March 2019

# Analyses of Unorthodox Overlapping Gene Segments in Oxytricha Trifallax

Shannon Stich
*University of South Florida*, sstich01@yahoo.com

# Analyses of Unorthodox Overlapping Gene Segments In Oxytricha Trifallax

by

Shannon Stich

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Master of Arts
Department of Mathematics & Statistics
College of Arts and Sciences
University of South Florida

Major Professor: Masahiko Saito, Ph.D.
Nataša Jonoska, Ph.D.
Dmytro Savchuk, Ph.D.

Date of Approval:
March 19, 2019

Keywords: Topological Data Analysis, Homology, Persistent Homology, Vietoris- Rips filtration, ciliates

## Dedication

My family for their love and support.

## Acknowledgments

I would like to thank Professor Nataša Jonoska, Ph.D. and Professor Masahiko Saito, Ph.D. for their assistance with this paper and the USF math biology research group. I would like to thank my loved ones for their support and care.

## Table of Contents

**Analyses of Unorthodox Overlapping Gene Segments In Oxytricha Trifallax**

**Shannon Stich**

## ABSTRACT

A ciliate is a phylum of protozoa that has two types of nuclei, macronuclei and micronuclei. There may be more than one of each type of nucleus in the organism [1]. The macronucleus is the structure where protein synthesis and cell metabolism occur [1]. The micronucleus stores genetic information and is mobilized during a sexual reproduction process called conjugation [1]. The somatic macronucleus (MAC) is developed from the germ-line micronucleus (MIC) through genome rearrangement during a sexual reproduction process called conjugation [6, 8]. Segments of the MIC that form the MAC during conjugation are called macronuclear destined sequences (MDSs) [8]. During sequencing each MDS is given coordinates where the MDS sequences begin and end in the MIC. The orientation of a MDS in the MIC can be taken to be positive or negative. If the direction of the MDS in the MIC agrees with the direction in the MAC then the orientation is positive otherwise it is a negative orientation. In this thesis we analyze various aspects of the gene assembly during the rearrangment process of the ciliate *Oxytricha trifallax* that were recently sequenced [15]. Some of the properties analyzed include overlapping MDSs, orientation, MDSs starting and ending position in the MIC and the gaps of overlapping MDS pairs. A gap of an overlapping MDS pair is the order difference of two MDSs for a particular MAC contig that overlap in the MIC contig. We use 120 MAC contigs from [15] that have overlaps among their own MDSs. These 120 MAC contigs make up the data set we call $\mathcal{D}_4$.

We explore the patterns of overlapping MDSs in the MIC in $\mathcal{D}_4$. To quantify such patterns, we associate a vector $V(A_n)$ to each MAC contig $A_n$, where $V(A_n) = (v_1(A_n), v_2(A_n), v_3(A_n))$ is a vector in $\mathbb{R}^3$. The first entry is the number of overlapping MDS pairs divided by the number of MDSs. The second entry is the sum of gaps of overlapping MDS pairs divided by the sum of all possible gaps. The final entry is the total

number of overlapping base pairs divided by the total length of the MAC contig. We computed the distance matrix $M = (d_{ij})$ where $d_{ij}$ is the Euclidean distance between $V(A_i)$ and $V(A_j)$. The MAC contig vectors and $M$ were computed using Python.

To analyze $\mathcal{D}_4$ we applied Topological Data Analysis (TDA). TDA uses topological constructs to assess shapes in data [3, 12]. From the data entries of the distance matrix $M = (d_{ij})$ we applied a Vietoris-Rips filtration to generate the barcodes of the persistent homology in dimensions $0, 1$ and $2$. The persistence barcode of $0$-dimensional homology illustrates clusters of the data while the $1$-dimensional homology represents non-trivial loops in the simplicial complex [3, 13]. The application of TDA on the ciliate *Oxytricha trifallax* identified ten MAC contig clusters at $\epsilon = 0.1$ in $\mathcal{D}_4$ and several loops that were persistent for two or three $\epsilon$ values. Other TDA methods can be applied to the Vietoris-Rips filtration to further identify which MAC contigs appear in each cluster.

# Chapter 1
# Introduction

A ciliate is a phylum of protozoa that has two types of nuclei, macronuclei and micronuclei. A specie may have more than one of each type of nuclei [1]. The somatic macronucleus (MAC) is developed from the germ-line micronucleus (MIC) through genome rearrangement during a sexual reproduction process called conjugation [6, 8]. The macronucleus is the structure where protein synthesis and cell metabolism occur [1]. The micronucleus stores genetic information [1]. Segments of the MIC that form the MAC during conjugation are called macronuclear destined sequences (MDSs) [8]. Here we analyze specific subset of the sequenced data in Chen et al. [15] by considering aspects of the gene assembly of the ciliate *Oxytricha trifallax* such as orientation of MDSs and the overlapping MDSs in the MIC. A contig is an adjoining length of a genomes sequence [5] and the sequenced data in Chen et al. [15] is given as a set of contigs. A segment of the MIC contig that gets deleted and interrupts two MDSs during conjugation is called an internally eliminated sequence or IES for short [7]. The positions of MDSs may appear scrambled and the orientation may be reversed in the MIC. Rearrangements during conjugation include IES deletion from the MIC and MDS unscrambling in the MAC [7]. They become unscrambled during the formation of the MAC [7]. Macronuclear DNA generally are short chromosomes that most of the time encode for one gene [1]. The *Oxytricha trifallax* macronuclear genome is made up of about 16,000 nanochromosomes [1]. We refer to the data used in [15] as $\mathcal{D}_1$. The analysis in that paper considered the MAC contigs and MIC loci of the MAC gene segments that were scrambled, alternate processes of MDSs that produce multiple genes and chromosomes [15]. In [8], patterns of scrambling of MDSs were studied using double occurrence words and assembly graphs. In the previous analysis [8, 15], MAC contigs with unorthodox overlaps between non-consecutive MDSs were excluded. In this thesis we focus on these MAC contigs with overlaps. We constructed the data set $\mathcal{D}_4$ consisting of MAC contigs with non-consecutive MDSs that have overlaps in the MIC which were present in $\mathcal{D}_1$.

We explore the patterns of overlapping MDSs in the MIC in $\mathcal{D}_4$. To quantify such patterns, we associate a

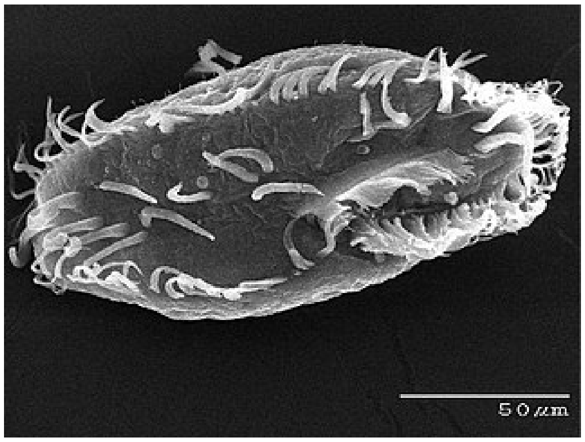


**Figure 1.** Ciliate *Oxytricha trifallax* [16].

vector $V(A_n)$ to each MAC contig $A_n$, where $V(A_n) := (v_1(A_n), v_2(A_n), v_3(A_n))$ is a vector in $\mathbb{R}^3$. The MAC contig vector $V(A_n)$ consists of three entries. The first entry $v_1(A_n)$ is the number of overlapping MDS pairs divided by the number of MDSs. The second entry $v_2(A_n)$ is the sum of gaps of overlapping MDS pairs divided by the sum of all possible gaps. The final entry $v_3(A_n)$ is the total number of overlapping base pairs divided by the total length of the MAC contigs. We computed a distance matrix $M = (d_{ij})$ where $d_{ij}$ is the Euclidean distance between $V(A_i)$ and $V(A_j)$. The MAC contig vectors and the distance matrix $M$ were computed using Python. We applied topological data analysis (TDA) to the distance matrix $M$. The outputs of the TDA are persistence barcodes. The persistence barcodes were computed using an online version of Ripser [20]. The persistence barcodes revealed that for the zero-dimensional homology group, for example, we had connected components for $\epsilon = 0.1$. The zero-dimensional homology group for this particular value of $\epsilon$ geometrically corresponds to 10 MAC contig clusters. Most of the loops in the simplicial complexes for the one-dimensional homology appeared in the range of $\epsilon = 0.04$ to $\epsilon = 0.1$. There was one hollow void present for the two-dimensional homology group.

The organization of this thesis is as follows: in Chapter 2 we provide the background on the genome rearrangements that occur in *Oxytricha trifallax*. In Chapter 3 we include various applications of TDA and the important terminology relevant to our output and analysis. In the remaining sections we examine the output of persistent homology on $\mathcal{D}_4$. We also include the Python script and the MAC contig vector entries in Appendix A.

# Chapter 2

# Biology: An Overview of Ciliates Gene Assembly

A ciliate is a phylum of protozoa that has two nuclei, macronuclei and micronuclei. Genomic rearrangement through conjugation happens in a ciliate when it is not thriving in the current environment and at that time a macronucleus is formed from a newly conjugated micronucleus. In this process a large segment of the DNA is recombined [6]. Segments of the MIC that form the MAC during conjugation are called macronuclear destined sequences (MDSs) [8]. A segment of a MIC contig that gets deleted and interrupts two MDSs in the MIC is called an internally eliminated sequence or IES for short [7]. Since MDSs may have different orientations in the MIC with respect to their MAC orientation, we classify these as positive or negative. If the direction of the MDS in the MAC agrees with the direction in the MIC then the orientation is positive, otherwise it is a negative orientation.

