Persistent Homology on Streaming Data

Anindya Moitra, Nicholas O. Malott, and Philip A. Wilsey
Dept. of EECS, University of Cincinnati, Cincinnati, OH 45221, USA
Email: moitraaa@mail.uc.edu, malottno@mail.uc.edu, philip.wilsey@uc.edu

***Abstract*—This paper introduces a framework to compute *persistent homology*, a principal tool in Topological Data Analysis, on potentially unbounded and evolving data streams. The framework is organized into online and offline components. The online element maintains a summary of the data that preserves the topological structure of the stream. The offline component computes the persistence intervals from the data captured by the summary. The framework is applied to the detection of horizontal or reticulate genomic exchanges during the evolution of species that cannot be identified by phylogenetic inference or traditional data mining. The method effectively detects reticulate evolution that occurs through reassortment and recombination in large streams of genomic sequences of Influenza and HIV viruses.**

***Index Terms*—Persistent homology, Topological Data Analysis, data stream mining, viral evolution, computational genetics.**

I. Introduction

Persistent homology is a powerful tool for data analysis that extracts information about topological features such as connected components, loops, and voids in a point cloud at different spatial connectivities [1]–[5]. Given a set of data points, persistent homology creates a sequence of increasingly connected and nested subspaces. The *persistence* of each topological feature is then recorded as its lifespan as the feature appears and subsequently disappears through the sequence of nested subspaces. As a result, persistent homology segregates significant topological features from noise based on their lifespans, and often discovers insight not discernible by conventional methods of data mining [6]–[11]. The capabilities of persistent homology come at the cost of its high computational complexity. In particular, the memory required for the computation grows exponentially with the number and dimension of data objects in the input point cloud [5].

A data stream is a potentially unbounded sequence of continuously arriving data objects that cannot be stored in the memory available to a computer. Since random access to the data is unavailable, algorithms dealing with data streams must make only one, or very few, passes through the data [12]. Moreover, the data generation process can be non-stationary, resulting in a data stream that *evolves* over time. Such unique challenges of data stream mining coupled with the high computational cost of persistent homology are the primary reasons why persistent homology has not yet been applied to data stream mining.

Contributions: This paper is a first step to bridge the gap between data stream mining and persistent homology. We develop a general-purpose framework for computing persistent homology on potentially unbounded streaming data. Consistent with the standard computational paradigm [13] for processing data streams, our approach consists of two principal components: (i) *online*, and (ii) *offline*.

The online component involves continuously partitioning the data stream into small clusters (*microclusters*) and updating a *summary* of the stream with the help of a data structure, known as the *feature vector* [13]–[17]. This data structure is designed to maintain a bounded space summary of the stream for preserving the "meaning" of the original data points without the need of actually storing them. Moreover, if the data stream evolves over time, the older microclusters are *faded out* to assign more importance to the recent data points [13].

The offline component comprises the computation of persistent homology at fixed intervals on the centers of the microclusters maintained during the online step. Persistent homology captures the topological structures of the data by computing a set of intervals (or lifespans) during which those structures exist. The lifespans of topological features are the final outcomes of our framework that are displayed in one of the standard output formats such as barcodes or persistence diagrams. Thus, by continuously monitoring the output of the framework, one can visualize the current *state* of the stream and detect any changes in the topological properties with the progression of the stream.

The proposed framework is applied to streams of genomic sequences in order to detect *reticulate genomic exchanges* during the evolution of viruses. While the phylogenetic tree structure [18] is the accepted paradigm to represent the vertical or clonal evolution of species, it cannot effectively capture horizontal or reticulate events that typically occur through species hybridization, lateral gene transfer, or recombination and reassortment. Persistent homology has been shown to provide a comprehensive representation of both vertical and horizontal evolution at the same time [9]. While our framework can be applied to any data stream, the effectiveness of the approach is demonstrated by extending the study of viral evolution in [9] to *streams* of genomic sequences of Influenza and HIV viruses. By monitoring the output of this framework, one can identify the occurrence of reticulate events during the evolution of organisms. Our approach is not limited by the length or the number of genomic sequences, and the insight it derives is not available to traditional methods of data mining or the classical approaches to the study of evolution [9].

Support for this work was provided in part by the National Science Foundation under grants ACI–1440420 and IIS–1909096.

II. Background and Related Work

Background: The first part of this section briefly explains the introductory ideas of persistent homology. An intuitive visual presentation of the basic concepts can be viewed at [19]. A formal exposition of the subject is available at [20], [21].

Homology is a way of counting the topological features of a space. In topological data analysis, a given set of data points is assumed to be sampled from an underlying space $\mathbb{S}$ that has an unknown probability distribution. Computing the homology of such arbitrary topological spaces is difficult. To overcome this obstacle, the topology of $\mathbb{S}$ is approximated by a combinatorial structure, called a *complex*, for which homology can be computed algorithmically [5]. Simplicial, cubical, and CW complexes [22] are some of the commonly used complexes. Since the simplicial complex is the most widely used with a richer theoretical foundation than others [5], persistent homology computed from simplicial complexes is examined in this paper.

A simplicial complex K is a set of points, edges, triangles, tetrahedrons and so on. K comprises all possible subsets that can be constructed from the distinct points in K. Each topological feature of a simplicial complex is assigned a *dimension* p. The set of p-dimensional features forms a *group*, called the p-th *homology group* H_p.

A subset $K_i \subseteq K$ is called a *subcomplex* if K_i itself is a simplicial complex. A *filtration* of a complex K is a sequence of nested subcomplexes $\varnothing = K_0 \subseteq K_1 \subseteq K_2 \subseteq ... \subseteq K_n = K$. A complex with a filtration is called a *filtered complex*.

