THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

L2-UniFrac

ANDREW MILLWARD
SPRING 2022

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

Reviewed and approved* by the following:

David Koslicki
Associate Professor of Computer Science and Engineering, Biology, and the Huck Institutes of the Life Sciences
Thesis Supervisor

Danfeng Zhang
Associate Professor of Computer Science and Engineering
Honors Adviser

Paul Medvedev
Associate Professor of Computer Science and Engineering, Biochemistry and Molecular Biology, and the Huck Institutes of the Life Sciences
Faculty Reader

*Electronic Approvals are on File.

Abstract

Within the field of bioinformatics, a rather popular method of quantitatively assessing the differences between microbial samples is the UniFrac metric. This metric assigns a "distance" measure between two samples based on experimental abundances and weights across a phylogenetic tree. Over the past two decades, UniFrac has advanced in several key iterations that primarily focused on improving the performance of the algorithm with reducing execution time as the core focus. Following the introduction of EMDUniFrac, a theoretical expression of EMDUniFrac in terms of L1-norm differences was proposed. This paper outlines a re-expression of the EMDUniFrac algorithm in terms of the L2-norm as well as in terms of L2-norm differences. It also shows the utility of such a metric in enabling the computation of averages with respect to such L2-norm while maintaining natural constraints to allow for more efficient comparisons between different microbial communities.

We first outline a mathematical basis for the introduction of L2-normalization on a theoretical level from prior work. Then, using this understanding, we redefine the structure of the method required to preprocess vectors into L2-UniFrac space with respect to shared phylogeny and the L2-norm as well and provide a method to reverse such process. Additionally, we provide a new algorithm to compute the standard L2-UniFrac metric directly without preprocessing, as well as a scheme for obtaining this same value by first pushing vectors into L2-UniFrac space and computing the L2-norm of their difference. Finally, we propose a process to compute the average of samples with respect to the L2-norm.

This work then outlines several experiments that serve to evaluate the efficacy of the L2-UniFrac metric. Such tests include verifying similar PCoA and clustering performance on individual samples compared with prior UniFrac metrics, testing for the presence of negative abundances, evaluating the clustering of closely related sample averages with PCoA, analyzing the taxonomic structure of such averages both directly and with KronaTools, and applying differential abundance testing to the representative sample.

The results of this experimentation verify that L2-UniFrac performs very closely to prior work such as EMDUniFrac with regards to individual samples and vastly exceeds the performance of prior methods when applied to averaged, representative samples. L2-UniFrac provides a more natural representation of representative samples for entire microbiomes and maintains key natural constraints that are otherwise violated when applying prior metrics to this application. Finally, the averaged vectors provided by L2-UniFrac provide a natural framework to analyze specific OTU discrepancies between microbiomes with results supported by prior literature.

Table of Contents

List of Figures v

List of Tables vi

1 Introduction 1

2 Required De nitions 5

2.1 Biological Definitions . . . 5

2.1.1 Phylogenetic Trees . . . 5

2.1.2 UniFrac . . . 6

2.2 Technical Definitions . . . 7

2.2.1 L1 Norm . . . 7

2.2.2 L2 Norm . . . 7

2.2.3 Packages . . . 8

2.2.4 PCoA . . . 8

2.2.5 Clustering Techniques . . . 9

2.2.5.1 Agglomerative Clustering . . . 9

2.2.5.2 K-Medoids Clustering . . . 10

2.2.6 Clustering Scoring Algorithms . . . 11

2.2.6.1 Rand Index Score . . . 11

2.2.6.2 Adjusted Rand Index Score . . . 12

2.2.6.3 Normalized Mutual Info Score . . . 13

2.2.6.4 Adjusted Mutual Info Score . . . 14

2.2.6.5 Fowlkes Mallows Score . . . 14

2.2.7 Krona . . . 15

3 Methods 17

3.1 Characterization of L1 and L2 UniFrac . . . 17

3.2 Closure Under Averages . . . 19

3.3 Push Up . . . 26

3.4 Inverse Push Up . . . 29
3.5 Computing L2-UniFrac with Push-Up and Inverse-Push-Up . . . 31
3.6 Computing Averages with Respect to L2-UniFrac . . . 32
3.7 L2-UniFrac Weighted Plain . . . 34
3.8 Main Function . . . 35

4 Experiments 37

4.1 L1 vs L2 . . . 38
4.1.1 Principal Coordinate Analysis of Samples . . . 38
4.1.2 Clustering Analysis . . . 40
4.2 Presence of Negatives . . . 41
4.3 Principal Coordinate Analysis of Means . . . 44
4.4 Taxonomic Distribution . . . 45
4.5 Krona Evaluation . . . 47
4.6 Most Representative Organism . . . 48

5 Results 52

5.1 Principal Coordinate Analysis of Samples . . . 52
5.2 Clustering Analysis . . . 55
5.3 Presence of Negatives . . . 60
5.4 Principal Coordinate Analysis of Averages . . . 62
5.5 Taxonomic Distribution of Averages . . . 64
5.6 Krona Visualizations . . . 71
5.7 Differential Abundance . . . 79

6 Conclusion 87

List of Figures

1 L1-UniFrac PCoA Plot of All Samples . . . 52

2 L2-UniFrac PCoA Plot of All Samples . . . 53

3 Alternate Perspective of L2-UniFrac PCoA Plot of All Samples . . . 54

4 Clustering PCoAs for L1-UniFrac . . . 57

5 Clustering PCoAs for L2-UniFrac . . . 59

6 PCoA Plots of Group Averages . . . 63

7 Taxonomic Distribution of Average Fecal Body Site . . . 65

8 Taxonomic Distribution of Other L1 Body Sites . . . 68

9 Taxonomic Distribution of Other L2 Body Sites . . . 70

10 Krona Visualizations of L1-UniFrac Averages by Group . . . 73

11 Krona Visualizations of L2-UniFrac Averages by Group . . . 76

12 Proteobacteria Krona Analysis in Vaginal Body Site with L2-UniFrac Average . . . 77

13 Differential Abundances using L2-UniFrac with Internal Nodes . . . 80

14 Differential Abundances using L2-UniFrac without Internal Nodes . . . 82

15 Brief Comparison of L1-UniFrac to L2-UniFrac in Differential Abundances . . . 85

List of Tables

1 Arguments for use when calling main.py 36

2 Clustering Scores Under Clustering and Scoring Algorithms for L1 and L2-UniFrac 55

3 Negative Probability Count by Group Averages 61

4 List of Negatives by Body Site 61

5 Most Representative OTUs by Body Site 84

1 Introduction

When comparing microbial communities, an important and commonly-used metric used to quantitatively assess the differences is the UniFrac metric [25]. Specifically, UniFrac is applied with respect to the phylogeny of the microbial communities using the weighted distances or branch lengths between taxa in the chosen phylogenetic tree. The standard variant of UniFrac, as originally intended, was designed to quantitatively measure the "distance" between two samples, which, in turn, can be used to perform a principal coordinate or clustering analysis to derive more information such as the contributions of various factors that contribute to the microbial communities' similarities [25]. However, numerous researchers and papers have developed improvements over the originally proposed algorithm to address several shortcomings.

One paper of note that improved upon this existing metric is that of Fast UniFrac [16] which reduces the execution time of UniFrac by 10-100 times. This performance enhancement is accomplished by transposing the phylogenetic tree into arrays which can perform accelerated vector operations which take advantage of system cache more than node objects and array slicing. It is important to note that the improvements present in Fast UniFrac are implementation improvements that more efficiently allocate system resources rather than direct algorithmic improvements. A later improvement known as EMDUniFrac, on the other hand, developed an algorithmic improvement to UniFrac that achieves linear time performance [29]. As noted by McClelland and Koslicki, UniFrac can be viewed similarly to the Kantorovich-Rubinstein metric, otherwise known as the "Earth-Mover's distance," abbreviated as EMD, which has been found useful in various other applications [44]. The EMD is applied on the phylogenetic tree by substituting the earth mover's ground distance matrix with a pairwise distance matrix across every sample and the flow denoting the total abundance pushed from the start to ending node in a particular path [29]. Together, EMDUniFrac uses these adjustments to minimize flow up the phylogenetic tree and determine the total UniFrac distance between two microbial samples in linear time.

With more and more data becoming available over time, computing this pairwise distance matrix across every sample is naturally extraordinarily burdensome and expensive to produce. Some datasets such as that of the American Gut Project [14] contains a total of $33;570$ samples. Computing the pairwise distances for each of these samples would require a total $1;126;944;900$ cells of data, which, if stored as 8-byte doubles, would consume roughly 9GB of data and, assuming one is to use EMDUniFrac to compute the pairwise distances, each of these cells would take linear time with respect to the number of samples. As can be seen, the computational complexity of computing such

a matrix to compare microbial communities explodes dramatically as data size increases.

Dr. McClelland, who had worked on the aforementioned paper EMDUniFrac, later developed a characterization for UniFrac in terms of L1-norm differences [28]. This development is an important milestone for UniFrac as a whole since it opens the doors to address some of the aforementioned concerns with the base form of EMDUniFrac when deploying it across larger communities. This formulation simply expands upon EMDUniFrac to allow it to consider individual samples as atomic components that can be brought into L1-UniFrac space. The differences of these preprocessed samples is then with respect to the UniFrac metric, and the L1-norm of this vector is precisely the computation given by EMDUniFrac. Rather than computing the pairwise distance matrix of all samples, one could instead preprocess each of the samples by scaling each vector by a matrix of phylogenetic tree edge lengths, W, where ancestors of a given node in a row are assigned the same edge weight as that node has along the main diagonal. This scheme will be explored in more depth in section 3.1. At this point, depending on the propagation technique, one could compute the difference between two preprocessed samples and compute exactly the UniFrac distance between the two using the respective norm. Thus, using such a preprocessing procedure, it would be possible to pre-compute the pushed-up vector for all of the aforementioned $33,570$ samples in the American Gut Project [14] and then find the pairwise distance matrix with $1,126,944,900$ vector operations. Since vector operations offer significantly improved parallelism in many implementations such as those of interpreted languages, the process of computing these L1-norms of the vector differences is much faster in practice than the direct pairwise computation of EMDUniFrac in many implementations. Dr. McClelland provides a useful linear algebra characterization of UniFrac, and his work suggests the ability to take averages with respect to the UniFrac metric. This ability to take averages can be accomplished simply by using the propagated samples to compute averages across entire microbial communities as defined by metadata and pushed down into a representative averaged sample that can be directly compared with other microbial communities in linear time, thereby bypassing the need to compute the entire distance matrix altogether for this application. With that said, the ability to take averages with the L1 norm gives a signed measure which leads to problems with interpretability as a result of negative abundances. To prove this claim formally, section 3.2 will perform an in-depth analysis of why this is the case and why, using the future formulation of L2-UniFrac, it is not possible. Additionally, there are several other issues with the existing preprocessing framework that we will be discussed in part in section 3.3 and more in directly in section 3.4 that this work will address.

For the purposes of this paper, we will refer to EMDUniFrac as L1-UniFrac, since it utilizes the L1-norm within its core algorithms and can be expressed as the L1-norm of the differences of two sample vectors. Some of the work

on L1-UniFrac, earlier denoted as "Median UniFrac," was briefly developed but unpublished as a result of these findings that significantly large negative abundances were possible. We will formally prove this in proposition 1 and section 5.3. Abundances should be strictly positive, so developing a modification to this algorithm would be necessary in order to maintain a strictly non-zero average computation. This motivates us to switch the L1-norm technique to the L2-norm due to the fact that L2 averages of probability vectors are also probability vectors themselves as will be outlined in proposition 2.

Further noted by Dr. McClelland and noted in the work of Evans and Matsen, L2-norm distance metrics hold more biological significance due to the fact that their derivation directly yields a function that, for a given community, takes the sum of pairwise UniFrac distances times their relative abundances and subtracts the average spread of the same metric within a community and between communities [28, 10]. As a result, switching to this metric also allows researchers to quantify the evolutionary spread of a given community versus a group of communities, allowing for more differentiation among distinct species. For the purposes of this experimentation, we will denote our work as L2-UniFrac since it is the L2-norm variant of the EMDUniFrac metric. We aim to experimentally compare EMDUniFrac's intended use cases with L2-UniFrac to show that our new proposed method retains usefulness in clustering of microbial communities and also compare L2-UniFrac's averaging against EMDUniFrac applied to the Median UniFrac averaging scheme to illustrate the importance of L2-UniFrac over prior methods.

Several driving motivations behind this improvement to EMDUniFrac and other related UniFrac algorithms lies in the ability to better assess various factors related to microbial communities. For instance, when assessing the difference between a healthy and unhealthy human microbiome, one such approach is differential abundance testing which illustrates the increase and decrease in abundance of specific microbes relative to the other sample. The definition of differential abundance vectors is given in [29] at the end of section 2. However, applying this approach directly to individual samples can lead to potential sampling bias which, without introducing statistical models and analysis of variance, can lead to inaccurate conclusions. Using the average of entire groups of microbial communities directly, however, it is much easier to directly use a differential abundance test against another similarly obtained average with greater degrees of confidence that can be assessed with other statistical methods. As more and more randomly taken samples are added and averaged, the sampling bias decreases substantially. Such averaged vectors produced by L2-UniFrac have other useful applications in taxonomic analysis for a snapshot of the composition of entire regions of potential samples as well as for assessing similarities and differences between different groups directly using PCoA plots of the averaged vectors instead. Rather than necessitating the production of a $n \ n$ distance matrix,

with n representing the total number of samples, to compare group similarities and differences, the method outlined in this paper allows for far fewer direct comparisons to be necessary by averaging groups first and then computing a much smaller distance matrix of size $k \quad k$, with k representing the total number of groups, resulting in a very large time and space savings, with the vectorization speedups provided through the preprocessing of samples as well as the reduction of pairwise computations necessary since $k << n$, thus significantly reducing the physical size of the group-averaged distance matrix.

L2-UniFrac characterizes a novel improvement to EMDUniFrac that enables the averaging of samples while retaining physical biological constraints. Though prior work has been shown to accurately and quickly compute the UniFrac distance between communities, as mentioned earlier, on rare occasions, taking group averages using prior methods produces unexpected results in the form of negative sample abundances. Since these results do not make sense in a biological context, this paper proposes a solution that shifts the L1-norm of Median UniFrac to an L2-norm and eliminates the possibility of negative sample abundances while maintaining similar levels of performance and utility in principal coordinate and clustering analyses. Additionally, this work shows additional value in terms of taxonomic and differential abundance applications with regards to the L2-UniFrac metric by leveraging this new technique to compute averages across entire communities. Such averages can subsequently be used to measure the spread between communities in terms of the UniFrac distance between the biological mean of one community to that of another as well as common features of one community that are absent from others. As a whole, L2-UniFrac is an improvement upon L1-UniFrac that retains useful individual sample characteristics of earlier work while enabling additional functionality of sample averages that are useful in a biological context.

2 Required Definitions

During this experimentation, this research leverages several existing tools in order to perform data analysis or visualization. Several examples of these include PCoA plot generation, clustering of distance matrices, scoring algorithms for clustering performance, and others. All of these tools are essential to properly analyzing the differences between this newly proposed method in comparison to the baseline method of L1-UniFrac. Additionally, this section will also seek to outline several basic biological definitions that may frequently appear in the paper in detail. Such definitions for both biological and technical aspects of this report will detail the computation process, if applicable, the purpose of such element, and any other necessary information or related terms that may appear later in this thesis.

2.1 Biological Definitions

To begin, some context is necessary to understand the purpose of several biological aspects of this paper. This section is intended to briefly outline such concepts to a level necessary to digest the rest of the content present within this paper. Since these concepts are directly relevant to several of the components that may appear in the technical definitions, it is critical to first understand these few concepts to continue in the paper.

2.1.1 Phylogenetic Trees

One of the core biological concepts present in this paper is that of phylogenetic trees. At its core, phylogenetic trees portray the evolutionary tree of life of species, with branches indicating common ancestors where multiple operational taxonomic units (OTUs) diverged and leaf nodes indicating each of the modern OTUs. Most commonly, phylogenetic trees, such as the one obtained from GreenGenes for this study, is constructed through sequencing of 16S rRNA with an objective tree construction tool such as FastTree [31, 41]. This data represents the evolutionary history of various representative organisms, denoted as OTUs and common relations along branches of various lengths. Connections between these branches in the tree are denoted using temporary (internal) nodes.

From these trees, we also derive several key variables that may be referenced later. These variables indicate different relations present in the tree that are critical to the understanding of UniFrac's algorithms. The first of these variables is named "Tint." Tint represents the ancestor dictionary whereby, given a node in the phylogenetic tree, Tint will index into the temporary node from which that node branched out. Since every node except the root has

a parent, all nodes will connect upwards in Tint and propagate up to the root node. Additionally, another key term present is "lint." lint is a data structure that, given a tuple of two nodes within the phylogenetic tree connected by an edge, will index into the length of that edge between those two nodes. Finally, the third major variable derived from the phylogenetic tree is titled "nodes_in_order." nodes_in_order simply contains the taxonomic IDs of each node in the tree (and temporary node IDs for temporary nodes) in a list in order from the left of the bottom-most level of the tree to its right, then the next level, etc. The final node in nodes_in_order will, therefore, be the root node for the entire tree.

2.1.2 UniFrac

As outlined in the introduction, several variations of "UniFrac" have existed over the years. However, the core question not yet fully addressed is what exactly is UniFrac? In its simplest form, UniFrac is a lineage-based phylogenetic distance metric defined as the fraction of branch lengths within a phylogenetic tree leading from one sample to the other but not to both [25]. As such, the concept of UniFrac can be used to attain a distance metric, effectively indicating the evolutionary distance between two microbial communities. That is, the UniFrac distance quantifies some degree of adaptations necessary for one community to adapt into another community. Higher UniFrac distances, therefore, indicate high dissimilarity between the two communities, and lower or zero UniFrac distances indicate more similar or exactly the same microbial community respectively.

The calculation of this metric, under the most trivial implementation, is computed by counting the total lengths of branches that are not shared and dividing it by the count of the total length of all branches [25]. That is, given the sum of all branch lengths leading to descendants such that the branches are not shared by both communities B_N and the total sum of all branch lengths B, the most basic UniFrac metric U is given as:

$$U = \frac{B_N}{B}$$

However, that computation primarily serves the unweighted UniFrac metric. The other variant of UniFrac of interest is the weighted variant of UniFrac, which will be more critical to this paper in particular. Weighted UniFrac makes a key modification to the summation in that it simply multiplies each branch length by the differential fractional abundance of that particular operational taxonomic unit (OTU) under that edge between each sample [4]. Weighted UniFrac, naturally, is the more popular choice in metagenomics research and will be the primary focus of

this paper as it directly relates distances to OTU abundances.

2.2 Technical Definitions

In addition to biological definitions of importance, as this paper is within the field of bioinformatics, it is also necessary to elaborate on the technical aspects used in this analysis. Such aspects may include elements such as major python packages that are utilized in the analysis, various formulae and algorithms to compute probabilities and other metrics of interest, and visualization packages used to generate all of the visuals used in this thesis.

2.2.1 L1 Norm

L1 Norms, otherwise known as the Manhattan norm, effectively computes the sum of the absolute value of each component of a vector. This norm can simply be expressed by the following mathematical notation:

$$k V\ k_1 = \sum_{v_i 2 V} j v_i j$$

In the context of UniFrac, the L1 norm would operate on the difference vector between each sample, thus giving the Manhattan distance between those two samples. The only adjustment to this is the scaling by edge lengths above each given node and still summing the absolute value.

It is important to note with the L1 norm that, by itself, it is not possible for an L1 norm to be negative since every value is added to the summation using an absolute value.

2.2.2 L2 Norm

The L2 Norm, otherwise known as the Euclidean norm, effectively is derived from the formula for the Euclidean distance which takes the square root of the sum of squares. For the norm, one simply sums the squared components of a vector and applies the square root at the end. This norm can be expressed by the following mathematical notation:

$$k V\ k_2 = \sqrt{\sum_{v_i 2 V} v_i^2}$$

In the context of UniFrac, the L2 norm also operates on the difference vector between each sample, thus giving the Euclidean distance between those two sample abundances. The adjustment, in this case, is that one must scale

by the edge lengths above each given node inside of the square root and sum such that the root applies at the end as well since the edge length is a component of the distance the mass must move directly. This will be elaborated upon more in section 3.7.

2.2.3 Packages

Firstly, it is important to cover all of the Python modules and packages needed to compute several of the computations. Of the major packages used, NumPy and SciPy are frequently used across testing and experimentation [17, 56]. NumPy is specifically utilized for its base computational abilities in the UniFrac algorithms themselves to simplify several operations. SciPy is used for its clustering and scoring algorithms mentioned in section 4.1.2 but also for its sparse matrix functionality to reduce algorithmic overhead in terms of storage space requirements. Specifically, several of these additional packages come from Scikit-Learn in the case of the clustering and clustering score metrics [39] and Scikit-Bio for the ordination tools and related functionality [47]. Additionally, we have also made use of the DendroPy package in some of the code in order to the phylogenetic tree files used for the experimentation [54]. For graphing capabilities, this paper makes use of Matplotlib and its available libraries and functions [19]. The python Pandas module was also used briefly for its DataFrame capabilities which were used for the graphing in part as well [37, 59]. For the packages that directly related to the dataset used, we utilized the Biom package to import the raw data into our program for analysis [30]. Finally, many of the built-in python modules were used as well, including those such as os, itertools, multiprocessing, time, sys, and several others.

2.2.4 PCoA

There exist several key method of utilizing multidimensional scaling to form models. Two of the most prominent techniques include principal component analyses (PCA) and principal coordinate analyses (PCoA). Despite their similar naming, there exist several key defining features. However, for the purposes of this research, the primary discussion will center around PCoA rather than PCA due to precedent in prior works related to this, including EMDUniFrac [29] and structure, function and diversity of the healthy human microbiome [5].

To begin, PCoA, otherwise known as a Principle Coordinate Analysis is a form of multidimensional scaling that takes in a measure of dissimilarities between components and produces a graphical representation grouping more similar elements closer together [33, 3]. That is, the primary function of PCoA is to find some set of dimensions that maximize the distance between dissimilar elements, which consequently emphasizes clustering among more similar

elements. Specifically, a PCoA analysis seeks to generate a graph where some strain or loss function is minimized. This function for strain in classical multidimensional scaling is as follows from Borg p.262 [3]:

$$Strain(\mathbf{X}) = \quad \mathbf{X}\mathbf{X}^T + \frac{1}{2}\mathbf{J}\Delta^{(2)}\mathbf{J}$$

where $\mathbf{X}$ is defined as the point coordinate matrix, $\mathbf{J}$ is the centering matrix, and Δ is the matrix of dissimilarities (i.e. distance matrix) with values squared. For clarity purposes, we will outline these matrices as follows [3]:

$$\mathbf{J} = I_n \quad \frac{1}{n}(I_n \quad 1_n \quad 1_n^T)$$

$$\Delta^{(2)} = d_{ij}^2; \ 8d_{ij} \ 2 \ D$$

Note that $\mathbf{I}$ is the identity matrix of the corresponding size. Thus, given these equations, the value of the strain can be computed. As a result, the strain of a point coordinate matrix $\mathbf{X}$ is defined as the L2 norm of the difference between the product of the point coordinate matrix and its transpose and the matrix product of negative half the centering matrix times the distance matrix times the centering matrix. The final steps in the computation is to simply compute the largest eigenvalues for the desired number of dimensions to be output and plot along those dimensions.

2.2.5 Clustering Techniques

In order to quantitatively compute the differences between L1 and L2-UniFrac, generating clustering reports will be necessary. This section will follow closely with section 2.2.6 which assigns scores to the clusters generated in this section using standard methods. When it comes to clustering, there are many forms of clustering algorithms, but for the purposes of this report, we will be focusing on two of the more popular algorithms: agglomerative clustering and k-medoids clustering.

2.2.5.1 Agglomerative Clustering

Agglomerative clustering is a method of clustering that starts by considering every individual point as a singular cluster and then merges them together according to some linkage property until the desired number of clusters are reached [20, 47]. For our implementation, we will be using a complete linkage which seeks to use the maximum distance between the observations of two sets when selecting sets to merge together.

The complete linkage variant of agglomerative clustering utilizes the following principle. Starting with a set of elements S, each element i of that set is assigned to its own cluster c_i among a set of clusters C. Additionally, we will have a similarity matrix A where each a_{ij} symbolizes the similarity between two elements (or in the case of our experimentation, we will use a dissimilarity matrix under a similar (built-in) procedure that handles it directly). In each step, two clusters c_i and c_j will be merged together such that the following equation is maximized:

$$s(c_i; c_j) = \min_{u \in c_i; v \in c_j} a_{uv} \quad [7]$$

That is, we merge the clusters c_i and c_j such that the minimum similarity between elements of each cluster is maximized. In doing so, we iteratively obtain the desired number of clusters which are contained as groups of elements that are considered most similar according to the similarity matrix.

2.2.5.2 K-Medoids Clustering

K-Medoids, on the other hand, is a modified variant of the K-Means clustering algorithm which uses a partitional method, starting with the desired number of groups in the form of random points. Then, iterating through all of the points, the clustering algorithm simply adds each of the new elements to the group with a mean coordinate closest to the new point until all points are exhausted and the groups are formed [26, 20]. K-Medoids uses a slightly different approach to K-Means. Rather than using the mean of each cluster, K-Medoids attempts to minimize the distance to every point in the cluster itself. The rationale for this change is the fact that K-Means suffers from very large impacts caused by outliers within a particular cluster, reducing overall performance [23, 38, 12]. In the algorithm for K-Medoids by Park and Jun, the updating procedure present in K-Means is updated to find the medoid rather than the mean. This is done by computing for every point the sum of distances to every point and minimizing this quantity to select the point which will then be known as the medoid of that cluster [38]. The following steps are effectively the same as K-Means where new points just compare the medoid of the cluster, get added to the closest cluster, and then the new medoid for that cluster is computed.

In more specific terms, the K-Medoids algorithm works in the following manner. Firstly, given a number of desired clusters k and using a dissimilarity matrix D containing pairwise dissimilarities d_{ij} between elements in a set S, we first compute a vector V such that

$$v_i = \sum_{i=1}^{n} \frac{d_{ij}}{\sum_{l=1}^{n} d_{il}} \quad [38]$$

for all $1 \quad j \quad n$. Now, selecting the k smallest elements from V, we can start building our clusters. These first k

points are now considered to be the initial cluster centers or medoid, and we then assign every other point to the nearest medoid group. Now, we begin a cyclical process of computing the medoid of the cluster, using the algorithm above and selecting the element that best minimizes the distances between the elements of the cluster. Then the points are reassigned new clusters according to these updated k medoids and we repeat the medoid computation loop until the medoids stop changing and we reach a base case or the algorithm reaches a defined iteration limit.

2.2.6 Clustering Scoring Algorithms

Now that the clustering algorithms have been introduced, it is also important to introduce the scoring algorithms that will be used to assign a score to assess the clustering performance. The following algorithms were used to score the clusters: Rand Index Score, Adjusted Rand Index Score, Normalized Mutual Info Score, Adjusted Mutual Info Score, and the Fowlkes Mallows Score.

2.2.6.1 Rand Index Score Like all of the other evaluation metrics, the Rand Index score will yield a result between 0 and 1 with 1 being the most precise clustering and 0 being the least precise. This particular score is calculated by computing the total number of times one cluster (the true values of the samples, given by the metadata) agree with the clustering algorithm used (number of pairs assigned the same cluster in both plus the number of pairs assigned different clusters in both) and dividing that by the total number of pairs [43].

Rand gives the following formulation for the Rand Index Score function $c(Y; Y^0)$ where c is the similarity between one cluster Y and another cluster Y^0 across N different points. Also, let n_{ij} denote the number of points in both the i^{th} cluster of Y and the j^{th} cluster of Y^0. Thus, c is equivalent to the following equation.

$$c(Y; Y^0) = \frac{2\left(\binom{N}{2}\left(\sum_{i=1}\sum_{j=1} n_{ij} + \sum_{j=1}\sum_{i=1} n_{ij}\right)^2 - \sum_{i=1}\sum_{j=1} n_{ij}^2\right)}{\binom{N}{2}} \quad [43]$$

In the case, Y would be considered to be the ground truth or the actual clustering given by the metadata, and Y^0 would be considered to be the experimentally obtained clustering through the blind clustering algorithms such as K-Medoids and Agglomerative Clustering. This equation effectively functions to compute the quantities related to the number of points together in both, different in both, and mixed between the two clustering sets. The total is then divided by the total number of possible pairs $\binom{N}{2}$ to get the ratio between 0 and 1 for the similarity index.