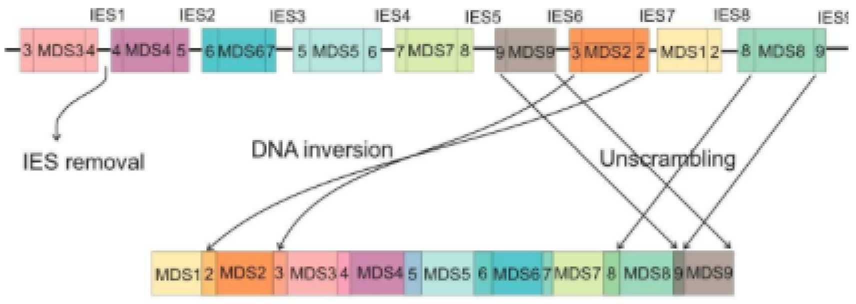


**Figure 2.** Various rearrangement types [17].

Rearrangements of MDSs from the MIC to the MAC involve IES deletions, inversions and unscrambling. Figure 2 above indicates that each of these rearrangements occurs on Actin I of *Oxytricha Nova*. All IESs are removed from the MIC. An orientation of the MIC contig is specified to be from left to right. The orientation of the start and end coordinate positions of MDS2 is of opposite orientation in the MAC. Therefore the orientation of MDS2 is negative. A DNA inversion occurs in MDS2 during the rearrangement process.

Lastly MDS9 and MDS8 undergo unscrambling when they form the MAC [17].

The segments that are the overlapping region of two consecutive MDSs in the MAC are called pointers. It is considered that the pointers guide scrambling in the rearrangement process. Non-consecutive overlapping MDSs are considered unorthodox since their rearrangements cannot be explain by pointer alignments. Therefore in the previous analysis [8, 15], MAC contigs with overlaps between non-consecutive MDSs were excluded. In this thesis we study the overlapping patterns of MAC contigs.

Four data sets arise from the overlapping MDS sequences of MAC contigs in the ciliate *Oxytricha trifallax*. The first data set $\mathcal{D}_1$ consists of the set of contigs used in the analysis of MAC and MIC contigs in [15]. The second data set $\mathcal{D}_2$ consists of MAC contigs from $\mathcal{D}_1$ that do not have the following two conditions: having MAC contigs with non-consecutive MDSs that overlap in a MIC contig or the MAC contig is an alternative fragmentation of a longer MAC contig [8]. We created a third data set, $\mathcal{D}_3$, which consist of MAC contigs in $\mathcal{D}_1$ that are excluded from $\mathcal{D}_2$. We have that $\mathcal{D}_2$ and $\mathcal{D}_3$ are disjoint and their union is $\mathcal{D}_1$.The data set $\mathcal{D}_3$ consists of 409 MAC contigs. From these 409 MAC contigs there are 120 MAC contigs that have overlaps among their own non consecutive MDSs. These 120 MAC contigs make up the data set $\mathcal{D}_4$. Data sets $\mathcal{D}_1$ and $\mathcal{D}_2$ have already been analyzed in [7, 8, 10, 15].

In the original data set $\mathcal{D}_1$ in [15] each MAC contig segment is given a name. For instance, $\mathrm{Contig14925.0\_MDS}$ is a name of one particular MAC contig. Each MDS is given coordinates where the sequences begin and end in the MIC. The orientation of a MDS in the MIC is indicated as positive or negative. If the direction of the MDS in the MIC agrees with the direction of the MAC then the orientation is positive, and otherwise it is a negative orientation. Further explanation of the data is presented in Chapter 4.

# Chapter 3

# Topological Data Analysis

Section 3.1 provides various examples of Topological Data Analysis (TDA) and the usefulness of TDA. The remaining three sections of this chapter include background information and definitions that are key in describing the persistence barcode results.

## 3.1 Motivation and Applications of Topological Data Analysis

Topological Data Analysis (TDA) uses topological constructs such as persistent homology to assess shapes in various types of data [3, 12]. From a data set in Euclidean space, a simplicial complex is constructed. Persistent homology demonstrates various changes in the overall shape of the data which depends on simplicial complexes for the corresponding radii of neighborhoods around each data point in the data set []. An $\epsilon$- neighborhood of $x$, also called a ball of radius $\epsilon$ denoted $B_\epsilon(x)$ in Euclidean space $\mathbb{R}^n$, is defined as

$$B_\epsilon(x) = \{y \in \mathbb{R}^n \mid d(x, y) < \epsilon\}.$$

There are different dimensions for homology groups. Zero dimensional homology refers to clusters of data points. One dimensional homology represents non-trivial loops and two dimensional homology represents hollow voids in Euclidean space [3, 13].

Persistent homology can be applied to various types of data ranging anywhere from movie plot text to sports statistics and biological data [2, 14, 11]. Persistent homology can be used to determine if a sports team is successful. An analysis of hockey players who spent a long duration on ice and 12 statistics of each player were recorded such as goals, assists, setup passes, primary points, etc. in order to determine the success of a team. According to Daniel Goldfarb's analysis of hockey teams via persistent homology, he found that hockey teams with long time $0$-dimensional persistent homology and short or non-existent $1$-dimensional persistent homology were successful [2]. This is because $0$-dimensional homology is related to clusters meaning that the players are diverse [2]. Goldfard stated that short lived $1$-dimensional persistent homology means that the team has uniform distribution of qualities. He used 1-dimensional persistence homology when 0-dimensional homology did not provide enough information to compare differences between teams [2]. Persistent homology can also be applied to analyze non-numerical data such as movie genre classification. The hit rate and Jaccard index are both attained via the barcode graphs of the persistent homology data [11]. Persistent homology is used to analyze breast cancer rates via Betti number comparisons [14]. The Betti number for the zero dimensional homology group is the number connected components [14]. Persistent diagrams are also used to analyze the characteristics between different types of cancers [14].

## 3.2 Chain Complexes and Homology

Homology is from the Greek word homologia meaning agreement. Homology describes the geometric properties or shape of the space such as clusters, loops or hollow voids in low dimensions that we can visualize [3, 14]. The following definitions follow along with Munkres literature on Algebraic Topology [9].

**Definition 3.2.1.** Let $\{x_0, x_1, \ldots, x_n\}$ be a collection of $n+1$ points in Euclidean space $\mathbb{R}^m$, such that the vectors $x_1 - x_0, \ldots, x_n - x_0$ are linearly independent. Then an *$n$-simplex* $\sigma$ spanned by $\{x_0, x_1, \ldots, x_n\}$ is defined by $\sigma = \{x \in \mathbb{R}^m \mid x = \sum_{i=0}^{n} a_i x_i, \sum_{i=0}^{n} a_i = 1$ and $a_i \geq 0$ for all $i\}$. The points $\{x_0, x_1, \ldots, x_n\}$ are called the vertices of $\sigma$.

**Definition 3.2.2.** Let $\sigma$ be an $n$-simplex spanned by $\{x_0, x_1, \ldots, x_n\}$. A simplex spanned by a subset of $\{x_0, x_1, \ldots, x_n\}$ is called a *face* of $\sigma$.

**Example 1.** A 0-simplex is a single point in Euclidean space, a 1-simplex is an edge, a 2-simplex is a triangle with its interior filled in, and a 3-simplex is a tetrahedron (see Figure 3).

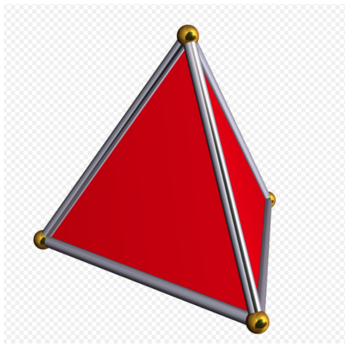


**Figure 3.** 3-simplex from [18]

**Definition 3.2.3.** A *simplicial complex* $K$ in $\mathbb{R}^m$ is a collection of simplices in $\mathbb{R}^m$ such that

1. Every face of a simplex of $K$ is in $K$.
2. For two simplices $A$ and $B$ in $K$ the intersection is a face of both $A$ and $B$.

**Definition 3.2.4.** Let $\sigma$ be an $n$-simplex spanned by $\{x_0, x_1, \ldots, x_n\}$. Two orderings of $\{x_0, x_1, \ldots, x_n\}$ are equivalent if they differ by an even permutation. The orderings consist of two equivalence classes that are called the orientations of $\sigma$. An *oriented* $n$-simplex is an $n$-simplex with its orientation specified and is denoted by $[x_0, x_1, \ldots, x_n]$ with the orientation specified by this order.

**Definition 3.2.5.** Let $K$ be a simplicial complex in $\mathbb{R}^m$ . The *n-th chain group* denoted by $C_n(K)$ is the free abelian group generated by oriented $n$-simplices of $K$. Elements of $C_n(K)$ are called $n$-chains. For an oriented $n$-simplex $\sigma$, the same simplex with opposite orientation is identified with $-\sigma$.

**Definition 3.2.6.** Let $[x_0, \ldots, x_n]$ be an oriented simplex of a simplicial complex $K$ in $\mathbb{R}^m$, regarded as an element of $C_n(K)$. Define $\partial_n[x_0, \ldots, x_n]$ by $\sum_{i=0}^{n}(-1)^i[x_0, \ldots, \hat{x_i}, \ldots, x_n] \in C_{n-1}(K)$, where $\hat{x_i}$ means that the point $x_i$ is deleted. By linearly extending this map we obtain a homomorphism $\partial_n : C_n(K) \to C_{n-1}(K)$ which we call a boundary operator. It follows from the definition that $\partial_{n-1} \circ \partial_n = 0$ for all $n$.

**Definition 3.2.7.** The kernel of the homomorphism $\partial_n : C_n(K) \to C_{n-1}(K)$ is called the $n$-cycle group $Z_n(K)$. The image of the mapping $\partial_{n+1} : C_{n+1}(K) \to C_n(K)$ is called the n-boundary group $B_n(K)$. Note that $\partial_{n-1} \circ \partial_n = 0$ implies that $B_n(K) \subset Z_n(K)$.

