THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GENOME-METAGENOME SIMILARITY GRAPH: A NOVEL DATA STRUCTURE FOR CONTAINMENT-BASED METAGENOME COMPARISON

Isaac Thomas
SPRING 2021

A thesis
submitted in partial fulfillment
of the requirements
for the baccalaureate degree
in Computer Science
with honors in Computer Science

Reviewed and Approved* by the following:

David Koslicki
Professor of Computer Science & Engineering
Thesis Supervisor

John Hannan
Professor of Computer Science & Engineering
Associate Department Head
Honors Adviser

*Signatures are on file in the Schreyer Honors College.

Abstract

Metagenomics is a new and promising field for the analysis of microbial communities, particularly the human microbiome. With standard techniques of metagenomic profiling, one can reveal the influential residents within any sampled microbial community of interest, paving the way for forms of further exploration. Metagenome comparison is such an avenue of interest, and to combat the computational infeasibility of alignment-based methods for large samples, k-mer based methods have been developed to estimate famous measurements of metagenome similarity, such as the Jaccard Index. However, such methods do not reveal enough information about which organisms make two metagenomes similar. Such information is important since the notion of metagenome similarity between two metagenomes implies the presence of common organisms. To address this shortcoming, we propose a novel graph structure, a Genome-Metagenome Similarity Graph (GMSG), for modeling containments between various organisms and two metagenomes of interest. We then explore various functions of these containments to compute measurements of similarity and distance between two metagenomes.

Table of Contents

List of Figures

List of Definitions

List of Theorems

Acknowledgements

I'm grateful to have the people in my life whose guidance and support culminated in the creation of this thesis. I'd like to thank my thesis supervisor, Dr. David Koslicki, for giving me the creative freedom and necessary feedback to develop and implement the research presented in this paper, all while building my interest and appreciation for this field of study along the way. Additionally, I'd like to thank my honors advisor, Dr. John Hannan, for his academic guidance and throughout my undergraduate career, and his helpful feedback on the numerous iterations of this paper. I'd also like to thank Shaopeng Liu, a graduate student and colleague of mine in Koslicki Lab, for helping me overcome many obstacles in implementing the methods and simulations described in this thesis. Finally, I'd like to thank my parents, Anil and Sujitha Thomas, for constantly encouraging me to follow my interests and push my limits.

Introduction

0.1 Thesis Goals

This thesis serves two purposes: to build upon the previous work in k-mer based methods for metagenomic similarity by developing a new data structure for computing metagenomic similarity, and to compare the behavior of its derived similarity indices/distance metrics to previously created indices in order to measure its effectiveness. The data structure we develop appeals to the motivation of metagenomic similarity – namely, the presence of common organisms – by modeling genome-metagenome containment relationships and preserving such information when computing similarity, while other purely k-mer based methods do not.

0.2 Thesis Outline

Chapter 1 will give an overview of the field of metagenomics, why its techniques are superior to those of standard genomics for studying microbial communities, and different classes of methods – namely, allignment-based methods and k-mer based methods –for metagenomic analysis and their properties. Since our novel method lies in the latter of these categories, chapter 2 will focus on k-mer based methods by reviewing famous indices of similarity and containment used for metagenomic analysis, along with well-known algorithms and their implementations for computing such indices. Chapter 3 will build upon the previous work in chapter 2 to develop a novel data structure, a Genome-Metagenome Similarity Graph (GMSG), for modeling genome-metagenome containment relationships and computing metagenomic similarity/distance, with the intent of preserving information on organism presence which purely k-mer based methods do not. Chapter 4 will present results on the behavior of GMSG-based indices/metrics in comparison to the Jaccard index for simulated genomes of varying similarities, and an interpretation of these results will be discussed, along with avenues of further research.

Chapter 1

Background

1.1 Microbiomes and Their Importance

Some of Earth's most influential communities of life escape the gaze of the human eye. Their inhabitants are bacteria, fungi, and many other microorganisms – referred to as *microbes* – which exert tremendous influence within their habitats. In particular, microbes comprise powerful communities which reside on and within ourselves. The human body houses numerous complex microbial communities which interface with tissue and contribute to the proper functionality of central organs. Furthermore, abnormalities in microbial abundances may adversely impact organ functionality. For instance, dominance of *Staphylococcus aureus* in the skin's microbial communities has been associated with symptom severity among atopic dermatitis patients [1]. Internally, a similar tale arises: small intestine bacterial overgrowth (SIBO) may play a role in the development of Irritable Bowel Syndrome [2]. Although there is debate in the community over whether such microbial irregularities are the cause or an effect of such disorders, there is no doubt that further study of microbial analysis with modern sequencing and analysis techniques will yield useful insights for potential treatments. Therefore, it is important to increase our understanding of the relationship between the human body and its microbial tenants, which could lead to breakthroughs in treating diseases with potential microbial causes.

1.2 Shortcomings of Standard Genomics Techniques

Standard genomic analysis techniques of petri dish culturing and subsequent analysis rely on the pure-culture paradigm – the notion that any microbe in a community can be cultured individually. This paradigm falls short in the study of microbial communities. For instance, an estimated 0.1-1% of living bacteria present in soil is culturable [3]. Even if one were to study only the culturable microbes of a community, it would be very easy to miss the organisms which make the greatest impact on the community, which defeats the purpose of studying the community at all. The key insight is that, within a community, individual organism growth and development is the product of complex interactions between all its members. For instance, organism x in a community might produce byproducts that organism y consumes to thrive. In turn, organism y might produce byproducts which deactivate compounds produced by some organism z which are toxic to organism x . Even if we narrow our scope of analysis to a single organism in the community, we can end up with a taxonomically and functionally significant amount of intra-species genetic diversity. Thompson et al. found at least 1000 distinct genotypes of *Vibrio splendidus* in a sample of a coastal bacterial community [4]. It's infeasible to reproduce such copious genetic variation by culturing a single organism from this species in isolation. From these observations, a complex reality emerges: a microbial community is far greater than the sum of its parts, so studying its components in isolation will not yield any significant insights about its structure or functionality. Thus, one needs a set of techniques to study microbial communities not just as sets of isolated organisms, but as whole entities.

1.3 What is Metagenomics?

The field of metagenomics addresses the shortcomings of genomics by sampling the DNA of microbial communities directly from their natural environment instead. We refer to such communities as *metagenomes* and their representative sets of DNA reads as *metagenomic samples*. By analyzing metagenomic samples, researchers can retain unculturable organisms and capture the genetic diversity missing from an equivalent lab-cultivated sample. The goal of sampling from a metagenome is to obtain genetic material representing all organisms in the sample accurately with respect to presence and quantity, as this information is crucial for sound analyses in later steps of metagenomic pipelines. Performing this task requires care and attention to the type of metagenome, which could influence the quality and quantity of genetic information that can be extracted. For instance, some communities, such as those found in the human breast milk microbiome, yield low amounts of DNA depending on the extraction technique used, as was found by Douglas et al. in the human breast milk microbiome. [5].

1.4 Analyzing Metagenomic Samples

Once we have sequenced the genetic material of our sample, there are multiple paths we could pursue for our analysis. We could, for instance, assemble the sample reads into a novel genome to discover a novel organism. This process, known as metagenomic assembly, can aid in the discovery of previously unknown organisms in metagenomes. But one might want to pursue an easier task – determining which known organisms the sample contains, as well as the abundance of each organism. This information forms what is known as a taxonomic profile. There are two main ways of computing a taxonomic profile for a metagenomic sample. Alignment-based profiling involves mapping individual reads to genomes. A sample read is assigned to the genome with which the highest-quality alignment results, and the organism corresponding to that genome is deemed present in the sample. This method, while more accurate, is slow due to the time complexities of sequence alignment algorithms. k-mer based profiling methods, which produce small read fragments of length k and operate on a sample of those fragments, are less accurate but significantly faster for large metagenomes. We focus on the latter of these techniques throughout this paper, as they are the foundation of our novel method.

1.5 Metagenomic Similarity

Taxonomic profiles serve as a useful stepping stone via which k-mer based methods can extract further insight from metagenomes. One crucial form of insight involves not just one metagenome, but its relationship to another. Suppose we are given respective samples S_A and S_B of two metagenomes A and B, respectively. It is useful to compute a measurement of similarity/distance between S_A and S_B. To solve this problem, one aspires to create some sort of distance function - namely, a metric on the set of metagenomic samples, as this function captures the notion of spatial distance using the attributes stored in each sample's taxonomic profile. When a metric is not available, one can use a similarity index, a function with fewer mathematical properties but still helpful if constructed carefully.

1.5.1 Similarity Indices & Distance Metrics

We review metrics and similarity indices more rigorously. We first present the definition of a metric.

Definition 1.1 (*metric*)**.** *Let X be a set. A metric on X is a function $d : X \times X \to \mathbb{R}$ with the following properties:*

1. $d(x, y) \geq 0$ *for any $x, y \in X$ (d is nonnegative).*
2. $d(x, y) = 0$ *if and only if $x = y$ for any $x, y \in X$ (d separates only distinct points).*
3. $d(x, y) = d(y, x)$ *for any $x, y \in X$ (d is symmetric).*
4. $d(x, y) + d(y, z) \geq d(x, z)$ *for all $x, y, z \in X$ (d obeys the triangle inequality).*

In the physical world, two points can't have a negative distance, only at least zero distance between them; the distance cannot be zero if the points are different, as it doesn't make sense for two different physical locations to be indistinguishable. Switching the order of the points shouldn't change the distance between them. And finally, it should never require more distance to cut across from one point directly to the other than to visit a point in between (or anywhere else) along the way. By fulfilling these four properties, a metric captures a notion of spatial distance between its inputs. Because it's often useful to augment data with a notion of distance that obeys the rules of physical space, metrics are often preferred for solving clustering problems. However, situations my arisee where a metric is not available. In this case, one can resort to a similarity index. A similarity index does not have a rigorous definition in the traditional sense, so we do not present one here; instead, we present desired properties of similarity indices that make it easy to quantify how many things in common two input sets share. A similarity index is a symmetric function which usually takes two sets of elements as input and outputs a real number within some interval, usually $[0, 1]$. An effective similarity index should output 0 when the two sets are completely disjoint, 1 when the sets are identical, and should increase as the input sets become more similar. Using a similarity index, one can construct a dissimilarity index by taking the complement of the similarity index. This way, one can construct a distance function of sorts, though it will lack some of the desirable properties of a metric.

