Robust 16S rRNA classification based on a compressed LCA index

Omar Y. Ahmed, Christina Boucher and Ben Langmead

Genome Res. published online August 25, 2025
Access the most recent version at doi:10.1101/gr.279846.124

Robust 16S rRNA Classification Based on a Compressed LCA Index

Omar Ahmed[1]
oahmed6@jhu.edu
ORCID: 0000-0002-9933-8508

Christina Boucher[2]
cboucher@cise.ufl.edu
ORCID: 0000-0001-9509-9725

Ben Langmead[1*]
langmea@cs.jhu.edu
ORCID: 0000-0003-2437-1976

[1]Department of Computer Science, Whiting School of Engineering,
Johns Hopkins University, Baltimore, MD, USA

[2]Department of Computer and Information Science and Engineering,
Herbert Wertheim College of Engineering,
University of Florida, Gainesville, FL, USA

[*]Corresponding author

Abstract

Taxonomic sequence classification is a computational problem central to the study of metagenomics and evolution. Advances in compressed indexing with the r-index enable full-text pattern matching against large sequence collections. But the data structures that link pattern sequences to their clades of origin still do not scale well to large collections. Previous work proposed the document array profiles, which use $\mathcal{O}(rd)$ words of space where r is the number of maximal-equal letter runs in the Burrows-Wheeler transform and d is the number of distinct genomes. The linear dependence on d is limiting, since real taxonomies can easily contain 10,000s of leaves or more. We propose a method called cliff compression that reduces this size by a large factor, over 250 times when indexing the SILVA 16S rRNA gene database. This method uses $\Theta(r \log d)$ words of space in expectation under a random model we propose here. We implemented these ideas in an open source tool called Cliffy that performs efficient taxonomic classification of sequencing reads with respect to a compressed taxonomic index. When applied to simulated 16S rRNA reads, Cliffy's read-level accuracy is higher than Kraken 2's by 11-18%. Clade abundances are also more accurately predicted by Cliffy compared to Kraken 2 and Bracken. Overall, Cliffy is a fast and space-economical extension to compressed full-text indexes, enabling them to perform fast and accurate taxonomic classification queries. Cliffy's accuracy underscores the advantages of full-text indexes, which offer a more precise solution compared to k-mer indexes designed for a specific k value.

Introduction

Compressed indexes are increasingly used for read alignment [Garrison et al., 2018, Sirén et al., 2021, Armstrong et al., 2020] and classification [Ahmed et al., 2021, Ahmed et al., 2023b] with respect to pangenomes and other collections. These methods exploit the repetitiveness inherent in the Burrows-Wheeler Transform (BWT) of a sequence collection by compressing the BWT into its maximal equal-letter runs [Mäkinen and Navarro, 2005]. Such approaches can straightforwardly determine if a query string is present or absent in the index [Mäkinen and Navarro, 2005], but a harder task is to find where the query string occurs. For this, Gagie et al. [Gagie et al., 2018] proposed a subsampling strategy for the suffix array (SA) along with efficient algorithms for "locate" queries, returning all offsets where the substring occurred. This subsampled suffix array fits in a space budget of $\mathcal{O}(r)$ words, where r is the number of BWT runs. Compressed indexes are relevant for taxonomic classification since they enable finding and locating substrings while also compressing away repetitive sequence content. In practice, solving the taxonomic classification problem does not require the full resolution of a locate query. Instead, indexes for taxonomic classification typically support broader queries that either (a) list which genomes the query string occurs in ("document listing"), or (b) find which portion of the taxonomic tree the substring likely originated from ("taxonomic classification").

We previously introduced a data structure for document listing called the *document array profiles* [Ahmed et al., 2023a]. This data structure requires $\mathcal{O}(rd)$-space, where d is the number of genomes (also called "documents"), and enables document listing in time proportional to the number of documents containing the query substring ($ndoc$). This was the first data structure to enable document listing queries in space proportional to r rather than n. However, the linear dependence on d makes that structure too large for typical taxonomic classification scenarios where d can be 10,000 or more. For example, the SILVA NR99 SSU reference database (510,508 sequences, 1.2 GB) spans 9,118 distinct genera, which would require a document array profiles structure larger than 2 TB.

Here, we develop new methods that drastically reduce the space usage. In so doing, we sacrifice the ability to perform full document listing, but we retain the ability to perform accurate taxonomic classification. Detecting and measuring alleles of the 16S rRNA gene allows for the study of bacterial communities, a common task since its introduction in the 1970s [Woese et al., 1975]. The gene contains both conserved and hypervariable regions, allowing for primer design (in conserved portions) [D'Amore et al., 2016, Tringe and Hugenholtz, 2008] and measurement of evolutionary distance (in variable portions) [Kim et al., 2014, Sun et al., 2009]. Projects like the Human Microbiome Project [Consortium, 2012], Earth Microbiome Project [Thompson et al., 2017] and MetaSUB [The MetaSUB Consortium, 2016] have used 16S sequencing to characterize the microbes present in the gut, skin, soil, urban environments, and water. In clinical settings, the 16S rRNA gene has been used, for example, to study spatial patterns of microbial infection in lungs of cystic fibrosis patients [Willner et al., 2012] as well as microbial communities in the saliva of Behçet's syndrome patients [Coit et al., 2016].

Existing software tools for 16S rRNA classification use databases such as Greengenes [DeSantis et al., 2006], RDP [Maidak et al., 1997], and SILVA [Quast et al., 2012] as the reference collection. In 2018, Almeida et al. [Almeida et al., 2018] performed a benchmarking study that showed the QIIME2 [Bolyen et al., 2019] was more accurate than other tools for quantifying reads at the genus and family levels, but was computationally very expensive, with time and memory usage 2 and 30 times higher respectively than other tools [Almeida et al., 2018]. More recently, Lu et al. [Lu and Salzberg, 2020] extended the popular shotgun metagenomic classifier Kraken 2 [Wood et al., 2019] to support 16S rRNA databases, showing that their method was up to 300 times faster than QIIME2 while achieving higher accuracy and being the only tool to provide per-read classifications.

Recent innovations in pangenomic indexing also address the scalability and efficiency challenges inherent in classification to large reference collections. Recent methods like Movi [Zakeri et al., 2024], Moni-align [Varki et al., 2025], Sampled tag arrays [Depuydt et al., 2025], and Chain Statistics [Brown et al., 2025] have advanced pangenomic indexing by improving efficiency and accuracy. Movi uses the move structure for cache-efficient backward searches, while Moni-align is a full-featured aligner based on the r-index optimized for mapping accuracy. Depuydt et al. [Depuydt et al., 2025] combine the move-structure with a sampled tag array to identify super-maximal exact matches (SMEMs) to use for classification. Chain Statistics improves classification resolution by modeling dependencies between hits but requires additional storage. In this work,

we introduce a method that allows for correct answers to LCA queries over repetitive collections such as 16S databases, which is a functionality not supported by any of those tools.

Our methods are implemented in a new tool called *Cliffy* that performs per-read 16S rRNA gene classifications using a compressed full-text index. Cliffy uses a novel compression scheme to reduce the size of the document array profiles [Ahmed et al., 2023a] by two orders of magnitude to make it practical for 16S rRNA classification. In addition, Cliffy incorporates strategies such minimizer digestion and a backward search lookup table to speed up its queries. Using a similar approach to the benchmarking study by Almeida et al. [Almeida et al., 2018], we generate realistic read datasets from different biomes and show that Cliffy consistently outperforms Kraken 2 in terms of per-read accuracy as well as genus abundance profiling. Given Cliffy's accuracy and its capability to find matches of any length, along with the anticipated growth of reference databases, we propose that Cliffy offers a more accurate and scalable alternative to k-mer based indexes designed for a single value of k.

Methods

Preliminaries

We define a string $T[1..n]$ with length n as a concatenation of characters $T[1] \cdots T[n]$ drawn from alphabet Σ of size σ. The *empty* string, denoted ε, is the only string with length 0. We assume T ends with $\$ \notin \Sigma$, a special symbol lexicographically smaller than all the symbols in Σ.

Given two integers $1 \leq i \leq j \leq n$, we denote $T[i..j] = T[i] \cdots T[j]$ the *substring* of T spanning positions i to j. Otherwise if $i > j$, we define $T[i..j] = \varepsilon$. We refer to $T[i..n]$ as the i-th *suffix* of T and $T[1..i]$ as the i-th *prefix* of T. Given two strings $T[1..n]$ and $S[1..m]$, $\mathsf{lcp}(S, T)$ denotes the length of the longest common prefix (LCP) of T and S.

Suffix array, inverse suffix array, and longest common prefix array. Given a string T, the *suffix array* [Manber and Myers, 1993] $\mathsf{SA}_T[1..n]$ is the permutation of $\{1, \ldots, n\}$ that lexicographically sorts the suffixes of T. The *inverse suffix array* $\mathsf{ISA}_T[1..n]$ is the inverse permutation of $\mathsf{SA}_T[1..n]$, i.e., for all $i = 1, \ldots, n$ $\mathsf{ISA}_T[\mathsf{SA}_T[i]] = i$. The *longest common prefix* array $\mathsf{LCP}_T[1..n]$ stores the length of the LCP between lexicographically consecutive suffixes of T, formally, $\mathsf{LCP}[1] = 0$ and for all $i = 2, \ldots, n$ $\mathsf{LCP}[i] = \mathsf{lcp}(T[\mathsf{SA}_T[i-1]..n], T[\mathsf{SA}_S[i]..n])$.

Burrows-Wheeler transform. Given a string T, the *Burrows-Wheeler transform* [Burrows and Wheeler, 1994] $\mathsf{BWT}_T[1..n]$ is a reversible permutation of T defined as the last column of the matrix of the lexicographically sorted rotations of T. When T is terminated by \$ we can define for all $i = 1, \ldots, n$, $\mathsf{BWT}_T[i] = T[\mathsf{SA}_T[i] - 1]$ where $T[0] = T[n]$. LF*-mapping* is the the permutation of $\{1, \ldots, n\}$ that maps every character in the BWT_T to its predecessor in text order, formally $\mathsf{LF}[i] = \mathsf{ISA}_T[\mathsf{SA}_T[i] - 1]$ for i where $\mathsf{SA}_T \neq 1$ and $\mathsf{LF}[i] = \mathsf{ISA}_T[n]$ when $\mathsf{SA}_T = 1$. We define r as the number of maximal equal-letter runs of BWT_T. When clear from the context we refer to SA_T, ISA_T, LCP_T, and BWT_T as SA, ISA, LCP, and BWT respectively.

r-index. The r-index [Gagie et al., 2020] is a full-text index based on the run-length encoded BWT and the SA entries sampled at run boundaries. Given a text $T[1..n]$ and a pattern $P[1..m]$ the r-index can locate all occurrences of P in T in $\mathcal{O}(m \log \log_w(\sigma + n/r) + occs \log \log_w(n/r))$ time and $\mathcal{O}(r)$ words of space, where $occs$ is the number of occurrences of P in T.

Document array and document array profiles. We denote a collection of documents (i.e. strings) with $\mathcal{D} = \{T_1, \ldots, T_d\}$ and we denote a concatenation of documents with $\mathcal{T}[1..n] = T_1 \cdots T_d$. The *document array* [Muthukrishnan, 2002] $\mathsf{DA}[1..n]$ stores for each position i the document index of $\mathcal{T}[\mathsf{SA}_\mathcal{T}[i]..n]$. The document array profiles [Ahmed et al., 2023a] is a data structure that enables the r-index to perform *document listing* queries.

