CSIC 5011 - HW8

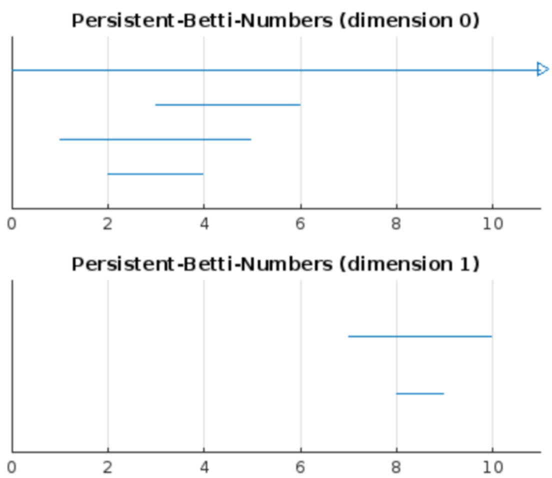
NGUYEN Huu Canh
SID: 20818623

Q1. Javaplex setup and playground

a)

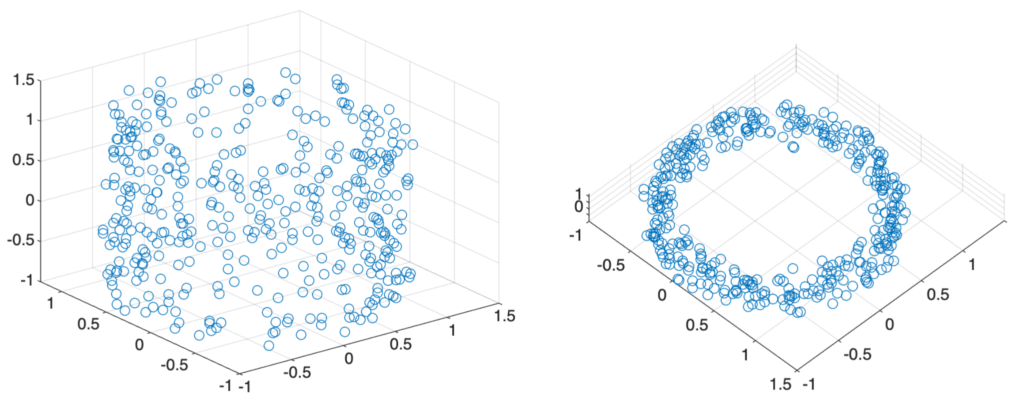
```
>> stream = api.Plex4.createExplicitSimplexStream();
>> stream.addVertex(0,0);
>> stream.addVertex(1,1);
>> stream.addVertex(2,2);
>> stream.addVertex(3,3);
>> stream.addElement([0,2],4);
>> stream.addElement([0,1],5);
>> stream.addElement([2,3],6);
>> stream.addElement([1,3],7);
>> stream.addElement([1,2],8);
>> stream.addElement([1,2,3],9);
>> stream.addElement([0,1,2],10);
>> stream.finalizeStream();
>> num_simplices = stream.getSize()

num_simplices =

    11
```

b)



