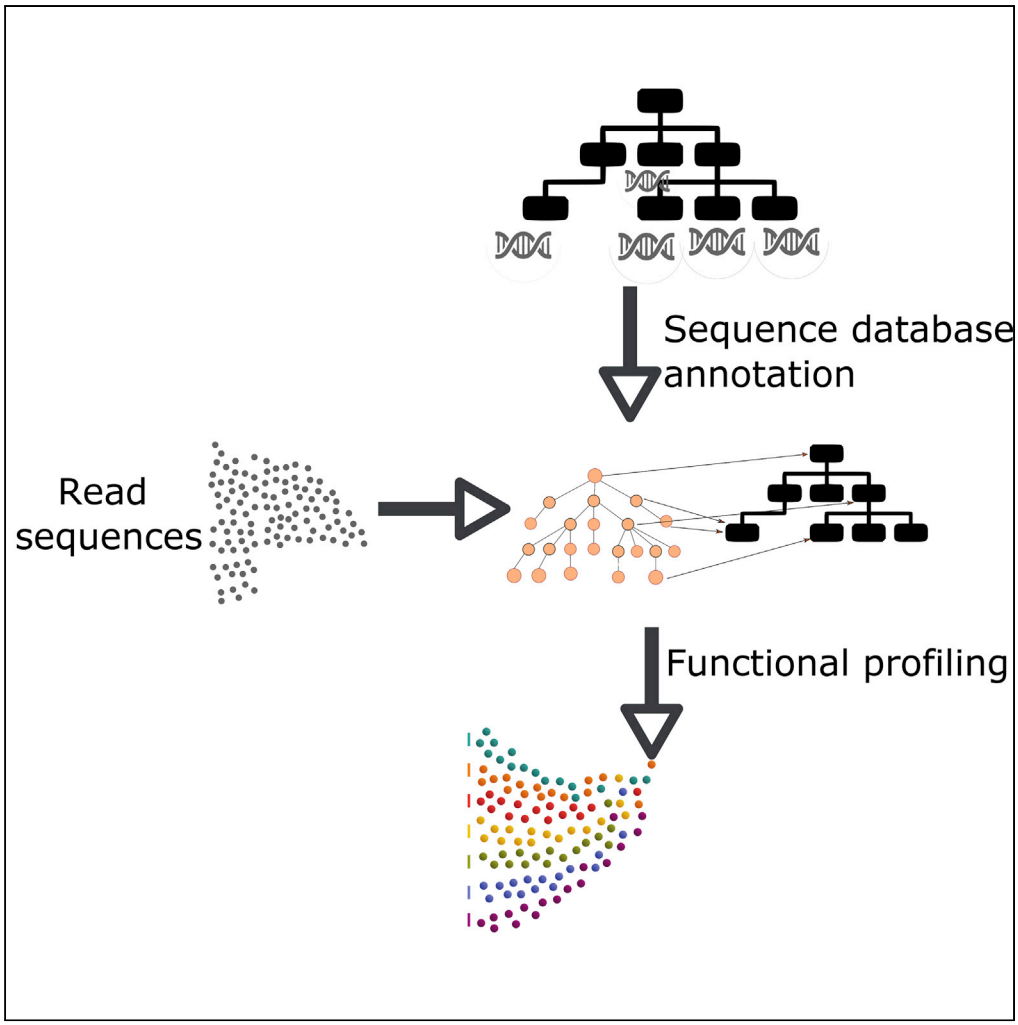
iScience

CellPress

Article

Hierarchically Labeled Database Indexing Allows Scalable Characterization of Microbiomes



Filippo Utro, Niina Haiminen, Enrico Siragusa, ..., Ritesh Krishna, James H. Kaufman, Laxmi Parida

jhkauf@us.ibm.com (J.H.K.)
parida@us.ibm.com (L.P.)

HIGHLIGHTS

Microbiome function can be characterized with respect to an annotation hierarchy

An efficient method was developed for functional classification of sequencing reads

Direct lowest taxonomic unit assignment enabled improved classification time

Biologically relevant pathways were revealed from metatranscriptomic sequencing data

DATA AND CODE AVAILABILITY

https://github.com/ComputationalGenomics/PRROMenade

Utro et al., iScience *23*, 100988
April 24, 2020 © 2020 The Author(s).
https://doi.org/10.1016/j.isci.2020.100988

iScience

CellPress

Article

Hierarchically Labeled Database Indexing Allows Scalable Characterization of Microbiomes

Filippo Utro,[1,4] Niina Haiminen,[1,4] Enrico Siragusa,[1] Laura-Jayne Gardiner,[2] Ed Seabolt,[3] Ritesh Krishna,[2] James H. Kaufman,[3,*] and Laxmi Parida[1,5,*]

SUMMARY

Increasingly available microbial reference data allow interpreting the composition and function of previously uncharacterized microbial communities in detail, via high-throughput sequencing analysis. However, efficient methods for read classification are required when the best database matches for short sequence reads are often shared among multiple reference sequences. Here, we take advantage of the fact that microbial sequences can be annotated relative to established tree structures, and we develop a highly scalable read classifier, PRROMenade, by enhancing the generalized Burrows-Wheeler transform with a labeling step to directly assign reads to the corresponding lowest taxonomic unit in an annotation tree. PRROMenade solves the multi-matching problem while allowing fast variable-size sequence classification for phylogenetic or functional annotation. Our simulations with 5% added differences from reference indicated only 1.5% error rate for PRROMenade functional classification. On metatranscriptomic data PRROMenade highlighted biologically relevant functional pathways related to diet-induced changes in the human gut microbiome.

INTRODUCTION

Microbiome analysis involves determining the composition and function of the community of microorganisms in a particular locale (Claesson et al., 2017). Variation in the human microbiome has been linked to numerous health conditions and diseases such as obesity, inflammatory bowel disease, cancer, and neurodegenerative diseases (Elinav et al., 2019; Gilbert et al., 2018). In addition to identifying the members of a diverse microbial community, an important task is understanding the biological processes that can occur in that community. Determining the metabolic processes performed by microbes, and their related host interactions, is critical for understanding how the microbiome functions, and eventually for perturbing disease-related processes. Functional annotation has been approached, for example, by alignment-based approaches matching metagenomic and metatranscriptomic sequencing reads against functionally annotated protein databases (Franzosa et al., 2018; Zhu et al., 2018) and other methods as discussed by, e.g., Knight et al. (2018) and Niu et al. (2017).

Microbial reference databases typically contain many sequences that are distinct yet highly similar, resulting in reads frequently matching multiple sequences equally well. Furthermore, microbial sequences can be organized in terms of a hierarchical annotation tree, e.g., a taxonomy of genomes or functional units. In light of these observations, an optimal strategy would thus assign reads directly to the relevant *lowest taxonomic unit* (LTU) in a taxonomic tree. This paper focuses on efficient functional classification of microbiome sequencing reads in terms of a *functional taxonomy* (such as KEGG enzyme codes Kanehisa et al., 2017). The challenge is to efficiently and accurately assign reads to the relevant LTU, given a large database of sequences that have been annotated with a functional taxonomy.

Rapid methods for classifying microbiome reads against a phylogenetic taxonomy (e.g., NCBI reference taxonomy Pruitt et al., 2007) have been introduced, such as Kraken (Wood and Salzberg, 2014) and Kraken 2 (Wood et al., 2019) that employ a *k*-mer index, i.e., a hash-based full-text index for patterns of *fixed* length *k*. This has the drawback of operating on fixed-length substrings of the read and then reconciling their matches to assign the read in the context of the taxonomy. The value of *k* also needs to be determined prior to building the search index, and *k*-mer reduction may be required with large databases to reduce the search index size. Kaiju (Menzel et al., 2016), on the other hand, allows searching for variable-length exact

[1]IBM Research, T.J. Watson Research Center, Yorktown Heights, NY 10598, USA

[2]IBM Research, The Hartree Centre, Warrington, WA4 4AD, UK

[3]IBM Research, Almaden Research Center, San Jose, CA 95120, USA

[4]These authors contributed equally

[5]Lead Contact

*Correspondence: jhkauf@us.ibm.com (J.H.K.), parida@us.ibm.com (L.P.)

https://doi.org/10.1016/j.isci.2020.100988

This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

matches in amino acid databases, with speed and accuracy comparable with or better than with methods using fixed size *k*-mers. However, answering an LTU query with Kaiju involves post-processing, and the time required depends on the number of alternative matches.

