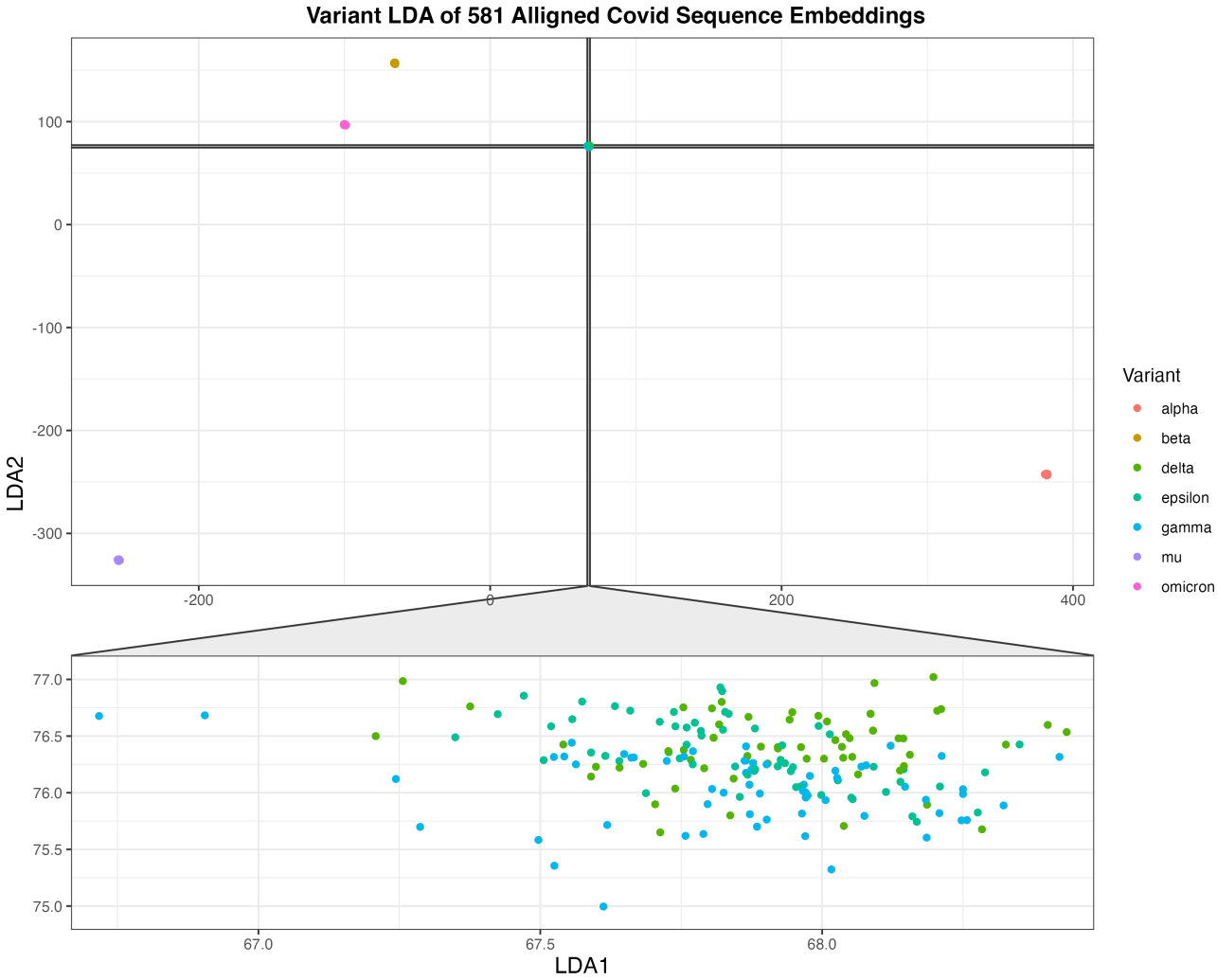


Figure 15: Variant LDA Plot

#### 5.1.3.2 Principal Component Analysis

Moving on to PCA, we see that in Figure 16 and 17, both variants and proteins are perfectly separable, indicating good separability from the GenSLM algorithm for both variants and proteins.

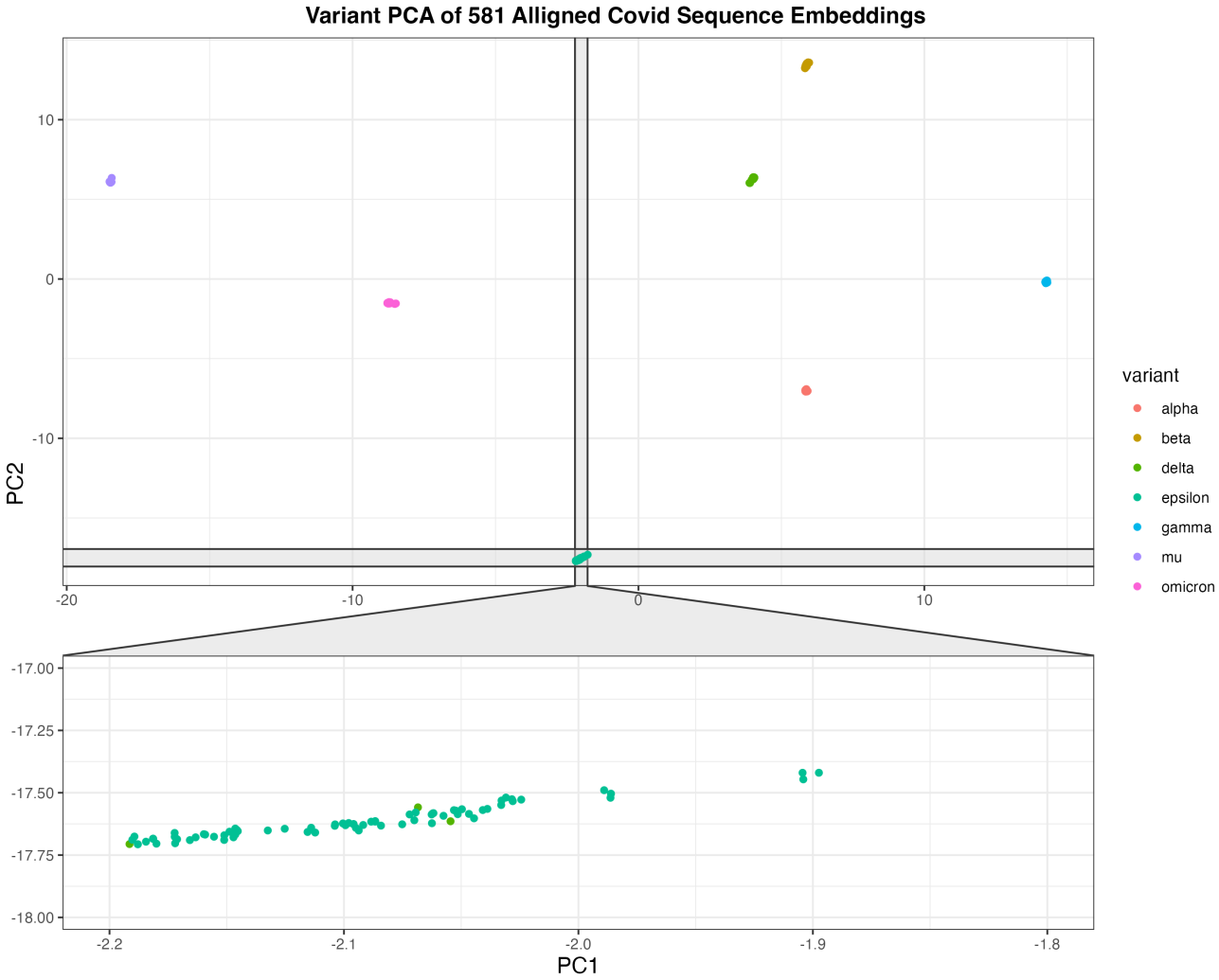


Figure 16: Variant PCA Plot

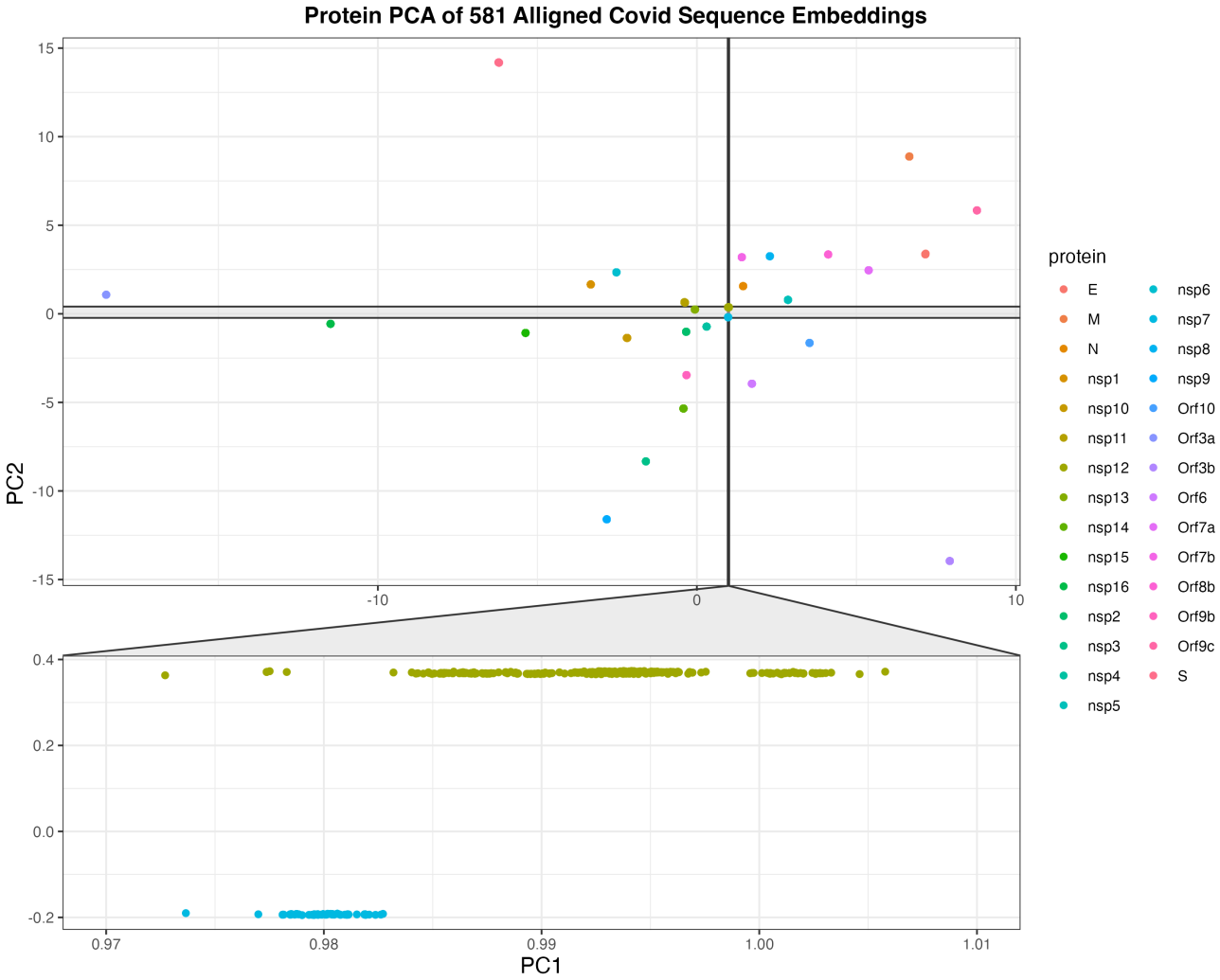


Figure 17: Protein PCA Plot

#### 5.1.3.3 t-Distributed Stochastic Neighbor Embedding

The t-SNE analysis was a third method for examining the separability of the GenSLM embeddings. Similar to the PCA analysis, the t-SNE shows perfect separability of both variants and proteins.