To design a helpful metagenomic similarity index or metric, one must consider how to represent its two input metagenomes. Sets of k-mers are a popular choice, as they are compatible with well-known set similarity measurements like the Jaccard Index [6]. But there are other avenues one can pursue. For instance, we can add another layer to the k-mer representation and compute containments of organisms in each metagenome, then arrange these containments into a representation of each metagenome. We discuss this approach in more detail in chapter 3. We preface this discussion with a review of popular indices of similarity and containment jointly used with k-mer representations of metagenomes – namely, the Jaccard index and containment index – in the next chapter.

Chapter 2

Previous Work

2.1 The Jaccard Index

The Jaccard index, created by Paul Jaccard in 1912 [6], measures the similarity between two sets by comparing the size of their intersection (common elements) to the size of their union (all elements).

Definition 2.1 (*Jaccard Index*)**.** *Let S_1 and S_2 be finite sets. The Jaccard index $J(S_1, S_2)$ of S_1 and S_2 is defined as follows:*

$$J(S_1, S_2) = J(S_2, S_1) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

Intuitively, $J(S_1, S_2)$ increases as the number of common elements of S_1 and S_2 increases, and $J(S_1, S_2)$ decreases when $|S_1|, |S_2|$ increase, as the common elements become a smaller fraction of the total number of elements in S_1 or S_2. Naturally, we could consider the one-compliment of the Jaccard index, known as the Jaccard distance, as defined below.

Definition 2.2 (*Jaccard distance*)**.** *Let S_1 and S_2 be finite sets. The Jaccard Index $d_J(S_1, S_2)$ between S_1 and S_2 is defined as follows:*

$$d_J(S_1, S_2) = 1 - J(S_2, S_1) = 1 - \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \frac{|S_1 \triangle S_2|}{|S_1 \cup S_2|}$$
$$d_J(\emptyset, \emptyset) = 1$$

Where $A \triangle B$ denotes the symmetric difference of A and B (all elements in one set but not the other). Interestingly enough, the Jaccard distance is a metric on the space of finite sets. To prove this is beyond the scope of this thesis, but one can do so by first proving that the size of symmetric difference $d'(A, B) = |A \triangle B|$ is a metric on the same space and converting d' into d_J via the metric-preserving Steinhaus transform [7].

A simple but crucial insight is that $J(S_1, S_2)$ is function of two *sets*, as is $d_J(S_1, S_2)$. That is, all of the data involved is binary, since a given element can be only either present in a set or absent from it. Thus, the Jaccard Index can measure the similarity between any two entities which can be *binarized* – that is, represented with data on presence or absence of constituent elements. For instance, one can apply the Jaccard index to measuring the similarity of text documents, as one can represent such entities by sets of sentence fragments present in each document. We can extend this technique of fragment-based representation, known as *shingling*, to reads from genomes and metagenomes. For a set of reads S, simply pick a value of k and let $S^{(k)}$ be the set of all unique k-mers in S. Then indeed, $S^{(k)}$ can be represented with binary data, and if we repeat this process for two metagenomic samples A and B to obtain $A^{(k)}$ and $B^{(k)}$, then we can compute $J(A^{(k)}, B^{(k)})$ to measure the similarity between A and B. We now explore ways of computing $J(A^{(k)}, B^{(k)})$, from brute force methods to fast estimates based on random subsampling.

2.1.1 Computing $J(A^{(k)}, B^{(k)})$

The Naive Approach

First, a naive approach for computing $J(A^{(k)}, B^{(k)})$ would involve computing $A^{(k)} \cap B^{(k)}$ and ultimately $|A^{(k)} \cap B^{(k)}|$. We note for a given value of k, there are up to 4^k possible k-mers present in $A^{(k)}$ or $B^{(k)}$. We could sort $A^{(k)}$ and $B^{(k)}$ and compare the ordered sets to find the intersection, but this immediately has time complexity $O(m \log m)$, where $m = \max(|A^{(k)}|, |B^{(k)}|)$ due to the sorting overhead. Alternatively, we could use a hash table to check for memberships of k-mers in both sets. This approach has time complexity $O(|A^{(k)}| + |B^{(k)}|)$ and space complexity $O(\min(|A^{(k)}|, |B^{(k)}|))$. When we use a practical k-mer size like $k = 21$, either set could contain up to 4^{21} k-mers, which makes computing set intersections alone infeasible via brute force. Thus, for large datasets, there is simply too much input to efficiently compute $J(A^{(k)}, B^{(k)})$ using all k-mers from both sets.

A Faster Approximation: The MinHash Algorithm

If using all of our k-mers is too slow, we could instead estimate $J(A^{(k)}, B^{(k)})$ with a smaller input – namely, a random sample of $A^{(k)}$, $B^{(k)}$, and $A^{(k)} \cup B^{(k)}$. This is exactly the premise of the MinHash algorithm from Broder [8]. The MinHash algorithm takes a random sample from $A^{(k)} \cup B^{(k)}$, using a process called *sketching*. Sketching a set U is done with a hash function $h : U \to \mathbb{Z}$, which permutes the elements of U and assigns them randomly to output values. A crucial requirement is that h is *min-wise independent* – that is, any element in U is equally likely to map to the minimum value of h. If this is true, and if we assume that h has no collisions, then picking the minimum output of h is equivalent to drawing an element from U uniformly at random. The property of min-wise independence can be extended to second-min-wise independence, third-min-wise independence, etc. so that we can randomly sample p elements of U to produce a random sample or *sketch* of U, denoted by $S_p(U)$, by picking the p smallest outputs of h. Naturally, this method of sampling is known as a *bottom sketch*. Using this technique, we can produce a sketch of $A^{(k)}$, $B^{(k)}$, and $A^{(k)} \cup B^{(k)}$ each containing p elements. It then follows from Broder [8] that

$$E_J = \frac{|S_p(A^{(k)} \cup B^{(k)}) \cap S_p(A^{(k)}) \cap S_p(B^{(k)})|}{|S_p(A^{(k)} \cup B^{(k)})|}$$

is an unbiased estimator of $J(A^{(k)}, B^{(k)})$. After sketching, $A^{(k)}$, $B^{(k)}$, and $A^{(k)} \cup B^{(k)}$, the MinHash algorithm computes E_J to produce an estimate for $J(A^{(k)}, B^{(k)})$. Because MinHash samples a fixed number of p k-mers from $A^{(k)}$ and $B^{(k)}$ and computes E_J with sketches of size p, MinHash takes $O(p \log p)$, which is constant with respect to the sizes of our inputs A and B.

2.1.2 MinHash Implementations

There are two well-known implementations of the MinHash algorithm. Ondov et. al developed Mash, a C++ library which implements the MinHash algorithm with the same bottom sketching technique as previously described [9]. For two sets of reads A and B, Mash first slides a k-length window along the reads of A and B to produce k-mers for each, then hashes the k-mers and sketches A and B to produce $S_p(A^{(k)})$ and $S_p(B^{(k)})$. Mash then computes $J(A^{(k)}, B^{(k)})$ using the estimate given by Broder [8] . Sourmash, a Python library developed by Pierce et al., uses a variation on MinHash known as scaled MinHash, which computes sketches of A and B whose sizes depend on $|A|$ and $|B|$, respectively [10]. This way, large sets and small sets are sketched at the same compression rate, instead of the larger set losing a higher percentage of its information. Sourmash then computes an all-pairs Jaccard matrix between the sets of reads specified (two, in this case), where each entry is the estimate from Broder [8]. Sourmash then takes the smallest value in this matrix as its estimate of $J(A^{(k)}, B^{(k)})$.

2.2 Containment Index

The *containment index* of set S_1 in S_2 describes the fraction of S_1 which is contained in S_2, as defined below:

Definition 2.3 (*Containment Index*)**.** *Let S_1 and S_2 be finite sets. The containment index of S_1 in S_2 is defined as*

$$C(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1|}$$

One can see that the containment index replaces denominator of $J(S_1, S_2)$ with just $|S_1|$. Intuitively, we are now measuring the percentage of S_1 "covered" by the elements in S_2. From this, one can deduce that S_1 can cover a different number of elements in S_2 than S_2 can cover in S_1. This intuition yields an important point: unlike $J(S_1, S_2)$, $C(S_1, S_2)$ is not symmetric, for it is not necessarily the case that $C(S_1, S_2) = C(S_2, S_1)$. Thus, unlike the Jaccard similarity, the complement $1 - C$ of the containment is not a metric. Despite these irreconciliable differences, the containment index and Jaccard index still seem related at a glance. A more thorough investigation reveals that this is the case – and as we are about to see, this relation yields a way to produce higher-quality estimates of the Jaccard Index.