For the computation of persistent homology, a filtered simplicial complex is constructed on a given set of data points. Each subcomplex in the filtration is associated to a distinct value of a scale parameter ε. Since the topological structure of each subcomplex is usually different from those of other complexes in the filtration, we say that the topology of a simplicial complex changes with the scale parameter ε. At $\varepsilon = 0$, the simplicial complex is a set of disconnected points. As ε increases, the points start becoming connected to one another by edges, and subsequently form triangular faces, tetrahedrons, and so on. With increasing ε, the connected components become longer, existing connected components are merged into one another, holes and voids appear (or, *are born*) and eventually get filled (or, *die*). Persistent homology tracks the *birth* and *death times* of these topological features as ε increases from 0 to a user-specified threshold ε_{max}. A topological feature is born at ε_{birth}, and dies at ε_{death}.

The output of persistent homology is a set of pairs of real numbers $(\varepsilon_{birth}, \varepsilon_{death})$. The difference between the ε_{death} and ε_{birth} times is called the lifespan or *persistence* of a topological feature. The lifespans of significant topological features are much longer than those of noise. The pairs $(\varepsilon_{birth}, \varepsilon_{death})$ can be displayed as a set of lines, called a *barcode*, or as a 2-dimensional scatter plot, known as a *persistence diagram*.

Related Work: To the best of our knowledge, there is no existing framework or technique for computing persistent homology on data streams. Here, we briefly outline the literature our framework is based on. The method for continuous summarization of the stream is based on the online component of data stream clustering algorithms such as DenStream [16], CluStream [15], ClusTree [17], and streaming k-means [12]. It has been shown that when a set of data objects is partitioned into another set of small clusters, replacing the original data objects by the cluster centers results in a bounded-error approximation of the initial data [23], [24]. In the same spirit, the centers of the microclusters produced during the summarization of the stream accurately represent an abstract of the data stream and preserve its topological properties. This is demonstrated in Section IV of this paper. The offline component of our framework, that comprises the construction of simplicial complexes and computation of persistence intervals, is based on the works of [21], [25], [26].

It is worth noting that Kerber *et al.* [27] developed a streaming algorithm for the matrix reduction procedure that generates the persistence intervals. The procedures described in [27] construct a pipeline for computing persistence of large simplicial *towers* using streaming algorithms of bounded space and time. The input to their algorithm is a large *tower*, a generalized representation of a filtered complex, that is typically stored on the disk. However, their solution can not be directly applied to the real time streaming applications because they do not include any strategy to bound the size of the complex itself as data points continuously arrive from a potentially infinite stream. The solution of [27] can be regarded as a bounded memory algorithmic pipeline for computing persistence from a large data set stored on the disk.

III. Persistent Homology on Data Stream

This section presents the approach for computing persistent homology on streaming data that is outlined in Section I. The presentation is divided into two parts, namely: (i) the *data summarization* or the online component, and (ii) the *computation of persistence intervals* or the offline component.

A. Data Summarization

The objective of the online component is to maintain a *bounded summary* or *abstract* of a potentially unbounded stream. It is important to bound the number of data objects in the summary up to the limit that will result in a simplicial complex (in the subsequent offline step) that can be accommodated within the available memory.

The summary of the data stream is maintained by utilizing the concept of microclusters commonly employed in stream clustering algorithms. In this section, we outline a generic data summarization model that is similar to the online component of several algorithms such as DenStream [16], CluStream [15], ClusTree [17], and streaming k-means [12]. Each algorithm, however, adds to or modifies the generic model to include additional features and capabilities. In practice, the choice of the data summarization model may vary depending on the nature of the stream and the application.

In order to process evolving streams, each data point is assumed to have a weight $\beta^{-\lambda\Delta t}$ that decreases exponentially with the time (Δt) elapsed since its arrival. $\beta > 0$ is often set to 2. $\lambda > 0$ is a user defined input parameter, called the *decay parameter*. Higher values of λ denote lower importance of the historical data points compared to the recent ones.

Microclusters are small groups of *similar* data points. A microcluster at time t for a group of points $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ with arrival time stamps $T_1, T_2, ..., T_n$ is defined as $\{w, \mathit{LS}, \mathit{SS}\}$, where $w = \sum_{i=1}^{n} \beta^{-\lambda(t-T_i)}$ is the weight, $\mathit{LS} = \sum_{i=1}^{n} \beta^{-\lambda(t-T_i)}\,\mathbf{x}_i$ is the weighted linear sum, and $\mathit{SS} = \sum_{i=1}^{n} \beta^{-\lambda(t-T_i)}\,\mathbf{x}_i^2$ is the weighted squared sum of the points in the microcluster. The tuple $\{w, \mathit{LS}, \mathit{SS}\}$ is called the feature vector of a microcluster. From the components of the feature vector, the center μ and radius r of a microcluster are computed as $\mu = \frac{\mathit{LS}}{w}$, and $r = \sqrt{\frac{\mathit{SS}}{w} - \left(\frac{\mathit{LS}}{w}\right)^2}$. Microclusters have a user defined maximum radius r_{max}.

A new data point $\mathbf{x}_j$ can be added to a microcluster by incrementally updating its feature vector: $w \leftarrow w + 1$, $\mathit{LS} \leftarrow \mathit{LS} + \mathbf{x}_j$, and $\mathit{SS} \leftarrow \mathit{SS} + \mathbf{x}_j^2$. When a new data point $\mathbf{x}$ arrives, we check if $\mathbf{x}$ can be added to its nearest microcluster c by computing the updated radius r of c. If $r \leq r_{max}$, then $\mathbf{x}$ is added to c. If $r > r_{max}$, a new microcluster is created with only one data point $\mathbf{x}$ in it.

