# PERSISTENT HOMOLOGY ON PHONOLOGICAL DATA: A PRELIMINARY STUDY

CATHERINE WOLFRAM

Abstract. Languages are full of structure, and thus it is worth asking: can that structure be studied mathematically? This paper is a preliminary study which attempts to do this. In particular, we will investigate the phonological structure of languages using persistent homology. Given a language, we will study its inventory of sounds in IPA. To do this, we will take a dataset of words in the language, and look at the contexts of each sound in the dataset. Using cosine similarity on the resulting context sets, we will embed the $n$ sounds of this language in $\mathbb{R}^n$. We will then apply persistent homology to the set of sounds (as embedded in $\mathbb{R}^n$) for each language. We will preliminarily investigate these results, and suggest some next directions of study. Ultimately, however, we do not yet have enough context data in each language. The embedding depends significantly on the context data, and thus we cannot come to further conclusions without more context data.

Contents



*Date*: August 14, 2017.

## 1. Introduction

This paper is an attempt to study the phonological structure of languages mathematically. Phonology is the study of the structure and system of relationships between sounds in a language (phonetics is the study of how these sounds are made). This paper will explore the topology of this system of relationships between for different languages. In particular, each language has an inventory of sounds. Using cosine similarity, we will construct a metric on this inventory of sounds. We will use this to explore the structure of the inventory of sounds in this language: are there some fixed number of types of sounds? Are all sounds equally interchangeable? How much does this vary between languages? Does this variation correspond to historical language families? This study is by no means complete, needs far more data, and is by no means the only way that such a thing can be done. But it is a first approach, and the methodology we describe can hopefully be useful in future ones.

For each of the seventeen languages that we consider, we will construct a metric on its sounds (in IPA). Given a dataset of words in that language, we will do this by looking at the "contexts" each sound appears in (in the dataset). To determine how similar two sounds are, we look at the cosine similarity between their context sets. We then use this to embed the $n$ sounds of a language in $\mathbb{R}^n$, or, equivalently, to construct an $n \times n$ symmetric matrix with 1s along the diagonal and entries between 0 and 1 off the diagonal. We will apply persistent homology to these for each language, creating a barcode graph, and study these.

Organizationally, this paper will begin in section 2 by briefly noting a few things in linguistics which will make it possible to understand what we are discussing. Section 3 will describe how we construct the metric on the sounds of each language. The final subsection of this section will be dedicated to decisions and judgment calls that were made that, should something like this be repeated in the future, could be done differently. We do not put this in the main body of the text so as to articulate the methodology in a more straightforward way, but include it because we believe that it is important information to communicate. In sections 4 and 5, we will shift back to mathematics, and describe simplicial and then persistent homology. In section 6, we will use the program Ripser to apply persistent homology to each language using the metric from section 3, and show our results. In section 7 we will try to understand these results from a few different perspectives. This section is exploratory and not conclusive. In section 8, we will suggest some next directions.

## 2. Linguistics

2.1. **The IPA.** The International Phonetic Alphabet, or IPA, is a standardized way to phonetically describe sounds based on how they are physically made. It

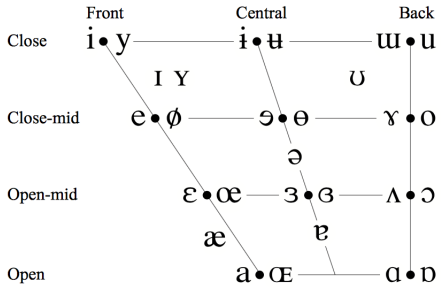
classifies consonants by three main parameters: manner of articulation (where air flow is restricted most), mode of articulation (how the air flows), and voice (whether your vocal cords are vibrating or not). For vowels, the three main parameters are height (how high your tongue is in your mouth), backness (how far back it is), and roundedness (whether your lips are rounded or not). In this way, the IPA is a useful way to approximate sounds. It is not perfectly precise. Mathematically, most of these parameters are continuous, but the IPA classifies them in discrete categories. It is also possible to have a sound that has more than one manner of articulation, and realistically when we speak we blur one sound into another. Regardless, the IPA is still an extremely useful tool. Here are the main IPA charts for consonants and vowels:

CONSONANTS (PULMONIC) © 2015 IPA

| | Bilabial | Labiodental | Dental | Alveolar | Postalveolar | Retroflex | Palatal | Velar | Uvular | Pharyngeal | Glottal |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Plosive | p b | | | t d | | ʈ ɖ | c ɟ | k g | q ɢ | | ʔ |
| Nasal | m | ɱ | | n | | ɳ | ɲ | ŋ | ɴ | | |
| Trill | ʙ | | | r | | | | | ʀ | | |
| Tap or Flap | | ⱱ | | ɾ | | ɽ | | | | | |
| Fricative | ɸ β | f v | θ ð | s z | ʃ ʒ | ʂ ʐ | ç ʝ | x ɣ | χ ʁ | ħ ʕ | h ɦ |
| Lateral fricative | | | | ɬ ɮ | | | | | | | |
| Approximant | | ʋ | | ɹ | | ɻ | j | ɰ | | | |
| Lateral approximant | | | | l | | ɭ | ʎ | ʟ | | | |

Symbols to the right in a cell are voiced, to the left are voiceless. Shaded areas denote articulations judged impossible.

VOWELS




Where symbols appear in pairs, the one to the right represents a rounded vowel.

The IPA is of interest to us for a few reasons. First of all, it is of interest because spelling is not standardized. English spelling is by no means completely phonetic, and if you pronounced a French word as you would if it were English, you would not sound like you are speaking French. Different languages have more and less phonetic spelling traditions. The standardization of the IPA helps to mitigate the influence of different spelling traditions. Further, it gives us a well-defined way to talk about the "inventory of sounds" for a given language, as a subset of the IPA. It is worth noting that this means we are choosing a particular dialect of each language that we consider. But since we are studying the phonological structure of an inventory of sounds, this is necessary to fixing that inventory and its relationships in the first place. Anyone interested in learning more about the IPA can refer to [4], and anyone interested in learning more about phonetics can refer to [5].

2.2. **Language Families.** This context is not important to the methodology of what we are doing, but relevant to the sorts of patterns that we might want to look for. Languages families are a part of a taxonomy of languages (which can be thought of much like the taxonomy of animals). This taxonomy is historical, and does not necessarily correspond to phonology. We are interested in this classification because it is worth exploring whether there is any relationship between the phonological structure of a language (which we are looking at) and its language family or subfamily. Like most historical classifications, there is some debate about how these families should be constructed—it is difficult to reconstruct history. Regardless, language families are useful and interesting. A language family is a collection of languages that are hypothesized to descend from the same ancestor. German, English, French, and Hindi, for example, are all Indo European languages, which is to say that they are all hypothesized to descend from Proto Indo European. Finnish and Chinese, on the other hand, are members of different languages familes, namely Uralic and Sino Tibetan, respectively. Indo European, Uralic, and Sino Tibetan are all examples of language families. Within a language family there are more relationships, with subfamilies of languages such that members of the subfamily descend from another more recent ancestor and are therefore more related. This set of relationships is often represented as a tree. This representation is not perfectly precise. A classic example of this is English. English is Germanic (a subfamily of Indo European). But with the invasion of England by the French-speaking William the Conqueror in 1066, English was significantly influenced by French (a Romance language, in other words a member of a different subfamily of Indo European). In this way, the "tree" of Indo European does not perfectly represent all of the historical relationships between languages. And while it is worth knowing this, the simpler tree structure is still interesting, and sufficient for our current interests. Appendix I includes the family tree of Indo European. Anyone interested in more about language families, and historical linguistics more generally, can refer to [6].

## 3. A Mathematization of Phonological Structure

In this section, we will describe the way that we construct a metric on sounds for each language. The ideas in this section come from or are inspired by ideas in [10] and [11]. They are also very similar to the way that a metric on words was constructed for word2vec.

3.1. **Data.** For each language that we consider, we take its Swadesh list written in IPA. All of these come from Wiktionary, [9]. A Swadesh list is a standardized list of 100 or 207 common words (all of ours are 207 words). The set of words in the Swadesh list is the same for all languages. This is a good way to find data because

(1) It is standardized, so we are getting close to the same amount of data for each language
(2) It is possible to find Swadesh lists in IPA

It is a bad way to get data because

(1) It is not very much data for each language
(2) While it is possible to find Swadesh lists in IPA, and it is a standardized form for which we have been able to find more data, we still do not have that many languages, nor that much variety in languages

We have 17 languages:

American English, Greek, Russian, Hmong, Mandarin Chinese, Armenian, Spanish, German, French, Dutch, Icelandic, Polish, Italian, Arabic, Min Dong, Burmese, Min Nan

Most of these are Indo European, but there are a few others: Arabic is Afro-Asiatic, Hmong is Hmong-Mien, and Chinese, Burmese, Min Dong, and Min Nan are Sino Tibetan. There are many more languages in each of these families, and many more language families not represented here.

3.2. **Contexts.** Contexts are the basis on which we will build a metric. We will consider two sounds "similar" in a given language if they are used in similar contexts. I think this is a fairly intuitive idea of similarity: two things are similar if they often appear in the same situations. When we say *contexts*, we mean something very particular:

**Definition 3.1.** Fix a sound $i$ that occurs in the dataset. For an occurrence of $i$ in the dataset, the **context** of $i$ in that occurrence is the ordered pair of sounds $(j, k)$ that come immediately before and after it. If $i$ occurs at the beginning of the word, then the first element of this ordered pair will be a space. Similarly, if $i$ occurs at the end of a word, then the second element of this ordered pair will be a space. We will denote the set of contexts of $i$ over of all occurrences in the dataset by $C(i)$.

**Example 3.2.** In the word "spectacular", the contexts of "a" are {t, c} and {l, r}, while the contexts of "c" are {e, t} and {a, u}.

3.3. **Cosine Similarity.** To see how similar two contexts sets are, we will compute the cosine similarity between them. The standard definition of cosine similarity between vectors $\mathbf{A}$ and $\mathbf{B}$ is

$$\frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}||||\mathbf{B}||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

The context data that we have collected, however, is in terms of sets. In this case the equivalent definition, for sets $A$ and $B$, is

$$\frac{|A \cap B|}{\sqrt{|A||B|}}$$

This cosine similarity for sets is equivalent to the standard cosine similarity for vectors if sets are encoded as bit vectors. Given an underlying set $X$ that contains both $A$ and $B$, we can think of the elements of $X$ as defining a basis for $\mathbb{R}^{|X|}$. Given this, the bit vector for $A$ has a 1 in every coordinate that corresponds to an element of $X$ that is an element of $A$ and a 0 otherwise. Here is an example which shows how to get from one definition to the other:

**Example 3.3.** Let

$$A = \{a, b, b, c, c, d\}, \quad B = \{a, c, c, e, f\} \quad \Longrightarrow \quad A \cap B = \{a, c, c\}$$

Thus according to the set definition, we get

$$\frac{n(A \cap B)}{\sqrt{n(A)}\sqrt{n(B)}} = \frac{3}{\sqrt{6}\sqrt{5}} = \sqrt{\frac{3}{10}}$$

To use the vector definition, we can encode these sets with bit vectors, putting a 0 if the element is not in the set and a 1 if the element is in the set. Take the underlying set $X = \{a, a, b, b, c, c, d, d, e, e, f, f\}$. Then

$$\mathbf{A} = (1,0,1,1,1,1,1,0,0,0,0,0) \quad \mathbf{B} = (1,0,0,0,1,1,0,0,1,0,0,0) \implies \mathbf{A} \cdot \mathbf{B} = 3,$$

and

$$\begin{aligned}\frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}||||\mathbf{B}||} &= \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} = \frac{3}{\sqrt{\sum_{i=1}^{n} A_i^2 \sum_{i=1}^{n} B_i^2}} \\ &= \frac{3}{\sqrt{\sum_{x \in A} 1^2 \sum_{y \in B} 1^2}} = \frac{3}{\sqrt{6 \cdot 5}} = \sqrt{\frac{3}{10}}\end{aligned}$$

Intuitively, cosine similarity is about projection. For vectors, it is the cosine of the angle between those vectors. Thus if they are pointing in the same direction, it is 1, and if they are orthogonal, it is 0. In terms of context sets, we are computing the "projection" of one context set onto the other, and then normalizing for the fact that the context sets are likely not the same size (each sound won't occur the same number of times in the dataset, both because the data isn't perfect and because not all sounds are equally common in the first place). Note also that cosine similarity is a symmetric relation, so

$$\text{cs}(C(i), C(j)) = \frac{|C(i) \cap C(j)|}{\sqrt{|C(i)||C(j)|}} = \text{cs}(C(j), C(i))$$

3.4. **Using Cosine Similarity to Embed Sounds in $\mathbb{R}^n$.** Fix a language $X$, and let $n$ be the number of sounds in $X$. We will describe these sounds in terms of their context sets, and use cosine similarity on their context sets to compare them and look for relationships. We will consider $\mathbb{R}^n$ with "basis" given by the $n$ contexts sets. We will denote the standard unit vector in the $j$ direction by $\vec{e}_j$. Given a sound $i$, we will embed it as a vector $\vec{v}_i$ in $\mathbb{R}^n$ where the $j^{th}$ coordinate is the cosine similarity between $i$ and $j$:

$$\vec{v}_i = \sum_{j=1}^{n} \frac{|C(i) \cap C(j)|}{\sqrt{|C(i)||C(j)|}} \vec{e}_j$$

Note that the cosine similarity between $i$ and itself is 1, so $\vec{v}_i \cdot \vec{e}_i = 1$. In each other coordinate, we are effectively asking "how $j$ is $i$, with respect to contexts?" The cosine similarity answers this question by saying what the projection of $C(i)$ onto $C(j)$ is, and then we place $\vec{v}_i$ so that the projection of $\vec{v}_i$ onto $\vec{e}_j$ is that. If $i$ and $j$ share no contexts, then the $j^{th}$ coordinate of $i$ will be 0. If $i$ and $j$ share a large fraction of their contexts, then the $j^{th}$ coordinate of $i$ (and similarly the $i^{th}$ coordinate of $j$) will be close to 1.

So, what is this? Basically, we are quantifying sounds in terms of their contexts. Consider some extreme cases. If contexts tell you nothing about a sound—i.e. every sound has the same multiplicity of every context—then the whole set will be embedded at the single point $(1, ..., 1)$. If contexts tell you everything about a sound—i.e. all the context sets are disjoint—then the sounds will be embedded as the standard unit vectors. If there are two groups of sounds $A$ and $B$ such that the context sets of the sounds within a group are identical to each other and sounds in different groups share no contexts, then the whole set will be embedded at exactly two points.

A useful way to represent these coordinates is as a coordinate matrix, where the $i^{th}$ row of the matrix is the coordinate for the sound $i$. This matrix has the following properties:

(1) It is symmetric
(2) It has 1s on the diagonal and values between 0 and 1 off the diagonal
(3) It is positive semi-definite (this follows from (1) and (2))

3.5. **Example: English.** We constructed this metric for all 17 languages listed in section 3.1. As an example, the steps of constructing it for (Standard American) English are including in Appendix IV. For each language, we have used Mathematica to find the context sets and construct the coordinate matrix. To include the code, this is Appendix IV.

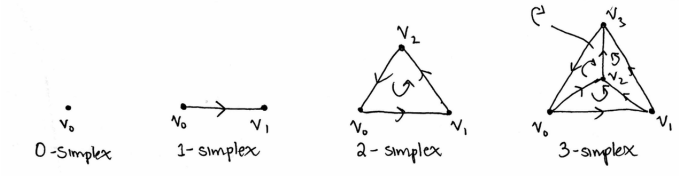
3.6. **Decisions and Judgments.** There are a number of decisions that went into this construction, which may or may not have been the ideal decisions. We note them here.