Problem 1. *Document listing: Given a collection $\mathcal{D} = \{T_1, \ldots, T_d\}$ and a pattern P, return the set of documents $\mathcal{L} \subseteq \mathcal{D}$ where P occurs.*

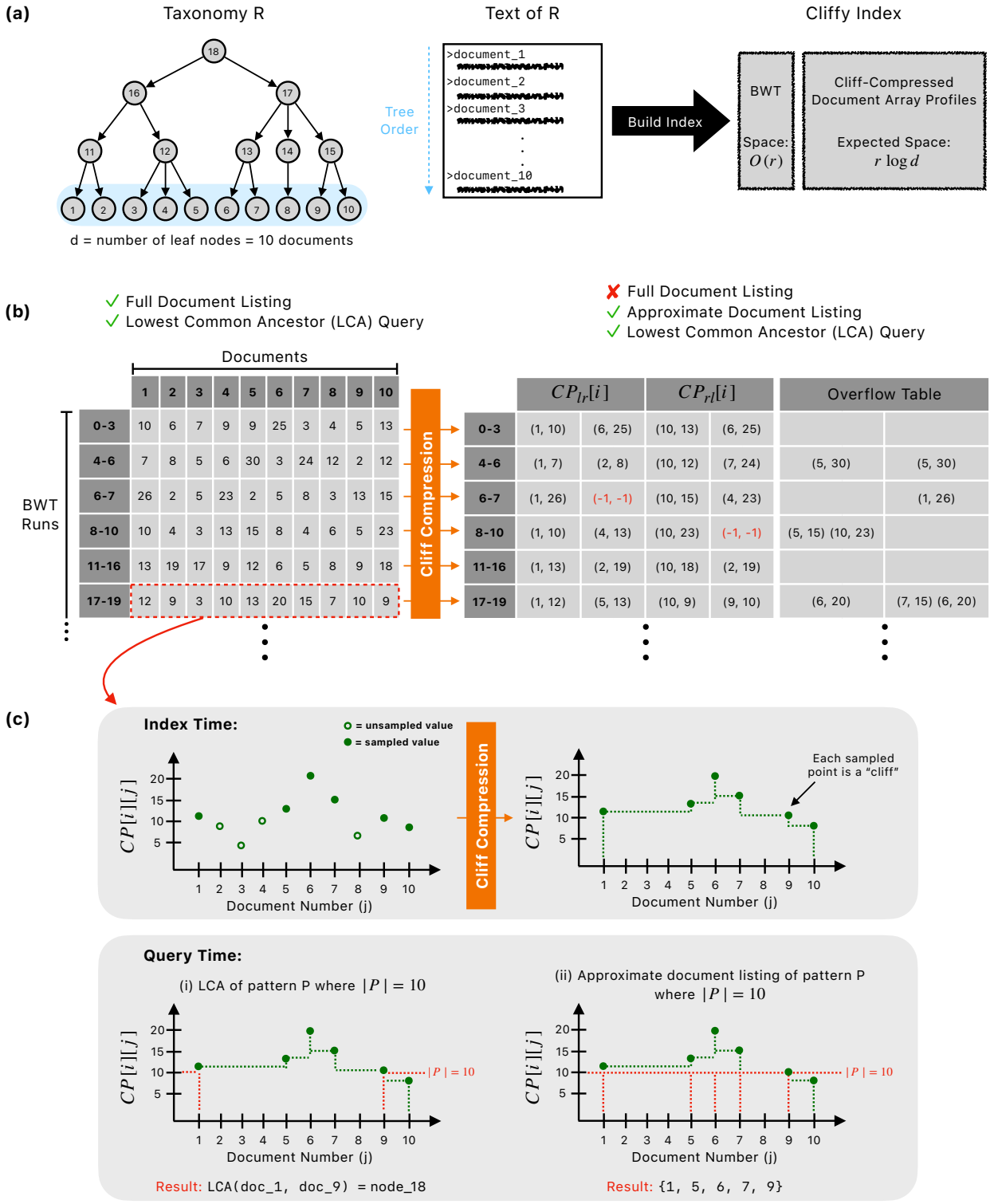


Figure 1: Overview of Cliffy's taxonomy indexing and compression approach. (a) Shows an example taxonomy where $d = 10$ and an example tree order that guides the generation of the text of the taxonomy prior to building an index. (b) Visualizes the difference between the document array profiles and the cliff-compressed document array profiles. The cliff-compressed document array profiles consists of a "main table" with a fixed number of columns as well as an "overflow table" that stores any pairs that could not be included in the main table. (c) Using a profile (i.e. row of document array profiles) as an example, visualizes cliff-compression at the index building step, followed by how each profile is used at query time for either the lowest common ancestor query or approximate document listing.

Given an r-index for the concatenated text $\mathcal{T}[1..n]$, the document array profiles supports document listing for a pattern $P[1..m]$ in $\mathcal{O}(m \log \log_w(\sigma + n/r) + ndoc)$ time and $\mathcal{O}(rd)$-space, where $ndoc$ is the number of documents containing the pattern P.

Definition 1. *(From [Ahmed et al., 2023a]) Document array profiles: For all positions* $1 \leq \mathsf{LF}(i) \leq n$ *where* i *is a run head or tail in the* BWT *of* T*, the document array profile* $P_{DA}[i][1..d]$ *stores for each position* $j = 1, ..., d$ *the length of the longest common prefix between* $\mathcal{T}[SA[i]..n]$ *and all suffixes of document* T_j*.*

$$P_{\mathsf{DA}}[i][j] = \max\left\{\mathsf{lcp}(\mathcal{T}[\mathsf{SA}[i]..n], \mathcal{T}[\mathsf{SA}[k]..n]) \mid 1 \leq k \leq n \text{ and } \mathsf{DA}[k] = j\right\}$$

Document listing query using document array profiles In our previous work [Ahmed et al., 2023a], we demonstrated how to compute a document listing for a pattern P using the document array profiles. The algorithm requires that we store $P_{DA}[LF(i)]$ for all i corresponding to a BWT run head or tail, following the "toehold lemma" strategy [Policriti and Prezza, 2016], i.e., the stored rows are sufficient to recover the document listing for a query pattern P.

Given a query pattern P, we perform backward search for P and store a pointer to a "sampled" row $P_{DA}[i]$ of the document array profiles where $\mathsf{BWT}[i] = P[j]$ where j is the last character of P that was searched. At each step of the backward search, we either sample a new row or simply increment a counter variable k by 1. If the backward search range $\mathsf{BWT}[s..e]$ spans a BWT run head or tail, we sample a new row $P_{DA}[i]$ where $s \leq i < e$ and reset k to 0. Otherwise, if the backward search range is contained within a BWT run, we simply increment k by 1.

In order to generate the document listing, we add k to each value in the "sampled" row $P_{DA}[i][1..d]$ and return all values of j where $P_{DA}[i][j] \geq |P|$. Assuming the LCP values in $P_{DA}[i][1..d]$ are stored in sorted order, we only need to scan the first $ndoc+1$ values to generate the document listing. As previously mentioned, the document array profiles supports document listing for a pattern $P[1..m]$ in $\mathcal{O}(m \log \log_w(\sigma+n/r)+ndoc)$ time and $\mathcal{O}(rd)$-space. More details on the proof for these bounds can be found in [Ahmed et al., 2023a].

Taxonomic classification with the pattern lowest common ancestor query

We aim to classify a sequence according to where it likely originated from in the tree of life, represented by a *taxonomy*. Let a taxonomy R be a rooted multiway tree with d leaves. Given a taxonomy R, we define the collection of documents of the taxonomy, denoted by $\{R_1, ..., R_d\}$ as the collection of reference sequences of the taxonomic clades at the leaves of the taxonomy (Figure 1a). Leaves of the taxonomy R map in a one-to-one fashion with the documents. Internal nodes represent common ancestors, and do not have directly associated strings; rather, they are implicitly associated with the union of the documents in the leaves in their subtree. Since a one-to-one mapping exists between leaves and documents, we often refer to these interchangeably.

Definition 2. *The occurrences of a string* P *in taxonomy* R *is the set of leaves* $\mathcal{F}$ *where* P *occurs as a substring of the associated document.*

We summarize where a string P originated using a *pattern lowest common ancestor* query:

Problem 2. *Pattern lowest common ancestor (PLCA): Given a taxonomy* R *and string* P*, return the deepest node* X *such that all of* P*'s occurrences are in leaves of* X*'s subtree.*

In order to process a set of occurrences of a pattern P in a taxonomy, we define the *lowest common ancestor* query:

Definition 3. *Lowest common ancestor (LCA): Given a taxonomy* R *and set of nodes* $\mathcal{F}$ *from* R*, the LCA of* $\mathcal{F}$ *is the deepest node* X *in* R *where all nodes in* $\mathcal{F}$ *are present in* X*'s subtree.*

PLCA can be solved by first identifying all occurrences of a pattern P in the leaves, then computing the LCA of those leaves. However, a simpler approach becomes possible if we first consider the leaf nodes to be in a particular order [Gagie et al., 2022]:

Definition 4. *Tree order: A tree order of the* d *leaves of taxonomy* R *is any ordering such that all leaves descended from any internal node are consecutive in the order.*

Alternately, we could say that the tree order ensures that descendants of sibling nodes in the tree cannot be interleaved in the order. From here on, we assume that the concatenated text $\mathcal{T}[1..n]$ respects tree order for the taxonomy R. We also assume that the document ids associated with the leaves follow this order, i.e., the leftmost (earliest) leaf is document 1, the next is document 2, etc.

Lemma 5. *(From [Gagie et al., 2022]) Assuming tree order, the PLCA query can be answered by computing the LCA of only the leftmost (earliest) and rightmost (latest) leaf nodes containing P in the order.*

Proof. The proof is by contradiction. We assume a pattern P has occurrences $\mathcal{F} = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ in taxonomy R. Further, we assume the occurrences are sorted by document ids, such that $\mathcal{F}_1$ is an occurrence in the leftmost of the documents that P occurs in and $\mathcal{F}_n$ is an occurrence in the rightmost document that P occurs in. We let $\text{LCA}(\mathcal{F})$ denote the LCA of the full set of occurrences, and $\text{LCA}(\mathcal{F}_1, \mathcal{F}_n)$ denote the LCA of the leftmost and rightmost occurrences.