To provide better granularity and to better understand that equation, we can also provide another formulation

given by Rand, using a slightly neater notation with specific variables that we will define. This is useful because these same variables can also be used when constructing the definition of the Adjusted Rand Index Score later on. To begin, given two partitions P and L with using N objects with R subsets in P and C subsets in L, and a table T indicating group overlap, we have [50]:

$$a = \frac{\sum_{r=1}^{R}\sum_{c=1}^{C} t_{rc}^2 \quad N}{2};$$

$$b = \frac{\sum_{r=1}^{R} t_{r+}^2 \quad \sum_{r=1}^{R}\sum_{c=1}^{C} t_{rc}^2}{2};$$

$$c = \frac{\sum_{c=1}^{C} t_{+c}^2 \quad \sum_{r=1}^{R}\sum_{c=1}^{C} t_{rc}^2}{2};$$

and

$$d = \frac{\sum_{r=1}^{R}\sum_{c=1}^{C} t_{rc}^2 + N^2 \quad \sum_{r=1}^{R} t_{r+}^2 \quad \sum_{c=1}^{C} t_{+c}^2}{2}$$

In this case, a corresponds the number of any pair of objects where that pair is placed in the same group in P and L, b corresponds to the number of any pair of objects where that pair is placed in the same group in P and a different group in L, c corresponds to the number of any pair of objects where that pair is placed in the same group in L and a different group in P, and finally, d corresponds to the number of any pair of objects where that pair is placed in a different group in P and a different group in L [50]. Using this notation, we can then simply state the Rand Index Score (RI) to be the following quantity:

$$\text{RI} = \frac{a+d}{a+b+c+d}$$

This quantity follows from our initial equation but allows for us to see more directly how exactly the equation functions as a ratio of different subsets of categorizations between similarities and differences in the clustering patterns.

2.2.6.2 Adjusted Rand Index Score

The Adjusted Rand Index score, by contrast, is a little more complicated in structure in that it corrects for chance. This process is accomplished by first establishing a baseline measurement which is intended to correct for expected similarity that is seen in order to add weight to differences. The most basic

formulation of this was initially formulated by Hubert and Arabie (1985) which is as follows [43, 18]:

$$adjusted\_rand\_index = \frac{Index \quad Expected\_Index}{Max\_Index \quad Expected\_Index}$$

This form was later adjusted by several other papers to allow clustering evaluation to better reflect other datasets that are not as accurately represented by the permutation model used for adjustment, and many of these are in use in more modern implementations [55, 13].

To formalize this, we can use the variables a, b, c, and d from the Rand Index Score section to form the Hubert and Arabie adjusted Rand Index (ARI) [18, 50]:

$$\text{ARI} = \frac{\binom{N}{2}(a+d) \quad [(a+b)(a+c)+(c+d)(b+d)]}{\binom{N}{2}^2 \quad [(a+b)(a+c)+(c+d)(b+d)]}$$

This is almost identical to the initial formula with just an extra $\binom{N}{2}$ term present in both the numerator and denominator which, in the RI, would just be cancelled out. However, in this case, it is necessary which is why it is now present. The expression $[(a+b)(a+c)+(c+d)(b+d)]$ is equivalent to the expected index and adjusts for chance. The rest of this equation remains the same as the previous expression for the RI, so it can be left as such.

2.2.6.3 Normalized Mutual Info Score

Next, the Normalized Mutual Info Score relates to the mutual information between two particular sets–in this case, the predicted clustering and the actual clustering information–and normalizes it using the square root of the product of the individual entropy of the predicted clustering and the actual clustering and relates to the Pearson Correlation Coefficient [51]. This process in the discrete case is formalized by Thomas and Joy [6] where the mutual information is a double summation of the probability mass function between the first set times the log of the probability mass function divided by the marginal probability product of the two sets. The individual entropy, by contrast, is calculated by computing the negative sum of the probability of an event in a discrete variable occurring times the log of the probability of that same event occurring [49, 48].

Formally speaking, let construct a matrix N where the rows correspond to clusters in a partition A and columns correspond to clusters in a partition B. Index N using the form N_{ij} to index into row i and column j. Each N_{ij} then corresponds to the number of objects in both cluster i of A and cluster j of B such that the rows sum to N_i and the columns sum to N_j. Finally, let c_A and c_B equal the number of clusters in A and B respectively. We can then express the mutual information in the following manner [34, 24, 2]:

$$\text{MI(A,B)} = \sum_{i=1}^{|A|} \sum_{j=1}^{|B|} \frac{N_{ij}}{N} \log \frac{N_{ij} N}{N_i N_j}$$

However, this is just the standard metric for mutual information, but this experimentation will make use of the normalized variant. The difference being that normalized mutual information, according to the initial authors, attempts to optimize the mutual information score with regards to its criteria [2]. Their proposed equation for Normalized Mutual Information is as follows [24, 2]:

$$\text{NMI(A,B)} = \frac{2 \sum_{i=1}^{|A|} \sum_{j=1}^{|B|} N_{ij} \log \frac{N_{ij} N}{N_i N_j}}{\sum_{i=1}^{|A|} N_i \log \frac{N_i}{N} + \sum_{j=1}^{|B|} N_j \log \frac{N_j}{N}}$$

The primary difference between NMI and MI is the multiplication by $\frac{2N}{\sum_{i=1}^{|A|} N_i \log \frac{N_i}{N} + \sum_{j=1}^{|B|} N_j \log \frac{N_j}{N}}$. This term can be considered to be the normalization factor where the denominator, in particular, divides the score by the sum of rows in partition A times the logarithm of their terms over the entire matrix plus the sum of columns in partition B times the logarithm of their terms over the entire matrix. The $2N$ in the numerator then adjusts for the division by N in the MI score to divide by the new denominator instead as well as the 2 which is left after their simplification.

2.2.6.4 Adjusted Mutual Info Score

The fourth metric that we will be using to score our clustering performance will be the Adjusted Mutual Info Score. Similar to its normalized variant, it will use the aforementioned computation for mutual information. However, in this case, it will also compute an expected mutual information for the set, thus also accounting for chance in the process. This method simply computes the maximum of the two and uses that. Both the mutual information and the maximal entropy have the expected mutual information subtracted to yield a score. Scores closer to 0 in this case indicate more random results or, in other words, results that yield a mutual information similar to the expected mutual information for a random cluster. Scores closer to 1 indicate more identical sets [55].

2.2.6.5 Fowlkes Mallows Score

Finally, the last metric that we will be using to evaluate the performing of our clustering will be the Fowlkes Mallows Score (FM). This metric is relatively straight-forward to compute and is related to the number of true positives, false positives, and false negatives in the clustering analysis. Formally, this can be given by the square root of the product of two criteria, each denoted as $W_I(C_1; C_2)$ and $W_{II}(C_1; C_2)$ where the first is the number of point pairs that are in the same cluster under both C and C^0 divided by a quantity that

equates to the above quantity and the number of false positives. The second is very similar except at the end, it substitutes false positives for false negatives [11, 15]. In the end, we would expect Fowlkes Mallows to perform better than metrics such as the Rand Index in cases where the data is more unrelated as it factors in such conditions of false negatives and positives that may occur more robustly [15].

Interestingly enough, it is also possible to simply state the Fowlkes Mallows Score in terms of the a, b, c, and d variables defined in the Rand Index section for a more simplistic representation. We can rewrite this expression formally as [50, 15]:

$$\text{FM} = \frac{a}{\sqrt{(a+b)(a+c)}}$$

Notice how the above equation does not include d anywhere. This is because the Fowlkes Mallows Score is only based in terms of the number of true positives, false positives, and false negatives as noted before. The term d, however, as Steinley notes, is implicitly stored in the denominator as a component of this product of summations [50]. As can be see, however, the FM score effectively divides the number of matching pairs in both subsets by the square root of the product of the criteria that implicitly include d. For better understanding, we can undo the square root on the top term to have a better picture with the following equation:

$$\text{FM} = \frac{\sqrt{\frac{a^2}{(a+b)(a+c)}}}{} = \frac{\sqrt{\frac{a}{(a+b)}\ \frac{a}{(a+c)}}}{}$$

The term $\frac{a}{(a+b)}$ effectively represents the fraction of results where the pairs that are in the same group in P also happen to be in the same group in L, and the term $\frac{a}{(a+c)}$ represents the fraction of results where the pairs that are in the same group in L also happen to be in the same group in P.

2.2.7 Krona

Another tool that will be used in this paper for evaluation purposes will be Krona, otherwise known as KronaTools. KronaTools is a bioinformatics package containing numerous scripts with the purpose of allowing for better metagenomic visualizations to be generated in an interactive manner [36]. Krona supports a wide range of file imports, including BLAST, text, XML, NCBI Taxonomy IDs, and many other file formats. For this paper, the primary file format of interest is just the basic text file. The text file is structured in a tab-separated format with each line corresponding to one particular OTU's taxonomy. The first entry on each line is the abundance of a particular OTU

within the sample. Following the abundance, every line entry afterwards is simply reserved for the taxonomy or other form of classification, from kingdom through species, each separated by tabs.

Importing this text file is as simple as generating a bash subprocess using the command ktImportText, followed by the text file name, followed by other parameters such as -o to specify an output file. The result of this process will be an HTML5 file that enables taxonomic exploration in an interactive manner. For instance, assume the outermost kingdom is bacteria. Simply double-clicking on bacteria will expand the bacteria subgraph and display a krona chart that only displays elements that are contained within the bacteria section. This is rather useful for analysis purposes as the total abundance of a particular taxonomic classification can be directly ascertained as a sum of all OTU's containing that taxonomic level. For instance, assume a researcher would like to determine the percentage of a sample that is a bacteria. To do this, one would need to look at the classification and abundance of each OTU and sum up the abundance of every sample in the kingdom of bacteria. Using Krona, however, will automatically allow them to import their raw OTU data and simply click through to the particular taxonomic level of interest—or by using the built-in search function to quickly highlight relevant matches—to grab the total abundance.

3 Methods

This section outlines the implementation of various key functions and techniques leveraged in this paper. For each relevant section, we will discuss in detail the pseudocode or process used to compute the respective results and show its correspondance to L2-UniFrac in the case of section 3.3 and section 3.4.

3.1 Characterization of L1 and L2 UniFrac

From the prior work mentioned earlier, it was found that the L1-norm varient of UniFrac allowed for the application of the UniFrac metric to medians across groups. We will now more formally lay out that characterization in terms of the Wasserstein metric (also known as the Kantorovich-Rubinstein metric or the Earth-Mover's Distance). In the paper on Striped UniFrac's supplementary materials, the following equation characterizes the relationship between the UniFrac metric and L1 averages [32].

$$\text{UniFrac}(P;Q) = k\,W(P \quad Q)\,k_{L1} \tag{3.1}$$

$$= \sum_{i=1}^{} \sum_{j=1}^{} l(e_i)v_i(j)\,|P(v_j) \quad Q(v_j)| \tag{3.2}$$

As can be noted, this formulation precisely represents EMDUniFrac, otherwise denoted as L1-UniFrac. This quantity reflects the summation of the difference between probability distributions P and Q multiplied by the corresponding edge weight leading to subtree i, given as the quantity $l(v_i)$, and multiplied by the corresponding position in the basis vector v_i of the root of that subtree. However, this expression is useful only for L1-UniFrac. For L2-UniFrac, we must turn to another paper by Dr. McClelland that characterizes the following form of the Wasserstein metric for L2-norm [28]. This is done by first equating UniFrac with respect to the L2 norm with the scaled Wasserstein metric, given by $d_{UFL2}(P;Q)^2 = k\,W^{p}\text{-}(P \quad Q)\,k_{L2}$ with the first term representing L2-UniFrac and the second term with respect to the Wasserstein metric. This term can then be expanded according to Dr. McClelland to the following:

$$k\,W^{p}\text{-}(P \quad Q)\,k_{L2} = \sum_{j;k2T} P(j)Q(k)\;\; d(j;k) \quad \frac{1}{2}\left(\sum_{j;k2T} P(j)P(k)\;\; d(j;k) + \sum_{j;k2T} Q(j)Q(k)\;\; d(j;k)\right) \tag{3.3}$$

Put simply, this equation effectively subtracts the evolutionary spread between members of each particular sample P and Q from the evolutionary spread between P and Q [28]. Although the primary usage of this formulation is for computationally efficient Double Principle Coordinate Analyses (DPCoA), this formula illustrates some un-

derstanding of the biological context behind some of these ideas. This is demonstrated in that the first summation quantity of the term corresponds to the weighted distances between community members, and each of the subsequent summations take the weighted distances between each of the each respective communities' members itself totaled together and arithmetically averaged. In simplified terms, this equation directly relates the difference between the biological spread between two communities and the average spread within each respective community itself. The first term on the left side represents the L2 formulation of UniFrac, with the primary difference from the L1 formulation being the scaling by slightly different Wasserstein matrix and an L2-norm at the end. The different Wasserstein matrix is instead computed using reciprocal edge-length scaling on rows.

This characterization of the UniFrac metric in terms of the discrete 2-Wasserstein metric and a summation of differences allows us to consider portions of the equation at a time, rather than in its entirety. For instance, it is possible to compute and store either of the two summations on the right side in advance to reduce times later on since they only depend on themselves. This alternative approach is noted [28] in that:

$$d_{UFL2}(P;Q)^2 = \sum_{i \in T} (w_i^t P \quad w_i^t Q)^2 \tag{3.4}$$

This formulation gives additional justification behind L2-UniFrac. In essence, the square of L2-UniFrac is equal to the sum of the squares of the differences between $w_i^t P$ and $w_i^t Q$. These two terms represent each represent $\frac{1}{2}$ of the sum of products of all pairwise probabilities multiplied by the sum of the distances between the each of the two pairwise nodes and the selected root node minus the distance between them. Formally written, with only loose manipulation of the equations from Dr. McClelland's dissertation, this comes out to the following equation without loss of generality with the trivially expanded case below for clarity purposes [28].

$$\sum_{i \in T} (w_i^t P)^2 = \frac{1}{2} \sum_{j,k \in T} P(j)P(k) \quad (d(j;\ ) + d(k;\ ) \quad d(j;k)) \tag{3.5}$$

$$\sum_{i \in T} w_i^t P = \sqrt{\frac{1}{2} \sum_{j,k \in T} P(j)P(k) \quad (d(j;\ ) + d(k;\ ) \quad d(j;k))} \tag{3.6}$$

We will consider this expression in eq. (3.6) as the preprocessed, or pushed-up, quantity of L2-UniFrac. The most important characteristic about this equation now is the ability for one to take a single sample, process it through eq. (3.6), and later apply equation eq. (3.4) with some manipulation to separate the squared binomial into a sum of three monomials, each consisting of only $(w_i^t P)^2$, $(w_i^t Q)^2$, or the square root of either. This construction allows for

potential preprocessing of probability vectors given by samples prior to computing L2-UniFrac. Furthermore, since the sum of these differences yield the square of L2-UniFrac, the final L2-UniFrac distance can be obtained by taking the square root of both sides of eq. (3.4) along with some substitutions from eq. (3.6) as follows:

$$d_{UFL2}(P;Q) = \sqrt{\sum_{i \in T} (w_i^t P \quad w_i^t Q)^2} \tag{3.7}$$

$$= \sqrt{\sum_{i \in T} (w_i^t P)^2 + (w_i^t Q)^2 \quad 2(w_i^t P)(w_i^t Q)} \tag{3.8}$$

Most crucially, however, we can directly rearrange terms in section 3.1 to obtain the terms WP and WQ using simple distribution. This is the direct characterization of the aforementioned setup in eq. (3.4) but in terms of matrices. Both of these considerations will be useful in future proofs within this paper, but WP and WQ directly show that simply scaling by the edge length matrix is sufficient to preprocess vectors. Additionally, with the terms W^p-P and W^p-Q obtained from the same process for eq. (3.3), we obtain the same preprocessing notion as before but for L2-UniFrac instead.

The necessary substitutions should now be very evident in eq. (3.7), given the expansions in eq. (3.5) and 2.6. As a result, we look to propose an algorithm to efficiently preprocess samples using the mathematical basis outlined in eq. (3.6) and some software optimizations to better utilize existing data structures, an algorithm to invert this process, and an implementation of eq. (3.7) for similar functionality to EMDUniFrac with respect to the L2-norm. In addition to our algorithmic implementations of these functions, we will outline a process to take averages with respect to such quantities.

3.2 Closure Under Averages

We can show this possibility mathematically by considering UniFrac space as an abstract convex hull. When a set of vectors are pushed up into UniFrac space using a transformation similar to that in section 3.3 and corresponding to that proposed in section 3.1, only those vectors contained within the convex hull containing all pushed-up vectors is a valid vector within that space. One potential consequence of a vector being outside of this convex hull is the possibility that, when pushed down, negative abundances appear.

Proposition 1. *It is possible for the component-wise median of vectors forming the convex hull or are otherwise contained within the convex hull to result in an output vector existing outside of the initial convex hull. Moreover, the median of transformed probability vectors is not necessarily a probability vector itself after the inverse transformation.*

Proof of Proposition 1. Consider a set of vectors $p_1' = (2; 0; 0)$; $p_2' = (1; 2; 3)$; and $p_3' = (0; 0; 2)$. The component-wise median of these vectors is defined simply as $p_4' = (1; 0; 2)$. Now, compute the determinant of the coordinate matrix of these four coordinates. If it is zero, then p_4' must lie on the plane spanned by $p_1' = (2; 0; 0)$; $p_2' = (1; 2; 3)$; and $p_3' = (0; 0; 2)$. Otherwise, p_4' cannot be in the convex hull.

$$\det \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \det \begin{bmatrix} 2 & 1 & 0 & 1 \\ 0 & 2 & 0 & 0 \\ 0 & 3 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \det \begin{bmatrix} 2 & 1 & 0 & 1 \\ 0 & 2 & 0 & 0 \\ 0 & 3 & 2 & 2 \\ 0 & \frac{1}{2} & 1 & \frac{1}{2} \end{bmatrix}$$

$$= \det \begin{bmatrix} 2 & 1 & 0 & 1 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 1 & \frac{1}{2} \end{bmatrix} = \det \begin{bmatrix} 2 & 1 & 0 & 1 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

$$= 2 \; 2 \; 2 \; \frac{1}{2}$$

$$= 4$$

$\therefore$ the component-wise median cannot lie inside the convex hull formed by the three vectors since it is not co-planar and is thus invalid.

□

The second part of this proof (transformation *!* median *!* inverse transformation not being closed) will be deferred until section 5.3 to prove that this process can result in averages that are themselves no longer probability vectors. That is, probability vectors must have all components sum to 1 and each individual component be bounded between 0 and 1. If one of these is violated, as we will see later, the result is no longer a probability vector.

However, in this section, there are a few other things to note. Firstly, another trivial example using strictly probability vectors can be seen with $p_1' = (1; 0; 0)$; $p_2' = (0.17; 0.33; 0.5)$; and $p_3' = (0.33; 0.5; 0.17)$. The determinant

of these three vectors and the median, using the same process, yields -0.03 which is non-zero, thereby indicating the result is not on the plane. Additionally, it is trivial to see in this case that with $\vec{p}_4 = (0.33, 0.33, 0.17)$, $\sum_{i \in p_4} i = 0.83 \neq 1$. Therefore, the median of these three probability vectors is not itself a probability vector.

We can also formalize the W and W^{p}- matrices as well for these examples for a more concrete basis. Consider $\vec{p}_1 = (1, 0, 0)$; $\vec{p}_2 = (0.17, 0.33, 0.5)$; and $\vec{p}_3 = (0.33, 0.5, 0.17)$ from earlier. All of these are probability vectors that sum to 1. We construct a distance matrix for a sample phylogenetic tree. Since we only express the probabilities on three nodes, our probability tree therefore must have three nodes, shown below simply using nodes A, B, and C.

If we now consider the edge weights, let the edge between B and A have a length of 0.4 and the edge between C and A have a length of 0.6, and it is understood that the branch above node A has unit length. Thus, using the preprocessing scheme derived from Dr. McClelland's work, we obtain the following matrix W and W^{p}- for the preprocessed vectors.

$$W = \begin{bmatrix} 0.6 & 0 & 0 \\ 0 & 0.4 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$W^{p} = \begin{bmatrix} 0.774600 & 0 & 0 \\ 0 & 0.632456 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Note that the vector W^{p}- exactly corresponds to W with the square root operation applied to all elements of the matrix, as expected. These matrices can be constructed using the edge weights directly but were independently verified with a system of equations.

Additionally, the inverse transformation matrix of W can be defined as the matrix below. The inverse of W^{p}- can be calculated in the same manner. We see the same square root correspondance here too.

$$W^{-1} = \begin{bmatrix} \frac{5}{3} & 0 & 0 \\ 0 & 2.5 & 0 \\ \frac{5}{3} & 2.5 & 1 \end{bmatrix}$$

$$W_{p'}^{-1} = \begin{bmatrix} 1.291 & 0 & 0 \\ 0 & 1.581 & 0 \\ 1.291 & 1.581 & 1 \end{bmatrix}$$

With that said, if we explicitly follow the process outlined before of $W^{-1}Wp'_4 = p'_4$, we get the vector (0.33, 0.33, 0.34) which does follow the constraints put on probability vectors. As it turns out, counterexamples for this are very rare but do exist. We will follow this process and show that 4 out of 5 body sites in section 5.3 are indeed counterexamples to this problem.

Now, for the sake of completeness, let us consider the component-wise mean of $p'_5 = (1, \frac{2}{3}, \frac{5}{3})$. Let us compute the determinant of this point to see if it lies within the convex hull spanned by these points.

$$\begin{aligned}
\det \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} &= \det \begin{bmatrix} 2 & 1 & 0 & 1 \\ 0 & 2 & 0 & \frac{2}{3} \\ 0 & 3 & 2 & \frac{5}{3} \\ 1 & 1 & 1 & 1 \end{bmatrix} = \det \begin{bmatrix} 2 & 1 & 0 & 1 \\ 0 & 2 & 0 & \frac{2}{3} \\ 0 & 3 & 2 & \frac{5}{3} \\ 0 & \frac{1}{2} & 1 & \frac{1}{2} \end{bmatrix} \\
&= \det \begin{bmatrix} 2 & 1 & 0 & 1 \\ 0 & 2 & 0 & \frac{2}{3} \\ 0 & 0 & 2 & \frac{2}{3} \\ 0 & 0 & 1 & \frac{1}{3} \end{bmatrix} = \det \begin{bmatrix} 2 & 1 & 0 & 1 \\ 0 & 2 & 0 & \frac{2}{3} \\ 0 & 0 & 2 & \frac{2}{3} \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
&= 2 \ 2 \ 2 \ 0 \\
&= 0
\end{aligned}$$

Since this evaluates to zero, it is known for certain that this point lies on the plane spanned by these three points. However, this does not by itself prove that it lies within the convex hull of those three points since this is just a single example. Instead, this result must be proven for all possible convex hulls and their respective component-wise mean. It is trivial to prove the case where it is possible for medians to lie outside, but to prove that means must always lie inside takes a slightly more involved path. We will move to prove this formally now.

Proposition 2. *The transformation of the probability simplex, and moreover the convex hull formed by probability vectors, by an invertible matrix is closed under component-wise means.*

With more verbosity, the mean of a set of n probability vectors, spanning an (n-1)-simplex space P^{n-1}, must also be contained within P^{n-1}: Let us define a function $M : R^{n \times m} \to R^n$ which takes a set of vectors and outputs the component-wise mean of those vectors. Given the following set of point vectors

$$V = ((x_1^1; x_2^1; \quad ; x_m^1); \quad ; (x_1^n; x_2^n; \quad ; x_m^n)); \ \forall v_i \in V; v_i \in R^n$$

with subscripts denoting column and superscripts denoting row, the function yields the following vector Y.

$$Y = M(V) = (\frac{x_1^1 + x_1^2 + \quad + x_1^n}{n}; \frac{x_2^1 + x_2^2 + \quad + x_2^n}{n}; \quad ; \frac{x_m^1 + x_m^2 + \quad + x_m^n}{n})$$

Therefore, given a set of n points forming the probability simplex P^{n-1}, one must show that $Y \in P^{n-1}$: Let us briefly considered the geometric perspective of a generalized simplex in a non-rigorous manner. First consider the base case of a 0-simplex. In such a case, the component-wise mean is simply that exact point. Since it trivially lies within the 0-simplex, this shows that 0-simplexes are closed under the component-wise mean. Now, consider a 1-simplex, consisting of two points p_1 and p_2. The component-wise mean, in this case, is simply computed as $\frac{p_1+p_2}{2}$ which is the midpoint of a line segment, thus showing that the 1-simplex is closed under the component-wise mean. Continuing this process to two dimensions, and we see the formula for the centroid of a triangle emerge which is known to always lie inside the triangle. In 3-dimensions, this mean is given as the centroid of a tetrahedron which too must lie inside its hull. In fact, we can formulate a loose inductive generalization of this into (n-1)-simplex space for n vectors using the following formulation.

$$Y_n = \frac{1}{n} \sum_{i=1}^{n} v_i$$

This precisely describes the formula for a component-wise mean and will come up formally later. Continuing on, if $Y_n \in P^n$, then this must inductively show that $Y_{n+1} \in P^{n+1}$. Since at least n vectors span the n-simplex space, it is also known that n+1 vectors must span the corresponding (n+1)-simplex space. We can thus reformulate our equation inductively using the above equation as follows:

$$Y_{n+1} = \frac{n}{n+1} Y_n^{\emptyset} + \frac{v_{n+1}}{n+1}$$

This formulation then effectively describes a linear shift along a new dimension that is linearly independent from all other existing dimensions due to the fact that an n-simplex forms a convex hull in n-dimensions and any new point must also be on the convex hull, implying it should form a basis in the new dimension. As such, the new point Y_{n+1} is located somewhere between the centroidal position of the n-simplex and the new vector along the vector between the n-centroid and the (n+1)-vector to form a (n+1)-simplex. As such, Y_{n+1} should also be located inside of the (n+1)-simplex.

Now that the rationale behind a generalized simplex is clear, we can introduce some restrictions that have been mentioned earlier that are specific to UniFrac to obtain a more rigorous proof. To start, we need only show that the probability simplex is closed under component-wise means in the case where the probability simplex is first transformed with an invertible matrix, given as the "push-up" of the sample vectors; that is, all vectors $8v_j^i \ 2 \ V; jv_j j = n; jV j = m$ forming the transformed simplex share a property that the inverse transformation of those vectors uphold $\sum_{i=1}^{n} v_j^i = 1$. As a result, should we show that the component-wise mean of a set of transformed probability vectors will always give a probability vector when applied to an inverse transformation after the mean operation, the result is guaranteed to be inside the probability simplex.

Proof of Proposition 2. Consider a set of vectors $v = fv^1; : : : ; v^m g$ where each $v^i \ 2 \ \mathrm{R}^n$ is a member of the probability simplex: $0 \quad v_j^t \quad 1$ and $\sum_{j=1}^{n} v_j^t = 1$. Consider some invertible transformation matrix W of size $m \quad n$. Finally, from section 3.1, we recall from Dr. McClelland that W is an invertible matrix for use in UniFrac. The transformation of a vector v^t, otherwise denoted as $(Wv^t)_i$, is defined as

$$(Wv^t)_i = \sum_{j=1}^{} w_{ij} v_j^t :$$

Consider the component-wise mean function $M : \mathbb{R}^{m \times n} \to \mathbb{R}^n$ defined via

$$M(v^1, \ldots, v^m)_i = \frac{1}{m} \sum_{t=1}^{m} (Wv^t)_i \tag{3.9}$$

$$= \frac{1}{m} \sum_{t=1}^{m} \sum_{j=1} w_{ij} v_j^t \tag{3.10}$$

$$= \frac{1}{m} \sum_{j=1} \sum_{t=1}^{m} w_{ij} v_j^t \tag{3.11}$$

$$= \frac{1}{m} \sum_{j=1} w_{ij} \sum_{t=1}^{m} v_j^t \tag{3.12}$$

$$\tag{3.13}$$

Now, let u be a vector such that

$$u_j = \sum_{t=1}^{m} v_j^t$$

Observe now that in aggregate over all values of i, W corresponds to the quantity $\sum_{j=1}^{n} w_{ij}$ when applied to v_j, we find the quantity:

$$M(v^1, \ldots, v^m)_i = \frac{1}{m} Wu$$

This quantity represents the transformed set of vector mean. Now, observe that:

$$W^{-1} M(v^1, \ldots, v^m)_i = W^{-1}(\frac{1}{m} Wu) \tag{3.14}$$

$$= W^{-1} W \frac{1}{m} u \tag{3.15}$$

$$= \frac{1}{m} u \tag{3.16}$$

Thus the summation of the inverted mean scaled set of vectors is defined by the following:

$$\sum_{i=1} W^{-1} M(v^1, \ldots, v^m)_i = \sum_{j=1} \frac{1}{m} u \tag{3.17}$$

$$= \sum_{j=1} \frac{1}{m} \sum_{t=1} v_j^t \tag{3.18}$$

$$= \frac{1}{m} \sum_{j=1} \sum_{t=1} v_j^t \tag{3.19}$$

$$= \frac{1}{m} \sum_{t=1} \sum_{j=1} v_j^t \tag{3.20}$$

$$= \frac{1}{m} \sum_{t=1} 1 \tag{3.21}$$

$$= \frac{1}{m} m \tag{3.22}$$

$$= 1 \tag{3.23}$$

Thus the quantity $W^{-1} M(v^1, \ldots, v^n)$ is also a member of the probability simplex. □

As a result, we have shown that the inverted mean of transformed probability vectors lies strictly within the probability simplex or convex hull formed by those probability vectors. Furthermore, we have shown that for this proposed L1-norm technique, it is possible for it to yield vectors outside of probability distribution's hull, leading to potential negative abundances in certain circumstances when the vector is moved outside of UniFrac space. As a result, EMDUniFrac suffers from negative elements in averaged probability vectors. As part of this paper, we will experimentally verify this to be true in section 4.2. Finally, from this proof, it is important to note that the quantity Wv precisely corresponds to the set of pushed-up or preprocessed vectors with respect to the L2-UniFrac metric as will be outlined later.