We propose a highly scalable method, PRROMenade, that combines efficient variable-length sequence classification with direct assignment of a read to its LTU node in the annotation hierarchy. PRROMenade thus utilizes the desirable aspects of both Kaiju and Kraken. To accomplish scalable classification, we employ the *generalized Burrows-Wheeler transform* (GBWT) (Burrows and Wheeler, 1994). We present a method to annotate generalized suffix arrays (Manber and Myers, 1990) and bidirectional BWTs (Schnattinger et al., 2012) in order to answer LTU queries in constant time for patterns of *arbitrary* length (see Transparent Methods and Figure S1).

We applied PRROMenade with the OMXWare database (Seabolt et al., 2019) and the KEGG enzyme function taxonomy (Kanehisa et al., 2017). In addition to simulated data confirming PRROMenade accuracy, we applied it on experimental metatranscriptome sequencing data to demonstrate its applicability in functional characterization of human microbiomes. As many as 67% of examined metatranscriptomic reads matched multiple database sequences equally well, indicating that direct hierarchical labeling could improve classification performance. Indeed, PRROMenade took less than half the time of Kaiju to classify experimental reads.

PRROMenade supports analyzing high-throughput metagenomic and metatranscriptomic sequencing experiments, in conjunction with large-scale reference databases (nucleotide or amino acid) and hierarchies for naming or functional annotation. Furthermore, our annotation method is not limited to microbiome annotation and can be used for general-purpose string annotation.

RESULTS

Database Indexing

Building the PRROMenade index on the OMXWare database (3.7 billion AA from 11.9 million protein domains) took approximately 11 h (on single core; indexing approximately 3 h and annotation approximately 8 h) and peak memory of 60 GB. The finished index had size 57 GB. On the GS database (1.1 million AA from 2,810 proteins), index building took approximately 3 min and peak memory of 70 MB. Timing could be further improved by employing parallelization techniques.

Accuracy on Simulated Data

PRROMenade classified all simulated reads on the OMXWare database. On the reads containing 5% sequencing errors, classification speed was 9.4 million reads/min, PRROMenade assigned 91.5% to the correct functional node (87.8%) or its non-root ancestor (3.7%), with 7.1% assigned to the root and only 1.5% incorrectly assigned reads. The average MEM length was 17.0 AA. In order to test situations where sequences are highly divergent from the reference database, we also simulated reads with 10% and 30% errors (differences from database sequences) to represent divergent organisms. For these data, 11.0% and 32.7% of read assignments were erroneous, respectively, indicating a linear relationship between sequence divergence and classification error rate. The average MEM length decreased to 11.4 AA and 8.4 AA, respectively. For the metatranscriptomic reads average MEM was 8.3 AA, resembling the reads with 30% errors.

Hierarchical Database Structure

The premise of this work is that protein domain sequences are more similar within subtrees of the functional hierarchy than between subtrees. One way to demonstrate this is to examine the assignments for perfect simulated reads. Since almost all database sequences are placed at leaf nodes in the tree, if reads are assigned to internal nodes it indicates shared sequences across the subtree below that node. The distribution of simulated read assignments compared with database sequences (Figure S2) shows enrichment for assignments of perfect reads particularly immediately above the leaf level (level 3; 3.9% reads versus 1.0% database content). This indicates shared sequence content in the corresponding subtrees, supporting the premise of hierarchical functional classification. The fraction of reads assigned to the root increases with the read error rate, and the metatranscriptomic data distribution resembles most that of reads with 30% errors (Figure S2).

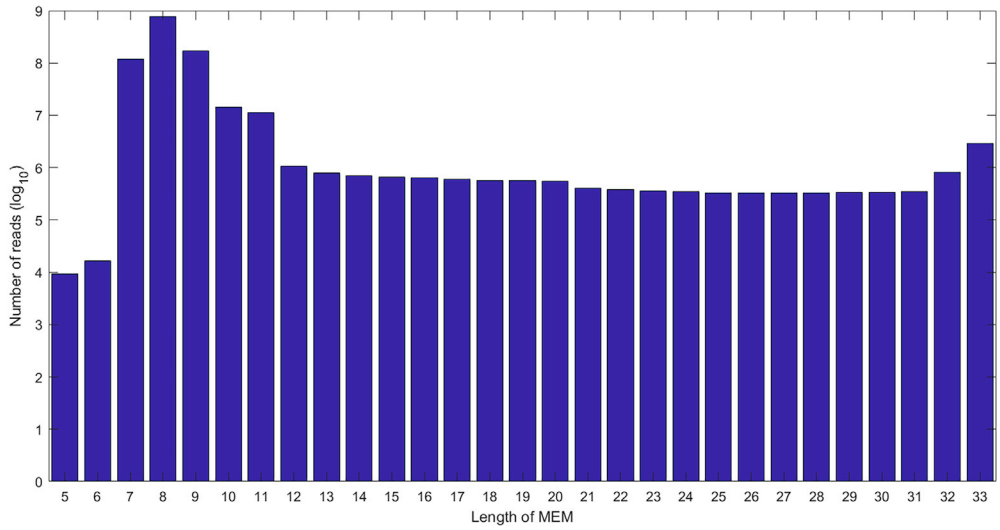


Figure 1. Distribution of Maximal Exact Match Lengths

Distribution of maximal exact match lengths on experimental metatranscriptomic data (5–33 AA) against the OMXWare database. Average MEM length is 8.3 AA.

Redundancy in the Database

We examined the degree of redundancy in the OMXWare protein domain database in terms of multiple MEMs during read classification. On simulated data with 5% sequencing errors, 10% of the reads would need an LTU computation, with an average 14.6 alternative MEMs per read. On the experimental metatranscriptomic data, 67% of the reads would need an LTU computation, with on average 11.7 alternative MEMs per read. PRROMenade avoids the LTU computation in these cases, on one hand by assigning each MEM to its corresponding node in the hierarchy directly (instead of all the individual sequences where it appears) and on the other hand by selecting a random representative in case of multiple equally long MEMs. Multi-matching is becoming an even greater issue in practice, with the number of related sequences in reference databases increasing. For example, there are currently over 200,000 assembled *Salmonella* strains in EnteroBase (Alikhan et al., 2018).

Comparison with Alignment-Based Approach

Alignment-based approach mi-faser (Zhu et al., 2018) with their GS database was compared with PRROMenade using the same database. On the simulated 50k read pairs (with 5% errors), mi-faser was orders of magnitude slower, taking over 20 min (0.00236 million reads/min), whereas PRROMenade took less than 1 s (11.2 million reads/min). In terms of accuracy, mi-faser classified 98.9% of the reads, with 98.9% of the answers to the correct function, 0.9% to an ancestor and 0.2% root (when considering the multi-mapping reads), with 0.001% errors. PRROMenade classified all reads, with 98.3% assignments to the correct function, 0.2% to an ancestor, and 1.1% to the root, with only 0.4% errors (mean MEM of 16.3 AA). However, on the experimental metatranscriptomic data mi-faser classified fewer than 1% of the reads, which combined with the classification speed makes mi-faser unusable in practice for this type of data.

Application on Metatranscriptomic Data

Classification speed on metatranscriptome reads from a study of diet-related changes in the fecal microbiome David et al. (2014) was 15.4 million reads/min for PRROMenade, compared with 6.3 million reads/min for Kaiju. PRROMenade assigned 38% of reads to a functional category and 62% to the root.

The average size of the maximal exact matches on OMXWare database was 8.3 AA (25 nt), see Figure 1, shorter than a typical *k*-mer indexing length of 31 nt. In fact, the fraction of reads with a match of 10 AA or longer was only 2.3%. Increased flexibility is indeed needed when matching reads to microbial databases inevitably missing many sequences detected in sampled microbiomes.