Q2. Torus example

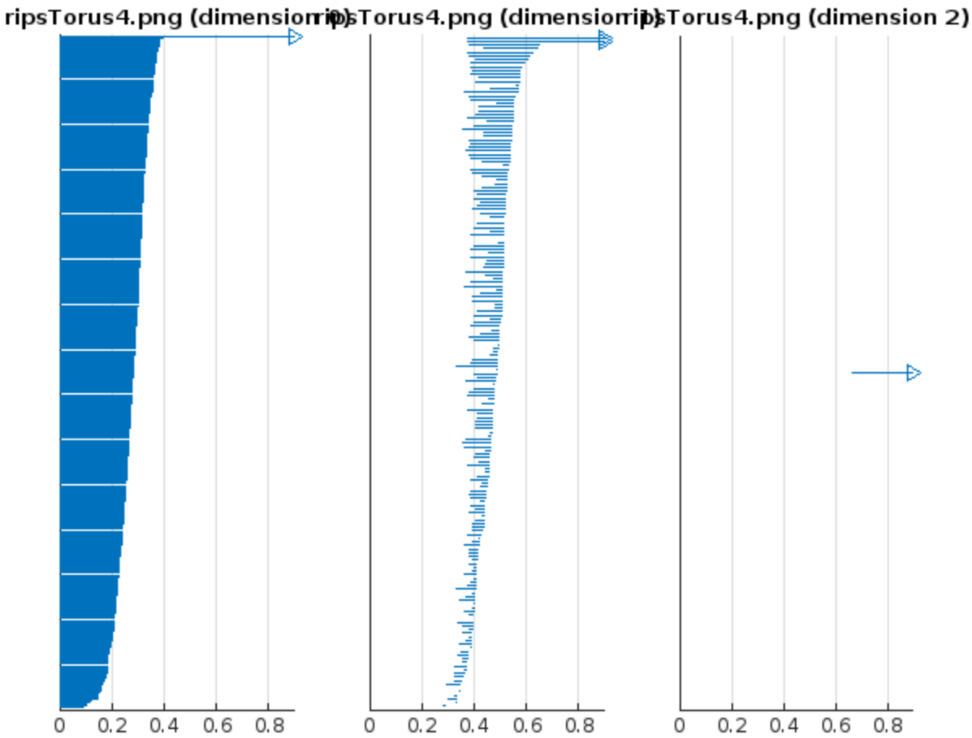
a)



b)

```
>> max_dimension = 3;
>> max_filtration_value = 0.9;
>> num_divisions = 1000;
>> stream = api.Plex4.createVietorisRipsStream(pointsTorusGrid, ...
max_dimension, max_filtration_value, num_divisions);
>> num_simplices = stream.getSize()

num_simplices =

       82479
```

c)   d)



e)

The diameter of this torus (before adding noise) is $\sqrt{8}$, so choosing tmax = 0.9 likely will not show all homological activity. However, the torus will be reasonably connected by this time. Note the semi-infinite intervals match the correct numbers $\mathrm{Betti}_0 = 1$, $\mathrm{Betti}_1 = 2$, $\mathrm{Betti}_2 = 1$ for a torus.

```
>> infinite_barcodes = intervals.getInfiniteIntervals();
>> betti_numbers_array = infinite_barcodes.getBettiSequence()

betti_numbers_array =

  3×1 int32 column vector

   1
   2
   1
```

This example makes it clear that the computed semi-infinite intervals do not necessarily persist until t = ∞: in a Vietoris-Rips stream, once t is greater than the diameter of the point cloud, the Betti numbers for VR(Z, t) will be $\mathrm{Betti}_0 = 1$, $\mathrm{Betti}_1 = \mathrm{Betti}_2 = \ldots = 0$. The computed semi-infinite intervals are merely those that persist until t = tmax.

Q3. Single Linkage Clustering and Persistent 0-Homology

a) *0-dimensional Persistent Homology of n points in a metric space is equivalent to single linkage clustering without labeling, i.e. permutation invariant;*

To compute the 0-dimensional persistent homology, we consider the set of all possible radius *r* and construct a sequence of nested balls with radius *r* around each point. We then consider the union of all balls with radius less than or equal to *r*, and compute the connected components of this union. The number of connected components will change as we increase the radius, and we can track how these changes occur by using the concept of persistence.

Now, let's consider the single linkage distance. This distance is defined as the minimum distance between any two points in different clusters. In other words, we start with all points as separate clusters, and then we merge the two closest clusters until we have *n* clusters (when we merge *n-1* times). If we consider the single linkage distance at different scales, we can construct a dendrogram or hierarchical clustering that shows how the clusters merge as we increase the distance threshold. The dendrogram will have *n* leaves, each corresponding to a point, and the ordering of the leaves will not affect the resulting clustering.

To see the equivalence between the 0-dimensional persistent homology and single linkage clustering, we can observe that the number of connected components in the union of balls with radius *r* is the same as the number of clusters in the dendrogram when the distance threshold is set to *2r*. This is because a ball with radius *r* contains all points that are at most distance *r* away from a given point, and the single linkage distance between two points in different balls with radius less than or equal to *r* is greater than *2r*.

The resulting clustering from the single linkage distance is permutation invariant. This is because the distance between two clusters only depends on the minimum distance between any two points in the two clusters, regardless of their order.