Authors: Daniel McDonald, Yoshiki Vazquez Baeza, David Koslicki, Jason McClelland, Nicolai Reeve, Zhenjiang Xu, Antonio Gonzalez, Rob Knight

Title: Striped UniFrac: enabling microbiome analysis at unprecedented scale.

Abstract: UniFrac is synonymous with microbiome research, yet it no longer scales to large datasets. We propose a new algorithm, Striped UniFrac, which produces identical results, reduces space and time complexities by >10x and exhibits near linear parallel scaling. We highlight it by computing UniFrac on 113,721 samples in 48 hours using 256 CPUs. A BSD-licensed implementation is available that produces a C shared library linkable by any programming language.

[main text: presently 1389 words]

The study of the microbiome has rapidly expanded, largely because of the insight afforded by UniFrac [1]. UniFrac is a phylogenetic measure of beta-diversity that assesses differences between pairs of microbiome profiles, and provides the method underlying some of the field's most iconic insights [2–4]. UniFrac is central to microbial community studies because it accounts for evolutionary relationships between microbes present within a sample, whereas other distance metrics such as Euclidean distance, Bray-Curtis, and Jaccard make the unrealistic implicit assumption that all organisms are equally related (see [5] for more detail), leading to statistical artifacts on the resulting sparse data matrices.

Microbiome studies have recently transitioned from experimental designs with a few dozen samples to designs spanning tens of thousands of samples. Large-scale studies, such as the Earth Microbiome Project [6], afford the statistical crucial for untangling the many factors that influence microbial community composition. Unfortunately, these large studies make prohibitive demands on our algorithms and data structures. In the summer of 2015, we set out to apply UniFrac to the Earth Microbiome Project, a dataset spanning over 25,000 samples. We discovered that Fast UniFrac [7], irrespective of implementation, could not process this dataset even with months running on specialized hardware. Here we describe a novel algorithm with a shared library implementation usable by any programming language that can process the EMP dataset on a laptop in <24 hours.

Five important advances were made in the design of Striped UniFrac. First was to reduce the average case space requirement from by performing a postorder aggregation of the sample proportions to compute UniFrac at each internal node (fig. 1A; proof in supplemental text), similar to EMDUniFrac [8], avoiding the necessity of a dense matrix representation of sample proportions at all vertices. The second advance was to orient the pairwise sample comparisons along diagonals, or stripes, of the resulting distance matrix. This transform allows for substantial vectorization through single instruction multiple data (SIMD) operations, because groups of pairwise distances can be computed in a single instruction (fig. 1BC). Memory locality is also preserved by representing proportions and stripes as contiguous C-style arrays. The third advance was to allow independent execution of distance matrix stripes allowing considerable

task-level parallelism (either by threads or processes). Fourth, in a bifurcating tree, approximately 50% of the vertices are represented by the tips; we reasoned that collapsing the phylogeny slightly by disregarding UniFrac computations at the tips would yield a highly correlated result (e.g., similar to the minimal differences caused by moving from 99% to 97% OTUs) (fig. 1D). This heuristic is optional. Last, we represent the phylogeny using balanced parentheses [9], a succinct data structure that supports rapid tree reductions and traversals, and has minimal memory requirements.