Clustering of the functional profiles shows better agreement with diet labels at levels 3 and 4 compared with level 2 (Figure 2). The leaf-level (level 4) clustering provides a clear separation between the diets, with only two animal-

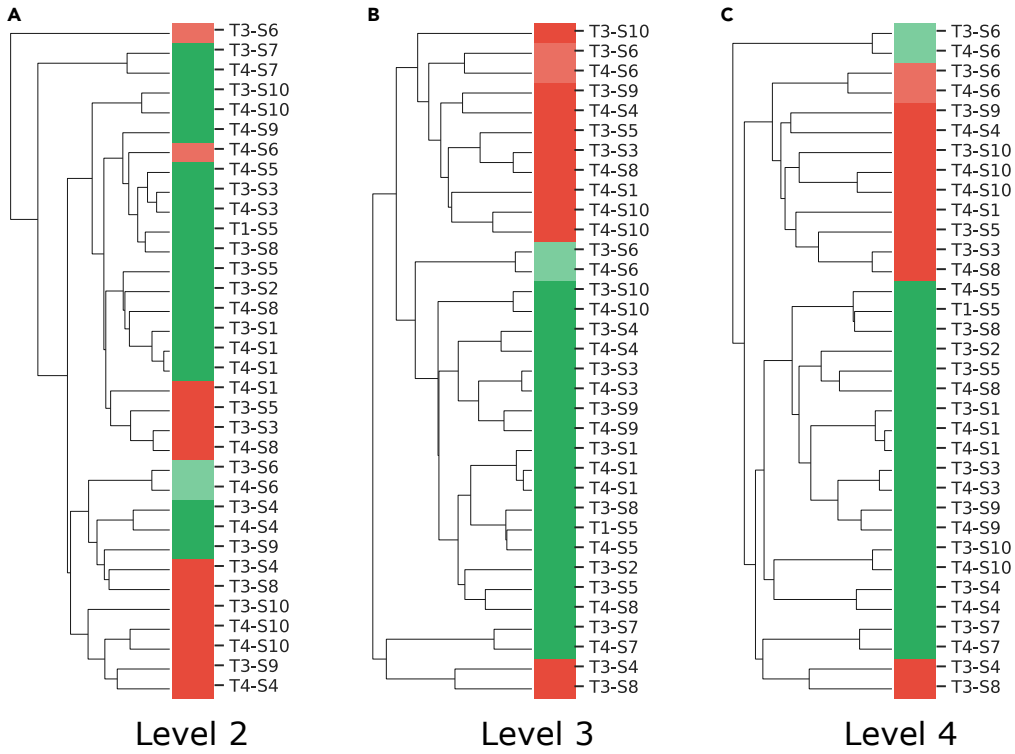


Figure 2. Clustering of Functional Profiles

Clustering of metatranscriptome PRROMenade profiles for plant- (green) and animal-based (red) diet samples at various levels of functional hierarchy. The vegetarian subject (S6) samples are denoted with a lighter shade.

based diet samples (T3-S4 and T3-S8: time T3 subjects S4 and S8) intermixed with the plant-based cluster (also observed in the original publication). Animal-based diet sample T3-S5 clusters here with other animal-based diet samples, contrary to the original publication. Samples from the same subject tend to cluster together, in particular the replicate samples for T4-S1 and T4-S10. The vegetarian subject's (S6) plant-based samples outlie the main clusters, indicating different functionality compared with all other samples.

The top differential functions are visualized in Figure 3. Although there are commonalities among the associated differential metabolic pathways and the original analysis of the data (David et al., 2014), we discovered additional functional changes in the microbiome during plant- and animal-based diets. Many of the discovered pathways relate to amino acid catabolism and biosynthesis and therefore overlap those previously noted for these data. Our analysis indicated enrichment in the plant-based diet for biosynthesis of secondary metabolites, biosynthesis of antibiotics, arginine and proline metabolism, and starch and sucrose metabolism. Streptomycin biosynthesis was an additional discovery from our analysis compared with the previous study. Streptomycin (an antibiotic) is produced by *Streptomyces* bacteria that are abundant in soils and enriched in the root microbiomes of many different plant species (Newitt et al., 2019). In the animal-based diet we detected enrichment for propanoate metabolism and microbial metabolism in diverse environments. Additionally, we detected enrichment for purine metabolism, which fits as animal-based foods tend to have higher purine content (Jakše et al., 2019).

DISCUSSION

We propose high-throughput sequencing read classification with respect to a functional hierarchy. To accomplish efficient classification, we propose a labeled indexing approach, implemented as PRROMenade, that directly provides the lowest taxonomic unit (LTU) of the maximal exact match (MEM) for a query sequence. The LTU assignment is achieved in constant time for a query sequence of any length, once the MEM has been located. This makes read classification more efficient as it avoids locating all possible multiple matches for a query and instead reports their LTU directly. PRROMenade improves on

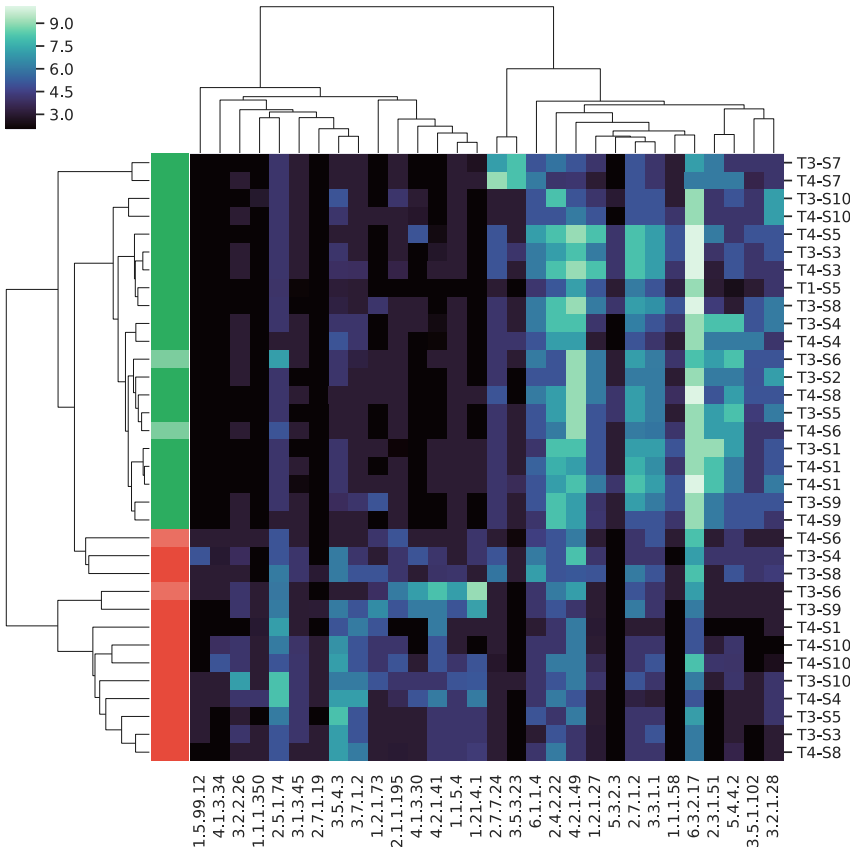


Figure 3. Clustering of Differentiating Functions

Clustering of top 30 differentially abundant functions at level 4 (columns) and of samples (rows), colored by RoDEO projected values; the left cluster shows functions enriched in animal-based diet and right cluster those enriched in plant-based diet.

the state of the art for the problem of sequence labeling, especially when faced with databases riddled with multiple near-identical sequences such as closely related microbial strains.

Our simulated experiments demonstrate that read classification in the context of a functional hierarchy with PRROMenade is efficient and accurate. Although in simulations read classification error rate scaled with the sequence divergence rate, on experimental data we detected meaningful and previously unreported differences between functional profiles of metatranscriptomes obtained for plant- and animal-based diet groups. On the experimental data the majority of sequence reads had multiple database matches and thus benefitted from the direct labeling approach, making PRROMenade more than twice as fast as Kaiju.

With very large datasets required to capture the natural microbial sequence diversity, there is a need for scalable approaches like PRROMenade. Functional classification enables advancing beyond naming of organisms to providing insights into the functional capacity of a microbiome based on high-throughput sequencing experiments. As future work, parallelizing the search index construction would allow more rapid updates of the reference, which may be required as databases keep increasing in size.