**Definition 3.2.8.** The n-dimensional homology group is defined as the quotient group $H_n(K) = \frac{\mathrm{Ker}(\partial_n)}{\mathrm{Im}(\partial_{n+1})} = \frac{Z_n(K)}{B_n(K)}$.

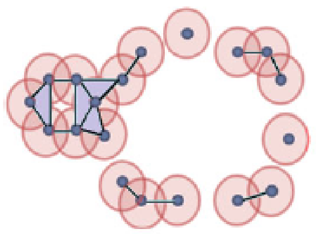
## 3.3 Filtrations

Let $K$ be a simplicial complex in $\mathbb{R}^N$. A function $f : K \to \mathbb{R}$ is called a *monotonic* function for two simplices, $\vartheta$ and $\sigma$ of $K$, when $\vartheta$ is a face of $\sigma$ we have that $f(\vartheta) \leq f(\sigma)$. A subcomplex is a subcollection of simplices from a simplicial complex which also forms a simplicial complex [14]. A subcomplex $K_a$ of a simplicial complex $K$ is called the sublevel set and is defined as $K_a = f^{-1}(-\infty, a]$. The sequence of subcomplexes that are arranged in an increasing inclusion $\emptyset = K_{a_0} \subseteq K_{a_1} \subseteq \cdots \subseteq K_{a_n} = K$ is called a filtration of a monotonic function $f$ defined on the simplicial complex $K$ [4].

Let P be a set of data points in $\mathbb{R}^N$ and let $K$ be the complete simplicial complex of the data points P. For $\epsilon \geq 0$, a subcollection $\sigma$ =$\{x_{i_0}, \ldots, x_{i_k}\} \subset P$ forms a simplex of $K_\epsilon$ in the Čech filtration if $\cap B_\epsilon(x_{i_s}) \neq \emptyset$ for all $s = 0, 1, 2, \ldots, k$. For the Čech filtration we define a function $f : K \to \mathbb{R}$ such that for a $k$-simplex $\sigma$ in $K$ formed by $\{x_{i_0}, \ldots, x_{i_k}\}$ we have that $f(\sigma) = \min\{r \geq 0 \mid$ there is a point x $\in \mathbb{R}^N$ such that $d(x, x_{i_s}) < r$ for $s = 0, \ldots, k$ $\}$. The Čech filtration is the sublevelset filtration on $f$ and we have the chain of inclusion maps: $K_{\epsilon_0} \subseteq K_{\epsilon_1} \subseteq \cdots \subseteq K_{\epsilon_n}$ [14].

In the Vietoris-Rips filtration, for $\epsilon \geq 0$, a subcollection $\sigma$ =$\{x_{i_0}, \ldots, x_{i_k}\} \subset P$ forms a simplex of $K_\epsilon$ if $B_\epsilon(x_{i_s}) \cap B_\epsilon(x_{i_j}) \neq \emptyset$ for all pairs $x_{i_s}$ and $x_{i_j}$ in $\sigma$ [14]. For the Vietoris-Rips filtration we define a function $f' : K \to \mathbb{R}$ such that for a $k$-simplex $\sigma$ formed by $\{x_{i_0}, \ldots, x_{i_k}\}$ we have that $f'(\sigma) = \max_{x_{i_s}, y_{i_j} \in \sigma}$ $\{d(x_{i_s}, x_{i_j})\}$ [14]. The Vietoris-Rips filtration is the sublevel set filtration of our function $f'$ and we have the chain of inclusion maps: $K_{\epsilon_0} \subseteq K_{\epsilon_1} \subseteq \cdots \subseteq K_{\epsilon_n}$ [14].

In the images below the simplices are shown in blue and the neighborhoods are the shaded red circles. As $\epsilon$ increases the neighborhoods gradually get connected. In the left figure, there are 6 connected components including four 2-simplices, and 1 non-trivial loop at this particular $\epsilon$ value. In the right figure, all the points encompass a single component, and which is a higher dimensional simplicial complex.



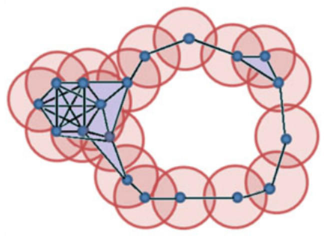


**Figure 4.** This is an example of a Vietoris-Rips filtration of a data set from [14].

In Section 4 the Vietoris-Rips filtration of the MAC contig distance vectors are computed using Ripser. When we apply the Vietoris-Rips filtration to $\mathcal{D}_4 = \{x_0, \ldots, x_n\}$ we set $K = K_{\{x_0,\ldots,x_m\}}$.

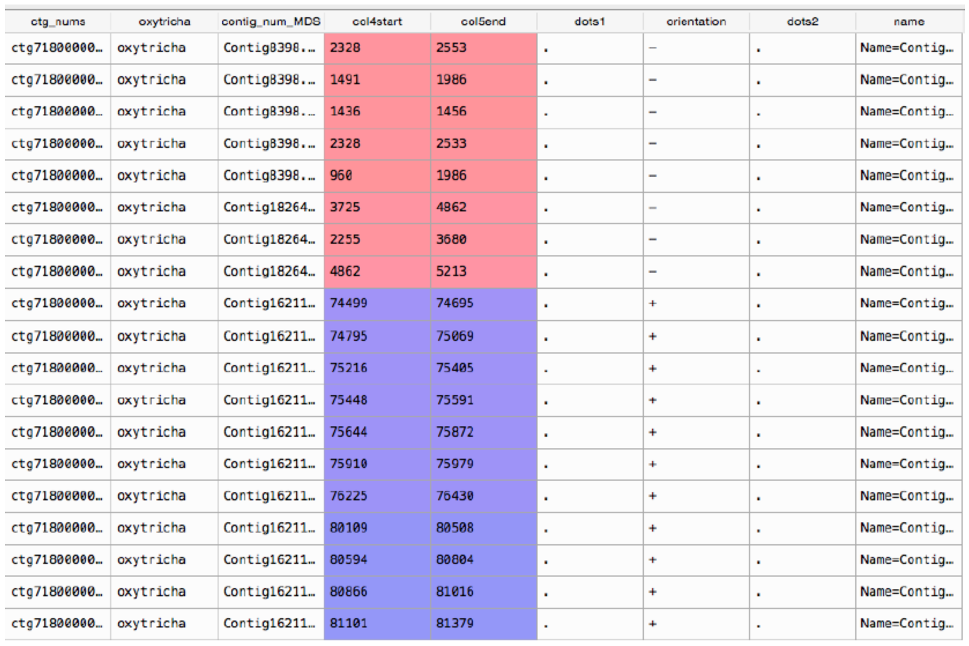
### 3.4 Persistent Homology

Define $K_i \hookrightarrow K_j$ with $i \leq j$ to be an inclusion map of simplicial complexes and let $f_n^{i,j}$ be such that $f_n^{i,j} : H_n(K_i) \to H_n(K_j)$ is a homomorphism induced from this inclusion map. The $n$-persistent homology groups are $\mathrm{Im}(f_n^{i,j})$ of the map above for $0 \leq i \leq j \leq n$ [4]. The term *persistent* in persistent homology refers to the long range from birth and death of the images of $f_n^{i,j}$ [4]. Typical elements of $H_n$ are $y + B_n$ where $y$ is an element from $Z_n$. A class is a coset of $H_n$ with the operation of the group being addition [4]. Let $y$ be a class in the $\mathrm{Im}(f_n^{i,i})$. The class $y$ is born at $K_i$ if $y$ is not in $\mathrm{Im}(f_n^{i-1,i})$. If the class $y$ is born at $K_i$ it dies entering $K_j$ if $f_n^{i,j-1}(y)$ is not in $\mathrm{Im}(f_n^{i-1,j-1})$ but $f_n^{i,j}(y)$ is in $\mathrm{Im}(f_n^{i-1,j})$ [4]. The difference between birth and death values of a class is called persistence [4]. If a class never dies it can have an infinite value of persistence [4].

# Chapter 4

# Description of the Data Set

There are two data sets of the ciliate *Oxytricha trifallax* that have been analyzed in [15] and [8]. The first data set $\mathcal{D}_1$ consists of those MAC and MIC contigs analyzed in [15]. The second data set $\mathcal{D}_2$ consists of MAC contigs from $\mathcal{D}_1$ that do not have the following two conditions: MAC contigs with non-consecutive MDSs that overlap in a MIC contig, or the MAC contigs that are alternative fragmentations of longer MAC contigs [8]. We considered a third data set, $\mathcal{D}_3$, which consists of MAC contigs in $\mathcal{D}_1$ that are excluded from $\mathcal{D}_2$. We have that $\mathcal{D}_2$ and $\mathcal{D}_3$ are disjoint and their union is $\mathcal{D}_1$. Figure 5 shows how $\mathcal{D}_3$ is organized. The columns in the data frame that are relevant to our analysis start at the third column. The data in the third column of the figure corresponds to the MAC contigs name. In columns four and five each MDS is given coordinates where the sequences begin and end in the MIC, denoted by $\mathrm{s}(\mathrm{MDS_i})$ and $\mathrm{e}(\mathrm{MDS}_i)$, respectively. The orientation of a MDS in the MIC is positive or negative and is present in column seven. If the direction of the MDS in the MAC agrees with the direction in the MIC then the orientation is positive otherwise it is a negative orientation. The program starts with 409 MAC contigs in $\mathcal{D}_3$ and identifies 120 MAC contigs that have overlaps among their own non-consecutive MDSs. The data $\mathcal{D}_4$ consists of these 120 MAC contigs.



| ctg_nums | oxytricha | contig_num_MDS | col4start | col5end | dots1 | orientation | dots2 | name |
|---|---|---|---|---|---|---|---|---|
| ctg71800000… | oxytricha | Contig8398.… | 2328 | 2553 | . | – | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig8398.… | 1491 | 1986 | . | – | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig8398.… | 1436 | 1456 | . | – | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig8398.… | 2328 | 2533 | . | – | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig8398.… | 960 | 1986 | . | – | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig18264… | 3725 | 4862 | . | – | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig18264… | 2255 | 3680 | . | – | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig18264… | 4862 | 5213 | . | – | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig16211… | 74499 | 74695 | . | + | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig16211… | 74795 | 75069 | . | + | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig16211… | 75216 | 75405 | . | + | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig16211… | 75448 | 75591 | . | + | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig16211… | 75644 | 75872 | . | + | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig16211… | 75910 | 75979 | . | + | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig16211… | 76225 | 76430 | . | + | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig16211… | 80109 | 80508 | . | + | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig16211… | 80594 | 80804 | . | + | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig16211… | 80866 | 81016 | . | + | . | Name=Contig… |
| ctg71800000… | oxytricha | Contig16211… | 81101 | 81379 | . | + | . | Name=Contig… |

**Figure 5.** Preview of $\mathcal{D}_3$.

## 4.1 The MAC Contig Vector

We define a MAC contig vector for each MAC contig in $\mathcal{D}_4$ as follows. Let $A_n$ be the $n$-th MAC contig in $\mathcal{D}_4$ where $n = 1, \ldots, 120$. First we will introduce some useful notation and the definitions of the numerical properties used in the construction of the MAC contig vector $V(A_n)$.