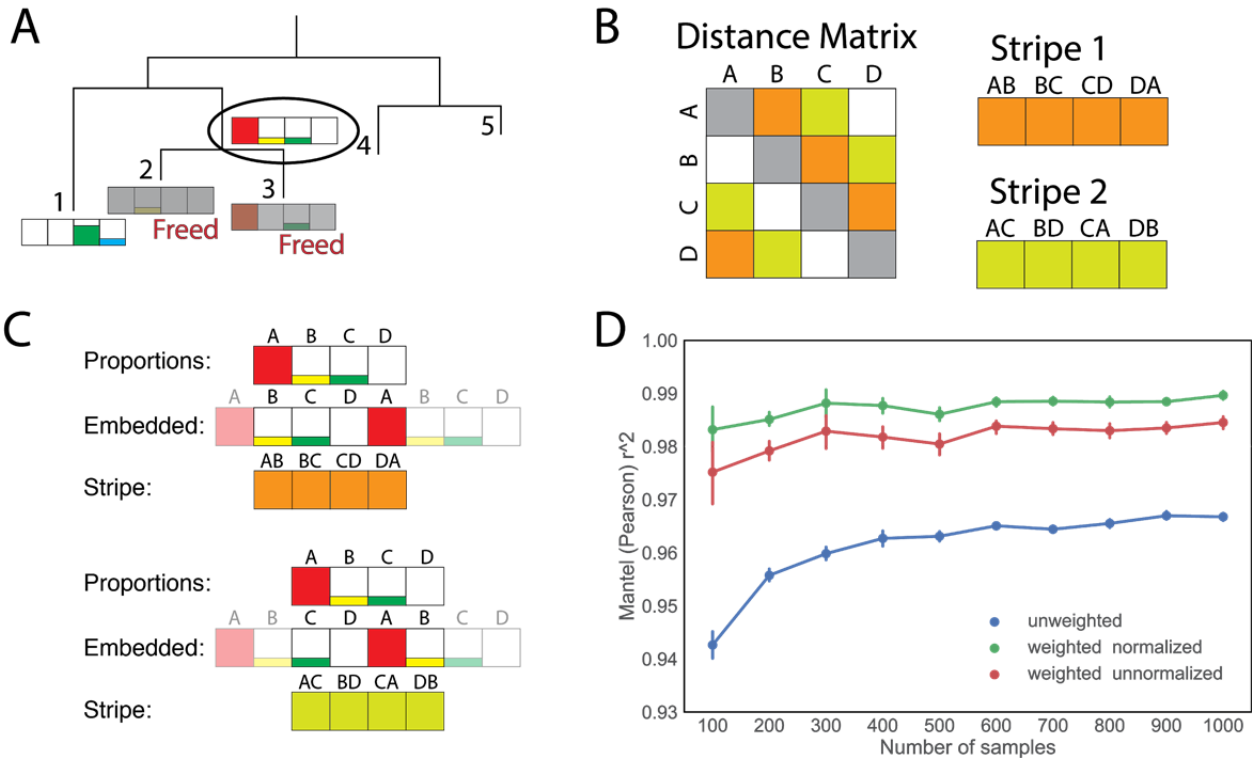

Figure 1. Algorithm highlights. (**A**) A depiction of the fourth vertex evaluated in a postorder traversal and the resulting sample proportions (circled). This vertex is the parent of tips “2” and “3”. The sample proportions for this vertex represent the aggregate (sum) of the sample proportions of the features which descend. The memory for features “2” and “3” is no longer needed, and can be freed. (**B**) A schematic of the two stripes in a four sample logical distance matrix; the labels above the stripes denote the pairwise comparison represented (e.g., “AB” indicates the distance between samples “A” and “B”). (**C**) A nodes sample proportions are embedded, duplicating the proportion information. This duplication allows the sample proportions to be slid along the embedded proportions, allowing comparison of linear blocks of memory for all pairwise combinations of samples. (**D**) Mantel tests (Pearson) between Strided State UniFrac in exact mode, which produces identical results to UniFrac vs. fast mode in which the UniFrac distances are not computed at the tips of the tree during traversal. Each data point represents 10 random subsets of the Earth Microbiome Project Deblur 90nt dataset, with the median R^2 value depicted. Error bars are 95% CI.

In order to benchmark this new algorithm, we randomly sampled the Earth Microbiome Project Deblur 90 nucleotide dataset at increasing numbers of samples with ten iterations at a given

number of samples. For each sOTU table, the EMP phylogeny was sheared to only the features contained to avoid benchmarking different phylogenetic data structures. For each table and tree pair, we computed unweighted UniFrac, weighted normalized UniFrac and weighted unnormalized UniFrac with Striped UniFrac, Fast UniFrac as used by QIIME1 [10] (the reference implementation as implemented in PyCogent [11]), Fast UniFrac in QIIME2 (a Cython'd implementation in scikit-bio, and Fast UniFrac in phyloseq (a independent R implementation)[12]. For each execution, runtime and space were tracked using GNU Time (fig. 2AB). To explore parallelism, we executed Striped UniFrac on the full EMP table (25,145 samples) using shared computational nodes with each process using two threads, and tracked the per process time and memory (fig. 2C) for walltime distributions, (fig. S1) for memory distributions.