We presume $\text{LCA}(\mathcal{F}) \neq \text{LCA}(\mathcal{F}_1, \mathcal{F}_n)$. Then it follows that there is a node $\mathcal{X} \subseteq \mathcal{F} \setminus \{\mathcal{F}_1, \mathcal{F}_n\}$ that occurs either in a document prior to the one matched in occurrence $\mathcal{F}_1$, or in a document after the one matched in occurrence $\mathcal{F}_n$. This implies the ordering allows for interleaving of the descendants of sibling nodes, contradicting the assumption of tree ordering. □

Lemma 6. *Given an r-index for a tree-ordered text of a taxonomy $\mathcal{T}(R) = R_1 \cdots R_d$ extended with the document array profiles P_{DA}, we can compute the PLCA of the pattern $P[1..m]$ in $\mathcal{O}(m \log \log_w(\sigma + n/r) + ndoc)$-time and $\mathcal{O}(rd)$-words of space.*

Proof. As summarized in the preliminaries, we previously showed how to compute the document listing for any length substring in $\mathcal{O}(m \log \log_w(\sigma + n/r) + ndoc)$-time by combining backward search with a scan of a "sampled" row of the document array profiles [Ahmed et al., 2023a]. Since we have a tree order, the PLCA for the pattern P can be computed as the LCA of the leftmost and rightmost leaves for which P occurs in their associated document. The LCA of these two leaves is then computed with an auxiliary LCA data structure requiring $2d + o(d)$ bits of space with $\mathcal{O}(1)$ query time [Fischer, 2010, Navarro, 2016]. These additional operations are both constant time, $\mathcal{O}(1)$, therefore, total time complexity is $\mathcal{O}(m \log \log_w(\sigma + n/r) + d)$. □

Shrinking the document array profiles with cliff compression. In the previous section, we proved that we can compute the PLCA for a given pattern using the document array profiles, however, it requires $\mathcal{O}(rd)$-space. We note that in practical settings, d can reach tens of thousands of species (documents). We propose a strategy that we refer to as *cliff compression*, which has an average space usage of $\Theta(r \log d)$ while still allowing for the exact computation of PLCAs. The insight is that to determine the leftmost document containing an ℓ-length match, it is sufficient to identify the smallest j such that $P_{DA}[i][j] \geq \ell$, where i represents the current BWT offset. Similarly, when we ask which is the rightmost document containing an ℓ-length match, it is sufficient to know the largest j such that $P_{DA}[i][j] \geq \ell$. As a result, we only need to store the maxima-so-far, i.e., $\max(P_{DA}[i][1..j])$ and $\max(P_{DA}[i][j..d])$ for all j when they first occur. Though we must store these for both the left-to-right and right-to-left maxima-so-far, the number of instances m where the maxima-so-far changes and we need to store an additional value is small in practice ($m \ll d$), making them significantly smaller to store compared to the full document array profile row $P_{DA}[i][1..d]$.

For added intuition, we visualize the profile $P_{DA}[i][1..d]$ (Figure 1c) with the maxima-so-far step functions drawn as dotted lines. The leftmost and rightmost documents containing a pattern P corresponds to the j values where a height-$|P|$ horizontal line crosses the maxima-so-far function. The LCA of the documents at the crossings is the PLCA. Longer matches correspond to higher horizontal lines, which tend to cross a narrower portion of the maxima-so-far function and so leads to deeper PLCAs.

Definition 7. *Cliff compression: Given the full document array profiles P_{DA}, we cliff-compress each row by first scanning $P_{DA}[i][1..d]$ from left to right and storing all $(j, P_{DA}[i][j])$ pairs where $j = 1$ or $P_{DA}[i][j] > \max(P_{DA}[i][1..j-1])$. We store these in an array denoted as $CP_{\ell r}[i]$, in ascending order by j. We also scan $P_{DA}[i][1..d]$ from right to left and store all $(j, P_{DA}[i][j])$ pairs, where $j = d$ or $P_{DA}[i][j] > \max(P_{DA}[i][j+1...d])$. We collect these in an array called $CP_{r\ell}[i]$, in descending order by j. $CP_{\ell r}$ and $CP_{r\ell}$ together constitute the cliff-compressed document array profiles.*

We can compute PLCAs using only the cliff-compressed document array profiles.

Theorem 8. *Given the r-index built over the tree-ordered text of a taxonomy $\mathcal{T}(R) = R_1 \cdots R_d$, extended with the cliff-compressed document array profiles $CP_{\ell r}$ and $CP_{r\ell}$, we can compute the PLCA for a pattern P in $\mathcal{O}(m \log \log_w(\sigma + n/r) + d)$ time in $\mathcal{O}(rd)$ words of space.*

Proof. We adapt the query algorithm given in the preliminaries to work with cliff-compressed document array profiles. Instead of storing $P_{DA}[LF[i]]$ at each head or tail, we store the pair $(CP_{\ell r}[LF[i]], CP_{r\ell}[LF[i]])$. The space usage remains $\mathcal{O}(rd)$ in the worst case. During backward search, when the range includes a run head or tail, we retrieve $(CP_{\ell r}[LF[i]], CP_{r\ell}[LF[i]])$ for i corresponding to the (contained) head or tail. When the range falls entirely within a run, we obtain our new sample by incrementing each $P_{DA}[i][\cdot]$ value (i.e. LCP) stored in the $CP_{r\ell}[i]$ and $CP_{\ell r}[i]$ lists from the previous step by 1.

After backward search, we find the leftmost and rightmost documents containing P. To find the leftmost, we scan the pairs of $(j, P_{DA}[i][j]) \in CP_{\ell r}[i]$, retrieving the first value j where $P_{DA}[i][j] \geq |P|$. By definition of $CP_{\ell r}$, this is the leftmost document containing P. Next, we scan the pairs $(j, P_{DA}[i][j]) \in CP_{r\ell}[i]$ and retrieve the first value j such that $P_{DA}[i][j] \geq |P|$ (recalling that elements of $CP_{r\ell}[i]$ are in descending order by j). By definition of $CP_{r\ell}$, this is the rightmost document containing P. These scans each require $\mathcal{O}(d)$-time.

Finally, we query the LCA data structure with leftmost and rightmost documents in $\mathcal{O}(1)$-time [Fischer, 2010, Navarro, 2016] to complete the PLCA query. Thus, we can compute the PLCA for a pattern P using the cliff compressed version of P_{DA} in $\mathcal{O}(m \log \log_w(\sigma + n/r) + d)$-time. □

Random model for cliff compression ratio Cliff compression does not improve worst-case space usage compared to the original document array profiles—both require $\mathcal{O}(rd)$ words of space, with the worst-case for cliff compression of $CP_{\ell r}[i]$ (resp. $CP_{r\ell}[i]$) being the case where the LCP values in $P_{DA}[i]$ are strictly increasing (resp. decreasing). However, we now show that the average-case space complexity when applying cliff compression to the document array profiles is significantly smaller. To accomplish this, we begin by proposing a random model that leads to the *average* compression ratio of cliff compression, and then we will describe the links between the random model and the cliff-compression scenario.

We model the values in a row i of the document array profiles ($P_{DA}[i]$) as a random permutation of the d integers $0, 1, \ldots, d-1$. We then consider the "minimum-so-far" computed over the permutation, by computing the minimum over all non-empty prefixes of the permutation. We model the number of values we must store in the $CP_{\ell r}[i]$ or $CP_{r\ell}[i]$ list as the number of times that the the minimum-so-far sequence decreases, plus one. Adding one accounts for the fact that we must always store at least one pair in the array. Note that once the minimum-so-far reaches 0 it cannot decrease further.

In practice, the document array profiles deviate from this model. Most importantly: (1) the documents in the taxonomy are related to each other, violating any assumptions of uniform probability of different LCP values, and (2) the values in $P_{DA}[i]$ are not all distinct as in the simple model. Nonetheless, we found that the model was effective at predicting the number of pairs in the cliff-compressed document array profiles (Table 1).

Given a random permutation of the n integers $0, 1, \ldots, n-1$, we define the random variable S_n as the number of minima-so-far decreases before reaching 0. The base case is $S_0 = 0$, i.e., if the leftmost value in the permutation is 0 then the minimum-so-far is uniformly 0 with no decreases. Another easy case is $S_1 = 1$, i.e., if the leftmost value is 1 then exactly one decrease (to 0) will occur. For S_2, we must consider two cases: we reach 1 before reaching 0, or we reach 0 before reaching 1. In the first case, $S_2 = S_1 + 1 = 2$. In the second, $S_2 = S_0 + 1 = 1$. The cases are equally likely since the permutation is random. Therefore, we have $\mathbf{E}[S_2] = 1 + 1/2 \cdot (\mathbf{E}[S_1] + \mathbf{E}[S_0]) = 3/2$. In general, we have:

$$\mathbf{E}[S_n] = 1 + \frac{1}{n} \sum_{j=0}^{n-1} \mathbf{E}[S_j] \tag{1}$$

A closed form of the above formula equals the n^{th} harmonic number:

$$1 + \frac{1}{n} \sum_{j=0}^{n-1} \mathbf{E}[S_j] = \sum_{j=1}^{n} \frac{1}{j} = H_n \tag{2}$$

See Supplemental Section 1 for the proof of this formula. Finally, we note that H_n grows with $\Theta(\log n)$, as is proved in Supplemental Section 2. That is, the expected number of decreases in the minima-so-far of a random permutation the integers in $0, 1, \ldots, n-1$ grows with $\log n$. We summarize this finding in the following theorem.

Theorem 9. *Given the r-index built over the tree-ordered text of a taxonomy $\mathcal{T}(R) = R_1 \cdots R_d$, extended with cliff-compressed document array profiles where the LCPs follow the distribution of random permutations described in the above model, we can compute the PLCA for pattern P in $\mathcal{O}(m \log \log_w(\sigma + n/r) + d)$ time and $\Theta(r \log d)$ words of space.*

Proof. In theorem 8, we showed that we can compute the PLCA for a pattern P using the cliff-compressed document array profiles in $\mathcal{O}(m \log \log_w(\sigma + n/r) + d)$-time in $\mathcal{O}(rd)$-space. Assuming the LCP values in each row of the original document array profiles follow the distribution of random permutations, we prove there is a lower average space bound. In the Supplemental Section 2, we show that the expected number of pairs that will be stored in $CP_{\ell r}[i]$ and $CP_{r\ell}[i]$ is $\Theta(\log d)$, therefore, the total average space for the cliff-compressed document array profiles is $\Theta(r \log d)$. □

Approximate document listing

The cliff-compressed document array profiles enables us to perform an additional type of query called *approximate document listing*. Rather than returning the PLCA of the pattern P, this query returns a subset of the documents in the true document listing for P.

Definition 10. *Approximate document listing: Given a sampled cliff-compressed profile, $CP_{\ell r}[i]$ and $CP_{r\ell}[i]$, and pattern P, the approximate document listing of P is obtained by scanning all the pairs $(j, P_{DA}[i][j])$ in both $CP_{\ell r}[i]$ and $CP_{r\ell}[i]_R$ and returning the set of j where $CP_{lr,rl}[i][j] \geq |P|$.*

The motivation for this query is to avoid the potential for vague PLCAs, especially those near the root of the taxonomy. For example, suppose we have a substring M from a read that occurs in n documents, the PLCA of M would be the root of taxonomy even if $n-1$ of the occurrences of M were at the left-end (earliest) of the tree-order and the last occurrence was in the rightmost (latest) document in the tree-order. This illustrates a scenario when the PLCA of M gives an unclear classification even though M most likely originated from a document in the left side (earlier) of the tree-order.

We note that both the PLCA and approximate document listing queries can be performed by using the same cliff-compressed profile. Despite the fact that it fails to be a full document listing, we show in Results that the approximate document listing can be a more informative query for classification compared to the PLCA in practice.

Classification methods in Cliffy

Cliffy can perform both read-level classification and abundance profiling. Given a sequencing read, Cliffy begins at the end of the read and performs maximal left-extension using backward search on the tree-ordered text of the taxonomy. Once the backward search range is empty, Cliffy resets the range to the full BWT at position i, where the previous maximal left-extension ended. Thus, the features for classification used by Cliffy are the exact matches (of varying lengths) between the read and the reference database. Each exact match will have an associated cliff-compressed profile that allows us to either compute the PLCA or approximate document listing. For example, if we would like to classify at the genus level of the taxonomy and there are d distinct genera, then we initialize a vote array $V[1..d]$ with all zeroes. Next, we iterate through each exact match M, and compute the PLCA for M that returns the leftmost ℓ and rightmost r occurrence of M in the taxonomy. We add $\frac{|M|}{r-l+1}$ to all entries between $V[l]$ and $V[r]$, inclusive on both ends, in order to distribute the length of the exact match among all of the leaf nodes in the subtree of the PLCA of M. If we would like to use approximate document listing instead to classify the read, then we consider each exact match M and retrieve the approximate document listing $\mathcal{L}$ and then add $\frac{|M|}{|\mathcal{L}|}$ to the entries in V corresponding to documents j, where $j \in \mathcal{L}$.