Let $\mathrm{s(MDS_i)}$ denote the starting coordinate position of the $i$-th MDS within a MIC contig and $\mathrm{e}(\mathrm{MDS}_i)$ be the ending coordinate position of the $i$-th MDS in the MIC contig. The starting and ending coordinates of MDSs for each MAC contig are listed in columns 4 and 5 in $\mathcal{D}_3$. An overlap in the MIC between two MDSs, $\mathrm{MDS}_i$ and $\mathrm{MDS}_j$, within the same MAC contig is denoted by $O(\mathrm{MDS}_i, \mathrm{MDS}_j)$ is defined as:

$$O(\mathrm{MDS}_i, \mathrm{MDS}_j) = |\min\{\mathrm{e}(\mathrm{MDS}_i), \mathrm{e}(\mathrm{MDS}_j)\} - \max\{\mathrm{s}(\mathrm{MDS}_i), \mathrm{s}(\mathrm{MDS}_j)\}| + 1.$$

Two MDSs, $\mathrm{MDS}_i$ and $\mathrm{MDS}_j$ are said to be overlapping if $O(\mathrm{MDS}_i, \mathrm{MDS}_j) > 0$.

Let $\wp(A_n) := \{(\mathrm{MDS}_i, \mathrm{MDS}_j) \mid i < j,\ O(\mathrm{MDS}_i, \mathrm{MDS}_j) > 0\}$ be defined as the set of overlapping pairs and define $P(A_n) = |\wp(A_n)|$ to be the number of overlapping MDS pairs.Since we have that if $\mathrm{MDS}_i$ and $\mathrm{MDS}_j$ are an overlapping MDS pair then so are $\mathrm{MDS}_j$ and $\mathrm{MDS}_i$, this condition $i < j$ guarantees that overlapping MDS pairs are not double counted. Let

$$S(A_n) := \sum_{(\mathrm{MDS}_i, \mathrm{MDS}_j) \in \wp(A_n)} (O(\mathrm{MDS}_i, \mathrm{MDS}_j))$$

be the sum of the lengths of the overlapping segments of the pairs in $\wp(A_n)$.

For each MAC contig $A_n$, if $A_n$ consists of MDSs $\{\mathrm{MDS}_1, \ldots, \mathrm{MDS}_k\}$, then define $N(A_n) = k$ to be the total number of MDSs that comprise that particular MAC contig $A_n$. For a distinct pair, $\mathrm{MDS}_i$ and $\mathrm{MDS}_j$, the gap of the MDSs, $g(\mathrm{MDS}_i, \mathrm{MDS}_j)$, is defined as follows:

$$g(\mathrm{MDS}_i, \mathrm{MDS}_j) = \begin{cases} |i-j|-1 & \text{if } \mathrm{sign}(\mathrm{MDS}_i) = \mathrm{sign}(\mathrm{MDS}_j); \\ |i-j| & \text{if } \mathrm{sign}(\mathrm{MDS}_i) \neq \mathrm{sign}(\mathrm{MDS}_j), \end{cases} \tag{4.1}$$

where $\mathrm{sign}(\mathrm{MDS}_i)$ represents the orientation of the $\mathrm{MDS}_i$ in the MIC. We consider the gaps only for overlapping pairs. The gap value between two MDSs counts the number of MDSs that have been skipped over in the MAC contig. Let

$$G(A_n) := \sum_{(\mathrm{MDS}_i, \mathrm{MDS}_j) \in \wp(A_n)} g(\mathrm{MDS}_i, \mathrm{MDS}_j)$$

where the sum is taken over all overlapping MDS pairs of the MAC contig $A_n$ for $i < j$. Let

$$l(\mathrm{MDS}_i) := |\mathrm{s}(\mathrm{MDS}_i) - \mathrm{e}(\mathrm{MDS}_i)| + 1$$

be the length of a particular MDS and define

$$L(A_n) := \sum_i (l(\mathrm{MDS}_i)).$$

This is an approximate length of the MAC contig. Gaps can take the values $1, 2, \ldots, m-1$. The sum of all possible gaps is

$$\sum_{i=1}^{i=m-1} i = \frac{m(m-1)}{2} = \binom{m}{2}$$

with $m = N(A_n)$.

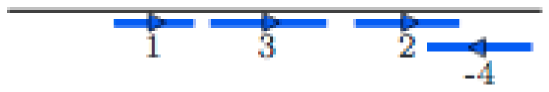


**Figure 6.** The figure above corresponds to an example of one MAC contig consisting of four labeled with MDSs [21]. There is an overlap between $\mathrm{MDS}_2$ and $\mathrm{MDS}_4$ in the MIC. The negative sign indicates an opposite orientation of $\mathrm{MDS}_4$ in the MIC relative to the other MDSs. Using the formulas described above, $g(\mathrm{MDS}_2, \mathrm{MDS}_4) = 2$. There is only one overlapping MDS pair for this particular MAC contig so we have that $G(A_n) = 2$. The total number of MDSs is four so $N(A_n) = 4$ and $\binom{4}{2} = 6$. Therefore we have that $v_2(A_n) = \frac{2}{6}$.

A vector $V(A_n) := (v_1(A_n), v_2(A_n), v_3(A_n)) \in \mathbb{R}^3$ is defined for each MAC contig $A_n$ as the follows:

$$v_1(A_n) := \frac{P(A_n)}{N(A_n)}, \quad v_2(A_n) := \frac{G(A_n)}{\binom{m}{2}} \quad \text{and} \quad v_3(A_n) := \frac{S(A_n)}{L(A_n)}.$$

## 4.2 The Vietoris-Rips Persistence Barcodes

From the data set $\mathcal{D}_4$ containing the 120 MAC contigs we computed the distance matrix $M = (d_{ij})$ that consists of the distance between each pair of the 120 MAC contig vectors $V(A_n)$. We used Ripser, a C++ based software to compute Vietoris-Rips persistence barcodes [20]. In the 0-dimensional case, starting from the bottom of the barcode graph, we find that a particular MAC contig has the shortest lifespan. This means that for small a $\epsilon$, $d(V(A_i), V(A_j)) < 2\epsilon$ and the vectors $V(A_i)$ and $v(A_j)$ are regarded as being merged to the same component. As $\epsilon$ increases we see that there is a progressive increase in the lifespan of the MAC contig vectors. For $\epsilon = 0.1$ there are 10 connected components. All of the components are connected at $\epsilon \geq 0.175$. The 0-dimensional homology group geometrically corresponds to clusters of MAC contigs. Most of the loops representing non-trivial elements in the 1-dimensional homology group appeared in the range of $\epsilon = 0.04$ to $\epsilon = 0.1$. There are two loops appearing at $\epsilon = 0.15$ and $\epsilon = 0.19$ that persistent. There is one hollow void representing a non-trivial element in the 2-dimensional homology group for $\epsilon = 0.059$ with a short persistence.