We then obtained 120,790 Illumina 16S rRNA V4 samples processed by Deblur [13] from Qiita (Qiita study IDs, sample counts and study titles in table S1). The 5,522,523 fragments were inserted into Greengenes [14] 13_8 99% tree using SEPP [15,16]. Rarefaction to 500 sequences per sample reduced the total number of samples to 113,721. By distributing the computation over 256 processors, we computed unweighted UniFrac in under 48 hours walltime, using 7,977 CPU hours with a peak resident memory during the distributed computation at 1.3GB. We then performed Principal Coordinates Analysis using centered FSVD [17] on the resulting distances, and visualized them through EMPeror [18] (fig. 2D). This level of integration shows the dramatic range of diversity associated with the American Gut Project (in press), which includes samples from only a single host type (humans) suggesting a high degree of "unique" microbiomes remain to be found by deeper sampling in other host types.

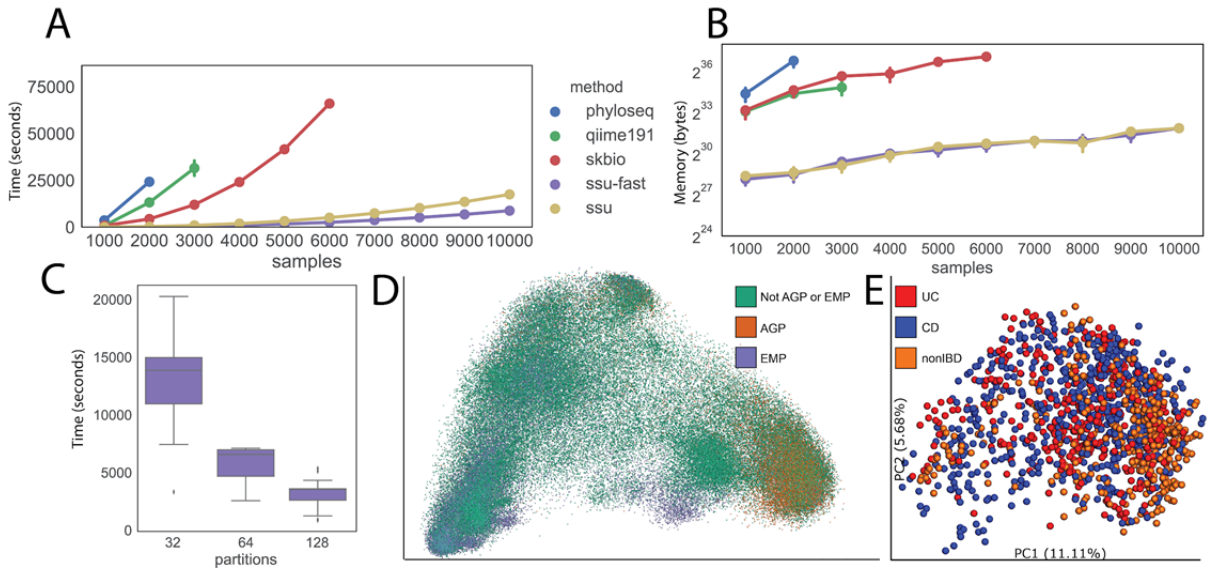

Figure 2. Space and time complexities. (**A-B**) Time and space comparisons between phyloseq, QIIME v1.9.1, scikit-bio, Striped UniFrac in exact mode, and Striped UniFrac in fast mode. Each datapoint represents 10 random subsets of the Earth Microbiome Project 90nt dataset at increasing numbers of samples. All methods were run single threaded on non-shared compute nodes which were not running other compute tasks. A job was killed if it exceeded 24 hours walltime or 256GB of memory (system max). (**C**) Walltime distributions of independent

processes operating on the full Earth Microbiome Project dataset (over 25,000 samples) executing on shared compute nodes. An individual partition represents a single independent process, and each process was run with two threads; 32 partitions indicates 32 processes using two threads each. A higher partition count means each individual process is doing less work. Given sufficient available resources, the maximum for a distribution represents a near upper bound on walltime. (**D**) An ordination plot of unweighted UniFrac distances over 113,721 samples sourced from Qiita. (**E**) Unweighted UniFrac applied to the metagenomic data from the integrated Human Microbiome Project data using the taxonomy as the tree. UC is Ulcerative Colitis, CD is Crohn's Disease and nonIBD is non-Inflammatory Bowel Disease.