3.3 Push Up

L2-UniFrac is designed with the ability to preprocess probability vectors in mind using a method known as "push up." This notion is derived from the left part of eq. (3.3) where we get W^p-P and W^p-Q as two "preprocessed" vectors. However, in order to preprocess efficiently, there are several primary considerations. The first consideration is the information we will have available. This includes a probability vector P, a dictionary of ancestors $Tint$, a dictionary of edge lengths $lint$, and a list of nodes from the bottom to the top of the phylogenetic tree $nodes\_in\_order$.

If we were to apply the matrix-vector multiplication of W^p- directly, we would need to first construct the matrix and then perform the multiplication. Construction could be achieved by iterating through *nodes_in_order* and, for each node, adding the weight leading to all ancestors of that node at the corresponding index in the column of W^p-. This process would take, in the worst case scenario, $O(n^2)$ time since nodes can be all descendants of each other. With several hundred thousand OTUs or more possible, it is clear that this process would be very computationally demanding. With that said, the matrix multiplication process is very efficient since such operations are highly optimized. Additionally, it is important to note that the $O(n^2)$ time only ever occurs once and obtains a W^p- for the phylogenetic tree that can be used directly to preprocess many vectors efficiently. Therefore, this process is likely optimal when many samples need to be preprocessed and non-optimal when only a single or handful of samples need to be preprocessed. The algorithm pseudocode to construct W^p- is as follows in algorithm 1:

Algorithm 1 Build W^p-

```
Inputs:
        Tint  ! Ancestor dictionary
        lint  ! Edge length dictionary
        nodes_in_order  ! List of nodes bottom to top
function Build_W2(Tint, lint, nodes_in_order)
    n     jnodes_in_orderj
    W^p-     [[0...0]...[0...0]] of size n    n
    for i = 1 to jnodes_in_orderj do
        cur_node     i
        while cur_node 6= n do
            if cur_node is a key in Tint then                    . Add root edge length to ancestors on column i
                W^p-[cur_node, i]     p lint[cur_node; Tint[cur_node]]
                cur_node     Tint[cur_node]
            else                                                 . Root node; 100% of edges are below it
                W^p-[cur_node, i]     1
                cur_node     cur_node + 1
            end if
        end while
    end for
    return W^p-
end function
```

This algorithm simply seeks to construct the W^p- matrix that is a core component of the mathematics in section 3.1. As mentioned earlier, when the number of samples to preprocess is very large, this method helps to optimize the preprocessing by utilizing efficient matrix data structures and their operations. However, this approach is only efficient for when the total number of OTUs is small since it takes quadratic time with respect to the number of OTUs for the preprocessing phase.

As such, an alternative method that we propose for L2-UniFrac's push-up procedure is designed to take one input

at a time to preprocess with respect to the entire tree. The core component of the prior algorithm multiplied the mass (probability) at each OTU by the edge weights leading to every ancestor and summed it together. In effect, we obtained the sum of weights scaled by the edge length on each OTU. In other words, we have the edge length leading to the entire subtree defined by each OTU scaled by the collective mass of the entire subtree. Therefore, we can introduce an optimization to this algorithm to more efficiently compute individual preprocessed samples without constructing the entire W^p- matrix. The entire procedure can be seen in the following pseudocode:

Algorithm 2 L2 Push Up Function

```
Inputs:
        P  ! Sample probability vector
        Tint  ! Ancestor dictionary
        lint  ! Edge length dictionary
        nodes_in_order  ! List of nodes bottom to top
epsilon    2 ^ 52                                  . Minimum representable floating point
function Push_Up(P, Tint, lint, nodes_in_order)
    P_Pushed    P + 0                              . Preserve P with new variable
    for i = 1 to jnodes_in_orderj   1 do
        if lint[i, Tint[i]] is 0 then
            lint[i, Tint[i]]    epsilon            . Make everything nonzero but minimal
        end if
        P_Pushed[Tint[i]] += P_Pushed[i]           . Push mass (probabilities) up
        P_Pushed[i] *=   p lint[i, Tint[i]]        . Multiply root of edge length down
    end for
    return P_Pushed
end function
```

In the above pseudocode, the parameters required include a probability distribution P, a dictionary of each node's ancestor within the phylogenetic tree $Tint$, a dictionary of all edge lengths along the tree $lint$, and a list of all nodes in order of the level-by-level traversal going up the tree with the first elements corresponding to the elements deepest in the tree and the last elements corresponding to those near and including the root. For every node excluding the root, we then push the probability up the tree to the ancestors and scale the children vectors by the root of the distance between them. This process effectively forces the total mass recorded at every node to coincide with the mass of the subtree formed by that node by the time we reach that node in order before multiplying by the root edge length as before. In effect, we obtain a linear time algorithm with respect to the number of OTUs to preprocess individual samples.

Of course, as alluded to earlier, this algorithm would naturally be less efficient in comparison to algorithm 1 when the number of OTUs is low (low upfront computational cost) and the number of samples is high. Building W^p- first will result in a single $O(n^2)$ instance followed by m highly optimized matrix dot product operations for each of m

samples on n OTUs. Strictly using push-up, on the other hand, will result in m total $O(n)$ operations which are not as well-optimized as simple matrix dot products in Python. However, if implemented in other languages that more efficiently handle loops such as C, Rust, or Julia, it could be the case that this difference becomes minimal if not negligible. The push-up procedure in algorithm 2 will be the primary focus of the rest of this work since many OTUs will be used.

The core function of push-up introduces an optimization to that of eq. (3.6), however. Note that in eq. (3.6), the summation is over all j and k in T in a pairwise manner. More directly, we are implementing the matrix-vector product from eq. (3.3) as described earlier. Such a process, if implemented directly would result in quadratic complexity by using pairwise abundances. Instead, we can keep track of the mass of every subtree at each stage, given as the sum of the mass at all other points below it, and scale by the distance from that subtree to its parent node. Since we no longer need to compute all pairwise quantities to the root of the subtree by storing the values as we move up the tree, we greatly simplify the implementation to that of algorithm 2 with respect to individual samples. As a result, moving up the tree, we simply accumulate mass on the dictionary and scale that mass of each entire subtree we encounter by the square root of the edge leading into that subtree to obtain the final "pushed-up" vector.

3.4 Inverse Push Up

Where the push-up method pushes mass from child nodes to ancestor nodes and scales the child by the square root of the edge length leading to it to obtain a preprocessed vector, inverse push-up does the exact opposite. Instead, we wish to re-scale the mass at each index by dividing by the root edge length and then pull that quantity of mass down from the ancestor node directly above. This exactly counteracts push-up to obtain an inversive operation to bring preprocessed vectors back into a probability vector form.

As with earlier, there exist two different approaches to do this operation. The first of these is by computing $W^p{}^{-1}$ and multiplying it to the pushed-up vector p like $W^p{}^{-1}p$. The other approach is to perform this operation iteratively by exactly inversing the push-up operation at each node. There exist pros and cons to each approach similar to the prior section.

We will begin with the $W^p{}^{-1}$ method. This approach assumes we already computed W^p at some point in the past. Once we have that, inverting it is as simple as inverting the matrix. We know that the matrix will always be invertible since phylogenetic tree weights are strictly non-zero, which means that the diagonal of the W^p matrix will

be strictly non-zero and the matrix will be triangular since ancestor nodes can only be of a higher number (lower row) compared to their respective descendant node.

The algorithm to find W_P^{-1} is as follows:

Algorithm 3 Compute W_P^{-1}

```
Inputs:
        W2  ! Preprocessing Wasserstein matrix with indexes scaled by the square root.
function Inverse_W2(W2)
    inv_W2     (W2)  1                                          . Simple matrix inversion
    return inv_W2
end function
```

With that said, using this approach has several major disadvantages of note. The first of these is the fact that matrix inversion is generally a very computationally demanding task. In fact, the best time complexity yet achieved in terms of matrix multiplication is bounded by $O(n^{2:37286})$ [1]. For data sets with hundreds of thousands of OTUs, the total number of operations is enormous, ignoring scalars. For instance, at 600,000 OTUs, the number of operations is some scalar times $5{:}13407 \ \ 10^{13}$ or over 51 trillion operations. Naturally, this process does not scale well and proves prohibitively expensive. Therefore, we propose the following alternative algorithm that instead computes the inverse of individual preprocessed vectors in linear time with respect to the same quantity (number of OTUs).

The algorithm for this is as follows:

Algorithm 4 L2 Inverse Push Up Function

```
Inputs:
        P  ! Pushed-up probability vector
        Tint  ! Ancestor dictionary
        lint  ! Edge length dictionary
        nodes_in_order  ! List of nodes bottom to top
epsilon     2  52                                  . Minimum representable floating point
function Inverse_Push_Up(P, Tint, lint, nodes_in_order)
    P_Pushed      [0; 0; :::0] and of size jPj      . Initialize array of 0 of size jPj
    for i = 1 to jnodes_in_orderj     1 do
        if lint[i, Tint[i]] is 0 then
            edge_length      epsilon                . Make edge length nonzero but minimal
        else
            edge_length = lint[i, Tint[i]]          . Carry edge length over
        end if
        p_val      P[i]                             . Duplicate probability of current node
        P_Pushed[i] += p_val/edge_length            . Undo push-up edge length adjustment
        P_Pushed[Tint[i]] -= p_val/edge_length      . Pull above mass down from ancestor
    end for
    P_Pushed[-1] += P[-1]                           . Include the root in the output
    return P_Pushed
end function
```

As opposed to the operations performed in push-up, we can clearly see that inverse push-up simply reverses it. For every subtree, we simply push down the mass divided by the reciprocal of the edge length to the subtree's root node and subtract the same quantity from its ancestor. In doing so, we undo the earlier scaling operation by dividing by the square root of the edge length where before we multiplied, and we push down the mass that was moved up earlier by subtracting it from the ancestor and adding it back to the child. It is important to note, however, that in this implementation, we start with a zero vector and assign masses back to it based on those of the pushed-up vector. This was done to avoid an in-place procedure as it could result in incorrect `p_val` variables later on. Using this separate list eliminates this problem and preserves the values. As a result, we can see that the two functions are equivalent, and any probability vector pushed-up and then inverse pushed-up will result in the original vector again by assertion.

We can see some clear advantages to using this inverse-push-up approach over computing the W_{P-}^{-1} matrix. The time for single samples is cut down to $O(n)$ where n is the number of OTUs and the time for m samples is cut down to $O(n \quad m)$. Additionally, finding W_{P-}^{-1} also requires the overhead of necessitating the finding of W_{P-} first which takes $O(n^2)$ time to solve whereas pushing up m vectors takes $O(n \quad m)$ time to compute as well. Naturally, when this is used for a single experiment and $m \quad n$ (most cases), push up and inverse push-up are ideal. When $m > n$, W_{P-} and inverse push-up are likely ideal. There are few cases where computing W_{P-}^{-1} is worthwhile since, on small phylogenetic trees, inverse push-up is very fast already, so the difference would be negligible even if the number of samples was very large. Moreover, on large phylogenetic trees, it is clear that the $O(n^{2:37286})$ matrix inversion operation is prohibitively expensive. As a result, our method for L2-UniFrac preprocessing is more efficient than the past characterization in terms of the W_{P-} and W_{P-}^{-1} matrices.

3.5 Computing L2-UniFrac with Push-Up and Inverse-Push-Up

The exact computation of L2-UniFrac is given by eq. (3.4), or, more precisely, the root of it, shown in eq. (3.7). The L2-norm occurs in that the equation takes the root of the sum of the squared differences between the two preprocessed probability vectors over all nodes in the tree. As a result, this process yields the L2-norm distance between those two vectors. Since we know that this process exactly corresponds to L2-UniFrac according to eq. (3.4), this result is valid.

In our implementation, using push-up to preprocess the nodes, since we have shown that push-up is equivalent to the scaling of the Wasserstein terms, it is evident that subtracting two of such vectors and applying the Frobenius

norm along a single vector, given in our implementation using the numpy.linalg.norm module in python, yields the same result as the aforementioned derivation for L2-UniFrac.

To provide some illustration into this process, a small demonstration of the steps needed to take in pseudocode to compute L2-UniFrac are as follows:

Algorithm 5 L2-UniFrac from Pushed-Up Vectors

```
Inputs:
        P  !  Sample 1 probability vector
        Q  !  Sample 2 probability vector
        Tint  !  Ancestor dictionary
        lint  !  Edge length dictionary
        nodes_in_order  !  List of nodes bottom to top
v1    push_up(P, Tint, lint, nodes_in_order)          . Push up first vector
v2    push_up(Q, Tint, lint, nodes_in_order)          . Push up second vector
L2_UniFrac    k v1    v2 kL2                          . L2-Norm of Vector Difference
```

It is important to note that the ideas presented in section 3.3 are still relevant here with regards to W^p-. In the place of v1 push_up(P, Tint, lint, nodes_in_order) and v2 push_up(Q, Tint, lint, nodes_in_order), it is equally possible to have W2 build_W2(Tint, lint, nodes_in_order); v1 W2 P; v2 W2 Q.

Since these computations already occur with respect to the L2-norm no further steps are necessary. This process effectively preprocesses two probability vectors from two samples and takes the L2-norm of the difference between them to exactly compute the L2-UniFrac score.

3.6 Computing Averages with Respect to L2-UniFrac

Now that we are capable of computing L2-UniFrac with respect to preprocessed values, a new ability emerges that allows us to evaluate entire microbial communities rapidly. This process is accomplished by pushing up every sample taken from that specific community (i.e. a specific body site or elsewhere), and taking the mean of these vectors. Since the mean of probability vectors is itself also a probability vector, taking the mean with respect to these values will yield the average, representative sample for that particular community, just in pushed-up format that can be converted back into a probability vector. In other words, this averaged pushed-up vector is the center of mass for this community. Since the average was computed with respect to the pushed-up vector rather than the probability vector, this process accounts for the weighted differences in the phylogenetic tree that could yield different, inaccurate results otherwise.

The main reason behind this is because pushing up a probability vector will bring that vector into "UniFrac

space." That is, the vector now exists in a state where its values are formed using the edge weights and the weights of each node's descendants. It is no longer strictly a probability vector. When we then compute L2-norms and averages on these vectors, these processes are instead being computed with respect to the UniFrac metric instead of just probabilities by themselves. If we did not push the vectors up, we would instead be performing L2-norms and averages with respect to a Euclidean metric which is not present within the formulation present in eq. (3.4).

However, our method of pushing it up to obtain the average then lets us inverse the push-up process to obtain the representative probability vector for the entire community. This representative sample can be used as normal using previous UniFrac methods or can be used in the L2-UniFrac Weighted Plain function present in section 3.7 for on-the-fly computation with respect to L2-UniFrac in the same manner that would have been accomplished in prior work such as EMDUniFrac. Additionally, leaving the representative sample pushed up enables the ability to rapidly compute a distance matrix using a much more simple computation than L2-UniFrac Weighted Plain by just using an L2-norm in order to generate a PCoA plot or something similar, illustrating the relative biological spread between different communities. This feature will be explored in more depth later during our experimentation.

Below is a representation of the process behind computing averages of vectors with respect to L2-UniFrac in pseudocode. Given a set of samples $p_1; p_2; \quad ; p_n$, we will compute the average, representative sample.

Algorithm 6 Computing averages with respect to L2-UniFrac

```
Inputs:
        n  !  Number of probability vectors to average
        m  !  Length of probability vector
        P  !  List of sample probability vectors
        Tint  !  Ancestor dictionary
        lint  !  Edge length dictionary
        nodes_in_order  !  List of nodes bottom to top
p_pushed    [[], [],...,[]] of size n
for i=1 to n do
    p_pushed[i]    Push_Up(P[i], Tint, lint, nodes_in_order)        . Push up every p_1; p_2; ; p_n
end for
pushed_count    [0, 0,...,0] of size m
for i=1 to n do                                                     . Get sum of each column
    for j=1 to m do
        pushed_count[j] += p_pushed[i][j]
    end for
end for
average_pushed    [0, 0,...,0] of size m
for i=1 to m do                                                     . Compute component-wise (column-wise) mean
    average_pushed[i] = pushed_count[i] / n
end for
average    Inverse_Push_Up(average_pushed, Tint, lint, nodes_in_order)        . Probability vector
return average
```

3.7 L2-UniFrac Weighted Plain

As mentioned earlier, L2-UniFrac Weighted Plain is a more immediately available variant of L2-UniFrac that we have developed with the intent of enabling smaller computations between individual samples or even averages that have already had their push-up inverted and are, therefore, probability vectors without the added complexity of pushing up vectors before computing the norm. As a result, while algorithm 5 computes the L2-UniFrac metric with respect to preprocessed vectors that are not explicitly probability vectors, algorithm 7 computes the L2-UniFrac metric with respect to probability vectors. The ideal use case for such a function is simply to compute the distance between only a handful of species rather than entire distance matrices since preprocessing with push-up is typically more optimized in such situations.

Algorithm 7 L2-UniFrac Weighted Plain

```
Inputs:
        P  ! Sample 1 probability vector
        Q  ! Sample 2 probability vector
        Tint  ! Ancestor dictionary
        lint  ! Edge length dictionary
        nodes_in_order  ! List of nodes bottom to top
function P, Q, L2Unifrac_weighted_plain(Tint, lint, nodes_in_order)
    num_nodes    jnodes_in_orderj
    Z    0
    eps    10  8                                          . Significance threshold
    partial_sums    P   Q                                 . Difference between probability vectors
    for i = 1 to num_nodes  1 do
        val    partial_sums[i]                            . Grab current index probability difference
        if abs(val) > eps then
            partial_sums[ancestors[i]] += val             . Push probability to ancestor
            Z += edge_lengths[i, ancestors[i]] (val^2)     . Sum probability^2 times edges traversed
        end if
    end for
    Z    p Z                                              . Square root of sum of squares
    return Z
end function
```

As shown above, L2-UniFrac Weighted Plain takes in a dictionary of all nodes and their ancestors for lookup, a dictionary of edge lengths between two nodes, a list of nodes in order from the deepest to shallowest in the tree, and both sample probability vectors P and Q. Starting with the mass Z (the weighted UniFrac distance) equal to zero and the difference between the two probability vectors, we then iterate up the tree, pushing all significant values up to their ancestor and incrementing the mass by the distance travelled times the square of the mass at the child node. At the end, we are left with Z^2, so we must take the square root to get the exact value for L2-UniFrac.

This process follows loosely from the principles of eq. (3.3) in that the probability differentials are being multiplied

by the distances traversed as we move up the tree. eq. (3.3) effectively takes the distance between the vectors minus the spread within each vector itself to resolve the distance where we take the difference between the two vectors immediately to produce the partial sums and then push the mass up to find the total distance between said partial sums. However, a more robust analog to L2-UniFrac Weighted Plain is simply the algorithm for EMDUniFrac, where, instead of focusing on the abundances of P and Q independently, we instead focus on finding the differential abundances between them, pushing that vector up by accumulating mass to parental nodes, accumulating the sum of squares of the masses, scaled by edge lengths on each node, and applying a square root to the output to complete the L2-norm. The primary distinction between this algorithm and EMDUniFrac lies on two main lines, shown below.

`Z += edge_lengths[i, ancestors[i]]` $(\texttt{val}^2)$

`Z` $\sqrt{Z}$

From the function defined for the L2-norm in section 2.2.2, a proper L2-norm should follow $\|V\|_2 = \sqrt{\sum_{v_i \in V} v_i^2}$, and in the case of this formulation, rather than taking the absolute value of `val` in the first line, we simply adjust to square and then take the square root of the entire summed quantity at the end, treating the edge lengths as an independent adjustment factor in both cases. As a result, it is evident that in the manner EMDUniFrac conforms to the L1-norm, L2-UniFrac conforms to the L2-norm in this function.

We can also see clearly that this algorithm is effectively computing the quantity $W^p \cdot P \quad W^p \cdot Q$ or $W^p \cdot (P \quad Q)$. Given what we know from section 3.3 and section 3.4, we can see clearly that the partial sums variable captures the differential mass $P \quad Q$ for each subtree. Additionally, this differential mass is scaled by the edge length leading to that subtree, thereby giving us the direct computation from before. The only difference is as follows. Since the L2-UniFrac quantity is given as the L2-norm of the differences between these scaled vectors, the only remaining step here is the L2-norm. This is accomplished through the accumulation of the Z quantity which is the sum of squared weights scaled by the aforementioned edge lengths. The square root is then applied to the result of this sum of squares to complete the L2-norm, thereby correctly giving a result for L2-UniFrac directly.

3.8 Main Function

For ease of use, many of the functions have been implemented in a script titled `main.py`. Each of the functions can be invoked by running `main.py` and using the appropriate tags to run a specific function. Each function allows for intermediate usage/storing which is intended to save and intermediate computations in preset file locations, from which, the program can later pull to compute a different computation with a substantial time savings. Otherwise,

many of the functions would have to run a very computationally expensive distance matrix computation which is used for evaluation purposes. Additional input parameters allow for the inclusion of L1-UniFrac in the results rather than the default of L2-UniFrac only, specify the number of threads to use for parallelized operations such as preprocessing samples and distance matrix computation, whether or not the console should log various moments during execution, otherwise known as verbosity, and the output file for the specified functions. The output file name should be input simply as a base string; since some functions will naturally have multiple associated outputs, the name will be a template on which the script will append information and the associated file extension. The following table illustrates some of the parameters taken by `main.py` and their associated use cases.

`main.py` Arguments		
Argument	Purpose	Additional Args
-v	Print out progress report/timing information	No
-th	Number of threads to use	int
-i	Stores intermediate report generation	No
-p	Uses intermediate report data to accelerate output	No
-bi	Biom file: Sample dataset in the BIOM file format	File Address
-tr	Tree file: OTU Tree in TREE file format	File Address
-md	Metadata file: Sample names will be the first column and body habitats are the third column, following "OBERON:"	File Address
-tx	Taxonomy file: Conversion between Tree Node and Taxonomy	File Address
-o	Generic base output file name. Specific requested operations will append tags and filetypes to the end when applicable	Base Name (str)
-t	Compute total PCoA and generate plot	No
-g	Compute group PCoA averages and generate plot	No
-c	Compute clustering report for distance matrix	No
-k	Compute Krona visualization for each averaged group	No
-d	Compute differential abundances for each group pair	No
-L1	Compute using L1 UniFrac only	No
-U	Compute using both L1 and L2 UniFrac	No

Table 1: List of all available arguments available to use when calling main.py. Most tests have prerequisite information including the biom, metadata, and tree files. Certain tests also require the taxonomy file as well for classification purposes. The tests include the tags -t, -g, -c, -k, and -d, and at least one of those is necessary for any output to be observed. An example command to generate a clustering report for L2-UniFrac is as follows: `python main.py -vip -bi fbiom file pathg -tr ftree file pathg -md fmetadata file pathg -tx ftaxonomy file pathg -o foutput file base nameg -c`.

The full working implementation of this project's code is available at `https://github.com/KoslickiLab/L2-UniFrac`.

4 Experiments

The data for this thesis was primarily acquired from the Qiita study titled "Structure, function and diversity of the healthy human microbiome (V35)" [14]. This particular source was selected due to its relatively large size of 6346 samples as well as the diversity of body sites present within the data. Since a large portion of this report focuses on the clustering and averaging capabilities of L2-UniFrac, having a more diverse set of microbial communities allows us to demonstrate the usefulness of our techniques more clearly. This particular Qiita study was part of a publication of interest that we will be referring to periodically for background information. From this, it is important to note now that the samples we observed during our experimentation were taken from a healthy human microbiome, meaning that the insights gained during our tests are in reference to the clustering metrics associated with such a sampling environment [5].

Additionally, beyond the particular .biom files for a particular study of interest, we also require several background files necessary for UniFrac in general. Such files include a phylogenetic tree (.tree), a file containing all sample name metadata, and a file containing conversions between tree node IDs into taxonomies for level comparisons (.tax). The data for each of these was acquired from a variety of sources.

Firstly, the phylogenetic tree was obtained from the Greengenes database and constructed using FastTree on real-world 16S OTU data with a 99% similarity threshold to match all OTUs in the study [8, 42]. 16S OTU data is important because it is present in every known bacteria, so the dataset is able to encompass all known bacteria, if present. Next, the metadata files were obtained from using the microbe search engine's database [22, 21, 53, 52]. It is important to note that some metadata file terms can be applied to surrounding regions as well. In [5], the convention of "skin," "saliva," "vagina," "oral cavity," and "feces" is generalized to "skin," "nasal," "urogenital," "oral," and "gastrointestinal" respectively. In descriptions later in this report, this paper will predominately use the former terminology since it is directly derived from the raw data. However, on occasions where differences between categories is necessary, the latter terminology may be mentioned in brief as a distinguishing factor. Finally, the taxonomic data was also obtained from the Greengenes database with the same 99% similarity 16S OTU numbering corresponding to the samples, consisting of data obtained by McDonald D. et al. and Werner JJ. et al [31, 58]. This data was additionally classified using the Mothur classifier [46, 57]

Aside from the data, we performed several experiments dedicated to demonstrating the differences between L1 and L2-UniFrac, median and mean computations, and computation of the most representative organism using this

metric for various body sites. These tests are aimed primarily to demonstrate the similarities in performance between the L1 and L2 variants of the UniFrac metric while giving added advantages that will become particularly apparent in section 4.2.

4.1 L1 vs L2

Evaluating the differences between L1 and L2-UniFrac can lead to key insight into where one algorithm may perform better or worse than the other. This paper will look in detail at several different evaluation metrics for each, including the presence of negatives within a selection of real-world data and an analysis of each algorithm's effectiveness at clustering points. In this discussion, we will define each of the main tests we will be performing, the purpose of performing that test, and the associated data requirements. All of the experiments outlined within this subsection relate to the utility of L1 compared with L2-UniFrac under normal UniFrac use cases. That is, for pairwise analyses between individual samples. The primary focus of the other experiments not within this section focus more on nuances regarding the averages of such samples. Our goal within this section is to show that L2-UniFrac performs similarly to L1-UniFrac in the type of experiments designed for L1-UniFrac.

4.1.1 Principal Coordinate Analysis of Samples

The first objective of our direct comparison on a normal use case is to qualitatively demonstrate the principal coordinate analysis clustering of samples with distances computed using L2-UniFrac. This analysis utilizes a method of multidimensional scaling that is dependent upon the differences between each point or, in other words, the distance between every point and every other point in a pairwise manner. The goal of this portion of the experimentation will be to visualize both the similarities and differences between each of the samples, colored by their respective microbial community. These plots will be relativistic in nature and only exist to visualize the relative clustering the relative separation between different samples.

Plotting both L1 and L2-UniFrac using the PCoA plots will allow us to visualize the clustering abilities of each algorithm. As stated earlier, our objective is to show that L1 and L2-UniFrac exhibit similar performance with regards to clustering while improving the key metrics such as preventing the existence of negative probabilities in averaged, representative samples. As such, we aim to verify that we can visually see distinct clusters of microbial samples. These clusters, should they exist as anticipated, given the results from Gonzalez et al. [14], will exist as distinct colored regions of points. We expect this to hold for L1-UniFrac as it uses the base metric similar to that

used by [14] (except this time represented using a PCoA plot rather than a PCA plot), but we need to verify that L2-UniFrac also yields similar qualitative clustering performance. That is, we should be able to distinctly group together regions of the graph that exist almost exclusively for specific microbial communities.

To generate the PCoA plots for evaluation, we will use the following methodology. Firstly, we must compute the distance matrix which is the primary parameter used for the PCoA algorithm. This process can be accomplished in a variety of ways thanks to the flexibility of our implementation. The first method is by using the algorithm L2-UniFrac Weighted Plain, noted in section 3.7, which takes two probability vectors and the associated parameters required for the phylogenetic tree and iterations and outputs the L2-UniFrac distance between those two vectors. For our sample, this would be repeated a total of $6{,}346^2$ times to compute the entire pairwise distance matrix. An alternative approach would be to utilize the preprocessing algorithm called push-up, noted in section 3.3, which will first linearly iterate over each of the 6,346 samples and then store their preprocessed values. Then, the script would simply use the process outlined in section 3.5 to use those pushed-up values and compute L2-UniFrac using the L2-norm of each vector's difference. This will be computed $6{,}346^2$ times to form the distance matrix. Of these two possibilities, the second one would be faster because the first process runs a loop of length nodes_in_order for each of the $6{,}346^2$ iterations whereas the preprocessing approach runs 6,346 for loops of length nodes_in_order and then $6{,}346^2$ more optimized computations across vectors. However, we will be using L2-UniFrac Weighted Plain for L2-UniFrac for this section and the already published EMDUniFrac as L1-UniFrac [29].

Once the distance matrix is computed, the next step will be to use it, along with the biom file, to construct a scikit-bio DistanceMatrix object [47]. Then, the metadata can be extracted and imported into a pandas DataFrame object for usage in the plot itself, specifically as a tool to add metadata-based color coding of points [37, 59]. Then, using the DistanceMatrix, we can utilize the PCoA tools provided by scikit-bio's stats.ordination extension to transform our distance matrix into a PCoA plot object [47]. The final step in this process then is to simply graph the PCoA plot using Matplotlib's Pyplot package [19]. We can then repeat this process using L1-UniFrac.

Now that the plotting is complete, we will then have the a visual representation of the clustering of our samples. As a result, we will then be able to qualitatively examine the differences between the L1-UniFrac and L2-UniFrac approaches, each of which should yield slightly different clustering visually.