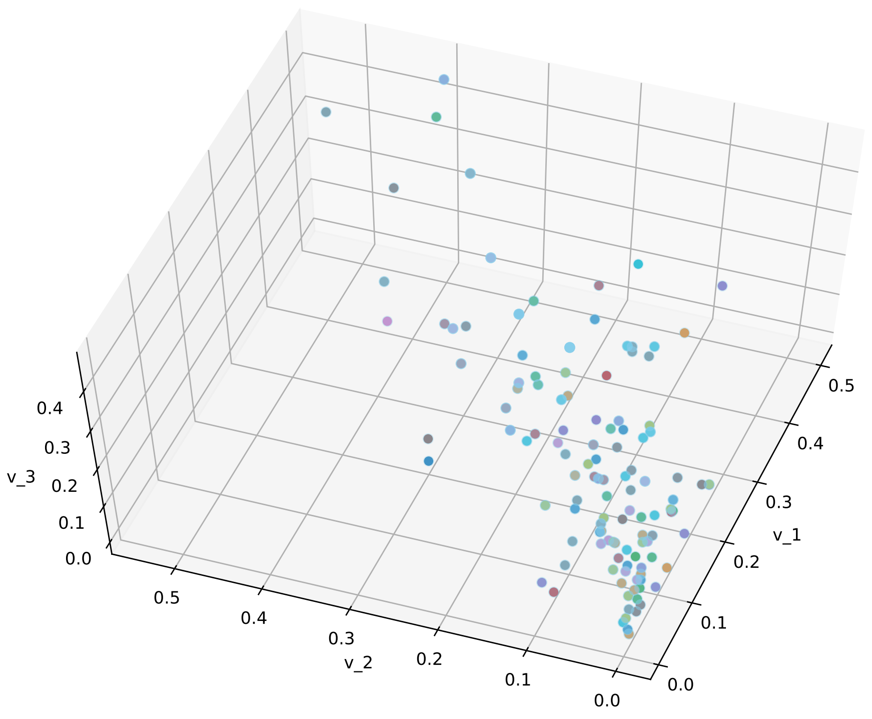
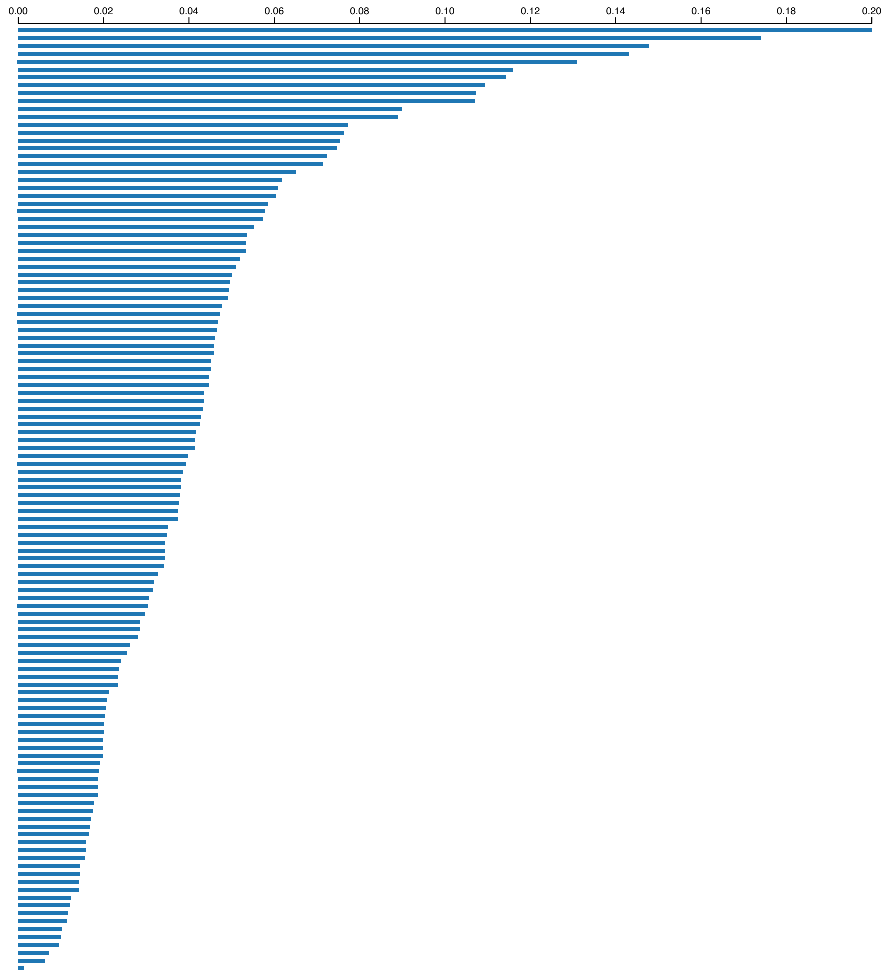


**Figure 7.** 3D vector plot of the 120 MAC Contig Vectors.



Elapsed time: 0.116 seconds

**Figure 8.** Persistent intervals in dimension 0 [20].

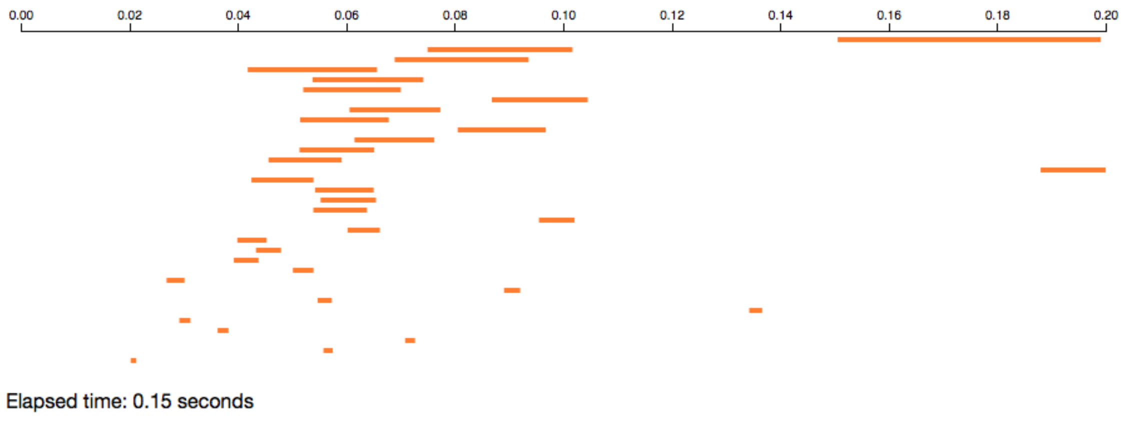


**Figure 9.** Persistent intervals in dimension 1 [20].

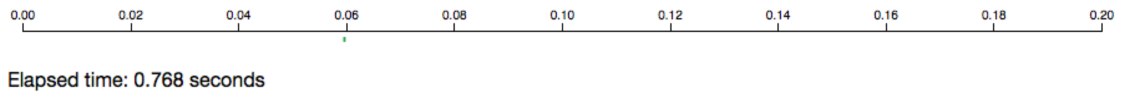


**Figure 10.** Persistent intervals in dimension 2 [20].

# Chapter 5
# Conclusion

We analyzed patterns of unorthodox overlapping gene segments in *Oxytricha trifallax* using topological data analysis (TDA). We obtained a data set $\mathcal{D}_4$ consisting of 120 MAC contigs that have overlapping MDS pairs in the MIC contig. From the Vietoris-Rips barcode analysis of our 120 MAC contigs for the 0-dimensional homology group, for example, we found that there are 10 connected components for $\epsilon = 0.1$. The zero-dimensional homology group geometrically corresponds to MAC contig clusters. The number of clusters reduces to 5 at $\epsilon = 0.12$. Most of the loops in the simplicial complexes for the 1-dimensional homology group appeared in the range of $\epsilon = 0.04$ to $\epsilon = 0.1$. There are two loops appearing at $\epsilon = 0.15$ and $\epsilon = 0.19$ that are persistent while all of the components in the 0-dimensional homology group are merged after approximately $\epsilon = 0.175$. There is one barcode for $\epsilon = 0.059$ for the 2-dimensional homology group that has a short persistence. The presence of non-trivial homology groups implies that the distribution of $V(A_n)$ is not uniform, which can be observed in Figure 8. Now other TDA methods such as dendogram analysis can be applied to our Vietoris-Rips filtration to further identify how many and which MAC contigs appear in each cluster, thus giving further insight into the patterns of unorthodox overlapping gene segments in *Oxytricha trifallax*.

## References

[1] A. D. Goldman, E. M. Stein, J. R.Bracht, and L. F. Landweber, Programmed Genome Processing in Ciliates, in *Discrete and Topological Models in Molecular Biology* (eds. N. Jonoska and M. Saito), **Springer** (2014) 289-307.

[2] D. Goldfarb, An Application of Topological Data Analysis to Hockey Analytics, arXiv:1409.7635 (2014).

[3] E. Munch, A User's Guide to Topological Data Analysis, *Journal of Learning Analytics*, **4:2** (2017) 47-61.

[4] H. Edelsbrunner and J. Harer, *Computational Topology An Introduction*, **American Mathematical Society** (2009).

[5] H. Nordberg, M. Cantor, S. Dusheyko, S. Hua, A. Poliakov, I. Shabalov, T. Smirnova, IV. Grigoriev, I. Dubchak, The Genome Portal of the Department of Energy Joint Genome Institute: 2014 updates, *Nucleic Acids Res*, **42:1** D26-31 (2014).

[6] I. Petre and G. Rozenberg, Gene Assembly in Ciliates, *Scholarpedia*, **5:1** (2010) 9269.

[7] J. Braun, L. Nabergall, R. Neme, L. Landweber, M. Saito, N. Jonoska, Russian Doll Genes and Complex Chromosome Rearrangements in *Oytricha trifallax*, *G3: Genes, Genomes, Genetics*, **8: 5** (2018) 1669-1674.

[8] J. Burns, D. Kukushkin, X. Chen, L. F. Landweber, M. Saito, and N. Jonoska, Reoccurring Patterns Among Scrambled Genes in the Encrypted Genome of the Ciliate *Oxytricha trifallax*, *Journal of Theoretical Biology*, **410** (2016) 171–180.

[9] J. R. Munkres, *Elements of Algebraic Topology*, **Addison- Wisely Publishing Company** (1984).

[10] H. Mustafa, N. Jonoska, D. Kukushkin, M. Saito, *Graph Based Analysis for Gene Segment Organization In a Scrambled Genome*, arXiv:1801.05922 (2018).

[11] P. Doshi and W. Zadrozny, Movie Genre Detection Using Topological Data Analysis, *Lecture Notes in Computer Science*, **11171** (2018).

[12] R. Ghrist, Three Examples of Applied and Computational Homology, *Nieuw Arch/ Wiskd*, **5:9** (2008) 122-125.

[13] T. Vialar, *Handbook of Mathematics*, **Books on Demand** (2015).

[14] V. Nanda and R. Sazdanovič, Simplicial Models and Topological Inference in Biological Systems, in *Discrete and Topological Models in Molecular Biology* (eds. N. Jonoska and M. Saito), **Springer** (2014) 109-141.

[15] X. Chen, J. R. Bracht, A. D. Goldman, E. Dolzhenko, D. M. Clay, E. C. Swart, D. H. Perlman, T. G. Doak, A. Stuart, C. T. Amemiya, R. P. Sebra, L. F. Landweber, The Architecture of a Scrambled Genome Reveals Massive Levels of Genomic Rearrangement during Development, *Cell*, **158:5** (2014) 1187-1198.