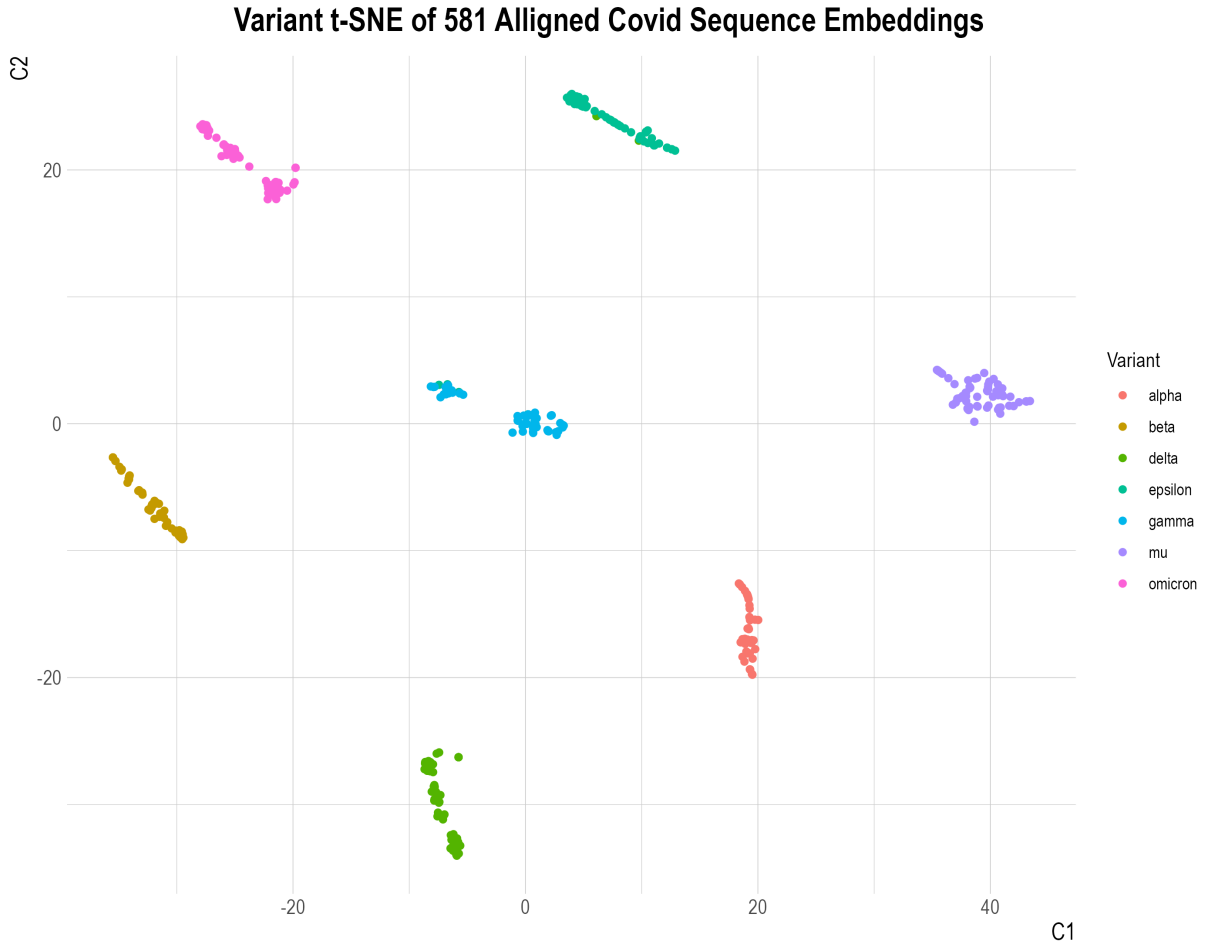


Figure 18: t-SNE Clustering of Variants from Embedded Sequences

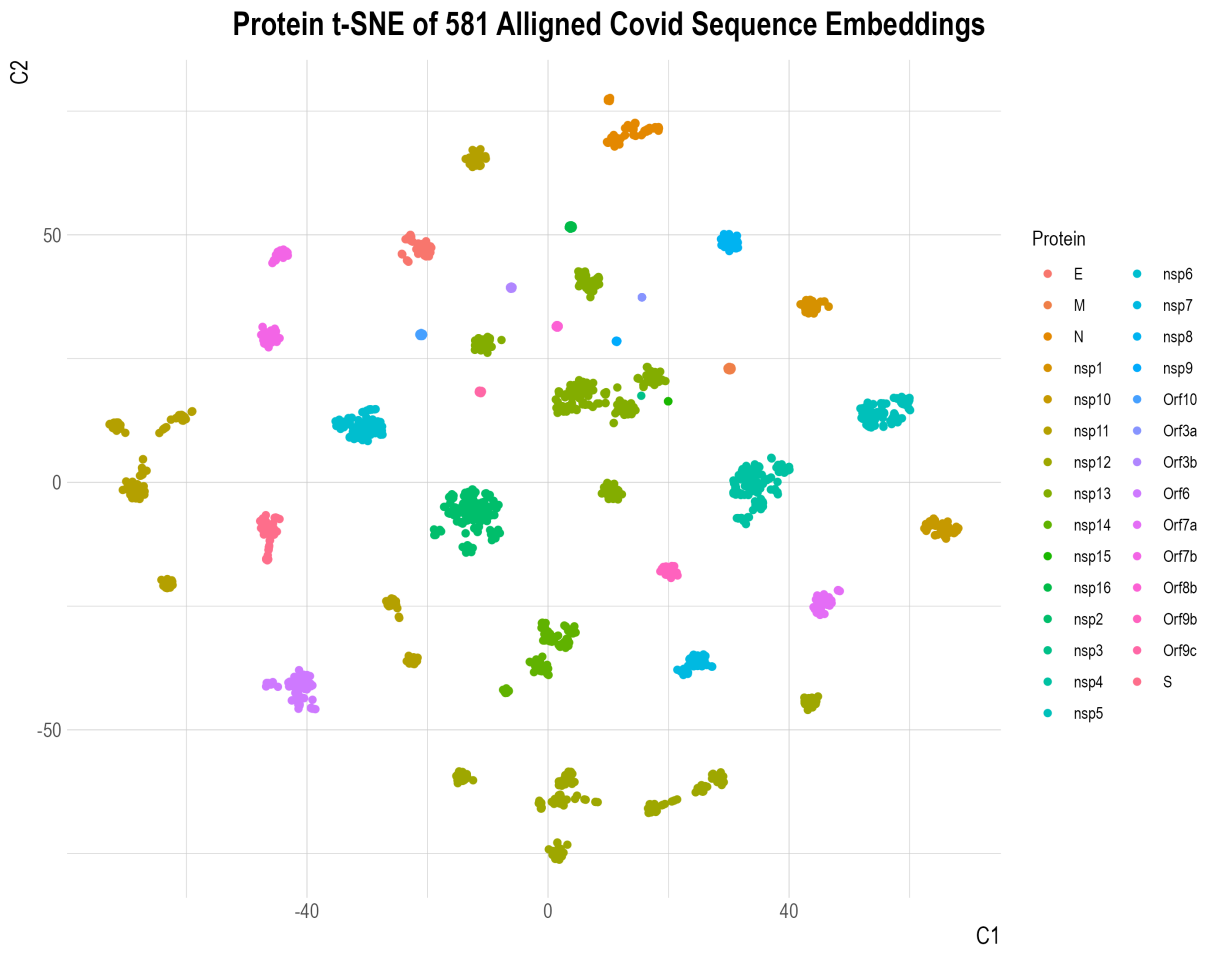


Figure 19: t-SNE Clustering of Proteins from Embedded Sequences

## 5.2 Extrinsic evaluation

### 5.2.1 Variant Classification

The CART model fit for classifying variants produces the tree as seen in Figure 20. Notably, the classification rate is near perfect, with a classification rate of 97.71% and four misclassified variants, see Table 2. Overall, the classification rate of 97.71% reflects well on the GenSLM algorithm, given that a simple learner (CART) can classify nearly 100% of variants using the embedded GenSLM data. The CART model classified the embedding data with ten leaves. In contrast, another CART model, using One-Hot-encoded whole genome sequence data produced a tree with 341 leaves with only a 10.28% classification rate, see Figure 21. The deep tree shown in Figure 21 is the result of the known flaw of using decision tree-based learners on One-Hot-Encoded data. However, the magnitude of the difference in classification from 97.71% to

10.28%, gives context to the size of performance difference that can be observed between different encodings of genomic data. The difference between classification performance, moreover, further highlights the value of GenSLM and embedding algorithm's in general.

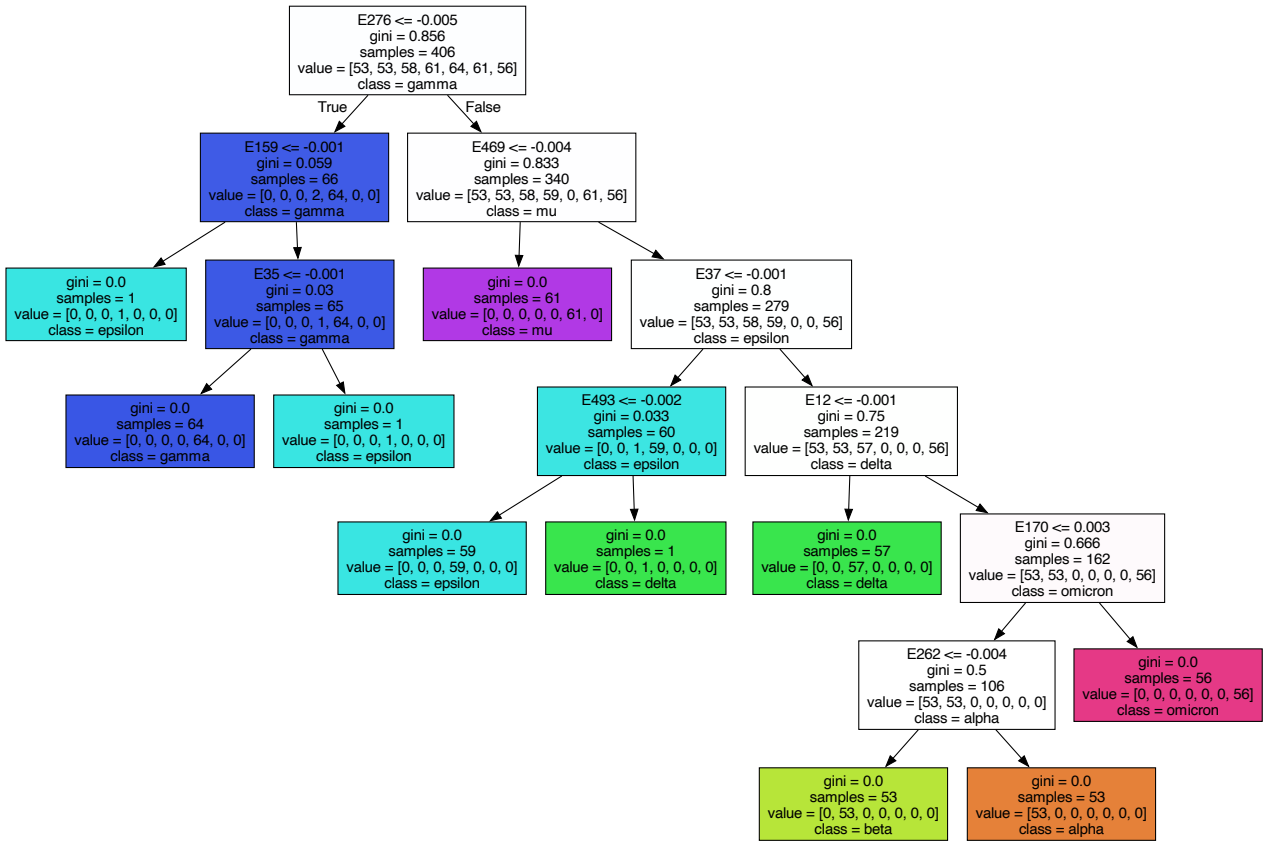


Figure 20: CART Variant Classification (97.71%) on Aligned Sequence Embeddings

Table 2: Misclassified Variants

| Sequence ID | Actual | Predicted |
|---|---|---|
| 244 | delta | epsilon |
| 268 | epsilon | delta |
| 335 | gamma | epsilon |
| 408 | gamma | epsilon |