For extensibility, we implemented the algorithm in C++ under the BSD open source license, and provide a shared C library with examples of interfacing to it using C and R, and a comprehensive Python using the C-API (https://github.com/biocore/unifrac). In addition, we implemented kernels to support other variants of UniFrac such as Generalized UniFrac [19] and Variance Adjusted UniFrac [20]. To facilitate broader adoption, a generalized x86-64 build of library is now part of QIIME2's "q2-diversity" plugin, the same plugin used by Qiita.

The design of Striped UniFrac allows UniFrac to scale well into the future, with demonstrated operational capability of >100,000 samples. Its parallel model and empirical scaling suggests application to datasets an order of magnitude larger than the EMP, and already benefits users operating on studies of "merely" thousands of samples such as the Integrated Human Microbiome Project dataset, in which UniFrac can be computed in 0.5 seconds on a laptop, allowing interactive analyses rather than batch-mode (fig. 2E). These reductions are critical for methods assay variability introduced by rarefaction, such as jackknifed beta diversity. Similarly, these reductions continue the democratization of analysis, bringing projects at the scale of the Earth Microbiome Project from a supercomputer to your laptop.

Methods

Postorder traversal memory reduction

At initialization, a stack is created to store sample proportion vectors of type double and length *N* where *N* is the number of samples. The vectors in the stack are used to represent sample proportions across the tree. The stack is used to avoid reallocation of sample proportion vectors over the tree; allocated memory is reused after a vertex has been evaluated with more memory allocated to the stack only as needed. The stack is combined with a hash table indexed by a unique node identifier so specific sample proportions can be retrieved if presently stored.

Over a postorder traversal of the input tree, if the vertex examined does not have children (i.e., is a leaf), then a proportion vector is popped from the stack (allocating memory if the stack is empty), and the proportions in this vector are set to the sample proportions for the observed feature from the input BIOM table. An entry is then added into a hash table mapping the index of the node to the address of the vector. If the vertex evaluated instead has children (i.e., is not a leaf), then a vector is popped from the stack (allocating memory if the stack is empty), the sample proportions for each child of the vertex are obtained from the hash table, and the sample proportions of the children are summed to create the proportion vector for the vertex

under evaluation. The sample proportions of the children are then pushed on to the onto the stack, and the hash table entry for their vertex identifiers is erased. Because the traversal is performed in postorder, the children of a vertex are always evaluated first, which ensures the sample proportions of the children are present in the hash table.

Stripe aggregation

A matrix of size K x N, where K is the number of stripes and N is the number of samples is allocated. The number of stripes for a full distance matrix (N + 1) / 2, resulting in an allocation of ((N + 1) / 2) * N) elements. For an odd number of samples, the number of elements in this matrix is exactly the number of elements in the upper triangle of the distance matrix. For an even number of samples, (N / 2) space is replicated. The elements of these stripes span both the upper and lower triangles of the logical distance matrix.

For each vertex, the sample proportions P for the vertex are embedded into a vector E of length 2N, such that the first N elements are a copy of the sample proportions, and the N to 2N elements are a copy of the sample proportions. This embedding allows for bulk pairwise comparisons between samples by allowing the execution of a distance kernel D (e.g., unweighted UniFrac) into a single stripe k with $D(P, E[k:k+N])$. This approach results in the comparison of two linear and C contiguous blocks of memory of length N, storing into a linear and C contiguous block of memory of length N.

Distance kernels and parallelism

Each individual distance metric (e.g., unweighted UniFrac, Generalized UniFrac, etc) is expressed as a compute kernel. These compute kernels operate per vertex during the tree traversal, compute a metric at every node within the tree, adding the computed distances into the stripes. Profiling suggests the vast bulk (>99%) of the computational time is expended in these kernels.

The compute of any diagonal (stripe) in the resulting distance matrix is independent of any other diagonal. This property allows the computation to be expressed as a map-reduce problem, where we *map* stripe sets to processing engines, and *reduce* by consolidating the stripes into a logical distance matrix or condensed form matrix.

Data sets

The full Deblur 90nt table rarefied at 1000 sequences per sample, and corresponding phylogenetic tree were obtained from the Earth Microbiome Project. Sets of samples were randomly pulled from this table at from 1000 to 10000 samples in steps of 1000. At each sample size, 10 iterations were performed producing 10 tables. For a given set of samples, the subset of the tree vertices and edges ancestral to the features in a sample set were retained, all other vertices and edges were pruned out. Any single descendant nodes were collapsed, aggregating the branch length toward the root.