Our hypothesis for this experimentation is that L2-UniFrac will exhibit similar visual grouping performance to L1-UniFrac. L2-UniFrac should, therefore, tend to cluster samples with similar body sites closer together and those with different body sites further apart. The extent to how far they are separated should be representative of how

similar the microbiomes themselves are. For instance, saliva and oral cavity would be expected to be very close together in a PCoA since they are physically close together on the body, whereas saliva and feces would be expected to be much more distinct and thus further apart. As a result, we expect L2-UniFrac to show such clustering occurs under the new algorithm, and we will use L1-UniFrac as a performance baseline for this new dataset.

4.1.2 Clustering Analysis

Of the quantitative metrics to analyze the effectiveness of L2-UniFrac in comparison to L1-UniFrac, the first major tool of interest is performing a clustering analysis. That is, we wish to determine how accurate each metric is given different clustering and evaluation algorithms to see how well individual sample groups (saliva, skin, etc.) are isolated from other groups. Knowing the comparative effectiveness of clustering techniques demonstrates potential trade-offs between L1 and L2 UniFrac in terms of performance. Better clustering scores, given by various scoring metrics, indicate that one algorithm yields better grouping of similar features which can allow users to more easily identify groupings of like samples in the absence of known metadata.

To begin this process of performing the clustering analysis, we will first process all of the data to generate a pairwise distance matrix using both the L1 and L2-UniFrac algorithms. From here, we will send the distance matrix into both the Agglomerative Clustering and K-Medoids Clustering algorithms to produce the resultant clustering in various specifications. Each "cluster" produced will essentially be a vector with a length equal to the number of OTUs and each element being an assigned group number (0-4) rather than associating any particular group to a body site directly. These results are now ready to be scored. To do so, both clusters will be passed into each of the scoring algorithms from scikit-bio [47] and their results will be displayed. Upon completion of this process, we will then be able to directly compare the relative performance of L1 and L2-UniFrac.

From what we know about the two algorithms, we would expect both to perform relatively similarly on most metrics. The reasoning behind this is the idea that both of these algorithms, in the absence of taking averages, function relatively similarly in that they just compute different variants of the UniFrac metric under similar procedures with respect to different norms. If we see any major differences across the board in the clustering performance, that may indicate that one of these methods actually turns out to be worse comparatively speaking.

It is also necessary to evaluate any performance gap, if it exists, using a principal coordinate analysis. Since PCoA tends to be the predominate choice among researchers as compared with clustering metrics, it may be able to give more detailed insights than a black-box ratio from a scoring algorithm may be able to provide. Such an analysis

will proceed in the following manner.

Begin by taking the clustering vectors from earlier (Agglomerative and K-Medoids) and package it into a pandas DataFrame object. Then, extract all necessary data from the distance matrix and package it into a scikit-bio DistanceMatrix object. Doing this provides all necessary information needed to graph a PCoA plot using the DistanceMatrix object and color code it using each of the clustering vectors with the DataFrame object.

Once all of the four PCoA plots are generated (two per UniFrac variant, split between Agglomerative and K-Medoids clustering), perform a visual analysis to check for the significance of the clustering results. Clustering that conforms closely visually to the ground truth with groups generally lying in the proper locations on the plot will strongly support the prior conclusions from the scoring algorithms, but clustering with less commonalities with the ground truth PCoA can support lower scoring. Additionally, it is possible that we may see visual clustering that appears to contradict the relative differences in scores such as poor visual clustering with a high score or very close-to-baseline visual clustering with a low score. Both of these cases can indicate that the results may be dataset dependent.

Our hypothesis for this experiment is almost the same as section 4.1.1. We expect the quantitative clustering scores achieved by both L1 and L2-UniFrac to be similar on an absolute scale (though they may differ in a proportional sense). Additionally, we expect to see grouping occur in similar regions as with the baseline PCoA plot for both UniFrac variants to show that, in the absence of metadata, clustering still occurs within the same regions using both metrics to show L2-UniFrac is just as useful as L1-UniFrac in this regard.

4.2 Presence of Negatives

As discussed in the introduction, one hypothetical advantage of L2-UniFrac over L1-UniFrac lies in the strictly non-negative nature of all indices of probability vectors for averaged samples. When we compute the average of the vectors in UniFrac space, as shown earlier, there is a potential that that average may contain elements outside of the convex hull formed by each of the individual coordinates. Since the averaging occurs within UniFrac space, the result at this point will be positive as expected since none of the input values are negative, but as soon as we inverse push-up to bring the median out of UniFrac space into Euclidean space, each quantity will now be converted into some probability vector under a process similar to that outlined in section 3.4 with a slight modification for L1-UniFrac. To be specific, L1-UniFrac will add the multiplicative inverse of the edge length to the children and subtract it from its parent for each node in order. This causes the mass to be pulled down from ancestor nodes using

values obtained through a median and not a mean, which, as mentioned earlier, are not guaranteed to lie within the convex hull formed by all of the probability vectors composing that microbial community. As such, if that particular index lies outside, the inverse push-up vector will then contain a negative number at that index. We will formally present this algorithm shortly in algorithm 8.

By contrast, in our experimentation, we would not expect such a thing to appear for L2-UniFrac, given our prior observations. That is, when we take the mean of pushed-up vectors in UniFrac space, we would expect each component of those vectors to remain within the convex hull formed by every other vector of that microbial community. As a result, when the push-up procedure is inverted, we expect no significant negatives to appear in the resultant representative probability vector for that community, barring those of very small magnitude that appear due to computational errors from limited bit representation.

To test this hypothesis, we plan to use the following experimental design. Firstly, push every sample up into UniFrac space using L2-UniFrac's push-up method. This is the basis for our averaging protocol, so it must be done on every sample of interest. Then for every sample within the biom file outlined at the start of section 4, we will classify that sample using the associated metadata for that sample. This process will result in a dictionary containing all of the body sites where samples were taken as the keys and all of the associated sample probability vectors as a list in the values for that sample. For instance, say there are three samples associated with the saliva body site. In this case, the index 'tongue' in the dictionary will contain a list containing 3 probability vectors. This is done to separate out each of the samples by the region of the body it was taken. Now that the samples are separated by type, we compute the component-wise mean of each pushed-up vector taken from a particular body site. This will yield the representative vector for that body site in UniFrac space, so to return the probability vector, we simply invert the push-up procedure on it to obtain the representative probability vector for the presence of various OTUs within that community's average sample. A pseudocode implementation of the above description is present in section 3.6.

Finally, our experiment will conclude by counting the total number of OTUs containing a negative abundance greater in magnitude than 10^{-12}. This value was selected because epsilon is set at $2.220446049250313 \times 10^{-16}$, so to account for loss of decimal precision and subsequent error propagation throughout this process, we have given some padding in the threshold to account for such errors in computation. We will confirm that the selection of this error threshold is not too generous by comparing the magnitude of any potentially captured negatives greater in magnitude than the threshold are not within only one or two orders of magnitude of the threshold. That is, if our largest magnitude negative is -10^{-11}, but we selected -10^{-12} as our filtering threshold, this would indicate that the

selection of the filtering threshold biased the results. If however, we capture a negative of 10^{-7}, for instance, this is 5 orders of magnitude larger in size, indicating that the threshold had no impact on the results, thereby confirming our findings.

We then repeat this process using L1-UniFrac and its associated methods. Since L1-UniFrac relies on the L1-norm, we use the median of the pushed-up vectors instead of the mean which is used for L2-norms. As such we must make a few additional modifications to the aforementioned algorithm. Most notably, instead of the push-up and inverse-push-up functions from section 3.3 and section 3.4 respectively, we will use the L1 variant instead, the difference being that in push-up, the line $P\_Pushed[i] = \sqrt{lint[i;Tint[i]]}$ is replaced with $P\_Pushed[i] = lint[i;Tint[i]]$ for push-up. For inverse-push-up, there are a few more adjustments, but they are that the line $P\_Pushed[i] += \sqrt{\frac{p\_val}{edge\_length}}$ is replaced with $P\_Pushed[i] += \frac{p\_val}{edge\_length}$ and the line $P\_Pushed[Tint[i]] = \sqrt{\frac{p\_val}{edge\_length}}$ is replaced with $P\_Pushed[Tint[i]] = \frac{p\_val}{edge\_length}$. We will call these modified functions L1_Push_Up and L1_Inverse_Push_Up respectively. Now, using these modified algorithms, we can apply them to a new algorithm which will compute the median with respect to L1-UniFrac. The primary difference here outside of the switch to L1-UniFrac will be that we need to take column-wise medians instead of means. For the sake of brevity, we will use a built in macro 'median' which takes a list of probability vectors and returns the column-wise median of those vectors. That is, it returns the median abundance for each OTU in the respective OTU position. If the list length is even, it will return the arithmetic mean of the two median candidates. The algorithm is as follows for the median with respect to L1-UniFrac.

Algorithm 8 Computing averages with respect to L1-UniFrac

```
Inputs:
        n  !  Number of probability vectors to average
        m  !  Length of probability vector
        P  !  List of sample probability vectors
        Tint  !  Ancestor dictionary
        lint  !  Edge length dictionary
        nodes_in_order  !  List of nodes bottom to top
p_pushed     [[], [],...,[]] of size n
for i=1 to n do
    p_pushed[i]     L1_Push_Up(P[i], Tint, lint, nodes_in_order)          . Push up every p_1; p_2;     ; p_n
end for
pushed_col     [[], [],...,[]] of size m
for i=1 to n do                                                          . Get column values in m lists
    for j=1 to m do
        pushed_col[j].append(p_pushed[i][j])
    end for
end for
average_pushed     [0, 0,...,0] of size m
for i=1 to m do                                  . Compute component-wise (column-wise) median
    average_pushed[i] = median(pushed_col[i])
end for
average     L1_Inverse_Push_Up(average_pushed, Tint, lint, nodes_in_order)          . Probability vector
return average
```

If our hypothesis is supported, we will see that L2-UniFrac successfully eliminates all of the negative abundances as was indicated by the mathematics behind its development. By contrast, we expect to see some potential negative abundances in our runs of L1-UniFrac, although this may be relatively infrequent since it is more common for the component-wise median to lie within the convex hull than outside. However, in our example, we are utilizing 6,346 total samples, each of which uses probability vectors spanning hundreds of thousands of total OTUs. Such a large number of OTUs increases the probability of such an issue occurring significantly, so showing that this works even on very large samples containing a large number of OTUs is important to show L2-UniFrac's efficacy at effectively handling very large datasets as opposed to its predecessors.

4.3 Principal Coordinate Analysis of Means

Similar to section 4.1.1, this section will focus on the principal coordinate analysis of the means of the samples. The motivation behind such an analysis is simply to see the separation of the representative samples for each of the clusters present in the first PCoA plots. This process exists primarily to demonstrate the utility of representative samples in more efficiently comparing entire microbial communities with each other.

To accomplish this process, we will use a similar process to that of the previous section. However, rather than

using the entire distance matrix, we will instead first compute all of the averages for the microbial communities. This can be accomplished using a linear scan to separate each of the samples by their body site metadata. Then, we pass those groups of samples through the algorithm outlined in section 3.6 which returns the component-wise average for those samples. If we were to analyze L1-UniFrac in this manner, instead we use the median_of_vectors present within our L1U module. Then, we use our new average vectors to create a much smaller distance matrix than before. However, since L1-UniFrac is not designed to average vectors within UniFrac space due to numerous reasons that will be outlined in other sections such as section 5.3, section 5.5 section 5.6 and it is unlikely to obtain interesting findings for it in this test, we will not be using it for this experiment.

In the dataset provided by [14] there exists 5 different body sites among all of the 6*;* 346 samples. As such, we simply need to use these 5 average vectors to create a 5x5 distance matrix. As before, we can now create a scikit-bio DistanceMatrix object, using numbers rather than sample names this time, representing each of the groups [47]. Then, we create our pandas DataFrame like before using the metadata for color-coding of the points [37, 59]. The DistanceMatrix can then be used by scikit-bio's stats.ordination extension to transform it into a PCoA plot object [47]. Finally, we again utilize Matplotlib's Pyplot package to visually graph the final PCoA plot [19].

As you can see, much of this process is similar to that of section 4.1.1 with the exception of generating the pairwise distance matrix of the averages of the microbial communities rather than of all of the samples. This, as one would expect, is significantly less time consuming to compute than the entire pairwise distance matrix for every single sample, meaning that a simple computation of the pairwise distance matrix generated by the averages of the microbial communities would be more readily available without significant waiting involved.

Our hypothesis for this experiment is that L2-UniFrac will be an effective tool to take meaningful averages. Specifically, similar microbiomes will show close proximity to other similar microbiomes and large distances from less-similar and more distant microbiomes. Since the results of the ordination will likely select fundamentally different dimensions to maximize dissimilarities between all other points, the graphs may exhibit different shapes, but general trends will be evident.

4.4 Taxonomic Distribution

As expected, the primary goal of this research is to be able to obtain a representative microbiome for each body site as a direct point of comparison between body sites. Having the mean or median body site microbiome will allow researchers to quickly compute the evolutionary distances between body sites as well as to differentiate amongst them

in different applications. New samples similar to an existing average sample will likely also belong to that sample.

The procedure for computing this representative microbiome will be as follows. First, push up every sample within that particular body site. Then, take the column-wise mean of the n sample vectors V, each of length h. That is, $\forall i;\ 0 \quad i < n$, compute $\frac{\sum_{j=0}^{n-1} V[i][j]}{n}$. A pseudocode implementation of this algorithm is presented in section 3.6 and follows the above justification. For L1-UniFrac, the corresponding algorithm is mentioned in section 4.2.

The method of forming "representative" microbiomes for L1-UniFrac is outlined earlier in section 4.2. This approach instead uses the median as is typical of the L1-norm in place of the mean from L2-UniFrac averages.

Once both of these algorithms are computed, we will now have representative samples for entire body sites, and we can use them to launch an investigation into their structure. The most notable test that we will be performing on these representative samples directly will be to first convert the nodes present in the final vector into their respective taxonomic classifications. Then, using the values, we will compute the summation of mass present at the lowest level of classification available for each node. That is, if a node's classification is listed in the form of "k_xyz;p_abc;c_mno", we can see that the lowest level of classification present at that node is the class (c), so the mass at that node would be added to the class total. The motivation behind this is that, since samples lie entirely within leaf nodes in the phylogenetic tree (i.e. nodes that do not serve a purpose of connecting other nodes), a robust averaging algorithm should produce results primarily in actual classifications (kingdom, phylum, class, order, family, genus, species) rather than in connecting nodes (shared history). This will allow the representative sample to be as useful as possible in determining characteristics about the body site such as the average presence of a particular OTU across that body site or determining similarities between newly taken samples and body sites without necessitating referencing distances on the phylogenetic tree explicitly. For our experimentation, we will compute the mass present at each taxonomic level and report for both L1-UniFrac and L2-UniFrac to determine advantages and disadvantages of each.

The raw data that is computed will show the abundances for each body site for both L1 and L2-UniFrac. We will primarily focus on the fecal microbiome in this section since many of the same conclusions can be drawn from the other microbiomes. However, we will include the graphs for and briefly note any other findings from this test for the other body sites as well. The tests in section 4.5 investigate this further for every microbiome.

Our hypothesis for this experiment is that L1-UniFrac may result in more higher-order taxonomy than L2-UniFrac as a consequence of the inverse push-up procedure failing to capture all mass correctly. We expect L2-UniFrac mass to be concentrated almost entirely on lower-order taxonomic classifications such as species, genus, family, and order.

4.5 Krona Evaluation

As outlined in section 2.2.7, KronaTools provides a powerful environment to visually and quantitatively analyze abundances across taxonomic distributions or other classification system. In the case of this work, taxonomic analysis is the primary motivation of this portion of the experimentation. Such evaluation can potentially lead to new insights not covered in the prior analysis of the lowest-level taxonomic classification computed in section 4.4. Since Krona introduces finer details on every taxonomic level at once while including abundances, it provides greater insight into specific categorizations and sub-categorizations throughout the averaged sample's taxonomic classifications. For reproducibility purposes, these steps assume prior installation of the KronaTools package from section 2.2.7.

To accomplish this task, the first step is to retrieve the L1 and L2 averages for each body site, initially obtained in the experimentation outlined in section 4.4. Then, using these averages, go through each OTU and use the taxonomy reference to compute the entire taxonomic hierarchy for that OTU. This hierarchy includes everything from kingdom through species, if applicable. Some OTUs will stop early in their hierarchy since OTUs can encompass entire groups of species that share many commonalities. The hierarchy will include the precise names of each category going down the taxonomic level for that particular OTU. For instance, the kingdom of an OTU may be listed as Bacteria and the phylum may be Bacteroidetes. If the OTU is not present within the taxonomic lookup list, it is a temporary internal node and must be classified simply as "internal" as it does not exist as a modern OTU and is instead an ancestor of modern species. The internal designation will simply be considered as a kingdom for graphing purposes.

The next step in this process is to format all of this information into an acceptable Krona input format. For this experiment, the selected format of choice will be the simple text file format which is built as a tab-separated file to form columns. The first of these columns is set to be the abundances for each particular OTU. The next level will then be the kingdom, followed by the phylum, then class, and then continue on until species. Since this file format is relatively flexible with allowable inputs, it is hypothetically possible to classify any particular subdivision quantity and not explicitly biological quantities. However, once this process is complete for all OTUs present, Krona can directly take the file as an input using the ktImportText command followed by a -o specifier for output files.

Once all of the body sites have their respective Krona charts computed for each UniFrac varient, an analysis will be conducted to assess any potential underlying differences between L1 and L2-UniFrac with regards to taxonomic allocation following the push-up, average in UniFrac space, and inverse-push-up procedure is completed for each of the body sites. An ideal Krona visualization for an averaged sample will ideally behave similarly to a Krona

visualization of a single OTU. Such a visualization will consist of exclusively real-world OTUs that are classifiable (no internal nodes present) as well as fine granularity including both highly abundant elements as well as elements that are very sparsely present within the population. In the event that few differences are noticeable between the two variants of UniFrac, it may also be necessary to perform a closer analysis at the particular abundances given by each of the averaged vectors and explain any potential discrepancies that may occur in how each metric's averaging quantifies the differences.

Our hypothesis for this experiment is that we will see similar results to that of section 4.4 but also notice greater biodiversity in the number of OTUs present in L2-UniFrac when compared to L1-UniFrac. Additionally, we expect to see OTUs that may have been entirely ignored by L1-UniFrac as a consequence of medians only accepting OTUs present in $> 50\%$ of the samples where L2-UniFrac takes a direct average.

4.6 Most Representative Organism

The most representative organism within a sample is defined as the organism or OTU with the greatest differential abundance as compared with some other sample. The definition of differential abundance vectors is given in [29] at the end of section 2. Since this research aims to provide a framework from which researchers are able to compute averages of entire sets of samples accurately and efficiently, it is also possible to use these averaged abundance vectors to compute the most differentially abundant OTU within a particular group in relation to every other group. Such a computation would allow generalizations of regions, body sites, or other specifically targeted areas of importance into a single OTU being a key characteristic of that region that explains patterns in that region's alpha and beta diversity. For instance, one target application of this testing would be to compare a particular body site of a healthy patient to that of a patient who is suffering from some condition in that region to identify potential factors or even a set of microbes responsible or as a consequence of the condition.

To begin this process, the first objective is to be able to track differential abundance at each OTU. In this work, we propose a small modification to the L2_Unifrac_weighted_plain algorithm, denoted as L2_Unifrac_weighted, that captures the magnitude of the signed difference in the abundances present between two samples on every subtree, defined by each edge, weighted by that edge. This implementation is handled identically to EMDUniFrac (L1-UniFrac) since this operation is not with respect to the L1 or L2-norm as it is not a summation of absolute values or of squares, so we apply the prior algorithm to our L2-UniFrac base to obtain the following pseudocode.

Algorithm 9 L2-UniFrac Weighted

```
Inputs:
        P  ! Sample 1 probability vector
        Q  ! Sample 2 probability vector
        Tint  ! Ancestor dictionary
        lint  ! Edge length dictionary
        nodes_in_order  ! List of nodes bottom to top
function L2Unifrac_weighted(P, Q, Tint, lint, nodes_in_order)
    num_nodes    jnodes_in_orderj
    Z    0
    diffab    dict()
    eps    10  8                                                    . Significance threshold
    partial_sums    P    Q                                         . Difference between probability vectors
    for i = 1 to num_nodes  1 do
        val    partial_sums[i]                                     . Grab current index probability difference
        if abs(val) > eps then
            partial_sums[ancestors[i]] += val                      . Push probability to ancestor
            if val 6= 0 then:                                      . Capture differential abundance on non-zero edges
                diffab[(i, ancestors[i])]    edge_lengths[i, ancestors[i]]*val
            end if
            Z += edge_lengths[i, ancestors[i]] (val^2)             . Sum probability^2 times edges traversed
        end if
    end for
    Z    p Z                                                       . Square root of sum of squares
    return Z, diffab
end function
```

As is clear in the above algorithm, the only changes to L2_Unifrac_weighted_plain the is necessary is the addition of a differential abundance dictionary, a conditional to verify if the value is nonzero, and a mapping of each edge to the value at the root of its subtree scaled by the length of that edge. The end result of this process yields the L2-UniFrac score between those two vectors P and Q, denoted Z and the differentially abundant weighted subtrees between those two vectors.

A small observation about the differential abundance metric will lead to the realization that differential abundances compare entire subtrees, meaning that the largest differential abundances in magnitude are most likely to be internal nodes as their partial sums will be increased or decreased (signed quantity) by every one of their descendants. This is especially true when the overwhelming majority of the descendants belong to only one sample. This distinction means that an internal node that has descendants more present in one sample as compared with the other sample is more likely to have a greater cumulative mass than any individual descendent leaf node. Thus, running direct differential abundance testing would be expected to yield primarily internal nodes.

To address this shortcoming and allow the data to be more readable, we propose the following process to assign some taxonomic rank to every internal node other than the root node for the purposes of differential abundance

testing.

1. Find every leaf node in nodes_in_order by checking if it is not internal.
2. Create a descendant dictionary where keys represent all nodes and values represent the list of all descendants to that node.
3. For every leaf node, recursively trace up the tree until reaching the root node. For each node reached, append that leaf node to the descendent dictionary.
4. Iterate through every node in order again and perform the following steps:
   (a) List out the taxonomy of every leaf node that descends from that OTU.
   (b) If the length of this list is zero, it is itself a leaf node and can be converted directly.
   (c) Otherwise, create several counting dictionaries for kingdom, phylum, class, order, family, genus, and species to track the number of times each occurs in the taxonomy list.
   (d) Iterate through the taxonomy list and accumulate the counts at each taxonomic level.
   (e) Look to see if any taxonomic level was present in all leaf nodes by comparing counts. If the count equals the length of the taxonomy list in (a), then append that taxonomy level to the output string.
   (f) If no shared taxonomy exists between any level of the taxonomic hierarchy, then it must be a root node.
   (g) Finally, append the final string yielded to the output taxonomic list.

We will simply refer to this classification scheme as "most shared" or "universally shared" taxonomy. That is, at every node, we take only the taxonomic classification shared by all descendants of that node if the node itself is not classifiable.

We also propose, for L2-UniFrac exclusively due to the findings of section 5.5 and section 5.6, another taxonomic classification scheme titled "leaf-only" taxonomy. As the name describes, this method only plots the differential abundances on leaf nodes within the tree, meaning that we only see the differential abundance of real-world OTUs. As a result, where universally shared taxonomy gives a high-level evolutionary perspective of the divergence, leaf-only taxonomy gives a real-world visualization of which specific taxa are more present in one body site relative to another.

Now that we have a comprehensive taxonomy, the next step is to use the earlier-mentioned L2_Unifrac_weighted algorithm to find the differential abundances between each pair of averages. To do this, the first step is, of course,

finding the average vectors. This process has been covered earlier in this paper in section 3.6 and section 4.4, so we will just use the process for L2-UniFrac again. The next step in this process is to simply compute the differential abundance between each pair of averages using the L2_Unifrac_weighted function from earlier. For the universally shared taxonomy, the differential abundances will be cast into a dictionary with taxonomies as keys to sum across all taxonomies with the same name and then remapped back to the first respective OTU position with that taxonomy in order to remove duplicate taxonomic entries when only higher orders are shared among descendants.

Using the differential abundances, nodes in order, taxonomy in order, and several other minor configurations, we can now directly push our data into a function initially built for EMDUniFrac [29] for differential abundance plotting, with only minor modifications specific to this paper. Aside from modifications to limit the total number of visible samples in the output and changing graph dimensions, the primary modification of interest is that we instead pipe the taxonomy directly into the labels rather than the node names. Since the node names are given as numbers primarily from the source data, taxonomy provides much greater readability and interpretability, which is the main reason this decision was made.

At the end of this process, we will finally have a series of differential abundance graphs for L2-UniFrac. We can then repeat this process with L1-UniFrac for continuity, but the primary goal of this particular experiment is to focus specifically on L2-UniFrac for reasons that will be made clear in section 5.5 and section 5.6 and will be touched upon again in section 5.7. As a result, L1-UniFrac will be only discussed in brief in this section where we will introduce an example outlining such difference on a truncated case where we make one further modification to algorithm 9 in that we only output diffab on non-internal nodes. These graphs will be then analyzed to draw conclusions about particularly differentially abundant taxonomies present within each sample group.

Our hypothesis for this experiment is that the final OTUs selected as most differentially abundant within our differential abundance graph will all be corroborated by prior literature, showing that these species L2-UniFrac's average protocol was able to isolate are indeed present in above-average quantities within that specific microbiome they were identified in.

5 Results

Having run the outlined experimentation in section 4 using the datasets presented at the beginning of that section, we now can interpret the results in the context of our stated hypotheses. This includes the PCoA of samples and groups, the clustering analysis, the presence of negatives, the taxonomic tree distribution of averages, the Krona evaluation, and the differential abundance testing.

5.1 Principal Coordinate Analysis of Samples

In this section, we will analyze the results of our experimentation related to performing a PCoA analysis on the pairwise distances between every sample, initially outlined in section 4.1.1. As noted before, we would expect L2-UniFrac to perform similarly compared with L1-UniFrac in its PCoA performance visually. This is an innately qualitative metric for evaluation but is frequently used within the field of bioinformatics for visualizing differences between samples corresponding to specific body sites.

Figure 1: A plot showcasing the principal coordinate analysis (PCoA) plot of all of the 6346 samples in the dataset. The distance matrix which was used for the PCoA was obtained by computing the pairwise distances between every sample using the EMDUniFrac (L1-UniFrac) algorithm with individual sample pairs. The PCoA was then computed automatically using the `skbio.stats.ordination.pcoa` function from the `scikit-bio` python package. The coloration in the plot corresponds to samples originating from different body sites, and their visual proximity to other samples indicates their relations in terms of their L1-UniFrac distance.

The above diagram, fig. 1, visualizes the principal coordinates analysis of the pairwise distance data generated by each of the samples using the L1-UniFrac method from prior research [29]. As can be seen, some clustering is visually present within this sample, especially notable for the skin, oral cavity, vagina, and feces segments which compose different regions of the plot, generally speaking. We also can note that the saliva itself appears to be spread throughout the region dominated by the oral cavity, which makes sense considering that the saliva itself would be expected to share many of the same microbes that are present within the general cavity it is located in. However, we also note that these clusters are relatively loose clusters–that is, we notice dense regions of points with fuzzing on the outer edges of each cluster where points gradually become less and less frequent. This observation will come into play as we analyze L2-UniFrac.

Figure 2: A plot showcasing the principal coordinate analysis (PCoA) plot of all of the 6346 samples in the dataset. The distance matrix which was used for the PCoA was obtained by computing the pairwise distances between every sample using the L2-UniFrac algorithm with individual sample pairs. The PCoA was then computed automatically using the `skbio.stats.ordination.pcoa` function from the `scikit-bio` python package. The coloration in the plot corresponds to samples originating from different body sites, and their visual proximity to other samples indicates their relations in terms of their L1-UniFrac distance.

In fig. 2 produced using L2-UniFrac instead, we can see a very visually different form of clustering occurring. The same four groups as L1-UniFrac exhibit similar divisions in the graph as before, but we can notice that the "fuzzing" effect of L1-UniFrac is significantly less noticeable in L2. This is likely due to the larger relative penalties to outliers due to the L2-norm within L2-UniFrac which graphically clusters points closer together on a PCoA plot.

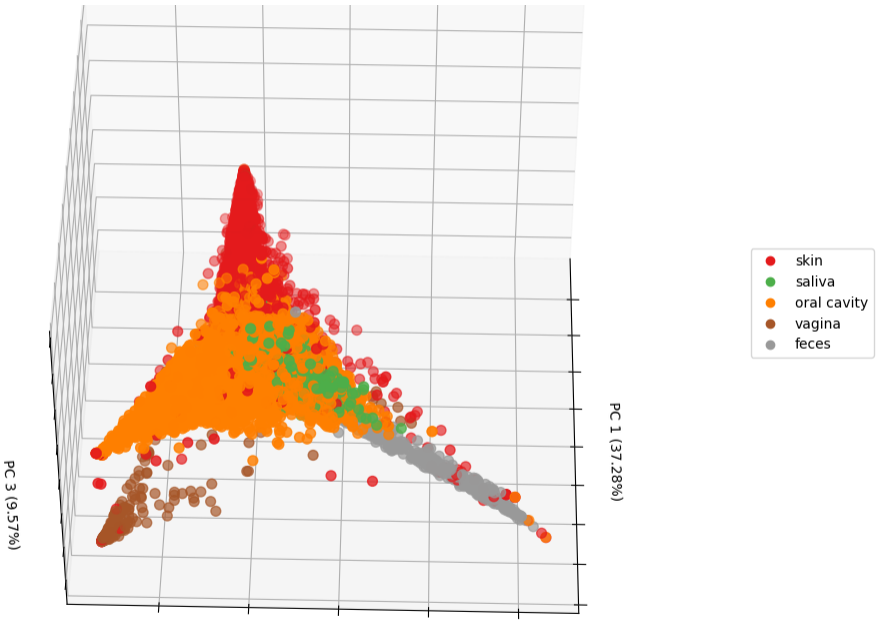


Figure 3: The same plot produced and displayed in fig. 2 with a different perspective. The plot is constructed in the same manner described in the referenced plot but shifted to focus more on the PC 2 axis from a downward-facing perspective. This graph focuses on the grouping present within the saliva body site from the prior figure.