Figure 21: CART Variant Classification (10.28%) of Aligned Sequence One-hot-encodings
- the image was cropped showing only 82 leaves out of 341 shown (zoom in to see tree) -

### 5.2.2 Protein Classification

In our analyse we did two types of protein classification using CART. The difference in each analysis was the data used. The first analysis used the GenSLM embedding data, while the second analysis used 28 components from the PCA reduction of the GenSLM embedding.

The CART classification task using the regular embedding, correctly classified 100% of proteins, as seen in Figure 22. This result demonstrates that the GenSLM algorithm effectively preserved the data. While the tree looks complex with a tree depth of 25, there are 29 proteins to classify, indicating that the tree is not over-fit. There are several possible reasons why the CART classification worked better for proteins than variants. One explanation could be due to the way the protein data was embedded and structured. By embedding the proteins from 582 sequences and then further expanding each sequence variant into 29 additional rows (for each protein), the feature matrix for proteins was significantly enlarged to 16,878 rows (582 * 29 = 16,849). The large increase in the number of rows provides more data, which might have provided more useful information for accurately classifying proteins. Another possible explanation is that proteins differ more in their genomic characteristics than variants do. For instance, different Covid proteins have vastly different amounts of nucleotides; the protein 'nsp13' has 39 nucleotides, while 'nsp3' has 5,835 nucleotides. On the other hand, the CART variant classification task was more difficult because the sequence length of each embedded genome were all of the same length[17]. Inherently groups are easier to classify when they naturally differ; there is a lot of natural variation in protein nucleotides compared to whole genomes.

Meanwhile for classification using the PCA reduced feature matrix, as seen in Figure 23, the classification rate is still 100%, and only using 28 dimensions instead of GenSLM's 512. This finding indicates that further dimension reduction is possible within the GenSLM embedding without impacting classification performance, highlighting its redundancy. Thus, while GenSLM scores favourably in terms of separability and preservation of information, the GenSLM embedding is still highly redundant.

[17] As shown by the vertical line in Figure 6.

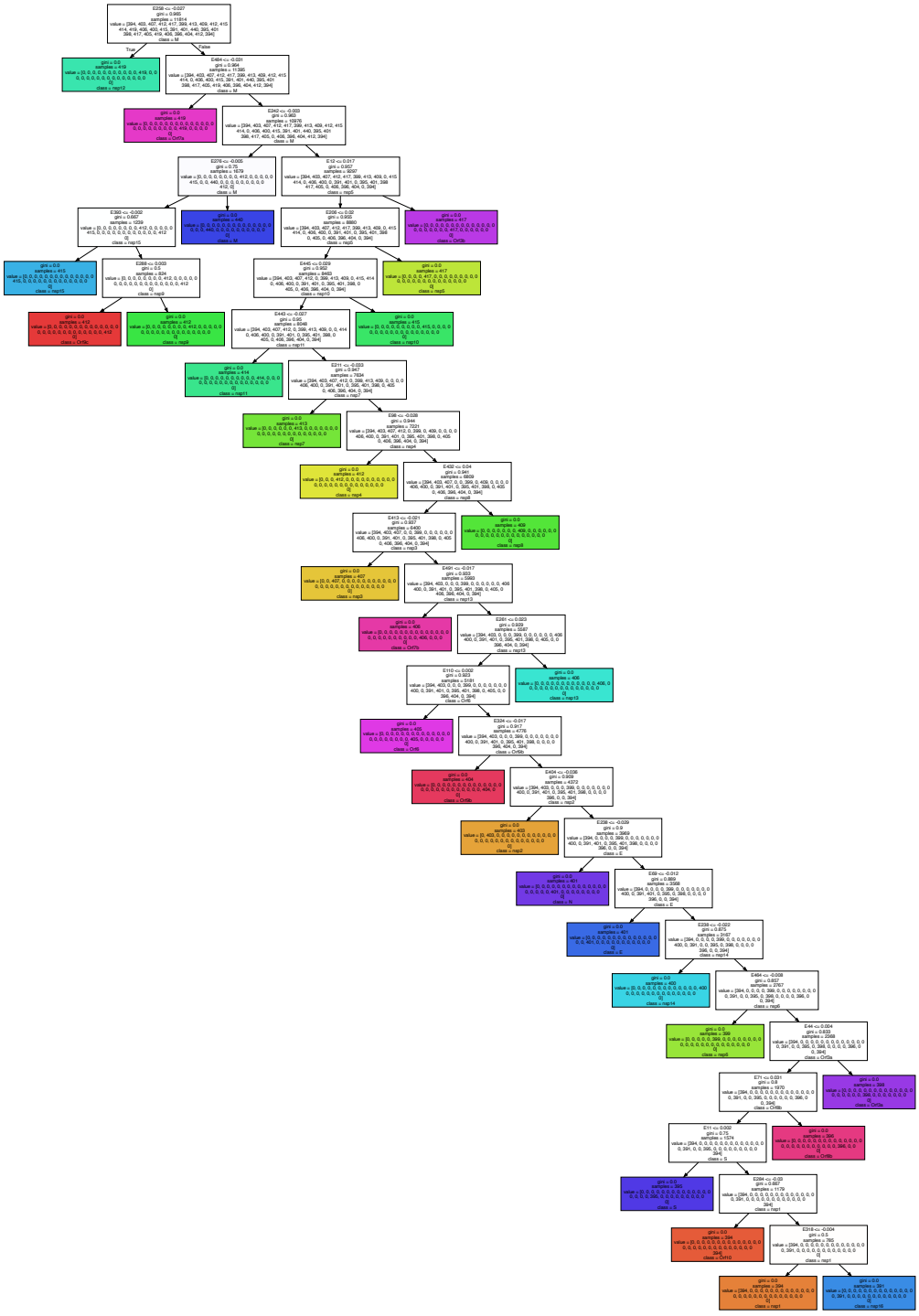

Figure 22: CART Protein Classification (100%) for Aligned Sequence Embeddings (581 patients)

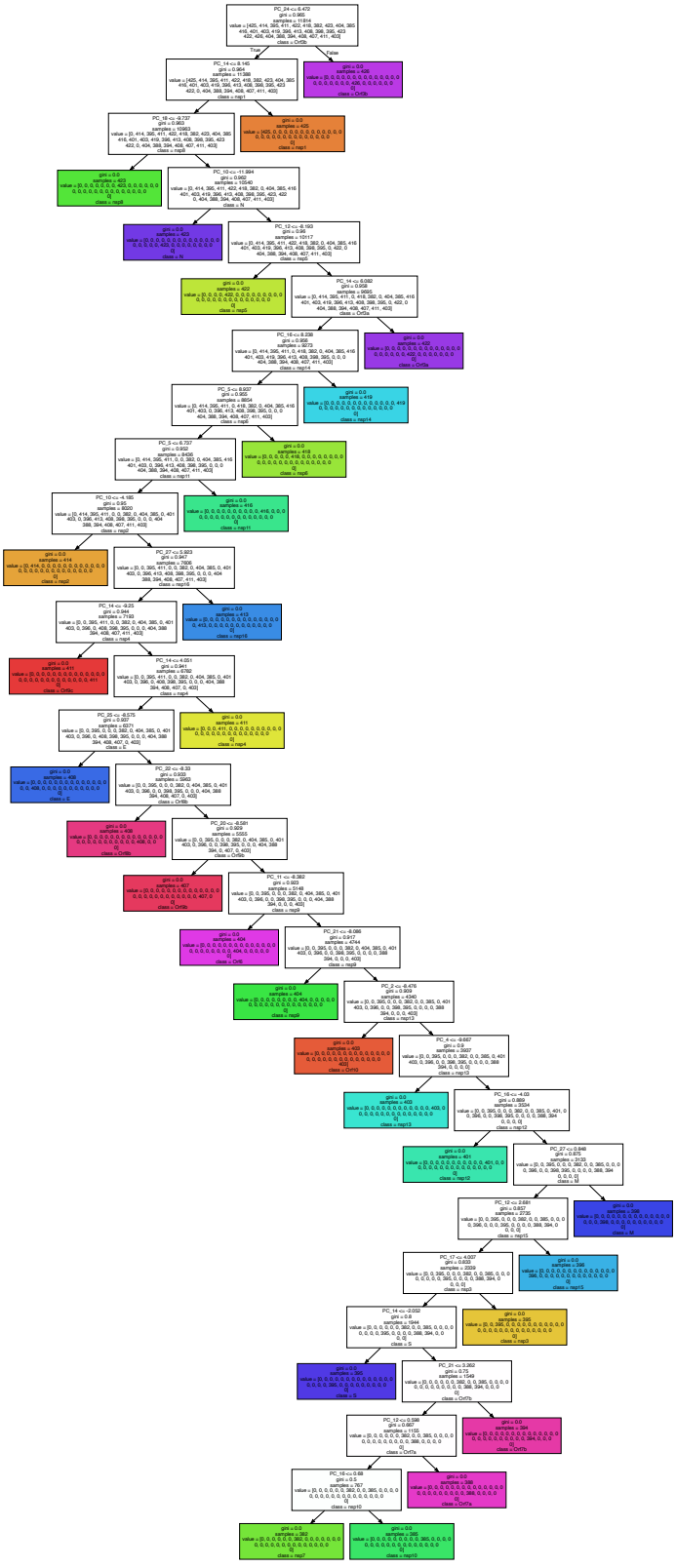

Figure 23: CART Protein Classification (100%) of 29 PCA Components (581 patients)

# 6 Conclusion

This study aimed to assess the quality of GenSLM's embeddings through intrinsic and extrinsic evaluations. It focused on characterizing the embeddings in terms of redundancy, separability, and information preservation using established methods. The results demonstrated multiple favorable aspects of the GenSLM embedding, alongside several suboptimal outcomes. GenSLM performed well in classifying variants and proteins when using the learner CART; classification rates were 97.71% and 100% respectively. Strikingly, for variant classification with CART, there was a +87.43% difference in the classification rate between the GenSLM embedding and the One-Hot-Encoded versions of the feature matrix. In addition, GenSLM performed very highly for separability. However, less favorable aspects resulted from the preservation of information and redundancy analyses. Distance matrices showed that fine scale differences between whole genomes were lost in the embedding transformation process. Furthermore, in comparing continuous and discrete KL divergence, GenSLM distorted the genetic distance of Delta and Mu variants from the Wuhan reference. Additionally, GenSLM achieved only modest success in preserving semantic distance and also displayed considerable dimensional redundancy, both of which are suboptimal outcomes.