The iHMP metagenomic data were obtained from the iHMP Qiita portal (https://ihmp.ucsd.edu), study 10001. Preparation ID 5 of the metagenomic data were downloaded as a BIOM table.

Only observations at the species level were retained. A tree was constructed using the lineage information embedded in the IDs. Unweighted UniFrac was computed on this table using the Python API for Striped UniFrac using 4 threads.

Pseudocode representation of the algorithm with unweighted UniFrac

```
function unweighted(embedded_props, stripes, stripe_totals)
    n_samples = number_of_columns(stripes)
    for stripe_index in stripes
        start = stripe_index
        end = start + n_samples
        stripe_props = embedded_props[start:end]

        unique = embedded_props[:n_samples] XOR stripe_props
        total = embedded_props[:n_samples] OR stripe_props

        # inplace operation
        stripes[stripe_index] += unique
        stripe_totals[stripe_index] += total


function initialize_child_proportions()
    stack_of_proportions = empty()
    hashmap_of_proportions = empty()
    return (stack_of_proportions, hashmap_of_proportions)


function merge_child_proportions(child_props, tree, node)
    node_props = empty()
    for child in children(tree, node)
        child_prop = child_props.hashmap_of_props.pop(child)
        node_props += child_prop
        child_props.stack_of_props.push(child_prop)
    return node_props


function associate_node_props(child_props, node, props)
    child_props.hashmap_of_props[node] = props


function get_prop_vector(child_props)
    if child_props.stack_of_props.empty()
        return empty_vector_of_length_number_of_samples
    else
        return child_props.stack_of_props.pop()
```

```
function embed_props(vector)
    embedded = zeros(length(vector) * 2)
    embedded[:length(vector)] = vector
    embedded[length(vector):] = vector
    return embedded


function get_and_set_leaf_vector(table, node, props)
    for index, value in getleaf(table, node)
        props[index] = value


function deconvolute(stripes)
    n_samples = length(stripes[0])
    matrix = zeros(n_samples, n_samples)

    for index, stripe in enumerate(stripes)
        k = 0
        row = 0
        col = index + 1
        while row < n_samples
            if(col < n_samples):
                matrix[row, col] = stripe[k]
                matrix[col, row] = stripe[k]
            else
                matrix[col % n_samples][row] = stripe[k]
                matrix[row][col % n_samples] = stripe[k]
            row = row + 1
            col = col + 1
            k = k + 1
    return matrix


function unifrac(table, tree, kernel)
    n_samples = number_of_samples(table)
    n_stripes = ceil((n_samples^2 - n_samples) / 2)
    stripes = zeros(n_stripes, n_samples)
    stripe_totalss = zeros(n_stripes, n_samples)
    child_props = initialize_child_props()

    for node in postorder(tree)
        if isleaf(node)
            props = get_prop_vector(child_props)
            get_and_set_leaf_vector(table, node, props)
```

```
            associate_node_props(child_props, node, props)
        else
            props = merge_child_props(child_props, tree, node)

        embedded_props = embed_props(props)
        kernel(embedded_props, length(node), stripes, stripe_totals)

    if kernel is normalized
        for stripe_index in stripes:
            unique = striped[stripe_index]
            total = stripe_totals[stripe_index]
            stripes[stripe_index] = unique / total

    return deconvolute(stripes)
```


1. Lozupone, C. & Knight, R. UniFrac: a new phylogenetic method for comparing microbial communities. *Appl. Environ. Microbiol.* **71,** 8228–8235 (2005).
2. Smith, M. I. *et al.* Gut microbiomes of Malawian twin pairs discordant for kwashiorkor. *Science* **339,** 548–554 (2013).
3. Yatsunenko, T. *et al.* Human gut microbiome viewed across age and geography. *Nature* **486,** 222–227 (2012).
4. Ley, R. E., Lozupone, C. A., Hamady, M., Knight, R. & Gordon, J. I. Worlds within worlds: evolution of the vertebrate gut microbiota. *Nat. Rev. Microbiol.* **6,** 776–788 (2008).
5. Kuczynski, J. *et al.* Microbial community resemblance methods differ in their ability to detect biologically relevant patterns. *Nat. Methods* **7,** 813–819 (2010).
6. Thompson, L. R. *et al.* A communal catalogue reveals Earth's multiscale microbial diversity. *Nature* **551,** 457–463 (2017).
7. Hamady, M., Lozupone, C. & Knight, R. Fast UniFrac: facilitating high-throughput phylogenetic analyses of microbial communities including analysis of pyrosequencing and PhyloChip data. *ISME J.* **4,** 17–27 (2010).
8. McClelland, J. & Koslicki, D. EMDUniFrac: exact linear time computation of the UniFrac metric and identification of differentially abundant organisms. *J. Math. Biol.* (2018).