(1) Using IPA Swadesh lists as datasets. This is arguably just not enough context data. This decision was mostly made because Swadesh lists were something that could be found in IPA. But there are some sounds that, for example, only appear once in the entire dataset. Because of this, the results that we find may change if these methods were repeated on larger datasets. This is a significant concern and will come up again later.
(2) Many languages have diacritics, accents, and other extra markers in IPA. In most cases, we have chosen to remove these.
(3) Some of these languages are tonal. The IPA has symbols for tones, and we have decided to keep these.
(4) The decision to use contexts, as defined, to characterize sounds. There are many other ways we could describe them, and looking at contexts was a decision. We could have described sounds based on their features in the IPA, by how they are represented in spelled forms of the language, based on more context information (for example, consider the two sounds before or two sounds after), or many other things.
(5) The decision to use cosine similarity to construct the metric. Even still using contexts, there are other ways we could have done this.
(6) In our construction of context sets, we chose to keep multiplicity. Thus if a sound appears in the same context twice, then this corresponds to two elements of its context set. If $i, j$ occur in context $X$ regularly and $k$ occurs in it only very rarely, it seems more representative to say that $i$ and $j$ are more similar to each other than they are to $k$. This is why we have chosen to keep multiplicity information. On the other hand, when we think of context sets as bit vectors, keeping multiplicity means that the dimension of the space those bit vectors are in depends on the dataset. This seems like a strange decision mathematically. We defend it by arguing that we are not really thinking about the cosine similarity between contexts in this space. We are only using it as a similarity measure between sets, and in that context this problem does not appear.

## 4. Simplicial Homology

With the set up of the previous sections in place, we need to take a step back and talk about the mathematical tools we will use. Homology groups are topological invariants. Unlike most topological invariants, they are actually fairly easy to compute. There is a homology group for each dimension, and approximately the $k^{th}$ homology group $H_k$ is an abelian group structure on $k$ dimensional holes. Persistent homology only requires simplicial homology, and so that is all that we will discuss. The main purpose of the paper is to use these things, and thus this section will be very informal. Anyone interested in more formal mathematical details can refer to [3]. The reader should also know that there are other equivalent renditions of homology, and that the scope of homology goes far beyond simplicial complexes.

An $n$-simplex is an $n$ dimensional triangle:




Notationally, we denote an $n$ simplex with vertices $v_0, ..., v_n$ by $[v_0, ..., v_n]$. A **simplicial complex** is a collection of simplices that are completely determined by their vertices. This means that there are no double faces or double edges, and that any face of a simplex in the complex is also in the complex. Every edge or face has an orientation. It does not matter what the orientations are, only that we are consistent with them once we choose them. The **boundary** $\partial$ of a simplex is the alternating sum of its faces.

Homology groups are topological invariants of a space based on its holes of different dimensions. Given a simplicial complex $S$, let $C_k$ be the free abelian group with its $k$-simplices as its basis. In other words, elements of this group are linear combinations of these $k$ simplices. The boundary operator $\partial$ then gives us a chain of maps

$$C_{k+1} \xrightarrow{\partial_{k+1}} C_k \xrightarrow{\partial_k} C_{k-1} \to ... \to C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} 0$$

Since the boundary is the alternating sum of faces, taking the boundary of a boundary causes everything to cancel, and so $\partial_{k+1}\partial_k = 0$ always. These boundary maps allow us to detect holes, and we will use them to construct the homology groups.

As a motivational example, consider the circle $S^1$. This is a 1-manifold and the boundary of the 2-dimensional disk $D^2$. If both $S^1$ and $D^2$ are part of our space, then there is no hole. But if $S^1$ is a part of the space and $D^2$ is not, then we do have a hole. A boundary might be the boundary of something in the space, in which case it does not mark a hole, or it might be the boundary of something that is not in the space, in which case it does mark a hole. We will call things that have the potential to be boundaries **cycles**, and now reserve the term **boundary** for things that actually are boundaries within our space. We are trying to find holes, so we want to find the cycles that are *not* boundaries. Since $\partial_{k+1}\partial_k = 0$ always, the set of cycles (things that have the potential to be boundaries) is $\ker \partial_k$. The set of things that actually are boundaries is simply $\text{im}\partial_{k+1}$. Thus we define the $k^{th}$

homology group to be

$$H_k = \ker \partial_k / \text{im} \partial_{k+1}$$

A few important things to know about homology:

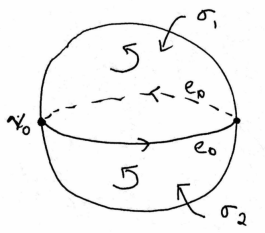
(1) If the highest dimension simplex in a simplicial complex is a $k$ simplex, then $H_n = 0$ for all $n > k$, since $C_n = 0$.
(2) $H_0(X) = \mathbb{Z}^n$, where $n$ is the number of path components in $X$. The best way to think about this is that a path is a sequence of edges. The image of $\partial_0$ is 0, so $C_0 = \ker \partial_0$ always. The boundary of an edge is $v - w$ for two vertices $v, w$. The set of differences between vertices has 1 fewer linear independently elements than the set of vertices, so if there is a path of edges between any pair of vertices (and thus there is one path component), then $H_0 \cong \mathbb{Z}$. If there exists a $u$ such that there is no path from $v$, then the image of $\partial_1$ loses a generator, and $H_0 \cong \mathbb{Z}^2$.
(3) $H_1(X) = \pi_1(X)/[\pi_1(X), \pi_1(X)]$, i.e. $H_1$ is the abelianization of $\pi_1$. This is not true for higher homology and homotopy groups, since higher homotopy groups are already abelian, and not the same as homology groups.
(4) Homology is homotopy invariant, i.e. if $X$ is homotopic to $Y$, then they have isomorphic homology groups. This implies that homology is a topological invariant. The converse of this statement, however, is *not* true.
(5) Simplicial homology is independent of simplicial decomposition; it does not matter how you choose to do the simplicial decomposition.
(6) Since homology groups are abelian, the structure theorem for abelian groups says if they are finitely generated, then they are of the form:

$$\mathbb{Z}^\beta \times \prod_{i=1}^{m} \mathbb{Z}/p_i^{k_i}\mathbb{Z}$$

The number of copies of $\mathbb{Z}$ in $H_n$ is called the $n^{th}$ **Betti number** $\beta_n$.
(7) We can also consider homology groups written with coefficients from rings other than $\mathbb{Z}$. For example, if we use coefficients from $\mathbb{F}_2$, then the structure theorem becomes just $\mathbb{F}_2^k$, where $k$ is the number of $k$ dimensional loops, without regard for whether they are free or torsion. When we compute persistent homology later, we will use $\mathbb{F}_2$ coefficients.

Before we finish this section, we would like to give a few examples. Some of these examples do not adhere perfectly simplicial complex construction, but we think they can still be understood, and are useful for developing intuition.

**Example 4.1.** Consider $S^2$ as the upper and lower hemisphere glued together on the equator (this is not a simplicial complex, but is a useful example).



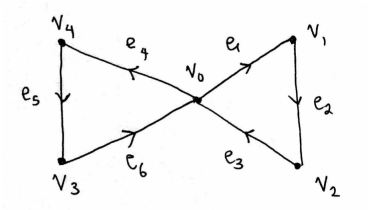
Then we have

$$C_2 = \{\sigma_1, \sigma_2\},\ C_1 = \{e_0\},\ C_0 = \{v_0\}$$

Next, we have that $\partial\sigma_1 = \partial\sigma_2 = e_0$, and so $\ker\partial_2 = \partial(\sigma_1 - \sigma_2) = 0$, and that $\partial e_0 = v_0 - v_0 = 0$. Therefore we have the following homology groups:

$$\begin{aligned} H_0(S^2) &= \langle v_0\rangle/\langle 0\rangle \cong \mathbb{Z} \\ H_1(S^2) &= \langle e_0\rangle/\langle e_0\rangle \cong 0 \\ H_2(S^2) &= \langle \sigma_1 - \sigma_2\rangle/\langle 0\rangle \cong \mathbb{Z} \\ H_k(S^2) &= 0 \ \text{ for } k \geq 3. \end{aligned}$$

**Example 4.2.** Consider the one point union of two cycles $S^1 \vee S^1$. We can represent this as the following simplicial complex:



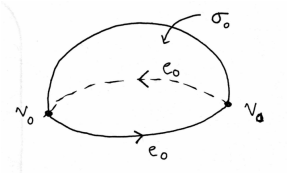

Here we have that

$$C_0 = \langle v_0, ..., v_4\rangle,\ C_1 = \langle e_1, ..., e_6\rangle,\ C_2 = 0$$

Thus the only nonzero homology groups are $H_0$ and $H_1$:

$$\begin{aligned} H_0 &= C_0/\text{im}\partial_1 = C_0/\langle v_0 - v_1, v_0 - v_2, v_3 - v_4, v_3 - v_5\rangle \cong \mathbb{Z} \\ H_1 &= \ker\partial_1/\text{im}\partial_2 = \ker\partial_1 = \langle e_1 + e_2 + e_3, e_4 + e_5 + e_6\rangle \cong \mathbb{Z}^2 \end{aligned}$$

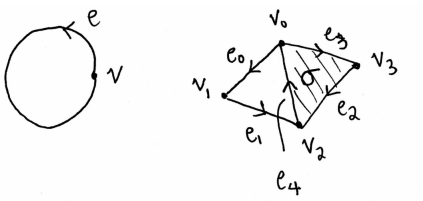
**Example 4.3.** Consider the real projective plane $\mathbb{R}P^2$ as $S^2/\sim$ where $\sim$ identifies antipodal points $x \sim -x$. Since the interior of the lower hemisphere is completely identified to the upper hemisphere, we can draw this as a hemisphere with identified boundary:




Here we have that $C_0 = \langle v_0\rangle$, $C_1 = \langle e_0\rangle$, and $C_2 = \langle\sigma_0\rangle$ (given the identified boundary). Then $\partial\sigma_0 = 2e_0$, so $\ker\partial_2 = 0$. On the other hand, $\partial e_0 = v_0 - v_0 = 0$. Thus we have

$$\begin{aligned} H_0 &= \langle v_0\rangle/0 \cong \mathbb{Z} \\ H_1 &= \langle e_0\rangle/\langle 2e_0\rangle \cong \mathbb{Z}/2\mathbb{Z} \\ H_2 &= \langle 0\rangle = 0 \end{aligned}$$

**Example 4.4.** Consider the circle $S^1$ and this simplicial complex structure:




This simplicial complex structure is homotopy equivalent to $S^1$, and we can check that they have do in fact have the same homology groups. For $S^1$, we have

$$H_0(S^1) = \langle v_0 \rangle / 0 \cong \mathbb{Z}$$
$$H_1(S^1) = \langle e_0 \rangle / 0 \cong \mathbb{Z}$$

For the simplicial complex $C$, we have that

$$H_0(C) = \langle v_0, v_1, v_2, v_3 \rangle / \langle v_0 - v_1, v_0 - v_2, v_0 - v_3 \rangle \cong \mathbb{Z}$$
$$H_1(C) = \langle e_0 + e_1 + e_4, e_4 + e_2 + e_3 \rangle / \langle e_4 + e_2 + e_3 \rangle \cong \mathbb{Z}$$

## 5. Persistent Homology

Persistent homology is a method from topological data analysis. It is often very hard to know what's going on with data, particularly when you have a large amount of data in a large number of dimensions. Methods from statistics (like regression) have often been applied to make sense of data. But when it is not clear what the overall structure of the data is, statistics are not particularly useful—there is nothing meaningful in fitting a circle with a line. Still, it is hard to tell what aspects of a point cloud of data reflect its overall structure, and what aspects are only noise. It is this goal of looking for "overall structure" that makes topology a worthwhile thing to consider. Persistent homology aims to represent the overall structure of a point cloud by putting simplicial complex structures on it and computing their homology. It has been shown that persistent homology is fairly stable, i.e. that adding noise to the data does not affect the persistent homology. This stability is evidence that persistent homology is representative of the overall structure that we seek to find.

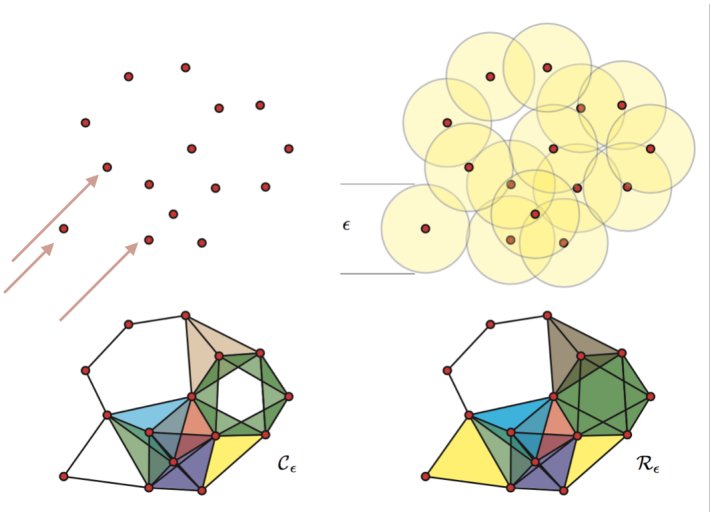
It is worth remarking that persistent homology is *a* way to interpret the arrangement of a point cloud as topological structure. It is not that a point cloud necessarily has the structure given to it by persistent homology—this is a claim that would have to be proven. But if persistent homology is a useful method, it is because the structure that persistent homology constructs on this point cloud usefully reflects the hidden overall structure of the point cloud. Persistent homology is not the only way to do this interpretation, and in the spirit of noting all the decisions that could have been made differently, we could have chosen a different method.

There are some computational details about how persistent homology is actually done so as to minimize memory usage. We will not talk about these aspects, but refer the reader to the references. There are also some other constructions for putting the simplicial complex structure on data. We will describe only the one that we have used. If the reader is interested in more details, we suggest [2] and [1].

5.1. **Putting a Simplicial Complex Structure on a Point Cloud.** Persistent homology requires points in a metric space. Given $\epsilon > 0$, it adds simplices based on the intersections between balls of radius $\epsilon/2$. There are two versions of this construction:

(1) **Cech Complex.** If the closed $\epsilon/2$ balls around a set of $k+1$ points in the dataset have a point of common intersection, then those points are the vertices of a $k$ simplex.
(2) **Rips Complex.** If $k+1$ points in the dataset are all pairwise distance less than $\epsilon$ apart, then those points are the vertices of a $k$ simplex.

Overall, the Cech complex builds a $k$ simplex based on the intersection of $k+1$ balls, and the Rips complex builds a $k$ simplex based on $k$ intersections of 2 balls. The Rips complex construction builds more simplices at a smaller $\epsilon$. Here is a example, taken from [2]. The left simplicial complex is the Cech complex and the right is the Rips complex.

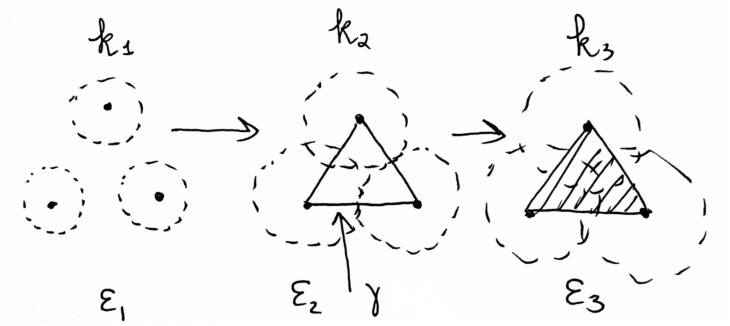



It is clear that the Rips complex and the Cech complex are different, and have different homology groups. In particular, note the three points to which we have put arrows, as they are a good illustration of the difference. Balls around these points do not all intersect at the same time, but they do pairwise intersect.

There are two questions that it is very natural to ask based on this construction so far. First, which $\epsilon$ should we be looking at? And second, should we be using the Cech construction or the Rips construction, and why? Both of these questions are extremely reasonable questions to ask. But the first question is in fact somewhat out of place, and the key idea of persistent homology is to avoid answering this question. To answer the question "which $\epsilon$?" realistically requires us to know what we are trying to find. We cannot experimentally determine the "right" $\epsilon$, because we do not know what structure the point cloud has (we are trying to find it). And there is no "right" $\epsilon$ for all datasets, since scale can vary. Thus instead of choosing an $\epsilon$, we want to *vary* $\epsilon$, and see which features of the simplicial complex structures that we have built *persist* for many $\epsilon$. This is the key idea. In fact, as we will explain in the next section, varying $\epsilon$ makes the two constructions described above equivalent.

