Lightweight compositional analysis of metagenomes with sourmash gather

This manuscript (permalink) was automatically generated from dib-lab/2020-paper-sourmash-gather@80b32eb on December 1, 2021.

Authors

- **Luiz Irber**
  0000-0003-4371-9659 · luizirber · luizirber
  Graduate Group in Computer Science, UC Davis; Department of Population Health and Reproduction, UC Davis · Funded by Grant XXXXXXXX

- **Phillip T. Brooks**
  0000-0003-3987-244X · brooksph · brooksph
  Department of Population Health and Reproduction, UC Davis

- **Taylor Reiter**
  0000-0002-7388-421X · taylorreiter · ReiterTaylor
  Graduate Group in Food Science, UC Davis; Department of Population Health and Reproduction, UC Davis

- **N. Tessa Pierce**
  0000-0002-2942-5331 · bluegenes · saltyscientist
  Department of Population Health and Reproduction, UC Davis · Funded by NSF 1711984

- **David Koslicki**
  0000-0002-0640-954X · dkoslicki
  Computer Science and Engineering, Pennsylvania State University

- **C. Titus Brown**
  0000-0001-6001-2677 · ctb
  Department of Population Health and Reproduction, UC Davis

Abstract

The assignment of reference genomes and taxonomy to metagenome data underlies many microbiome studies. Here we describe two algorithms for compositional analysis of metagenome sequencing data. We first investigate the *FracMinHash* sketching technique, a derivative of modulo hash that supports Jaccard containment estimation between sets of different size [1]. We implement *FracMinHash* in the sourmash software and demonstrate large-scale containment searches of metagenomes using all 700,000 currently available microbial reference genomes. We next frame shotgun metagenome compositional analysis in terms of min-set-cover, i.e. as the problem of finding a minimum collection of reference genomes that "cover" the known k-mers in a metagenome. We implement a greedy approximate solution using *FracMinHash* sketches, and evaluate its accuracy in taxonomic assignment using a CAMI community benchmark. Finally, we show that the minimum set cover can be used to guide the select of reference genomes for read mapping. sourmash is available as open source under the BSD 3-Clause license at github.com/dib-lab/sourmash/.

Introduction

Shotgun DNA sequencing of microbial communities is an important technique for studying host-associated and environmental microbiomes. By sampling the DNA sequence content of microbial communities, shotgun metagenomics enables the taxonomic and functional characterization of microbiomes [2]. However, this characterization relies critically on the methods and databases used to interpret the sequencing data [3].

Metagenome function and taxonomy is typically inferred from available reference genomes and gene catalogs, via direct genomic alignment [5], large-scale protein search [7], or k-mer matches [9]. For many of these methods, the substantial increase in the number of available reference genomes (1.1m in GenBank as of DATE) presents a significant practical obstacle to comprehensive compositional analyses, and most methods choose representative subsets of available genomic information to analyze; for example, bioBakery 3 provides a database containing 99.2k reference genomes [4].

Here, we describe a lightweight and scalable approach to compositional analysis of shotgun metagenome data based on finding the minimum set of reference genomes that accounts for all known k-mers in a metagenome. We use a mod-hash based sketching approach for k-mers to reduce memory requirements, and implement a polynomial-time greedy approximation algorithm for the minimum set analysis.

Our approach tackles the selection of appropriate reference genomes for downstream analysis and provides a computationally efficient method for taxonomic classification of metagenome data. Our implementation in the open source `sourmash` software works with reference databases containing a million or more microbial genomes and supports multiple taxonomies and private databases.

Results

We first describe *FracMinHash*, a sketching technique that supports containment estimation for metagenome datasets using k-mers. We next frame reference-based metagenome content analysis as the problem of finding a *minimum set cover* for a metagenome using a collection of reference genomes. We then evaluate the accuracy of this approach using a taxonomic classification benchmark. Finally, we demonstrate the utility of this approach by using the genomes from the minimum set cover as reference genomes for read mapping.

FracMinHash sketches support accurate containment operations

We define the *fractional MinHash*, or FracMinHash, on an input domain of k-mers, W, as follows:

$$\mathbf{FRAC}_s(W) = \{ w \leq \frac{H}{s} \mid \forall w \in W \}$$

where H is the largest possible value in the domain of $h(x)$ and $\frac{H}{s}$ is the value in the FracMinHash.

The FracMinHash is a mix of MinHash and ModHash [1]. It keeps the selection of the smallest elements from MinHash, while using the dynamic size from ModHash to allow containment estimation. However, instead of taking $0 \mod m$ elements like $\mathbf{MOD}_m(W)$, a FracMinHash uses the parameter s to select a subset of W.

FracMinHash supports containment estimation with high accuracy and low bias. **(Analytic work from David HERE.)**

- approximation formula (eqn 13 from overleaf)
- for queries into large sets (large $|A|$), bias factor is low.
- refer to appendix for derivation.

Given a uniform hash function h and $s = m$, the cardinalities of $\mathbf{FRAC}_s(W)$ and $\mathbf{MOD}_m(W)$ converge for large $|W|$. The main difference is the range of possible values in the hash space, since the FracMinHash range is contiguous and the ModHash range is not. This permits a variety of convenient operations on the sketches, including iterative downsampling of FracMinHash sketches as well as conversion to MinHash sketches.

A FracMinHash implementation accurately estimates containment between sets of different sizes

We compare the *FracMinHash* method to CMash (*Containment MinHash*) [10] and Mash Screen (*Containment Score*) [11] for containment queries in data from a mock bacterial and archaeal community where the reference genomes are largely known [12]. This data set has been used in several methods evaluations [11,13,14,15].