2.2.1 Better Estimates of $J(A^{(k)}, B^{(k)})$: Containment MinHash

While MinHash can estimate $J(A^{(k)}, B^{(k)})$ quickly, Koslicki et al. found that the variance of the MinHash estimate grows exponentially as the true Jaccard index of $A^{(k)}$ and $B^{(k)}$ becomes smaller [11]. Since the true Jaccard index is influenced by $|A^{(k)} \cup B^{(k)}|$ (and thus $|A^{(k)}|$ and $|A^{(k)}|$), it follows that the MinHash estimate of $J(A^{(k)}, B^{(k)})$ can attain high variance when $|A^{(k)}| << |B^{(k)}|$. Koslicki et al. addressed this issue by developing the containment MinHash algorithm, which estimates $J(A^{(k)}, B^{(k)})$ using $C(A^{(k)}, B^{(k)})$. The premise of containment MinHash is analogous to that of MinHash – namely, that

$$E_C = \frac{|S_p(A^{(k)}) \cap S_p(B^{(k)})|}{|S_p(A^{(k)})|}$$

is an unbiased estimator of $C(A^{(k)}, B^{(k)})$ [8]. Once we estimate $C(A^{(k)}, B^{(k)})$ this way, we can convert it into an estimate of $J(A^{(k)}, B^{(k)})$ by applying the inclusion-exclusion principle and observing that $|A^{(k)} \cap B^{(k)}| = \frac{|A^{(k)} \cap B^{(k)}|}{|A^{(k)}|}|A^{(k)}| = C(A^{(k)}, B^{(k)})|A^{(k)}|$. We have

$$\begin{aligned} J(A^{(k)}, B^{(k)}) &= \frac{|A^{(k)} \cap B^{(k)}|}{|A^{(k)} \cup B^{(k)}|} = \frac{|A^{(k)} \cap B^{(k)}|}{|A^{(k)}| + |B^{(k)}| - |A^{(k)} \cap A^{(k)}|} \\ &= \frac{|A^{(k)} \cap B^{(k)}|}{|A^{(k)}| + |B^{(k)}| - C(A^{(k)}, B^{(k)})|A^{(k)}|} \\ &= \frac{C(A^{(k)}, B^{(k)})|A^{(k)}|}{|A^{(k)}| + |B^{(k)}| - C(A^{(k)}, B^{(k)})|A^{(k)}|} \end{aligned}$$

This estimate of $J(A^{(k)}, A^{(k)})$ has a lower variance than the estimate produced by MinHash [11]. Additionally, we only need the sizes of $A^{(k)}$ and $B^{(k)}$ and not of $A^{(k)} \cup B^{(k)}$.

2.2.2 Containment MinHash Implementation

Koslicki et al. developed CMash [11], a Python implementation of the containment MinHash algorithm. To estimate $C(K_A, K_B)$, CMash uses a bloom filter, a data structure widely used for fast set membership testing. A bloom filter F [12] consists of an array contain w bits, along with k hash functions $g_1...g_k$, each of which maps an input element to a position in the array of F. To add an element to F, we set each position $g_1(x)...g_k(x)$ to 1, which denotes that x is now present in the filter. To query for the presence of an element a in F, we check if each and every position $g_1(a)...g_k(a)$ is set to 1. If so, F reports that a is in the filter. If not, F reports that a is not in the filter. Because some inputs can collide with others via $g_1...g_k$, it's possible that F reports false positives. For n queries, the rate at which this error occurs is approximately

$$E = (1 - e^{-\frac{nk}{w}})^k.$$

CMash populates a bloom filter with the elements of $A^{(k)}$ and queries the bloom filter with the elements in the sketch $S_p(A^{(k)})$ of $A^{(k)}$. The fraction of queries found in the bloom filter estimates $C(A^{(k)}, B^{(k)})$, with a small error rate due to potential false positives reported by the filter. Of course, CMash also has functionality to estimate $J(A^{(k)}, B^{(k)})$ from $C(A^{(k)}, B^{(k)})$.

2.3 Shortcomings of k-mer based methods

While purely k-mer based methods such as Mash and Sourmash can quickly compute estimates of set similarity indices such as the Jaccard index, they provide no information about the organisms likely present in both metagenomes which contribute to similarity - they only give information about which k-mers do so. Since the notion of similarity between two metagenomes is motivated by the presence of common constituent organisms and not just DNA fragments, it would be helpful to represent two metagenomes by the extent to which certain organisms are present within them, then use this information to compute an index of similarity/distance. Additionally, k-mer based methods are sensitive to errors from sequencing machines, which can result in common k-mers being mutated and ultimately placed outside the intersection of sets in question. This phenomenon could artificially lower indices such as Jaccard similarity. By taking a containment-based approach to metagenomic similarity, one can reflect organism presence and preserve similarity in the presence of noise by checking related organisms which could contain mutated but originally shared k-mers. We present an approach which fulfills these goals in chapter 3.

Chapter 3

Methods

3.1 Representing Metagenomes with Containment Index

Any containment-based index for similarity/difference relies on the notion of a containment-based representation of a metagenome. We first formulate a way to represent metagenomes in this manner, so that we are equipped with information regarding organism presence and absence, which we can feed into the similarity and distance formulae that we present in later sections. Consider two metagenomic samples A and B, and suppose we are equipped with a reference database of n reference genomes $D = \{R_1, R_2, ..., R_{n-1}, R_n\}$. Additionally, pick $k \in \mathbb{Z}^+$ as our k-mer length. We first assume that D is *complete* – in other words, that every organism with genetic material in A and B corresponds to some genome in D. Given k, we can then represent A by the ordered tuple $(a_1^{(k)}, a_2^{(k)}, ..., a_n^{(k)})$, where $a_i^{(k)} = C(R_i^{(k)}, A^{(k)})$. Similarly, we represent B by the ordered tuple $(b_1^{(k)}, b_2^{(k)}, ..., b_n^{(k)})$, where $b_i^{(k)} = C(R_i^{(k)}, B^{(k)})$. Intuitively, we characterize A and B by the extent to which each organism in our database is genetically present in each of A and B. It's important to note that since the accuracy of our representations of A and B depends on prior information stored in D, the containment-based representation is useful only for characterizing well-studied microbial communities. For metagenomes with ill-known compositions, this representation could miss important microbial constituents whose genomes are not in D. With A and B represented in this manner, it may just seem like we have two ordered tuples with which we don't know what to do. But we have more than that – each coordinate in $(a_1, a_2, ..., a_n)$ describes an asymmetric relationship between A and some genome in D, and each coordinate in $(b_1, b_2, ..., b_n)$ does the same for B. It just so happens that directed graphs capture such asymmetric relations, so we present one that relates A and B.

3.2 Genome-Metagenome Similarity Graph (GMSG)

Now that we have established the notion of a containment-based representation of A and B with respect to D, we now define a Genome-Metagenome Similarity Graph (GMSG) in terms of A, B and D.

Definition 3.1 (*Genome-Metagenome Similarity Graph*)**.** *For two metagenomic samples A and B, a database of reference genomes $D = \{R_1, R_2, ..., R_{n-1}, R_n\}$, and $k \in \mathbb{Z}^+$, we define a Genome-Metagenome Similarity Graph (GMSG) $G = (A, B, D, k)$ via the following construction:*

1. *Let A, B and all $R_i \in D$ be nodes.*
2. *Place a directed edge from A to each $R_i \in D$, with weight $a_i^{(k)} = C(R_i^{(k)}, A^{(k)})$.*
3. *Place a directed edge from each $R_i \in D$ to B, with weight $b_i^{(k)} = C(R_i^{(k)}, B^{(k)})$.*

The result is a GMSG on A, B with respect to D and k. It's intuitive to capture the asymmetric containment relationships between each R_i, A, and B with directed edges. That being said, it might seem counterintuitive to have edges going from B to all R_i, instead of in the opposite direction. We've done so in order to create the flow similarity index (defined in next section), which follows almost immediately from the maximum flow across the GMSG.

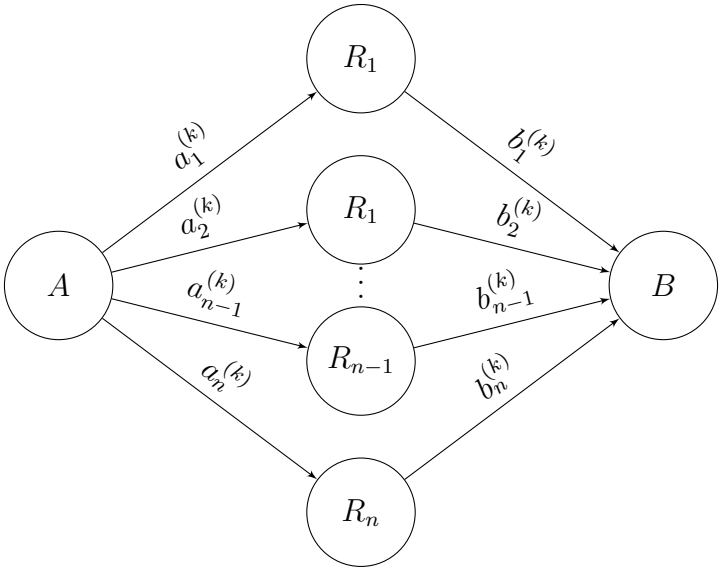


Figure 3.1: A Genome-Metagenome Similarity Graph with on A, B, with respect to D, and k.

As shown in figure 3.1, each edge from A quantifies the percent of genetic material from each R_i is inside A via $C(R_i^{(k)}, A^{(k)})$, and each edge to B quantifies how much genetic material from each R_i is inside B via $C(R_i^{(k)}, B^{(k)})$. Since metagenomic similarity is motivated by the presence of common organisms, we'd like to characterize the similarity of A and B by whether the genetic material of a particular organism in D is present in both samples. For each organism $R_i \in D$, the extent to which this is the case depends on the lowest possible containment of R_i in each of A and B. If we sum up the value used for the lowest containment of each R_i in both samples, then we arrive at a sensible but unscaled measurement of similarity. Interestingly enough, this is exactly the maximum flow of the GMSG, which we prove here.