Finally, the read is classified by identifying the document j with the maximum value $V[j]$ in V. In order to perform abundance profiling at the genera level, we classify each read individually and return the proportion of reads classified to each genus.

Running Kraken 2/Bracken for classification and profiling

In our experiments, we compared against Kraken 2 for read-level classification and the Kraken 2 and Braken pipeline for abundance profiling. The commands to build the Kraken 2/Bracken indexes as well as using them for classification are listed in the Supplemental Section 3.

Simulating realistic 16S rRNA datasets

In order to benchmark Cliffy and Kraken 2, we simulated 16S rRNA reads using an approach of Almeida et al. [Almeida et al., 2018]. More specifically, we implemented and used a Python tool called `MicrobeMixer` that allows us to simulate realistic 16S rRNA reads for different biomes. The commands used can be found in the Supplemental Section 4.

Similar to Almeida et al. [Almeida et al., 2018], we used EBI's MGnify API [Richardson et al., 2023] to query public metagenomic datasets from three different environments (Human Gut, Aquatic and Soil) to identify the top 100 most common genera of bacteria. Next, using the SILVA SSU NR99 database, we take all of the reference sequences for the top 100 genera and extract the sequences in different hypervariable regions (V1-V2, V3-V4, V4, V4-V5) (exact primers can be found in the Supplemental Section 4. using in-silico PCR where we allow up to 3 mismatches in the primer sequence. Next, paired-end sequencing reads 250 bp in length are simulated from these extracted regions using *ART* [Huang et al., 2012]. Finally, `MicrobeMixer` sampled reads from each of the top 100 genera, mimicking their abundance in the public data, to generate a dataset containing 10 million paired-end reads.

Reference digestion with minimizers

Cliffy uses minimizer digestion similar to SPUMONI 2 [Ahmed et al., 2023b] in order to reduce of the size of text prior to indexing. The first type of digestion is the classical case and we refer to it as "DNA minimizer", where we have a small (k) and large (w) window size. The $w - k + 1$ k-mers in large window are hashed and the minimizer is the k-mer with the smallest hash value. As Cliffy digests the text, it only concatenates minimizers that are distinct from the preceding one. The second type of digestion treats minimizers themselves as characters opposed to using a "DNA minimizer" which is a concatenation of k characters. This strategy was previously implemented in assembly-based tools such as mdbg [Ekim et al., 2021] and ntJoin [Coombe et al., 2020]. Cliffy uses 4 for the value of k since with four DNA nucleotides (A, C, G, T) there are 256 possible 4-mers which can all be represented with 1 byte. Both minimizer approaches allow Cliffy to reduce the size of the text prior to indexing which in turn reduces the size of the index.

Accelerated querying optimizations in Cliffy

There are two main optimizations in Cliffy that accelerate the query: ftab and minimizer digestion. Firstly, ftab is a lookup table used in backward search to bypass a pre-specified amount of backward search steps inspired by Bowtie 2 [Langmead and Salzberg, 2012] and Rowbowt [Mun et al., 2023]. In Cliffy, the ftab the stores the backward search ranges (start and end positions) for all possible 3-mers in the minimizer alphabet. Then at query-time, it begins by checking if the 3-mer at the end of the read is present in the text using the ftab, if so, it moves to the 4th character and proceeds with backward search as normal. Whenever Cliffy resets the backward search range to the full BWT, it will check to see if the ftab can be used to accelerate the query.

Secondly, as previously discussed, Cliffy uses minimizer digestion to reduce both the input text and the reads to shorter sequences. Importantly, this digestion accelerates the query since Cliffy treats each minimizer as its own character; thereby, if $k = 4$, Cliffy can replace four backward search steps with one.

Results

Experimental Setup

We performed the experiments on an Intel Xeon gold 6248R 24-core 3GHz processor with 1,500 GB of RAM. We ran on a 64-bit Linux platform. Time was measured using GNU `time`. Source code and the experimental

scripts using Snakemake [Köster and Rahmann, 2012] are available on GitHub (see Software availability).

We simulated 12 realistic read datasets from three different environments ("biomes") and four different hypervariable regions within the 16s rRNA gene. Each read set consisted of 10 million 250 bp Illumina paired-end reads. The exact simulation parameters can found in the Supplemental Section 4.

Comparable Methods

Other taxonomic classification methods can be broadly categorized as: (1) k-mer based approaches that index the k-mers from the reference, extract k-mers from the read, and use these as keys for finding longer matches [Wood et al., 2019, Ounit et al., 2015, Breitwieser et al., 2018, Şapcı et al., 2024, Bolyen et al., 2019], and (2) full-text approaches that create a full-text index to query with substrings of any length or perform alignments to determine the likely origin of a read [Kim et al., 2016, Song and Langmead, 2024, Ahmed et al., 2023b, Menzel et al., 2016, Altschul et al., 1990, Matsen et al., 2010].

A k-mer based index typically maps k-mers to a summary indicating which taxonomic clades contain the k-mer. Popular tools like Kraken [Wood and Salzberg, 2014] and Kraken 2 [Wood et al., 2019] store the lowest common ancestor of the genomes that include the k-mer (or minimizer of the k-mer, in the case of Kraken 2). In contrast, full-text based approaches [Kim et al., 2016, Song and Langmead, 2024, Ahmed et al., 2023b], including Cliffy, maintain data structures over the entire reference database to rapidly perform document listing queries on substrings from the sequencing read.

We compare here to Kraken 2 since it is the only other method designed for classifying individual 16S rRNA reads rather than abundance profiling. Moreover, though QIIME2 performed well in some prior benchmarking studies [Almeida et al., 2018, Odom et al., 2023], Lu et al. [Lu and Salzberg, 2020] demonstrated that Kraken 2 [Wood et al., 2019] is 300 times faster than QIIME2 and achieves higher accuracy therefore we focused on comparing to Kraken 2. The software versions for the tools tested in the Results are Cliffy (v2.0.0), Kraken 2 (v2.1.2), and Bracken (v2.9).

Cliff compression reduces index size substantially

Digestion	**Document Array Profiles** (Ahmed et al., 2023)	**Cliffy**	**Reduction Ratio**	**Mean # of Pairs**	**Variance # of Pairs**
No Digestion	2265	9.014	251×	7.095	4.633
DNA minimizer	1955	7.858	249×	7.211	4.732
Minimizer	1172	4.266	275×	5.137	2.664

Table 1: Cliffy index sizes and compression ratios on the SILVA SSU NR99 dataset. Cliffy index sizes and compression ratios on the SILVA SSU NR99 (510,508 rRNA sequences, $d = 9118$ genera). Each digestion method shows original size, Cliffy-compressed size, reduction ratio, and pair statistics (mean and variance). The expected mean number of pairs based on harmonic series ($H_{9118} + 1 = 10.695$) is discussed in Methods. All size on in GB.

We indexed the SILVA SSU NR99 database (510,508 rRNA gene sequences, 1.2 GB) to classify 16S rRNA reads and observed a substantial compression ratio (249-275-fold) for all three index types using different digestion methods (Table 1). For the minimizer index, the average profile storage decreased from $d = 9118$ values to 5.137 pairs in the cliff-compressed profile, leading to a significant reduction in index size. Based on the simple random model previously described, we expected to store $H_{9118} + 1 = 10.695$ pairs after cliff-compression. As seen in Table 1, we only needed approximately 5–7 pairs on average across index types. This discrepancy can be explained by the deviations of the real data from the simple model, where the LCP values are neither all unique nor independent from one another.

Despite this substantial reduction, Kraken 2's index remained considerably smaller than Cliffy's index (144 MB vs 4.266 GB). This is expected since Cliffy is a full-text index and not limited to a specific k-mer length. In the discussion, we explore additional methods to compress the document array profiles while retaining the ability to perform both LCA and approximate document listing.

Cliffy classifies reads with moderate runtime overhead

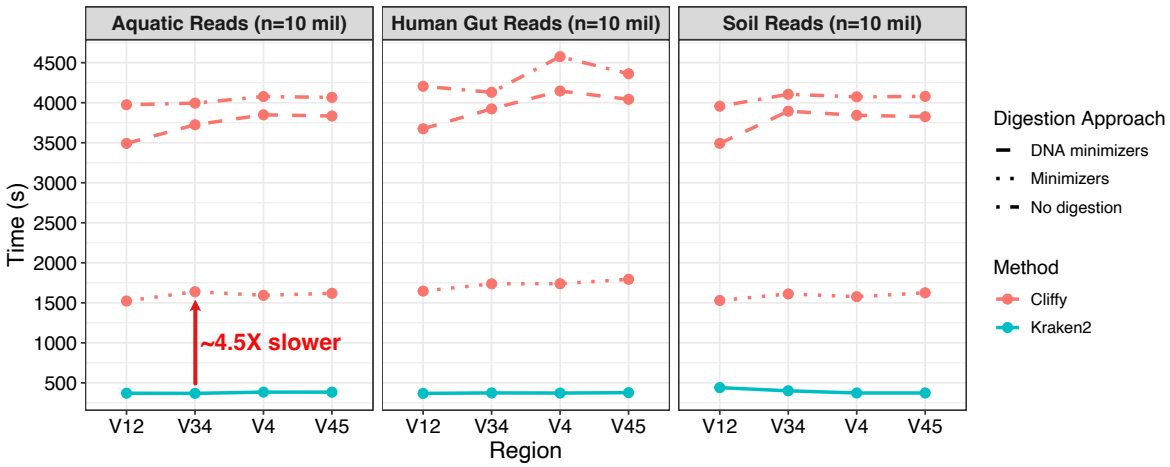


Figure 2: Classification runtime of Cliffy and Kraken 2 on simulated Illumina reads. The figure shows the time required for Cliffy (using approximate document listing) and Kraken 2 to classify 10 million simulated Illumina reads (simulated with `art_illumina` [Huang et al., 2012] using MiSeqv3 error model, 250 bp paired-end). We simulated reads for three different biomes and for each one, simulated reads from four different hypervariable regions.

We used both Cliffy and Kraken 2 to perform taxonomic read classification over these datasets and measured the running time for both. We observed that, while Cliffy was slower than Kraken 2 in all its configurations, configuring Cliffy to use its minimizer-based index yielded a running time that was the closest to Kraken 2's, being on average 4.5 times slower than Kraken 2 (Figure 2). By contrast, Cliffy's other modes are closer to 10–11× slower. This result is consistent with previous studies [Ahmed et al., 2023b, Ekim et al., 2021] that showed that using minimizers as well as a minimizer-based alphabet can help to speed up read classification. While the full-text Cliffy index is slower than Kraken 2, we suggest possibilities for improving the efficiency in the discussion.

Cliffy improves read-level taxonomic assignment accuracy

Having obtained taxonomic classifications for the simulated reads, we next compared Cliffy's classification accuracy to Kraken 2's. While we assess accuracy at all taxonomic levels, we are chiefly concerned with accuracy at the lowest level, which for our 16s RNA taxonomy was the genus level.