5.2. **Persistence.** To understand a bit more about what persistent homology is telling us, we have to understand how the simplicial complex we are building evolves as $\epsilon$ gets larger. Overall, we start at $\epsilon = 0$ with a set of disjoint points, and eventually expand to a single component and then a single contractible component as $\epsilon$ increases. Let $\mathcal{K}_\epsilon$ denote the simplicial complex structure we construct at $\epsilon$. The important thing about persistence is that when $\alpha < \beta$, if a simplex $\sigma \in \mathcal{K}_\alpha$, then $\sigma \in \mathcal{K}_\beta$. Thus there is a natural inclusion from $\mathcal{K}_\alpha \hookrightarrow \mathcal{K}_\beta$ for all $\alpha < \beta$. These inclusions induce homomorphisms between the homology groups of $\mathcal{K}_\alpha$ and $\mathcal{K}_\beta$.

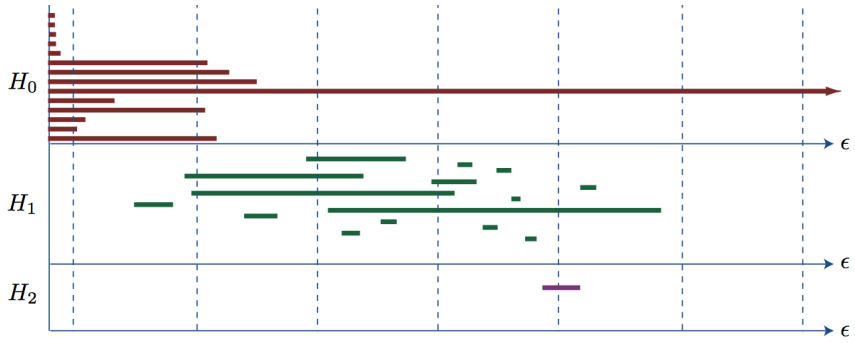
We say that a generator $\gamma$ of a homology group $H_k(\mathcal{K}_\epsilon)$ is **born** at the first $\epsilon$ it appears at, and **dies** when its cycle is filled in.




In the above picture, $\gamma$ is born at $\epsilon_2$ and dies at $\epsilon_3$. With this, we can quickly and easily show that the Cech and Rips constructions are equivalent in persistence, since we have the inclusions

$$\mathcal{R}_\epsilon \hookrightarrow \mathcal{C}_{\sqrt{2}\epsilon} \hookrightarrow \mathcal{R}_{\sqrt{2}\epsilon}.$$

5.3. **Barcode Graphs.** Visually, we can represent persistence with a **barcode graph**. The horizontal axis is $\epsilon$, and above this we include horizontal lines for each feature. A line begins at the $\epsilon$ where the feature it represents is born and persists until the $\epsilon$ where that feature dies. Here is an example barcode graph, also from [2]:

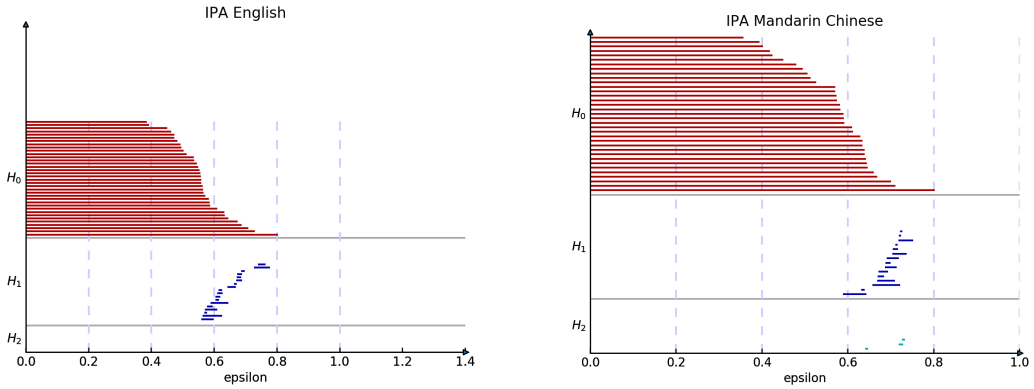



As we can see, there are a few $H_1$ features that persist for a bit and seem like they might be significant. The $H_2$ feature is very likely not significant. We often compute the homology groups at a given $\epsilon$ with finite field coefficents (this sort of diagram does not distinguish between free and torsion generators). Analysis of these barcode graphs is ultimately fairly heuristic. But this is, in fact, consistent with their goal.

We are trying to turn precise quantitative data into a more general qualitative picture, so it makes some sense that this is not an entirely precise process.

## 6. Persistent Homology on Phonological Data

We used the Ripser program [8] to compute persistent homology (available on GitHub). It computes persistent homology using the Rips complex construction described in the previous section. By default, it computes the homology groups with $\mathbb{F}_2$ coefficients, and this is what we have done. Here we include two example barcode graphs, for English and Mandarin Chinese. To fit in the default plotrange of the Ripser program, we divided all distances by 2, so note that if you are interested in precise $\epsilon$ values, you should multiply by 2 (in particular remember this if you look at the barcodes and the graph from the English example in Appendix IV, since the radius 1.5 there corresponds to $\epsilon = 1.5$, which is then 0.75 on the $x$ axis of the barcode graphs). We include the barcode graphs for all 17 languages in Appendix II. We recommend that the reader look at all of these before proceeding to section 7, but include them in an appendix so as not to clutter things here.




## 7. Understanding Results

The hardest thing about something like this is understanding what results actually mean. The barcode graphs that we have computed from languages are hard to parse. They are objects whose features can have (1) mathematical meaning and (2) linguistic meaning. The goal is to find a relationship between these things; to find something mathematical that represents something linguistic. This is not easy. Ultimately, we believe that the amount of data we have for each language is insufficient to come to a conclusion. The cosine similarity matrices that we constructed are fairly sparse, meaning that many sounds do not share any contexts. If this were true for a much larger dataset, it would be very interesting. But since our datasets (Swadesh lists) are small, we do not know whether this sparseness reflects linguistic information or just the size of the dataset. If these matrices become less sparse with more data, that would completely change our results. In this section, we will detail our attempts to analyze these barcodes. These methods would still be useful with more data. The lack of data prevents us from coming to any strong conclusions,

but it does not prevent us from starting to develop methodologies. The purpose of this section is to explain methodologies.

7.1. **First Questions: What Persists and How does it Persist?** The first two mathematical questions that we should ask are:

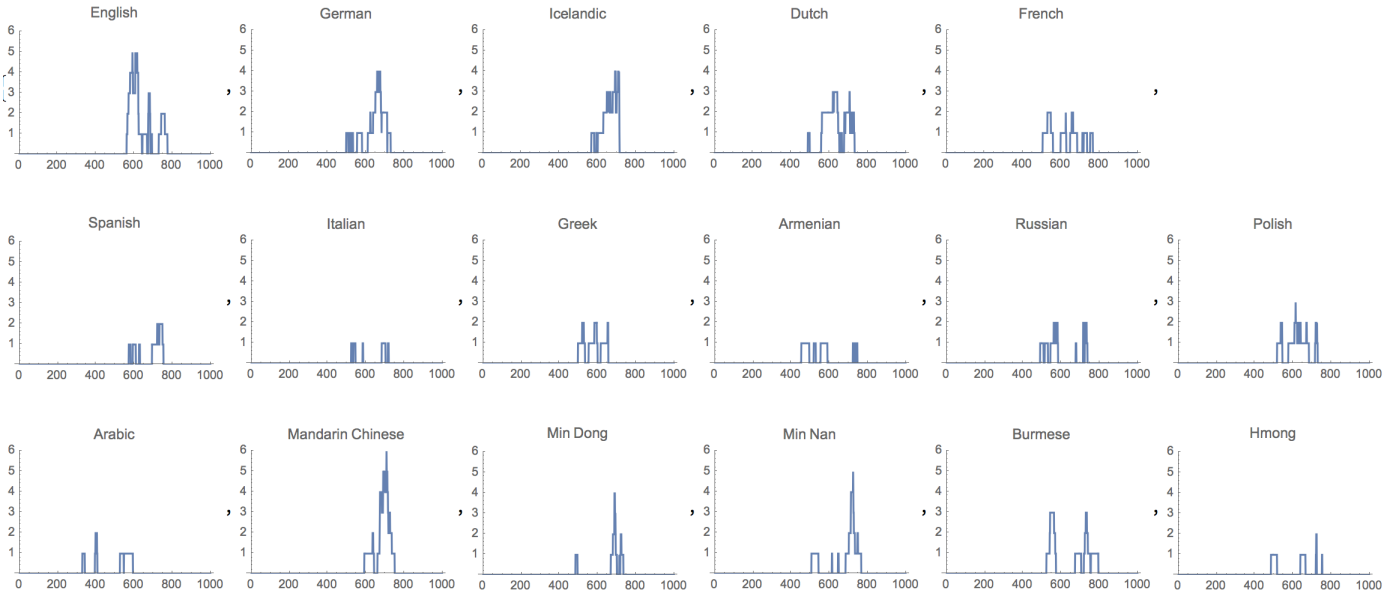
(1) What persists?
(2) How do things persist?

For $H_0$, in any situation we start with $n$ components (where $n$ is the number of sounds) and eventually end with 1 component. What persists, then, is the number of components that stays constant for a large number of $\epsilon$. In all of our languages, none of the $H_1$ and $H_2$ features persist for long. There are very few $H_2$ features in any language, and these are likely noise. No $H_1$ feature persists for long enough to say that it is a significant feature, but there are too many small $H_1$ features to ignore them entirely. It is worth looking at how the $H_1$ features persist. Note however that since nothing persists in the long run, we do not have enough information to determine the topology of any of our languages from their barcodes.

The next two subsections are beginnings of ways to analyze the second question: how do things persist? This requires more investigation. The main thing we will do is develop another method of visualization: plotting the evolution of betti numbers $\beta_1$ and $\beta_0$ as a function of $\epsilon$. The first subsection does this for $H_1$ and the second for $H_0$.

7.2. $H_1$ **Betti Numbers.** Heuristically, it looks like there could be patterns in the staircase-like tower of these $H_1$ barcodes. It is tempting to look for a way to quantify this, and we tried. But it is difficult to construct a quantification that truly represents the barcode graphs. In this subsection, we will describe an alternative visualization (alternative to barcode graphs) in terms of betti numbers, an unsuccessful attempt to quantify this, why this attempt was unsuccessful, and characteristics of this visualization that could be useful in further inquiry.

From the barcode graph, we can read $\beta_1$ at any $\epsilon$. Plotting the evolution of $\beta_1$ as a function of $\epsilon$ is another useful visualization. To do this, at each $\epsilon$, draw a vertical line and count the number of $H_1$ features that the line intersects. Computationally, we have to choose a "resolution", or how many $\epsilon$ we want to check. We tried both 100 and 1000. In the 1000 case (which we will present here), we sampled 1000 equally spaced $\epsilon$ every 0.0001. Here are the plots for all 17 of these languages:



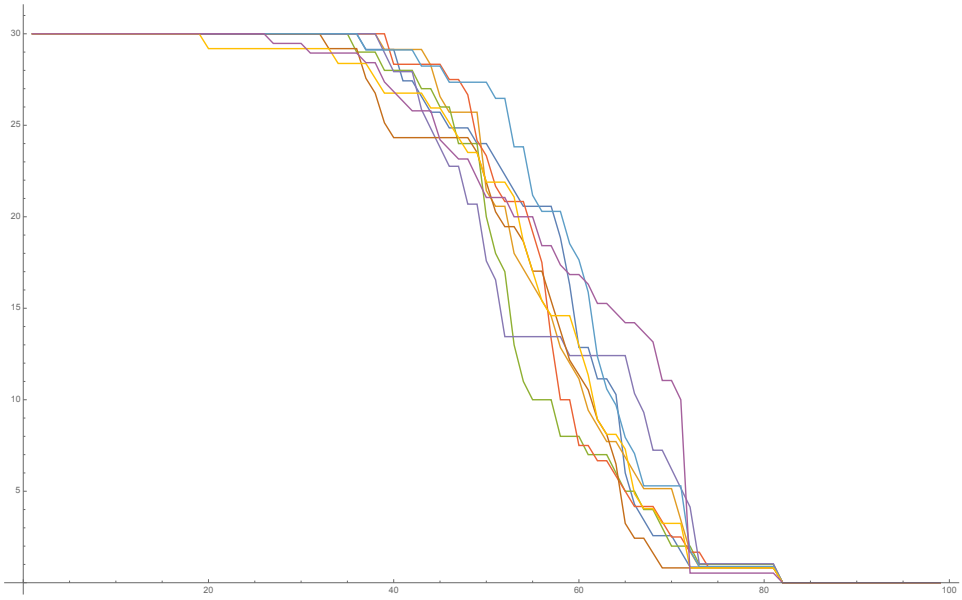

These are a useful representation of the barcode graph which preserves the $H_1$ information. There seem to be a few different types of things going on here. Some of these have tall peaks (i.e. higher maximum betti number), some of them multiple peaks, and some of them have short peaks. The Germanic languages seem to have the highest peaks of any Indo European language, but beyond that looking at peak height does not seem to correspond to language family or subfamily. The Sino Tibetan language also seem to mostly have tall peaks, and mostly two peaks. These patterns are all post-hoc patterns: we see them in part because we are looking for them. If we weren't looking for anything, we would group Chinese and Icelandic, Hmong and Arabic, or Greek and Russian. With more data patterns might emerge, but as of now it is unclear.

It would be useful to quantify this, and have some more precise notion of "similar" betti number plots. We initially tried to do this by computing an integral difference between betti numbers: for langauges $A$ and $B$, at each $\epsilon$ take $|\beta_{A,1,\epsilon} - \beta_{B,1,\epsilon}|$, where $\beta_{A,1,\epsilon}$ is the $H_1$ betti number of $A$ at $\epsilon$. Then take a sum of these values over the 1000 sampled $\epsilon$. But this method has two significant failings. First, it is likely to say that languages with few features are similar regardless of their arrangements simply because the numbers that we are subtracting are smaller. Second, this idea is taking information that persistent homology isn't claiming to provide. Part of the idea of persistent homology is to *avoid* picking an $\epsilon$. The meaningful thing, topologically, is probably not the $\epsilon$ at which a feature occurs. But if we take this integral difference, we are putting significant emphasis on the $\epsilon$ at which a feature occurs.

A better way to look at these is likely something like curve fitting: how well can we fit one curve to another? We have not tried to do this, but suggest it for further inquiry (with more data). This is probably a good method, particularly because it will pick up on things like peak height (maximum betti number), number of peaks (times that the betti number approaches maximum), peak slopes

(how quickly the betti number changes), peak spacing, and other features that are relevant information that persistent homology is trying to look for.

7.3. $H_0$ **Normalized Betti Numbers.** We can do the same betti number analysis of $H_0$ persistence. Since $H_0$ is the number of components, initially $\beta_0$ will always be the number of sounds in the language. As such, we will look at $\beta_0$ normalized so that all languages start with 30 sounds. Since $\beta_{0,\epsilon}$ monotonically decreases as $\epsilon$ increases, we do not have to worry about when a feature is born as we did with $H_1$. Since we are normalizing the number of components, we also don't have to worry about different languages having different numbers of features: in this normalized form, they all have 30 features. Here we sampled only 100 $\epsilon$. Here is the plot of normalized $\beta_0$ as a function of $\epsilon$. Each color is a different language:




It's hard to say whether these are "similar" or not overall, because we don't know the overall situation—we don't know the range of things that could have happened. The most promising direction of analysis here (in normalized form) is likely also curve fitting.