Theorem 3.1 (*Maximum flow across GMSG*)**.** *Let A and B be two metagenomic samples, $D = \{R_1, R_2, ..., R_{n-1}, R_n\}$ a database of reference genomes, $k \in \mathbb{Z}^+$. Let $G = (A, B, D, k)$ be a GMSG on A, B with respect to D and k. Then the maximum flow across G is given by*

$$f(G) = \sum_{i=1}^{|D|} \min\{C(R_i^{(k)}, A^{(k)}), C(R_i^{(k)}, B^{(k)})\}$$

Proof. Because of how G is defined, there are three nodes in each path from A to B, and each R_i is a member of exactly one path from A to B. Since there are $|D|$ nodes R_i, there are $|D|$ disjoint paths from A to B, each containing exactly one node from D. Since all paths are disjoint, the maximum flow through one path does not affect the maximum flow through any other path, so we can simply sum the maximum flows through each path to find the maximum flow. Consider $i \in \{1...|D|\}$. For the path containing R_i, the maximum flow through this path is bounded by the minimum weight of the two edges in the path. These weights are $a_i^{(k)} = C(R_i^{(k)}, A^{(k)})$ and $b_i^{(k)} = C(R_i^{(k)}, B^{(k)})$. Thus, $\min\{a_i^{(k)}, b_i^{(k)}\}$ gives the maximum flow through the path containing R_i. Combining the maximum flows through all $|D|$ paths yields

$$\begin{aligned} f(G) &= \min\{a_1^{(k)}, b_1^{(k)}\} + \min\{a_2^{(k)}, b_2^{(k)}\} + \cdots + \min\{a_{|D|-1}^{(k)}, b_{|D|-1}^{(k)}\} + \min\{a_{|D|}^{(k)}, b_{|D|}^{(k)}\} \\ &= \sum_{i=1}^{|D|} \min\{a_i^{(k)}, b_i^{(k)}\} \end{aligned}$$

as desired. □

Note that $0 \leq f(G) \leq |D|$, since each path through G can have a flow of at most 1, since any edge weight in the $|D|$ paths of G is at most 1. We can normalize $f(G)$ to better fit the properties of a similarity index by mapping to $[0, 1]$, We do so in the following definition. -

3.3 Computing Similarity & Distance with GMSG

3.3.1 GMSG Flow Similarity

To define the flow similarity between A and B, we simply divide f by the maximum possible max flow across any G – namely, $|D|$ when all containment edge weights are 1.

Definition 3.2 (*GMSG Flow Similarity*)**.** *Let A and B be two metagenomic samples, $D = \{R_1, ..., R_n\}$ a database of reference genomes, and $k \in \mathbb{Z}^+$ our k-mer size. The flow similarity between A and B with respect to D and k is the normalized version of $f(G)$, where $G = (A, B, D, k)$:*

$$\Phi_D^{(k)}(A, B) = \frac{f(G)}{|D|} = \frac{1}{|D|} \sum_{i=1}^{|D|} \min\{C(R_i^{(k)}, A^{(k)}), C(R_i^{(k)}, B^{(k)})\}$$

As always, it's useful to analyze Φ in extreme cases of A and B. However, it's important to note that a solely theoretical case analysis, while still helpful, will not accurately reflect the behavior of Φ since we must overlook a crucial parameter whose effects precede anything we have defined so far: sequencing coverage. Loosely, sequencing coverage is the extent to which the reads generated by a sequencing machine cover the genetic material being sequenced. If our sequencing coverage is sufficient enough to cover all of the genetic material involved, then the presence of an organism will likely correspond to all its genetic material being present in a sample of such genetic material. If our coverage is too low, then it's possible that we observe only part of the genome of an organism despite its full presence in our sample. A theoretical standpoint won't make any assumptions about sequencing coverage, which makes it possible for any amount of DNA from a particular genome

to be present inside a metagenomic sample. A more realistic approach will make assumptions about sequencing coverage, as we will describe soon. For these reasons, we take on both a purely theoretical and real-world perspective when analyzing the behavior of Φ.

A Purely Theoretical Case Analysis of Φ

As mentioned, a theoretical analysis of Φ makes no assumptions about sequencing coverage – thus, a genome could be contained in A or B to any extent. Such an unconstrained analysis reveals some odd behavior in extreme cases. Fix D and k. When $A^{(k)} = B^{(k)}$, it's possible that $C(R_i^{(k)}, A^{(k)}) < 1$ for some $R_i \in D$, which would mean that $\Phi_D^{(k)}(A, B) < 1$ despite the fact that $A^{(k)} = B^{(k)}$. This violates one of the desirable properties of similarity indices mentioned in chapter 1. When $A^{(k)} \cap B^{(k)} = \emptyset$, it's still possible that $R_i^{(k)} \cap A^{(k)} \neq \emptyset$ and $R_i^{(k)} \cap B^{(k)} \neq \emptyset$. This implies that $\min\{C(R_i^{(k)}, A^{(k)}), C(R_i^{(k)}, B^{(k)})\} > 0$ which means that $\Phi_D^{(k)}(A, B) > 0$ despite the fact that $A^{(k)}$ and $B^{(k)}$ are completely disjoint. This violates yet another desirable property of similarity indices mentioned in chapter 1. While this behavior of Φ seems absurd, a more real-world analysis shows that proper sequencing coverage makes the both scenarios very unlikely, and that the second scenario may even be useful when sequencing coverage is insufficient.

A More Real-World Case Analysis of Φ

Taking into account real-world parameters like sequencing coverage sheds new light on the behavior of Φ. Fix D and k again, and suppose our sequencing coverage of the machine used to produce A and B is sufficient. If $A^{(k)} = B^{(k)}$, then due to our sequencing coverage, if an organism x is in the metagenome of A, then given the genome $R_i \in D$ corresponding to x, $R_i^{(k)}$ is completely contained in $A^{(k)}$. Thus, $C(R_i^{(k)}, A^{(k)}) = 1$, and since $A^{(k)} = B^{(k)}$, we must have that $C(R_i^{(k)}, B^{(k)}) = 1$ as well. Thus the k-mers corresponding to any organism present in the metagenome of A will be completely contained in both $A^{(k)}$ and $B^{(k)}$, so it follows that $C(R_i^{(k)}, A^{(k)}) = C(R_i^{(k)}, B^{(k)}) = 1$ for all R_i contained in A and B. If we choose the genomes in D wisely, so that all reference organisms are contained in A and B (hence the importance of comparing well-studied communities), then it follows that $C(R_i^{(k)}, A^{(k)}) = C(R_i^{(k)}, B^{(k)}) = 1$ for all R_i, which gives us $\Phi_D^{(k)}(A, B) = \frac{|D|}{|D|} = 1$. Conversely, when $\Phi_D^{(k)}(A, B) = 1$, we have that $\min\{C(R_i^{(k)}, A^{(k)}), C(R_i^{(k)}, B^{(k)})\} = 1$, which implies that $C(R_i^{(k)}, A^{(k)}) = C(R_i^{(k)}, B^{(k)}) = 1$. Thus, any $R_i^{(k)}$ is contained to the same extent in $A^{(k)}$ and $B^{(k)}$, and since D is complete, it must be the case that $A^{(k)} = B^{(k)}$. Thus, $A^{(k)} = B^{(k)} \Leftrightarrow \Phi_D^{(k)}(A, B) = 1$.

Now suppose that $A^{(k)} \cap B^{(k)} = \emptyset$. Then due to our sufficient sequencing coverage, if an organism y were present in the metagenomes of both A and B, then $R_j^{(k)}$ corresponding to y would be completely contained in both $A^{(k)}$ and $B^{(k)}$, which contradicts our assumption that $A^{(k)}$ and $B^{(k)}$ are disjoint. So with sufficient sequencing coverage, we can conclude that $A^{(k)} \cap B^{(k)} = \emptyset$ implies that the metagenomes of A and B share no organisms in common, which implies that at least one of $C(R_i^{(k)}, B^{(k)}), C(R_i^{(k)}, B^{(k)})$ is zero for all i. This means that $\Phi_D^{(k)}(A, B) = 0$. Conversely, if $\Phi_D^{(k)}(A, B) = 0$, then we have that $C(R_i^{(k)}, A^{(k)}) = 0$ or $C(R_i^{(k)}, B^{(k)}) = 0$ for all i. Thus, $R_i^{(k)} \cap A^{(k)} = \emptyset$ or $R_i^{(k)} \cap B^{(k)} = \emptyset$ for all i. Since we're assuming that D is complete, it follows that $A^{(k)}$, $B^{(k)} \subseteq \bigcup_{i \in I} R_i^{(k)}$. Since we're assuming sufficient sequencing coverage, it

must be that every $R_i^{(k)}$ is fully contained in at most one of $A(k),\ B^{(k)}$. Thus, $A^{(k)}$ is a union of all $R_i^{(k)}$ contained in only $A^{(k)}$, and $B^{(k)}$ is a union of all $R_j^{(k)}$ contained in only $B^{(k)}$. This forms a partition of $A^{(k)} \cup B^{(k)}$, so it must be that $A^{(k)}$ and $B^{(k)}$ are disjoint. Thus, with sufficient sequencing coverage, $\Phi_D^{(k)}(A, B) = 0 \Leftrightarrow A^{(k)} \cap B^{(k)} = \emptyset$.