Although we confirmed that GenSLM's embeddings excel in terms of separability and downstream task performance, these results are specific to whole genome and protein-focused DNA sequence analyses. Additionally, no other neural network embedding algorithms were compared to GenSLM, nor were alternative learners besides CART employed for evaluating classification tasks. For future research, I recommend benchmarking GenSLM against other neural network embedding algorithms like DNABERT2 and HyenaDNA. Future research should also include tasks assessing whole genome data, such as variant classification and more sensitive analyses like nucleotide frequency regression. A comprehensive benchmark across various algorithms would provide practical insights for practitioners to choose the most suitable algorithm for their specific needs.

# 7 References

## References

Sajia Akhter, Ramy K Aziz, Mona T Kashef, Eslam S Ibrahim, Barbara Bailey, and Robert A Edwards. Kullback leibler divergence in complete bacterial and phage genomes. *PeerJ*, 5:e4026, 2017.

Shun-ichi Amari. *Information geometry and its applications*, volume 194. Springer, 2016.

Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. Deep learning for computational biology. *Molecular systems biology*, 12(7):878, 2016.

Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*, 2017.

Chongzhi Bai, Qiming Zhong, and George Fu Gao. Overview of sars-cov-2 genome-encoded proteins. *Science China Life Sciences*, 65(2):280–294, 2022.

Ulrich Bodenhofer, Enrico Bonatesta, Christoph Horejs-Kainrath, and Sepp Hochreiter. msa: an r package for multiple sequence alignment. *Bioinformatics*, 31(24):3997–3999, 2015. doi: 10.1093/bioinformatics/btv494.

D. Charif and J.R. Lobry. dist.alignment: Pairwise distances from aligned protein or dna/rna sequences. https://www.rdocumentation.org/packages/seqinr/versions/4.2-36/topics/dist.alignment, 2024. Accessed: 2024-05-24.

Gabriele Corso, Zhitao Ying, Michal Pándy, Petar Veličković, Jure Leskovec, and Pietro Liò. Neural distance embeddings for biological sequences. *Advances in Neural Information Processing Systems*, 34:18539–18551, 2021.

I. Csiszar. $I$-Divergence Geometry of Probability Distributions and Minimization Problems. *The Annals of Probability*, 3(1):146 – 158, 1975. doi: 10.1214/aop/1176996454. URL https://doi.org/10.1214/aop/1176996454.

Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. Towards understanding linear word analogies. *arXiv preprint arXiv:1810.04882*, 2018.

Dariush D Farhud and Nooshin Mojahed. Sars-cov-2 notable mutations and variants: a review article. *Iranian Journal of Public Health*, 51(7):1494, 2022.

Erfaneh Gharavi, Aaron Gu, Guangtao Zheng, Jason P Smith, Hyun Jae Cho, Aidong Zhang, Donald E Brown, and Nathan C Sheffield. Embeddings of genomic region sets capture rich biological associations in lower dimensions. *Bioinformatics*, 37(23):4299–4306, 2021.

Hitoshi Iuchi, Taro Matsutani, Keisuke Yamada, Natsuki Iwano, Shunsuke Sumi, Shion Hosoda, Shitao Zhao, Tsukasa Fukunaga, and Michiaki Hamada. Representation learning applications in biological sequence analysis. *Computational and Structural Biotechnology Journal*, 19:3198–3208, 2021. ISSN 2001-0370. doi: https://doi.org/10.1016/j.csbj.2021.05.039. URL https://www.sciencedirect.com/science/article/pii/S2001037021002208.

Xiaoyang Jing, Qiwen Dong, Daocheng Hong, and Ruqian Lu. Amino acid encoding methods for protein sequences: a comprehensive review and assessment. *IEEE/ACM transactions on computational biology and bioinformatics*, 17(6):1918–1931, 2019.

Daniel Jurafsky and James H Martin. Speech and language processing 3rd edition draft, 2019.

Kaggle. Stanford ribonanza rna folding competition leaderboard. https://www.kaggle.com/competitions/stanford-ribonanza-rna-folding/leaderboard, 2023. Accessed: 2014-04-07.

Shubhangi Kandwal and Darren Fayne. Genetic conservation across sars-cov-2 non-structural proteins–insights into possible targets for treatment of future viral outbreaks. *Virology*, 2023.

Nada Lavrač, Vid Podpečan, and Marko Robnik-Šikonja. *Representation Learning: Propositionalization and Embeddings*. Springer, 2021.

Zhiyuan Liu and Maosong Sun. *Representation Learning and NLP*, pages 1–27. Springer Nature Singapore, Singapore, 2023. ISBN 978-981-99-1600-9. doi: 10.1007/978-981-99-1600-9_1. URL https://doi.org/10.1007/978-981-99-1600-9_1.

National Human Genome Research Institute. Base pair. Web page, May 2024. URL https://www.genome.gov/genetics-glossary/Base-Pair.

NCBI Reference Sequence. Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1, complete genome. National Center for Biotechnology Information, 2023. URL https://www.ncbi.nlm.nih.gov/nuccore/NC_045512.2/. Accessed: 04/23/2024.

Nextstrain. Genomic epidemiology of novel coronavirus - global subsampling, 2022. URL https://nextstrain.org/ncov/gisaid/global/2022-01-26?branchLabel=none&f_clade_membership=21A%20%28Delta%29,21C%20%28Epsilon%29,21G%20%28Lambda%29,21H%20%28Mu%29,21I%20%28Delta%

29,21J%20%28Delta%29,21K%20%28Omicron%29,21L%20%28Omicron%29&l=radial&m=div. Built with nextstrain/ncov. Maintained by the Nextstrain team. Data updated 2022-01-26. Enabled by data from GISAID. Showing 2156 of 3044 genomes sampled between Jan 2021 and Jan 2022. Filtered to 21A (Delta) (39), 21C (Epsilon) (5), 21G (Lambda) (5), 21H (Mu) (16), 21I (Delta) (157), 21J (Delta) (1544), 21K (Omicron) (377), 21L (Omicron) (13).

Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *Advances in neural information processing systems*, 36, 2024.

Emmanuel Paradis, Martin Smith, and Damien de Vienne. plot.phylo: Plot phylogenies. https://rdrr.io/cran/ape/man/plot.phylo.html, 2024. Accessed: 2024-05-24.

Søren Kamaric Riis and Anders Krogh. Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments. *Journal of Computational Biology*, 3(1):163–183, 1996.

ScienceDirect. Genetic code. Encyclopedia of Energy, 2004. URL https://www.sciencedirect.com/topics/physics-and-astronomy/genetic-code. A species is the aggregate genetic code, and its phenotype is the organism that can generate a sufficient energy profit to reproduce at some place along each possible environmental gradient.

Fabian Sievers, Andreas Wilm, David Dineen, Toby J Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Söding, et al. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular systems biology*, 7(1):539, 2011.

W Wirth and Duchene S GISAIDR. programmatically interact with the gisaid databases, 2022.

Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana Davuluri, and Han Liu. Dnabert-2: Efficient foundation model and benchmark for multi-species genome." arxiv. *arXiv preprint ArXiv:2306.15006*, 2023.

Maxim Zvyagin, Alexander Brace, Kyle Hippe, Yuntian Deng, Bin Zhang, Cindy Orozco Bohorquez, Austin Clyde, Bharat Kale, Danilo Perez-Rivera, Heng Ma, et al. Genslms: Genome-scale language models reveal sars-cov-2 evolutionary dynamics. *The International Journal of High Performance Computing Applications*, 37(6):683–705, 2023.