Visually speaking, we are able to retrieve a little more information about our samples using the L2 plot as opposed to L1 in a particularly noticeable manner. Under the default perspective, this is not as evident, but if we shift the perspective slightly, we are able to reveal more information about the relations of saliva relative to that of particular subsets of the oral cavity. Specifically, in fig. 3, we can see this different perspective. Rather than appearing to be interspersed throughout the oral cavity group as it appeared to be in our analysis of the L1-UniFrac PCoA graph, L2-UniFrac reveals that it actually clusters very tightly within a particular group that happens to be enveloped almost entirely by the oral cavity entirely. This would imply that saliva is more closely related to a particular subset of the oral cavity than other elements of the same group and it does not entirely encompass the microbial communities of the oral cavity itself. Such observations are not as immediately evident in PCoA plots produced using L1-UniFrac due to the "fuzzing" obscuring such similarities slightly more whereas in L2-UniFrac though they do exist. As a result, we have demonstrated our initial hypothesis that we would see similar performance in the visual principal coordinate analysis between L1 and L2-UniFrac to be correct. On a slightly less formal basis, it is arguable that L2-UniFrac exceeds the performance of L1-UniFrac in this instance due to the tighter visual clustering, but this can be simply a product of the dataset.

As a result, we have shown our hypothesis to be supported by these PCoA plots. L2-UniFrac does indeed exhibit

similar clustering performance to L1-UniFrac and is useful for separating samples based only on abundances of OTUs. Additionally, we have shown that L2-UniFrac places samples that are distinct from other samples far apart whereas more similar samples (such as those originating from the oral cavity and saliva) are placed close together.

5.2 Clustering Analysis

When we continue to the experiment outlined in section 4.1.2, we can begin by generating the clustering report file using the main program. These numbers are qualitative measures of the clustering across each UniFrac form, using both Agglomerative Clustering (paragraph 2.2.5.1) and K-Medoids (paragraph 2.2.5.2). The results of such scoring metrics show how closely the clustering algorithms approximated the ground truth without knowing any of the metadata. As a result, these values will show the extent to which clustering is present within a dataset. Higher clustering would, therefore, indicate that a distancing metric better separates distinct groups than a metric yielding lower clustering. Under our initial hypothesis from the introduction, L2-UniFrac should perform similarly to L1-UniFrac with regards to clustering, and all values should ideally be within reasonable margin of error.

Running the experiment, we obtain table 2, shown below.

Clustering Method	Scoring Metric	L1-UniFrac	L2-UniFrac
Agglomerative	Rand Index Score	0.558338	0.584673
	Adjusted Rand Index Score	0.138531	0.121903
	Normalized Mutual Index Score	0.298005	0.223136
	Adjusted Mutual Info Score	0.297123	0.222249
	Fowlkes Mallows Score	0.518455	0.455341
K-Medoids	Rand Index Score	0.773531	0.722562
	Adjusted Rand Index Score	0.471479	0.370212
	Normalized Mutual Index Score	0.550697	0.522974
	Adjusted Mutual Info Score	0.550256	0.522492
	Fowlkes Mallows Score	0.636532	0.575773

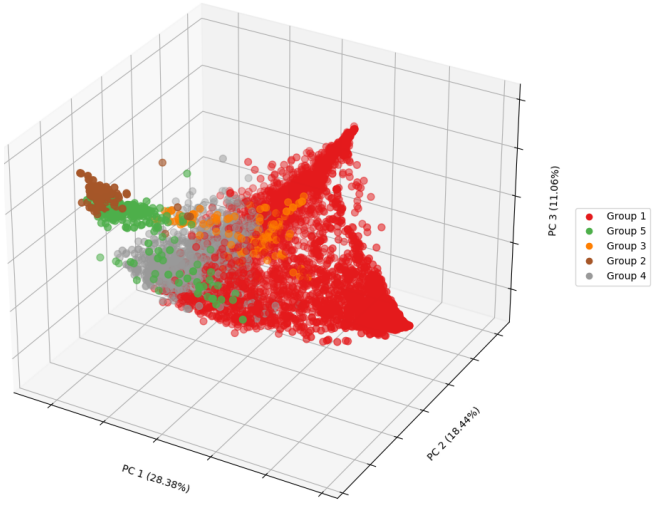
Table 2: Metadata-blind clustering scores computed for both L1 and L2-UniFrac. Clustering is computed using either agglomerative or k-medoids clustering algorithms discussed in paragraph 2.2.5.1 and paragraph 2.2.5.2. The scoring algorithms each take both the ground-truth assignments and the clustering assignments for each sample and then compute their respective scores using the processes outlined in each subsection of section 2.2.6. Scores closer to 0 indicate worse-possible clustering performance and scores closer to 1 indicate perfect clustering performance.

In table 2, we can see that L2-UniFrac outperforms L1-UniFrac marginally on only a single metric of the Rand Index Score for Agglomerative Clustering. Every other score indicates better performance for L1-UniFrac. However, it is important to note that many of these differences are only within a few hundredths of a point, with the largest deviation being the K-Medoids Adjusted Rand Index Score test with a 0.101267 point deviation. This shows that

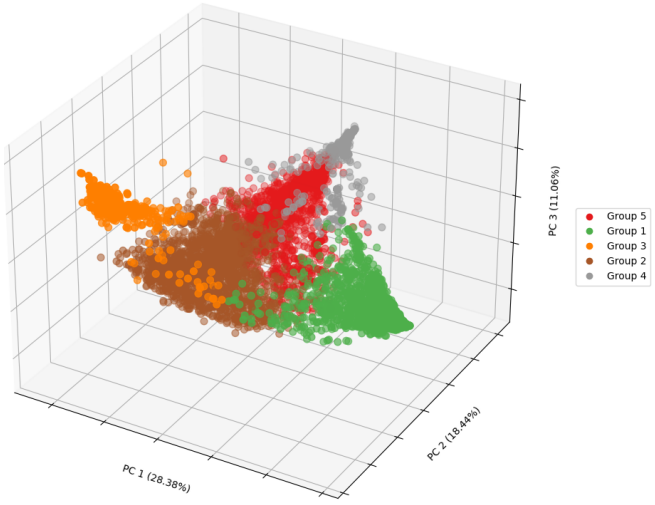
when it comes to qualitative clustering analyses, L1-UniFrac outperforms L2-UniFrac marginally for the wide array of tests we have selected for this experiment, but the performance is relatively similar across the board for each. Both forms of UniFrac perform closely in an absolute sense.

Likely as a direct consequence of the results we observed in section 5.1, the tendency of L2-UniFrac to pull outliers in also appears to have introduced some scoring issues with clustering around the plot as the clusters also visually appear closer together in fig. 2 than in fig. 1. This observation likely leads to blind clustering algorithms performing marginally worse on L2-UniFrac than on L2-UniFrac. However, this clustering assessment would not be complete without visualizations showcasing the accuracy of Agglomerative and K-Medoids clustering relative to the baseline for both L1 and L2-UniFrac. Following the second set of steps outlined in section 4.1.2, we can now perform the visual portion of this analysis in order to assess the relative significance of these results since it is possible that the prior results may be dataset dependent. The following figures illustrate which points were selected by each of the blind clustering algorithms on the bottom two images (indicated as figure (b) and (c) in fig. 4 and fig. 5 respectively) relative to the baseline PCoA on the top (indicated as figure (a) in both fig. 1 and fig. 2 respectively). In both figures, consider the Agglomerative and K-Medoids figures color coding as groups rather than corresponding to any particular region of the baseline figures.

(a) L1 Baseline PCoA



(b) L1 Agglomerative PCoA



(c) L1 K-Medoids PCoA

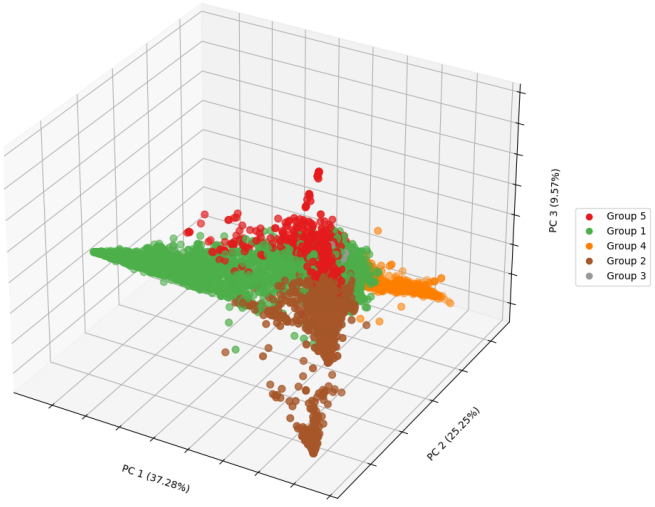
Figure 4: Results of metadata-blind clustering algorithms when using L1-UniFrac for distance matrix computation. Figure (a) is the ground-truth baseline PCoA from fig. 1 for reference. Figure (b) shows the group assignment for the agglomerative clustering algorithm. Figure (c) shows the group assignment for the k-medoids clustering algorithm. Each of (b) and (c) are computed by passing the distance matrix into each respective clustering algorithm which assigns an arbitrary group number and color to each and then passing that group metadata to the PCoA plotting algorithm where metadata would usually be passed.

In fig. 4, we notice several commonalities between the baseline PCoA and the Agglomerative and K-Medoids PCoA graphs. First looking at the largest group in the baseline of skin, we can see that the baseline is somewhere between the more conservative K-Medoids group and the less conservative group in the Agglomerative PCoA, ignoring small outliers. On the opposite end of the PCoA in the baseline figure, the feces segment sits relatively isolated, so it

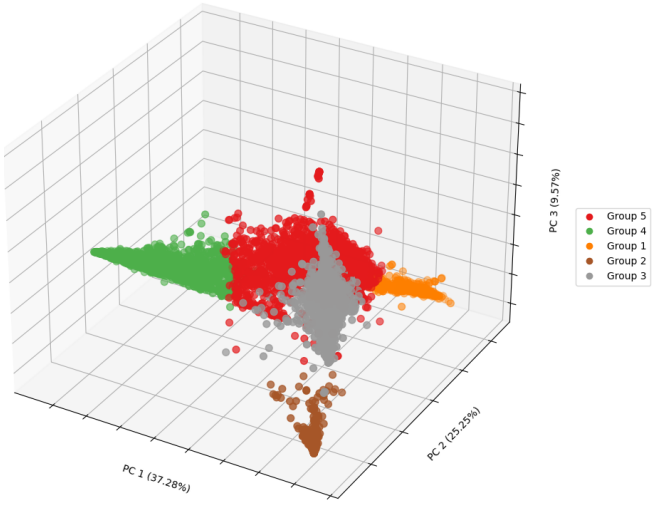
would be reasonable to expect it to be grouped well. As expected, we see some isolated grouping occur in that region in Agglomerative clustering and a much more comprehensive cluster for K-Medoids. Finally, between Agglomerative and K-Medoids, K-Medoids better captures the vaginal region of the baseline graph with a unique cluster appearing in the region.

Among the two, K-Medoids holds closer to the ground truth with this sample set. With that said, the rest of the graph is more mixed, with many of the centrally located elements being clustered along arbitrary lines that differ significantly from the ground truth in both. In Agglomerative clustering, much of the oral cavity, saliva, and feces region is absorbed by the region most closely corresponding to skin whereas in K-Medoids, the oral cavity region appears subdivided when it otherwise should not be. In this experimentation, L1-UniFrac-based blind clustering appears to perform best in the isolated regions of the graph while struggling towards the center.

(a) L2 Baseline PCoA



(b) L2 Agglomerative PCoA



(c) L2 K-Medoids PCoA

Figure 5: Results of metadata-blind clustering algorithms when using L2-UniFrac for distance matrix computation. Figure (a) is the ground-truth baseline PCoA from fig. 1 for reference. Figure (b) shows the group assignment for the agglomerative clustering algorithm. Figure (c) shows the group assignment for the k-medoids clustering algorithm. Each of (b) and (c) are computed by passing the distance matrix into each respective clustering algorithm which assigns a group number to each and then passing that group metadata to the PCoA plotting algorithm where metadata would usually be passed.

Similar to fig. 4, fig. 5 several commonalities exist between the baseline and each of the clustering PCoAs. Firstly, starting with the skin body site, both the Agglomerative and K-Medoids PCoA graphs show grouping in the correct region of the graph with some variation towards the center of the PCoAs. This holds true for the other external body sites of feces and vagina as well. Perhaps the most interesting difference between L1 and L2-UniFrac within this experimentation is the observation that both centrally located groupings of oral cavity and saliva appear in roughly

the same location as the baseline. In other words, under our visual analysis, L2-UniFrac clustering is performing as anticipated with groups clustering in roughly the same regions of the PCoA plot relative to the ground truth, and L1-UniFrac struggles to perform the same visually despite performing marginally better in the quantitative assessment.

In other words, all five body sites appear in roughly the correct location under both clustering algorithms using L2-UniFrac whereas only two to four were noticeably consistent within L1-UniFrac, depending on the clustering scheme. It is possible, however, that this distinction is only a characteristic of this dataset and may not hold across other datasets. Given the prior observations that L2 groupings were more distinct as a result of L2-norms being more sensitive to outliers when computing the distance matrix, this can potentially explain the observations here. In contrast, the L1-norm resulted in more visual mixing between the OTUs within the PCoA plot.

The question now remains as to why L1-UniFrac performs marginally better than L2-UniFrac in blind clustering scoring algorithms despite performing worse in visual cluster identification. If we consider the distribution in each of fig. 4 and fig. 5, the L1 PCoA shows a more even disbursement cloud relative to the L2 PCoA where OTUs are much more concentrated in the center with only a few tails jutting out to the sides. As a result, it is likely that the minor discrepancy in the scoring algorithms noted earlier is due to the higher concentration of central points of the baseline groupings being associated with different groups. This is especially noticeable in the largest body site of skin, where both clustering metrics cut off the nearest group before reaching the center, instead associating central points with oral cavity and saliva OTUs instead. In comparison, the K-Medoids clustering in L1-UniFrac contains most of the points in skin, the largest body site by sample count, excluding the sparser outliers towards the center.

In the end, the conclusions of our clustering analysis are that our hypothesis is supported and the differences between L1 and L2-UniFrac are very minor to negligible in metadata-blind clustering, and neither metric consistently outperforms the other in both scoring and application analyses, with differences in each being marginal or situational at best. Therefore, we can conclude that L2 and L1-UniFrac exhibit similar clustering performance.

5.3 Presence of Negatives

In our hypothesis for the presence of negatives for L2-UniFrac, we stated that L2-UniFrac should produce no negative components within the averaged probability vector, given the mathematics discussed in the introduction related to transformed probability simplexes being closed under the component-wise mean. Following the procedure in section 4.2, we have found the results for each body site median and mean for L1 and L2-UniFrac respectively.

UniFrac Mode	Skin	Saliva	Oral Cavity	Vagina	Feces
L1	0	3	1	1	6
L2	0	0	0	0	0

Table 3: Number of occurrences of negative OTU abundances present in each body site of the averaged vector for both L1 and L2-UniFrac. These counts are obtained by scanning through the output probability vector, which is averaged across the observed abundances for all samples in a given body site, and counting any negative number with a magnitude greater than 10^{-12} (i.e. any number less than 10^{-12}). The final counts are output into this table using `averages.py`.

As can be seen in the above table, L2-UniFrac meets our expectations of it experimentally eliminating this problem. While this issue is exceedingly rare for L1-UniFrac—among 406903 nodes on the phylogenetic tree, the maximum number of negatives observed for a group was 6—it is still possible. This means that L1-UniFrac has fundamental flaws when used for averaging across entire microbial communities as compared to L2-UniFrac which mitigates this issue entirely as was initially proved in the introduction.

To ensure this is not the result of some extraordinary computational error and the threshold did not bias the results, we can verify the counts by analyzing the values that were counted. When we record the negative values for L1-UniFrac at each of the thresholds, we obtain the following results.

Body Site	List of Negatives					
Skin						
Saliva	-0.000104	-1.511331e-06	-0.001841			
Oral Cavity	-2.803081e-05					
Vagina	-0.000148					
Feces	-9.287171e-05	-4.344159e-05	-0.000122	-6.083507e-05	-6.820654e-05	-3.143062e-05

Table 4: The list of negatives counted in table 3. Whenever a significant negative number is encountered, accounting for computational error, that number is recorded to this record as well with each row corresponding to a body site, and the columns on the right corresponding to a simple list of each negative number observed. The numbers are then rounded to six decimal places for reading purposes.

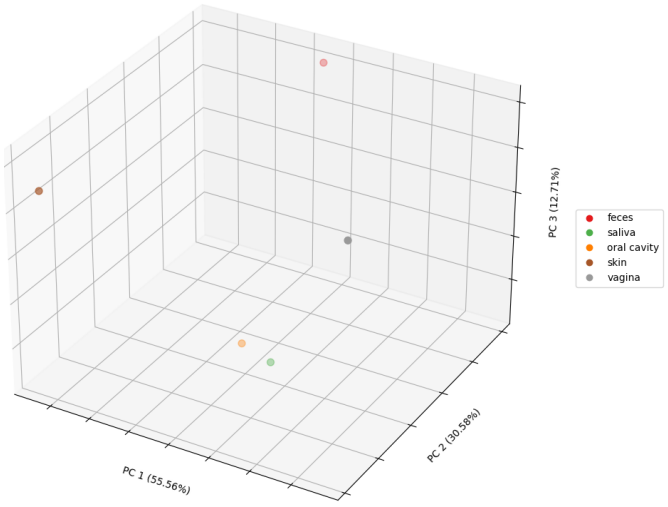
In the above table 4, we can now clearly see which values were counted. For reference, we are using a negatives filtering threshold of 10^{-12} to account for minor computational errors in a computer environment with limited binary representation. With that said, the smallest magnitude negative number observed is -1.511331e-06 which is roughly six orders of magnitude larger than the magnitude of the filtering threshold. The largest negative value is 9 orders of magnitude greater at -0.001841—or roughly an "abundance" of 0.1% of the entire sample, which is itself very large. The compounding impacts of these negative values can make up a large portion of the overall abundance, significantly detracting from the utility of the prior characterization of UniFrac in terms of the L1-norm when applied to averaged

vectors. As a result, this shows a significant advantage L2-UniFrac has that is fundamental to its construction, ensuring consistency of vectors and that no natural constraints are violated.

Our hypothesis for this experiment is, therefore, supported by the results. We have therefore proved, by counterexample, the second part of proposition 1 for such transformations with 4 counterexample body sites. As a direct consequence of proposition 1, L1-UniFrac averages result in negative abundances following their inversion out of UniFrac space. Additionally, as a consequence of proposition 2, we see supporting evidence in that significantly negative abundances do not appear in L2-UniFrac. Therefore, from a biological perspective, L2-UniFrac averages are more meaningful whereas L1-UniFrac averages contradict real-world limitations.

5.4 Principal Coordinate Analysis of Averages

As outlined in section 4.3, it is possible to take and compare averages using both L1 and L2-UniFrac. Once the averages of each of the five microbial communities is obtained, creating a PCoA plot is relatively trivial and primarily consists of recording the metadata of each in order, computing a distance matrix of all averaged samples, passing both the distance matrix and the metadata list to the PCoA function, and generating the graph. The end result of this process gave the following two figures.



(a) Entire PCoA Plot using L2-UniFrac

(b) Group PCoA Plot using L2-UniFrac

Figure 6: Visualizations of the L2-UniFrac principal coordinate analysis (PCoA) conducted for all samples and for the averaged, representative samples. Figure (a) represents the PCoA of the samples using L2-UniFrac across the entire distance matrix. Figure (b) represents the PCoA of the samples using L2-UniFrac's sample averages with a smaller respective distance matrix. Dots that are closer together represent more similar microbial OTUs or communities and dots that are separated more are more distinct. The placement of dots is computed iteratively using a strain function automatically with the `skbio.stats.ordination.pcoa` function in the `scikit-bio` package.

It is important to note that these PCoA plots are visually different from each other due to the ordination selecting different dimensions to maximize the apparent differences seen in the strain function of the PCoA. That is, when a body site is averaged from fig. 6a to fig. 6b, the percentage each dimension is responsible for in the ordination may change differently. Thus, the selected dimensions by the new averaged ordination will likely be different to capture the largest differences.

One can, however, very easily see the overall impact of averaging the samples in comparison to the original graph of all samples very clearly. In fig. 6a, it is evident that the skin, vaginal, and fecal body sites are predominantly clustered on the outer three points of the plot while the oral cavity and saliva body sites are very centrally located at the intersection between the other three. Thus, for an average of this to be useful, we would expect the averages of the skin, feces, and vagina to be relatively separated from each other with the oral cavity and saliva averages being the only two close elements.

Indeed when we look at fig. 6b, we see this exact trend emerge. The averages of the body sites using L2-UniFrac's averaging abilities are shown to form a rough tetrahedron shape along differently selected axes with the skin, vaginal, and fecal body sites completely isolated and the skin and oral cavity averages in very close proximity to one another.

Since we would not expect to see very close ties between the skin and oral cavity to any of the other three microbiomes, it also makes sense to see that they are both relatively isolated from the other groups as well.

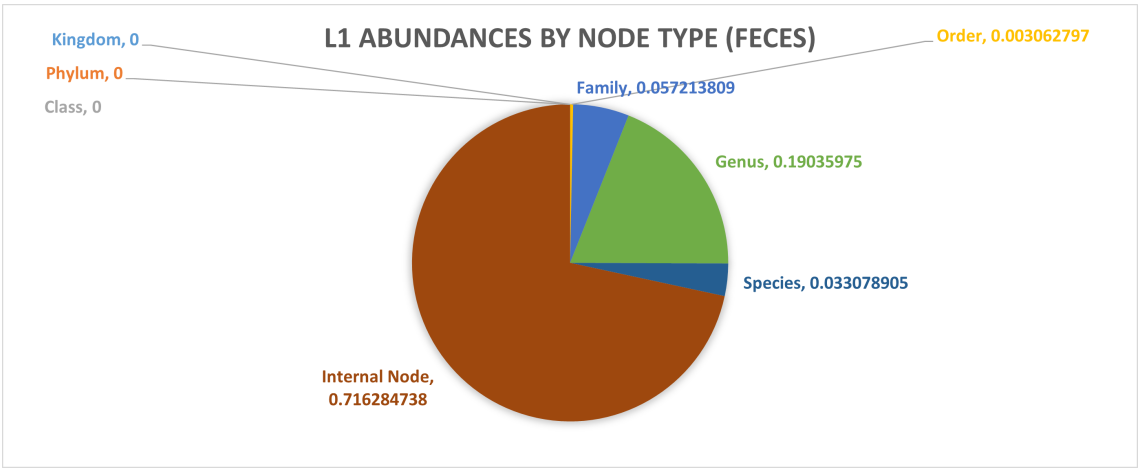
As a result, we can conclude in this section that L2-UniFrac averages successfully differentiate between different microbiomes in a meaningful way relative to the baseline PCoA of all samples and our hypothesis is supported. This means that the general composition of the average of each body site exhibits properties similar to the aggregate of all such samples composing that particular microbiome and serves as a direct center of mass for that body site's clustering. Similar microbiomes are shown to exist closer together when compared with the baseline PCoA, and dissimilar microbiomes are shown to exist far apart under the same comparison.

5.5 Taxonomic Distribution of Averages

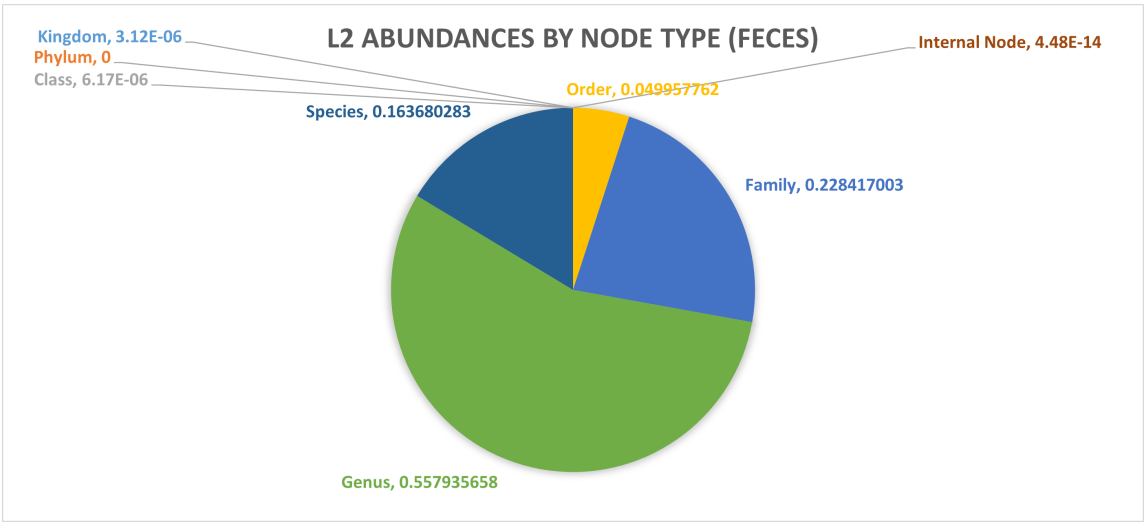
One major result that we have obtained in conducting our averaging across samples that is worth discussing in more detail in this section is the taxonomic distribution of OTUs present in the averages produced by L1-UniFrac and L2-UniFrac respectively. As the phylogenetic trees are formed, many of the nodes serve as temporary (internal) connecting nodes that serve as a common origin between many taxa. These nodes exist on the tree but naturally would not be present in samples. Such classifications are not present in nature. Rather, real-world samples would conform entirely to leaf nodes. This is the motivation for taking the averaged vectors across each sub group of L1 and L2-UniFrac and comparing the distribution of nodes among the taxonomic levels (kingdom, phylum, class, order, family, genus, and species) as well as internal nodes, to determine where the mass primarily lies in the averaged vector. In an ideal situation, all of the mass would be present entirely within the leaf nodes, representing real OTUs that would be present in real-world samples. Using the experimental results from section 5.4 in the production of these averaged vectors, we can use a taxonomy reference dictionary to convert node IDs within our phylogenetic tree to their respective taxonomy and add its mass to the respective lowest level taxonomic classification (species being the lowest). When a node does not exist, we consider this to be an temporary node if it has no classifiable ancestor, so we add it to an internal node mass total.

When we continue this for all averages across the five different body sites of interest in our data, we obtain a cumulative mass percentage for each of the each of the taxonomic levels for each of the body sites. We can see many of the same trends emerge in all of the microbiomes after running our experiments, so for the sake of brevity and to reduce redundancy with the more in-depth analysis of this in section 5.6 for all body sites, we will simply show a in-depth side-by-side comparison of the fecal microbiome under this test with only a brief discussion on the

other four body sites for each. Through this process of evaluating our results into a graphical format, we obtain the following distributions.



(a) L1-UniFrac Taxonomic Distribution of Fecal Body Site



(b) L2-UniFrac Taxonomic Distribution of Fecal Body Site

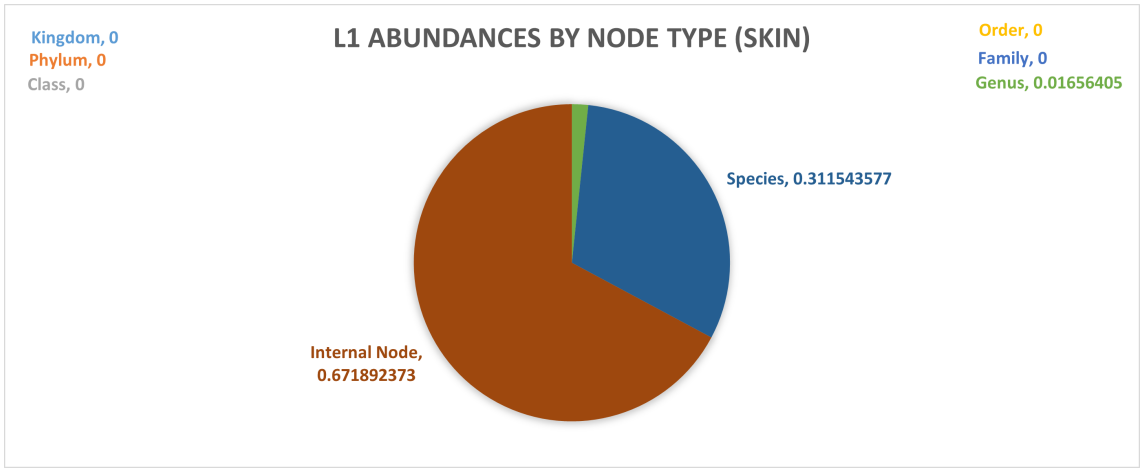
Figure 7: Pie graphs denoting the relative abundances of each lowest-level taxonomic level. Lowest-level taxonomic levels are defined as the level defined on a particular OTU that is the most specific and least general, with species being the most specific and kingdom being the least. Nodes without direct real-world taxonomy such as internal phylogenetic tree nodes that no longer exist are assigned the lowest-level classification of "Internal Node" to denote the node exists only to connect species in the phylogenetic tree and would not be present in real-world samples. To compute these charts, sums are iteratively obtained by finding the percentage of mass located in each OTU and computing the lowest-level taxonomy for that OTU. Quantities are fractional amounts where 0.50, for instance, implies 50% of the mass exists within that associated taxonomic level. Figure (a) and (b) show L1 and L2-UniFrac on the same fecal body site respectively.