When coverage is not sufficient but $A^{(k)} \cap B^{(k)} = \emptyset$, we run into an interesting situation. Since our sequencing coverage cannot guarantee that each a present organism's genome is completely contained, there could exist $R_i \in D$ such that $R_i^{(k)} \cap A^{(k)} \neq \emptyset$ and $R_i^{(k)} \cap B^{(k)} \neq \emptyset$, but $(R_i^{(k)} \cap A^{(k)}) \cap (R_i^{(k)} \cap B^{(k)}) = \emptyset$. In other words, $R_i^{(k)}$ is contained in both $A^{(k)}$ and $B^{(k)}$, which each contain disjoint sections of $R_i^{(k)}$. It follows that $\Phi_D^{(k)}(A, B) > 0$ while clearly $J(A^{(k)}, B^{(k)}) = 0$. Thus, the flow similarity detects these "hints" of common organisms in $A^{(k)}$ and $B^{(k)}$, while the Jaccard index detects no similarity at all. Thus, the GMSG flow similarity could be useful in situations where sequencing coverage is low, and we explore this possibility in chapter 4.

3.3.2 Shortcomings of Flow Similarity

Despite their qualities, the GMSG flow similarity and its complement lack more desirable properties, such as those of a metric. As mentioned before, metrics may be more desirable due to the notion of physical distance they endow a given space with. One may like to create a function which captures the notion of spatial distance between the containment representations of A and B. Although there is much more progress to be made, we take the first step here by defining a well-known class of metrics, then normalizing these metrics for computing metagenomic similarity between A and B.

3.3.3 Distance metrics Using GMSG Weights

In addition to similarity indices, one can contemplate how to make a distance metric out of the containment-based representations of A and B. A good starting point is the fact that A and B are represented by ordered $|D|$-tuples. From there, we can then consider these two metagenomic samples as points in a $|D|$-dimensional space – namely, $[0,1]^{|D|}$. Then, we can normalize a well-known class of metrics, known as the l_p metrics, to produce a distance metric between metagenomic samples. We first define the l_p metrics.

Definition 3.3 (l_p metric)**.** *The l_p metric on $\mathbb{R}^n$ is defined as*

$$d_{l_p}(x,y) = \sum_{i=1}^{n}(|x_i - y_i|^p)^{1/p},\ p \geq 1$$

Due to the bounds of the containment index, any containment representation of A or B will lie in $[0,1]^{|D|}$. Thus, the distance between any two metagenomic samples will be at least zero and at most $|D|^{\frac{1}{p}}$, so we scale the l_p metric down by this factor and recognize D and k as auxiliary parameters to arrive at the GMSG l_p metric.

Definition 3.4 (GMSG l_p-metric)**.** *Let A and B be two metagenomic samples, $D = \{R_1, ..., R_n\}$ a database of reference genomes, and $k \in \mathbb{Z}^+$ our k-mer size. The GMSG l_p-metric is defined as*

$$\Psi_D^{(k,p)}(A,B) = \frac{1}{|D|^{\frac{1}{p}}}\sum_{i=1}^{|D|}(|C(R_i^{(k)}, A^{(k)}) - C(R_i^{(k)}, B^{(k)})|^p)^{1/p},\ p \geq 1$$

It's clear that Ψ is a metric on $[0,1]^{|D|}$, since we're simply scaling the expression for the conventional l_p distance metric by a constant factor, which preserves the metric properties. Naturally, when $p = 1$, we get a scaled version of the Manhattan distance metric, and when $p = 2$, we have a scaled standard euclidean metric. However, it's important to note that Ψ is not a metric on the space of finite sets as opposed to something like the Jaccard index. Ψ is a metric on the set of containment-based representations of sets of k-mers. These are $|D|$-tuples in $[0,1]^{|D|}$ where each coordinate is a value of the containment index.

Additionally, note how Ψ subtracts these containments in each term, while Φ takes the minimum containment in each term. So while Φ only changes if the minimum containment of a particular organism changes, Ψ can change if the two containments simply become closer together or farther apart. To explore these characteristics of Ψ further, we perform a case analysis of Ψ, which depends on our acknowledgement of sequencing coverage as with Φ.

A Purely Theoretical Case Analysis of Ψ

First, we make no assumptions about sequencing coverage. Fix D, k, and p. Suppose $A^{(k)} = B^{(k)}$. Let x be an organism in A and let $R_i \in D$ correspond to x. Then since $A^{(k)} = B^{(k)}$, it must be that $C(R_i^{(k)}, A^{(k)}) = C(R_i^{(k)}, B^{(k)})$ for all i $\{1...|D|\}$, so it follows that $C(R_i^{(k)}, A^{(k)}) = C(R_i^{(k)}, B^{(k)}) = 0$ for all i as well. Thus, $\Psi_D^{(k,p)}(A,B) = 0$. Conversely, when $\Psi_D^{(k,p)}(A,B) = 0$, it follows that $C(R_i^{(k)}, A^{(k)}) = C(R_i^{(k)}, B^{(k)})$ for all i, which implies that $A^{(k)} = B^{(k)}$ since D is

complete. So $\Psi_D^{(k,p)}(A,B) = 0 \Leftrightarrow A^{(k)} = B^{(k)}$. This behavior of Ψ is sensible, as it finds zero distance between two metagenomic samples whose k-mer sets are equal. Note how unlike Φ, one does not need sufficient sequencing coverage for this property to hold.

But when $A^{(k)} \cap B^{(k)} = \emptyset$, we run into behavior analogous to that of Φ. It's desirable that $\Psi_D^{(k,p)}(A,B) = 1$ as $A^{(k)}$ and $B^{(k)}$ are maximally different, but we show that this is not the case. Consider $R_i \in D$. It's possible that $R_i^{(k)} \cap A^{(k)} \neq \emptyset$ and $R_i^{(k)} \cap B^{(k)} \neq \emptyset$, but such that $|C(R_i^{(k)}, A^{(k)}) - C(R_i^{(k)}, B^{(k)})| < 1$. This means that $\Psi_D^{(k,p)} < 1$ despite the fact that $A^{(k)}$ and $B^{(k)}$ are disjoint. And although one might be hopeful that disjoint k-mer sets correspond to a maximal distance in real-world situations, this turns out to not be the case even when we make realistic assumptions about sequencing coverage, as we are about to show.

A More Real-World Case Analysis of Ψ

Suppose that we have sufficient sequencing coverage. Then any organism present in A has its genome covered by the reads in A (the same logic applies to any organism present in B). Since $\Psi_D^{(k,p)}(A,B) = 0 \Leftrightarrow A^{(k)} = B^{(k)}$ holds regardless, we suppose that $A^{(k)} \cap B^{(k)} = \emptyset$. Consider the case where W.L.O.G. R_1 is fully contained in A and not contained at all in B, but $C(R_i^{(k)}, A^{(k)}) = C(R_i^{(k)}, B^{(k)}) = 0$ for $i \in \{2...|D|\}$. Then we have

$$\Psi_D^{(k,p)}(A,B) = \frac{1}{|D|^{\frac{1}{p}}} \left(\left(\sum_{i=2}^{|D|} |0| \right) + |1-0|^{\frac{1}{p}} \right)^{\frac{1}{p}} = \frac{1}{|D|^{\frac{1}{p}}} < 1$$

Here, Ψ does not output the maximal possible distance despite the fact that $A^{(k)}$ and $B^{(k)}$ are disjoint. This issue arises from the "naivety" of our scaling factor of $|D|^{\frac{1}{p}}$; even if our representations of k-mer sets are effectively confined to a subspace of dimension $q < |D|$ in $[0,1]^{|D|}$, Ψ will still divide by $|D|^{\frac{1}{p}}$ even though these extra dimensions don't actually contribute anything to the distance (because neither A nor B contain the organisms corresponding to those dimensions). It appears that this issue deflates the output of Ψ for metagenomic samples containing a small percentage of common organisms out of those in D, and we will see if this could indeed be the case in chapter 4.

3.3.4 GMSG Implementation

We implemented the GMSG and the defined indices/metrics in Python using CMash [11], Pandas [13], and NumPy [14]. Given a k-mer size and files containing the paths to both the input metagenomic samples A and B and the reference genomes D, our GMSG implementation first uses CMash to compute all containment indices $C(R_i^{(k)}, A^{(k)})$ and $C(R_i^{(k)}, A^{(k)})$. We stored all $C(R_i^{(k)}, A^{(k)})$ and $C(R_i^{(k)}, B^{(k)})$ in two separate NumPy arrays, then used NumPy's vectorized array operations to compute $\Phi_D^{(k)}(A,B)$ and all $\Psi_D^{(k,p)}(A,B)$. The source code for the GMSG implementation can be found at https://github.com/KoslickiLab/GMSG. We use this implementation to explore the behavior of the GMSG indices and metrics in chapter 4.

Chapter 4

Results

4.1 Simulations on Microbial Communities

4.1.1 Behavior of GMSG-Based Indices/Metrics

In this section, we present and interpret simulation results regarding how the GMSG-based indices and metrics behave in comparison to k-mer based computations of similarity/distance.

Motivation

As previously mentioned, the notion of metagenomic similarity between two metagenomic samples A and B is motivated by the presence of common organisms. Therefore, a useful similarity index will sense the presence of common organisms in two communities (or lack thereof) and reflect this in the quantity it produces. To observe such behavior, one can construct A and B to be similar/dissimilar to each other by using the similarity encoded into taxonomic rankings of organisms. Organisms are grouped under taxonomic trees by the extent to which they are genetically related – thus, organisms under deeper subtrees, such as species or genera, will be more similar to each other, while there there will be far more diversity among organisms in higher subtrees, such as phyla or classes. Using this observation, we can adjust the similarities of A and B by choosing the right taxonomic rank from which to sample their constituent organisms. For instance, constructing A and B using organisms randomly sampled from a certain species or genus should make A and B highly similar, and we would expect a well-behaving similarity index to produce a high value for A and B constructed in this manner. On the other hand, constructing A and B with organisms randomly sampled from some class or phylum will likely render A and B quite different from each other, and a well-behaving similarity index should reflect this lack of similarity with a low output.