The proposed annotation method can be used in conjunction with nucleotide sequence databases in addition to protein databases and phylogenetic hierarchies in addition to functional hierarchies. Furthermore, the approach is not confined to applications on biological taxonomies and can be used for general-purpose sequence annotation.

Limitations of the Study

In this work we chose to assign the reads based on their maximal exact match to the database, using a relatively short minimum length threshold. The threshold could be increased to reduce the number of false positives, as the

simulated experiments indicated error rate scales with sequence reads' divergence from the database. However, we observed that the resulting functional assignments still separated samples based on their phenotype and provided insights into the functional differences in the respective microbial communities.

PRROMenade was designed as a rapid method for labeling query sequences in terms of an annotation hierarchy. It can separate sequences into those that do not match the database, those that are assigned to the leaf and internal nodes in the hierarchy, and those that are assigned to the root and are thus not informative. For the reads in the last category, a post-processing step could be used with a slower, more sensitive method such as alignment to possibly yield additional assignments.

METHODS

All methods can be found in the accompanying Transparent Methods supplemental file.

DATA AND CODE AVAILABILITY

All datasets used in this work are available from publicly available sources as cited in the manuscript. PRROMenade is implemented in C++ using the SeqAn library (Reinert et al., 2017). The executable is available at https://github.com/ComputationalGenomics/PRROMenade.

SUPPLEMENTAL INFORMATION

Supplemental Information can be found online at https://doi.org/10.1016/j.isci.2020.100988.

ACKNOWLEDGMENTS

The author(s) received no specific funding for this work.

AUTHOR CONTRIBUTIONS

F.U. and N.H. designed the manuscript contents and experiments. F.U., N.H., and L.-J.G. performed the analysis on simulated and experimental data. E. Siragusa implemented the indexing, and F.U. implemented the classification in PRROMenade. E. Seabolt and J.H.K. used the OMXWare database to provide functional domain sequences. L.-J.G. and R.K. contributed to data analysis. L.P. conceived of PRROMenade. J.H.K. and L.P. designed this study. E. Siragusa completed the work at IBM Research prior to joining Amazon Web Services, Berlin, Germany. All authors have written, read, and approved the manuscript.

DECLARATION OF INTERESTS

N.H., L.P., E. Siragusa, and F.U. are listed (together or partially) as co-inventors on patent applications currently pending review at the USPTO.

Received: January 20, 2020
Revised: March 2, 2020
Accepted: March 11, 2020
Published: April 24, 2020

REFERENCES

Alikhan, N.F., Zhou, Z., Sergeant, M.J., and Achtman, M. (2018). A genomic overview of the population structure of *Salmonella*. PLoS Genet. *14*, e1007261.

Burrows, M., Wheeler, D.J., 1994. A block-sorting lossless data compression algorithm. Technical Report 124. Digital SRC Research Report.

Claesson, M.J., Clooney, A.G., and O'Toole, P.W. (2017). A clinician's guide to microbiome analysis. Nat. Rev. Gastroenterol. Hepatol. *14*, 585–595.

David, L.A., Maurice, C.F., Carmody, R.N., Gootenberg, D.B., Button, J.E., Wolfe, B.E., Ling, A.V., Devlin, A.S., Varma, Y., Fischbach, M.A., et al. (2014). Diet rapidly and reproducibly alters the human gut microbiome. Nature *505*, 559–563.

Elinav, E., Garrett, W., Trinchieri, G., and Wargo, J. (2019). The cancer microbiome. Nat. Rev. Cancer *19*, 371–376.

Franzosa, E.A., McIver, L.J., Rahnavard, G., Thompson, L.R., Schirmer, M., Weingart, G., Lipson, K.S., Knight, R., Caporaso, J.G., Segata, N., and Huttenhower, C. (2018). Species-level functional profiling of metagenomes and metatranscriptomes. Nat. Methods *15*, 962–968.

Gilbert, J.A., Blaser, M.J., Caporaso, J.G., Jansson, J.K., Lynch, S.V., and Knight, R. (2018). Current understanding of the human microbiome. Nat. Med. *24*, 392–400.

Jakše, B., Jakše, B., Pajek, M., and Pajek, J. (2019). Uric acid and plant-based nutrition. Nutrients *11*, 1736.

Kanehisa, M., Furumichi, M., Tanabe, M., Sato, Y., and Morishima, K. (2017). KEGG: new perspectives on genomes, pathways, diseases and drugs. Nucleic Acids Res. *45*, D353–D361.

Knight, R., Vrbanac, A., Taylor, B., Aksenov, Callewaert, C., Debelius, J., Gonzalez, A., Kosciolek, T., McCall, L.I., McDonald, D., et al.

(2018). Best practices for analysing microbiomes. Nat. Rev. Microbiol. *16*, 410–422.

Manber, U., Myers, G., 1990. Suffix arrays: a new method for on-line string searches, in: SODA '90: Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 319–327.

Menzel, P., Ng, K.L., and Krogh, A. (2016). Fast and sensitive taxonomic classification for metagenomics with Kaiju. Nat. Commun. *7*, 11257.

Newitt, J.T., Prudence, S.M.M., Hutchings, M.I., and Worsley, S.F. (2019). Biocontrol of cereal crop diseases using *Streptomycetes*. Pathogens *8*, E78.

Niu, S.Y., Yang, J., McDermaid, A., Zhao, J., Kang, Y., and Ma, Q. (2017). Bioinformatics tools for quantitative and functional metagenome and metatranscriptome data analysis in microbes. Brief. Bioinform. *19*, 1415–1429.

Pruitt, K.D., Tatusova, T., and Maglott, D.R. (2007). NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. Nucleic Acids Res. *35*, D61–D65.

Reinert, K., Dadi, T.H., Ehrhardt, M., Hauswedell, H., Mehringer, S., Rahn, R., Kim, J., Pockrandt, C., Winkler, J., Siragusa, E., et al. (2017). The SeqAn C++ template library for efficient sequence analysis: a resource for programmers. J. Biotechnol. *261*, 157–168.

Schnattinger, T., Ohlebusch, E., and Gog, S. (2012). Bidirectional search in a string with wavelet trees and bidirectional matching statistics. Inf. Comput. *213*, 13–22.

Seabolt, E., Nayar, G., Krishnareddy, H., Agarwal, A., Beck, K., Terrizzano, I., Kandogan, E., Roth, M., Mukherjee, V., and Kaufman, J. (2019). OMXWare, a cloud-based platform for studying microbial life at scale. arXiv, arXiv:1911.02095.

Wood, D., Lu, J., and Langmead, B. (2019). Improved metagenomic analysis with kraken 2. Genome Biol. *20*, 257.

Wood, D.E., and Salzberg, S.L. (2014). Kraken: ultrafast metagenomic sequence classification using exact alignments. Genome Biol. *15*, R46.

Zhu, C., Miller, M., Marpaka, S., Vaysberg, P., Ruhlemann, M.C., Wu, G., Heinsen, F.A., Tempel, M., Zhao, L., Lieb, W., et al. (2018). Functional sequencing read annotation for high precision microbiome analysis. Nucleic Acids Res. *46*, e23.

iScience, Volume *23*

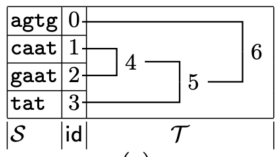
Supplemental Information

Hierarchically Labeled Database

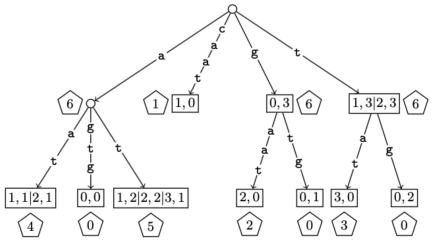
Indexing Allows Scalable

Characterization of Microbiomes

Filippo Utro, Niina Haiminen, Enrico Siragusa, Laura-Jayne Gardiner, Ed Seabolt, Ritesh Krishna, James H. Kaufman, and Laxmi Parida



(a)



(b)

		gsa		
i	tax	id	pos	$\mathcal{S}_{\text{gsa}}$
0	–	1	1	aat
1	4	2	1	aat
2	–	0	0	agtg
3	5	1	2	at
4	–	2	2	at
5	6	3	1	at
6	–	1	0	caat
7	6	0	3	g
8	–	2	0	gaat
9	–	0	1	gtg
10	6	1	3	t
11	–	2	3	t
12	–	3	2	t
13	–	3	0	tat
14	–	0	2	tg