A microcluster loses its weight during the time no new data point is added to it. If no data point is added to a microcluster $\{w, \mathit{LS}, \mathit{SS}\}$ during a time interval Δt, it is updated to $\{\beta^{-\lambda\Delta t} w,\ \beta^{-\lambda\Delta t}\mathit{LS},\ \beta^{-\lambda\Delta t}\mathit{SS}\}$. If the weight of a microcluster falls below a threshold w_{min}, it is considered to be outdated. We limit the growth of the number of microclusters in memory by eliminating those that did not receive new data points long enough to have become outdated. Assuming $\beta = 2$, the minimum time Δt_c required for a microcluster to decay into an outdated one can be computed from the equation $1 + 2^{-\lambda\Delta t_c} w_{min} = w_{min}$ as $\Delta t_c = \left\lceil \frac{1}{\lambda} \log\left(\frac{w_{min}}{w_{min}-1}\right) \right\rceil$. The weights of all microclusters are checked every Δt_c time period to prune out those that have become outdated. The microclusters are organized in a tree structure to speed up the search for the nearest microcluster of a new data point $\mathbf{x}$.

The proposed framework for computing persistent homology (up to dimension p and scale parameter ε) on streaming data based on the data summarization model described above is outlined in Algorithm 1. The maximum time required to process each new data point from the stream during the online component is linear in the number of microclusters. The offline component that involves the computation of persistence intervals is described in the following subsection.

Algorithm 1 Streaming Persistent Homology

Given: $stream, r_{max}, w_{min}, \lambda, p, \varepsilon$

$\Delta t_c = \left\lceil \frac{1}{\lambda} \log\left(\frac{w_{min}}{w_{min}-1}\right) \right\rceil$
while new data points arrive from $stream$ **do**
  Read the next point $\mathbf{x}$ from $stream$ at current time t.
  Find the microcluster c nearest to $\mathbf{x}$.
  Compute the updated radius, r of c, as if $\mathbf{x}$ is in c.
  if $r \leq r_{max}$ **then**
    Add $\mathbf{x}$ to c
  else
    Create a new microcluster using $\mathbf{x}$
  end if
  if (t modulo Δt_c) $= 0$ **then**
    for each microcluster c **do**
      if $w < w_{min}$ **then**
        Delete c
      end if
    end for
  end if
  if user requests the output of persistent homology **then**
    Compute persistence intervals
  end if
end while

B. Computation of Persistence Intervals

Every time the output of persistent homology is requested, a maximal simplicial complex K is constructed at $\varepsilon = \varepsilon_{max}$ on the set of centers $\mathcal{C}$ of the microclusters maintained during the online step. Every simplex σ of K is defined to have a weight ω that is the maximum of the lengths of all the edges in σ. A 0-simplex $\{\mathbf{x}\}$ has $\omega = 0$, and a 1-simplex $\{\mathbf{x}, \mathbf{y}\}$ has $\omega = dist(\mathbf{x}, \mathbf{y})$, the distance between the points $\mathbf{x}$ and $\mathbf{y}$. A total ordering is imposed on the simplices of K such that:

- the simplices are sorted according to their weights, and
- a face of a simplex precedes the simplex.

By assigning the weights and imposing the total ordering on the simplices, the filtration of K is extracted. The simplicial complex K is then called a *weight-filtered complex* [25].

Let the simplices with the total ordering imposed on them are denoted by $\sigma_1, \sigma_2, ..., \sigma_{n_K}$, where n_K is the total number of simplices in K. A square matrix ∂, called the *boundary matrix*, of order n_K is constructed as:

$$\partial[i,j] = \begin{cases} 1, & \text{if } \sigma_i \text{ is a co-dimension one face of } \sigma_j \\ 0, & \text{otherwise.} \end{cases}$$

The columns and rows of ∂ represent the simplices of the filtration arranged according to the total order. The boundary (or, co-dimension one face) of a simplex is recorded in its column (by a 1 in the corresponding row).

The boundary matrix ∂ is reduced to another $0-1$ matrix ∂_R by Algorithm 2, called the *standard algorithm* [21], [26]. Let $low(j)$ be the row index of the lowest 1 (*i.e.*, the highest row index of a 1) in column j. If the entire column is zero, then $low(j)$ is undefined. We scan the columns of ∂ from left to right, and when we reach a column j such that there is another column $j_0 < j$ with $low(j_0) = low(j)$, the column j_0 is added to j. The boundary matrix is *reduced* when $low(j_0) \neq low(j)$ for any two non-zero columns $j_0 \neq j$. The worst case run time of the standard algorithm is cubic in the number of simplices n_K. In practice, the algorithm has displayed a quasi-linear

behavior on real-world data [27]. A number of solutions have been designed to improve the worst case run time [5], [27].

Algorithm 2 The standard algorithm

```
for j = 1 to n_K do
    while there exists j_0 < j with low(j_0) = low(j) do
        add column j_0 to column j
    end while
end for
```

Once the boundary matrix ∂ is reduced to ∂_R, the birth and death times of topological features are recorded from ∂_R, and are plotted as barcodes or persistence diagrams. The sequence of barcodes or persistence diagrams, displayed at fixed time intervals during the progression of the stream, is the final outcome of our framework.

IV. Experiments

In this section, the proposed framework is applied to identify *reticulate genomic exchanges* during the evolution of two different types of viruses: Influenza A and HIV. In addition, we demonstrate the effect of the decay parameter λ on the summary of the topological features maintained in memory with the help of an artificially generated data set.

For maintaining a summary of the stream, the *online* component of *ClusTree* [17], a well-established data stream clustering algorithm, is used. It is worth mentioning that this work is not tied to ClusTree, and could have used any other stream clustering algorithm. The choice of ClusTree is motivated by some of its additional advantages, such as:

- the ability to process very fast as well as slower streams,
- a dynamic size of the data summarization model that adapts to the stream speed, and
- the ability to handle concept drift and outliers.

The centers of the microclusters maintained by ClusTree constitute a summary of the stream. Persistent homology is computed on the set of microcluster centers during the offline step at regular intervals. Below is a list of the implementations used for the experiments described in this paper.

- *ClusTree:* the `R` interface [28] to the algorithm implemented in `Java` for Massive Online Analysis [29].
- *Persistent Homology:* the `R` interface [30] to the GUDHI library [31], [32], written in `C++`.