# 8 Appendices

## 8.1 Figures

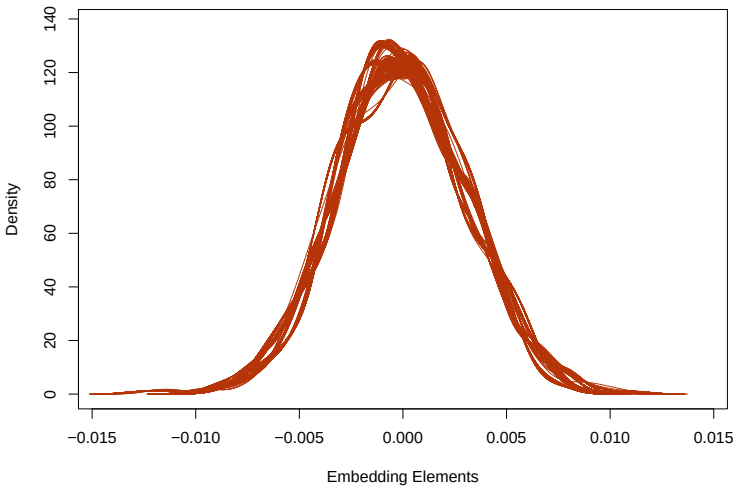


Figure 24: Density Plots of of 581 Plus Reference Embedding Vectors

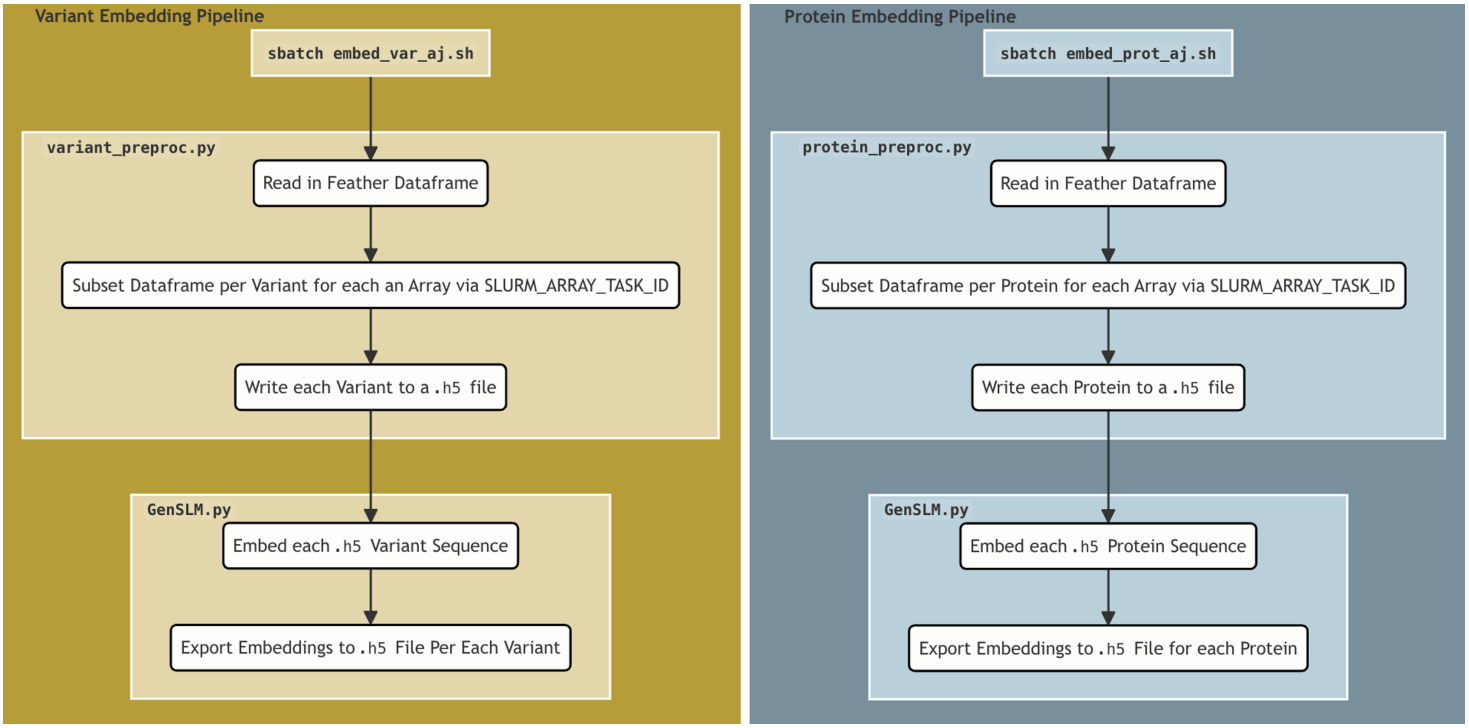


Figure 25: Embedding Pipelines

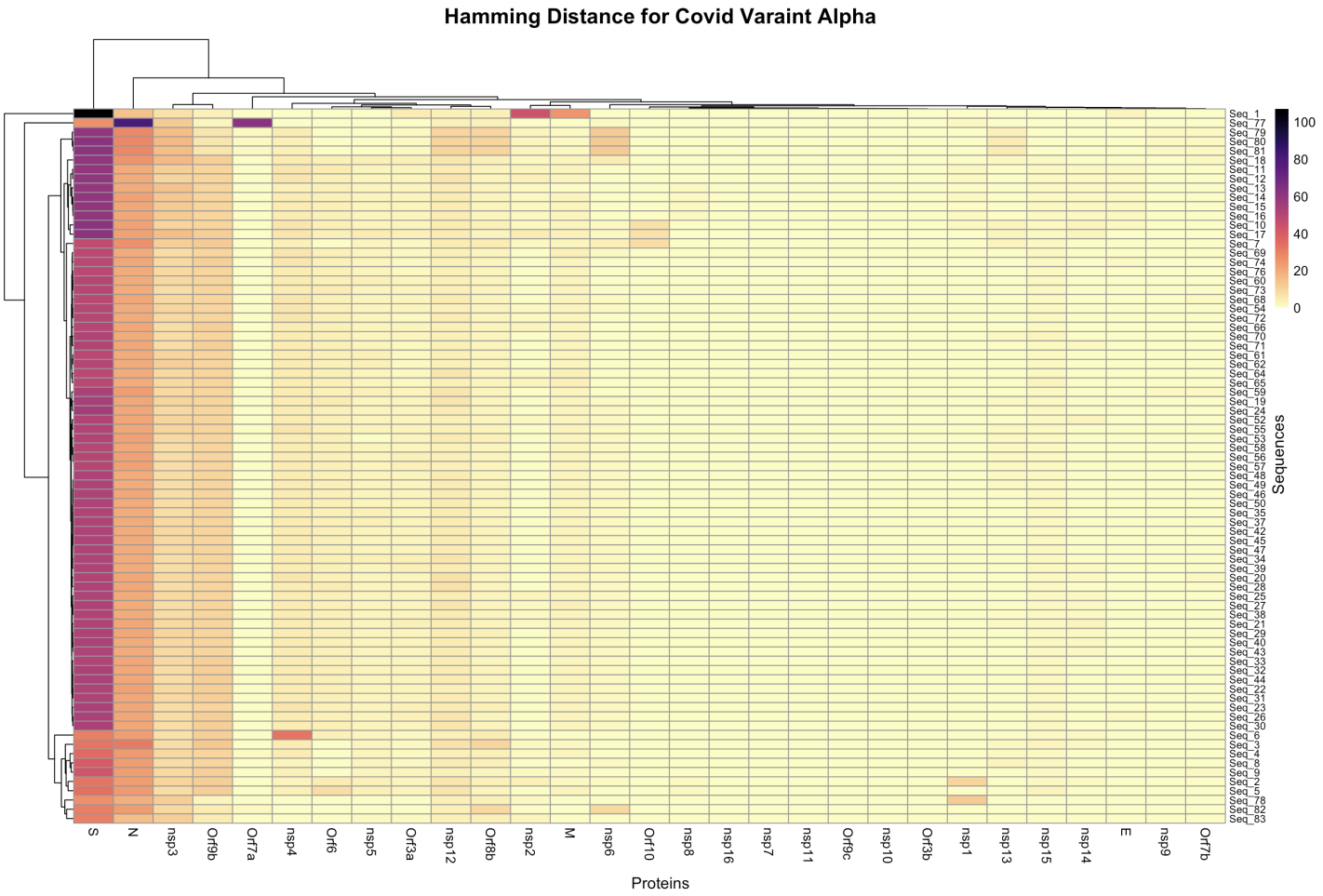


Figure 26: Hamming Distance of Aligned Alpha Proteins to Wuhan Reference Sequence