doi:10.1007/s00285-018-1235-9

9. Cordova, J. & Navarro, G. Simple and efficient fully-functional succinct trees. *Theor. Comput. Sci.* **656,** 135–145 (2016).

10. Caporaso, J. G. *et al.* QIIME allows analysis of high-throughput community sequencing data. *Nat. Methods* **7,** 335–336 (2010).

11. Knight, R. *et al.* PyCogent: a toolkit for making sense from sequence. *Genome Biol.* **8,** R171 (2007).

12. McMurdie, P. J. & Holmes, S. phyloseq: an R package for reproducible interactive analysis and graphics of microbiome census data. *PLoS One* **8,** e61217 (2013).

13. Amir, A. *et al.* Deblur Rapidly Resolves Single-Nucleotide Community Sequence Patterns. *mSystems* **2,** (2017).

14. McDonald, D. *et al.* An improved Greengenes taxonomy with explicit ranks for ecological and evolutionary analyses of bacteria and archaea. *ISME J.* **6,** 610–618 (2012).

15. Mirarab, S., Nguyen, N. & Warnow, T. SEPP: SATé-enabled phylogenetic placement. *Pac. Symp. Biocomput.* 247–258 (2012).

16. Janssen, S. *et al.* Phylogenetic Placement of Exact Amplicon Sequences Improves Associations with Clinical Information. *mSystems* **3,** e00021–18 (2018).

17. Halko, N., Martinsson, P., Shkolnisky, Y. & Tygert, M. An Algorithm for the Principal Component Analysis of Large Data Sets. *SIAM J. Sci. Comput.* **33,** 2580–2594 (2011).

18. Vázquez-Baeza, Y., Pirrung, M., Gonzalez, A. & Knight, R. EMPeror: a tool for visualizing high-throughput microbial community data. *Gigascience* **2,** 16 (2013).

19. Chen, J. *et al.* Associating microbiome composition with environmental covariates using generalized UniFrac distances. *Bioinformatics* **28,** 2106–2113 (2012).

20. Chang, Q., Luan, Y. & Sun, F. Variance adjusted weighted UniFrac: a powerful beta diversity measure for comparing communities based on phylogeny. *BMC Bioinformatics* **12,** 118 (2011).

Consider a rooted tree T with root ρ on n nodes as an n dimensional vector spaces over the real numbers. Identify the subtrees of T with the nodes of T, that is subtree i is the subtree which does not contain ρ formed by choosing node i as a root. The subtree corresponding to ρ is T. Choose a basis $\{v_i|\ 1 \leq i \leq n\}$ for T such that v_i is the indicator function for subtree i, that is $v_i(j) = 1$ for those nodes j in subtree i and zero otherwise. Label the $n-1$ edges of T such that edge e_i is the unique edge adjacent to node i on a path from node i to the root ρ.

Let W be the $n \times n$ matrix whose rows correspond to the basis vectors v_i scaled by the corresponding edge weight $l(e_i)$. Consider probability distributions P and Q on T as column vectors, ordered such that entry i corresponds to the root of subtree i.

Then

$$\text{UniFrac}(P, Q) = \|W(P - Q)\|_{L_1}$$

.

Thius follows as we note that

$$\|W(P - Q)\|_{L_1} = \sum_{i=1}^{n} \sum_{j=1}^{n} l(e_i) v_i(j) |P(v_j) - Q(v_j)|$$

This summation is the discrete integral over T of the distributions P and Q with respect to measure formed from the branch lengths of T. By [1] the value of this integral is equivalent to earth mover's distance between P and Q with respect to the tree T, and thus is equal to the UniFrac distance between P and Q.

References

[1] Evans SN, Matsen FA. *The phylogenetic Kantorovic Rubinstein metric for environmental sequence samples.* Journal of the Royal Statistical Society Series B, Statistical methodology. 2012