For processing the streams of viral genomic sequences, we use $maxHeight = 10$ (for the maximum height of the tree) and $horizon = 450$ (for the range of the time window) in the implementation of ClusTree. The centers of the microclusters are transformed into a Vietoris–Rips (VR) complex [25] each time the persistence intervals are computed. The VR complex, a type of simplicial complex, is the most widely used and the only practical complex for higher dimensional data. The execution times reported in the following subsections are captured on a computer with an Intel(R) Core(TM) i7-3630QM CPU @ 2.40 GHz and 8 GB of memory, and are averaged over 5 runs. The data sets used for the experiments were stored in the hard disk of the computer, and were read as file streams

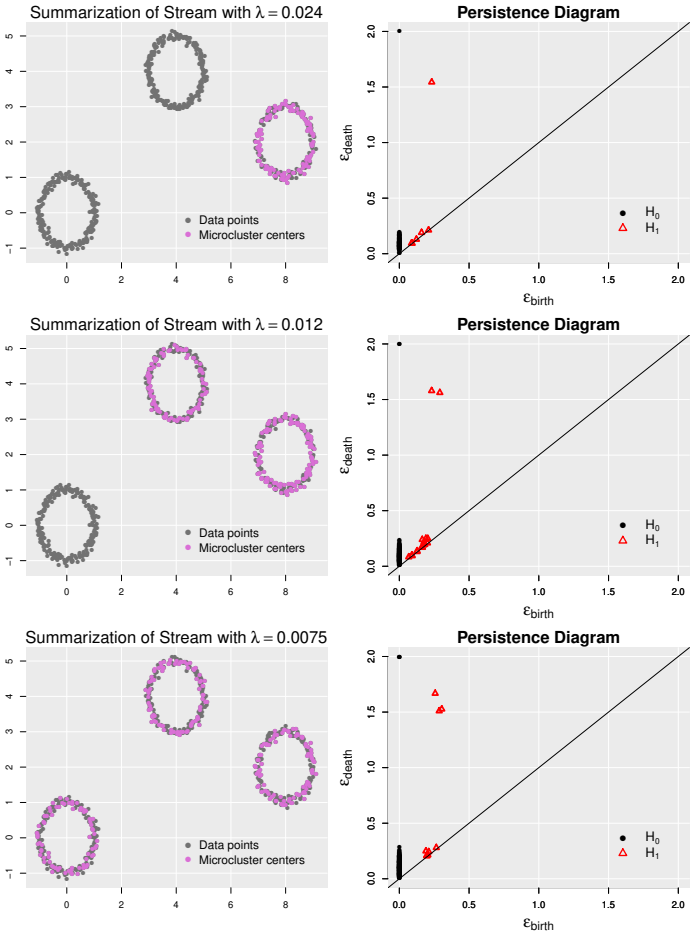

Fig. 1: Effect of the decay parameter λ on the retention of topological features in the data summary

(*i.e.*, one data point at a time). The performance and accuracy of our framework are demonstrated by the execution times, memory usage and its overall effectiveness in the detection of reticulate events during the evolution of viruses.

A. Effect of the Decay Parameter on Data Summary

ClusTree uses an exponential time-dependent decay function $f(\Delta t) = 2^{-\lambda \Delta t}$ to fade out existing microclusters with the time elapsed (Δt) since their creation. The decay parameter $\lambda > 0$ controls the rate at which the microclusters are outdated. The higher λ is, the faster the algorithm "forgets" older data. A user can, therefore, control the size of the data summarization model by varying the decay parameter λ. In terms of the topological structures of the stream, it means that the user can control the storage of the number of topological features in memory with the help of λ. A larger λ shortens the "history" of topological features maintained in memory, whereas a smaller λ expands the "history". This is illustrated in Figure 1 with the help of a synthetic data stream having a total of three significant topological features (*i.e.*, 1-dimensional loops). The left column of Figure 1 shows the original data points and the microcluster centers at the end of the stream. The right column shows the output of persistent homology, displayed as