Implementation

Prior to running our simulation, we required a data source for the genomes used in our reference database D. For our simulations, we used GenBank [15], a database of over 100,000 bacterial genome sequencing data, annotations, and taxonomic information from the National Center for Biotechnology Information, a division of the National Institute of Health. Out of all the genomes in GenBank, many consist of incomplete sequencing data for organism genomes. We only sought complete bacterial genomes to use for construction of our metagenomic samples. To find these genomes, we took advantage of the summary database the NCBI provides, which includes taxonomic information and metadata on genome completeness. We filtered out and downloaded only genomes that were marked complete (about 25,000 out of the total).

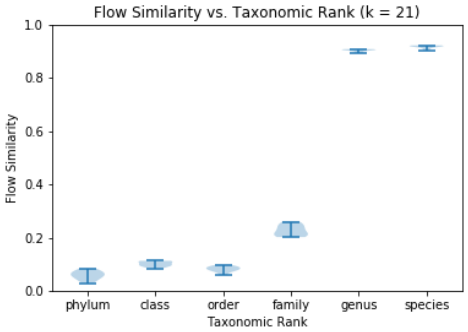
Now equipped with the genetic data for simulating metagenomic samples, we needed the facilities to to pick a random taxonomic rank of a certain level. More specifically, we wanted the taxonomic IDs of all phyla, all classes, etc. so that we could randomly pick one to simulate each of A and B with and run the rest of our computations. We first used TaxonKit [16], a high-performance toolkit for retrieving taxonomic information from the NCBI GenBank database, to retrieve all taxonomic IDs under a particular taxonomic rank. We then retrieved all bacterial taxonomic IDs with TaxonKit, making sure to show the rank of each retrieved ID, and we then grep-searched all taxIDs of each desired rank and placed all IDs of each rank in a separate file. We ended up with a file containing taxonomic IDs of all phyla, taxonomic IDs of all classes, etc.

At this point, we sought to pick random locally-stored bacterial genomes for the construction of A and B. To simulate A using the taxonomic rank of interest, we first retrieved the IDs of all organisms under our desired rank with TaxonKit, randomly selected 100 taxonomic IDs from these organisms, and searched the GenBank database file for each corresponding genome. We then merged the genomes we found and simulated A with InSilicoSeq [17], a metagenomic read simulator which mimics the Illumina [18] sequencing platforms with kernel density estimators. We generated one milion reads for A, using a uniform distribution for sequencing coverage and an exponential distribution for organism abundance. To give the reads realistic noise, we configured InSilicoSeq to use a MiSeq [19] error profile, which introduces read errors consistent with the Illumina MiSeq platform. We repeated this process to construct B. Finally, we computed all of the GMSG indices and metrics, as well as the Jaccard index via Mash and Sourmash. For the GMSG, Mash, and Sourmash computations, we used a k-mer size of $k = 21$. We repeated the random sampling, simulation, and index computation ten times for each rank. We then categorically plotted the estimates of each quantity against the taxonomic rank from which A and B were constructed.

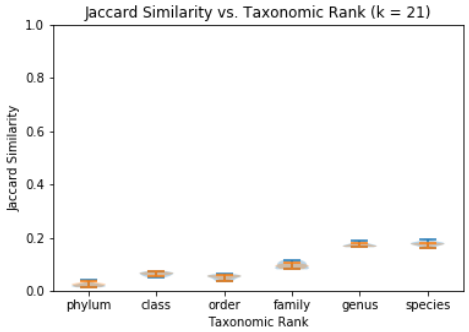
We describe this process more concisely a single taxonomic rank (an arbitrary phylum, denoted by ρ) for simplicity:

1. Repeat the following process 10 times:
   (a) Randomly select 100 genomes which are members of ρ, and construct A from these genomes with InSilicoSeq.
   (b) Randomly select 100 genomes which are members of ρ, and construct B from these genomes with InSilicoSeq.
   (c) Compute $\phi_D^{(k)}(A, B)$, $\Psi_D^{(k,1)}(A, B)$, $\Psi_D^{(k,\frac{3}{2})}(A, B)$, and $\Psi_D^{(k,2)}(A, B)$.
   (d) Compute $J(A, B)$ with both Mash and Sourmash.
2. Take the average of the ten point estimates for each of $\phi_D^{(k)}(A, B)$, $\Psi_D^{(k,1)}(A, B)$, $\Psi_D^{(k,\frac{3}{2})}(A, B)$, and $\Psi_D^{(k,2)}(A, B)$.

GMSG Flow Similarity Behavior



(a) GMSG flow similarity plotted with respect to the taxonomic rank from which A and B were consructed.



(b) Jaccard similarity via Mash (blue) and Sourmash (yellow), plotted with respect to the taxonomic rank from which A and B were constructed.

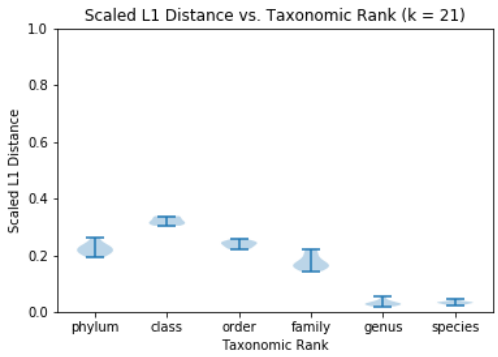
Figure 4.1: GMSG flow similarity and Jaccard similarity, plotted against the taxonomic rank from which A and B were constructed. Each element is a violin plot of the ten estimates of each quantity. The two horizontal bars describe the range of the estimates, and the color-filled region between the bars is an approximate distribution of the estimates.

In figure 4.1, we observe that both the flow similarity and Jaccard index generally increased as A and B took on more similar organisms in narrower taxonomic ranks. We see that both the GMSG flow similarity and the Jaccard similarities are low for A and B constructed from members of broad taxonomic ranks. This is expected, since if we are randomly sampling from a broad taxonomic rank like a class or phylum, we are extremely likely to construct A and B from organisms that are barely related to each other. For A and B constructed from organisms sampled from the species and genus ranks, the GMSG sharply increased to reflect the high similarity between the organisms from which the sample was drawn, while the Jaccard index, despite increasing, was still relatively lower. These results suggest that the GMSG flow similarity is more effective than the Jaccard index at detecting similarity between communities containing very similar organisms, and is as effective as the Jaccard index at reflecting a lack of similarity between communities constructed from random sample of a diverse organism set.

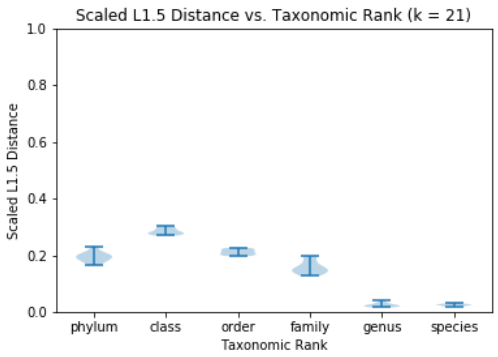
It's important to note that the Jaccard index can turn out to be artificially low if the k-mers produced depend on which of the two strands of a given DNA read were sequenced. For instance, if the majority of the k-mers in $A^{(k)} \cap B^{(k)}$ were on one strand but our dataset contains only the other strand, then a framework that doesn't account for this phenomenon will underestimate $|A^{(k)} \cap B^{(k)}|$ and therefore will produce a low value for $J(A^{(k)}, B^{(k)})$. Fortunately, both Mash and Sourmash innately address this issue by computing the k-mers of A and B in a strand-independent manner. For a given k-mer and its complement on the other strand, Mash and Sourmash compute $J(A^{(k)}, B^{(k)})$ using the lexicographically smaller of these two k-mers. Since this choice does not depend on the strand of the DNA the original k-mer lies on, the computation of $J(A^{(k)}, B^{(k)})$ does not depend on which of the two strands of DNA was used for our dataset. As a result, this phenomenon cannot be responsible for the low values of $J(A^{(k)}, B^{(k)})$ seen in figure 4.1. Some other underlying mechanism is at play, and we hypothesize about its inner workings in

section 4.1.1.

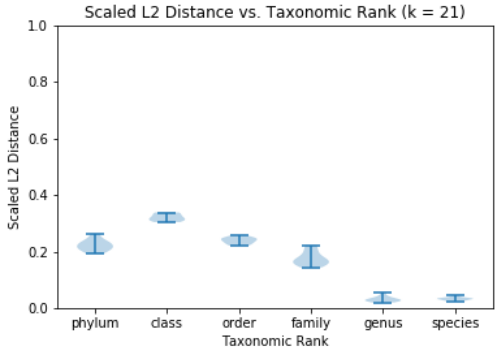
GMSG Distance Metrics Behavior



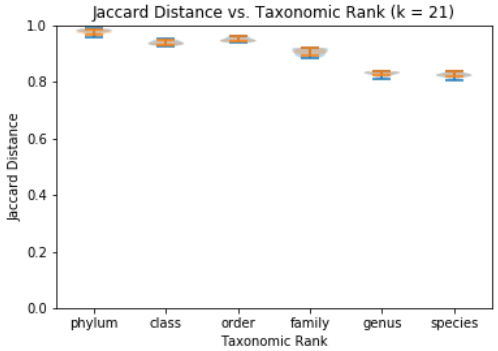
(a) GMSG l_1 metric vs. taxonomic rank



(b) GMSG $l_{\frac{3}{2}}$ metric vs. taxonomic rank



(c) GMSG l_2 metric vs. taxonomic rank



(d) Jaccard distance vs. taxonomic rank, via Mash (blue) and Sourmash (yellow)

Figure 4.2: GMSG distance metrics and Jaccard distance, plotted against the taxonomic rank from which A and B were constructed. Each element is a violin plot of ten estimates of the corresponding quantity. The two horizontal bars describe the range of the estimates, and the color-filled region in between the bars is an approximate distribution of the estimates.