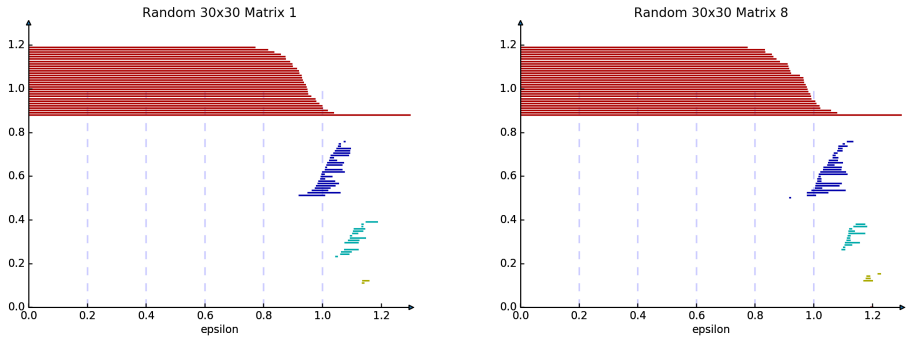
7.4. **Barcode Graphs and Matrices.** At least given the data that we have, we do not see an obvious correspondence between persistent homology results and language families in our investigation of $\beta_1$ or $\beta_0$ evolution. At this point, however, it is useful for us to understand the data that we have, and the mathematical circumstances that it comes from. We cannot really tell whether the persistent homology results we have for languages are very different or very similar from each other without taking into account what they could have been. Mathematically, all that we put in was the form of the coordinate matrix. We did not require anything else. As such, to get a sense of the circumstances, it is worth exploring the persistent homology of matrices of the same form as our coordinate matrices. The question of what the persistent homology of these matrices looks like is mathematically interesting in its own right, but our investigation of it here will be purely experimental.

We also want to know how stable our data is. We certainly do not have enough languages. Do we have enough context data in each language? Ultimately, no. Evidence for this comes from the fact the coordinate matrices for all of our languages are actually very sparse. It is possible that these 0s are reflecting something meaningful about phonological structure, but it is equally possible that they are simply a consequence of the size of the dataset. If the coordinate matrix changes with more data, then the persistent homology changes, and our results may look very different. Looking back at matrices shows us this sparseness, and emphasizes the need for more data before we can do meaningful analysis with persistent homology.

As a reminder, the requirements on the coordinate matrix are that

(1) It is symmetric
(2) It has 1s on the diagonal and values between 0 and 1 off the diagonal
(3) It is positive semi-definite (this follows from (1) and (2))

So, what is the persistent homology of a "random" such matrix? We construct "random" matrices by randomly generated numbers using the Mathematica function RandomReal [7], which generates uniformly distributed random real numbers between 0 and 1. This is not rigorous, and the study of random matrices deserves much more care and attention than we have given it. However, we still think it will be a useful way to gather information. We generated 8 such "random" $30 \times 30$ matrices. All of them had higher persistent homology than the languages did: five of them had features up to $H_3$, one up to $H_2$ (but many more features than any language did), one up to $H_4$, and one up to $H_5$. Here are two examples of these barcode graphs. The rest are in Appendix III.

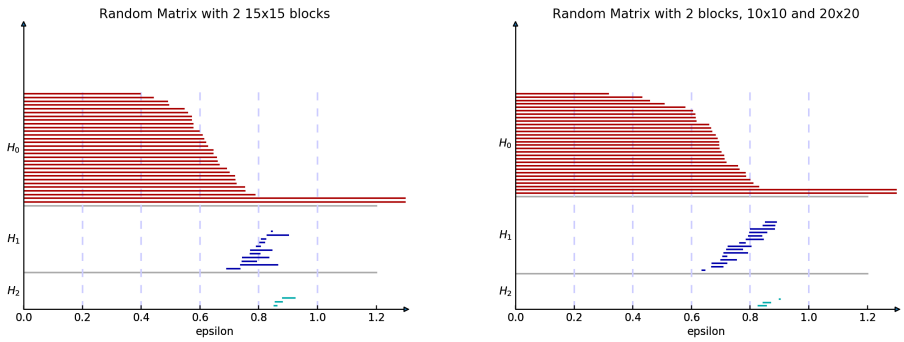



This at least suggests that the persistent homology of a language is detecting more structure than we are putting in: putting in a random coordinate matrix is not enough to make a barcode graph that looks like the ones we got for languages. Linguistically, it is not surprising that language is not represented by a random matrix like these. If it were, then context would be randomly correlated with sound. But if that were the case, then if I randomly bang my keyboard, I should be fairly likely to type a word. ksdflakjsalskef does not look like a word in any language.

In light of this, we want to try to figure out what additional structure the persistent homology is detecting, at least in terms of matrices. This is worth doing first because additional structure—if the matrices are representative of the phonology—is very interesting, and exactly what we are trying to study. Further, trying to see

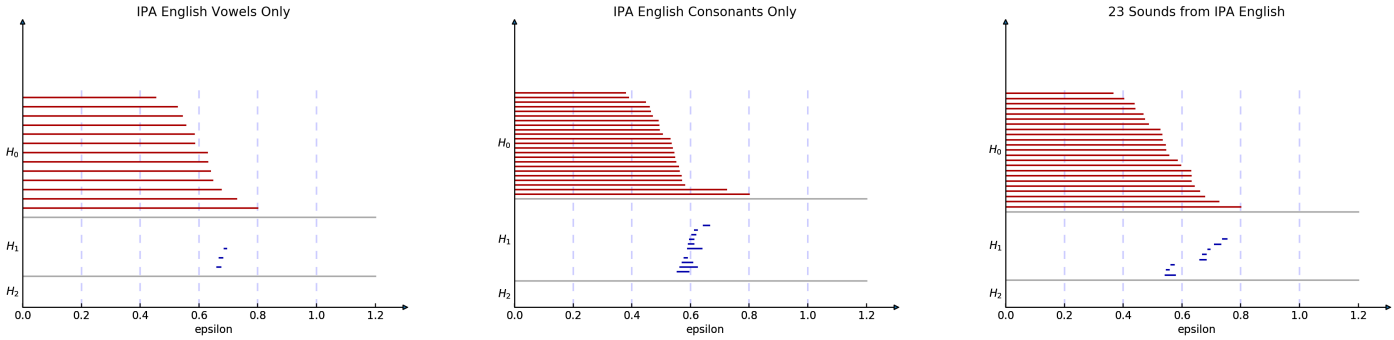
what this additional structure the persistent homology is detecting is a good way to check the data, and see whether this structure is a linguistic feature or a data problem.

What sort of form would we guess that languages have? We know that there are consonants and vowels, and that these are not completely interchangeable. Given that, the first guess worth making is to guess that there are two different types of sounds. For simplicity, suppose that they are completely disjoint with respect to context, and that there are the same number of each type. In terms of matrices, this means that we want to consider a $30 \times 30$ matrix with two similarly sized diagonal blocks. These look a lot more like languages. Here are two examples (the rest are in Appendix III):

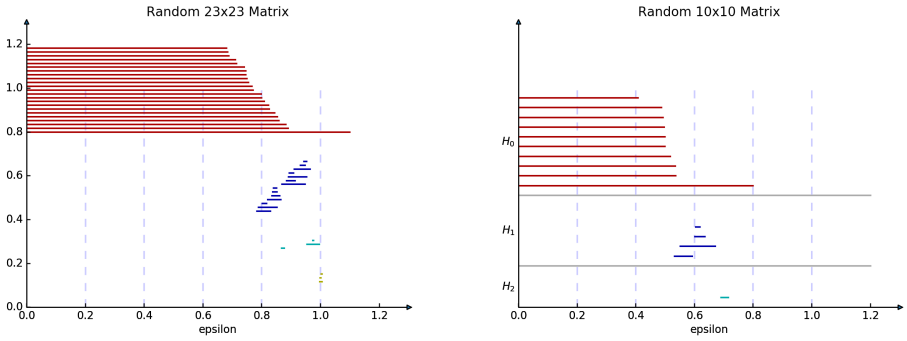



We can clearly see the 2-group nature of these in the fact that 2 components (aka $H_0$ features) survive much longer than the rest. This does not happen with any of our languages. It is possible, however, that the fact that the two types of sounds (consonants and vowels) are not completely disjoint with respect to context would dampen this effect. Thus it is worth testing this hypothesis: what do the coordinate matrix and persistent homology of only consonants look like? Similarly, what about only vowels?

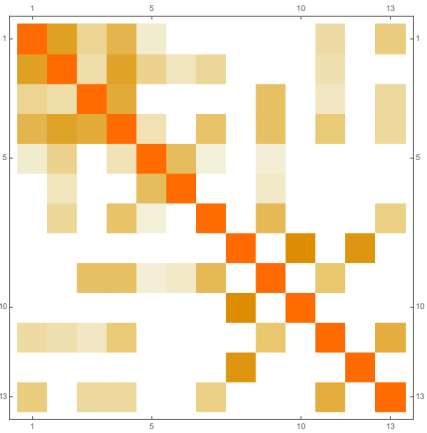
We tested this only in English, by looking at the context data for only consonants, only vowels, and for a subset of 23 sounds that contains both consonants and vowels. There are 23 consonants and 13 vowels in English. If persistent homology is detecting just that there are two categories of sounds, then the consonants alone and the vowels alone should have persistent homology that looks more like a random matrix of that size, while the subset of 23 sounds should perhaps look more like a language. This does not happen:



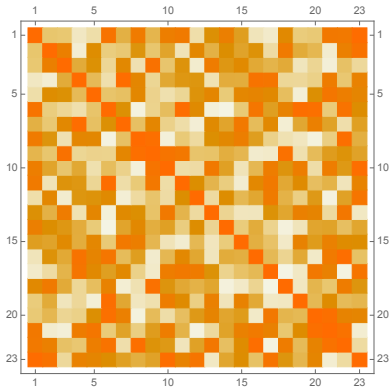

But random matrices of these dimensions still have higher homology:




The barcode for consonants looks more similar to the overall barcode for English than the others do, but none of them look like the random case. Instead, all of them have fewer features than the "random" case for their dimensions, which does not support the hypothesis that the persistent homology is detecting a divide between consonants and vowels. Since the barcode for only English consonants looks similar to the barcode for all English, this suggests that the persistent homology results for English mostly reflect consonant structure and relationships, and not relationships between vowels or between vowels and consonants. But part of this, it seems, is actually because the vowel part of the coordinate matrix is actually very sparse. We can see this in the matrix plot for English vowels:




This is a sign that we have insufficient context data. We do not know whether these 0s are meaningful or a consequence of insufficient data. Furthermore, the existence of these 0s has a significant influence on the resultant persistent homology. The matrix plot for a random matrix as constructed is much denser, and most likely had no 0s:




It is therefore likely that the "additional structure" of our languages that the persistent homology is detecting—beyond the requirements of the matrix—is in part simply how sparse the matrices are. If this is the case, then our persistent homology results may not be representative of the phonological structure that they seek to represent, since we do not know whether our coordinate matrices would be sparse if we constructed them with more data. In all likelihood they would at least be less sparse, but we do not know the extent. Ultimately, we need more context data to say whether the persistent homology and results that we have reflect the phonological structure of the language or simply the size of the dataset.

## 8. Next Directions

This paper is a preliminary study. The most important thing that any future attempt should include, however, is **much more context data for each language.** Without that, any future attempt will run into the same roadblocks as this one. We cannot stress this enough. It also goes without saying that studying more than 17 languages would also make this more interesting. With those things as a given, here are some other things worth exploring:

(1) In the previous section we mentioned curve fitting as a measure of similarity between betti number plots. Is this a reasonable methodology? What does it produce? How do its results change with more data? How does a betti number plot for a language derived from its Swadesh list compare to its betti number plot derived from more data?
(2) How does the coordinate matrix, and its persistent homology, change with more context data? When does it become stable (if it becomes stable)?
(3) All of the $H_1$ (and $H_2$) features that we see in the persistent homology are short-lived. Assuming this continues with more data, it might be worthwhile to study the geometry of these languages instead of the topology. Each one is made up of only about thirty points. Topology might be too coarse, and it would be interesting to consider the less coarse geometry. It may be that these short lived topological features represent more interesting geometric features. (This also brings up a question of the relationship between persistent homology and geometry in general, which is itself interesting.)
(4) How much variety is there in the persistent homology of languages? Are there only a few different "types" (or a single type) of phonological structure as represented by persistent homology, or is there no obvious way to classify them? If there are types, do these types correspond to any other linguistic patterns, either with respect to language families or otherwise?
(5) How does persistent homology reflect different features of the inventory of sounds in a language? For example, is it recognizable when a language has very few vowels?
(6) Is there any relationship between persistent homology of a language and how it sounds? (For example, people often say that German is a "coarse" language and French is a fluid one. Is this something we can see from a barcode?)
(7) There are many moving parts in this project, and many things that could have been done differently. How would the results be the same or different if we changed some of the decisions mentioned in section 3.5?
(8) We have considered languages transcribed in IPA. What about languages with standard spelling? Or different dialects of the same language? Is the variation in phonological structure between dialects more, less, or similar to the variation between different languages? How does the spelled form of a language compare with its IPA form? We started looking into this last question, but did not get far enough with it to discuss it here.
(9) Regardless of the linguistics problem, we also have the question of the persistent homology of these coordinate matrices. What is the persistent homology of a random matrix that has these characteristics (or other ones)? Is there an interesting relationship between matrices and their persistent homology?

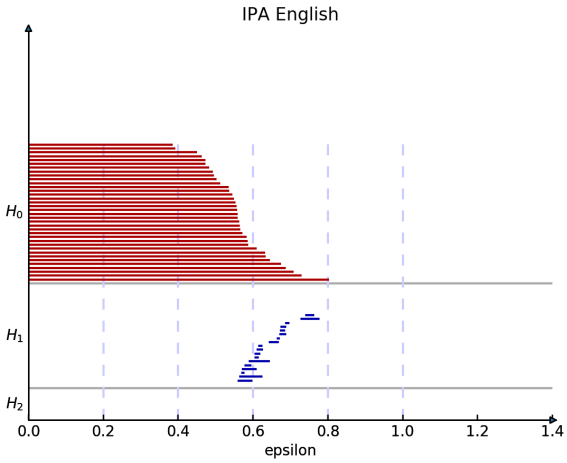
**Acknowledgments.** It is my pleasure to thank my mentor, Weinan Lin, for all his help with both math and programming. I would also like to thank John Goldsmith for some very helpful conversations and resources. Finally, I am happy to thank Peter May for organizing the REU at the University of Chicago and making this possible.

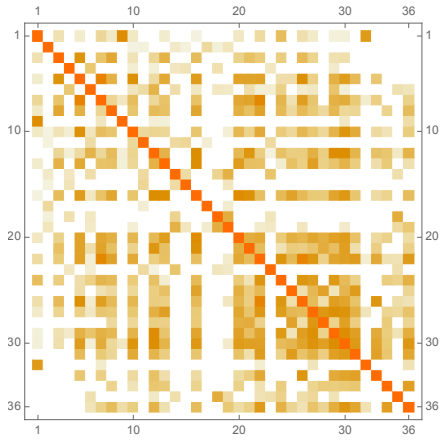
## References

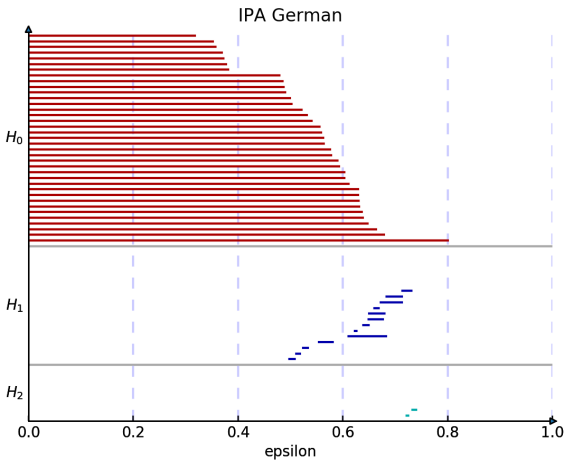
[1] Herbert Edelsbrunner and John L. Harer. Computational Topology, An Introduction. The American Mathematical Society. 2010.
[2] Robert Ghirst. Barcodes: The Persistent Topology of Data. Bulletin (New Series) of the American Mathematical Society, Vol. 45, no. 1, January 2008, pp. 61-75.
[3] Allen Hatcher. Algebraic Topology. Cambridge University Press. 2001.
[4] Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet. Cambridge University Press. 1999.
[5] Rachael-Anne Knight. Phonetics. Cambridge University Press. 2012.
[6] Lyle Campbell. Historical Linguistics: An Introduction (Third Edition). The MIT Press. 2013.
[7] http://reference.wolfram.com/language/tutorial/RandomNumberGeneration.html
[8] Ulrich Bauer. Ripser. https://github.com/Ripser/ripser
[9] https://en.wiktionary.org/wiki/Appendix:Swadesh_lists.
[10] John Goldsmith. Word manifolds presentation. July 2015.
[11] John Goldsmith. Information theoretic approaches to computational linguistics. 2015-2016.
[12] http://www.ancient.eu/Indo-European_Languages/.