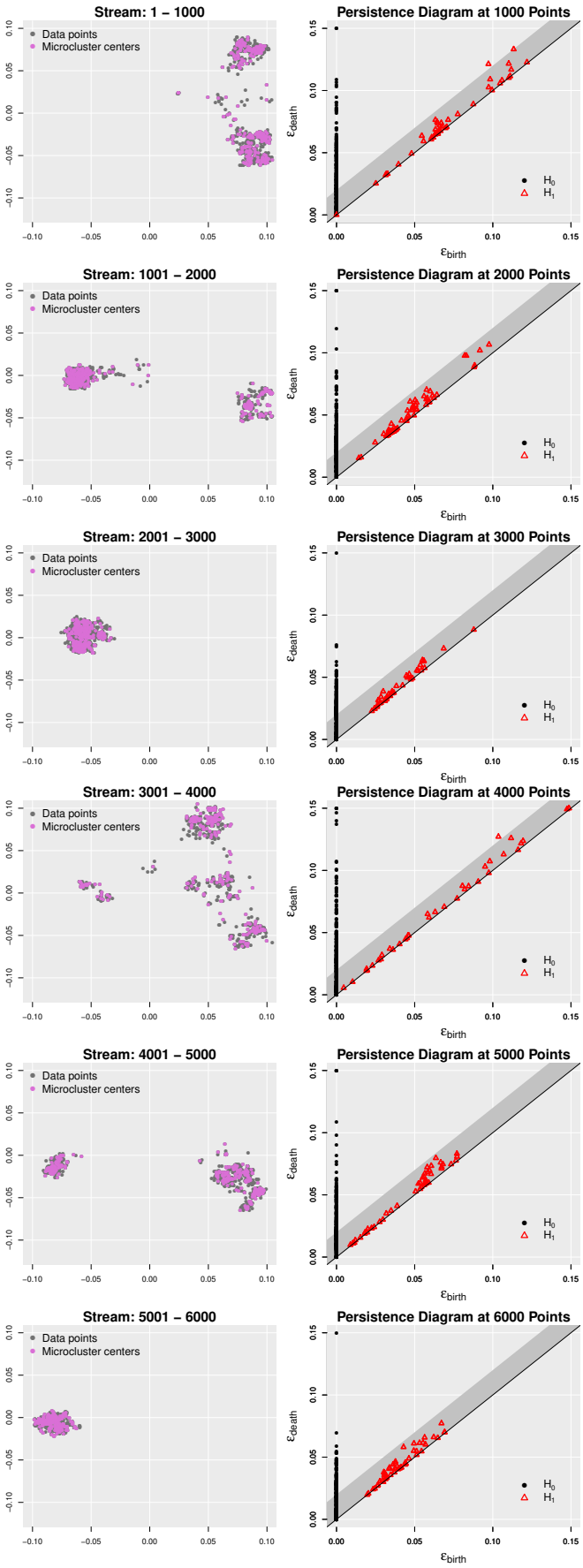

Fig. 2: Reticulate evolution in Avian Influenza virus

persistence diagrams, computed on the microcluster centers. The black dots and red triangles represent 0-dimensional connected components (H_0) and 1-dimensional loops (H_1) on the persistence diagrams. Since significant topological features have longer lifespans, their death times are much greater than their birth times. Thus, the red triangles lying far away from the diagonal (the $45°$ line that passes through the origin at $(0, 0)$) of the persistence diagram represent the loops formed by the microcluster centers. On the other hand, the points close to or on the $45°$ line represent noise.

Although not explicitly shown in Figure 1, the loops appear sequentially, one at a time, during the progression of the synthetic stream. By adjusting the value of λ, a user is able to retain as many or as few loops as desired in the summary of the stream. For example, setting $\lambda = 0.024$ retains only one loop, whereas $\lambda = 0.0075$ preserves all three loops at the end of the stream. This characteristic of the data summarization model will be useful in applications where the user intends to examine the "history" of topological features of an evolving stream beginning from different times in the past.

B. *Evolution of Influenza Virus*

As outlined in Section I, vertical or clonal evolution is usually described with the help of a phylogenetic tree structure. However, horizontal or reticulate evolution creates loops or cycles that cannot be represented by a tree. Persistent homology has been shown to provide a comprehensive mathematical structure that captures both vertical and horizontal evolutionary events at the same time [9]. The study in [9] shows that in the absence of reticulate evolution, one does not observe significant 1-dimensional (or higher) topological features in the genomic sequences of viruses. However, as horizontal evolution begins to occur through viral recombination and reassortment, the resulting loops manifest themselves as significant 1-dimensional topological features.

Individual protein segments unaffected by reassortment represent the absence of reticulate events in the genomic sequences of viruses. On the other hand, concatenated protein segments demonstrate the evidence of reassortment that causes horizontal evolution. Chan *et al.* [9] studied *separate* data sets of individual and concatenated segments to show, respectively, the absence and presence of reticulate evolution. Due to the high cost of computing persistent homology, they used a sampling strategy to select a smaller subset (called *landmark* points) of the original data.

Instead of using two separate data sets for individual and concatenated protein segments, our framework is applied to a stream of $9,350$ nucleotide sequences of Avian Influenza A that consists of both individual and concatenated segments. As a stream processing framework, our ability to compute persistent homology is not limited by the size of the stream. Avian Influenza sequences of PB2 protein were downloaded from the Influenza Virus Database [33] of the National Center for Biotechnology Information. Individual PB2 segments were concatenated using the `ape` package [34] in `R`. Among the $9,350$ sequences, the first $6,000$ sequences were individ-

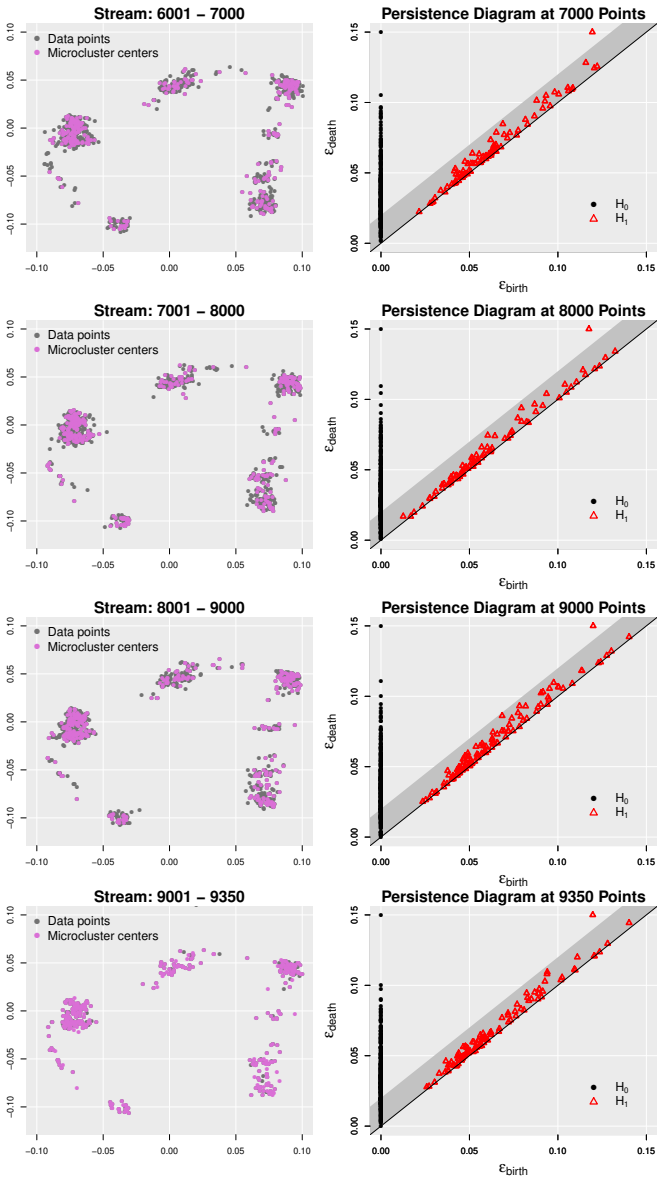

Fig. 2: Reticulate evolution in Avian Influenza virus (cont.)

ual segments, whereas the remaining sequences represented concatenated protein segments. Since the data summarization models typically work on numeric data, nucleotide sequences were mapped to numeric vectors in a 45-dimensional Euclidean space (see Section IV-D for details).