Measuring accuracy requires special consideration for the case where a tool's classification is non-specific (i.e. not at a leaf of the taxonomic tree) but also correct, in the sense that read's genome of origin is below the node reported by the classifier. In particular, we consider four possibilities: True Positives (TPs) are cases where the read is classified to the correct leaf node of the taxonomy. False Positives (FPs) are cases where the read is classified to a node that is neither the correct leaf node, nor an ancestor of the correct leaf node. *Vague Positives* (VPs) are cases where the read is classified to an ancestor of the correct node. Finally, *False Negatives* (FNs) are cases where the read is not classified at all by the tool, despite originating from one of the genomes in the taxonomy. Because all simulated reads have their true origin in a node in the taxonomy, True Negatives (TN) are not relevant here. Note that Cliffy reports classifications only at the taxonomic level that is specified by the user; in this case, that was the genus level. Thus, Cliffy does not report any vague positives.

To compute accuracy, we divide TP by the total number of reads in the dataset. This alone is a useful statistic for Cliffy, but for Kraken 2 we additionally report a result for accuracy that includes both TPs and VPs in the numerator. Kraken 2 results are shown as a range of values, representing a range of accuracy that depends on whether VPs are considered true or false (grey ribbon in Figure 3).

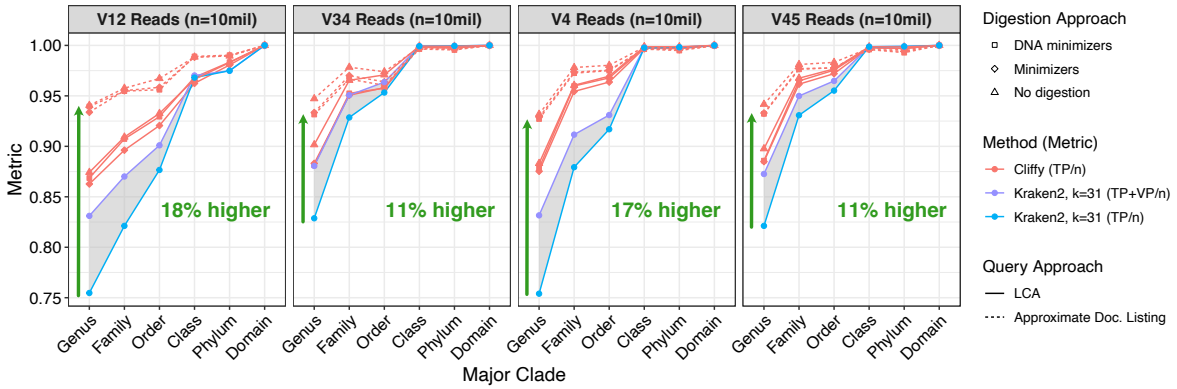


Figure 3: Read-level classification accuracy of Cliffy and Kraken 2 on the Aquatic dataset. Shows the read-level classification (n = 10 million Illumina 250 bp paired-end) accuracy for Cliffy and Kraken 2 on the Aquatic dataset using four different hypervariable regions at different levels of the tree. Each sub-plot shows Cliffy's performance when using three different types of digestion strategy, and two different query approaches: lowest common ancestor and approximate document listing. The grey shaded region ranges from Kraken 2's accuracy when treating all vague positives (VP) as incorrect to treating them all as correct.

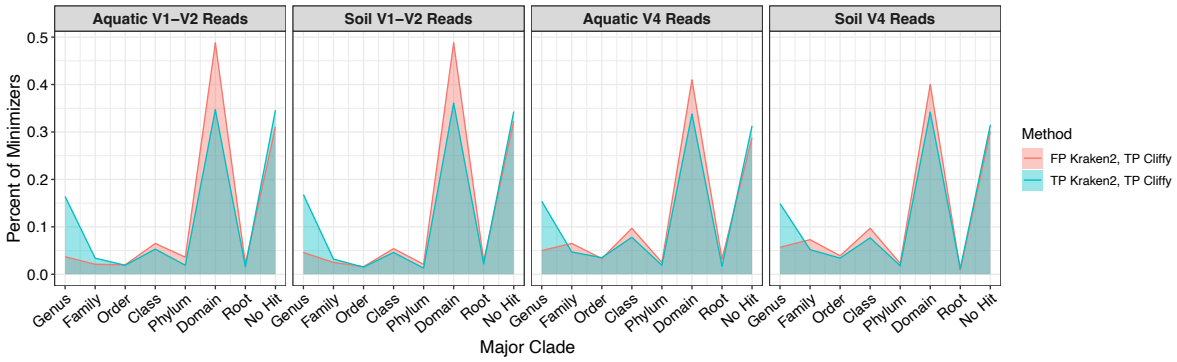


Figure 4: Analysis of minimizer hits to explain accuracy differences between Kraken 2 and Cliffy. Focuses on the four datasets with the largest accuracy gap between Kraken 2 and Cliffy (using LCA query and not approximate document listing). Looking only at the reads that were false positives for Kraken 2 and true positives for Cliffy or reads that were true positives for both tools, we added up the number of minimizers that Kraken 2 identified at each level of the taxonomy as well as minimizers that did not hit anything in the database to understand why Kraken 2's accuracy was lower than Cliffy despite both tools using the LCA query.

Consistently across the four hypervariable regions, Cliffy had higher accuracy than Kraken 2. Cliffy's accuracy percentages ranged from 11 to 18 percent points higher than Kraken 2's at the genus level. This result was consistent across the other two biomes as well. See Supplemental Figures S1-S2. To determine if adjusting Kraken 2's k-mer length would affect the its accuracy, we varied the minimizer length (k) from the default setting of $k = 31$, i.e., the maximum length allowed. We observed that Kraken 2's accuracy decreased when using smaller minimizer lengths. See Supplemental Figures S5-S7. We also observed that Cliffy's more detailed approximate document listing query was consistently more accurate than its lowest common ancestor query. Finally, when comparing the different digestion approaches, we saw that the default minimizer scheme of $k = 4$ and $w = 11$ yielded accuracy results that were near identical to those using the full input text with no digestion.

To better understand why Cliffy had a higher accuracy than Kraken 2, we focused on the four datasets for which the accuracy gap was the largest. We visualized where in the taxonomy Kraken 2 was identifying the lowest common ancestor nodes for the minimizers it identified from the reads (Figure 4). When comparing

reads that Kraken 2 classified correctly versus incorrectly (TP vs. FP), we see that Kraken 2 identifies less lowest common ancestor nodes at the genus level and instead it finds more at the domain level. Across these four datasets, we see about 30% of the minimizers are not found in the database (i.e. k is too long) and 30-50% of the minimizers are at the domain level (i.e. k is too short). This data suggests the minimizer length used by Kraken 2 is not always ideal, whereas Cliffy's ability to identify variable length matches allows it to overcome this limitation.

Cliffy accurately estimates genus-level abundances

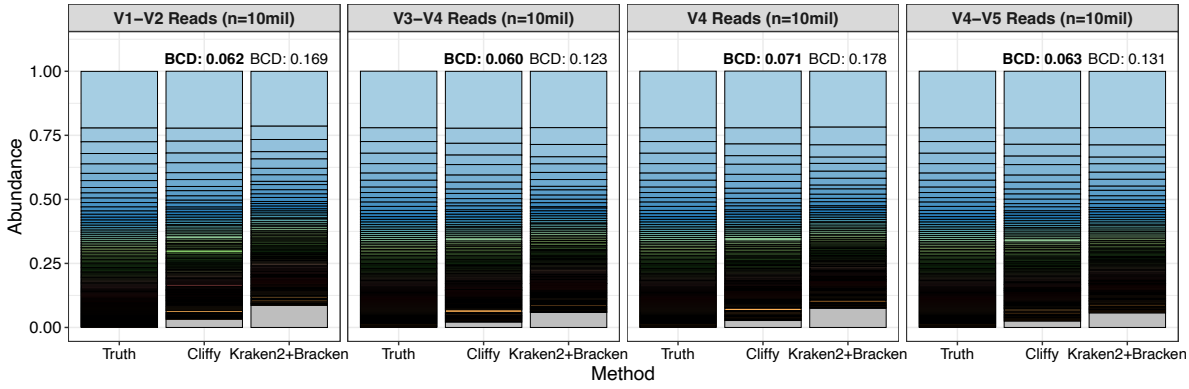


Figure 5: Comparison of genera abundance estimates by Cliffy, and the Kraken 2 and Bracken pipeline on the Aquatic dataset. Shows the genera abundance results for Cliffy and the Kraken 2 and Bracken pipeline for the Aquatic dataset compared the truth distribution. Each stacked bar is compared quantitatively with the truth distribution through Bray-Curtis Distance (BCD). The method with the closest distribution to the truth (i.e. lowest BCD) is distinguished with a bold BCD value. The grey section of the Cliffy and Kraken 2 bars correspond to predicted genera that are not truly present in the dataset.

The results discussed so far assessed accuracy at the granularity of individual reads. We further compared these tools at the dataset level, comparing their ability to correctly quantify the abundance of each genus. Figure 5 shows the estimated genus-level abundances for the 100 genera simulated in the Aquatic dataset. For all four regions, Cliffy generated a distribution with the lowest Bray-Curtis Distance (BCD) from the true distribution. That trend held as well for the other eight datasets. See Supplemental Figures S3-S4.

Discussion

We introduced Cliffy, which uses a full-text compressed index to perform accurate 16S rRNA gene classifications. We described a novel compression scheme for the document array profiles [Ahmed et al., 2023a], making it far more practical for taxonomic classification. Its use of full-text indexing, allows Cliffy to find exact matches of variable length, allowing for greater accuracy compared to Kraken 2 [Wood et al., 2019].

The document array profiles [Ahmed et al., 2023a] was the first document listing data structure to have a complexity ($\mathcal{O}(rd)$) that depended only on r and not n, making it particularly practical for repetitive collections like pangenomes. Here we proposed cliff compression, which greatly reduces index size at the expense of enabling only taxonomic classification queries and not full document-listing queries. The compressed index reaches an average size of $\Theta(r \log d)$, which we justified using a random model. Further, we showed that we tend to shrink the index somewhat more than expected in practice. This vast reduction in space makes Cliffy far more practical for taxonomic classification than any past full-text indexing approach.

k-mer based tools like Kraken 2 work by pre-selecting a particular value for k (e.g. $k = 31$), but there is no reason to expect any particular value of k is universally appropriate for all sequencing technologies and all portions of the tree of life. Here we saw evidence both of scenarios where a fixed value of k was too long (30% minimizers hit nothing in the database), and where it was too short (30%-50% of minimizers have an

LCA at the domain level). In the future, it will be important to continue studying full-text indexes with no pre-selection of k, as this could be critical to enabling high accuracy over time and over various technologies and clades.

Despite the query optimizations and compression, the main weakness of Cliffy is the lower computational efficiency compared to the k-mer based Kraken 2 method. Currently, Kraken 2 is about 4.5 times faster classifying reads compared to Cliffy when using minimizer digestion. Also, Kraken 2's index is about 30 times smaller (4.266 GB vs 144 MB). While this work represents a major closing of the index-size gap, we will continue to look for opportunities to close the gap in terms of both speed and index size. To reduce query time, we could consider using faster rank queries that could potentially speed up the backward search step in Cliffy's code for finding pattern matches [Ceregini et al., 2023, Díaz-Domínguez et al., 2023]. Further speed improvement could come from using an alternative data structure for the run-length-compressed BWT, such as the move-structure [Nishimoto et al., 2022, Zakeri et al., 2023] that allows for faster backward search (up to 30 times faster than current approaches) at the cost of a larger index.