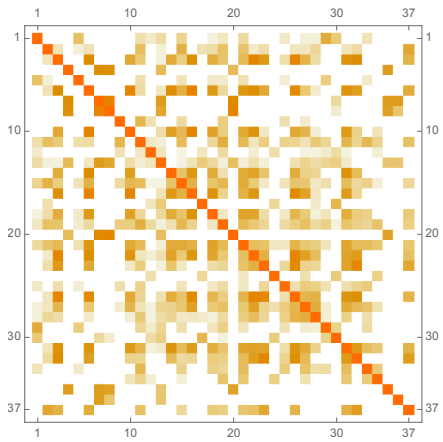
## 9. Appendix I: Family tree for Indo European

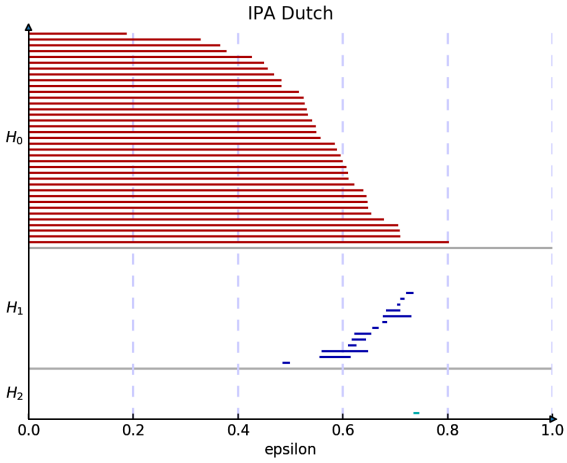
## 10. Appendix II: Barcode Graphs and Matrix Plots for Languages