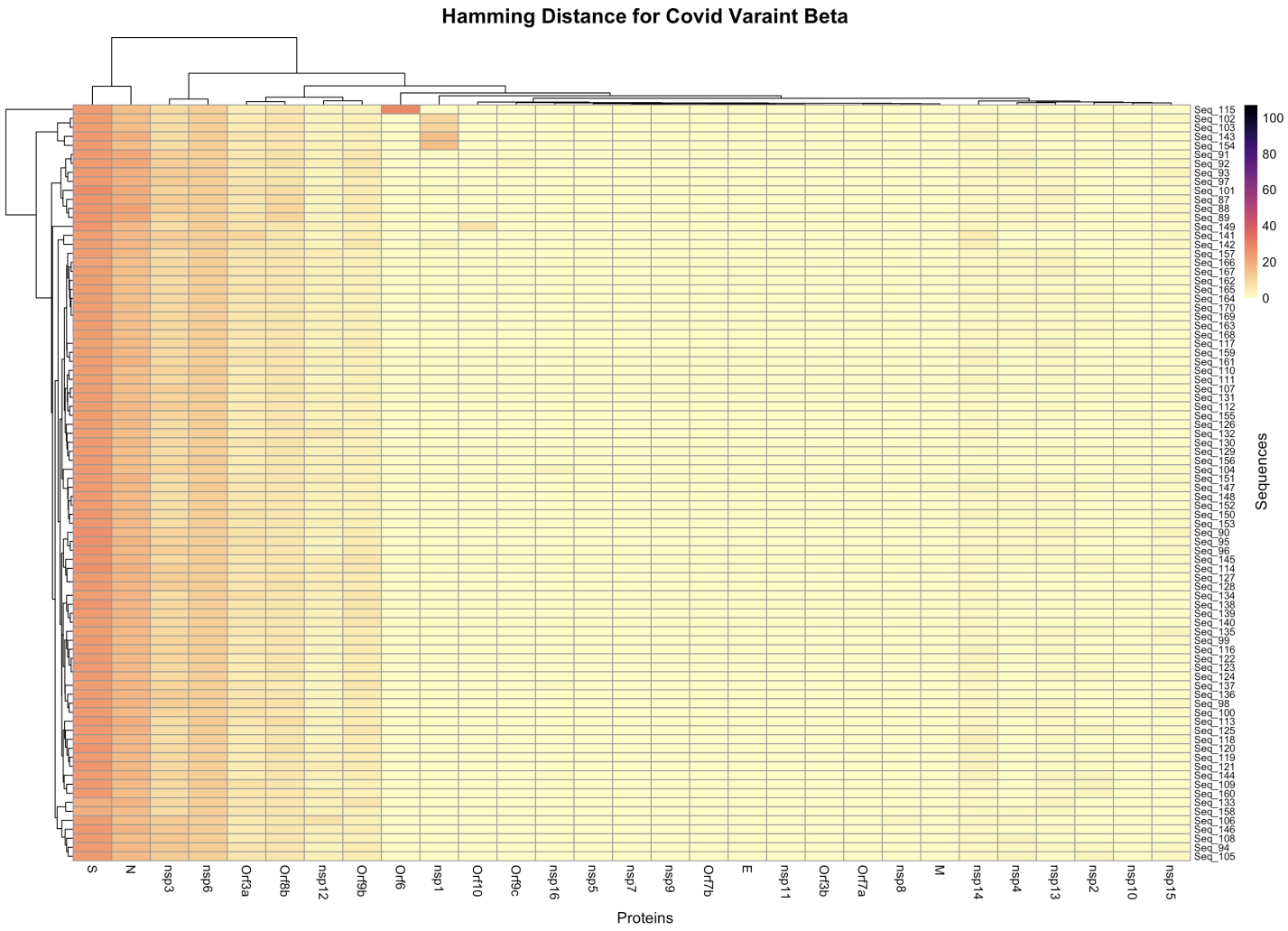


Figure 27: Hamming Distance of Aligned Beta Proteins to Wuhan Reference Sequence

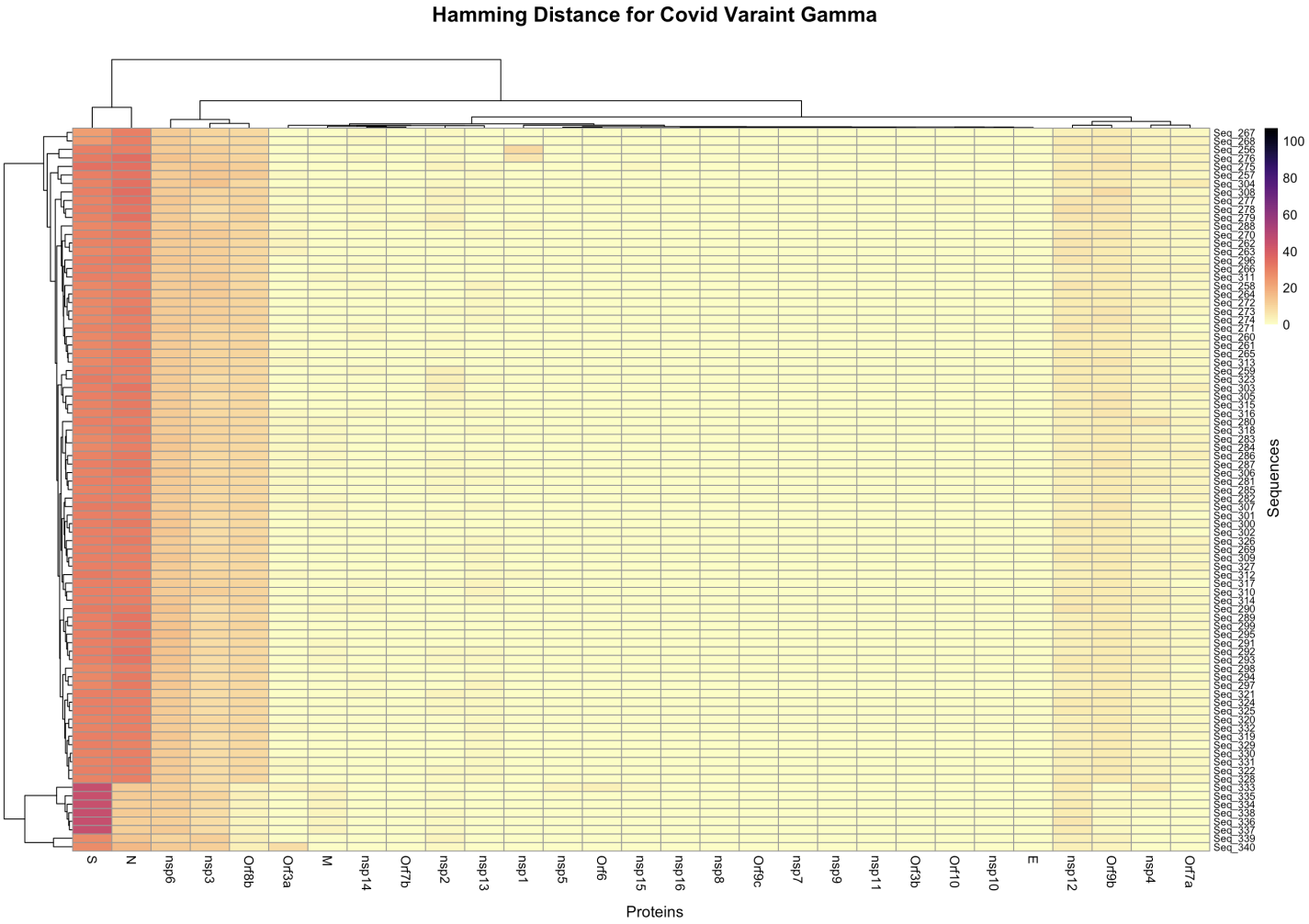


Figure 28: Hamming Distance of Aligned Gamma Proteins to Wuhan Reference Sequence

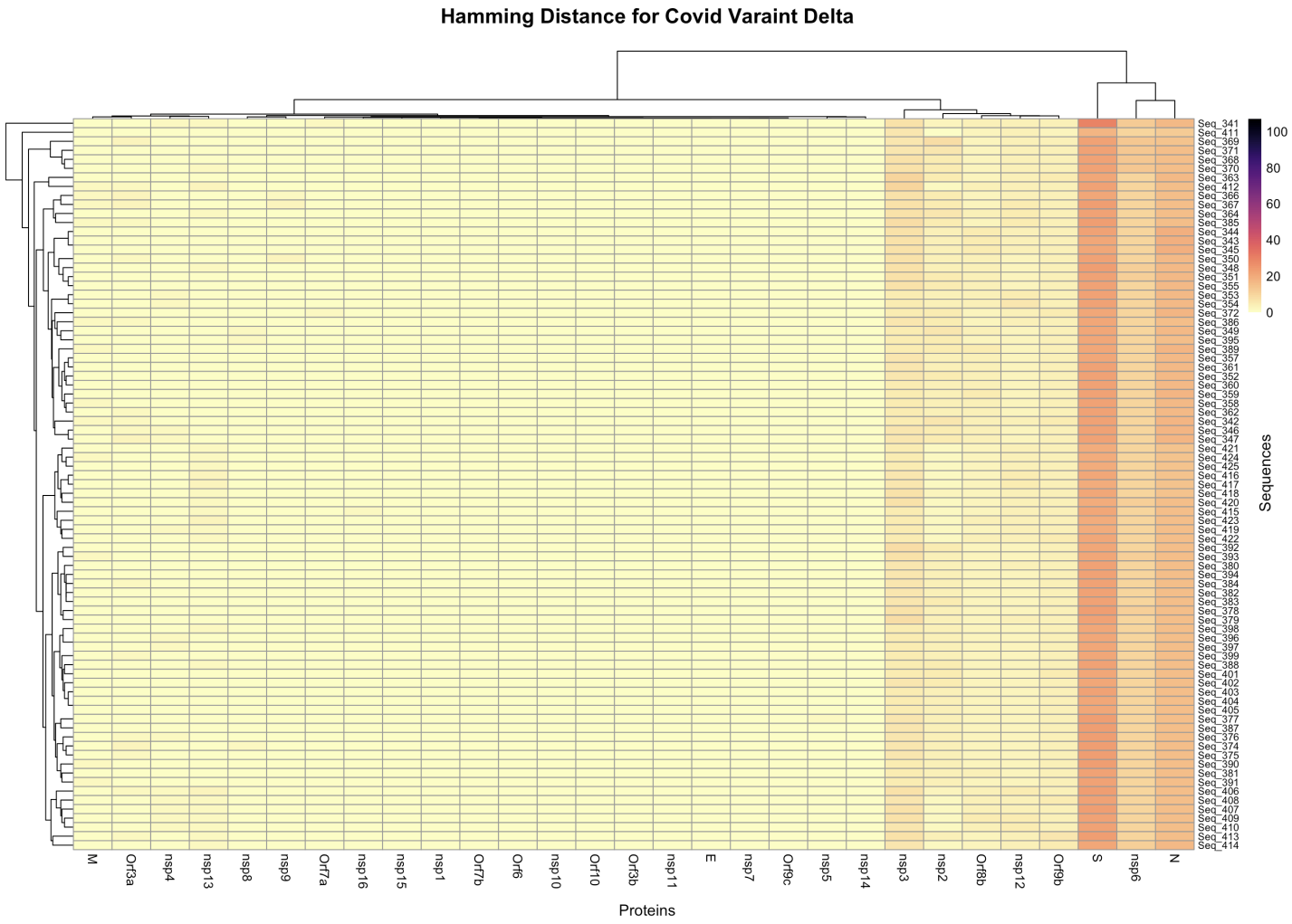


Figure 29: Hamming Distance of Aligned Delta Proteins to Wuhan Reference Sequence

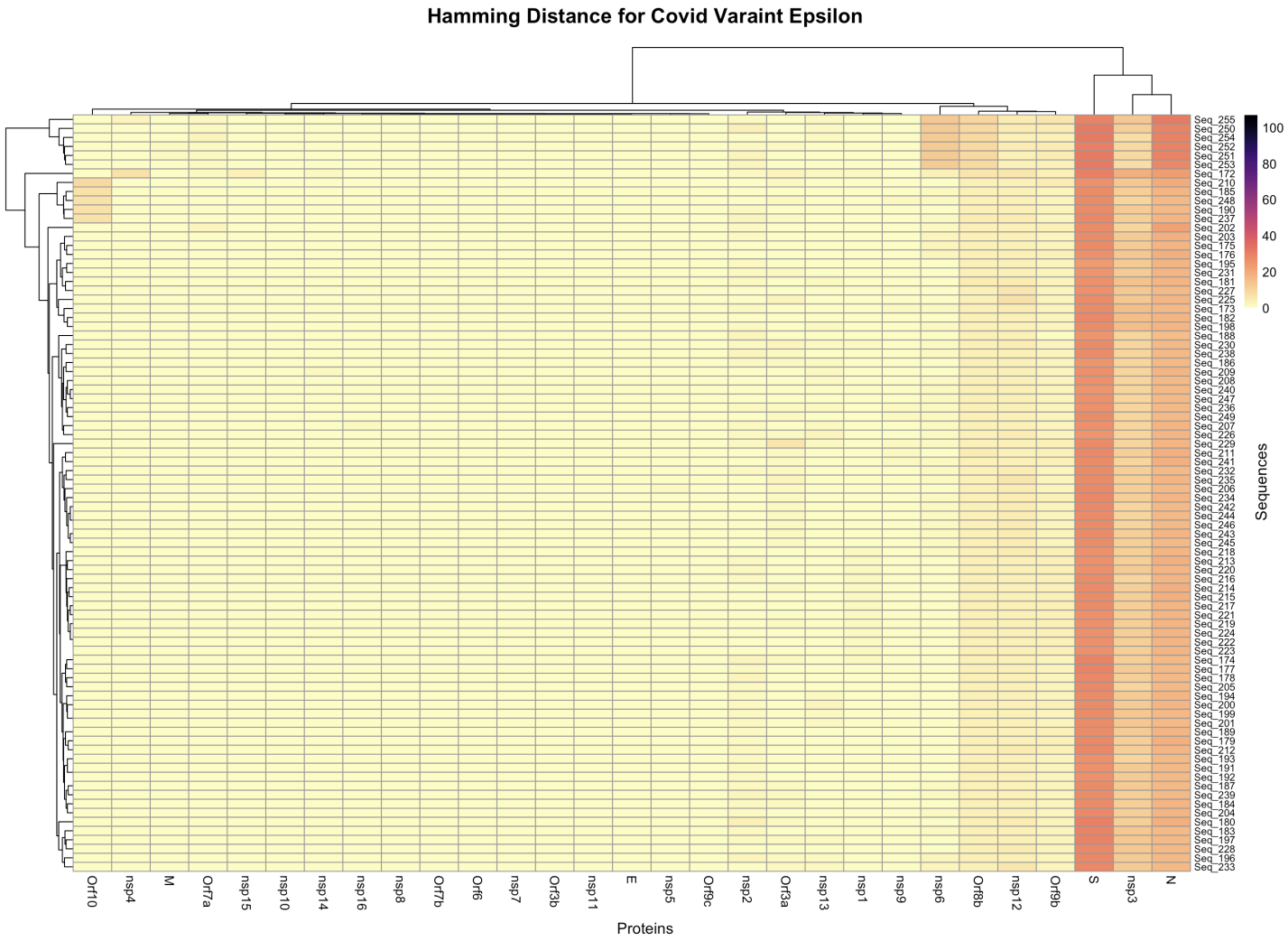


Figure 30: Hamming Distance of Aligned Epsilon Proteins to Wuhan Reference Sequence