Figure 2 shows the plots of the first two dimensions of the Euclidean vectors representing the viral sequences and the persistence intervals displayed as persistence diagrams at intervals of $1,000$ data points through the end of the stream. During the first $6,000$ sequences, the viral protein segments do not form loops and do not display significant 1-dimensional topological features (H_1) in the persistence diagrams. The noisy topological features are represented by groups of red triangles on or near the diagonals of the persistence diagrams. This phase in the viral evolution represents the occurrence of only vertical genetic exchanges. After the first $6,000$ sequences, horizontal genetic exchanges begin to occur through reassortment and recombination that result in the reticulate evolution. The viral sequences affected by reassortment are represented by concatenated protein segments. Figure 2 shows that such concatenated sequences form loops that are identified by distinct red triangular points lying outside of the gray regions of noise near the diagonals of the persistence diagrams.

Progression of Stream	Data Summary Run-time	Persistent Homology Run-time	Total Run-time	No. of Micro-clusters
1 – 1000	0.098	1.909	2.007	303
1001 – 2000	0.102	1.044	1.146	308
2001 – 3000	0.097	2.925	3.022	303
3001 – 4000	0.105	1.178	1.283	301
4001 – 5000	0.098	1.008	1.106	309
5001 – 6000	0.099	3.137	3.236	311
6001 – 7000	0.110	0.797	0.907	318
7001 – 8000	0.100	1.031	1.131	336
8001 – 9000	0.103	1.184	1.287	359
9001 – 9350	0.034	1.295	1.329	361

TABLE I: Run-times (in seconds) for processing each interval of $1,000$ points of the Avian Influenza stream

The run-times (in seconds) for summarizing the data (online component) and computing persistent homology (offline component) in the $1,000$ point intervals of the stream are shown in Table I. The total run-time for each $1,000$ points interval is the sum of the run-times for the online and offline components. In addition, Table I includes the number of microclusters accumulated after processing each $1,000$ points. The number of microclusters indicates the memory usage that remains fairly constant during the progression of the stream. The memory usage depends on the decay parameter λ. For this study, we allowed the ClusTree implementation itself to infer the value of λ from the stream. The choice of the interval of $1,000$ data points is reasonable for this data stream. Computing persistent homology too often can result in a waste of resources, while computing too infrequently may lead to missing important topological changes in an evolving stream. We recommend choosing the parameters of ClusTree ($horizon$, $maxHeight$ and λ) as well as the intervals for the offline component in a way that enables the effective identification of evolving topological structures in a data stream.

C. Evolution of HIV Virus

In this Section, the applicability of our framework is demonstrated by identifying reticulate exchanges during the evolution of the HIV virus. HIV is infamous for frequent homologous recombination and high mutation rates that lead to antiretroviral resistance [35] and immunodeficiency [36]. These factors confound the study of evolution of HIV virus by traditional methods that rely on phylogenetics [9].

As with the study of Avian Influenza, independent and concatenated gene segments of HIV virus represent, respectively, the absence and presence of reticulate genetic exchanges. This data stream consists of $3,000$ independent sequences of HIV-1 gag, and $2,522$ concatenated sequences of HIV-1 gag, pol, and env, the three largest genes of the genome.

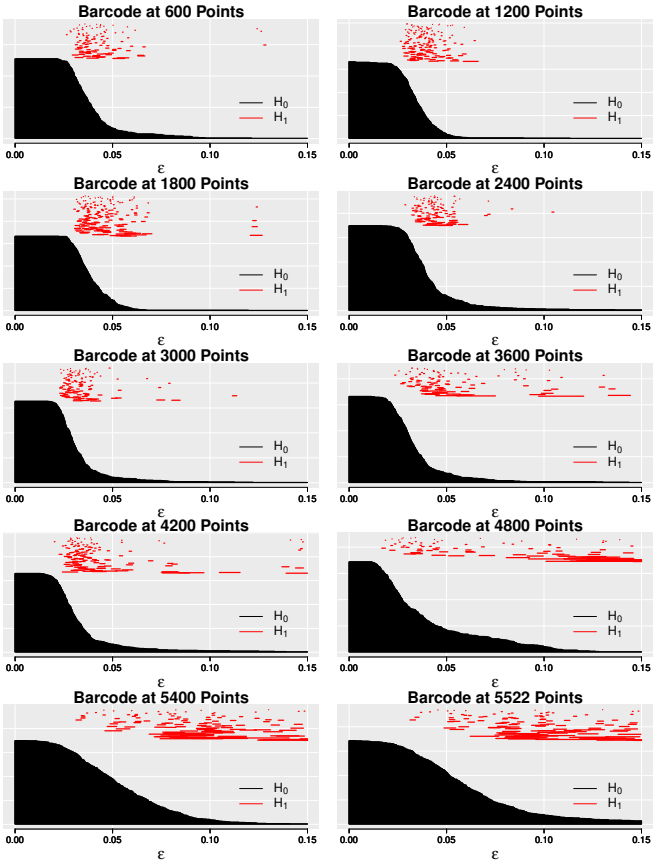

Fig. 3: Identification of reticulate evolution in HIV virus

The data, accessed from [37], was originally downloaded from the Los Alamos HIV Databases [38]. As before, the viral sequences were mapped to numeric vectors in a 45-dimensional Euclidean space. Figure 3 shows the output as barcodes computed at intervals of 600 data points through the end of the stream. The absence of reticulate genetic exchanges in the first $3,000$ independent sequences is demonstrated by the short H_1 bars that represent noise. After that, however, we begin observing longer H_1 bars in the barcodes that indicate horizontal evolutionary events. Thus, by monitoring the output of persistent homology, we are able to identify the occurrence of reassortment and recombination that lead to reticulate evolution of viruses.