In fig. 7a, it is evident that the majority of the mass is concentrated within internal nodes, rather than being present within classifiable taxonomies. This implies that L1 averaging, which uses medians, fails to fully perform

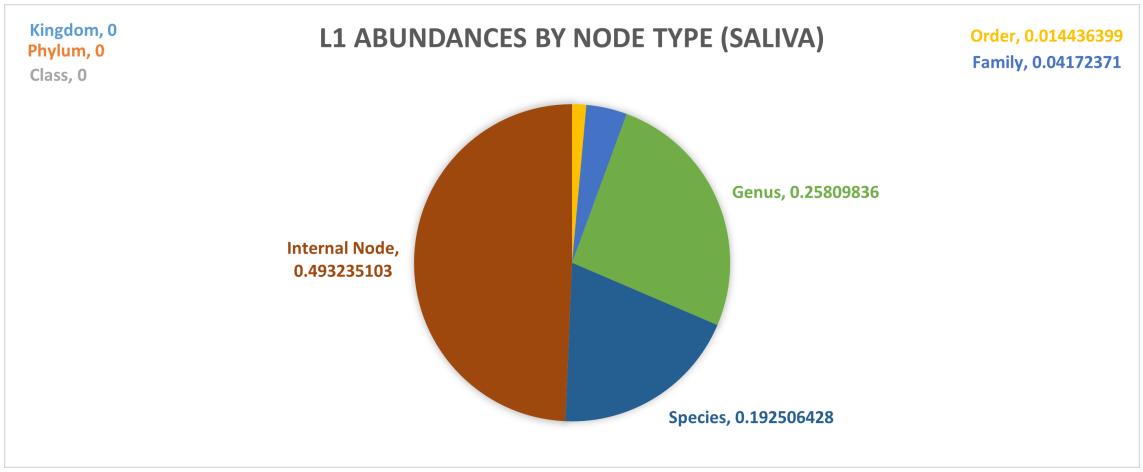
inverse-push-up, indicating that although inverse-push-up is proven to be correct for inverting the push-up process, medians introduce deviations in the mass distribution that cause some mass to be left behind in the internal nodes. The median is not well-behaved within UniFrac space as was illustrated in section 5.3, so it is not surprising to see issues caused in this context as well.

Contrast these findings to the taxonomic distribution of L2-UniFrac averages. Here, we see that the majority of mass lies within the genus and species levels of the tree. We analyzed the body sites individually as well and saw that some body sites such as the skin exhibit a majority abundance within the species level itself, so this majority appears to be interchangeable, depending on shared taxonomic identification present within the OTUs. Additionally, and most importantly, unlike L1-UniFrac, very little mass is left behind in the internal nodes, and the mass that is left is insignificant in comparison to that of any other taxonomic level with the closest being kingdom on the order of 5.5 million times more abundant, despite being the least abundant real taxonomic group.

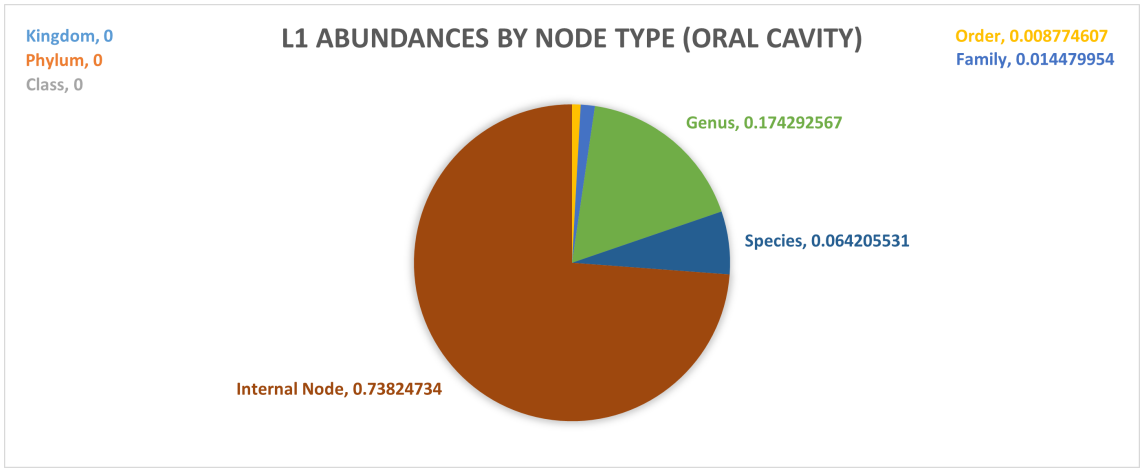
For the sake of completion, the following two figures will illustrate the same general differences between L1 and L2-UniFrac with only minor nuances that will be discussed briefly.



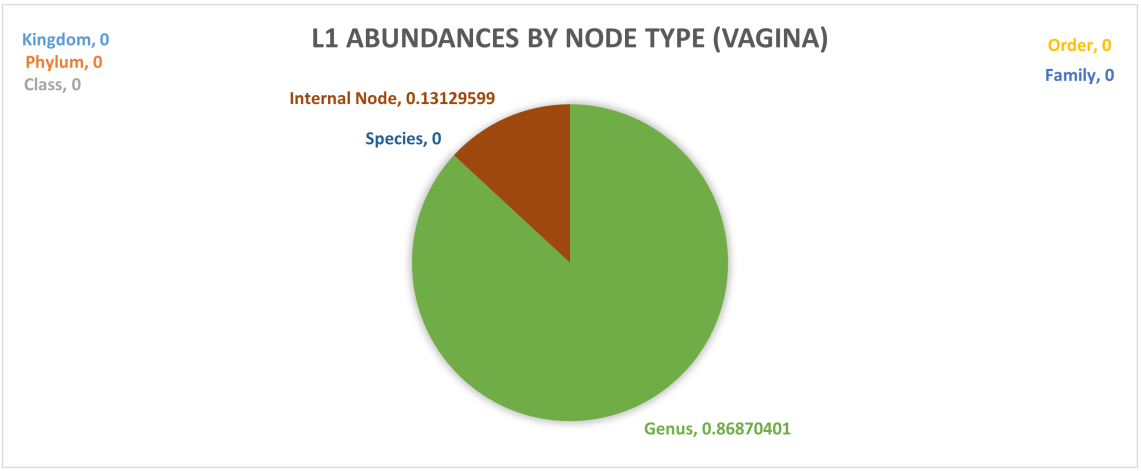
(a) L1-UniFrac Taxonomic Distribution of Skin Body Site



(b) L1-UniFrac Taxonomic Distribution of Saliva Body Site

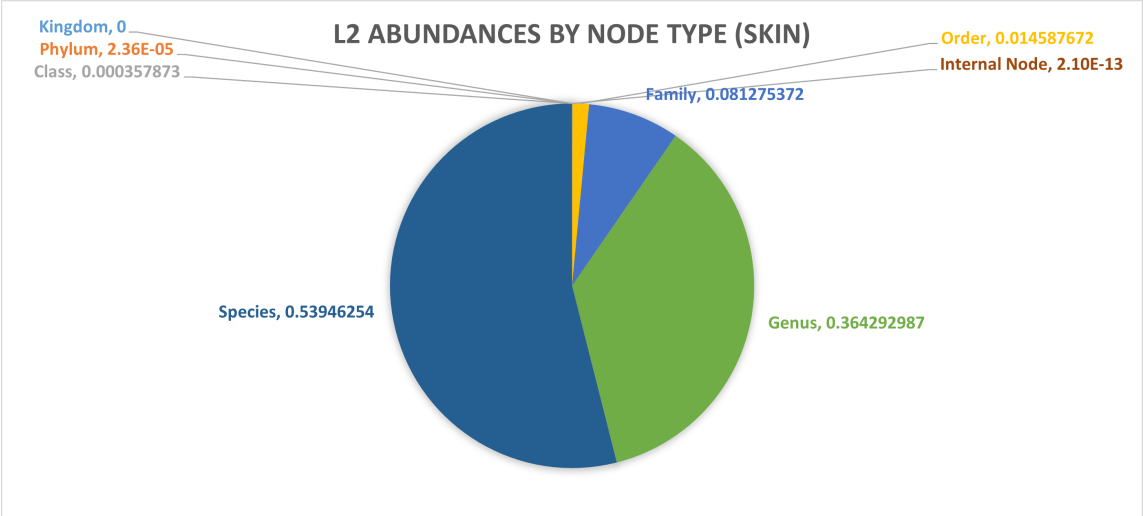


(c) L1-UniFrac Taxonomic Distribution of Oral Cavity Body Site

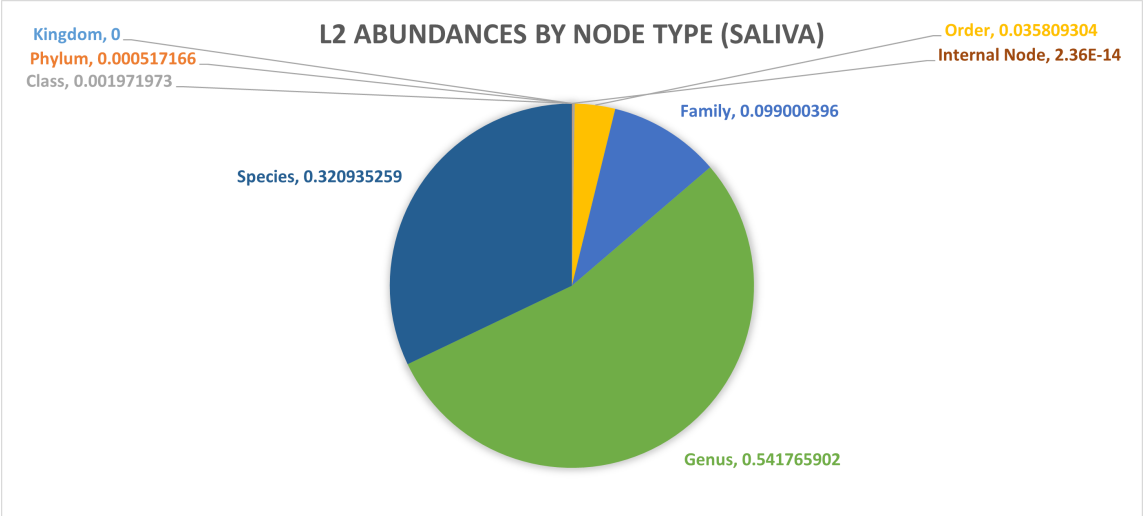


(d) L1-UniFrac Taxonomic Distribution of Vaginal Body Site

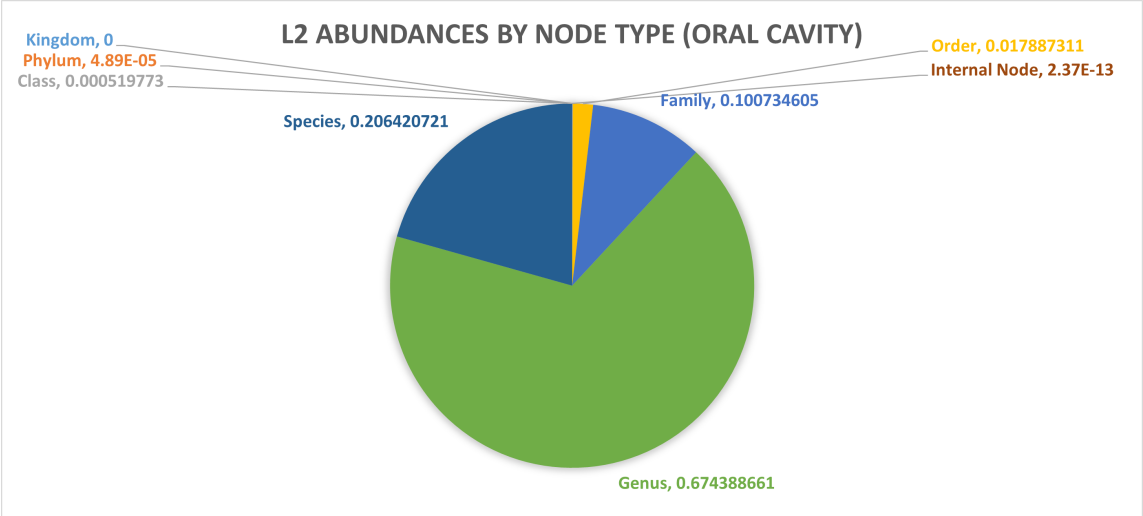
Figure 8: Pie graphs denoting the relative abundances of each lowest-level taxonomic level. Lowest-level taxonomic levels are defined as the level defined on a particular OTU that is the most specific and least general, with species being the most specific and kingdom being the least. Nodes without direct real-world taxonomy such as internal phylogenetic tree nodes that no longer exist are assigned the lowest-level classification of "Internal Node" to denote the node exists only to connect species in the phylogenetic tree and would not be present in real-world samples. To compute these charts, sums are iteratively obtained by finding the percentage of mass located in each OTU and computing the lowest-level taxonomy for that OTU. Quantities are fractional amounts where 0.50, for instance, implies 50% of the mass exists within that associated taxonomic level. Figure (a), (b), (c), and (d) show the skin, saliva, oral cavity, and vagina body sites with L1-UniFrac averages respectively.



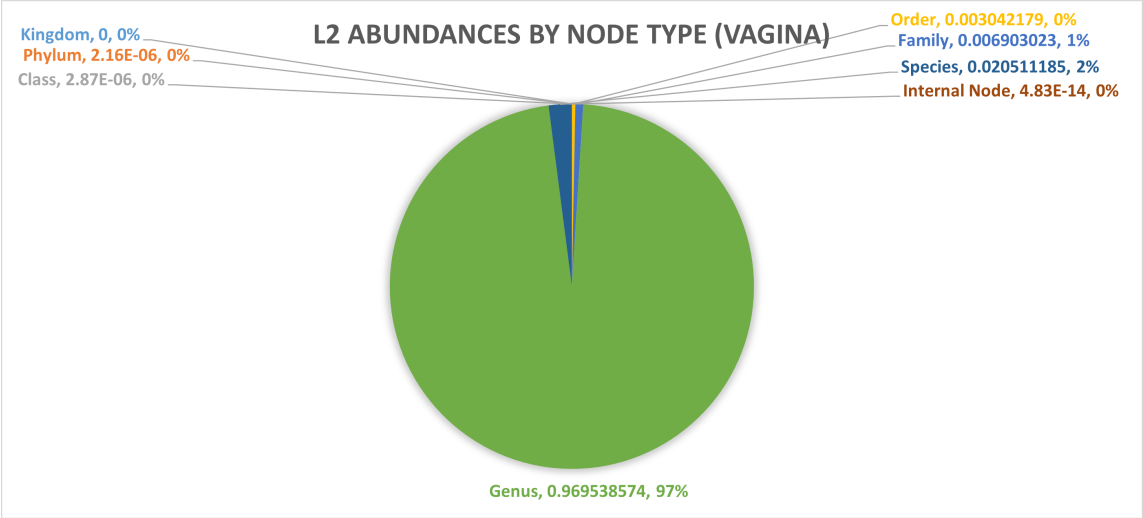
(a) L2-UniFrac Taxonomic Distribution of Skin Body Site



(b) L2-UniFrac Taxonomic Distribution of Saliva Body Site



(c) L2-UniFrac Taxonomic Distribution of Oral Cavity Body Site



(d) L2-UniFrac Taxonomic Distribution of Vaginal Body Site

Figure 9: Pie graphs denoting the relative abundances of each lowest-level taxonomic level. Lowest-level taxonomic levels are defined as the level defined on a particular OTU that is the most specific and least general, with species being the most specific and kingdom being the least. Nodes without direct real-world taxonomy such as internal phylogenetic tree nodes that no longer exist are assigned the lowest-level classification of "Internal Node" to denote the node exists only to connect species in the phylogenetic tree and would not be present in real-world samples. To compute these charts, sums are iteratively obtained by finding the percentage of mass located in each OTU and computing the lowest-level taxonomy for that OTU. Quantities are fractional amounts where 0.50, for instance, implies 50% of the mass exists within that associated taxonomic level. Figure (a), (b), (c), and (d) show the skin, saliva, oral cavity, and vagina body sites with L2-UniFrac averages respectively.

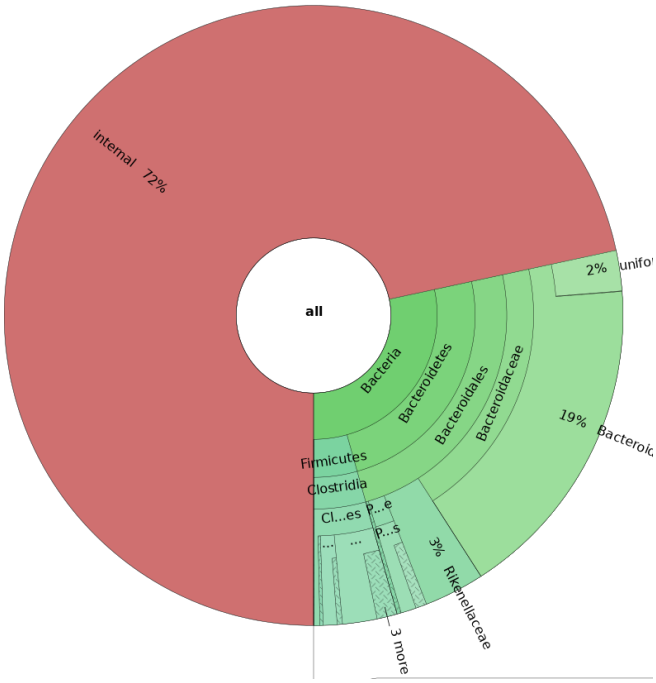
We largely see the same results emerge as before, especially in the skin, saliva, and oral cavity body sites. However, we see only a minor difference in the vaginal microbiome for L1-UniFrac. This difference is rather interesting since internal nodes only make up 13% of this body site compared with half to three-quarters on each of the others. However, to explore this further, section 5.6 will show the OTU-level granularity to provide a deeper dive into the taxonomy.

From these results, we have determined that L2-UniFrac vastly outperforms L1-UniFrac with regards to providing averages corresponding to abundances of real-world OTUs. This capability enables L2-UniFrac to utilize statistical methods to determine characteristics such as the most representative organism, outlined specifically for L2-UniFrac in section 4.6. In order to apply these findings, we will use the experiment outlined within section 4.6 to compute the statistically most representative organism (OTU) present within the L2 average. Our hypothesis is supported to an extent by our findings with the caveat that where we anticipated L1-UniFrac averages to lie more on higher-order taxonomic classifications, they tend to instead lie on internal nodes, though these do often coincide if you brute-force

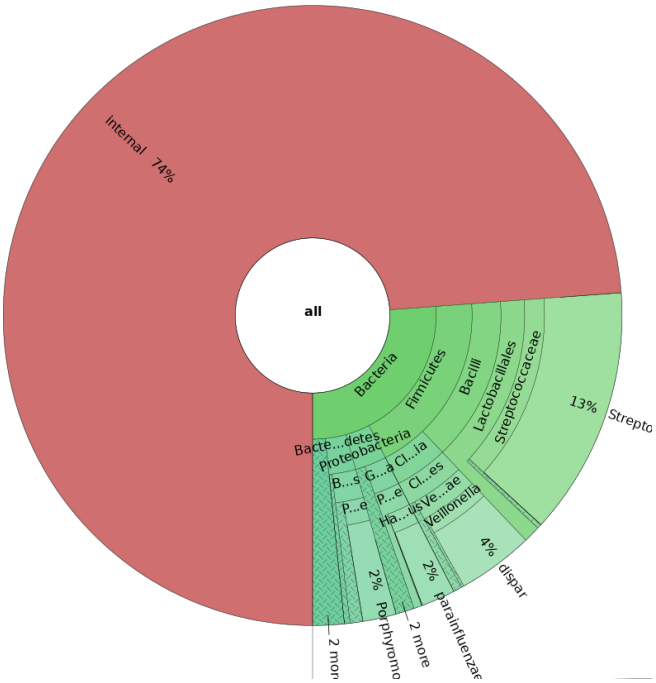
classifications using a technique similar to that used in section 5.7.

5.6 Krona Visualizations

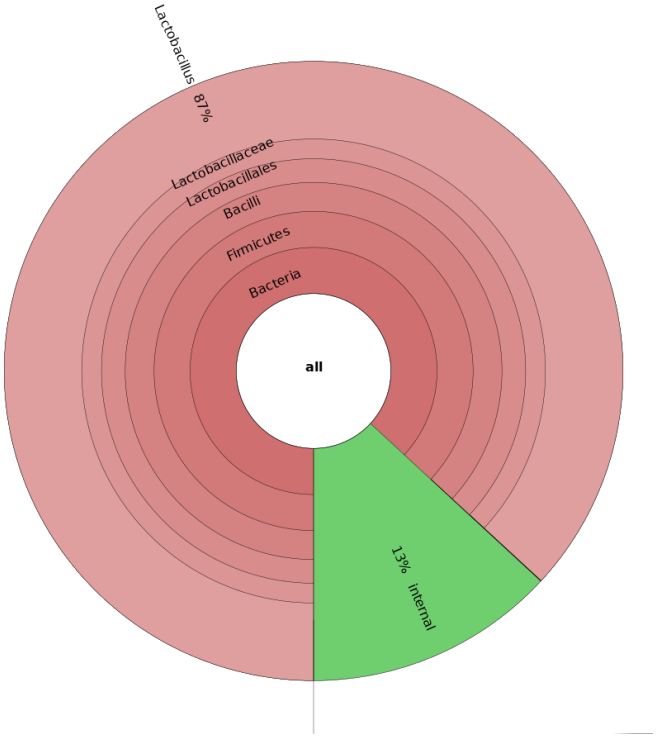
Utilizing the aforementioned KronaTools package, we can also visualize the impact the observations from section 5.5 have on the utility of Krona graphs of averaged samples and compare across L1 and L2 for each body site. It is important to reiterate that normal samples will be effectively entirely composed of classifiable nodes within the phylogenetic trees. It is only when we average a group of samples that internal nodes begin appearing in the abundances, especially with L1-UniFrac as noted earlier in section 5.5. Using Krona, we should be able to have some deeper insights into the specific structure of each of the taxonomic levels as well as the relative distributions of each OTU classification.



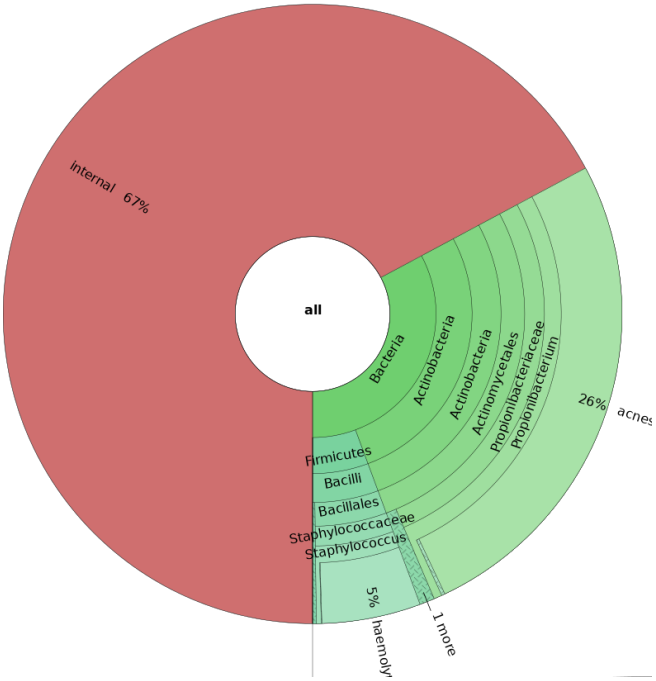
(a) L1 Feces Krona Graph



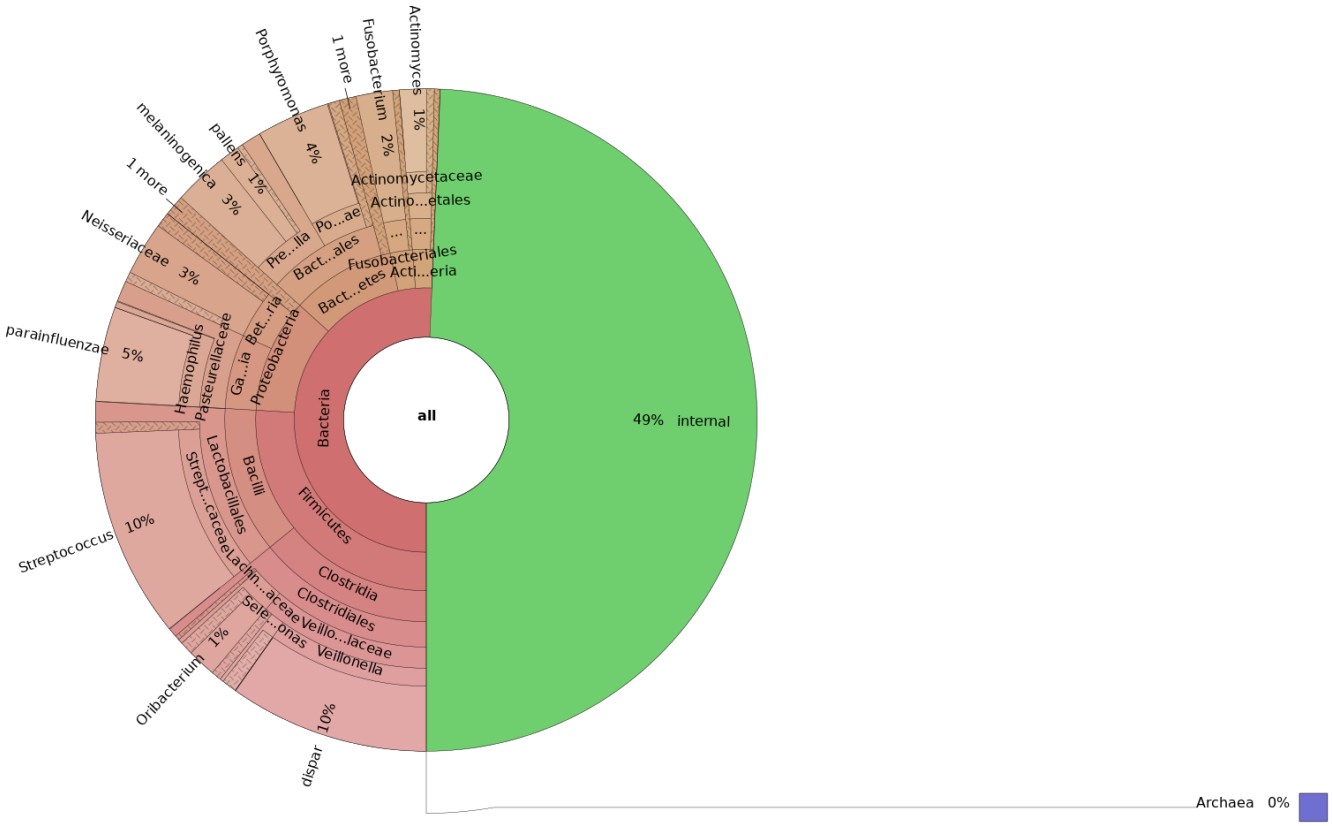
(b) L1 Oral Cavity Krona Graph



(c) L1 Vaginal Krona Graph



(d) L1 Skin Krona Graph

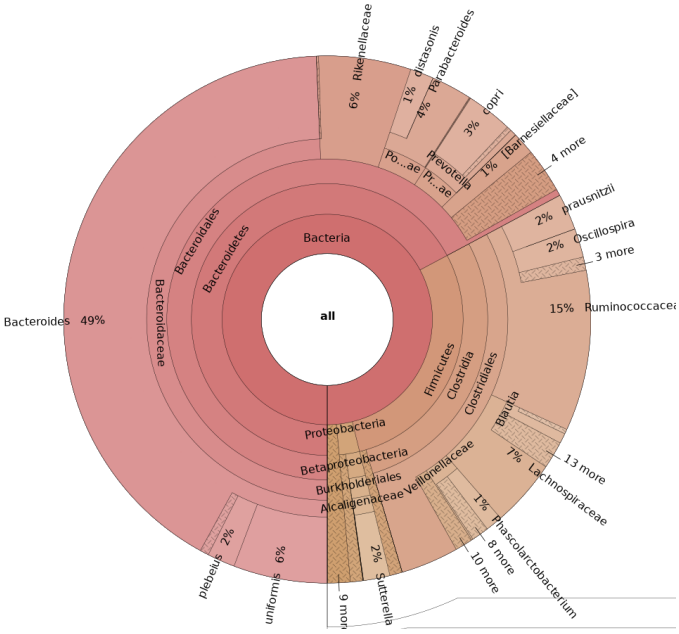


(e) L1 Saliva Krona Graph

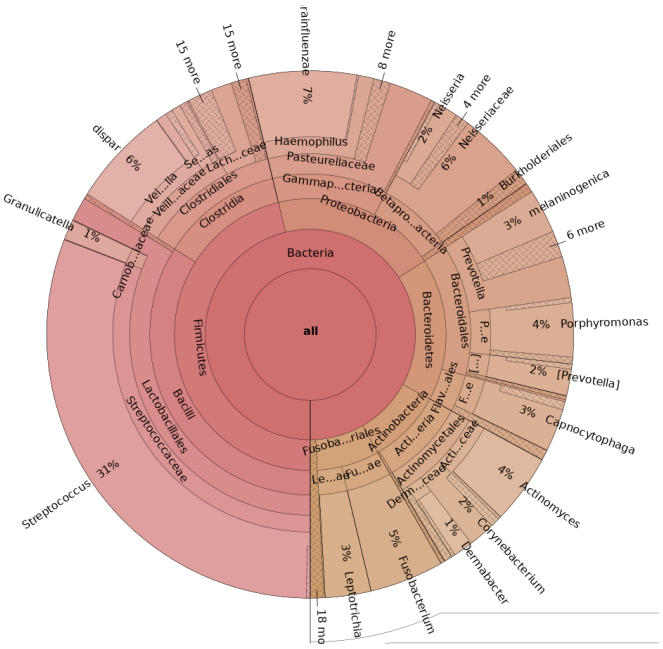
Figure 10: The above figures illustrate the taxonomic distribution of mass using L1-UniFrac's direct application to averages of body sites. Figure (a) shows the distribution of the fecal body site. Figure (b) shows the distribution of the oral cavity body site. Figure (c) shows the distribution of the vaginal body site. Figure (d) shows the distribution of the skin body site. Figure (e) shows the distribution of the saliva body site. Each figure represents the total taxonomy of every OTU, measured by cumulative abundance for that taxonomy. For instance, if the average sample in a set contain a cumulative 50% bacteria and 25% proteobacteria, proteobacteria would appear as a subset of bacteria, accounting for 50% of the bacteria semicircle and 25% of the entire circle. The graph is read from inside outwards, with the inner-most ring representing the kingdom, followed by the phylum, then class, order, family, genus, and finally species. Several of the most prominent OTUs in each body site are labeled directly.