[16] Figure 1, `https://evolutionnews.org/2014/12/ciliate_organis_1/`

[17] Figure 2, `http://knot.math.usf.edu/background/BGbio.pdf`

[18] Figure 3, `https://en.wikipedia.org/wiki/Simplex`

[19] Figure 4, `http://mathworld.wolfram.com/SimplicialComplex.html`

[20] Figure of Rips barcodes, `http://live.ripser.org`

[21] Figure 6, `http://knot.math.usf.edu/multimedia/An_Annotation_Algorithm_DNA_Rearrangements.pdf`

## Appendix A

**In appendix A we have included the Python scripts used in the calculation of the 120 MAC contig vectors $V(A_n)$ and the distance matrix, consisting of the distances between each pair of the 120 MAC contig vectors $V(A_n)$. The vector $(v_1(A_n), v_2(A_n), v_3(A_n))$ are labeled as entry1, e2 and e3 respectively in the Python script.**

**This first block of code calculates vector entry1.**

```
import pandas as pd
import numpy as np
df0=pd.read_csv("df0.csv", index_col=0)
d={}
d1={}
d2={}
d3={}
D=[]
Dprime=[]
GHI=[]
count={}
gap={}
for contig_num_MDS in df0["contig_num_MDS"]:
    d[contig_num_MDS]=[]
#This is initiation of a empty dictionary.
for index, row in df0.iterrows():
    if index != 0:
        d[row["contig_num_MDS"]].append
        (int(row["name"].split("_")[2]))
#The code above is pulling
#the index values the "i"
#of a each MDS corresponding each MAC contig.
#For example, if there are three MDSs for
#one MAC contig it would pull numbers 1,2 and 3.
#and putting them into a list.
#d is a dict consist of mac contig names
#and list(i's) value as the value of
#the dictionary d.
Rs2=[]
for name,highest_mds_num in d.items():
    mds=[(name, max(highest_mds_num))]
    v=list(mds)
    Ts2=[]
    for tups in v:
        Ts2.append(tups)
        Rs2.append(tups)
        #Note this just gives a
        #list of tuples consisting of
        #( MAC contig name, highest
        #index "i" )
        #with respect to all of the
        #indices for that
        #MAC contig.
#print(Rs2)
#print(dict(Rs2)) #This is the
#number of MDSs per mac contig name
#which is the denominator of vector
```

```
#entry 1.
bla=dict(Rs2)
for contig_num_MDS in df0["contig_num_MDS"]:
    d1[contig_num_MDS]=[]
for index, row in df0.iterrows():
    if index != 0:
        d1[row["contig_num_MDS"]].append
        ((row["contig_num_MDS"],
        int(row["col4start"]),
        int(row["col5end"]),
        index,
        int(row["name"].split("_")[2]),
        row["orientation"]))
#print(len(d1)) #Note that the length of d1 is #409
#since there are 409 unique mac contig names.
for contig_num_MDS in d1:
    for mds_1 in d1[contig_num_MDS]:
        for mds_2 in d1[contig_num_MDS]:
            if mds_1 != mds_2:
                if min(mds_1[2],mds_2[2])
                -max(mds_1[1],mds_2[1])+1 > 0:
                    Printstuffred=mds_1[5] ==
                    mds_2[5]
                    and mds_1[4]-
                    mds_2[4] == 1
                    and mds_1[1] <
                    mds_2[1]
                    and
                    mds_1[5] =="-"
                    #mds_1 and mds_2 are
                    #consecutive
                 #mds_1= i+1 starts
                #before mds_2 = i
                 #both orient neg
                    Printstuffblue=
                    mds_1[5] == mds_2[5]
                    and mds_1[4]-
                    mds_2[4] == 1
                    and mds_1[1] >
                    mds_2[1]
                    and
                    mds_1[5] == "+"
                    #same orientation
                    #mds_1 and mds_2
                    #are consecutive
                    #mds_1= i+1 starts
                    #after mds_2 = i
                    #both orient pos
                    if not Printstuffred
                    and not Printstuffblue:
                        c=(mds_1,mds_2),
                        min(mds_1[2],mds_2[2])-
                        max(mds_1[1],mds_2[1])+1
#"contig_num_MDS length of overlap
#between mds pairs"
                        #print(mds_1,mds_2,
            min(mds_1[2],mds_2[2])-
            max(mds_1[1],mds_2[1])+1)
                        L=list(c)
                        B=[]
                        C=[]
                        for element in L:
                            B.append(element)
                            C.append(element)
                        D.append(B)
#print(D)
#D is a list of list
#consisting of overlapping MDS pairs.
inpud={lst[0]:lst[1] for lst in D}
#print(inpud)
inpud2 = {v: k for k, v in inpud.items()}
#exchange keys, values
inpud = {v: k for k, v in inpud2.items()}
#exchange again
#print(inpud)
#Note that ipud now is a dictionary
#with distinct overlapping MDS pairs
#as keys and the overlapping values between
#the MDS pair as values of the dictionary.
#(Note the three different dictionary operations
#starting with inpud -> inpud2 -> inpud are
#absolutely necessary since
#we have double counting with MDS overlapping #pairs.So those
#three dictionary operations get rid of the #double counting.
#Now I need to get the number of
#overlapping pairs per name.
from collections import Counter
count=Counter(key[0][0] for key in inpud.keys())
#print(dict(count))
#This converts
```

```
#Counter({’Contig8398.0_MDS’:3,
#’Contig18264.0_MDS’:1}) to just inside.
#dict(count) above counts the number
#of overlapping MDS pairs per MAC contig.
entry1={k: float(count[k])/bla[k] for k in count}
#This is number of overlapping MDS pairs / the #number of MDSs
#which is recorded for each of the 120 MAC contigs.
```

**This second block of code demonstrates the calculation of the second vector entry.**

```
#Now  inpud has the overlapping MDS pairs as
#keys in the dictionary. Now to
#iterate over the pairs:
for key in inpud.keys():
    list(key)
    ABC=[]
    DEF=[]
    for element in list(key):
#This is taking the
#overlapping pairs #(lists)
#and putting them into a list of lists.
        ABC.append(element)
        DEF.append(element)
    GHI.append(ABC)
#print(GHI) #GHI is a list
#of list of overlapping MDS pairs.
#Now I need to compute the
#absolute value of i minus j
#and subtract 1
#if the signs are equal.
#Otherwise if the signs
#differ then it’s just the
```

```
#absolute value of i minus j.
#Then we take
#the sums of all computations.
#These computations above are computing the gaps #for each MAC contig.
#The i and the j correspond to the indices of the #overlapping MDS pair.
#This is done for each of the 120 MAC contigs.
U=[]
K=[]
#A Gap for a mac contig  is
#computed by take sum of abs(i-j)-1.
#If the signs equal each other.
A=([[(x[0],x[4],x[5]) for x in el] for el in GHI])
#This is taking the
#(MAC contig name,i value,sign)from GHI
#print(A)
E=dict([(y,z) for y,z in A])
#E is a dictionary with the key
#and values as the overlapping MDS pairs.
#Each key and value is an overlapping MDS pair.
#print(E)
for tup1,tup2 in E.items():
    if tup2[2]==tup1[2]:
#If the signs for mac contigs
#equal each other...
        a=(tup1,tup2,abs(tup1[1]-tup2[1])-1)
#This is the pair of overlapping
#MDS pairs and computation.
        list(a)
#This is a list of list
#of the pair of
```

```
#overlapping MDS pairs
#and computation.
        #print(list(a))
        F=[]
        G=[]
        for element in list(a):
            F.append(element)
            G.append(element)
        U.append(F)
#print(U)
for tup1,tup2 in E.items():
    if tup2[2]!=tup1[2]:
#If the signs for mac contigs
#do not equal each other...
        b=(tup1,tup2,abs(tup1[1]-tup2[1]))
#This is the pair of
#overlapping MDS pairs
#and computation.
        list(b)
#This is a list of list of the
#pair of overlapping MDS pairs
#and computation.
        #print(list(b))
        H=[]
        J=[]
        for element in list(b):
            H.append(element)
            J.append(element)
        K.append(H)
#print(K)
```

```
W=K+U
#print(W)
q={listz[0]:listz[2] for listz in W}
#q is a dictionary consisting
#of a tuple and computation value
#including the mac contigs name.
#It didn’t matter
#which one we pulled from W
#we just needed it to be able to
#pull out the MAC contig name for the key.
#The value is from the computation.
#Note the computation does not
#include the sum of gaps yet!
#No Sum has been taken yet.
#print(q)
s={}
M=[]
for x,y in q.items():
    s.setdefault(x[0],[]).append(y)
#print(s) #s is a dictionary
#combining all index  values so that a sum
#can be taken over it if the MAC contig name
#has more than one index.
#Now we need to sum up the values.
#I.e. we need to sum up the value=lists
#here in s.
for k,v in s.items():
    result=(k,sum(v))
#This is the sum of the gaps
#for the second coordinate entry of the vector.
```

```
    list(result)
    Z=[]
    V=[]
    for element in list(result):
        Z.append(element)
        V.append(element)
    M.append(Z)
#print(M)
#We have a list of list of sum
#of the gaps per MAC contig.
#print({listz[0]:listz[1] for listz in M})
#This code takes list of lists
#and converts the inner list
#into direct dictionary entries
#to make a dictionary.
#This is entry2 the result
#for the second entry of our vector
#it is the sum of the gaps per MAC contig name.
entry2={listz[0]:listz[1] for listz in M}
#This is the dictionary that is the
#sum of the gaps.
print(entry2)
#NOW we need sum of gaps/(n*(n-1)/2)
#where n is the number of MDSs per MAC contig #name.
#This modification is as of feb 1st.
en=({k: (v*(v-1))/2 for k,v in bla.items()})
#This is the denomiator v*(v-1)/2 for v
#being the number of MDSs.
print(en)
print({k: float(entry2[k])/en[k] for
k in entry2})
entry22=print({k: float(entry2[k])/en[k]
for  k in entry2})
#This is the sum of the
#gaps/v*(v-1)/2 where v is the # MDSs.
#print bla=(dict(Rs2)) #This is the
#number of MDSs per MAC contig  name the denominator of vector entry 1.
```