(c)

			GBWT($\mathcal{S}$)		GBWT($\overline{\mathcal{S}}$)	
i	tax	id	gbwt	$\mathcal{S}_{\text{gsa}}$	gbwt	$\overline{\mathcal{S}}_{\text{gsa}}$
0	–	–	g	$0	a	$0
1	–	–	t	$1	c	$1
2	–	–	t	$2	g	$2
3	–	–	t	$3	t	$3
4	–	–	c	aat$1	g	a$0
5	4	–	g	aat$2	t	aac$1
6	–	0	$0	agtg$0	t	aag$2
7	5	–	a	at$1	a	ac$1
8	–	–	a	at$2	a	ag$2
9	6	–	t	at$3	t	at$3
10	–	1	$1	caat$1	a	c$1
11	6	–	t	g$0	a	g$2
12	–	2	$2	gaat$2	t	ga$0
13	–	0	a	gtg$0	$0	gtga$0
14	6	–	a	t$1	a	t$3
15	–	–	a	t$2	$1	taac$1
16	–	–	a	t$3	$2	taag$2
17	–	3	$3	tat$3	$3	tat$3
18	–	0	g	tg$0	g	tga$0

(d)

Figure S1: **Illustration of a taxonomic database and LTU assignment**, related to Figure 1. (a) Example of taxonomic database $(\mathcal{S}, \mathcal{T})$. The LTU between 1 and 3 is 5; the LTU between 0 and 4 is 6; (b) Generalized suffix tree of $\mathcal{S}$ annotated for LTU queries on $\mathcal{T}$. Non-terminal nodes are round while terminal nodes are squares. LTU annotations are shown inside pentagons. The LTU of pattern `a` is 6 while the LTU of `aa` is 4. (c) Generalized suffix array of $\mathcal{S}$ annotated for LTU queries on $\mathcal{T}$. Pattern `agtg` corresponds to a singleton with interval $[2, 3)$ and its LTU is $\mathsf{id}[2] = 0$. Pattern `a` corresponds to a leftmost node with interval $[0, 6)$ and its LTU is $\mathsf{tax}[5] = 6$. Pattern `at` is not on a singleton nor on a leftmost node, its interval is $[3, 6)$ and its LTU is $\mathsf{tax}[3] = 5$; and, (d) Generalized bidirectional BWT of $\mathcal{S}$ annotated for LTU queries on $\mathcal{T}$. Note that here we are showing the terminating character \$ for illustrative purposes, while in practice we are not using it.

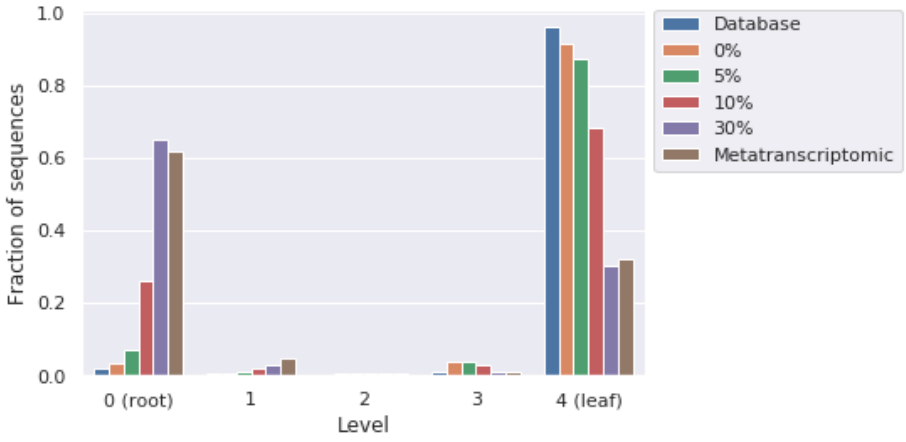


Figure S2: **Visualization of the functional hierarchy level assignments**, related to Figure 1. Distribution of OMXWare database sequence content, and of PRROMenade assignments of simulated reads with 0% to 30% errors representing divergence from the refrence, on the functional hierarchy tree from root to leaf level. In addition, metatranscriptomic read assignments are shown.

Transparent Methods

Lowest taxonomic unit problem

In the following, we describe how sequence database indexing with associated taxonomic labeling is performed using a generalized Burrows-Wheeler transform (GBWT) (Burrows and Wheeler, 1994). Here *taxonomy* refers to a microbial naming hierarchy or a functional hierarchy, organized as a tree. The database may contain either nucleotide or amino acid sequences, in the method description below we show an example using the four-letter nucleotide alphabet, for convenience.

Consider a taxonomic database $(\mathcal{S}, \mathcal{T})$ consisting of a collection of strings $\mathcal{S} = \{s_0, s_1, \ldots, s_{m-1}\}$ of total length n over the ordered alphabet $\Sigma = \{\mathtt{a}, \mathtt{c}, \mathtt{g}, \mathtt{t}\}$ and a taxonomic tree $\mathcal{T}$, i.e., a rooted tree with m leaves labeled by the indexes of strings in $\mathcal{S}$ and internal nodes referring to taxonomic units. Given a pattern string p over Σ, the problem is to retrieve the *lowest taxonomic unit* (LTU) in $\mathcal{T}$ where p occurs. Figure S1(a)shows an illustration. We denote lowest taxonomic

unit (LTU) similarly to the definition of lowest common ancestor of nodes u and v in $\mathcal{T}$ by $\textsc{lca}_{\mathcal{T}}(u, v)$ and the iterated LCA as:

$$\textsc{lca}_{\mathcal{T}}(u, \ldots, y, z) = \textsc{lca}_{\mathcal{T}}(u, \textsc{lca}_{\mathcal{T}}(\ldots, \textsc{lca}_{\mathcal{T}}(y, z))). \tag{1}$$

We define $\textsc{ids}(v) = \{k : (k, l) \in \textsc{suf}(v)\}$. The LTU of node v with $\textsc{cld}(v) = \{w_0, w_1, \ldots, w_{k-1}\}$ is:

$$\textsc{ltu}_{\mathcal{T}}(v) = \textsc{lca}_{\mathcal{T}}(\textsc{ids}(v), \textsc{ltu}_{\mathcal{T}}(w_0), \textsc{ltu}_{\mathcal{T}}(w_1), \ldots, \textsc{ltu}_{\mathcal{T}}(w_{k-1})). \tag{2}$$

We conceptualize taxonomic annotation on generalized suffix trees (Gusfield, 1997) and then show how to annotate generalized suffix arrays and bidirectional BWTs.

Generalized suffix tree

We first explain our LTU annotation algorithm on a conceptual level using generalized suffix trees. Description of practical implementations using generalized suffix arrays and generalized Burrows-Wheeler transforms follow. We adopt a practical definition of generalized suffix tree (GST) that is based on terminal nodes in addition to leaves and branching internal nodes, opposed to what traditionally done in (Gusfield, 1997). Our definition is analogous to that in (Cazaux et al., 2014) and allows us to work on arbitrary collections of strings without introducing sentinel characters, e.g., `$`. In what follows, we denote the l-th suffix of string s_k as $\mathcal{S}_{(k,l)} := s_k[l] \ldots s_k[|s_k| - 1]$ where $|s_k|$ is the length of string s_k.

The generalized suffix tree of $\mathcal{S}$, abbreviated as $\text{GST}(\mathcal{S})$, is a lexicographically ordered tree data structure having one node designated as the root. Each node v of $\text{GST}(\mathcal{S})$ provides the following operations: $\textsc{cld}(v)$ returns the nodes children of v; $\textsc{suf}(v)$ returns a list of pairs air (k, l) refers to suffix $\mathcal{S}_{(k,l)}$; $\textsc{label}(v)$ returns a string over Σ or the empty string if v is the root; the representative of a node v, $\textsc{repr}(v)$ returns $\textsc{repr}(u) \cdot \textsc{label}(v)$ where u is the parent of v or the empty string if v is the root. We say that node v is a leaf

if $|\textsc{cld}(v)| = 0$ and internal otherwise; node v is terminal if $|\textsc{suf}(v)| \geq 1$ and non-terminal otherwise.