To reduce Cliffy's index size, we can exploit the inherent feature of cliff-compression where the stored profile values ($CP_{\ell r,r\ell}[i][j]$) are strictly increasing, therefore, we could use a delta-encoding scheme to reduce the memory needed to store them. Recent works in k-mer indexes have used exploited the repetitiveness of colors to reduce the size of their indexes [Fan et al., 2024]. It is possible a similar approach can be applied to the cliff-compressed document array profiles if groups of documents ids tend to be stored together after applying the compression. Currently, this limitation is what makes it difficult for Cliffy to be applied directly to whole-genome databases such as GTDB [Parks et al., 2020] and initial experiments show that Cliffy's memory footprint grows rapidly when dealing with these large reference sequences. See Supplemental Figure S8 and Supplemental Table S1. Future work will focus on improving the construction algorithm to reduce the temporary disk footprint by parallelizing the construction and compressing the rows of the document array profiles on the fly opposed to delaying it till the end of the algorithm.

While this study evaluated *Cliffy* only in the context of 16s RNA analysis, the underlying algorithms and cliff compression are applicable in any taxonomic classification setting, including for whole-genome shotgun metagenomics. As with k-mer indexes or any other compressed index, it is always crucial to evaluate the achieved compression ratio. We observed that 16S rRNA sequences enable an impressive compression ratio of around 18-fold, meaning the n/r ratio achieved by the r-index was about 18. When applying these methods to other databases, the compression ratio could vary, and other compressed indexing methods, such as the one used by Centrifuger [Song and Langmead, 2024], might be more appropriate.

Software Availability

Cliffy software and the code for reproducing the experimental results presented in this study are available at GitHub (`https://github.com/oma219/cliffy` and `https://github.com/oma219/cliffy-experiments`, respectively), and as Supplemental Code.

Conflicts of Interest

The authors declare no competing interests.

Acknowledgements

We thank the Advanced Research Computing at Hopkins (ARCH) core facility (`https://www.arch.jhu.edu/`) for providing computational resources. ARCH is supported by the National Science Foundation under grant OAC-1920103. This work was also supported by the National Science Foundation under grant DBI-2029552, and by the National Institutes of Health under grants R01HG011392, R35GM139602, and T32GM119998. We also acknowledge the SILVA rRNA database maintainers for providing publicly available reference collections and the open-source software community for their contributions used in this study.

Author contributions:
Conceptualization: OA, BL;

Methodology: OA, CB;
Software: OA;
Investigation: OA;
Writing – original draft: OA;
Writing – review & editing: OA, CB, BL;
Supervision: CB, BL;
Approval of final version: all authors.

References

[Ahmed et al., 2023a] Ahmed, O., Rossi, M., Boucher, C., and Langmead, B. (2023a). Efficient taxa identification using a pangenome index. *Genome Research*, 33(7):1069–1077.

[Ahmed et al., 2021] Ahmed, O., Rossi, M., Kovaka, S., Schatz, M. C., Gagie, T., Boucher, C., and Langmead, B. (2021). Pan-genomic matching statistics for targeted nanopore sequencing. *IScience*, 24(6).

[Ahmed et al., 2023b] Ahmed, O. Y., Rossi, M., Gagie, T., Boucher, C., and Langmead, B. (2023b). SPUMONI 2: improved classification using a pangenome index of minimizer digests. *Genome Biology*, 24(1):122.

[Almeida et al., 2018] Almeida, A., Mitchell, A. L., Tarkowska, A., and Finn, R. D. (2018). Benchmarking taxonomic assignments based on 16s rrna gene profiling of the microbiota from commonly sampled environments. *GigaScience*, 7(5):giy054.

[Altschul et al., 1990] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410.

[Armstrong et al., 2020] Armstrong, J., Hickey, G., Diekhans, M., Fiddes, I. T., Novak, A. M., Deran, A., Fang, Q., Xie, D., Feng, S., Stiller, J., et al. (2020). Progressive Cactus is a multiple-genome aligner for the thousand-genome era. *Nature*, 587(7833):246–251.

[Bolyen et al., 2019] Bolyen, E., Rideout, J. R., Dillon, M. R., Bokulich, N. A., Abnet, C. C., Al-Ghalith, G. A., Alexander, H., Alm, E. J., Arumugam, M., Asnicar, F., et al. (2019). Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. *Nature Biotechnology*, 37(8):852–857.

[Breitwieser et al., 2018] Breitwieser, F. P., Baker, D. N., and Salzberg, S. L. (2018). KrakenUniq: confident and fast metagenomics classification using unique k-mer counts. *Genome Biology*, 19:1–10.

[Brown et al., 2025] Brown, N. K., Shivakumar, V. S., and Langmead, B. (2025). Improved pangenomic classification accuracy with chain statistics. In Sankararaman, S., editor, *Research in Computational Molecular Biology*, volume 15647 of *Lecture Notes in Computer Science*, pages 190–208. Springer, Cham, Switzerland.

[Burrows and Wheeler, 1994] Burrows, M. and Wheeler, D. J. (1994). A block sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation.

[Ceregini et al., 2023] Ceregini, M., Kurpicz, F., and Venturini, R. (2023). Faster Wavelet Trees with Quad Vectors. Technical report.

[Coit et al., 2016] Coit, P., Mumcu, G., Ture-Ozdemir, F., Unal, A. U., Alpar, U., Bostanci, N., Ergun, T., Direskeneli, H., and Sawalha, A. H. (2016). Sequencing of 16S rRNA reveals a distinct salivary microbiome signature in Behcet's disease. *Clinical Immunology*, 169:28–35.

[Consortium, 2012] Consortium, T. H. M. P. (2012). Structure, function and diversity of the healthy human microbiome. *Nature*, 486(7402):207–214.

[Coombe et al., 2020] Coombe, L., Nikolić, V., Chu, J., Birol, I., and Warren, R. L. (2020). ntJoin: Fast and lightweight assembly-guided scaffolding using minimizer graphs. *Bioinformatics*, 36(12):3885–3887.

[Depuydt et al., 2025] Depuydt, L., Ahmed, O. Y., Fostier, J., Langmead, B., and Gagie, T. (2025). Run-length compressed metagenomic read classification with smem-finding and tagging. *bioRxiv*.

[DeSantis et al., 2006] DeSantis, T. Z., Hugenholtz, P., Larsen, N., Rojas, M., Brodie, E. L., Keller, K., Huber, T., Dalevi, D., Hu, P., and Andersen, G. L. (2006). Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. *Applied and Environmental Microbiology*, 72(7):5069–5072.

[Díaz-Domínguez et al., 2023] Díaz-Domínguez, D., Dönges, S., Puglisi, S. J., and Salmela, L. (2023). Simple runs-bounded FM-index designs are fast. In *Proc. of the 21st International Symposium on Experimental Algorithms (SEA)*, page 7:1–7:16. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

[D'Amore et al., 2016] D'Amore, R., Ijaz, U. Z., Schirmer, M., Kenny, J. G., Gregory, R., Darby, A. C., Shakya, M., Podar, M., Quince, C., and Hall, N. (2016). A comprehensive benchmarking study of protocols and sequencing platforms for 16S rRNA community profiling. *BMC Genomics*, 17:1–20.

[Ekim et al., 2021] Ekim, B., Berger, B., and Chikhi, R. (2021). Minimizer-space de Bruijn graphs: Whole-genome assembly of long reads in minutes on a personal computer. *Cell Systems*, 12(10):958–968.

[Fan et al., 2024] Fan, J., Khan, J., Singh, N. P., Pibiri, G. E., and Patro, R. (2024). Fulgor: A fast and compact k-mer index for large-scale matching and color queries. *Algorithms for Molecular Biology*, 19(1):1–21.

[Fischer, 2010] Fischer, J. (2010). Optimal succinctness for range minimum queries. In *Latin American Symposium on Theoretical Informatics*, pages 158–169. Springer.

[Gagie et al., 2022] Gagie, T., Kashgouli, S., and Langmead, B. (2022). KATKA: A Kraken-like tool with k given at query time. In *International Symposium on String Processing and Information Retrieval (SPIRE)*, pages 191–197. Springer.

[Gagie et al., 2018] Gagie, T., Navarro, G., and Prezza, N. (2018). Optimal-time text indexing in BWT-runs bounded space. In *Proc. of the 29-th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1459–1477.

[Gagie et al., 2020] Gagie, T., Navarro, G., and Prezza, N. (2020). Fully functional suffix trees and optimal text searching in BWT-runs bounded space. *Journal of the ACM*, 67(1):1–54.

[Garrison et al., 2018] Garrison, E., Sirén, J., Novak, A. M., Hickey, G., Eizenga, J. M., Dawson, E. T., Jones, W., Garg, S., Markello, C., Lin, M. F., et al. (2018). Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nature Biotechnology*, 36(9):875–879.

[Huang et al., 2012] Huang, W., Li, L., Myers, J. R., and Marth, G. T. (2012). Art: a next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594.

[Kim et al., 2016] Kim, D., Song, L., Breitwieser, F. P., and Salzberg, S. L. (2016). Centrifuge: rapid and sensitive classification of metagenomic sequences. *Genome Research*, 26(12):1721–1729.

[Kim et al., 2014] Kim, M., Oh, H.-S., Park, S.-C., and Chun, J. (2014). Towards a taxonomic coherence between average nucleotide identity and 16s rrna gene sequence similarity for species demarcation of prokaryotes. *International Journal of Systematic and Evolutionary microbiology*, 64(Pt_2):346–351.

[Köster and Rahmann, 2012] Köster, J. and Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522.

[Langmead and Salzberg, 2012] Langmead, B. and Salzberg, S. L. (2012). Fast gapped-read alignment with Bowtie 2. *Nature Methods*, 9(4):357–359.

[Lu and Salzberg, 2020] Lu, J. and Salzberg, S. L. (2020). Ultrafast and accurate 16S rRNA microbial community analysis using Kraken 2. *Microbiome*, 8(1):124.

[Maidak et al., 1997] Maidak, B. L., Olsen, G. J., Larsen, N., Overbeek, R., McCaughey, M. J., and Woese, C. R. (1997). The RDP (ribosomal database project). *Nucleic Acids Research*, 25(1):109–110.

[Mäkinen and Navarro, 2005] Mäkinen, V. and Navarro, G. (2005). Succinct suffix arrays based on run-length encoding. In *Proc. of the 16th Annual Symposium Combinatorial Pattern Matching (CPM)*, pages 45–56. Springer.

[Manber and Myers, 1993] Manber, U. and Myers, G. (1993). Suffix arrays: a new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948.

[Matsen et al., 2010] Matsen, F. A., Kodner, R. B., and Armbrust, E. V. (2010). pplacer: linear time maximum-likelihood and bayesian phylogenetic placement of sequences onto a fixed reference tree. *BMC Bioinformatics*, 11:1–16.

[Menzel et al., 2016] Menzel, P., Ng, K. L., and Krogh, A. (2016). Fast and sensitive taxonomic classification for metagenomics with Kaiju. *Nature Communications*, 7:11257.

[Mun et al., 2023] Mun, T., Vaddadi, N. S. K., and Langmead, B. (2023). Pangenomic genotyping with the marker array. *Algorithms for Molecular Biology*, 18(1):2.

[Muthukrishnan, 2002] Muthukrishnan, S. (2002). Efficient algorithms for document retrieval problems. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 657–666.