The run-times (in seconds) for the online and offline components, as well as the total run-time at each step, are shown in Table II. The number of microclusters after each 600 points interval remains approximately constant during the progression of the stream. The decay parameter λ was set to 0.008 that led to stable memory usage and effective identification of the reticulate genetic exchanges in the HIV virus.

D. Data Preparation and Future Work

The data summarization models are based on the incremental updates to the feature vectors that require the data points of the stream to be numeric vectors embedded in a metric space. Since nucleotide sequences are sets of characters, they do not form a metric space and cannot be directly processed by the data summarization models. Our approach to transform the genetic sequences to Euclidean vectors is outlined below.

Progression of Stream	Data Summary Run-time	Persistent Homology Run-time	Total Run-time	No. of Micro-clusters
1 – 600	0.061	1.443	1.504	255
601 – 1200	0.064	3.902	3.966	333
1201 – 1800	0.070	3.931	4.001	333
1801 – 2400	0.064	4.555	4.619	349
2401 – 3000	0.058	3.025	3.083	327
3001 – 3600	0.065	2.145	2.210	332
3601 – 4200	0.056	2.089	2.145	330
4201 – 4800	0.062	0.821	0.883	344
4801 – 5400	0.067	1.871	1.938	349
5401 – 5522	0.013	0.988	1.001	343

TABLE II: Run-times (in seconds) for processing each interval of 600 points of the HIV stream

1) Align the sequences by Multiple Sequence Alignment.
2) Compute pairwise distances between the sequences using Jukes-Cantor method [39] that has been shown to preserve the topological properties of the evolution of genetic sequences [9].
3) Translate the pairwise distances to vectors in a d-dimensional Euclidean space using Classical Multidimensional Scaling (MDS) [40]. For the experiments in the previous sections, we empirically chose $d = 45$ that provided a good trade-off between the run time and the value of the loss function that MDS aims to minimize.

As a future direction, we are working towards a solution that would enable the mapping of genetic sequences to vectors in a metric space in an online fashion with the help of the recent progress in alignment-free sequencing methods [41]. In any event, our current framework provides the ability to identify horizontal evolution in large enough sets of sequences that cannot be dealt with existing mechanisms. For example, in the absence of a stream processing framework, the main bottleneck in the detection of reticulate events in the Avian Influenza data set of $9,350$ sequences would be the computation of persistence intervals. The memory requirement for the data pre-processing steps would be insignificant compared to that required by persistent homology for $9,350$ data points. Furthermore, we did not include the execution times for the data preparation steps in the previous subsections because they are not integral parts of the proposed framework. The data pre-processing was required only for the particular application of our framework to the study of viral evolution.

A Note on the Connected Components: One may ask, "What is the purpose of the 0-dimensional connected components (H_0) in the study of evolution?" Connected components are merely an alternative representation of dendrograms or trees that characterize vertical or clonal evolution. Although not studied in this paper, 0-dimensional components have been shown to demonstrate the same evolutionary relationship as that done by phylogenetic trees. For example, Chan *et al.* [9] reconstructed phylogeny from the connected components of different subtypes of the Hemagglutinin (HA) protein in Avian Influenza viruses. It is the horizontal or reticulate

evolution that, however, cannot be effectively characterized by the classical methods for the study of evolution. This is the reason why persistent homology is said to provide a comprehensive representation of both vertical and horizontal evolution at the same time.

V. Conclusion

We introduced the first computational model for applying persistent homology to potentially unbounded real data streams. The framework described in this paper is evaluated in terms of execution times, memory usage and the ability to effectively identify topological changes in evolving data streams. The applicability of the framework is demonstrated by the important task of the detection of reticulate evolution of viruses in streams of genomic sequences. Reticulate genomic exchanges that occur through recombination, reassortment and lateral gene transfer in viruses and bacteria are pervasive in nature and confound the discovery of treatments for diseases caused by such organisms. The proposed framework, being a stream processing model, can aid in the discovery of knowledge from arbitrarily large genomic databases, enabling the study of evolution of organisms that evolve quickly through mutation and recombination.

References

[1] G. Carlsson, "Topology and data," *Bulletin of the American Mathematical Society*, vol. 46, no. 3, pp. 255–308, Apr. 2009.

[2] F. Chazal and B. Michel, "An introduction to topological data analysis: fundamental and practical aspects for data scientists," *ArXiv e-prints*, Oct. 2017.

[3] H. Edelsbrunner and J. Harer, "Persistent homology – a survey," *Surveys on Discrete and Computational Geometry*, vol. 453, pp. 257–282, 2008.

[4] C. S. Pun, K. Xia, and S. X. Lee, "Persistent-homology-based machine learning and its applications – a survey," Nov. 2018.

[5] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington, "A roadmap for the computation of persistent homology," *EPJ Data Science*, vol. 6, no. 1, Aug. 2017.

[6] V. de Silva and G. Carlsson, "Topological estimation using witness complexes," in *Eurographics Symposium on Point-Based Graphics*, ser. SPBG '04, M. Gross, H. Pfister, M. Alexa, and S. Rusinkiewicz, Eds. The Eurographics Association, 2004.

[7] G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian, "On the local behavior of spaces of natural images," *International Journal of Computer Vision*, vol. 76, no. 1, pp. 1–12, Jan. 2008.

[8] R. Ghrist, "Barcodes: The persistent topology of data," *Bulletin of the American Mathematical Society*, vol. 45, no. 1, pp. 61–75, 2008.