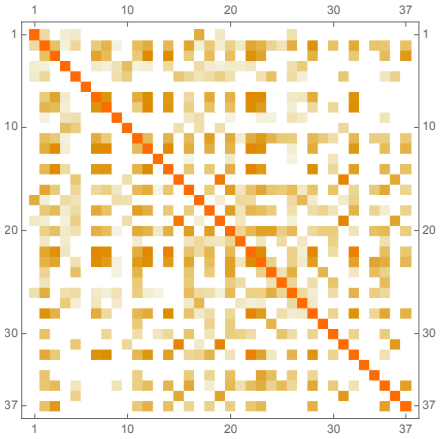


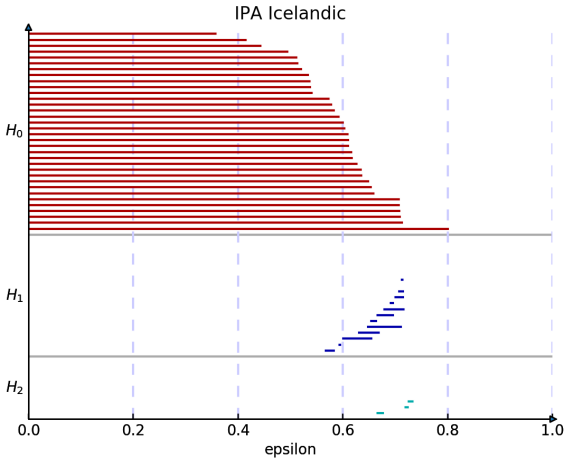



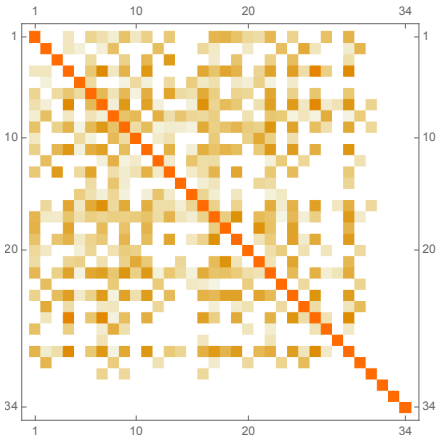



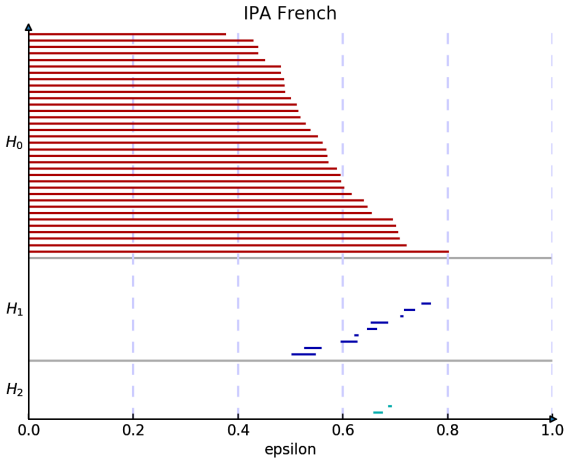



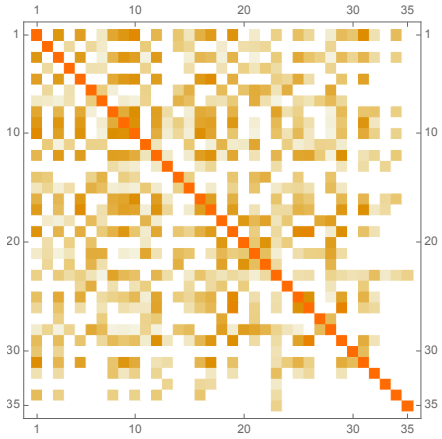

IPA Dutch
$H_0$
$H_1$
$H_2$
0.0
0.2
0.4
0.6
0.8
1.0
epsilon

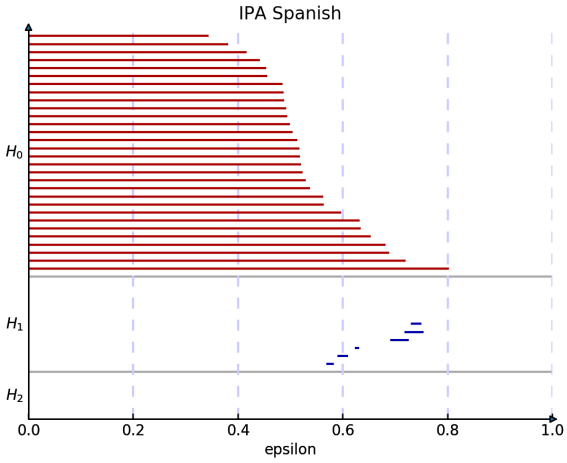

1
10
20
30
37

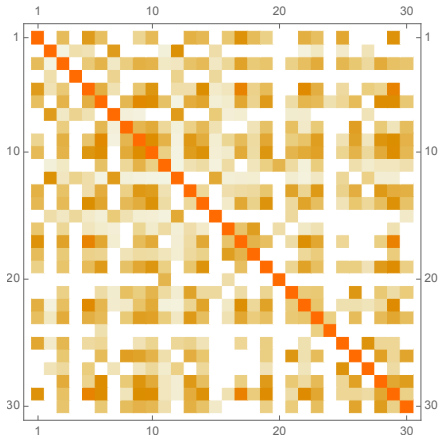

IPA Icelandic
$H_0$
$H_1$
$H_2$
0.0
0.2
0.4
0.6
0.8
1.0
epsilon

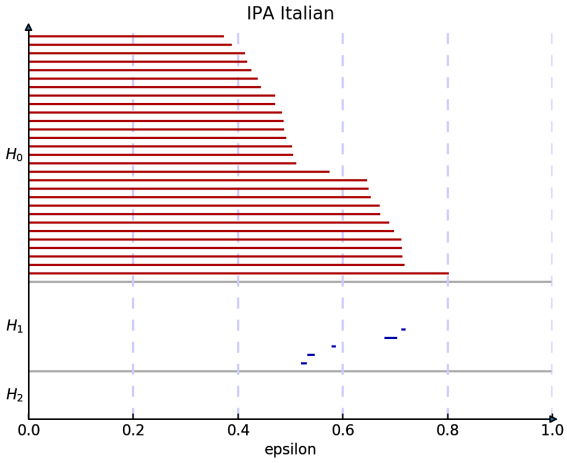

1
10
20
34

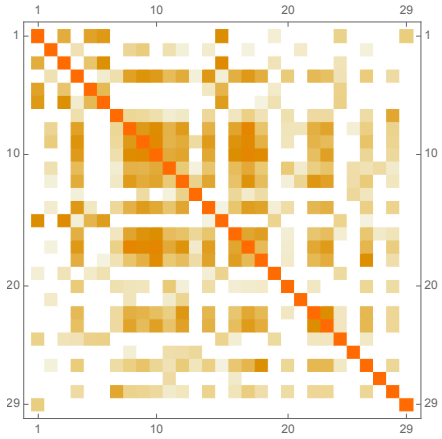

IPA French
$H_0$
$H_1$
$H_2$
0.0
0.2
0.4
0.6
0.8
1.0
epsilon

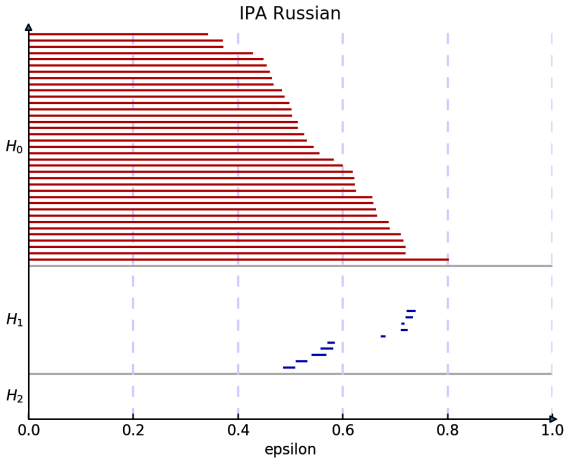

1
10
20
30
35

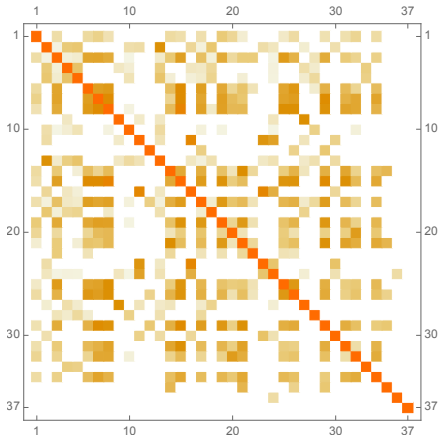

IPA Spanish
$H_0$
$H_1$
$H_2$
0.0
0.2
0.4
0.6
0.8
1.0
epsilon

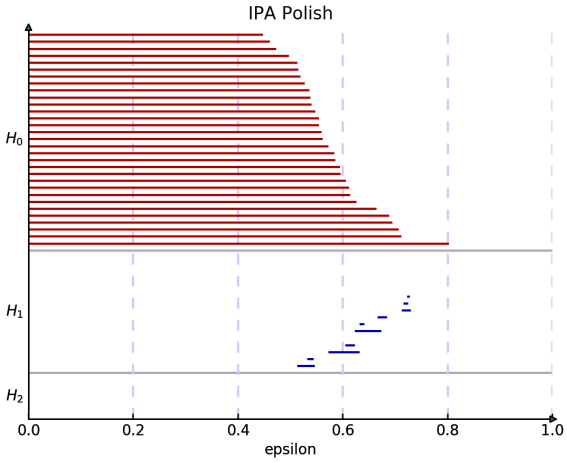

1
10
20
30

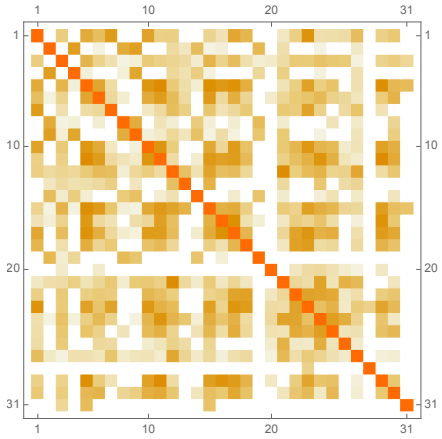

IPA Italian
$H_0$
$H_1$
$H_2$
0.0
0.2
0.4
0.6
0.8
1.0
epsilon

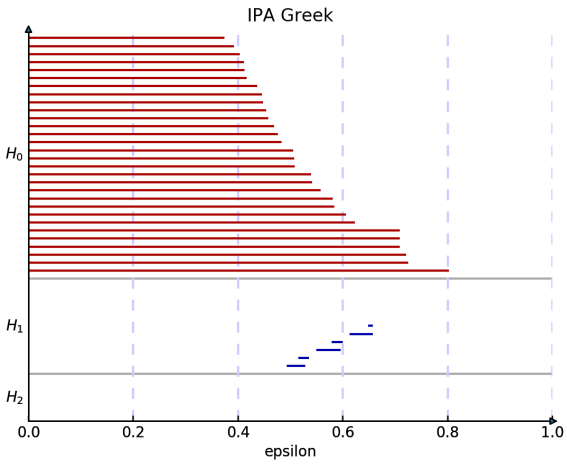

1
10
20
29

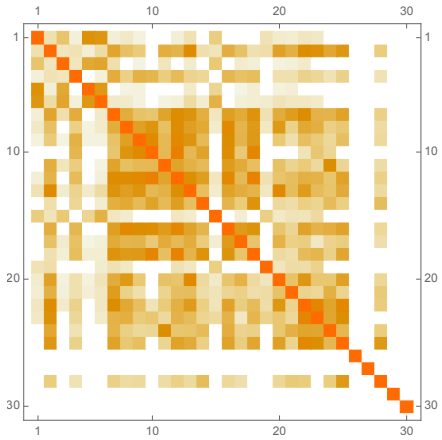

IPA Russian
$H_0$
$H_1$
$H_2$
0.0
0.2
0.4
0.6
0.8
1.0
epsilon

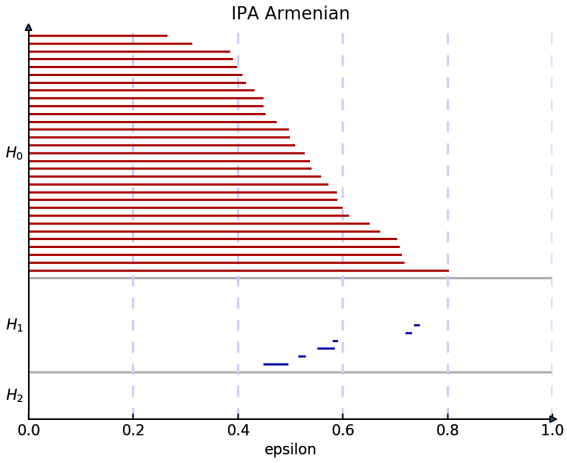

1
10
20
30
37

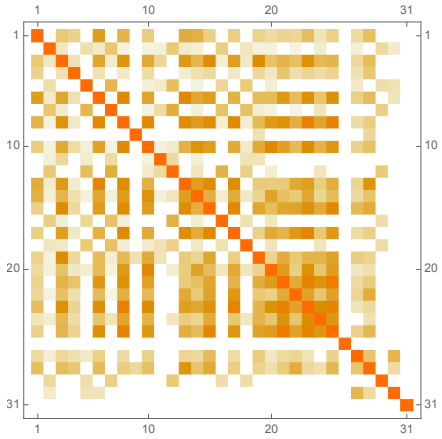

IPA Polish
$H_0$
$H_1$
$H_2$
0.0
0.2
0.4
0.6
0.8
1.0
epsilon

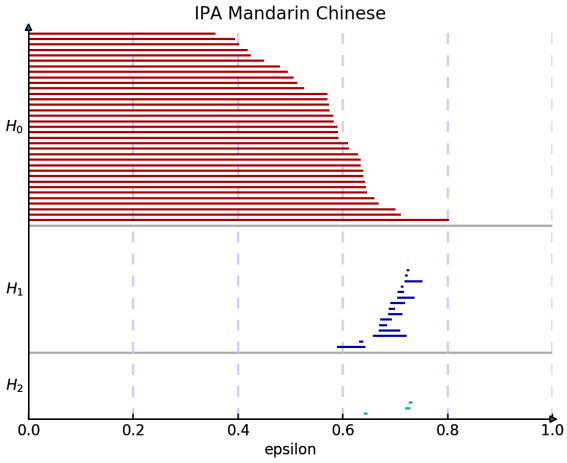

1
10
20
31

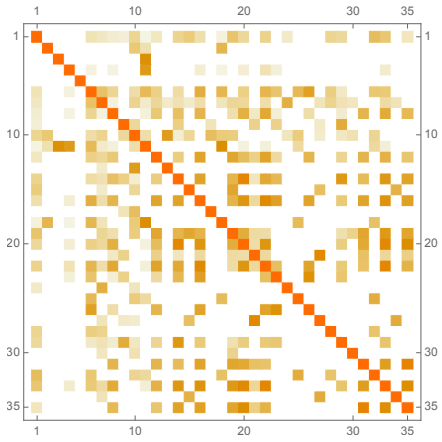

IPA Greek
$H_0$
$H_1$
$H_2$
0.0
0.2
0.4
0.6
0.8
1.0
epsilon

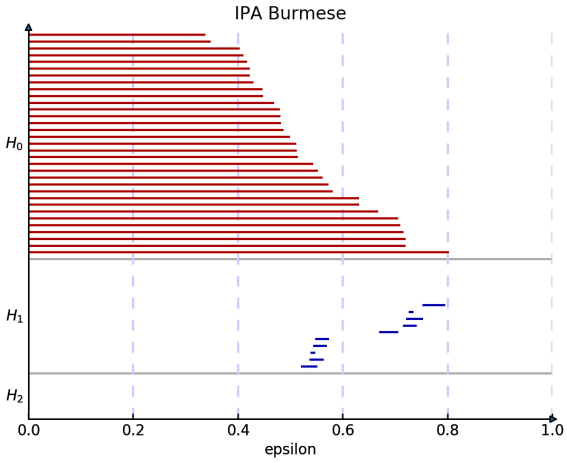

1
10
20
30

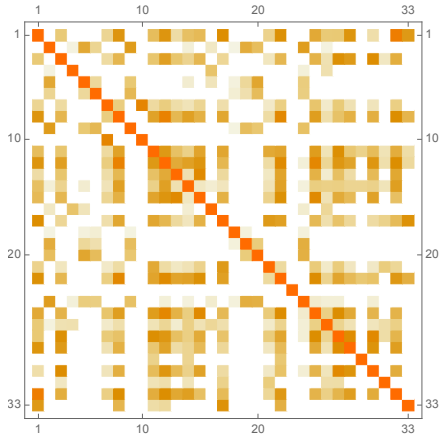

IPA Armenian
$H_0$
$H_1$
$H_2$
0.0
0.2
0.4
0.6
0.8
1.0
epsilon

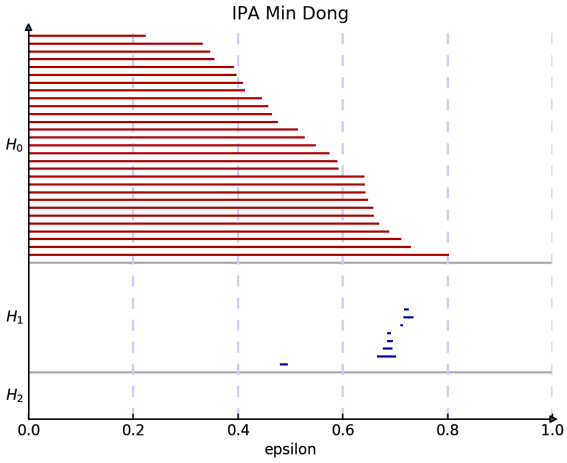

1
10
20
31

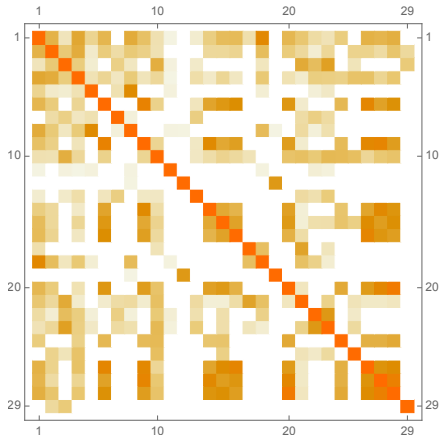

IPA Mandarin Chinese
$H_0$
$H_1$
$H_2$
0.0
0.2
0.4
0.6
0.8
1.0
epsilon

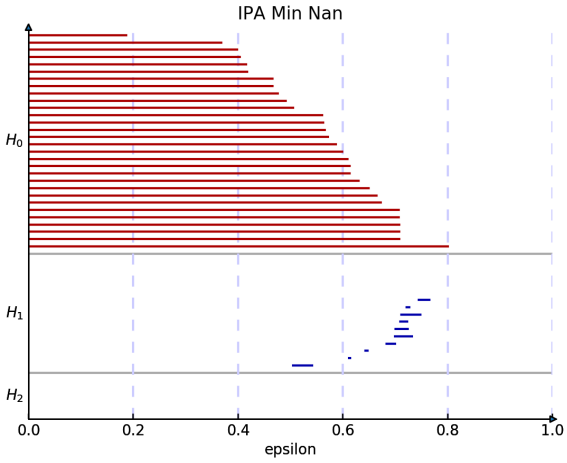

1
10
20
30
35

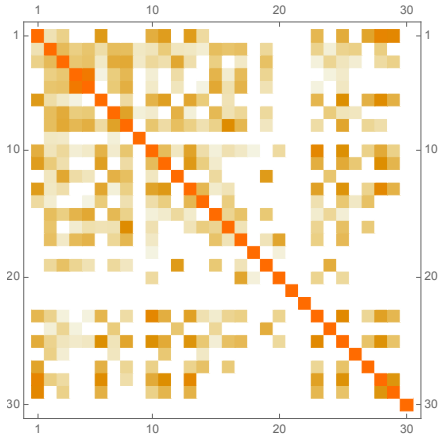

IPA Burmese
$H_0$
$H_1$
$H_2$
0.0
0.2
0.4
0.6
0.8
1.0
epsilon

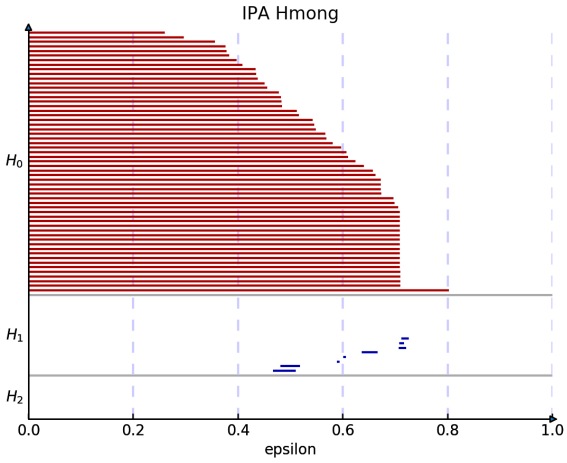

1
10
20
33

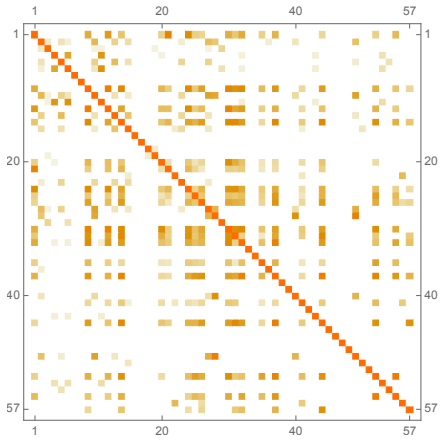

IPA Min Dong
$H_0$
$H_1$
$H_2$
0.0
0.2
0.4
0.6
0.8
1.0
epsilon

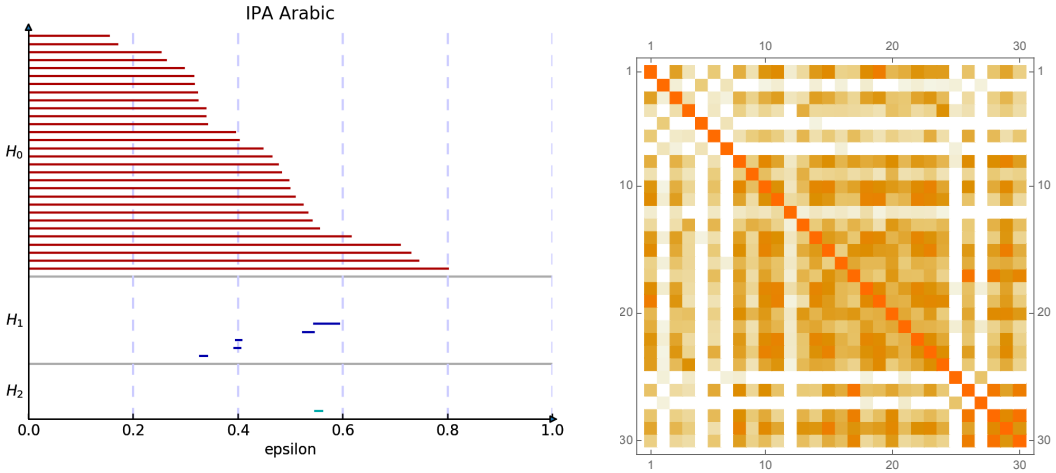

1
10
20
29


IPA Min Nan
$H_0$
$H_1$
$H_2$
0.0
0.2
0.4
0.6
0.8
1.0
epsilon

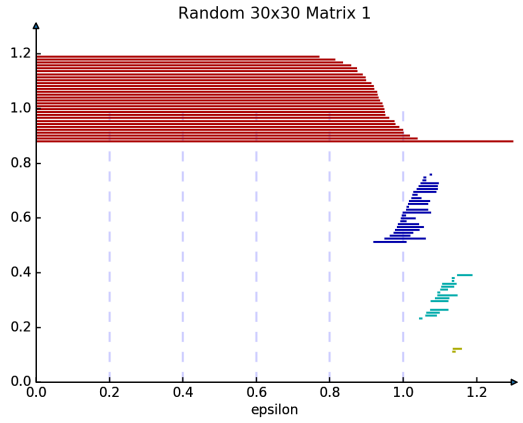

1
10
20
30

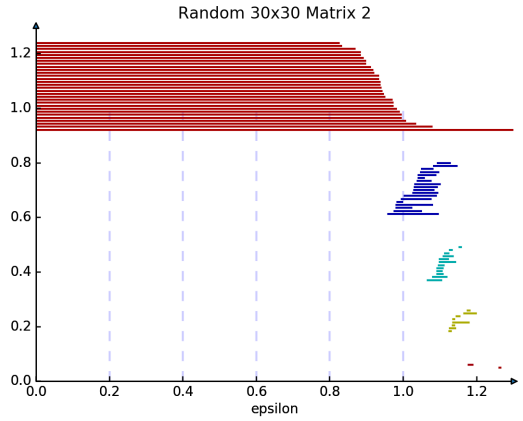

IPA Hmong
$H_0$
$H_1$
$H_2$
0.0
0.2
0.4
0.6
0.8
1.0
epsilon

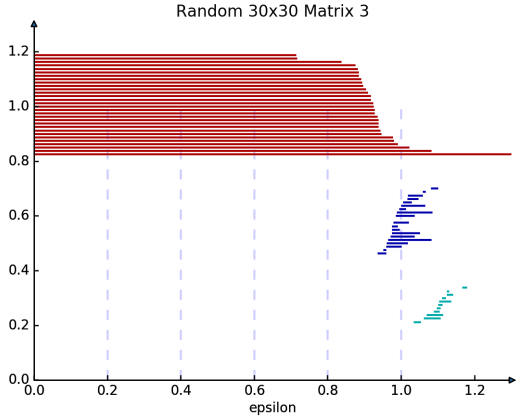

1
20
40
57

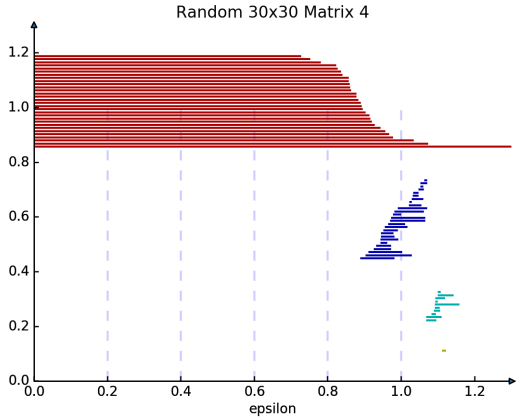


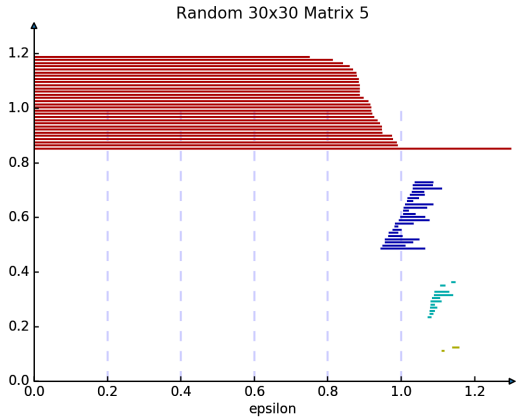

## 11. Appendix III: Barcode Graphs for Random Matrices

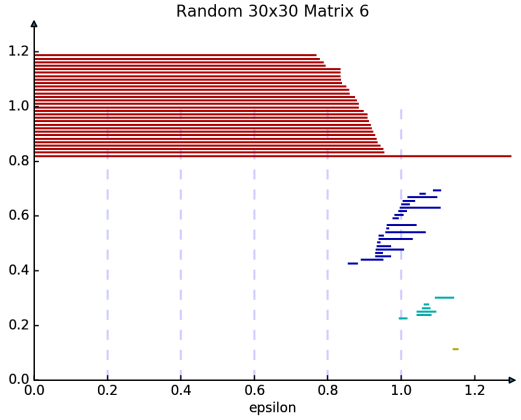


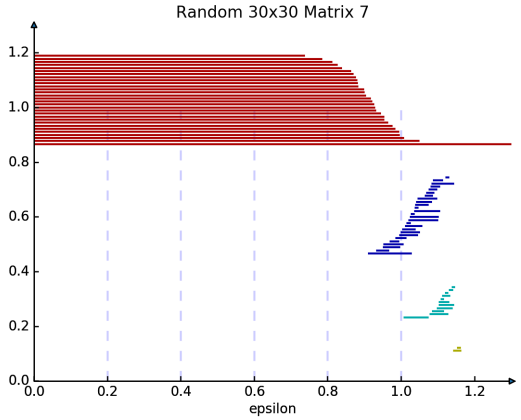



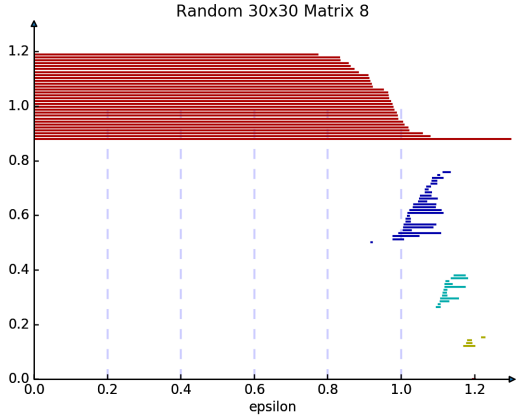

Random 30x30 Matrix 3
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
epsilon