The above figures in fig. 10 represent the taxonomic distribution across each of the five body sites in the experimentation. Notice that in every case, one of the major components is a segment labeled as "internal." Internal corresponds to temporary internal nodes within the phylogenetic tree. Such nodes exist only to act as common ancestors between multiple OTUs that themselves do not exist today. In evolutionary biology, when species diverge from the common ancestor across different lineages, that common ancestor no longer exists directly and would have gone "extinct" in a sense, but the OTUs that are derived from it are related to it directly. The extent to which they differ can be very small, but such divergence points are necessary to construct a fully connected phylogenetic tree as is used in this experimentation.

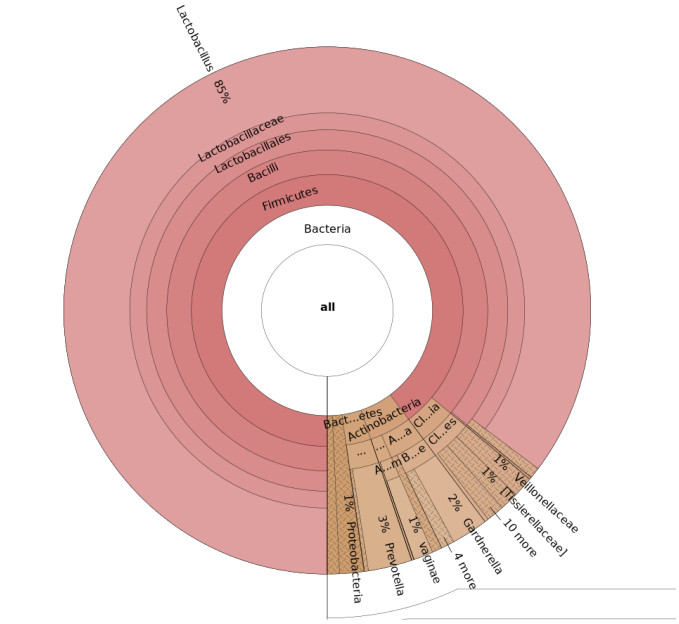
As a result, seeing many internal nodes within these Krona graphs indicates the L1-UniFrac algorithm leaves a large portion of the mass on such internal nodes, showing more about evolutionary commonalities than being applicable as a metric from which averages can be derived. In this sense, L1-UniFrac shows it is not effective at computing averages since as much as 74% of the mass in the case of fig. 10b is not actually representative of modern species and OTUs that may feasibly be present within a real-world sample. It is, therefore, not very productive to state that a presently non-existent OTU composes some non-zero (or very close to zero) percentage of the real-world sample when its presence is not physically possible. This was the same problem experienced in section 5.5.



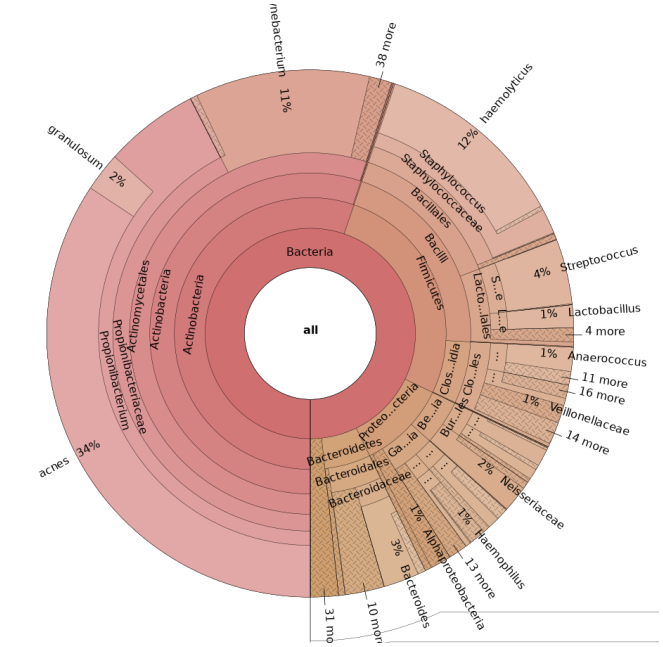
(a) L2 Feces Krona Graph



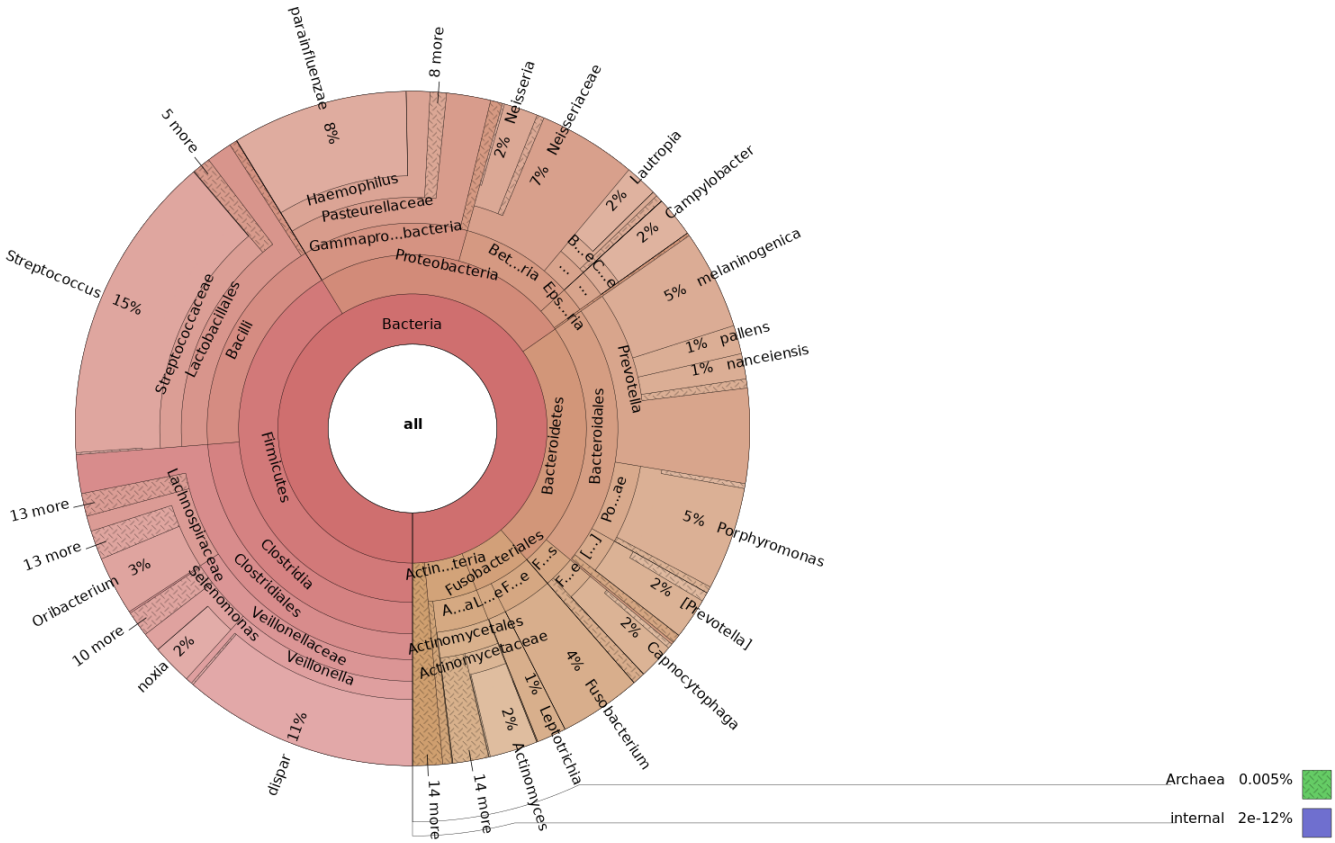
(b) L2 Oral Cavity Krona Graph



(c) L2 Vaginal Krona Graph



(d) L2 Skin Krona Graph



(e) L2 Saliva Krona Graph

Figure 11: The above figures illustrate the taxonomic distribution of mass using L2-UniFrac averages of body sites. Figure (a) shows the distribution of the fecal body site. Figure (b) shows the distribution of the oral cavity body site. Figure (c) shows the distribution of the vaginal body site. Figure (d) shows the distribution of the skin body site. Figure (e) shows the distribution of the saliva body site. Each figure represents the total taxonomy of every OTU, measured by cumulative abundance for that taxonomy. For instance, if the average sample in a set contain a cumulative 50% bacteria and 25% proteobacteria, proteobacteria would appear as a subset of bacteria, accounting for 50% of the bacteria semicircle and 25% of the entire circle. The graph is read from inside outwards, with the inner-most ring representing the kingdom, followed by the phylum, then class, order, family, genus, and finally species. Several of the most prominent OTUs in each body site are labeled directly.

In the L2-UniFrac variants of Krona shown in fig. 11, the results show a much larger variety of OTUs on first glance. Two prominent examples are fig. 11e and fig. 11c for different reasons. In fig. 11e, we see the most variety at the outermost layer of the Krona graph, something that was not present in fig. 10e under L1-UniFrac due to the graph being largely consumed by internal nodes. In contrast, fig. 11e displays significantly more detail down to the genus and species level that is otherwise captured more frequently in internal nodes under existing methods of L1-UniFrac. Additionally, fig. 11c illustrates another interesting phenomenon more directly. In its L1 counterpart of fig. 10c, the mass is entirely located in either the genus of Lactobacillus (87%) or in internal nodes (13%).

Moreover, no mass is detected in L1-UniFrac in any other lowest-level classification. In contrast, L2-UniFrac finds 85% of the mass in Lactobacillus as before as well as several other Lactobacillales accounting for a cumulative 1%. Additionally, 3.7% of the L2 Krona chart is composed of the Clostridia class, including several derivatives, 1.4% is composed of the Proteobacteria phylum (to be discussed shortly), 3% is composed of the Bacteroidetes phylum, 5% is composed of the Actinobacteria phylum, and the rest is composed of trace quantities of other taxonomic classifications. The following figure demonstrates a zoomed-in perspective of the Proteobacteria phylum which only exists in L2-UniFrac's averaged Krona chart for the vaginal region.

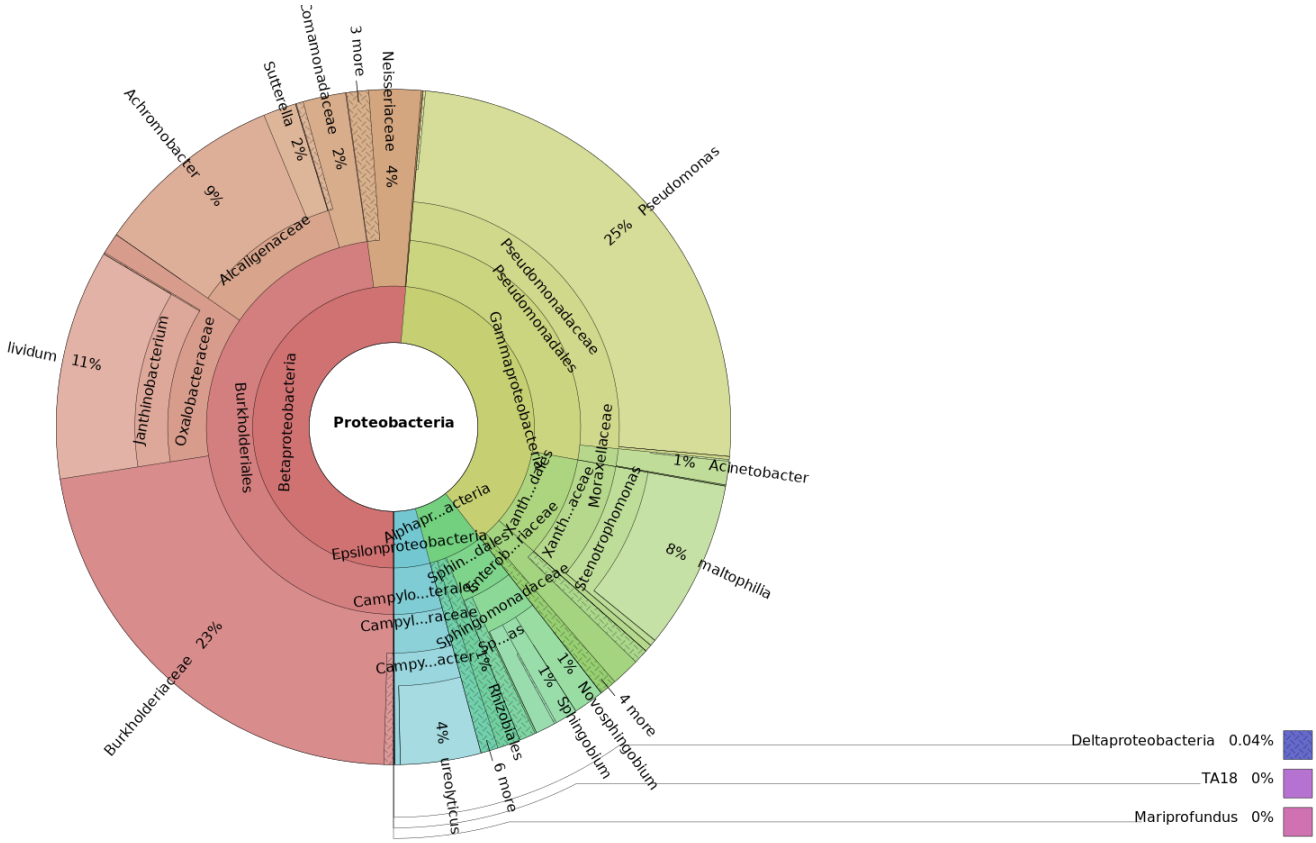


Figure 12: Focused visualization of the proteobacteria phylum from the context of the L2-UniFrac average of the vaginal microbiome. The graph is read from inside outwards, with the inner-most ring representing the kingdom, followed by the phylum, then class, order, family, genus, and finally species. Several of the most prominent OTUs in each body site are labeled directly. The abundances at each element correspond to the percentage of the proteobacteria phylum that is made up of that taxonomic classification, not the percentage of the entire mean sample. Color differentials represent larger degrees of separation at earlier taxonomic classifications within the subset of the proteobacteria phylum.

Despite making up a cumulative 1.4% of the total averaged sample within the vaginal region in the L2-UniFrac mean sample and not being accounted for at all as a separate entity in L1-UniFrac's median, a distinct abundance of biodiversity is present. This distinction demonstrates a significant advantage that L2-UniFrac holds above L1-UniFrac

with regards to abundance distribution on existing OTUs as opposed to bundling many of them into internal nodes as the mass is moved around the phylogenetic tree in the averaging process. All of this information and biodiversity was hidden in the L1-UniFrac average as it was present within internal nodes rather than on real-world OTUs.

One of the more interesting, yet subtle, observations present within fig. 11, specifically fig. 11e, is the presence of 0.005% Archaea. This is not very much, and we would not expect there to be much outside of bacteria within our samples. However, it is noticeably absent from fig. 10e which notes a presence of 0%. This fact was verified this by searching through each of the OTUs that comprise each of the two Krona, finding that indeed no abundance of Archaea is present in the average on L1 while a very small but non-zero amount is present in the L2 average. The specific entry responsible for this, in the case of the saliva, was the genus of Methanobrevibacter, the dominant archaeon present within the human gut microbiome [45], with a total average abundance of $5.437054411026653 \times 10^{-5}$ (or roughly 0.005% as noted earlier). Since Methanobrevibacter is known to be present within the human gut microbiome, this corroborates our findings from L2-UniFrac's average sample that shows it in small but non-zero findings where prior methods often discarded this information entirely. Since L1-UniFrac utilizes a component-wise median when averaging groups, it is possible, as with the aforementioned case of Methanobrevibacter, that a majority ($> 50\%$) of the samples find no presence of it while a small minority of samples do. Consequently, these infrequent abundances are completely ignored. L2-UniFrac provides greater granularity in observations to detect such abundances within averaged data.

As noted earlier, the primary observation between L1 and L2-UniFrac's Krona charts is the overwhelming presence of internal nodes in L1-UniFrac-generated charts. With the exception of the vaginal graph, temporary internal nodes make up roughly half to three quarters of the entire graph when these nodes are, for all intents and purposes, mass that is shared by many downstream taxonomies and is not assigned one direct taxonomy since that evolutionary branch point does not exist as it only serves a marker of common ancestors between two modern taxonomic groups. However, when compared directly with L2-UniFrac Krona visualizations, the internal nodes almost entirely vanish, consuming a rather insignificant $2 \times 10^{-12}\%$ of the total mass.

By averaging entirely into leaf nodes, L2-UniFrac presents a much more useful metric for sample averaging in real-world samples to find the expected abundances of particular real-world OTUs when sampling a region, body site, or other potential microbial community of interest. L2-UniFrac also maintains a higher level of granularity between OTUs, allowing for more fine-grained representation across many communities and giving insight into some of the less abundant OTU presences within samples that would otherwise be glossed over by L1-UniFrac. Therefore, our

hypothesis for this experiment is supported by our evidence.

5.7 Di erential Abundance

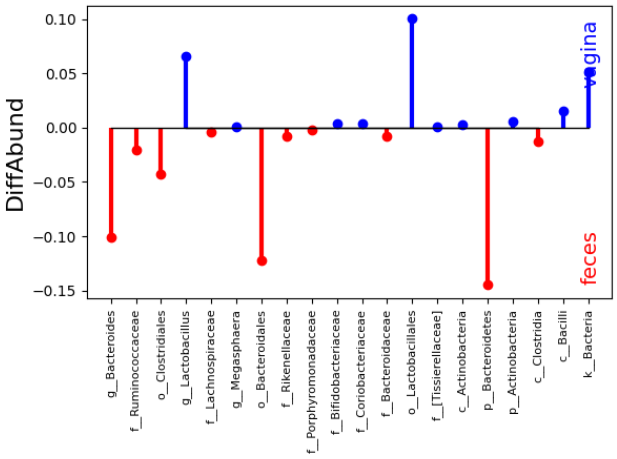
As outlined earlier in section 4.6, we also looked to determine the most differentially abundant taxonomic unit between each set of average samples. For reference, the definition of differential abundance vectors is given in [29] at the end of section 2. Since we are taking averages of many samples, L2-UniFrac enables such a process while reducing potential sampling bias and variance in the conclusions. The smallest individual group has 349 samples while the largest has 2982 samples, meaning that the mean among the strong clusters that emerged in fig. 2 and plotted in fig. 6b is likely very close to the ground-truth mean.

With that said, we obtained two different sets of results. The first of these sets is the results obtained by using the algorithm from section 4.6 to assign universally shared taxonomy to every internal node, based on their descendants. This method will give a phylogenetic and evolutionary perspective into the differentially abundant aspects of the two samples, especially when the body sites share less evolutionary history but has several flaws that will be addressed with the second set of results. This flaw in this approach is that multiple internal nodes can share the same common taxonomy. For instance, many internal nodes may only share the kingdom of bacteria with all of their descendant leaf nodes. In such a case, we would see many non-differentiable instances of bacteria appear in the differential abundance graph. To accommodate for this the universally shared taxonomy case will sum together all common taxonomies before plotting. Additionally, in such cases, we will also show the plots without regard for the differential abundance of internal nodes (leaf-only taxonomy). This alternative provides a more real-world perspective since it only relies on the differential abundance among OTUs directly, leaving out any OTUs that no longer exist. This process does leave out some of the evolutionary perspective noted in the first approach, but it has greater applications areas such as the diagnosis of and comparison between healthy and unhealthy human microbiomes.

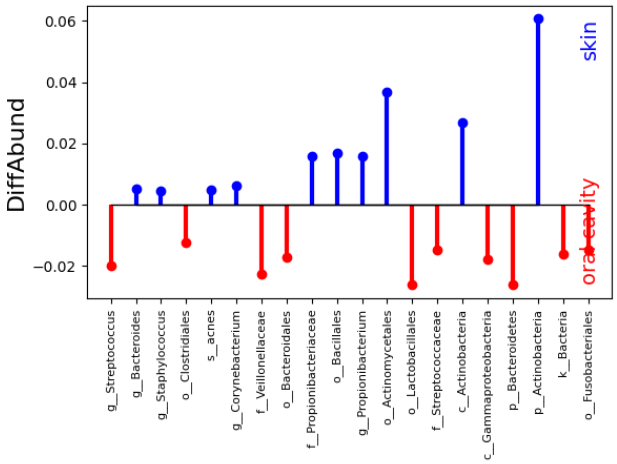
Finally, with regards to the charts for this section, there are twenty charts for each of the two plotting schemes, with ten unique combinations of body sites for each of L1 and L2-UniFrac. However, many of these charts convey very similar information, so for the sake of brevity, we will focus on a core selection of the charts that demonstrate the general observations present in all of the charts. Additionally, as mentioned earlier, the core goal of this section is to focus on L2-UniFrac since L1-UniFrac is fundamentally a direct application of existing algorithms to a use case we have found it to be ill-suited to handle.

To begin with the universally-shared taxonomy differential abundance charts, we obtain the following graphs with

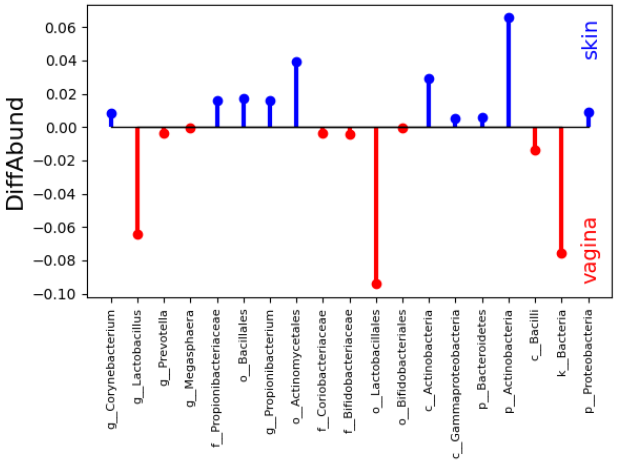
some representation from each of the body sites for L2-UniFrac.



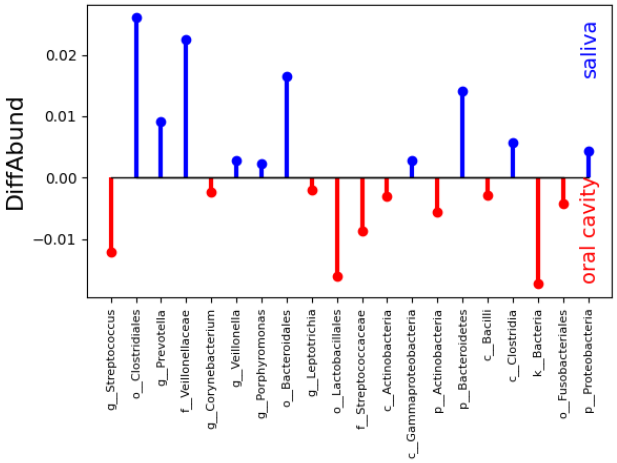
(a) L2 Vagina and Feces Di erential Abundance



(b) L2 Skin and Oral Cavity Di erential Abundance



(c) L2 Skin and Vagina Di erential Abundance



(d) L2 Saliva and Oral Cavity Di erential Abundance

Figure 13: Several selections of differential abundance plots with coverage across all body sites. Figure (a) shows the differential abundances between the average feces and average vaginal body site. Figure (b) shows the same for the average skin and average oral cavity body site. Figure (c) shows this for the average skin and average vaginal body site. Finally, figure (d) shows this for the average saliva and average oral cavity body site. Some overlap between samples is maintained for clarity. The differential abundance measures the difference between the abundances of each OTU from one average sample to the other and accumulates the differential mass of each OTU's respective subtree times the edge length leading to that subtree. This measure shows which subsets of the tree are more common in one sample compared with another. The classification for this figure is completed by finding which taxonomic classification is shared among all leaf nodes of that respective subtree and reporting the lowest-level shared classification available, with species being the lowest and kingdom being the highest. The magnitude of the differential abundance measure represents the average weighted differential between the two body sites with 0 representing no difference and non-zero numbers representing larger and larger differences.

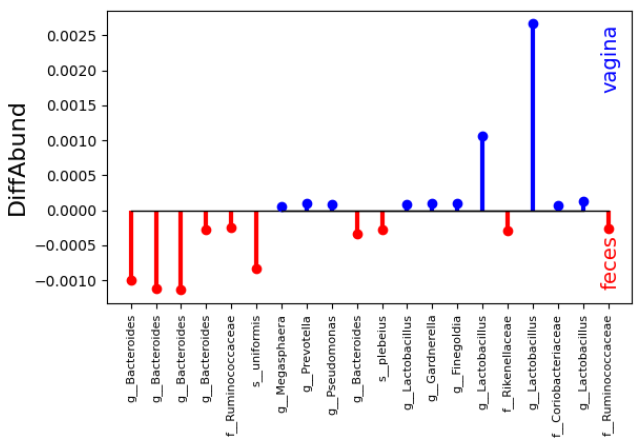
In the above figures, we can note several of the higher-order differences between each of the body sites on an evolutionary perspective. Since these figures are generated with a universally shared taxonomic scheme among descendants, many of the more prominent differential abundances will be displayed at higher taxonomic orders, which allows for some insight into the broader scope of the composition of a particular microbiome relative to others. To begin with fig. 13a, it is evident that the feces body site has a much more apparently differential abundance with regards to the phylum Bacteroidetes, the order Bacteroidales, and the genus Bacteroides whereas the vaginal body site has a higher prevalence of the order Lactobacillales, genus Lactobacillus, and kingdom of Bacteria as a whole. Compared with the other body-site comparisons, the differential abundances in this figure are generally of a higher magnitude, implying a larger divergence between the two body sites.

In comparison, looking at the saliva and oral cavity pair which were established to be very closely related in our PCoA analysis earlier, fig. 13d shows over an order of magnitude reduction in the differential abundant quantities on both sides of the plot compared to the previous chart. Additionally, the differences that do emerge primarily lie at the genus Streptococcus, order Lactobacillales, and kingdom Bacteria for the oral cavity and order Clostridiales, family Veillonellaceae, and order Bacteroidales for the saliva microbiome. Thus, as a result of their very low relative differential abundances, it is evident experimentally that these two body sites share significant overlap in microbiome composition, likely as a result of their very close physical proximity on the body.