The GST($\mathcal{S}$) has the following properties: (i) for each (k, l) in $\textsc{suf}(v)$, $\textsc{repr}(v)$ spells exactly $\mathcal{S}_{(k,l)}$; (ii) each leaf is terminal; (iii) each non-terminal node v is branching, i.e., $|\textsc{cld}(v)| \geq 2$, and $\textsc{label}(w)$ for all $w \in \textsc{cld}(v)$ begin with distinct characters.

As each suffix in $\mathcal{S}$ is associated with one terminal node, GST($\mathcal{S}$) has at most n terminal nodes. Also, because of the branching property, GST($\mathcal{S}$) has at most $n-1$ non-terminal nodes. Therefore, we have to annotate at most $2n-1$ nodes.

We now describe a conceptual annotation of GST($\mathcal{S}$) in order to answer *lowest taxonomic unit* (LTU) queries in constant time given node v. Intuitively, the LTU can be defined as the lowest common ancestor Gusfield (1997) between any two units on the taxonomic tree. Before annotating GST($\mathcal{S}$), we preprocess $\mathcal{T}$ to answer LTU queries in constant time. In practice, we reduce LTU queries to *range minimum queries* (Bender and Farach-Colton, 2000). Subsequently, we compute the LTU for all nodes of GST($\mathcal{S}$) in a single post-order traversal. The annotation of GST($\mathcal{S}$) thus takes $\mathcal{O}(n)$ time. Figure S1(b) shows an example of annotated GST.

Generalized suffix array

The *generalized suffix array* (GSA) (Manber and Myers, 1990; Shi, 1996) of $\mathcal{S}$ is a pair $(\mathsf{id}, \mathsf{pos}) := \mathsf{gsa}$ of tables of length n where $\mathsf{id}[i] = k$, $\mathsf{pos}[i] = l$ and $\mathsf{gsa}[i] = (k, l)$. Table gsa represents a permutation of all pairs (k, l) referring to suffixes $\mathcal{S}_{(k,l)}$ in $\mathcal{S}$. Pairs in gsa are ordered $\mathcal{S}_{\mathsf{gsa}[i-1]} <_{lex} \mathcal{S}_{\mathsf{gsa}[i]}$ for all $i \in [1, n)$.

In fact, table gsa corresponds to the pre-order concatenation of $\textsc{suf}(v)$ for all terminal nodes v in GST($\mathcal{S}$). Each node v of GST($\mathcal{S}$) is univocally identified by an half-open interval $[\textsc{lb}(v), \textsc{rb}(v))$ on table gsa. $\textsc{lb}(v)$ is defined as the smallest index l of GSA for which $S_{gsa}[l]$ starts with $\textsc{repr}(v)$. $\textsc{rb}(v)$ is defined as the greatest index $r > l$ for which $S_{gsa}[r-1]$ starts with $\textsc{repr}(v)$. Each node v of GST($\mathcal{S}$) can be determined by binary searching $\textsc{repr}(v)$ on gsa. GST($\mathcal{S}$) corresponds to a recursive partitioning of gsa: the

root node corresponds to interval $[0, n)$; if v is an internal node with $\textsc{cld}(v) = \{w_0, w_1, \ldots, w_{k-1}\}$ then its children intervals are $[\textsc{lb}(v) + |\textsc{suf}(v)|, \textsc{rb}(w_0))$, $[\textsc{rb}(w_0), \textsc{rb}(w_1))$, $\ldots$, $[\textsc{rb}(w_{k-1}), \textsc{rb}(v))$.

GSA($\mathcal{S}$) can be traversed in linear-time, bottom-up using the additional lcp table (Kasai et al., 2001) and top-down using lcp and child tables (Abouelhoda et al., 2004). If top-down traversal is bounded to relatively short patterns, binary search on table gsa is a practical alternative. GSA($\mathcal{S}$) supports pattern search in time $\mathcal{O}(|p| \log n)$ using only table gsa, $\mathcal{O}(|p| + \log n)$ using table lcp and $\mathcal{O}(|p|)$ using lcp (Manber and Myers, 1990) and child tables (Abouelhoda et al., 2004).

We now describe our method to store and retrieve LTUs in constant time once we reach a node v. Traversal on GST($\mathcal{S}$) to compute LTUs is readily translated onto GSA($\mathcal{S}$). The problem is how to store and retrieve annotations in GSA($\mathcal{S}$). We say that leaf v is singleton if $\textsc{lb}(v) = \textsc{rb}(v) - 1$. Furthermore, we say that a node v with parent u is leftmost if $\textsc{lb}(u) = \textsc{lb}(v)$; we determine this condition in constant time by remembering the parent u of v while traversing GSA($\mathcal{S}$) top-down. Note that we define the root to be non-leftmost. We introduce a table tax of size n to store the annotations of all non-singleton nodes. We annotate the LTU of node v as:

$$\textsc{ltu}_{\mathcal{T}}(v) = \begin{cases} \mathsf{id}[\textsc{lb}(v)] & \text{if } v \text{ is singleton,} \\ \mathsf{tax}[\textsc{rb}(v) - 1] & \text{if } v \text{ is leftmost,} \\ \mathsf{tax}[\textsc{lb}(v)] & \text{otherwise.} \end{cases} \tag{3}$$

Figure S1(c) shows an example of annotated GSA.

If v is singleton, its LTU is already annotated in table id at position $\textsc{lb}(v)$. We show by induction on GST($\mathcal{S}$) that each non-singleton node v is annotated at an available slot in tax. Prior to annotation both $\mathsf{tax}[\textsc{lb}(v)]$ and $\mathsf{tax}[\textsc{rb}(v)-1]$ are available; after annotation one of these two slots remains available.

1. If v is a leaf. All slots in tax within interval $[\textsc{lb}(v), \textsc{rb}(v))$ are available and $\textsc{lb}(v) < \textsc{rb}(v) - 1$ as v is non-singleton. Hence, if v is leftmost $\mathsf{tax}[\textsc{lb}(v)]$ remains available, otherwise $\mathsf{tax}[\textsc{rb}(v) - 1]$ remains available.

2. If v is an internal node with children $\text{CLD}(v) = \{w_0, w_1, \ldots, w_{k-1}\}$. Node w_{k-1} is not leftmost and $\mathsf{tax}[\text{RB}(w_{k-1}) - 1]$ is supposed to be available by induction. As $\text{RB}(w_{k-1}) = \text{RB}(v)$ then $\mathsf{tax}[\text{RB}(v) - 1]$ is available.

   (a) If v is non-terminal. We have $|\text{SUF}(v)| = 0$, $\text{LB}(w_0) = \text{LB}(v)$ and w_0 is leftmost. By induction $\mathsf{tax}[\text{LB}(w_0)]$ that is $\mathsf{tax}[\text{LB}(v)]$ is supposed to be available.

   (b) If v is terminal. All slots in tax within interval $[\text{LB}(v), \text{LB}(v) + \text{SUF}(v))$ are available and $\text{SUF}(v) \geq 1$, so $\mathsf{tax}[\text{LB}(v)]$ is available.

   After annotation, if v is leftmost $\mathsf{tax}[\text{LB}(v)]$ remains available, otherwise $\mathsf{tax}[\text{RB}(v) - 1]$ remains available.

Our annotation method is more convenient than that one proposed by Abouelhoda et al. (2004) with the purpose of encoding suffix links on the enhanced suffix array. Essentially, Abouelhoda et al. (2004) annotate each non-singleton node v at position $\text{RB}(w_0) - 1$. However determining the interval of w_0 requires either binary search on gsa or access to lcp and child tables.

Generalized BWT

We denote by $\overline{s} = s[|s| - 1] \ldots s[1]s[0]$ the reversed string s and by $\overline{\mathcal{S}}$ the collection $\mathcal{S}$ with all strings reversed. We consider the ordered alphabet $\Sigma_{\$} = \{\$_1, \ldots, \$_m\} \cup \Sigma$ and append $\$_i$ to each string $s_i \in \mathcal{S}$. This is to insure that $\mathcal{S}$ is primitive, i.e., that no string in $\mathcal{S}$ is a power of some other string (Crochemore et al., 2005).