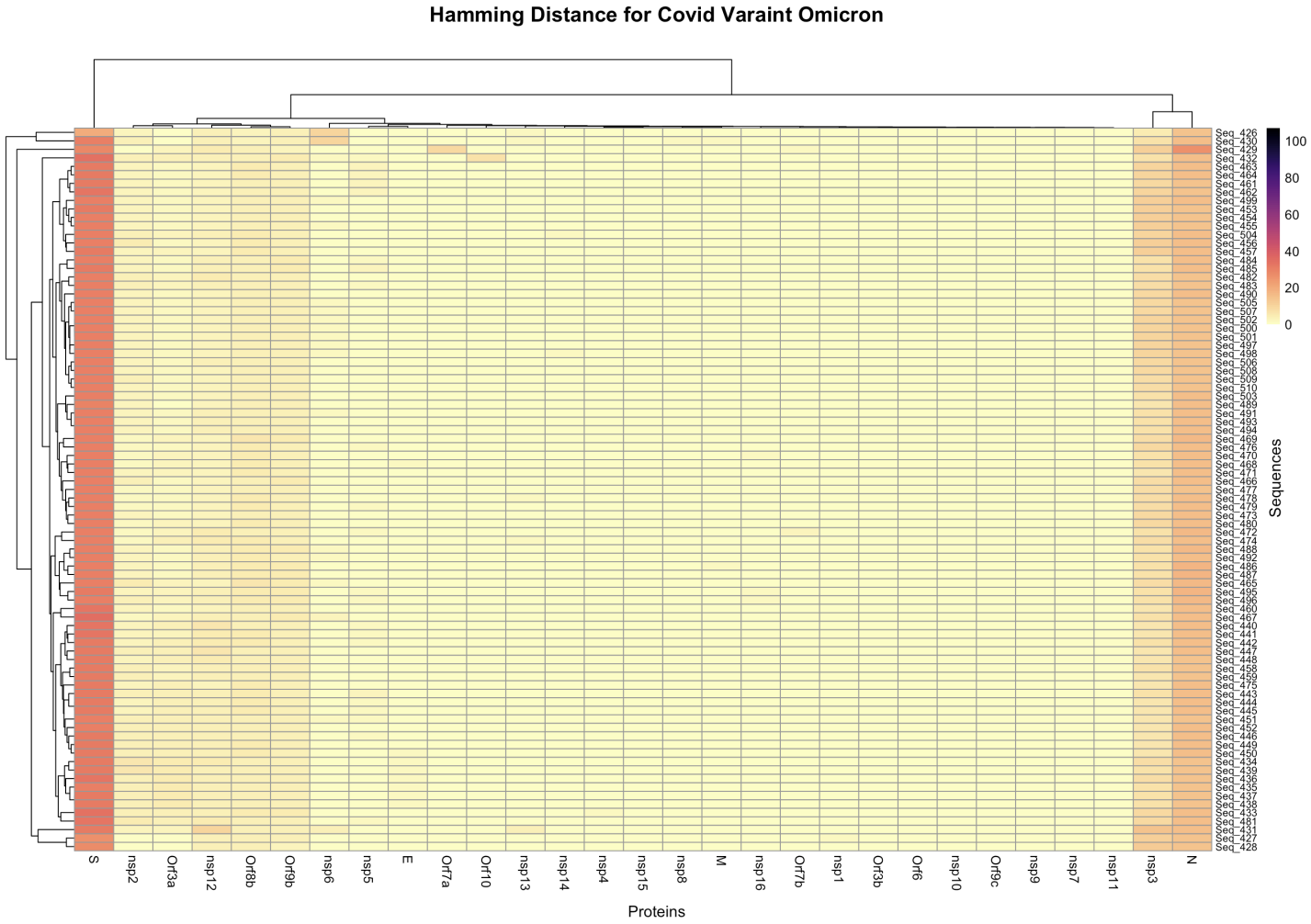


Figure 31: Hamming Distance of Aligned Omicron Proteins to Wuhan Reference Sequence

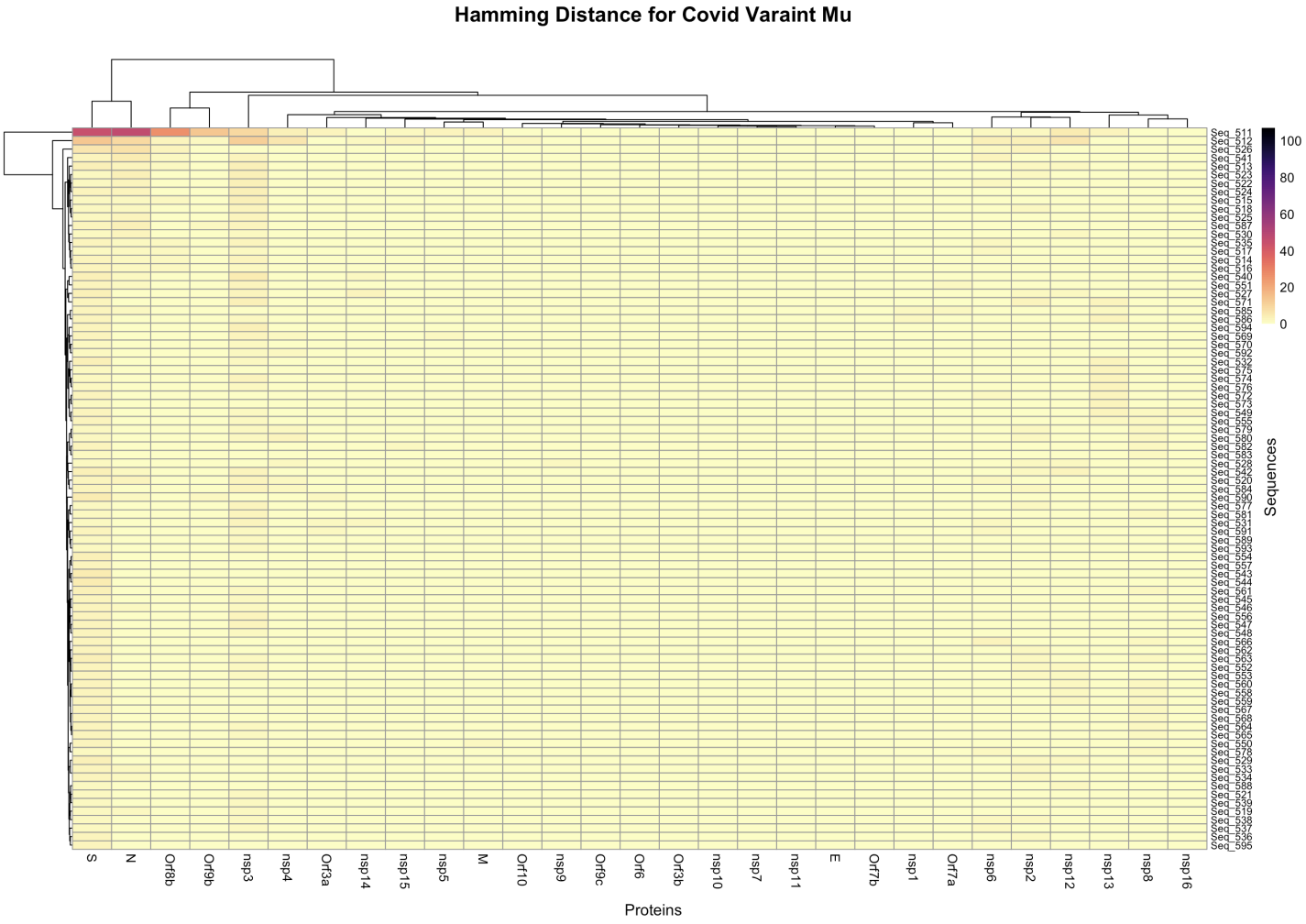


Figure 32: Hamming Distance of Aligned Mu Proteins to Wuhan Reference Sequence

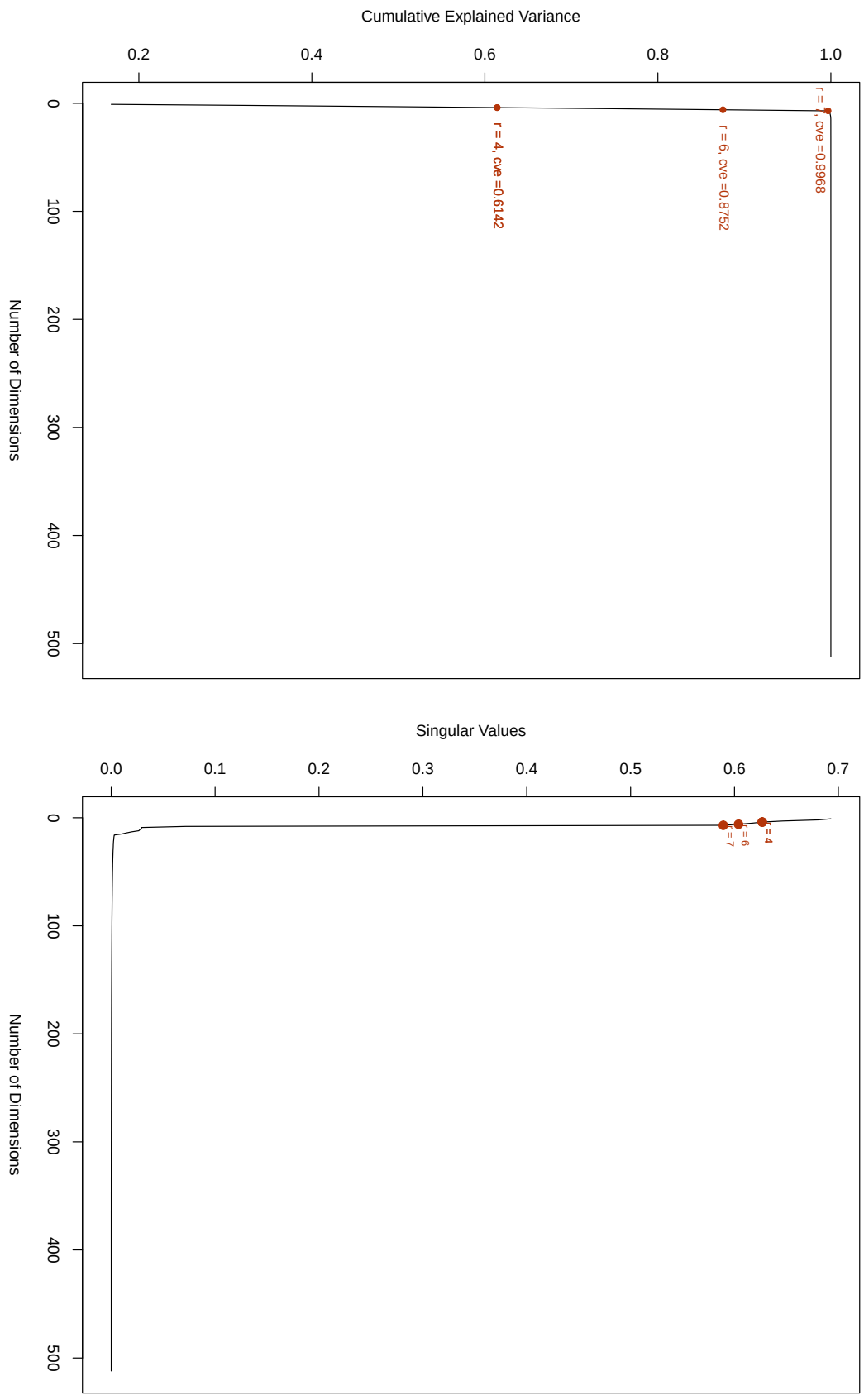


Figure 33: SVD for 581 Aligned Whole Genome Sequences

## 8.2 Code

### 8.2.1 Pre-Processing

```python
import argparse
import h5py
import pandas as pd

#### PARSING PIPED ARGUMENTS ####
description_text = (
    "Script for loader .feather file of dataframe of gisaid sequences and then "
    "wrangles df to a list of sequences indexing the protein type by the ARRAY_JOB_ID"
)
parser = argparse.ArgumentParser(description=description_text)
parser.add_argument(
    "-in",
    "--featherIn",
    type=str,
    required=True,
    help="Path to the input feather file with original DNA sequences.",
)
parser.add_argument(
    "-out",
    "--hdf5OutputPath",
    type=str,
    required=True,
    help="The root Path to save the output sequences in HDF5 format. ie /user/file/",
)
parser.add_argument(
    "-aid",
    "--arrayId",
    type=int,
    required=True,
    help="adjusted array job id. This value indexes the dataframe column allowing different proteins to
        get different jobs",
)
args = parser.parse_args()
protien_df = pd.read_feather(args.featherIn)
proteins = ['nsp1', 'nsp2', 'nsp3', 'nsp4', 'nsp5', 'nsp6', 'nsp7', 'nsp8', 'nsp9', 'nsp10', 'nsp11', '
    nsp12', 'nsp13', 'nsp14', 'nsp15', 'nsp16', 'E', 'M', 'S', 'N', 'Orf3a', 'Orf3b', 'Orf6', 'Orf7a', '
    Orf7b', 'Orf8b', 'Orf9b', 'Orf9c', 'Orf10']

protein_df_single_p = protien_df[proteins[args.arrayId]].tolist()
DNA = protein_df_single_p

# Write the mutated sequences to an HDF5 file:
with h5py.File(args.hdf5OutputPath, "w") as hdf:
    hdf.create_dataset("sequence", data=DNA)

print(f"Written to {args.hdf5OutputPath}")
```