**This last block of code computes the third vector entry and the distance matrix.**

```
#Below involves the computation of vector entry 3.
#This is entry 3:
#total number overlapping MDS pairs/
#(total length of the MAC contig) which is approximately equal to
# =(sum(length of overlap between MDS pairs)/
# the sum of length of MDSs.
#Below is computing the sum of overlap between
#the MDS pairs.
#inpud is a dictionary consisting
#of overlapping MDS pair as the key
#and overlapping value of the pair as the value.
print(inpud)
#Now i need to get all overlaps
#together per MAC contig name in a list
#to be able to sum the length.
s2={}
M2=[]
for x,y in inpud.items():
    s2.setdefault(x[0][0],[]).append(y)
print(s2)
#s2 is a dictionary consisting
#of MDS overlap pair mac contig name
```

```
#and length of overlap per that MAC contig name
#and the sums of overlap per MAC contig name as
#the value in a list format now.
#We need to sum the list.
for k,v in s2.items():
    result=(k,sum(v))
    list(result)
    Z2=[]
    V2=[]
    for element in list(result):
        Z2.append(element)
        V2.append(element)
    M2.append(Z2)
print(M2)
print({listz[0]:listz[1] for listz in M2})
#This code takes list of lists and converts
#the inner list into direct dict entries to
#make a dict.
#This last print statement above is the
#numerator of entry3.
z={listz[0]:listz[1] for listz in M2}
#Now we need to compute the sum of the
#length of MDSs so below we do :
d3={}
for contig_num_MDS in df0["contig_num_MDS"]:
    d3[contig_num_MDS]=[]
#This is initiation of a dictionary.
for index, row in df0.iterrows():
#This is taking rows from df0 and
#putting specific things into dictionary d3.
```

```
    if index != 0:
        d3[row["contig_num_MDS"]].append
        ((row["contig_num_MDS"],
        abs(int(row["col4start"])-
        int(row["col5end"]))+1))
print(d3)
#Note d3 consist
#of abs(start-end)+1 per contig name
#(i.e for each row in df0).
#This code below compute the sum of
#all such rows per name.
v4=[]
for value in d3.values():
    #print(value)
    #print({value[0][0]:sum(num[1]
    for num in value)} )
    #is mac contig name
    v1=[(value[0][0],
    sum(num[1] for num in value))]
    v2=list(v1)
    print(v2)
    v3=[]
    for element in v2:
        v3.append(element)
        v4.append(element)
print(v4)
#v4 is just a list with mac contig names
#and the sum of length of MDS for that name.
y=dict(v4)
print(y)
```

```
x={k: float(z[k])/y[k] for k in z}
#This is calculating sum of length
#of overlap between MDS pairs
#/sum of lengths of MDS.
print(x)
import pandas as pd
import numpy as np
G3=[]
G4=[]
entry1={k: float(count[k])/bla[k] for k in count}
e2={k: float(entry2[k])/en[k] for  k in entry2}
e3={k: float(z[k])/y[k] for k in z}
df=pd.DataFrame.from_dict(entry1,
orient='index', dtype=None,
columns=["entry1"])
#df=pd.DataFrame.from_dict(entry1,
orient='columns', dtype=None,
columns=None) #
#print(df)
#b=pd.DataFrame.from_dict(entry2,
orient="index", columns=["entry2"])
df["entry2"]=pd.Series(e2)
#print(df)
df["entry3"]=pd.Series(e3)
print(df)
#Note the df below indicates
#the 3 different vector entries.
c1=[tuple(x) for x in df.to_records(index=True)]
#print(tup1[0])# tup1[0]= name, name1, name2,name3
for tup1 in c1:
    for tup2 in c1:
        if tup1[0]!=tup2[0]:
            #b=(((tup2[1]-tup1[1])*
            (tup2[1]-tup1[1])+((tup2[2]-tup1[2])*
            (tup2[2]-tup1[2]))+((tup2[3]-tup1[3])*
            (tup2[3]-tup1[3])))**0.5)
            #print(((tup1[0],tup2[0]),
            ((tup2[1]-tup1[1])*(tup2[1]-tup1[1])+
            ((tup2[2]-tup1[2])*(tup2[2]-tup1[2]))+
            ((tup2[3]-tup1[3])*
            (tup2[3]-tup1[3])))**0.5))
            a3=([tup1[0],tup2[0],
            ((tup2[1]-tup1[1])*(tup2[1]-tup1[1])+
            ((tup2[2]-tup1[2])*(tup2[2]-tup1[2]))+
            ((tup2[3]-tup1[3])*
            (tup2[3]-tup1[3])))**0.5])
#This is a list of distance tuples.
            #list(a3)
            E4=[]
            C14=[]
            for element in list(a3):
#This can be used again to get
#the sum(of lengths of overlap between the pairs
#where list(a)=list(c) in our case we're
#appending each overlap values to get
#the pair of lists of lists this
#is going to be new G.
                E4.append(element)
                C14.append(element)
            G4.append(E4)
```

```
print(G4)
#G4 is a list of list
#of distance tuples i.e.
#the distances between
#each MAC contig vector.
#Now I am putting G4 our list
#of list of distance tuples
#into a pandas df below:
#note d(name, name)=0 so in order to
#avoid duplicate info I set that
#to zero manually with fillna 0 line.
df = pd.DataFrame(G4, columns=["a","b","c"])
#print(df)  #a is the mac contig name ,
#b is the mac contig name and c is the distance between them.
df.set_index("a",inplace=True)
df1=df.pivot_table(index='a', columns='b', values='c',
fill_value='')
#print(df1)
df1.replace('', np.nan, inplace=True)
#print(df1)
df1.fillna(0, inplace=True)
#print(df1)
points=df1.values
print(points)
#points is a 2D-array used for rips.
#distancematrix = pd.DataFrame(points)
distancematrix1 = pd.DataFrame(points)
#This coverts the 2D
#array into a pandas df in order
#to save to a csv to run in Rips.
```

```
#print(distancematrix1)
distancematrix1.to_csv("distancematrix1.csv", header=None,
index=False)
```

Table 1: The MAC Contig Vector Entries

| | entry1 | entry2 | entry3 |
|---|---|---|---|
| Contig7253.0_MDS | 0.08333333333333333 | 0.030303030303030304 | 0.06265206812652069 |
| Contig8398.0_MDS | 0.2 | 0.2 | 0.25101214574898784 |
| Contig9546.0_MDS | 0.2 | 0.022222222222222223 | 0.020229265003371546 |
| Contig6340.0_MDS | 0.05263157894736842 | 0.09941520467836257 | 0.0003303600925008259 |
| Contig20.1_MDS | 0.47058823529411764 | 0.07352941176470588 | 0.1630960608154803 |
| Contig732.0_MDS | 0.05263157894736842 | 0.002844950213371266 | 0.02123705866737457 |
| Contig3.1_MDS | 0.1956521739130435 | 0.19710144927536233 | 0.27379606601853945 |
| Contig225.0_MDS | 0.0625 | 0.09166666666666666 | 0.06434599156118144 |
| Contig346.0_MDS | 0.25 | 0.0 | 0.022388059701492536 |
| Contig2582.0_MDS | 0.15 | 0.02631578947368421 | 0.09024211298606016 |
| Contig3295.0_MDS | 0.013513513513513514 | 0.0022213994816734544 | 0.020671834625323 |
| Contig635.0_MDS | 0.24324324324324326 | 0.16066066066066065 | 0.18451498071788786 |
| Contig75.1_MDS | 0.3111111111111111 | 0.22626262626262628 | 0.2871299707788083 |
| Contig4777.0_MDS | 0.3333333333333333 | 0.1503267973856209 | 0.10127931769722814 |
| Contig12898.0_MDS | 0.17647058823529413 | 0.0 | 0.008908685968819599 |
| Contig12633.0_MDS | 0.2 | 0.08888888888888889 | 0.15670684790972045 |
| Contig1463.1_MDS | 0.2222222222222222 | 0.27952480782669464 | 0.32639593908629444 |
| Contig1113.1_MDS | 0.3125 | 0.11666666666666667 | 0.21302285467874083 |
| Contig3658.0_MDS | 0.19230769230769232 | 0.11692307692307692 | 0.10903614457831326 |
| Contig4797.0_MDS | 0.030303030303030304 | 0.013257575757575758 | 0.004217432052483599 |
| Contig10518.0_MDS | 0.25 | 0.1 | 0.11644859813084112 |
| Contig1807.0_MDS | 0.012195121951219513 | 0.0 | 0.01132835445162373 |