The *generalized Burrows-Wheeler transform* (GBWT) (Burrows and Wheeler, 1994) of $\mathcal{S}$ is a table gbwt of length n with:

$$\mathsf{gbwt}[i] = \begin{cases} s_{\mathsf{id}[i]}[\mathsf{pos}[i] - 1] & \text{if } \mathsf{pos}[i] > 0, \\ \$_{\mathsf{id}[i]} & \text{otherwise.} \end{cases} \tag{4}$$

Function $\text{RANK}(c, i)$ counts the number of occurrences of character $c \in \Sigma_{\$}$ in the half-open interval $[0, i)$ of gbwt and LF as:

$$\text{LF}(c, i) = \sum_{a<c} \text{RANK}(a, n) + \text{RANK}(c, i). \tag{5}$$

Similarly to GSA($\mathcal{S}$), each node v of GST($\mathcal{S}$) is univocally identified on GBWT($\mathcal{S}$) by an half-open interval $[\textsc{lb}(v), \textsc{rb}(v))$. This interval is now determined by searching $\textsc{repr}(v)$ backwards on gbwt using $\textsc{lf}$ (Ferragina and Manzini, 2000). For any two nodes u, v of GBWT($\mathcal{S}$) with $\textsc{repr}(v) = c\,\textsc{repr}(u)$ and $c \in \Sigma$, it holds:

$$\textsc{lb}(v) = \textsc{lf}(c, \textsc{lb}(u) + m) - m, \tag{6}$$

$$\textsc{rb}(v) = \textsc{lf}(c, \textsc{rb}(u) + m) - m. \tag{7}$$

We adjust the boundaries by m since we have introduced m characters \$ in $\mathcal{S}$. Function $\textsc{lf}$ is answered in $\mathcal{O}(1)$ time using $o(n\,|\Sigma| \log |\Sigma|)$ extra bits on top of gbwt Reinert et al. (2017). Pattern p is searched backwards in $\mathcal{O}(|p|)$ time.

The *bidirectional GBWT* (Schnattinger et al., 2012) consists of GBWT($\mathcal{S}$) and GBWT($\overline{\mathcal{S}}$) and allows searching simultaneously a pattern p backwards on GBWT($\mathcal{S}$) and its reverse $\overline{p}$ on GBWT($\overline{\mathcal{S}}$). Function $\textsc{lt}$ counts the number of characters lexicographically smaller than $c \in \Sigma_{\$}$ in gbwt within interval $[i, j)$:

$$\textsc{lt}(c, i, j) = \sum_{a<c} \textsc{rank}(a, j) - \sum_{a<c} \textsc{rank}(a, i). \tag{8}$$

If u is a node on GBWT($\mathcal{S}$), $\overline{u}$ is its corresponding node on GBWT($\mathcal{S}$) $\textsc{repr}$(u) $= \overline{\textsc{repr}}(\overline{u})$. Furthermore, if v is with $\textsc{repr}(v) = c\,\textsc{repr}(u)$, the interval of node $\overline{v}$ with $\textsc{repr}(\overline{v}) = \textsc{repr}(\overline{u})\,c$ is determined as:

$$\textsc{lb}(\overline{v}) = \textsc{lb}(\overline{u}) + \textsc{lt}(c, \textsc{lb}(u) + m, \textsc{rb}(u) + m), \tag{9}$$

$$\textsc{rb}(\overline{v}) = \textsc{lb}(\overline{v}) + \textsc{rb}(v) - \textsc{lb}(v). \tag{10}$$

We construct an unidirectional BWT to answer LTU queries using only table gbwt of GBWT($\overline{\mathcal{S}}$) plus tables id and tax of GBWT($\mathcal{S}$). If we search pattern $\overline{p}$ backwards on GBWT($\overline{\mathcal{S}}$) and arrive on a node $\overline{v}$ with $\textsc{repr}(\overline{v}) = \overline{p}$, then node v corresponds to the node reached by searching p backwards on GBWT($\mathcal{S}$) or forward on GSA($\mathcal{S}$). Therefore we can still access the annotation at node v using Eq. 3. To fill tables id and tax, we traverse top-down GBWT($\overline{\mathcal{S}}$) backwards and annotate nodes on GBWT($\mathcal{S}$) forward. Top-down traversal is $\mathcal{O}(n^2)$ in the

worst case but feasible in practice if it is bounded to relatively short patterns. Figure S1(d) shows an example of annotated BWT. We remark that table id can be obtained as a byproduct of certain BWT construction algorithms (Egidi and Manzini, 2017) instead of slicing table gsa. Furthermore, tables id and tax can be sparsified when the annotation is bounded to relatively short patterns.

Read classification

In this study we employed the approach of searching for maximal exact matches (MEM) per read in an amino acid (AA) database, as does Kaiju (Menzel et al., 2016), though other approaches could be applied. Reads with MEMs shorter than 5 AA were considered unclassified. When using nucleotide reads, the read and its reverse complement are translated, obtaining in total six AA sequences corresponding to all possible reading frames. The longest of the six MEMs is used to classify the read, in case of ties the LTU of the (at most six) alternative MEMs is chosen. The MEM is used to classify the read to the corresponding LTU node in the annotation hierarchy, and the read count for that node is incremented by one.

Paired-end sequencing typically employs fragment sizes of 500 nucleotides (167 AA). However, the median domain length in the OMXWare functional database is only 283 AA and 25% of the domains are shorter than 167 AA. Therefore, for paired-end reads we first processed each read file separately and then combined the results by summing the counts per node (for PRROMenade as well as for our experiments with Kaiju).

Functional profiling

The functional profiles (counts per node) were post-processed with a "push down" approach as in (Huson et al., 2016) to summarize the counts at various fixed levels of the hierarchy, by also including the contribution of counts assigned at higher hierarchy levels. Pairwise Spearman distances were calculated, as suggested previously (David et al., 2014) (nodes assigned $< 0.1\%$ of total counts

in each sample were first discarded) and weighted linkage hierarchical clustering applied.

Functional profiles were analyzed with RoDEO (Haiminen et al., 2014) (P=10, I=100, R = 10^7), using a two-sample Kolmogorov-Smirnov test to identify top differentially abundant functions. Thirty functional codes with the lowest p-values were selected for average linkage clustering with correlation distance, and for pathway inference by mapping to functional pathways. Enriched pathways were defined as those overlapping at least two more of the top 30 functional codes for one vs. the other diet (e.g. ec00230, *Purine metabolism*, overlaps 0 top functional codes enriched in plant-based diet and 3 top functional codes enriched in animal-based diet).

Functional hierarchy and reference database

We used the KEGG Enzyme Nomenclature codes (EC) reference hierarchy (Kanehisa et al., 2017) as the functional annotation tree. EC numbers define a four-level hierarchy with seven top-level categories that we denote as child nodes of the root. For example, 2.4.1 is a third level code (2 = "Transferases", 2.4 = "Glycosyltransferases", 2.4.1 = "Hexosyltransferases"). We used the OMXWare database of bacterial protein domains (Seabolt et al., 2019), assembled and annotated from public repositories. A subset of 11.9 million OMXWare domains (of length $\geq$ 5 AA) had associated EC annotations, providing a collection of 3.7 billion total AA representing 1,130 EC identifiers. When a domain was annotated with multiple EC numbers (4% of domains), it was instead labeled with their LTU. Median protein domain length is 283 AA (mean 312 AA). PRROMenade could also be applied on, e.g., metagenome-assembled reference genomes (Pasolli et al., 2019) and other protein database such as UniProt (The Uniprot Consortium, 2018). Indeed we also applied PRROMenade on the smaller GS database of mi-faser (Zhu et al., 2018) with 1.1 million AA from 2,810 proteins.

Read simulation

Sequencing reads were simulated uniformly at random from sequences in the OMXWare and GS databases, respectively. The protein sequences were reverse translated to DNA sequences using EMBOSS backtranseq (Chojnacki et al., 2017) and sequencing reads were simulated using SAMtools WGSIM(v. 0.3.1-r13) (Li, 2011). We generated paired-end sequencing reads of length 125bp (fragment size 250nt as many database sequences were short, i.e., 25% were shorter than 500nt) with an error rate of 5% (-e 0.05) and a mutation rate of 0.1% (-r 0.001). For the OMXWare and GS databases we generated 95,614,845 and 50,009 read pairs, respectively. For OMXWare this corresponds to 1x coverage of the database sequences and inclusion of 89.5% of them (5.8% of the sequences were skipped due to having a length shorter than 150bp or 50AA), while for the smaller GS database the process resulted in 2x read coverage and inclusion of all but one database sequence.