[9] J. M. Chan, G. Carlsson, and R. Rabadan, "Topology of viral evolution," *Proceedings of the National Academy of Sciences*, vol. 110, no. 46, pp. 18 566–18 571, 2013.

[10] G. Petri, M. Scolamiero, I. Donato, and F. Vaccarino, "Topological strata of weighted complex networks," *PLOS ONE*, vol. 8, no. 6, pp. 1–8, Jun. 2013. [Online]. Available: https://doi.org/10.1371/journal.pone.0066506

[11] M. Hajij, B. Wang, C. E. Scheidegger, and P. Rosen, "Visual detection of structural changes in time-varying graphs using persistent homology," *2018 IEEE Pacific Visualization Symposium*, pp. 125–134, 2018.

[12] M. Shindler, A. Wong, and A. W. Meyerson, "Fast and accurate k-means for large datasets," in *Advances in Neural Information Processing Systems*, 2011, pp. 2375–2383.

[13] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. de Carvalho, and J. ao Gama, "Data stream clustering: A survey," *ACM Computing Surveys*, vol. 46, no. 1, pp. 3.1–13.31, Oct. 2013.

[14] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '96. New York, NY, USA: ACM, 1996, pp. 103–114.

[15] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th International Conference on Very Large Data Bases*, vol. 29, 2003, pp. 81–92.

[16] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *SIAM Conference on Data Mining*, 2006, pp. 328–339.

[17] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The clustree: indexing micro-clusters for anytime stream mining," *Knowledge and Information Systems*, vol. 29, no. 2, pp. 249–272, 2011.

[18] D. Baum, "Reading a phylogenetic tree: The meaning of monophyletic groups," *Nature Education*, vol. 1(1):190, 2008.

[19] M. Wright. (2016) Introduction to persistent homology. [Online]. Available: https://www.youtube.com/watch?v=2PSqWBIrn90

[20] H. Edelsbrunner and J. Harer, *Computational Topology, An Introduction*. American Mathematical Society, 2010.

[21] A. Zomorodian and G. Carlsson, "Computing persistent homology," *Discrete Comput. Geom.*, vol. 33, no. 2, pp. 249–274, Feb. 2005.

[22] R. Forman, "Morse theory for cell complexes," *Advances in Mathematics*, vol. 134, no. 1, pp. 90–145, 1998.

[23] A. Moitra, N. Malott, and P. A. Wilsey, "Cluster-based data reduction for persistent homology," in *IEEE International Conference on Big Data*, Dec. 2018, pp. 327–334.

[24] N. O. Malott and P. A. Wilsey, "Fast computation of persistent homology with data reduction and data partitioning," in *2019 IEEE International Conference on Big Data*, ser. Big Data 2019, Dec. 2019, pp. 880–889.

[25] A. Zomorodian, "Fast construction of the vietoris–rips complex," *Computer and Graphics*, pp. 263–271, 2010.

[26] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological persistence and simplification," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, ser. FOCS '00. Washington, DC, USA: IEEE Computer Society, 2000.

[27] M. Kerber and H. Schreiber, "Barcodes of towers and a streaming algorithm for persistent homology," *Discrete & Computational Geometry*, Oct. 2018.

[28] M. Hahsler and J. Forrest, *streamMOA: Interface for MOA Stream Clustering Algorithms*, 2019, R package version 1.2-2. [Online]. Available: https://CRAN.R-project.org/package=streamMOA

[29] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive online analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[30] B. T. Fasy, J. Kim, F. Lecci, C. Maria, D. L. Millman, and V. Rouvreau, *TDA: Statistical Tools for Topological Data Analysis*, 2019, R package version 1.6.9. [Online]. Available: https://CRAN.R-project.org/package=TDA

[31] C. Maria, J.-D. Boissonnat, M. Glisse, and M. Yvinec, "The GUDHI library: Simplicial complexes and persistent homology," INRIA, Tech. Rep. RR-8548, 2014. [Online]. Available: https://hal.inria.fr/hal-01005601v2

[32] T. G. Project, *GUDHI User and Reference Manual*, 3rd ed. GUDHI Editorial Board, 2020. [Online]. Available: gudhi.inria.fr/doc/3.2.0/

[33] Influenza virus database. [Online]. Available: https://www.ncbi.nlm.nih.gov/genomes/FLU/Database/nph-select.cgi?go=database

[34] E. Paradis and K. Schliep, "ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R," *Bioinformatics*, vol. 35, no. 3, pp. 526–528, Jul. 2018.

[35] T. Nora, C. Charpentier, O. Tenaillon, C. Hoede, F. Clavel, and A. J. Hance, "Contribution of recombination to the evolution of human immunodeficiency viruses expressing resistance to antiretroviral treatment," *Journal of Virology*, vol. 81, no. 14, pp. 7620—7628, Jul. 2007.

[36] D. N. Levy, G. M. Aldrovandi, O. Kutsch, and G. M. Shaw, "Dynamics of HIV-1 recombination in its natural target cells," in *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 12, 2004, pp. 4204–4209.

[37] ph_datasets. [Online]. Available: github.com/RabadanLab/ph_datasets

[38] HIV databases. [Online]. Available: http://www.hiv.lanl.gov/

[39] T. H. Jukes and C. R. Cantor, "Evolution of protein molecules," in *Mammalian Protein Metabolism*, H. N. Munro, Ed. Academic Press, New York, 1969, vol. 3, ch. 24, pp. 21–132.

[40] M. A. A. Cox and T. F. Cox, "Multidimensional scaling," in *Handbook of Data Visualization*. Springer, Berlin, Heidelberg, 2008, pp. 315–347.

[41] B. D. Ondov, T. J. Treangen, P. Melsted, A. B. Mallonee, N. H. Bergman, and S. Koren, "Mash: fast genome and metagenome distance estimation using MinHash," *Genome Biology*, vol. 17, no. 132, Jun. 2016.