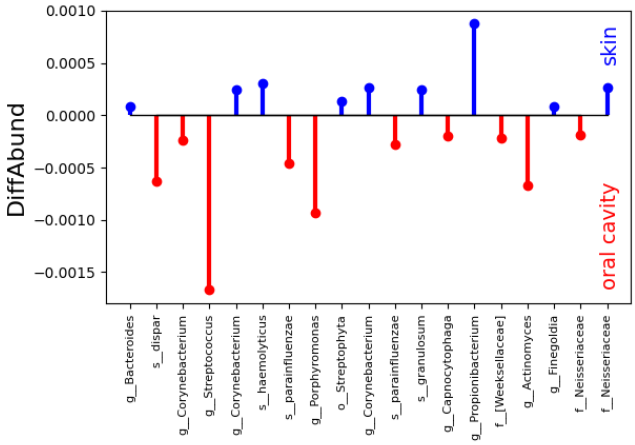
To briefly touch on the last body site that is shown in fig. 13b and fig. 13c, the skin has a modest magnitude of differential abundance compared to both of the other body sites, and the primary point of divergence seen through the charts is in the order Actinomycetales, phylum Actinobacteria, and class Actinobacteria, so in the case of this experimentation, we would expect that, given a sample with higher-than-normal differential Actinomycetales abundances compared to others, it would be most probabilistically fall in the skin microbiome if our choice set was restricted to only five microbiomes.

Finally, it is important to note that these observations of specific differentially abundant OTUs at each body site continued for each pairwise comparison graph. The additional six comparison graphs were omitted for the lack of additional insight that would be gained in their inclusion.

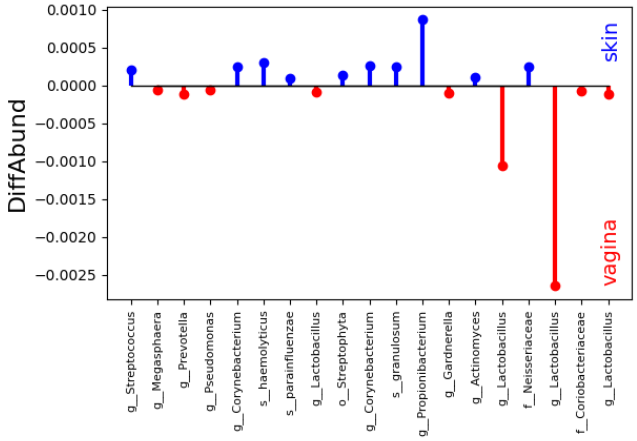
To achieve a higher level of granularity on each of these graphs with an emphasis on real-world application, we can instead narrow our focus on the leaf nodes exclusively. The following graphs are the same body site pairs but restricted to the differential abundances of leaf nodes exclusively.



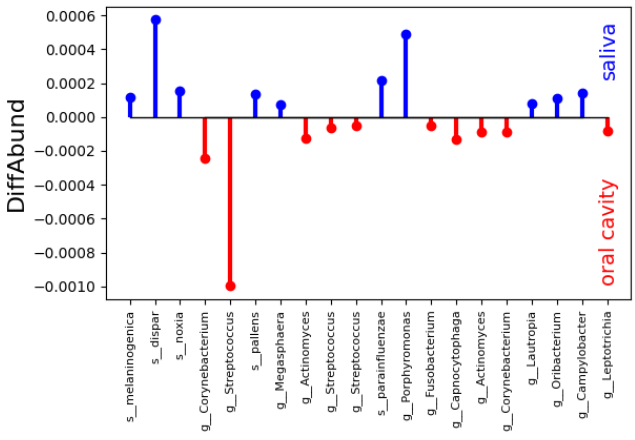
(a) L2 Vagina and Feces Di erential Abundance



(b) L2 Skin and Oral Cavity Di erential Abundance



(c) L2 Skin and Vagina Di erential Abundance



(d) L2 Saliva and Oral Cavity Di erential Abundance

Figure 14: Several selections of differential abundance plots with coverage across all body sites, measuring only OTUs currently existing in the real-world. Figure (a) shows the differential abundances between the average feces and average vaginal body site. Figure (b) shows the same for the average skin and average oral cavity body site. Figure (c) shows this for the average skin and average vaginal body site. Finally, figure (d) shows this for the average saliva and average oral cavity body site. Some overlap between samples is maintained for clarity. The differential abundance measures the difference between the abundances of each OTU from one average sample to the other and accumulates the differential mass of each OTU's respective subtree times the edge length leading to that subtree. This measure shows which subsets of the tree are more common in one sample compared with another. The classification for this figure is completed by directly converting each OTU of interest into its respective taxonomic classification using a reference and then displaying its lowest level taxonomy available with species being the lowest and kingdom being the highest. The magnitude of the differential abundance measure represents the average weighted differential between the two body sites with 0 representing no difference and non-zero numbers representing larger and larger differences.

With leaf nodes as the primary source, we begin to see much more information about the actual microbes present within each sample. It is very important to note that with L1-UniFrac, a leaf node analysis would not be quite as meaningful as a majority of the mass lies in the internal nodes, leaving very little differential abundances at the leaves. With that said, these charts again represent the L2-UniFrac differential abundances but purely restricted to leaf nodes.

Starting with fig. 14a, the visuals are very apparent on the species level of where everything lies. The vaginal microbiome still shows strong differential abundance within the genus of Lactobacillus which is known to be a common microbe within this body site in past literature [27]. Additionally, we can now gain insight into the lower-level taxonomies within the feces microbiome. From our experimentation, we have found a high prevalence in the genus of Bacteroides and species Uniformis which both are known to be common in this microbiome and not present in appreciable quantities in others [35].

Continuing on with fig. 14d, we can now see much more apparent differences between the two microbiomes. Most notably, while we continue to see the oral cavity dominated with Streptococcus as before, saliva now shows the presence of several species of microbes within its microbiome that are more commonly abundant than elsewhere in the oral cavity. These microbes include the species Dispar, species Parainfluenzae, and genus Porphyromonas. Dispar, specifically Veillonella dispar from the observations of the family seen in fig. 13d, has been sampled directly from the tongue according to past papers [9]. Parainfluenzae would be more expected to be prevalent in the respiratory system, but as this was not a body site from the sample, more data would be needed to have a proper comparison for this specific microbe.

Finally, we can obtain some deeper information into the skin microbiome. While the observations for each of the oral cavity and vagina remain consistent with fig. 14b and fig. 14c, these new observations in these figures for skin show very important and strong results. In both cases, the most strongly differentially abundant OTU is the genus Propionibacterium, a very common bacteria that is responsible for causing acne vulgaris on the skin [40]. As a result, our recorded observations with differential abundance match our expectations given the available body of research.

Given these findings, the most differentially abundant organism or OTU can be identified for each group that all coincidentally correspond with prior literature. These OTUs are those that are present within each group at higher rates than all other groups on average. Using the prior results, we can fill in the following table.

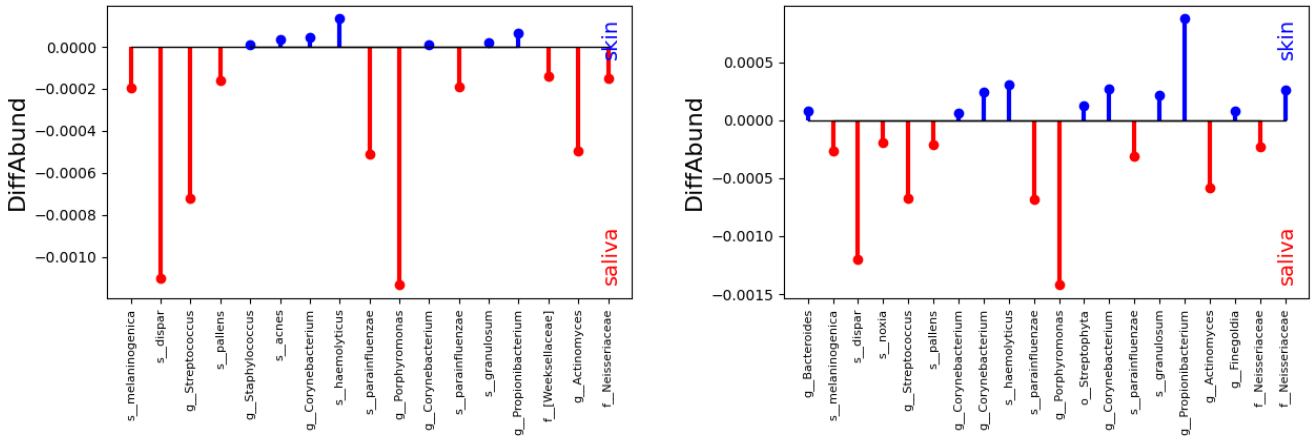
Body Site	Most Representative OTU
Skin	g__Propionibacterium
Saliva	s__Dispar
Oral Cavity	g__Streptococcus
Vagina	g__Lactobacillus
Feces	g__Bacteroides

Table 5: A table outlining the OTU that was observed to be the most differentially abundant for each body site average. In the event that multiple OTUs were more differentially abundant in some comparisons and not others, the competing OTU with the lowest taxonomic classification was selected, with species as the lowest-level taxonomic classification and kingdom as the highest.

Note that in some categories, there were multiple OTUs that were consistently differentially abundant. However, for the purposes of this table, the most representative OTU was selected based on which was typically more differentially abundant or by prioritizing lower-level taxonomy in cases where the OTUs were very close (e.g. species Veillonella Dispar and genus Porphyromonas in saliva).

It is also important to note that these findings can be corroborated by looking back at the Krona visualizations for L2-UniFrac. For example, in the case of Veillonella dispar's high differential abundance in the saliva relative to the oral cavity, we see that dispar makes up 11% of the mass in the saliva according to our Krona graphs, compared with only 6% of the oral cavity which are both shown in fig. 11.

For the sake of discussion, it is also possible to conduct this experimentation using L1-UniFrac, as mentioned earlier. However, since L1-UniFrac is not exactly designed for these purposes as established in prior sections of this paper, not many further conclusions will likely come from such a discussion. Thus, rather than delving into a detailed analysis between the minor nuances of these two constructions, we will briefly compare the skin and saliva differential plots between L1 and L2-UniFrac using the leaf nodes variant for the purposes of a short discussion. These plots are displayed below in fig. 15.



(a) L1 Skin and Saliva Di erential Abundance

(b) L2 Skin and Saliva Di erential Abundance

Figure 15: A comparitative selection of L1-UniFrac average differential abundances in the skin vs saliva body sites (a) and the L2-UniFrac average differential abundances in the same body site (b). The differential abundance measures the difference between the abundances of each OTU from one average sample to the other and accumulates the differential mass of each OTU's respective subtree times the edge length leading to that subtree. This measure shows which subsets of the tree are more common in one sample compared with another. The OTUs used for these computations were constrained to just leaf nodes (OTUs that can exist in the real-world). The classification for this figure is completed by directly converting each OTU of interest into its respective taxonomic classification using a reference and then displaying its lowest level taxonomy available with species being the lowest and kingdom being the highest. The magnitude of the differential abundance measure represents the average weighted differential between the two body sites with 0 representing no difference and non-zero numbers representing larger and larger differences.

Typically, around ten OTUs are selected, providing the differential abundance is greater than a filtering threshold of 0.000005, with lower values being deemed too insignificant to use. With that said, in the two graphs above, only L2-UniFrac has the maximum number of elements for both OTUs. L1-UniFrac, by contrast has much lower differential abundances on the skin metric, causing some OTUs to no longer be present. In nearly every one of the other comparisons, L1-UniFrac is shown to be less sensitive to differential abundances, sometimes by as much as an entire order of magnitude in comparison to L2-UniFrac for oral cavity results in the results between oral cavity and saliva. This observation is, naturally, explained by the issue where mass is primarily concentrated along internal nodes as discussed in prior sections. This then leaves less mass at the leaf nodes to be used in differential abundance testing, leading to lower thresholds which under represent some key distinctions between the body sites and reduce confidence. As a result, for differential abundance testing, it is only natural to utilize L2-UniFrac due to its numerous benefits and identifying characteristics.

From this set of experiments, we have shown our initial hypothesis to be supported. Each of the most differentially abundant OTUs in the pairwise comparison between each body site are specifically noted by past literature to be present within such microbiomes. This finding shows that L2-UniFrac averages are meaningful on the individual OTU scale as well, allowing such averages to rapidly find specific OTUs most responsible for discrepancies between two different microbiomes.

6 Conclusion

L2-UniFrac describes an alternative characterization of the EMDUniFrac metric with the goal of providing a more natural representation of biological phenomena with regards to the average abundances within communities. In this paper, we outline a modification to EMDUniFrac that changes the L1-norm accumulation of weighted mass into the L2-norm accumulation of weighted mass. Additionally, we formally prove that L1 medians cannot take meaningful averages of probability vectors with respect to the UniFrac metric whereas L2 means can. We further propose algorithms to directly use the Wasserstein matrix from [28] and modifications to the push up and inverse push up functions, first developed in unpublished work that applied EMDUniFrac directly medians, to accommodate for preprocessing with L2-UniFrac. Our proposed technique computes it with respect to the L2-norm and corresponds with W^{p}- and Wp^{1} respectively.

In our experimentation, we outline several key tests that we performed in order to assess the differences between the direct application of EMDUniFrac to medians, denoted L1-UniFrac, and our new method, denoted L2-UniFrac, as well as other evaluation tools that can highlight usage of this work. The primary experiments that were performed include testing for the presence of negative abundances in averaged samples, principal coordinate analyses of all samples, principal coordinate analyses of sample averages, clustering performance analyses, average taxonomic distribution of averages, Krona analyses, and differential abundance testing. Following each of these tests, we are able to draw several conclusions.

Firstly, to validate that L2-UniFrac serves as a valid distance metric when directly compared with EMDUniFrac (L1-UniFrac), we compared their performance on an EMDUniFrac was designed for by comparing with individual samples. That is, this experiment tested the direct published work against our modification to utilize L2-norms instead of L1-norms. The results of this section were slightly that not only did L2-UniFrac illustrate clustering characteristics that are absolutely essential to a useful UniFrac metric, but also, L2-UniFrac even exceeded expectations at some points, with particularly stronger visual clustering at the center of the PCoA plot. As a result, our hypothesis that they would perform similarly was supported.

Following the direct comparison of L2-UniFrac with L1-UniFrac with PCoA plots which showed strong visual clustering, demonstrating similar performance on this benchmark, it was also necessary to check if this clustering performance would hold under a blind clustering algorithm. This second test resulted in L2-UniFrac slightly lagging behind L1-UniFrac when scored using scoring metrics, but following a visual validation of the clustering algorithms

compared to the ground-truth baseline, it was clear that L2-UniFrac performed similarly to L1-UniFrac overall. The score differential that was noted was primarily present in the direct application of blind scoring algorithms, whereas for researchers, who rely on tools such as PCoA plots, the clustering was much clearer and applicable to the baseline. Our hypothesis that both UniFrac variants would perform similarly was supported.

It was also necessary to test L2-UniFrac's core ability to take averages. However, since L2-UniFrac is the first implementation of UniFrac that natively supports averages in a manner that makes sense in a biological context, we decided to use a direct application of L1-UniFrac to averages as a point of comparison. L1-UniFrac was used for some comparisons to show the disadvantages when necessary.

As anticipated following the proofs of proposition 1 and proposition 2, the presence of negatives was only experimentally measured beyond an error threshold in L1-UniFrac whereas L2-UniFrac showed no such presence of negative sample abundances. To verify that this count was not impacted by an arbitrary error threshold, we analyzed the values and found a six to nine order of magnitude difference between the threshold and every negative, thus showing they composed a significant portion of the mass and supporting our hypothesis.

To demonstrate that averaged vectors in L2-UniFrac are meaningful, we performed a simple analysis of the averaged vectors for each body site on a PCoA plot against each other. The results supported our hypothesis and show that L2-UniFrac is fully capable of taking usable and interpretable averages of subsets of data that cluster closely to similar averages and far apart from dissimilar averages, as determined using the baseline.

From this point, we analyzed the distribution of L2-UniFrac averages across two different experiments, namely through a direct taxonomic analysis of lowest-level taxonomy across all OTUs and by a Krona visualization analysis. Both of these tests resulted in the same primary conclusion that L2-UniFrac averages lead to a more natural representation of OTUs that are present in the real world whereas the direct application of L1-UniFrac to averages fails to accomplish such a task. Additionally the Krona analysis using L2-UniFrac averages showed a much greater variety and diversity in OTUs that would otherwise be either absorbed in internal nodes or hidden altogether due to the OTU being present only in a minority of samples. These observations support our hypotheses and show very large utility in L2-UniFrac for identifying the exact composition of the average sample within a body site or other microbiome that is otherwise not possible with any other existing methods.

Finally, the last experiment conducted explored the application of L2-UniFrac to differential abundance testing, in order to identify OTUs that are characteristic elements of a microbiome in the averaged case. The goal of this test was to simply be able to use L2-UniFrac to find OTUs that are known to be present within each body site

through a brief literature review. Should the most differentially abundant OTU for every body site be validated as a predominant microbe within that region of the human body or originate from somewhere nearby in appreciable quantities, it would confirm L2-UniFrac's ability to compute reliable averages that correspond to known data. In our results, we were able to show that L2-UniFrac consistently ranked OTUs that are known to be common within each body site as the most differentially abundant and defining characteristic of that body site. As a result, L2-UniFrac supports our hypothesis and provides a useful tool to apply to new research studies where such OTUs may not already be known in order to determine such OTUs. A particularly relevant use case for this ability of L2-UniFrac would exist when comparing many samples of a healthy microbiome and an unhealthy microbiome to see which OTUs may be more abundant that may result from or be a cause of the condition and may warrant further study.

In the end, this work presented a new characterization of the UniFrac metric that enables the computation of representative microbiomes of subsets of data. Such representative samples can then be used for rapid analysis of the microbiome as a whole. Our approach presents very large advantages over other state-of-the-art methods without notable sacrifices in terms of performance under workloads designed for prior methods.

References

[1] J. Alman and V. V. Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. SIAM, 2021.

[2] L. F. Ana and A. K. Jain. Robust data clustering. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–II. IEEE, 2003.

[3] I. Borg and P. J. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[4] Q. Chang, Y. Luan, and F. Sun. Variance adjusted weighted unifrac: a powerful beta diversity measure for comparing communities based on phylogeny. *BMC bioinformatics*, 12(1):1–14, 2011.

[5] H. M. P. Consortium et al. Structure, function and diversity of the healthy human microbiome. *nature*, 486(7402):207, 2012.

[6] T. M. Cover and J. A. Thomas. Elements of information theory second edition solutions to problems. *Internet Access*, pages 19–20, 2006.

[7] P. Dawyndt, H. De Meyer, and B. De Baets. The complete linkage clustering algorithm revisited. *Soft Computing*, 9(5):385–392, 2005.

[8] T. Z. DeSantis, P. Hugenholtz, N. Larsen, M. Rojas, E. L. Brodie, K. Keller, T. Huber, D. Dalevi, P. Hu, and G. L. Andersen. Greengenes, a chimera-checked 16s rrna gene database and workbench compatible with arb. *Applied and environmental microbiology*, 72(7):5069–5072, 2006.

[9] J. J. Doel, N. Benjamin, M. P. Hector, M. Rogers, and R. P. Allaker. Evaluation of bacterial nitrate reduction in the human oral cavity. *European journal of oral sciences*, 113(1):14–19, 2005.

[10] S. Evans and F. Matsen. The phylogenetic kantorovich-rubinstein metric for environmental sequence samples. *Journal of the Royal Statistical Society. Series B, Statistical methodology*, 74:569–592, 2012.

[11] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, 78(383):553–569, 1983.

[12] J. Friedman, T. Hastie, R. Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[13] A. J. Gates and Y.-Y. Ahn. The impact of random models on clustering similarity. *arXiv preprint arXiv:1701.06508*, 2017.

[14] A. Gonzalez, J. A. Navas-Molina, T. Kosciolek, D. McDonald, Y. Vázquez-Baeza, G. Ackermann, J. DeReus, S. Janssen, A. D. Swafford, S. B. Orchanian, et al. Qiita: rapid, web-enabled microbiome meta-analysis. *Nature methods*, 15(10):796–798, 2018.

[15] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of intelligent information systems*, 17(2):107–145, 2001.

[16] M. Hamady, C. Lozupone, and R. Knight. Fast unifrac: facilitating high-throughput phylogenetic analyses of microbial communities including analysis of pyrosequencing and phylochip data. *The ISME Journal*, 4(1):17–27, 2010.

[17] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020.

[18] L. Hubert and P. Arabie. Comparing partitions. *Journal of classi cation*, 2(1):193–218, 1985.

[19] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[20] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[21] G. Jing, L. Liu, Z. Wang, Y. Zhang, L. Qian, C. Gao, M. Zhang, M. Li, Z. Zhang, X. Liu, et al. Microbiome search engine 2: a platform for taxonomic and functional search of global microbiomes on the whole-microbiome level. *Msystems*, 6(1):e00943–20, 2021.

[22] G. Jing, Y. Zhang, L. Liu, Z. Wang, Z. Sun, R. Knight, X. Su, and J. Xu. A scale-free, fully connected global transition network underlies known microbiome diversity. *bioRxiv*, 2020.

[23] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 1990.

[24] L. I. Kuncheva and S. T. Hadjitodorov. Using diversity in cluster ensembles. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, volume 2, pages 1214–1219. IEEE, 2004.

[25] C. Lozupone and R. Knight. Unifrac: a new phylogenetic method for comparing microbial communities. *Applied and environmental microbiology*, 71(12):8228–8235, 2005.

[26] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[27] L. Mancabelli, W. Mancino, G. A. Lugli, C. Milani, A. Viappiani, R. Anzalone, G. Longhi, D. van Sinderen, M. Ventura, and F. Turroni. Comparative genome analyses of lactobacillus crispatus isolates from different ecological niches reveal an adaptation of this species to the human vaginal environment. *Applied and Environmental Microbiology*, 87(8):e02899–20, 2021.

[28] J. McClelland. Wasserstein β-diversity metrics over graphs: Derivation, efficient computation and applications. *Oregon State University*, 2018.

[29] J. McClelland and D. Koslicki. Emdunifrac: Exact linear time computation of the unifrac metric and identification of differentially abundant organisms. *Journal of mathematical biology*, 77(4):935–949, 2018.

[30] D. McDonald, J. C. Clemente, J. Kuczynski, J. R. Rideout, J. Stombaugh, D. Wendel, A. Wilke, S. Huse, J. Hufnagle, F. Meyer, et al. The biological observation matrix (biom) format or: how i learned to stop worrying and love the ome-ome. *Gigascience*, 1(1):2047–217X, 2012.

[31] D. McDonald, M. N. Price, J. Goodrich, E. P. Nawrocki, T. Z. DeSantis, A. Probst, G. L. Andersen, R. Knight, and P. Hugenholtz. An improved greengenes taxonomy with explicit ranks for ecological and evolutionary analyses of bacteria and archaea. *The ISME Journal*, 6(3):610–618, 2012.

[32] D. McDonald, Y. Vázquez-Baeza, D. Koslicki, J. McClelland, N. Reeve, Z. Xu, A. Gonzalez, and R. Knight. Striped unifrac: enabling microbiome analysis at unprecedented scale. *Nature methods*, 15(11):847–848, 2018.

[33] A. Mead. Review of the development of multidimensional scaling methods. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 41(1):27–39, 1992.

[34] M. Meilă. Comparing clusterings—an information based distance. *Journal of multivariate analysis*, 98(5):873–895, 2007.

[35] H. Nakanishi, H. Shojo, T. Ohmori, M. Hara, A. Takada, N. Adachi, and K. Saito. Identification of feces by detection of bacteroides genes. *Forensic Science International: Genetics*, 7(1):176–179, 2013.

[36] B. D. Ondov, N. H. Bergman, and A. M. Phillippy. Interactive metagenomic visualization in a web browser. *BMC bioinformatics*, 12(1):1–10, 2011.

[37] T. pandas development team. pandas-dev/pandas: Pandas, 2020.

[38] H.-S. Park and C.-H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341, 2009.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[40] A. Perry and P. A. Lambert. Propionibacterium acnes. *Letters in applied microbiology*, 42(3):185–188, 2006.

[41] M. N. Price, P. S. Dehal, and A. P. Arkin. Fasttree: computing large minimum evolution trees with profiles instead of a distance matrix. *Molecular biology and evolution*, 26(7):1641–1650, 2009.

[42] M. N. Price, P. S. Dehal, and A. P. Arkin. Fasttree 2–approximately maximum-likelihood trees for large alignments. *PloS one*, 5(3):e9490, 2010.

[43] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[44] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[45] B. S. Samuel, E. E. Hansen, J. K. Manchester, P. M. Coutinho, B. Henrissat, R. Fulton, P. Latreille, K. Kim, R. K. Wilson, and J. I. Gordon. Genomic and metabolic adaptations of methanobrevibacter smithii to the human gut. *Proceedings of the National Academy of Sciences*, 104(25):10643–10648, 2007.

[46] P. D. Schloss, S. L. Westcott, T. Ryabin, J. R. Hall, M. Hartmann, E. B. Hollister, R. A. Lesniewski, B. B. Oakley, D. H. Parks, C. J. Robinson, et al. Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Applied and environmental microbiology*, 75(23):7537–7541, 2009.

[47] T. scikit-bio development team. scikit-bio: A bioinformatics library for data scientists, students, and developers, 2020.

[48] C. Shannon. Tech. 27 (1948) 379; ce shannon, bell syst. *Tech*, 27:623, 1948.

[49] C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[50] D. Steinley. Properties of the hubert-arable adjusted rand index. *Psychological methods*, 9(3):386, 2004.

[51] A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[52] X. Su, G. Jing, D. McDonald, H. Wang, Z. Wang, A. Gonzalez, Z. Sun, S. Huang, J. Navas, R. Knight, et al. Identifying and predicting novelty in microbiome studies. *MBio*, 9(6):e02099–18, 2018.

[53] X. Su, G. Jing, Z. Sun, L. Liu, Z. Xu, D. McDonald, Z. Wang, H. Wang, A. Gonzalez, Y. Zhang, et al. Multiple-disease detection and classification across cohorts via microbiome search. *Msystems*, 5(2):e00150–20, 2020.

[54] J. Sukumaran and M. T. Holder. Dendropy: A python library for phylogenetic computing. *Bioinformatics*, 26:1569–1571, 2010.

[55] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:2837–2854, 2010.

[56] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van

Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[57] Q. Wang, G. M. Garrity, J. M. Tiedje, and J. R. Cole. Naive bayesian classifier for rapid assignment of rrna sequences into the new bacterial taxonomy. *Applied and environmental microbiology*, 73(16):5261–5267, 2007.

[58] J. J. Werner, O. Koren, P. Hugenholtz, T. Z. DeSantis, W. A. Walters, J. G. Caporaso, L. T. Angenent, R. Knight, and R. E. Ley. Impact of training sets on classification of high-throughput bacterial 16s rrna gene surveys. *The ISME Journal*, 6(1):94–103, 2012.

[59] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

Andrew J. Millward

andrew.j.millward@gmail.com
https://github.com/andrew-j-millward

EDUCATION

The Pennsylvania State University August 2018 - May 2022

Master of Science in Computer Science and Engineering

Bachelor of Science in Computer Science — Minor in Mathematics

EXPERIENCE

Teaching Assistant, The Pennsylvania State University August 2021 - May 2022

CMPSC 461 - Programming Language Concepts

- Hold two hours of weekly office hours to assist students in understanding course material
- Construct and grade homework assignments for 300+ student class
- Proctor and grade exams and assist students over class forum

Software Engineer Intern, Capital One June 2021 - August 2021

Richmond, VA

- Constructed proprietary API using JavaScript/Node.js for the Card Technology division at Capital One with a team of three other interns to fetch, cache, filter, and display management-requested data.
- Used Agile software development techniques in a Scrum framework to brainstorm, plan, build, test, and deploy new functionality to internal services in two-week cycles.
- Deployed QA-tested API to production, performed rigorous unit testing at >80% coverage, and branched out across the development stack to contribute to both front and back-end AWS services.

PROJECTS

Virtual Memory Manager, CMPSC 473 - Operating Systems March 2021 - April 2021

- Designed and implemented TLB and Page Table caches for frequently accessed memory regions in a three-person team. Included up to three level page-table support, both LRU and Clock eviction schemes for both. Maintained address integrity during eviction and retrieval from swap disk storage.
- Allowed for concurrent operation across up to 5 running processes with context switch handling.
- Kept track of 13 key metrics for each individual process to assess TLB/DRAM performance.

Siemens IOT2040/LR100 Bluetooth Connectivity, Siemens AG August 2020 - December 2020

- Worked with a six-person team to build a secure data transfer protocol between multiple LR100 field sensors to a central IOT2040 unit across Bluetooth using externally connected Arduino modules.
- Prepared and aggregated collected data into one packet in local database for transfer to the cloud.
- Managed project timeline and supplied updates to Siemens through weekly meetings to deliver a quality product that exceeded expectations. Presented project to sponsors using a working prototype.

COMPETENCES

Languages Python • JavaScript/Java • Vue.js/Node.js • C/C++ • HTML/CSS • LaTeX • SQL • MATLAB

Software and Tools Git • GitHub • AWS • Unit Testing • UNIX/Linux • Postman • SSH • VSCode

LEADERSHIP

Technology Director, TEDxPSU April 2020 - April 2022

- Manage six person TEDxPSU Technology team to update webpage, provide technical assistance and run slides at conference, publish team and speaker applications, and set up the conference livestream.
- Assist four teams in recruitment efforts, applicant screening, and general operations/logistics.

Eagle Scout, Boy Scouts of America August 2015 - May 2016

- Organized research effort and managed a group of eighteen scouts and adult volunteers to construct trail markers denoting 16 different tree species at Fairview High School in Erie, PA.
- Coordinated procurement of supplies from three different distributors for posts, signs, and tools.