Metatranscriptomic data

Metatranscriptomic sequencing data of fecal microbial communities during plant- and animal-based diets (David et al., 2014) was analyzed for functional read classification. A total of 59 samples from 11 subjects, one of them a life-long vegetarian (subject S6), were downloaded from the Gene Expression Omnibus (Edgar et al., 2002) (accession GSE46761). Trim Galore (Krueger, 2019) with options –length 50 –trim-n –max_n 10 was used to trim the 100 nt long paired reads. After trimming, 1.7M to 45.3M (mean 18.5M) reads per sample were retained. We filtered out potential human RNA content of up to 6.80% per sample (min 0.02%, mean 0.77%, median 0.20%), with bowtie2 (Langdon, 2015) mapping with local mode to their pre-built GRCh38 with 1K Genomes major SNPs index. All 59 samples (21 plant-based diet, 13 animal-based diet, 25 during diet transitions) were used for the match length and taxonomic level analysis, while only the 34 samples (21 plant-based, 13 animal-based) taken during the controlled diet periods (days 1–4) were used for downstream analysis shown in Figure 1.

Evaluation details

For simulated reads, if and only if the read was assigned somewhere on the path from its originating node to the root, it was recorded as correctly classified. Timing (elapsed wall clock) was recorded on an IBM SoftLayer cloud with 72 cores Intel® Xeon® Gold 6140 CPU @ 2.30GHz and 1.5TB of RAM running Ubuntu 16.04-64. Read classification was run with 72 threads. A Kaiju (Menzel et al., 2016) (v1.7.2) index was built on the OMXWare database with KEGG taxonomy, classification parameters were -a mem -m 5 -X -z 72. mi-faser (Zhu et al., 2018) (v1.52) was compared with PRROMenade on their GS database, using paired-end reads, 12 threads, and otherwise default settings (utilizing all available CPUs per thread). Classification speed is reported as the number of reads per experiment divided by total classification time (reads/min).

References

Abouelhoda, M.I., Kurtz, S., Ohlebusch, E., 2004. Replacing suffix trees with enhanced suffix arrays. Journal of discrete algorithms 2, 53–86.

Bender, M.A., Farach-Colton, M., 2000. The LCA problem revisited, in: LATIN, Springer. pp. 88–94.

Burrows, M., Wheeler, D.J., 1994. A block-sorting lossless data compression algorithm. Technical Report 124. Digital SRC Research Report.

Cazaux, B., Lecroq, T., Rivals, E., 2014. From Indexing Data Structures to de Bruijn Graphs, in: Combinatorial Pattern Matching. CPM 2014. Lecture Notes in Computer Science, pp. 89–99.

Chojnacki, S., Cowley, A., Lee, J., Foix, A., Lopez, R., 2017. Programmatic access to bioinformatics tools from EMBL-EBI update: 2017. Nucleic Acids Research 45, W550–W553.

Crochemore, M., Désarménien, J., Perrin, D., 2005. A note on the Burrows–Wheeler transformation. Theoretical Computer Science 332, 567–572.

David, L.A., Maurice, C.F., Carmody, R.N., Gootenberg, D.B., Button, J.E., Wolfe, B.E., Ling, A.V., Devlin, A.S., Varma, Y., Fischbach, M.A., Biddinger, S.B., Dutton, R.J., Turnbaugh, P.J., 2014. Diet rapidly and reproducibly alters the human gut microbiome. Nature 505, 559–63.

Edgar, R., Domrachev, M., Lash, A.E., 2002. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. Nucleic Acids Res 30, 207–210.

Egidi, L., Manzini, G., 2017. Lightweight bwt and lcp merging via the gap algorithm, in: International Symposium on String Processing and Information Retrieval, Springer. pp. 176–190.

Ferragina, P., Manzini, G., 2000. Opportunistic data structures with applications, in: Proceedings 41st Annual Symposium on Foundations of Computer Science, IEEE. pp. 390–398.

Gusfield, D., 1997. Algorithms on strings, trees, and sequences: Computer science and computational biology. Cambridge University Press, New York, NY, USA.

Haiminen, N., Klaas, M., Zhou, Z., Utro, F., Cormican, P., Didion, T., Jensen, C., Mason, C.E., Barth, S., Parida, L., 2014. Comparative exomics of Phalaris cultivars under salt stress. BMC Genomics 15 Suppl 6, S18.

Huson, D.H., Beier, S., Flade, I., Gorska, A., El-Hadidi, M., Mitra, S., Ruscheweyh, H.J., Tappu, R., 2016. MEGAN Community Edition – Interactive Exploration and Analysis of Large-Scale Microbiome Sequencing Data. PLoS Comput Biol 12, e1004957.

Kanehisa, M., Furumichi, M., Tanabe, M., Sato, Y., Morishima, K., 2017. KEGG: new perspectives on genomes, pathways, diseases and drugs. Nucleic Acids Res 45, D353–D361.

Kasai, T., Lee, G., Arimura, H., Arikawa, S., Park, K., 2001. Linear-time longest-common-prefix computation in suffix arrays and its applications, in:

Combinatorial Pattern Matching. CPM 2001. Lecture Notes in Computer Science, p. 181–192.

Krueger, F., 2019. TrimGalore. https://github.com/FelixKrueger/TrimGalore.

Langdon, W.B., 2015. Performance of genetic programming optimised Bowtie2 on genome comparison and analytic testing (GCAT) benchmarks. BioData Min 8.

Li, H., 2011. wgsim. http://github.com/lh3/wgsim.

Manber, U., Myers, G., 1990. Suffix arrays: a new method for on-line string searches, in: SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms, pp. 319–327.

Menzel, P., Ng, K.L., Krogh, A., 2016. Fast and sensitive taxonomic classification for metagenomics with Kaiju. Nat Commun 7, 11257. URL: https://www.ncbi.nlm.nih.gov/pubmed/27071849, doi:10.1038/ncomms11257.

Pasolli, E., Asnicar, F., Manara, S., Zolfo, M., Karcher, N., Armanini, F., Beghini, F., Manghi, P., Tett, A., Ghensi, P., Collado, M., Rice, B., DuLong, C., Morgan, X., Golden, C., Quince, C., Huttenhower, C., Segata, N., 2019. Extensive unexplored human microbiome diversity revealed by over 150,000 genomes from metagenomes spanning age, geography, and lifestyle. Cell 176, e20.

Reinert, K., Dadi, T.H., Ehrhardt, M., Hauswedell, H., Mehringer, S., Rahn, R., Kim, J., Pockrandt, C., Winkler, J., Siragusa, E., Urgese, G., Weese, D., 2017. The SeqAn C++ template library for efficient sequence analysis: A resource for programmers. J Biotechnol 261, 157–168.

Schnattinger, T., Ohlebusch, E., Gog, S., 2012. Bidirectional search in a string with wavelet trees and bidirectional matching statistics. Information and Computation 213, 13–22.

Seabolt, E., Nayar, G., Krishnareddy, H., Agarwal, A., Beck, K., Terrizzano, I., Kandogan, E., Roth, M., Mukherjee, V., Kaufman, J., 2019. OMXWare, A Cloud-Based Platform for Studying Microbial Life at Scale. arXiv:1911.02095 .

Shi, F., 1996. Suffix arrays for multiple strings: A method for on-line multiple string searches, in: Jaffar, J., Yap, R.H. (Eds.), Concurrency and Parallelism, Programming, Networking, and Security. Springer. volume 1179, pp. 11–22.

The Uniprot Consortium, 2018. UniProt: a worldwide hub of protein knowledge. Nucleic Acids Research 47, D506–D515.

Zhu, C., Miller, M., Marpaka, S., Vaysberg, P., Ruhlemann, M.C., Wu, G., Heinsen, F.A., Tempel, M., Zhao, L., Lieb, W., Franke, A., Bromberg, Y., 2018. Functional sequencing read annotation for high precision microbiome analysis. Nucleic Acids Res 46, e23.