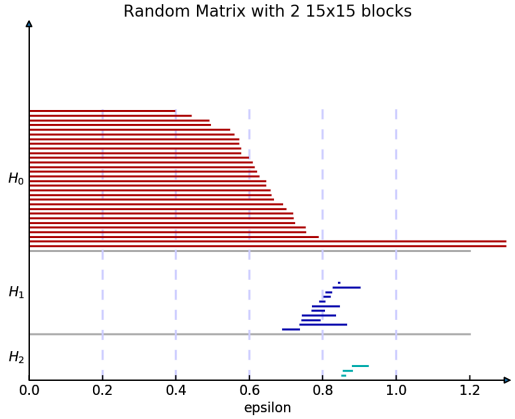

Random 30x30 Matrix 4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
epsilon

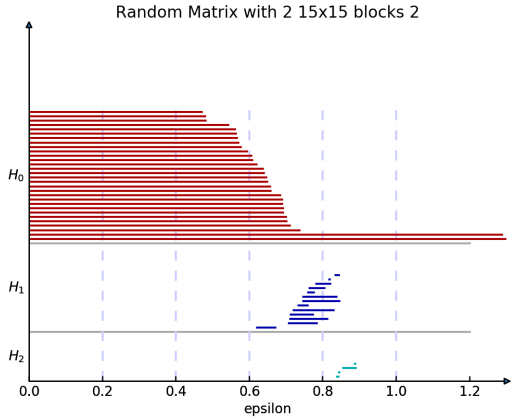

Random 30x30 Matrix 5
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
epsilon

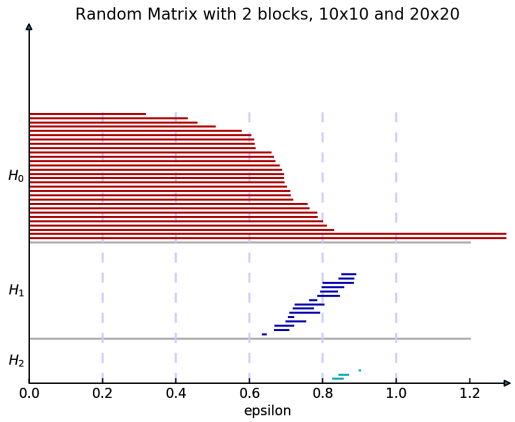

Random 30x30 Matrix 6
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
epsilon


Random 30x30 Matrix 7
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
epsilon


Random 30x30 Matrix 8
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
epsilon




## 12. Appendix IV: English

- The Swadesh list for (American) English in IPA:

In[500]:=
```
IPAamericanenglish =
 {aɪ, ju, hi, wi, ju , ðeɪ, ðɪs, ðæt, hir, ðɛr, hu, wʌt, wɛr, wɛn, haʊ, nɑt, ɔl,
  'mɛni, sʌm, fju, 'ʌðər, wʌn, tu, θri, fɔr, faɪv, bɪg, lɔŋ, waɪd, θɪk, 'hɛvi, smɔl,
  ʃɔrt, 'nɛroʊ, θɪn, 'wʊmən, mən , mən , ʧaɪld, waɪf, 'hʌzbənd, 'mʌðər, 'fɑðər,
  'ænəməl, fɪʃ, b3rd, dɔg, laʊs, sneɪk, w3rm, tri, 'fɔrəst, stɪk, frut, sid, lif,
  rut, bɑrk, 'flaʊər, græs, roʊp, skɪn, mit, blʌd, boʊn, fæt, ɛg, hɔrn, teɪl,
  'fɛðər, hɛr, hɛd, ir, aɪ, noʊz, maʊθ, tuθ, tʌŋ , 'fɪŋgər¡neɪl, fʊt, lɛg,
  ni, hænd, wɪŋ, 'bɛli, gʌts, nɛk, bæk, brɛst, hɑrt, 'lɪvər, drɪŋk, it, baɪt,
  sʌk, spɪt, 'vɑmət, bloʊ, brið, læf, si, hir, noʊ, θɪŋk, smɛl, fɪr, slip,
  lɪv, daɪ, kɪl, faɪt, hʌnt, hɪt, kʌt, splɪt, stæb, skræʧ, dɪg, swɪm, flaɪ,
  wɔk, kʌm, laɪ , sɪt, stænd, t3rn, fɔl, gɪv, hoʊld, skwiz, rʌb, wɑʃ, waɪp,
  pʊl, pʊʃ, θroʊ, taɪ, soʊ, kaʊnt, seɪ, sɪŋ, pleɪ, floʊt, floʊ, friz, swɛl,
  sʌn, mun, stɑr, 'wɔtər, reɪn, 'rɪvər, leɪk, si, sɔlt, stoʊn, sænd, dʌst,
  3rθ, klaʊd, fɑg, skaɪ, wɪnd, snoʊ, aɪs, smoʊk, 'faɪər, æʃ, tu b3rn, roʊd,
  'maʊntən, rɛd, grin, 'jɛloʊ, waɪt, blæk, naɪt, deɪ, jɪr, wɔrm, koʊld, fʊl,
  nu, oʊld, gʊd, bæd, 'rɑtən, 'd3rti, streɪt, raʊnd, ʃɑrp, dʌl , smuð, wɛt,
  draɪ, kə'rɛkt, nɪr, fɑr, raɪt, lɛft, æt, ɪn, wɪð, ænd, ɪf, bɪ'kɔz, neɪm};
```

- Put into useful form:

In[501]:=
```
flatipaamerican = StringReplace[StringReplace[StringReplace[StringReplace[
     Flatten[Characters[Map[ToString, Partition[IPAamericanenglish, {1}]]]],
     "'" → " "], "¡" → " "], "}" → " "], "{" → " " ]
```

Out[501]= { , a, ɪ, , , j, u, , , h, i, , , w, i, , , j, u, , , ð, e, ɪ, , , ð, ɪ, s, , , ð,
æ, t, , , h, i, r, , , ð, ɛ, r, , , h, u, , , w, ʌ, t, , , w, ɛ, r, , , w, ɛ,
n, , , h, a, ʊ, , , n, ɑ, t, , , ɔ, l, , , , m, ɛ, n, i, , , s, ʌ, m, , , f,
j, u, , , , ʌ, ð, ə, r, , , w, ʌ, n, , , t, u, , , ϴ, r, i, , , f, ɔ, r, , ,
f, a, ɪ, v, , , b, ɪ, g, , , l, ɔ, ŋ, , , w, a, ɪ, d, , , ϴ, ɪ, k, , , , h, ɛ,
v, i, , , s, m, ɔ, l, , , ʃ, ɔ, r, t, , , , n, ɛ, r, o, ʊ, , , ϴ, ɪ, n, , , ,
w, ʊ, m, ə, n, , , m, ə, n, , , m, ə, n, , , ʧ, a, ɪ, l, d, , , w, a, ɪ, f, , ,
, h, ʌ, z, b, ə, n, d, , , , m, ʌ, ð, ə, r, , , , f, ɑ, ð, ə, r, , , , æ, n,
ə, m, ə, l, , , f, ɪ, ʃ, , , b, ʒ, r, d, , , d, ɔ, g, , , l, a, ʊ, s, , , s,
n, e, ɪ, k, , , w, ʒ, r, m, , , t, r, i, , , , f, ɔ, r, ə, s, t, , , s, t, ɪ,
k, , , f, r, u, t, , , s, i, d, , , l, i, f, , , r, u, t, , , b, ɑ, r, k, ,
, , f, l, a, ʊ, ə, r, , , g, r, æ, s, , , r, o, ʊ, p, , , s, k, ɪ, n, , , m,
i, t, , , b, l, ʌ, d, , , b, o, ʊ, n, , , f, æ, t, , , ɛ, g, , , h, ɔ, r, n,
, , t, e, ɪ, l, , , , f, ɛ, ð, ə, r, , , h, ɛ, r, , , h, ɛ, d, , , i, r, ,
, a, ɪ, , , n, o, ʊ, z, , , m, a, ʊ, ϴ, , , t, u, ϴ, , , t, ʌ, ŋ, , , , f,
ɪ, ŋ, g, ə, r, , n, e, ɪ, l, , , f, ʊ, t, , , l, ɛ, g, , , n, i, , , h, æ, n,
d, , , w, ɪ, ŋ, , , , b, ɛ, l, i, , , g, ʌ, t, s, , , n, ɛ, k, , , b, æ, k,
, , b, r, ɛ, s, t, , , h, ɑ, r, t, , , , l, ɪ, v, ə, r, , , d, r, ɪ, ŋ, k, ,
, i, t, , , b, a, ɪ, t, , , s, ʌ, k, , , s, p, ɪ, t, , , , v, ɑ, m, ə, t, ,
, b, l, o, ʊ, , , b, r, i, ð, , , l, æ, f, , , s, i, , , h, i, r, , , n, o, ʊ,
, , ϴ, ɪ, ŋ, k, , , s, m, ɛ, l, , , f, ɪ, r, , , s, l, i, p, , , l, ɪ, v, , ,
d, a, ɪ, , , k, ɪ, l, , , f, a, ɪ, t, , , h, ʌ, n, t, , , h, ɪ, t, , , k, ʌ,
t, , , s, p, l, ɪ, t, , , s, t, æ, b, , , s, k, r, æ, ʧ, , , d, ɪ, g, , , s, w,
ɪ, m, , , f, l, a, ɪ, , , w, ɔ, k, , , k, ʌ, m, , , l, a, ɪ, , , s, ɪ, t, , ,
s, t, æ, n, d, , , t, ʒ, r, n, , , f, ɔ, l, , , g, ɪ, v, , , h, o, ʊ, l, d, ,
, s, k, w, i, z, , , r, ʌ, b, , , w, ɑ, ʃ, , , w, a, ɪ, p, , , p, ʊ, l, , , p,
ʊ, ʃ, , , ϴ, r, o, ʊ, , , t, a, ɪ, , , s, o, ʊ, , , k, a, ʊ, n, t, , , s, e,
ɪ, , , s, ɪ, ŋ, , , p, l, e, ɪ, , , f, l, o, ʊ, t, , , f, l, o, ʊ, , , f, r,
i, z, , , s, w, ɛ, l, , , s, ʌ, n, , , m, u, n, , , s, t, ɑ, r, , , , w, ɔ,
t, ə, r, , , r, e, ɪ, n, , , , r, ɪ, v, ə, r, , , l, e, ɪ, k, , , s, i, , ,
s, ɔ, l, t, , , s, t, o, ʊ, n, , , s, æ, n, d, , , d, ʌ, s, t, , , ʒ, r, ϴ, ,
, k, l, a, ʊ, d, , , f, ɑ, g, , , s, k, a, ɪ, , , w, ɪ, n, d, , , s, n, o, ʊ,
, , a, ɪ, s, , , s, m, o, ʊ, k, , , , f, a, ɪ, ə, r, , , æ, ʃ, , , b, ʒ, r,
n, , t, u, , , r, o, ʊ, d, , , , m, a, ʊ, n, t, ə, n, , , r, ɛ, d, , , g, r,
i, n, , , , j, ɛ, l, o, ʊ, , , w, a, ɪ, t, , , b, l, æ, k, , , n, a, ɪ, t, ,
, d, e, ɪ, , , j, ɪ, r, , , w, ɔ, r, m, , , k, o, ʊ, l, d, , , f, ʊ, l, , ,
n, u, , , o, ʊ, l, d, , , g, ʊ, d, , , b, æ, d, , , , r, ɑ, t, ə, n, , , ,
d, ʒ, r, t, i, , , s, t, r, e, ɪ, t, , , r, a, ʊ, n, d, , , ʃ, ɑ, r, p, , , d,
ʌ, l, , , s, m, u, ð, , , w, ɛ, t, , , d, r, a, ɪ, , , k, ə, , r, ɛ, k, t, ,
, n, ɪ, r, , , f, ɑ, r, , , r, a, ɪ, t, , , l, ɛ, f, t, , , æ, t, , , ɪ, n,
, , w, ɪ, ð, , , æ, n, d, , , ɪ, f, , , b, ɪ, , k, ɔ, z, , , n, e, ɪ, m, }

- All characters that appear in this data set:

In[502]:=
```
ipaenglishcharacters =
 DeleteCases[Delete[Flatten[Tally@StringSplit[flatipaamerican, ""]], 1], _Integer]
```

Out[502]=
```
{a, ɪ, j, u, h, i, w, ð, e, s, æ, t, r, ɛ, ʌ, n,
 ʊ, ɑ, ɔ, l, m, f, ə, ɵ, v, b, g, ŋ, d, k, ʃ, o, ʧ, z, ʒ, p}
```

In[503]:=
```
Length[ipaenglishcharacters]
```

Out[503]= 36

In[504]:=
```
f[x_] := Map[Delete[##, 2] &,
  Take[flatipaamerican, {# - 1, # + 1}] & /@ Flatten[Position[flatipaamerican, x]]]
```

- Given a character “x”, f[x] looks at every instance of x in the data above, and finds the order pair of the character immediately before it and immediately after it. For example:

In[505]:=
```
f["a"]
```

Out[505]=
```
{{ , ɪ}, {h, ʊ}, {f, ɪ}, {w, ɪ}, {ʧ, ɪ}, {w, ɪ}, {l, ʊ}, {l, ʊ}, { , ɪ},
 {m, ʊ}, {b, ɪ}, {d, ɪ}, {f, ɪ}, {l, ɪ}, {l, ɪ}, {w, ɪ}, {t, ɪ}, {k, ʊ},
 {l, ʊ}, {k, ɪ}, { , ɪ}, {f, ɪ}, {m, ʊ}, {w, ɪ}, {n, ɪ}, {r, ʊ}, {r, ɪ}, {r, ɪ}}
```

In[506]:=
```
f["h"]
```

Out[506]=
```
{{ , i}, { , i}, { , u}, { , a}, { , ɛ}, { , ʌ}, { , ɔ},
 { , ɛ}, { , ɛ}, { , æ}, { , ɑ}, { , i}, { , ʌ}, { , ɪ}, { , o}}
```

In[507]:=
```
f["e"]
```

Out[507]=
```
{{ð, ɪ}, {n, ɪ}, {t, ɪ}, {n, ɪ}, {s, ɪ}, {l, ɪ}, {r, ɪ}, {l, ɪ}, {d, ɪ}, {r, ɪ}, {n, ɪ}}
```

- This gives the contexts for all characters:

In[508]:=
```
ipaenglishenvironments = f[#] & /@ ipaenglishcharacters;
```

- This gives the environments for the 7th character from the list ipaenglishcharacters:

In[509]:=
```
ipaenglishenvironments[[7]]
```

Out[509]=
```
{{ , i}, { , ʌ}, { , ɛ}, { , ɛ}, { , ʌ}, { , a}, { , ʊ}, { , a}, { , ɜ}, { , ɪ}, {s, ɪ},
 { , ɔ}, {k, i}, { , ɑ}, { , a}, {s, ɛ}, { , ɔ}, { , ɪ}, { , a}, { , ɔ}, { , ɛ}, { , ɪ}}
```

- The following counts the number of shared environments between two English characters “x” and “y”. This count includes multiplicity, so if “x” and “y” both have the environment { , a} twice, then it will be counted twice.