[Navarro, 2016] Navarro, G. (2016). *Compact data structures: A practical approach*. Cambridge University Press.

[Nishimoto et al., 2022] Nishimoto, T., Kanda, S., and Tabei, Y. (2022). An Optimal-Time RLBWT Construction in BWT-Runs Bounded Space. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 229 of *LIPIcs*, pages 99:1–99:20.

[Odom et al., 2023] Odom, A. R., Faits, T., Castro-Nallar, E., Crandall, K. A., and Johnson, W. E. (2023). Metagenomic profiling pipelines improve taxonomic classification for 16S amplicon sequencing data. *Scientific Reports*, 13(1):13957.

[Ounit et al., 2015] Ounit, R., Wanamaker, S., Close, T. J., and Lonardi, S. (2015). CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC Genomics*, 16:1–13.

[Parks et al., 2020] Parks, D. H., Chuvochina, M., Chaumeil, P.-A., Rinke, C., Mussig, A. J., and Hugenholtz, P. (2020). A complete domain-to-species taxonomy for bacteria and archaea. *Nature Biotechnology*, 38(9):1079–1086.

[Policriti and Prezza, 2016] Policriti, A. and Prezza, N. (2016). Computing LZ77 in run-compressed space. In *Proc. of Data Compression Conference (DCC)*, pages 23–32.

[Quast et al., 2012] Quast, C., Pruesse, E., Yilmaz, P., Gerken, J., Schweer, T., Yarza, P., Peplies, J., and Glöckner, F. O. (2012). The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. *Nucleic Acids Research*, 41(D1):D590–D596.

[Richardson et al., 2023] Richardson, L., Allen, B., Baldi, G., Beracochea, M., Bileschi, M. L., Burdett, T., Burgin, J., Caballero-Pérez, J., Cochrane, G., Colwell, L. J., et al. (2023). Mgnify: the microbiome sequence data analysis resource in 2023. *Nucleic Acids Research*, 51(D1):D753–D759.

[Şapcı et al., 2024] Şapcı, A. O. B., Rachtman, E., and Mirarab, S. (2024). CONSULT-II: Accurate taxonomic identification and profiling using locality-sensitive hashing. *Bioinformatics*, page btae150.

[Sirén et al., 2021] Sirén, J., Monlong, J., Chang, X., Novak, A. M., Eizenga, J. M., Markello, C., Sibbesen, J. A., Hickey, G., Chang, P.-C., Carroll, A., et al. (2021). Pangenomics enables genotyping of known structural variants in 5202 diverse genomes. *Science*, 374(6574):abg8871.

[Song and Langmead, 2024] Song, L. and Langmead, B. (2024). Centrifuger: lossless compression of microbial genomes for efficient and accurate metagenomic sequence classification. *Genome Biology*, 25(1):106.

[Sun et al., 2009] Sun, Y., Cai, Y., Liu, L., Yu, F., Farrell, M. L., McKendree, W., and Farmerie, W. (2009). ESPRIT: estimating species richness using large collections of 16S rRNA pyrosequences. *Nucleic Acids Research*, 37(10):e76–e76.

[The MetaSUB Consortium, 2016] The MetaSUB Consortium (2016). The metagenomics and metadesign of the subways and urban biomes (MetaSUB) international consortium inaugural meeting report. *Microbiome*, 4(1):24.

[Thompson et al., 2017] Thompson, L. R., Sanders, J. G., McDonald, D., Amir, A., Ladau, J., Locey, K. J., Prill, R. J., Tripathi, A., Gibbons, S. M., Ackermann, G., et al. (2017). A communal catalogue reveals earth's multiscale microbial diversity. *Nature*, 551(7681):457–463.

[Tringe and Hugenholtz, 2008] Tringe, S. G. and Hugenholtz, P. (2008). A renaissance for the pioneering 16S rRNA gene. *Current Opinion in Microbiology*, 11(5):442–446.

[Varki et al., 2025] Varki, R., Rossi, M., Ferro, E., Oliva, M., Garrison, E., Langmead, B., and Boucher, C. (2025). Accurate short-read alignment through r-index-based pangenome indexing. *Genome Research*, 35(7):1609–1620. Published June 12, 2025; Accepted April 24, 2025.

[Willner et al., 2012] Willner, D., Haynes, M. R., Furlan, M., Schmieder, R., Lim, Y. W., Rainey, P. B., Rohwer, F., and Conrad, D. (2012). Spatial distribution of microbial communities in the cystic fibrosis lung. *The ISME journal*, 6(2):471–474.

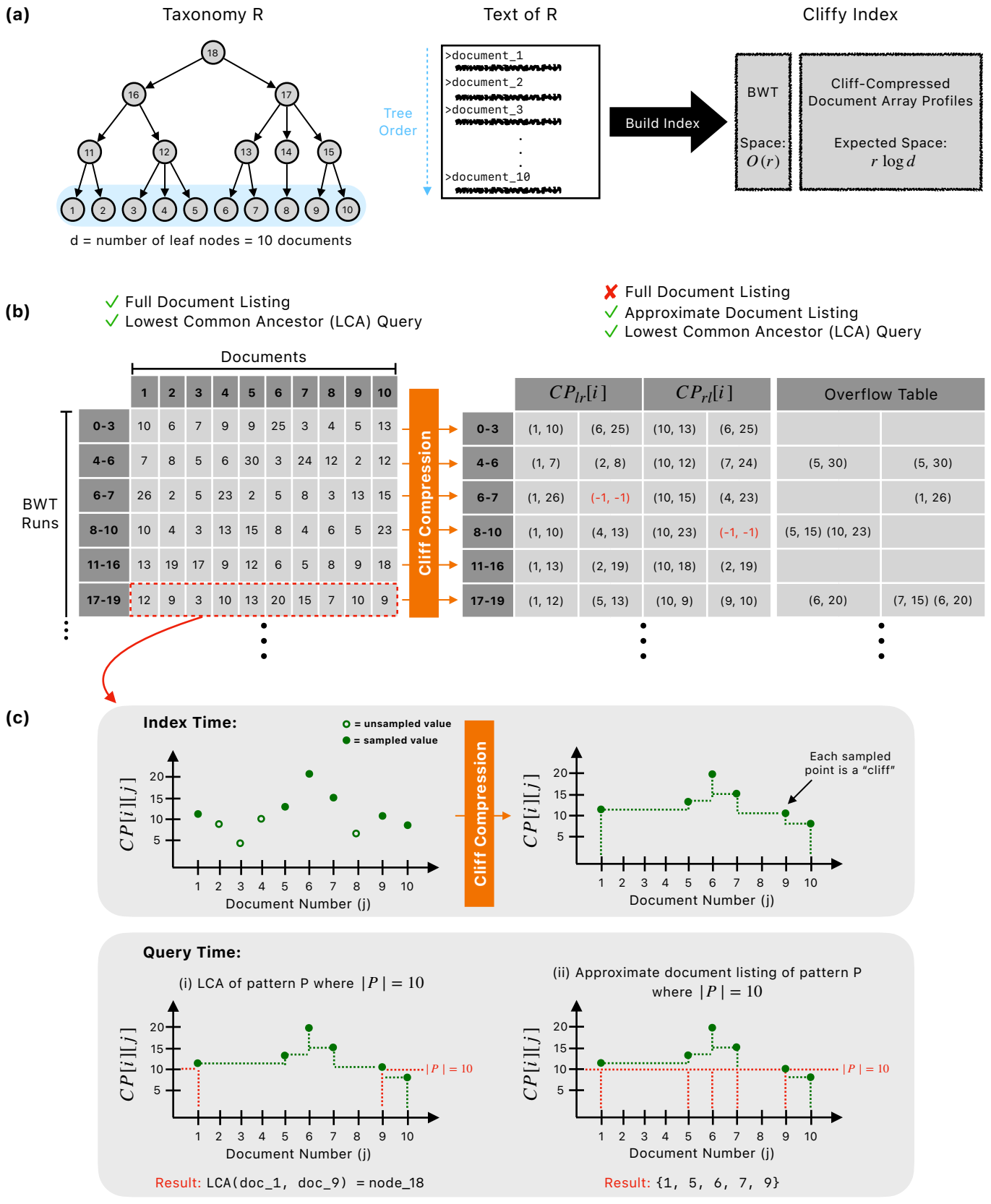
[Woese et al., 1975] Woese, C. R., Fox, G. E., Zablen, L., Uchida, T., Bonen, L., Pechman, K., Lewis, B. J., and Stahl, D. (1975). Conservation of primary structure in 16S ribosomal RNA. *Nature*, 254(5495):83–86.

[Wood et al., 2019] Wood, D. E., Lu, J., and Langmead, B. (2019). Improved metagenomic analysis with Kraken 2. *Genome Biology*, 20:1–13.

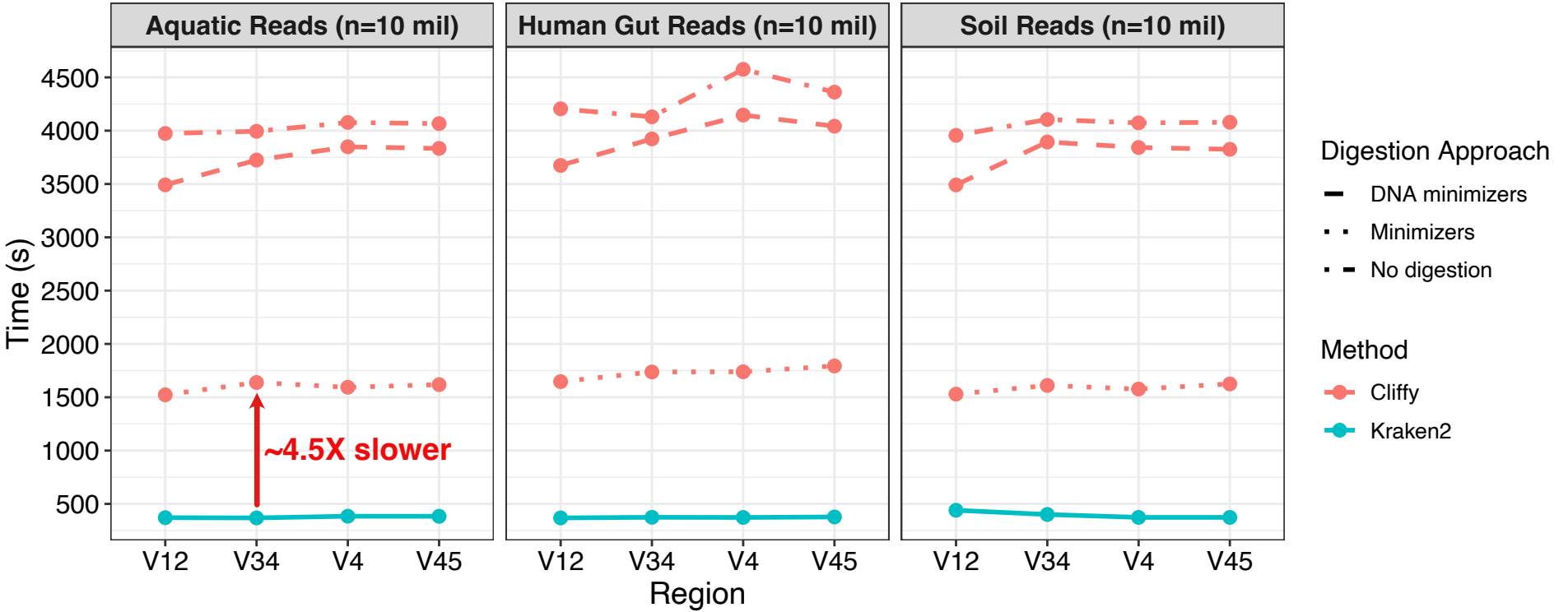
[Wood and Salzberg, 2014] Wood, D. E. and Salzberg, S. L. (2014). Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biology*, 15:1–12.