Table 1 continued from previous page

| | | | |
|---|---|---|---|
| Contig8270.0_MDS | 0.0967741935483871 | 0.008602150537634409 | 0.0750164923192913 |
| Contig1240.0.2_MDS | 0.25 | 0.3 | 0.28554107305244014 |
| Contig6856.0_MDS | 0.0625 | 0.11666666666666667 | 0.00034782608695652176 |
| Contig13850.0_MDS | 0.2222222222222222 | 0.0 | 0.06818181818181818 |
| Contig3653.0_MDS | 0.3333333333333333 | 0.4 | 0.11528934924566539 |
| Contig6202.0_MDS | 0.3076923076923077 | 0.11538461538461539 | 0.2336740488358887 |
| Contig11515.0.1_MDS | 0.11764705882352941 | 0.058823529411764705 | 0.05289672544080604 |
| Contig8494.0_MDS | 0.2727272727272727 | 0.10909090909090909 | 0.29247515380974914 |
| Contig8059.0_MDS | 0.11764705882352941 | 0.10294117647058823 | 0.11974891356832448 |
| Contig7210.0_MDS | 0.4 | 0.08888888888888889 | 0.14128256513026052 |
| Contig3631.0_MDS | 0.125 | 0.03260869565217391 | 0.12411347517730496 |
| Contig706.0_MDS | 0.14285714285714285 | 0.19047619047619047 | 0.29656040268456374 |
| Contig14704.0_MDS | 0.25 | 0.0 | 0.021026072329688814 |
| Contig1491.0_MDS | 0.043478260869565216 | 0.003952569169960474 | 0.017398684489709316 |
| Contig3550.0_MDS | 0.125 | 0.06666666666666667 | 0.04030479246039703 |
| Contig4620.0_MDS | 0.1875 | 0.06666666666666667 | 0.07370892018779343 |
| Contig10925.0_MDS | 0.021739130434782608 | 0.007729468599033816 | 0.0352882703777336 |
| Contig10119.0.1_MDS | 0.3 | 0.08888888888888889 | 0.2594782608695652 |
| Contig849.0_MDS | 0.125 | 0.0 | 0.18954641033343347 |
| Contig3400.0.0_MDS | 0.125 | 0.0 | 0.0010506960861570791 |
| Contig5949.0_MDS | 0.125 | 0.03571428571428571 | 0.012171684817424727 |
| Contig663.1_MDS | 0.15 | 0.09473684210526316 | 0.15719144800777454 |
| Contig15402.0_MDS | 0.25 | 0.16666666666666666 | 0.07902439024390244 |
| Contig4296.0_MDS | 0.1875 | 0.016666666666666666 | 0.13539823008849558 |
| Contig16750.0_MDS | 0.21052631578947367 | 0.15789473684210525 | 0.11405759908753921 |
| Contig4233.0_MDS | 0.3333333333333333 | 0.10476190476190476 | 0.17520325203252032 |
| Contig8394.0_MDS | 0.13333333333333333 | 0.0761904761904762 | 0.08312400091345055 |
| Contig334.0_MDS | 0.1875 | 0.05 | 0.23861029170763684 |
| Contig3972.0_MDS | 0.06666666666666667 | 0.009195402298850575 | 0.0638682252922423 |

Table 1 continued from previous page

| | | | |
|---|---|---|---|
| Contig5277.0_MDS | 0.046511627906976744 | 0.008859357696566999 | 0.0706896551724138 |
| Contig10027.0_MDS | 0.043478260869565216 | 0.003952569169960474 | 0.0535224153705398 |
| Contig8403.0_MDS | 0.14285714285714285 | 0.09523809523809523 | 0.2530541012216405 |
| Contig8215.0_MDS | 0.14285714285714285 | 0.08791208791208792 | 0.16754270696452037 |
| Contig19703.0_MDS | 0.16666666666666666 | 0.06666666666666667 | 0.028079132099553285 |
| Contig1046.1_MDS | 0.13043478260869565 | 0.04743083003952569 | 0.12485578771952313 |
| Contig6626.0_MDS | 0.5 | 0.5333333333333333 | 0.3277706461275139 |
| Contig3060.0_MDS | 0.15789473684210525 | 0.011695906432748537 | 0.10215154349859681 |
| Contig1488.0_MDS | 0.5 | 0.17857142857142858 | 0.12335746849021186 |
| Contig10852.0_MDS | 0.2 | 0.1111111111111111 | 0.11242093156986774 |
| Contig4341.0_MDS | 0.10526315789473684 | 0.023391812865497075 | 0.11378205128205128 |
| Contig13923.0_MDS | 0.2 | 0.17777777777777778 | 0.27200902934537247 |
| Contig6242.0_MDS | 0.1875 | 0.175 | 0.16586643387167177 |
| Contig5986.0_MDS | 0.2 | 0.08888888888888889 | 0.2262582056892779 |
| Contig1089.0.5_MDS | 0.2727272727272727 | 0.2857142857142857 | 0.2522338524380904 |
| Contig1085.0_MDS | 0.0625 | 0.025 | 0.08602639963586708 |
| Contig6730.0_MDS | 0.14285714285714285 | 0.10989010989010989 | 0.09653465346534654 |
| Contig19574.0_MDS | 0.09375 | 0.12701612903225806 | 0.15598290598290598 |
| Contig11791.0_MDS | 0.16666666666666666 | 0.06666666666666667 | 0.14636913767019666 |
| Contig7358.0_MDS | 0.3333333333333333 | 0.16666666666666666 | 0.23771790808240886 |
| Contig7030.0_MDS | 0.07692307692307693 | 0.01282051282051282 | 0.0599250936329588 |
| Contig2948.0_MDS | 0.5 | 0.4090909090909091 | 0.37209302325581395 |
| Contig1137.0_MDS | 0.07692307692307693 | 0.038461538461538464 | 0.08987752161383285 |
| Contig11433.0_MDS | 0.09090909090909091 | 0.0 | 0.005764796310530361 |
| Contig6260.0_MDS | 0.42857142857142855 | 0.42857142857142855 | 0.2944489139179405 |
| Contig889.1_MDS | 0.12 | 0.03 | 0.23060884070058382 |
| Contig2133.0_MDS | 0.07692307692307693 | 0.08974358974358974 | 0.10714285714285714 |
| Contig3849.0_MDS | 0.041666666666666664 | 0.013297872340425532 | 0.06024699599465955 |
| Contig7365.0_MDS | 0.18181818181818182 | 0.14545454545454545 | 0.2797266514806378 |

Table 1 continued from previous page

| | | | |
|---|---|---|---|
| Contig13810.0_MDS | 0.25 | 0.21428571428571427 | 0.2501141031492469 |
| Contig13175.0_MDS | 0.1 | 0.06666666666666667 | 0.10916334661354582 |
| Contig13330.0_MDS | 0.125 | 0.07142857142857142 | 0.15952773201537618 |
| Contig1822.0.0_MDS | 0.18181818181818182 | 0.01818181818181818 | 0.05004389815627744 |
| Contig597.0_MDS | 0.125 | 0.17857142857142858 | 0.2746858168761221 |
| Contig795.1_MDS | 0.16666666666666666 | 0.06060606060606061 | 0.1655975291043003 |
| Contig705.1_MDS | 0.2857142857142857 | 0.26373626373626374 | 0.4111111111111111 |
| Contig3929.0_MDS | 0.03225806451612903 | 0.008602150537634409 | 0.04137097991846595 |
| Contig11408.0_MDS | 0.25 | 0.16666666666666666 | 0.23061786698150336 |
| Contig118.0_MDS | 0.16666666666666666 | 0.13333333333333333 | 0.43951367781155015 |
| Contig20706.0_MDS | 0.2 | 0.3 | 0.004182642035552457 |
| Contig3281.0_MDS | 0.05555555555555555 | 0.026143790849673203 | 0.06402353600448304 |
| Contig7189.0_MDS | 0.07692307692307693 | 0.01282051282051282 | 0.020007020007020006 |
| Contig524.1_MDS | 0.18181818181818182 | 0.2545454545454545 | 0.313953488372093 |
| Contig2156.0.1_MDS | 0.5 | 0.4 | 0.4615568862275449 |
| Contig764.0_MDS | 0.375 | 0.32142857142857145 | 0.4534842589317297 |
| Contig649.0_MDS | 0.05714285714285714 | 0.010084033613445379 | 0.0802675585284281 |
| Contig2477.0_MDS | 0.08695652173913043 | 0.06324110671936758 | 0.15359741309620048 |
| Contig6256.0_MDS | 0.09090909090909091 | 0.01818181818181818 | 0.12161115414407436 |
| Contig6633.0_MDS | 0.09090909090909091 | 0.03463203463203463 | 0.09093721002165171 |
| Contig13154.0_MDS | 0.2 | 0.2 | 0.43531694695989653 |
| Contig3582.0_MDS | 0.125 | 0.10714285714285714 | 0.1951861602106055 |
| Contig15276.0_MDS | 0.23076923076923078 | 0.19230769230769232 | 0.23752569613156418 |
| Contig16535.0_MDS | 0.125 | 0.07608695652173914 | 0.2006015733456733 |
| Contig8451.0_MDS | 0.2857142857142857 | 0.14285714285714285 | 0.06251662676243681 |
| Contig2461.0_MDS | 0.2 | 0.3 | 0.0650730411686587 |
| Contig47.0_MDS | 0.08333333333333333 | 0.06060606060606061 | 0.10613751730503 |
| Contig7825.0_MDS | 0.16666666666666666 | 0.13333333333333333 | 0.16865742952699475 |
| Contig2213.0_MDS | 0.04878048780487805 | 0.03414634146341464 | 0.10943667025475422 |

Table 1 continued from previous page

| | | | |
|---|---|---|---|
| Contig14316.0_MDS | 0.18181818181818182 | 0.18181818181818182 | 0.15273095077545515 |
| Contig4928.0_MDS | 0.09523809523809523 | 0.06666666666666667 | 0.1147594278283485 |
| Contig738.1_MDS | 0.08333333333333333 | 0.045454545454545456 | 0.11706349206349206 |
| Contig1827.0_MDS | 0.18181818181818182 | 0.09090909090909091 | 0.23387703889585948 |
| Contig1880.0_MDS | 0.375 | 0.17857142857142858 | 0.25384122919334184 |
| Contig20128.0_MDS | 0.07692307692307693 | 0.01282051282051282 | 0.07802469135802469 |
| Contig347.1_MDS | 0.09523809523809523 | 0.009523809523809525 | 0.13762057877813505 |
| Contig5740.0_MDS | 0.09523809523809523 | 0.014285714285714285 | 0.11893369788106631 |
| Contig4952.0_MDS | 0.2857142857142857 | 0.38095238095238093 | 0.2961658841940532 |
| Contig11320.0_MDS | 0.25 | 0.07142857142857142 | 0.142518837459634 |
| Contig14221.0_MDS | 0.21428571428571427 | 0.06593406593406594 | 0.17099923220357574 |