In[510]:=
```
ipaenglishedgeweights[x_, y_] :=
 Total[Merge[KeyIntersection[{Counts[ipaenglishenvironments[[x]]],
     Counts[ipaenglishenvironments[[y]]]}], Min]]
```

- For example:

In[511]:=
```
ipaenglishedgeweights[2, 17]
```

Out[511]= 5

In[512]:=
```
ipaenglishcharacters[[2]]
```

Out[512]=
```
I
```

In[513]:=
```
ipaenglishcharacters[[17]]
```

Out[513]=
```
ʊ
```

- Thus “ɪ” and “ʊ” share 5 environments. These are:

In[514]:=
```
Merge[KeyIntersection[{Counts[ipaenglishenvironments[[2]]],
    Counts[ipaenglishenvironments[[17]]]}], Min]
```

Out[514]=
```
⟨|{a,  } → 1, {a, d} → 1, {w, m} → 1, {a, s} → 1, {a, ə} → 1|⟩
```

- This function gives the total number of times a character appears in the dataset:

In[515]:=
```
totalappearanceipaenglish[i_] := Length[ipaenglishenvironments[[i]]]
```

- Using this, we can compute the actual entries in the coordinate matrix, namely the cosine similarity between two context sets:

In[516]:=
```
ipaenglishmatrixentries[x_, y_] := If[x == y, 1, ipaenglishedgeweights[x, y] /
    Sqrt[totalappearanceipaenglish[x] totalappearanceipaenglish[y]]]
```

- And now we can put this in matrix form (the matrix gets a bit cutoff in this form unfortunately):

In[533]:=
```
ipaenglishmatrix = MatrixForm[
  Table[Round[N[ipaenglishmatrixentries[a, b]], 0.001], {a, 1, 36}, {b, 1, 36}]]
```

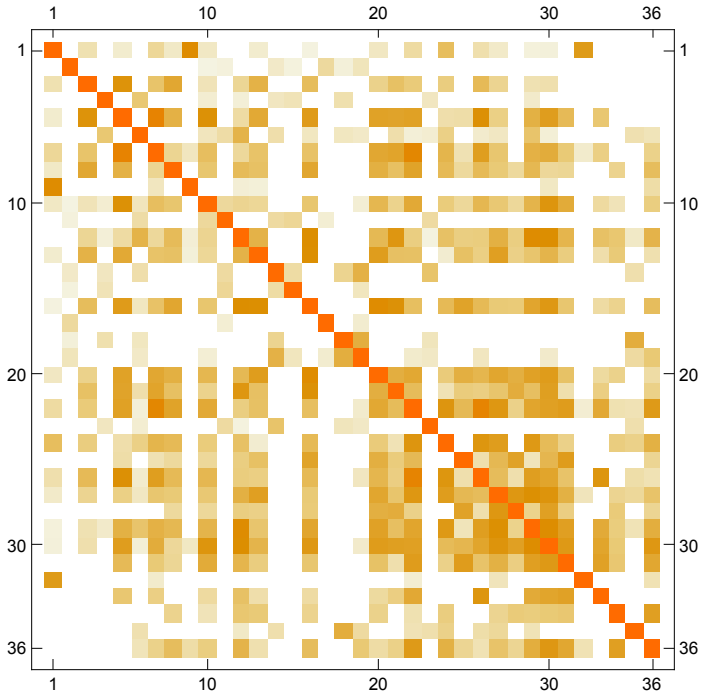
Out[533]//MatrixForm=

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1. | 0. | 0.085 | 0. | 0.049 | 0. | 0.121 | 0.057 | 0.399 | 0.059 | 0. | 0. | 0.049 |
| 0. | 1. | 0. | 0. | 0. | 0. | 0. | 0. | 0. | 0.02 | 0.032 | 0. | 0. |
| 0.085 | 0. | 1. | 0. | 0.346 | 0. | 0.191 | 0.27 | 0. | 0.07 | 0. | 0.123 | 0.231 |
| 0. | 0. | 0. | 1. | 0. | 0.181 | 0. | 0. | 0. | 0.045 | 0. | 0.04 | 0. |
| 0.049 | 0. | 0.346 | 0. | 1. | 0. | 0.55 | 0.234 | 0. | 0.363 | 0. | 0.106 | 0.267 |
| 0. | 0. | 0. | 0.181 | 0. | 1. | 0. | 0. | 0. | 0.065 | 0.104 | 0.229 | 0. |
| 0.121 | 0. | 0.191 | 0. | 0.55 | 0. | 1. | 0.129 | 0.064 | 0.2 | 0. | 0.117 | 0.193 |
| 0.057 | 0. | 0.27 | 0. | 0.234 | 0. | 0.129 | 1. | 0. | 0.188 | 0. | 0.207 | 0.195 |
| 0.399 | 0. | 0. | 0. | 0. | 0. | 0.064 | 0. | 1. | 0. | 0. | 0.041 | 0.039 |
| 0.059 | 0.02 | 0.07 | 0.045 | 0.363 | 0.065 | 0.2 | 0.188 | 0. | 1. | 0.117 | 0.129 | 0.121 |
| 0. | 0.032 | 0. | 0. | 0. | 0.104 | 0. | 0. | 0. | 0.117 | 1. | 0. | 0. |
| 0. | 0. | 0.123 | 0.04 | 0.106 | 0.229 | 0.117 | 0.207 | 0.041 | 0.129 | 0. | 1. | 0.231 |
| 0.049 | 0. | 0.231 | 0. | 0.267 | 0. | 0.193 | 0.195 | 0.039 | 0.121 | 0. | 0.231 | 1. |
| 0. | 0.055 | 0. | 0.063 | 0. | 0.091 | 0. | 0. | 0. | 0. | 0.109 | 0. | 0. |
| 0. | 0.031 | 0. | 0.07 | 0. | 0. | 0. | 0. | 0. | 0. | 0.121 | 0. | 0. |
| 0.028 | 0. | 0.202 | 0. | 0.311 | 0.063 | 0.193 | 0.273 | 0. | 0.188 | 0. | 0.373 | 0.37 |
| 0. | 0.112 | 0. | 0. | 0. | 0. | 0. | 0. | 0. | 0. | 0.044 | 0. | 0. |
| 0. | 0.038 | 0. | 0.087 | 0. | 0.063 | 0. | 0. | 0. | 0. | 0. | 0. | 0. |
| 0. | 0.068 | 0. | 0. | 0. | 0.056 | 0. | 0. | 0. | 0.042 | 0. | 0. | 0. |
| 0.06 | 0. | 0.141 | 0. | 0.286 | 0. | 0.27 | 0.238 | 0. | 0.222 | 0. | 0.217 | 0.306 |
| 0. | 0. | 0.195 | 0. | 0.282 | 0.091 | 0.279 | 0.197 | 0. | 0.136 | 0. | 0.33 | 0.197 |
| 0.107 | 0. | 0.169 | 0. | 0.293 | 0.039 | 0.483 | 0.285 | 0. | 0.266 | 0. | 0.156 | 0.195 |
| 0. | 0. | 0. | 0.063 | 0. | 0.046 | 0. | 0. | 0. | 0. | 0.109 | 0.03 | 0. |
| 0.2 | 0. | 0.158 | 0. | 0.091 | 0.147 | 0.226 | 0.213 | 0. | 0.166 | 0. | 0.194 | 0.046 |
| 0. | 0. | 0. | 0. | 0.098 | 0. | 0.081 | 0.114 | 0. | 0.118 | 0. | 0.156 | 0.195 |
| 0.089 | 0. | 0.211 | 0. | 0.365 | 0.049 | 0.302 | 0.213 | 0. | 0.221 | 0. | 0.162 | 0.152 |
| 0.055 | 0. | 0.129 | 0. | 0.149 | 0.06 | 0.185 | 0.174 | 0. | 0.18 | 0. | 0.238 | 0.298 |
| 0. | 0. | 0. | 0. | 0. | 0. | 0. | 0.114 | 0. | 0.118 | 0. | 0.156 | 0.146 |
| 0.035 | 0. | 0.082 | 0.053 | 0.236 | 0.19 | 0.234 | 0.22 | 0. | 0.228 | 0. | 0.401 | 0.189 |
| 0.037 | 0. | 0.088 | 0. | 0.304 | 0.041 | 0.251 | 0.118 | 0.059 | 0.337 | 0. | 0.377 | 0.228 |
| 0. | 0. | 0. | 0. | 0.211 | 0. | 0.174 | 0.123 | 0. | 0.255 | 0. | 0.224 | 0.158 |
| 0.312 | 0. | 0. | 0. | 0. | 0. | 0.05 | 0. | 0. | 0. | 0. | 0. | 0. |
| 0. | 0. | 0. | 0. | 0.183 | 0. | 0.151 | 0. | 0. | 0.11 | 0. | 0.194 | 0.091 |
| 0. | 0. | 0. | 0. | 0. | 0. | 0. | 0.135 | 0. | 0.07 | 0. | 0.184 | 0.173 |
| 0. | 0. | 0. | 0. | 0. | 0.085 | 0. | 0. | 0. | 0. | 0. | 0.056 | 0. |
| 0. | 0. | 0. | 0. | 0. | 0.07 | 0.142 | 0.201 | 0.101 | 0.156 | 0. | 0.229 | 0.086 |

- We can also see this matrix as a color plot, which gives us a more visual way to look at the matrix:

In[518]:=
```
MatrixPlot[Table[ipaenglishmatrixentries[a, b], {a, 1, 36}, {b, 1, 36}]]
```

Out[518]=




- From this, we define the coordinates of the sounds as the rows (or equivalently columns) of this matrix:

In[519]:=
```
ipaenglishcoordinates =
  Table[ipaenglishmatrixentries[a, b], {a, 1, 36}, {b, 1, 36}].UnitVector[36, #] & /@
   Range[1, 36];
```

- We now define the association between coordinates and sounds. The k^th of these vectors corresponds to the k^th character (in the order that they appear in ipaenglishcharacters above).

In[528]:=
```
ipaenglishpointstosounds =
  Merge[Association[ipaenglishcoordinates[[#]] → ipaenglishcharacters[[#]]] & /@
    Range[1, 36], Total];
```

- Using this, we can look up any element we want, for example:

In[521]:=
```
Lookup[ipaenglishpointstosounds, {ipaenglishcoordinates[[1]]}]
```

Out[521]=
```
{a}
```

In[522]:=
```
Lookup[ipaenglishpointstosounds, {ipaenglishcoordinates[[12]]}]
```

Out[522]=
```
{t}
```

- We can define a neighborhood of a sound x with a radius r and get the set of sounds that are within distance r of x in the coordinate embedding:

In[523]:=
```
ipaenglishneighborhood[x_, r_] := Lookup[ipaenglishpointstosounds,
  Nearest[ipaenglishcoordinates, ipaenglishcoordinates[[x]], {36, r}]]
```

- Here is are all the neighborhoods of radius 1.5. We will use these in a moment to construct a preliminary graph.

In[524]:=
```
ipaenghood = ipaenglishneighborhood[#, 1.5] & /@ Range[1, 36]
```

Out[524]=
```
{{a, e, o, θ, j}, {ɪ, ʊ, ɔ, ɛ, ʌ, æ, ɑ, ə, u, ɜ, o, i, e},
 {j, ð, h, r, m, w, b, n, θ, l, s, g, f, t, a}, {u, i, ʌ, ɑ, ə, ɛ, æ, ɪ, ʊ, ɜ, ɔ, o, e},
 {h, w, b, s, j, l, f, m, n, r, k, ð, d, ʃ, ʧ, g, t, θ},
 {i, u, æ, ɜ, ɛ, ə, ɑ, t, ɔ, θ, ʌ, ɪ, ʊ},
 {w, h, f, b, l, m, s, k, d, θ, j, r, n, ð, g, ʃ, ʧ, t, p},
 {ð, j, n, l, r, θ, m, f, s, t, b, d, g, h, p, w, ʃ, z, v, ŋ, k}, {e, a, o, ʊ, ɪ, ʌ, u, æ, ə},
 {s, h, k, ʃ, l, f, b, d, w, ð, g, n, θ, m, p, r, t, v, ʧ, ŋ, j, z},
 {æ, ʌ, ə, ɛ, i, ʊ, ɪ, u, ɔ, ɑ, ɜ, o, e},
 {t, d, n, k, m, g, l, r, ʃ, p, ð, θ, ŋ, s, z, v, ʧ, b, f, i, w, j, h},
 {r, n, l, g, m, ð, t, h, j, v, k, d, w, ʃ, ŋ, s, f, z, b, p, ʧ, θ},
 {ɛ, ɔ, ə, ɑ, æ, ʌ, ɜ, u, i, ɪ, ʊ}, {ʌ, æ, ɛ, u, ɔ, ɪ, ə, ʊ, ɑ, o, ɜ, i, e},
 {n, l, r, m, t, k, ð, d, h, v, g, s, ʃ, b, w, f, j, p, θ, ŋ, z},
 {ʊ, ɪ, æ, ɔ, ʌ, ə, u, ɛ, o, ɑ, ɜ, e, i},
 {ɑ, ɜ, ɔ, ɛ, u, ə, ɪ, i, ʌ, ʊ, æ}, {ɔ, ɑ, ɛ, ɜ, ɪ, ə, ʌ, ʊ, i, u, æ, p},
 {l, n, r, k, g, d, m, h, ð, s, w, f, t, v, ʃ, b, ŋ, θ, j, p, z, ʧ},
 {m, n, t, l, k, r, h, w, ð, ŋ, d, g, j, v, s, f, b, ʧ, ʃ, p, θ},
 {f, b, w, k, g, θ, d, ʃ, h, l, s, ð, p, n, m, r, ʧ, t, v, j},
 {ə, ɛ, æ, ɜ, ɑ, u, ɔ, i, ʌ, ɪ, ʊ, o, e},
 {θ, d, b, g, f, ð, p, w, s, t, l, n, k, j, ʃ, z, i, a, m, r, h, ɜ},
 {v, ŋ, ʃ, l, n, r, g, m, k, s, t, ð, p, f, z, d},
 {b, f, w, h, θ, d, ʧ, s, k, l, ð, ʃ, g, n, j, m, r, t, p},
 {g, d, k, l, ŋ, r, θ, f, t, ʃ, v, n, ð, s, p, m, b, w, z, h, j},
 {ŋ, v, g, m, l, k, z, r, ʃ, t, p, n, ð, s, d},
 {d, t, k, g, θ, ʃ, l, b, f, n, s, p, m, ð, w, r, ʧ, h, z, ŋ, v},
 {k, d, t, ʃ, g, l, n, s, m, f, p, b, r, h, w, ŋ, ʧ, v, θ, ð, z},
 {ʃ, k, d, v, s, f, p, ʧ, g, t, l, b, n, r, ŋ, z, w, ð, h, θ, m},
 {o, a, e, ʊ, ɪ, ʌ, u, æ, ə, ɜ}, {ʧ, b, ʃ, d, k, f, s, t, m, w, h, l, r},
 {z, p, ŋ, ʃ, t, r, ð, θ, g, d, l, s, k, n, v}, {ɜ, ɑ, ɔ, ə, ɛ, i, u, ɪ, ʌ, ʊ, æ, θ, o},
 {p, z, ʃ, k, θ, t, d, g, ð, f, s, n, ŋ, v, m, l, r, w, ɔ, b}}
```

- These next two functions give us the edges for the graph. Two sounds are connected by an edge if the distance between them is less than r = 1.5.

In[525]:=
```
takexth1[x_] :=
 Take[DeleteCases[DeleteDuplicates[Partition[Flatten[Riffle[ipaenghood[[#]],
                {ipaenglishpointstosounds[[#]]}] & /@ Range[1, 36]], 2]],
      {n_, n_}][[##]][[1]] & /@ Range[1, 412], {x}]
```

In[526]:=
```
takexth2[x_] :=
 Take[DeleteCases[DeleteDuplicates[Partition[Flatten[Riffle[ipaenghood[[#]],
                {ipaenglishpointstosounds[[#]]}] & /@ Range[1, 36]], 2]],
      {n_, n_}][[##]][[2]] & /@ Range[1, 412], {x}]
```

- Finally, here is the graph. This is not persistent homology, but it is still a useful visualization. The points in this graph are sounds, and two of them are connected if the distance between them (in the embedding) is < 1.5.

In[529]:=
```
Graph[Partition[Values[ipaenglishpointstosounds], 1],
 Flatten[takexth1[#] ⟷ takexth2[#] & /@ Range[1, 412]], VertexLabels → Flatten[
   Evaluate[# → First[#] & /@ Partition[Values[ipaenglishpointstosounds], 1]]]]
```

Out[529]=