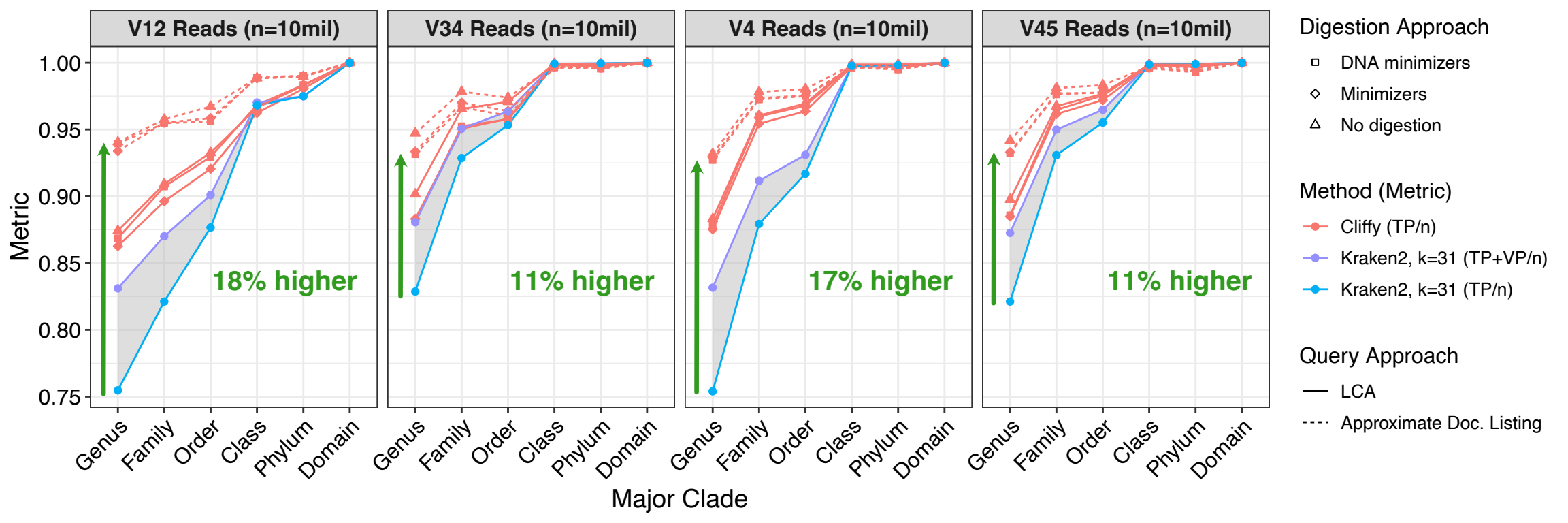
[Zakeri et al., 2023] Zakeri, M., Brown, N. K., Ahmed, O. Y., Gagie, T., and Langmead, B. (2023). Movi: a fast and cache-efficient full-text pangenome index. *bioRxiv*.

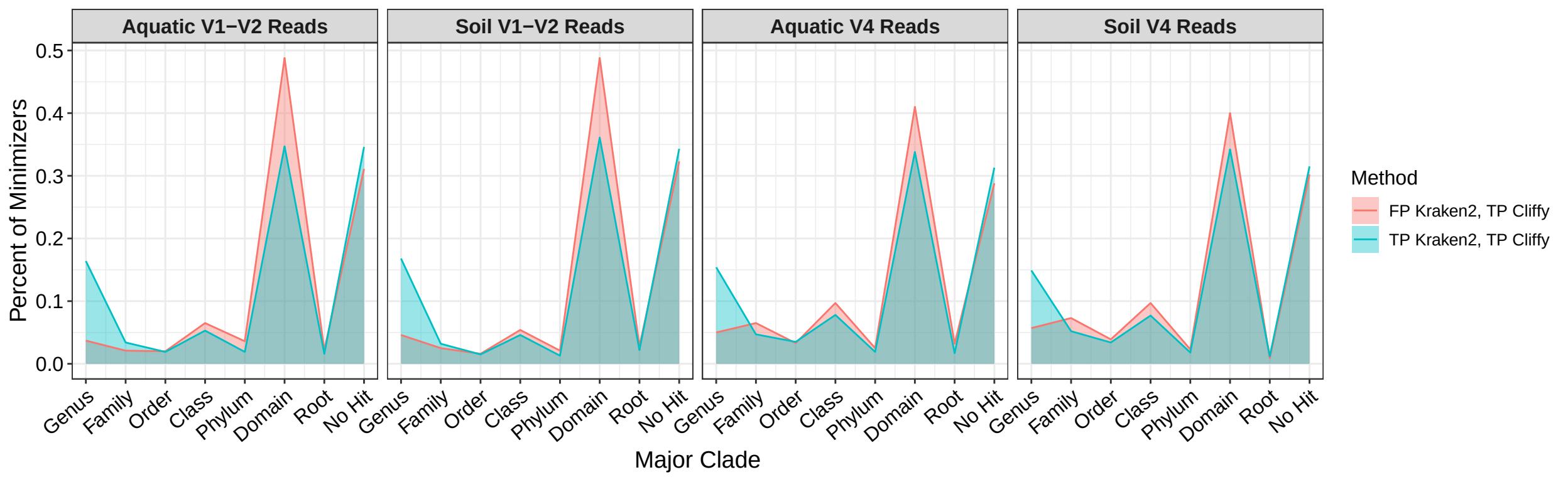
[Zakeri et al., 2024] Zakeri, M., Brown, N. K., Ahmed, O. Y., Gagie, T., and Langmead, B. (2024). Movi: A fast and cache-efficient full-text pangenome index. *iScience*, 27(12):111464.

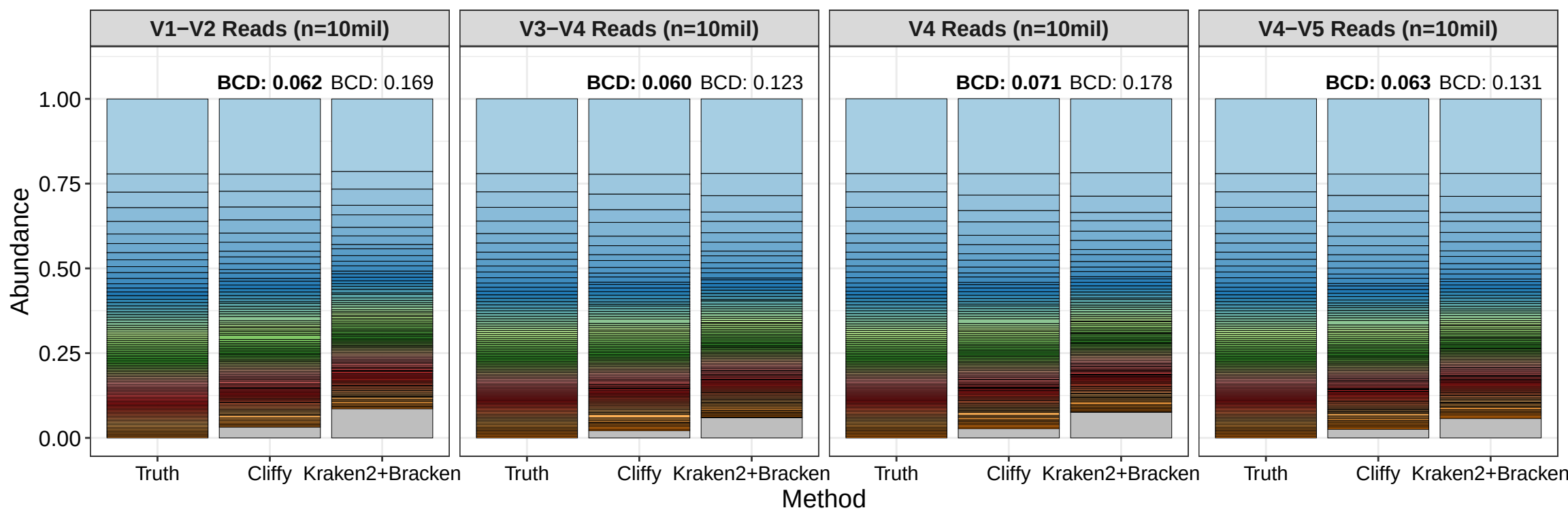


BWT Runs	1	2	3	4	5	6	7	8	9	10	$CP_{lr}[i]$		$CP_{rl}[i]$		Overflow Table	
0-3	10	6	7	9	9	25	3	4	5	13	0-3	(1, 10)	(6, 25)	(10, 13)	(6, 25)		
4-6	7	8	5	6	30	3	24	12	2	12	4-6	(1, 7)	(2, 8)	(10, 12)	(7, 24)	(5, 30)	(5, 30)
6-7	26	2	5	23	2	5	8	3	13	15	6-7	(1, 26)	(-1, -1)	(10, 15)	(4, 23)		(1, 26)
8-10	10	4	3	13	15	8	4	6	5	23	8-10	(1, 10)	(4, 13)	(10, 23)	(-1, -1)	(5, 15) (10, 23)	
11-16	13	19	17	9	12	6	5	8	9	18	11-16	(1, 13)	(2, 19)	(10, 18)	(2, 19)		
17-19	12	9	3	10	13	20	15	7	10	9	17-19	(1, 12)	(5, 13)	(10, 9)	(9, 10)	(6, 20)	(7, 15) (6, 20)


Aquatic Reads (n=10 mil)
Human Gut Reads (n=10 mil)
Soil Reads (n=10 mil)
Time (s)
4500
4000
3500
3000
2500
2000
1500
1000
500
~4.5X slower
V12
V34
V4
V45
Region
Digestion Approach
DNA minimizers
Minimizers
No digestion
Method
Cliffy
Kraken2


V12 Reads (n=10mil)
V34 Reads (n=10mil)
V4 Reads (n=10mil)
V45 Reads (n=10mil)
18% higher
11% higher
17% higher
11% higher
Metric
1.00
0.95
0.90
0.85
0.80
0.75
Genus
Family
Order
Class
Phylum
Domain
Major Clade
Digestion Approach
DNA minimizers
Minimizers
No digestion
Method (Metric)
Cliffy (TP/n)
Kraken2, k=31 (TP+VP/n)
Kraken2, k=31 (TP/n)
Query Approach
LCA
Approximate Doc. Listing


Aquatic V1−V2 Reads
Soil V1−V2 Reads
Aquatic V4 Reads
Soil V4 Reads
Percent of Minimizers
0.0
0.1
0.2
0.3
0.4
0.5
Genus
Family
Order
Class
Phylum
Domain
Root
No Hit
Major Clade
Method
FP Kraken2, TP Cliffy
TP Kraken2, TP Cliffy


V1−V2 Reads (n=10mil)
V3−V4 Reads (n=10mil)
V4 Reads (n=10mil)
V4−V5 Reads (n=10mil)
BCD: 0.062 BCD: 0.169
BCD: 0.060 BCD: 0.123
BCD: 0.071 BCD: 0.178
BCD: 0.063 BCD: 0.131
Abundance
1.00
0.75
0.50
0.25
0.00
Truth
Cliffy
Kraken2+Bracken
Method