In figure 4.2, we observe an analogous pattern for metagenomic distance. The GMSG distance metrics and Jaccard distance generally decreased as A and B took on more related organisms. For A and B constructed from organisms in broad taxonomic ranks, the Jaccard distance was close to 1, while the GMSG metrics were low, though noticeably greater than zero. For A and B constructed from highly related organisms in more specific taxonomic ranks, the Jaccard distance was still close to 1 despite decreasing, while the GMSG distance metrics were quite close to zero. These results suggest that the GMSG metrics are not sensitive enough to differences in organism containments in A and B. This issue could very well arise from the naivety of our scalng factor of $\frac{1}{|D|}$ as described in chapter 3, which one could possibly resolve choosing a factor which considers the number of organisms for which both containments in A and B are nonzero (and maybe past a certain threshold). Further research is needed to determine if these modifications would be useful.

Just as we observed a low Jaccard index between metagenomes that should be similar, we naturally observed a high Jaccard distance under the same circumstances. Having already ruled out one possible cause, we hypothesize about another possible source of this behavior in the next section, which speaks volumes to the importance of robustness of similarity indices to noise.

Why This Behavior?

We hypothesize a cause for the behavior seen in figure 4.1. The results in figure 4.1 may possibly be explained by the fact that k-mer based methods for computing the Jaccard index between two sets of k-mers $A^{(k)}$ and $B^{(k)}$ rely on the "purity" of the k-mers in the intersection of $A^{(k)}$ and $B^{(k)}$. Due to the errors and noise produced by sequencing machines, k-mers that were supposed to be in $A^{(k)} \cap B^{(k)}$ could mutate and thus fall into only one of $A^{(k)}$ or $B^{(k)}$, which would cause estimates of Jaccard similarity to be artificially low. Instead of purely counting k-mers in the intersection, GMSG flow similarity considers the containment of k-mers in relevant reference organisms. Thus, if a k-mer that was supposed to increase similarity is given an error and falls outside of the sketch of one organism, it still may be contained in a highly related organism in our database, so the similarity this k-mer was supposed to contribute is not lost despite the errors it contains. Generally, these phenomena are difficult to characterize because the error/noise applied to the k-mers depends on the error model of the read simulator being used. However, Koslicki et al. were able to characterize the distribution of errors in k-mers under a simple point mutation model [20]. It's warranted to attempt to extend this research to more complex error models, as this will pave the path to better understanding of how k-mer methods behave with realistic, noisy data.

4.1.2 Detecting Similarity from Poorly Covered Common Genomes

Motivation

Barring the presence of noise, the definition of the Jaccard index possibly creates a predisposition to ignoring similarity when it seems as if there is none. Chapter 3, we mentioned that in situations where sequencing coverage is insufficient, GMSG could possibly detect the presence of common organisms in disjoint $A^{(k)}$ and $B^{(k)}$ while the Jaccard index will not. That being said, it's quite difficult to create two metagenomes that are genuinely disjoint whlie stilll preserving realistic qualities. Therefore, we consider a more general situation. It's possible that if the reads of (not necessarily disjoint) A and B poorly cover the DNA of the only organisms present in both samples, then then Φ will detect the extra similarity that these organisms contribute despite their poor coverage, while the Jaccard index will do so to a noticeably lesser extent. We now detail a simulation that determines if this is the case.

Implementation

We selected 250 random genomes from all our locally stored complete bacterial genomes available in GenBank as our reference database. We allotted the first 100 of these for constructing A only, the next 100 for constructing B only, and the last 50 for constructing both A and B. We configured InSilicoSeq to use an exponential coverage distribution, which assigns an exponentially decaying sequence of coverages to the genomes in A and B. Due to the order in which these genomes were specified, both the genomes common to only A and those common to only

B received higher coverage, while the 50 genomes common to both received very low coverage. We constructed A and B by simulating one million reads for each, and we computed $\Phi_D^{(k)}$ and $J(A^{(k)}, B^{(k)})$, using a k-mer size of $k = 21$. We repeated this process ten times and plotted the ten point estimates for each index.

Low-Coverage Results

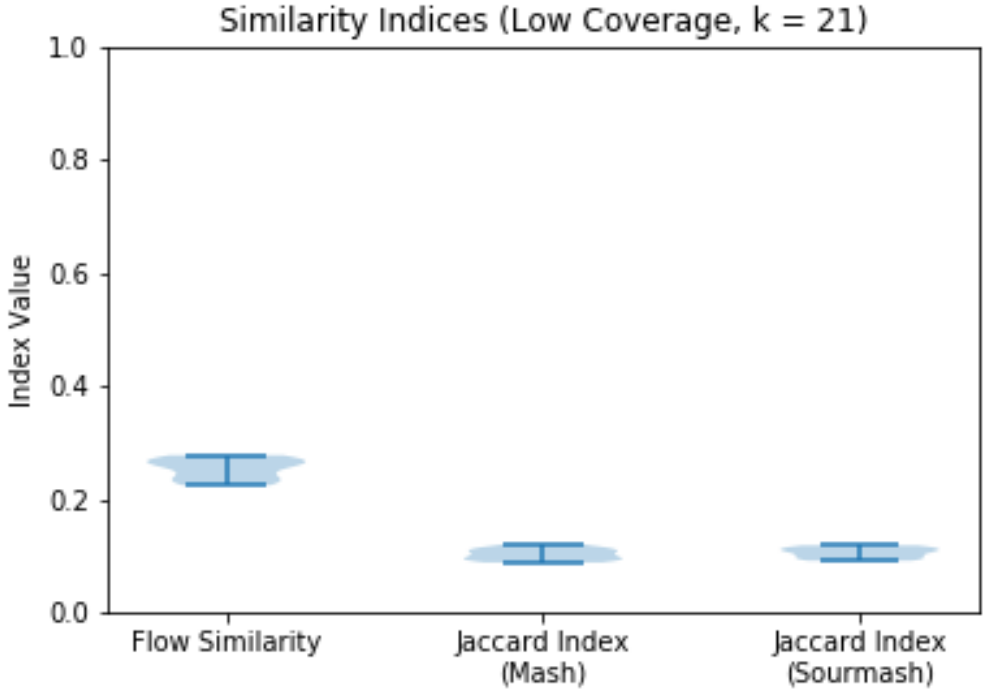


Figure 4.3: Comparison of GMSG flow similarity and Jaccard similarity between two metagenomes with poorly-covered common genomes. Each element is a violin plot of ten estimates of the corresponding quantity. The two horizontal bars describe the range of the estimates, and the color-filled region in between the bars is an approximate distribution of the estimates.

In figure 4.3 we can see that the GMSG flow similarity is higher than the Jaccard index for all ten estimates. These results suggest that out of the fifty ill-covered but common genomes in A and B, the GMSG flow similarity was able to detect the hints of their presence while the Jaccard index did not recognize k-mers from these genomes to the same extent. This suggests that in low-coverage situations, the GMSG flow similarity is more effective at detecting common but poorly covered organisms and increasing similarity accordingly. These results are a generalization of our intuition in section 3.3.1; while we analyzed the case where $A^{(k)}$ and $B^{(k)}$ were disjoint, this is a special case of the real-world scenario in which $A^{(k)}$ and $B^{(k)}$ are not disjoint, but their intersection is poorly covered by our data.

4.2 Implications

Improvements in metagenomic similarity indices can significantly impact the insight we gain into the relationships between multiple metagenomic datasets, for both ourselves and the world around us. The insights we gain with better-behaving similarity and distance metrics have implications for our understanding of increasingly prevalent conditions with potential microbial causes. As mentioned in chapter 1, severity of conditions like atopic dermatitis and irritable bowel syndrome have been associated with compositions of microbial communities on and within the human body. Using well-behaved, robust metagenomic similarity and distance metrics, one could perform clustering with the GMSG indices and metrics on metagenomic samples from patients with these conditions, grouping patients by symptom severity as it relates to microbial composition, which will pave the way for a continuum of personalized treatments for such patients. Even if one is not suffering from disease, gaining insight into our own microbiomes and their similarity to those of others can open up avenues for personalized preventative microbial supplements to maintain the health of our collective microbial companions, all based on which "cluster" our microbial profiles fall into. There are more opportunities along similar lines beyond the ones mentioned, but likely all of them relate back to our ability to extract insight into how microbial communities are related. The sensitivity and accuracy of similarity indices and distance metrics to common organisms and differences in microbial composition is crucial to performing this task well. While further research is needed, the GMSG similarity indices reflected increases in similarity quite well, and further improvements to both this index and the GMSG metrics will help advance our performance in extracting relational insights from metagenomic samples.

4.3 Further Research

4.3.1 Incorporating Genomic Similarity into GMSG Flow Similarity

As shown, the GMSG incorporates information about containment relationships between genomes and metagenomes, but the similarity amongst our reference genomes can play a role in measuring the similarity of A and B. For instance, if A contains organism x and not organism y, and B contains y but not x, but x and y are highly related, it would make sense to increase the similarity measurement of A and B to reflect their containment of genetically similar organisms. To do this, we would need to store the similarities between reference genomes in the GMSG, which can be done by placing undirected edges between genomes weighted by a symmetric similarity measurement like Average Nucleotide Identity (ANI) [21]. Adding these edges to the GMSG would create overlapping paths, so one would need a general maximum flow algorithm to compute a flow-based similarity of A and B. While it's possible that this extension to the GMSG data structure would convey better behavior of flow similarity, this approach would result in a significant increase in asymptotic runtime. Further research is needed to determine whether the increase in accuracy of this enhancement is worth the computational penalty.