Listing 1: Protein Pre-processing Script - `protein_preproc.py`

```python
import argparse
import h5py
import pandas as pd

#### PARSING PIPED ARGUMENTS ####
description_text = (
    "Script for loader .feather file of dataframe of gisaid sequences and then "
    "wrangles df to a list of sequences indexing the variant type by the ARRAY_JOB_ID"
)
parser = argparse.ArgumentParser(description=description_text)
parser.add_argument(
    "-in",
    "--featherIn",
    type=str,
    required=True,
    help="Path to the input feather file with original DNA sequences.",
)
parser.add_argument(
    "-out",
    "--hdf5OutputPath",
    type=str,
    required=True,
    help="The root Path to save the output sequences in HDF5 format. ie /user/file/",
)
parser.add_argument(
    "-aid",
    "--arrayId",
    type=int,
    required=True,
    help="adjusted array job id. This value indexes the orfConfig.yml allowing different variants to get
        different jobs",
)
args = parser.parse_args()
variant_df = pd.read_feather(args.featherIn)
variants = ["alpha", "beta", "gamma", "delta", "epsilon", "omicron", "mu", "lambda"]

variant_df_single_v = variant_df[variant_df["variant"] == variants[args.arrayId]]
variant_df_single_v = variant_df_single_v["sequence"]
variant_list_single_v = variant_df_single_v.tolist()
DNA = variant_list_single_v

# Write the mutated sequences to an HDF5 file:
with h5py.File(args.hdf5OutputPath, "w") as hdf:
    hdf.create_dataset("sequence", data=DNA)

print(f"Written to {args.hdf5OutputPath}")
```

Listing 2: Variant Pre-processing Script - `variant_preproc.py`

### 8.2.2 Embedding

```
#!/bin/bash
#SBATCH --account=mayocancerai
#SBATCH --job-name=dhintz_proteins_npat_400_array
#SBATCH --mail-type=ALL
#SBATCH --mail-user=dhintz1@uwyo.edu
#SBATCH --time=1-00:00:00
#SBATCH --partition=beartooth-hugemem
#SBATCH --error=slurms/%x_%A.err
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --array=0-28
#SBATCH --mem=60G
#SBATCH --output=/pfs/tc1/project/mayocancerai/GenSLM/job_array_out/arrays_ex01_%A_%a.out

GENSLM_PATH="/pfs/tc1/project/mayocancerai/GenSLM"
INPUT_FEATHER="${GENSLM_PATH}/proteins_all_seqs_df.feather"
POST_PROC_HDF5_OUT="${GENSLM_PATH}/data_protein/prot${SLURM_ARRAY_TASK_ID}_seq_400_patients.h5"
EMBEDDED_HDF5_OUT="${GENSLM_PATH}/data_protein/prot${SLURM_ARRAY_TASK_ID}_emb_400_patients.h5"

module load arcc/1.0 miniconda3/23.11.0
conda activate /pfs/tc1/project/mayocancerai/mayocancerai

# Explicitly use the Python interpreter from the specific Conda environment
PYTHON_EXEC="/pfs/tc1/project/mayocancerai/mayocancerai/bin/python"

${PYTHON_EXEC} ${GENSLM_PATH}/protein_preproc.py -in "${INPUT_FEATHER}" -out ${POST_PROC_HDF5_OUT} --
    arrayId ${SLURM_ARRAY_TASK_ID}
if [ $? -ne 0 ]; then
  echo "Failed to execute protein_preproc.py for Protein ${SLURM_ARRAY_TASK_ID}."
  exit 1
fi

${PYTHON_EXEC} ${GENSLM_PATH}/GenSLM.py -in ${POST_PROC_HDF5_OUT} -out ${EMBEDDED_HDF5_OUT}
if [ $? -ne 0 ]; then
  echo "Failed to execute GenSLM.py for Protein ${SLURM_ARRAY_TASK_ID}."
  exit 1
fi

echo "Emebedding for Protein ${SLURM_ARRAY_TASK_ID} completed successfully!"
```

Listing 3: Protein Embedding Script - `embed_prot_aj.sh`

```
#!/bin/bash
#SBATCH --account=mayocancerai
#SBATCH --job-name=dhintz_varaints_npat_400_array
#SBATCH --mail-type=ALL
#SBATCH --mail-user=dhintz1@uwyo.edu
#SBATCH --time=1-00:00:00
#SBATCH --partition=beartooth-hugemem
#SBATCH --error=slurms/%x_%A.err
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --array=0-7
#SBATCH --mem=60G
#SBATCH --output=/pfs/tc1/project/mayocancerai/GenSLM/job_array_out/arrays_ex01_%A_%a.out

GENSLM_PATH="/pfs/tc1/project/mayocancerai/GenSLM"
INPUT_FEATHER="${GENSLM_PATH}/variant_dfs.feather"
POST_PROC_HDF5_OUT="${GENSLM_PATH}/data_variant/var${SLURM_ARRAY_TASK_ID}_seq_400_patients.h5"
EMBEDDED_HDF5_OUT="${GENSLM_PATH}/data_variant/var${SLURM_ARRAY_TASK_ID}_emb_400_patients.h5"

module load arcc/1.0 miniconda3/23.11.0
conda activate /pfs/tc1/project/mayocancerai/mayocancerai

# Explicitly use the Python interpreter from the specific Conda environment
PYTHON_EXEC="/pfs/tc1/project/mayocancerai/mayocancerai/bin/python"

${PYTHON_EXEC} ${GENSLM_PATH}/variant_preproc.py -in "${INPUT_FEATHER}" -out ${POST_PROC_HDF5_OUT} --
    arrayId ${SLURM_ARRAY_TASK_ID}
if [ $? -ne 0 ]; then
  echo "Failed to execute variant_preproc.py for Variant ${SLURM_ARRAY_TASK_ID}."
  exit 1
fi

${PYTHON_EXEC} ${GENSLM_PATH}/GenSLM.py -in ${POST_PROC_HDF5_OUT} -out ${EMBEDDED_HDF5_OUT}
if [ $? -ne 0 ]; then
  echo "Failed to execute GenSLM.py for Variant ${SLURM_ARRAY_TASK_ID}."
  exit 1
fi

echo "Emebedding for Variant ${SLURM_ARRAY_TASK_ID} completed successfully!"
```

Listing 4: Variant Embedding Script - `embed_var_aj.sh`

```python
#!/usr/bin/env python
import argparse
import h5py
import numpy as np
from genslm import GenSLM, SequenceDataset
from torch.utils.data import DataLoader
import torch
import os

description_text = "Script for embedding mRNA sequences using GenSLM and saving in HDF5 format."
parser = argparse.ArgumentParser(description=description_text)
parser.add_argument('-in', '--hdf5In', type=str, required=True, help='Path to the input HDF5 file containing mRNA sequences.')
parser.add_argument('-out','--hdf5Out', type=str, default=None, help='Path to save the embedded sequences in HDF5 format. If not provided, it will overwrite the input HDF5.')
args = parser.parse_args()

model = GenSLM("genslm_25M_patric", model_cache_dir="/project/mayocancerai/GenSLM") # Initialize GenSLM
model.eval()

def embed_with_genslm(seq):
    dataset = SequenceDataset([seq], model.seq_length, model.tokenizer)
    dataloader = DataLoader(dataset, batch_size=1)
    with torch.no_grad():
        for batch in dataloader:
            outputs = model(batch["input_ids"], batch["attention_mask"], output_hidden_states=True)
            emb = outputs.hidden_states[0].detach().cpu().numpy()
            emb = np.mean(emb, axis=1)
    return emb

embedded_sequences = []
with h5py.File(args.hdf5In, 'r') as hdf:
    sequences = hdf['sequence'][:]  # Assuming all sequences are stored in a dataset named 'sequence'
    for seq_bytes in sequences:
        seq = seq_bytes.decode('utf-8')  # Assuming sequences are stored as bytes
        emb = embed_with_genslm(seq)
        embedded_sequences.append(emb)

embedded_sequences_array = np.vstack(embedded_sequences) # Convert list of embeddings to a numpy array (n x 512)
output_hdf5_path = args.hdf5Out

with h5py.File(output_hdf5_path, 'w') as hdf: # Save the embedded sequences array to HDF5
    hdf.create_dataset('embedded_sequences', data=embedded_sequences_array)

print(f"Written to {output_hdf5_path}")
```

Listing 5: Core GenSLM Embedding Script - `GenSLM.py`

# 9 Glossary

- **Amino Acids**: Organic compounds that combine to form proteins, serving as the building blocks of life.
- **Bash Pipelines**: A bash script or series of scripts part of a single execution that passes variables and/or data between sub-shell environments, often combining multiple coding languages.
- **BP (Base Pair)**: A unit consisting of two nucleotides on opposite complementary DNA or RNA strands that are connected via hydrogen bonds.
- **Codons**: Triplets of nucleotides in mRNA that specify which amino acid will be added next during protein synthesis.
- **Gaps**: In sequence alignments, spaces inserted to align sequences optimally; gaps can represent deletions or insertions.
- **Hamming's Distance**: A measure of the number of substitutions required to change one string into another, used especially in genetics to determine the difference between sequences.
- **High Coverage**: In genomic sequencing, refers to the number of times a particular region of the genome has been sequenced, indicating the reliability of the data.
- **Learner**: In the machine learning context, a learner refers to an algorithm or model that learns from data to make predictions or decisions.
- **MSA (Multiple Sequence Alignment)**: The alignment of three or more genomic sequences (protein, DNA, or RNA) to achieve maximal matching, used to identify regions of similarity that may indicate functional, structural, or evolutionary relationships.
- **Mutations**: Changes in the nucleotide sequence of the genetic material of an organism, which may alter the function or activity of gene products.
- **NSP (Non-structural Protein)**: Proteins encoded by a virus that are not part of its structural components but are crucial for its replication and usually for subverting the host's immune response.
- **Nucleotide**: The basic building block of DNA and RNA, consisting of a base attached to a sugar-phosphate backbone.

- **ORFs (Open Reading Frames)**: Sequences in nucleic acids that potentially encode proteins, starting with a start codon and ending with a stop codon (also known as a coding region).
- **Protein**: Large biomolecules, or macromolecules, consisting of one or more long chains of amino acid residues, essential for the structure, function, and regulation of the body's cells, tissues, and organs.
- **RNA (Ribonucleic Acid)**: A nucleic acid present in all living cells. Its principal role is to act as a messenger carrying instructions from DNA for controlling the synthesis of proteins.
- **SLURM Jobs Arrays**: A feature in SLURM (a job scheduler for Linux) that allows submission of multiple similar jobs with a single command using an array index.
- **SP (Structural Protein)**: Proteins that form the structure of an organism, such as those making up the cell cytoskeleton or a virus capsid.
- **Whole Genomes**: The complete set of DNA or RNA sequences of an organism, encompassing all its genes (coding regions) and non-coding regions.