4.3.2 Recovering Constituent Organisms with Phylogenetic Trees

When using the GMSG on real metagenomic data, we would want to address the possibility that our reference database is not complete, and to find the organisms the reference database missed. The genomes in our database are the only clues we have about the missing organisms, so we could search for a relative of the organisms present in one of A or B but not the other. Suppose A contains organism a and not organism b, and B contains b but not a. We could find the least common ancestor between a and b in a phylogenetic tree [22], then trace a path down the tree to an organism c that would be "most likely" to be contained in both A and B, then include c in our database and computation of flow similarity. Again, such an approach would incur significant space and time penalties due to the vast increase in information used to compute similarity, and further research is needed to determine if this enhancement is worth the increase in space and time complexity.

4.3.3 Characterizing Sensitivity of k-mer Methods to Sequencing Noise

As mentioned before, the noticeable difference in behavior between the GMSG metric and Jaccard index could be explained by the errors introduced into k-mers by realistic error models, such as those used by InSilicoSeq. Using the specifications of a particular model, it's possible to quantify the frequency at which errors occur in reads (and thus, their k-mers) based on the type of distribution from which base errors are generated. Thus, one could describe the portion of originally common k-mers which now fall outside $A^{(k)} \cap B^{(k)}$ due to base errors, which would yield insight into how far the estimated Jaccard Index deviates from the true Jaccard index as a function of some error model. With further research, one could apply the same thought to containment-based indices and metrics to quantify exactly how robust their estimates are to sequencing noise compared to those of other k-mer based indices.

4.4 Conclusion

Metagenomics is a promising improvement on standard genomics, offering methods for more complete and insightful analysis of microbial communities. Among the numerous techniques for metagenomic analysis, k-mer based methods are useful for quickly computing taxonomic profiles and comparing metagenomes to one another with indices like the Jaccard index. However, such k-mer motivated indices provide insufficient information on presence of common organisms and are highly sensitive to spurious k-mers arising from sequencing noise. As shown, containment-based methods like the GMSG and its derived similarity indices can offer more robust, well-behaved measurements of metagenomic similarity between well-studied communities, while GMSG distance metrics require further research to improve their behavior. Further improvements to the GMSG data structure and indices can recover units of metagenomic similarity for more informative containment-based metagenome comparison. Ultimately, such innovations on similarity indices and distance metrics can help us better understand our relationships with our own microbial communities, which will pave the way towards possible disease treatments and better humans' relationships with their microbial neighbors.

Bibliography

[1] Allyson L. Byrd, Clay Deming, Sara K.B. Cassidy, Oliver J. Harrison, Weng Ian Ng, Sean Conlan, Yasmine Belkaid, Julia A. Segre, and Heidi H. Kong. Staphylococcus aureus and Staphylococcus epidermidis strain diversity underlying pediatric atopic dermatitis. *Science Translational Medicine*, 9(397), jul 2017.

[2] William Chey and Stacy Menees. The gut microbiome and irritable bowel syndrome [version 1; referees: 3 approved], 2018.

[3] W. Wade. Unculturable bacteria - The uncharacterized organisms that cause oral infections. In *Journal of the Royal Society of Medicine*, volume 95, pages 81–83. Royal Society of Medicine Press, 2002.

[4] Janelle R. Thompson, Sarah Pacocha, Chanathip Pharino, Vanja Klepac-Ceraj, Dana E. Hunt, Jennifer Benoit, Ramahi Sarma-Rupavtarm, Daniel L. Distel, and Martin F. Polz. Genotypic diversity within a natural coastal bacterioplankton population. *Science*, 307(5713):1311–1313, feb 2005.

[5] Chloe A. Douglas, Kerry L. Ivey, Lito E. Papanicolas, Karen P. Best, Beverly S. Muhlhausler, and Geraint B. Rogers. DNA extraction approaches substantially influence the assessment of the human breast milk microbiome. *Scientific Reports*, 10(1):1–10, dec 2020.

[6] Paul Jaccard. THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.1. *New Phytologist*, 11(2):37–50, feb 1912.

[7] Michel Marie Deza and Elena Deza. *Encyclopedia of Distances*. 2016.

[8] Andrei Z. Broder. On the resemblance and containment of documents. In *Proceedings of the International Conference on Compression and Complexity of Sequences*, pages 21–29. IEEE, 1997.

[9] Brian D. Ondov, Todd J. Treangen, Páll Melsted, Adam B. Mallonee, Nicholas H. Bergman, Sergey Koren, and Adam M. Phillippy. Mash: Fast genome and metagenome distance estimation using MinHash. *Genome Biology*, 17(1):132, jun 2016.

[10] N. Tessa Pierce, Luiz Irber, Taylor Reiter, Phillip Brooks, and C. Titus Brown. Large-scale sequence comparisons with sourmash [version 1; peer review: 2 approved]. *F1000Research*, 8, 2019.

[11] David Koslicki and Hooman Zabeti. Improving MinHash via the containment index with applications to metagenomic analysis. *Applied Mathematics and Computation*, 354:206–215, aug 2019.

[12] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, jul 1970.

[13] Wes McKinney. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 56–61. SciPy, 2010.

[14] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy, sep 2020.

[15] Dennis A. Benson, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and Eric W. Sayers. GenBank. *Nucleic Acids Research*, 38(SUPPL.1):D46, nov 2009.

[16] Wei Shen and Jie Xiong. Taxonkit: a cross-platform and efficient ncbi taxonomy toolkit. *bioRxiv*, 2019.

[17] Hadrien Gourlé, Oskar Karlsson-Lindsjö, Juliette Hayer, and Erik Bongcam-Rudloff. Simulating Illumina metagenomic data with InSilicoSeq. *Bioinformatics*, 35(3):521–522, feb 2019.

[18] Illumina. An introduction to Next-Generation Sequencing Technology. Technical report.

[19] Melanie Schirmer, Rosalinda D'Amore, Umer Z. Ijaz, Neil Hall, and Christopher Quince. Illumina error profiles: Resolving fine-scale variation in metagenomic sequencing data. *BMC Bioinformatics*, 17(1):125, mar 2016.

[20] Antonio Blanca, Robert S. Harris, David Koslicki, and Paul Medvedev. The statistics of k-mers from a sequence undergoing a simple mutation process without spurious matches. *bioRxiv*, 2021.

[21] David R. Arahal. Whole-genome analyses: Average nucleotide identity. In *Methods in Microbiology*, volume 41, pages 103–122. Academic Press Inc., jan 2014.

[22] Cody E. Hinchliff, Stephen A. Smith, James F. Allman, J. Gordon Burleigh, Ruchi Chaudhary, Lyndon M. Coghill, Keith A. Crandall, Jiabin Deng, Bryan T. Drew, Romina Gazis, Karl Gude, David S. Hibbett, Laura A. Katz, H. Dail Laughinghouse, Emily Jane McTavish, Peter E. Midford, Christopher L. Owen, Richard H. Ree, Jonathan A. Rees, Douglas E. Soltis, Tiffani Williams, and Karen A. Cranston. Synthesis of phylogeny and taxonomy into a comprehensive tree of life. *Proceedings of the National Academy of Sciences*, 112(41):12764–12769, 2015.

Isaac Thomas

Education

Pennsylvania State University – Schreyer Honors College / University Park, PA / *January 2018 – May 2021*

B.S. in Computer Science with Honors, Minor in Mathematics

Coursework: Data Structures & Algorithms; Artificial Intelligence; Theory of Computation; Computer Organization & Design; Probability Theory & Stochastic Modeling; Statistical Inference; Systems Programming;

Experience

SWE Intern / Quantitative Engineering & Data / Susquehanna International Group / *June – August 2020*

- Developed tools to analyze and improve the performance of Riptide, an in-house data analytics framework used by quantitative researchers and traders for automated strategy development and market studies.
- Used bootstrap sampling, linear regression, hypothesis testing, and kernel density estimation with Numpy, Scipy, and Scikit-learn to develop a statistical comparison tool for Riptide's performance data.
- Implemented a genetic algorithm with Pymoo to optimize machine parameters for increased function performance.
- Developed a sampling-without-replacement algorithm with Numpy to generate data with desired distributions for study of fancy indexing performance.
- Used Numba to develop JIT-compiled implementations of commonly used Riptide data analysis functions.
- Built out performance benchmarks to measure performances of Riptide functions over a comprehensive input space.
- Presented work & findings to senior management.

Research Assistant / Koslicki Group / *January 2020 – Present*

- Contribute to CMash, a fast set-similarity estimation tool for metagenomic binning.
- Develop a k-mer input prefilter with KMC to increase CMash's performance on large metagenomes.
- Develop a novel graph structure for metagenome similarity estimation.

SWE Intern / Corporate Technology / JPMorgan Chase & Co. / *June – August 2019*

- Designed a cost-effective, deliverable tool to assist with automated, data-driven assessment and reporting of JPM's compliance with federal lending legislation.
- Implemented python-based data analysis libraries to develop a lightweight Business Rules Engine for analysis of quarterly consumer lending datasets, to be deployed internally and in place of an expensive third-party alternative.
- Presented project to the Chief Information Officer of Corporate Technology and senior management.

Research Assistant / Shao Group / *January – August 2019*

- Worked on early-stage project to detect micro-exons within RNA read alignments.
- Utilized integrative genomic visualization software and C++ RNA analysis frameworks to create distributions of known micro-exons in human pre-mRNA.
- Used sequence alignment data manipulation libraries to analyze mappings of RNA reads to the human genome.

Projects

Genome-Metagenome Similarity Graph / Schreyer Honors Thesis

A novel graph structure to measure the similarity of two metagenomes via containment relationships between metagenomes and their constituent genomes. Currently in development for submission in May 2021.

Skills

- Numpy
- Scipy
- Scikit-learn
- Pandas
- Tensorflow
- Pymoo
- Numba
- Python
- C/C++
- Git
- Bash
- Logic Pro