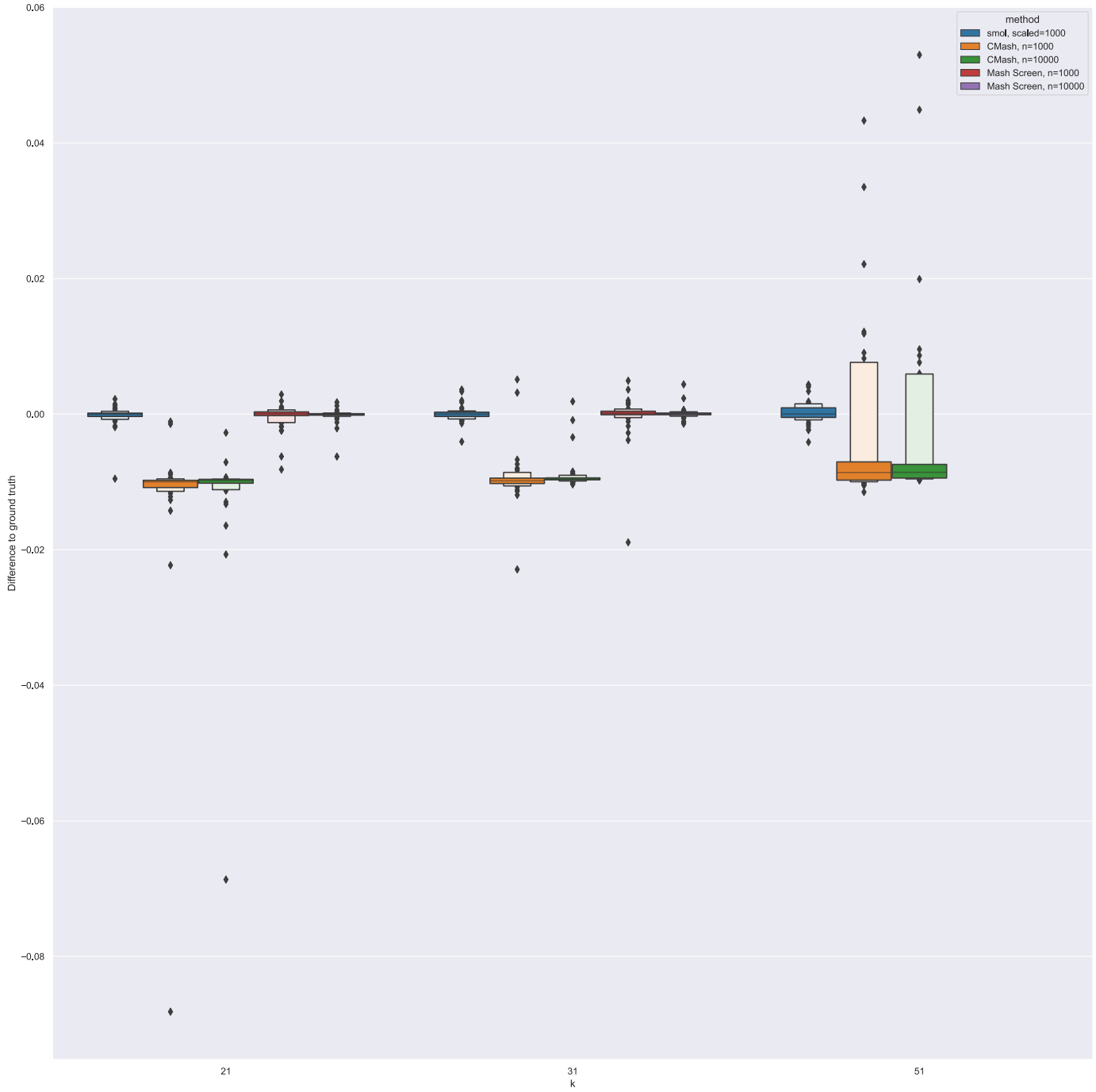

Figure 1: Letter-value plot [16] of the differences from containment estimate to ground truth (exact). Each method is evaluated for $k = \{21, 31, 51\}$, except for `Mash` with $k = 51$, which is unsupported.

Figure 1 shows containment analysis of genomes in this metagenome, with low-coverage and contaminant genomes (as described in [15] and [17]) removed from the database. All methods are within 1% of the exact containment on average (Figure 1), with `CMash` consistently underestimating the containment for large k and overestimating for small k. `Mash Screen` with $n = 10000$ has the smallest difference to ground truth for $k = \{21, 31\}$, followed by `FracMinHash` with `scaled=1000` and `Mash Screen` with $n = 1000$.

CTB TODO:

- switch figure to use sourmash; update description to reference sourmash.

We can use FracMinHash to construct a minimum set cover for metagenomes

We next ask: what is the smallest collection of genomes in a database that contains all of the known k-mers in a metagenome? Formally, for a given metagenome M and a reference database D, what is the minimum collection of genomes in D which contain all of the k-mers in the intersection of D and M? That is, we wish to find the smallest set $\{G_n\}$ of genomes in D such that

$$k(M) \cap k(D) = \bigcup_n \{k(M) \cap k(G_n)\}$$

This is the *minimum set covering* problem, for which there is a polynomial-time approximation [18]. **(Provide algorithm here.)**

This greedy algorithm iteratively subtracts k-mers belonging to the genome that has the highest containment count from the metagenome (ref alg above). This results in a progressive classification of the known k-mers in the metagenome to specific genomes, in rank order of number of contained hashes.[1]

In Figure 2, we show the results of this iterative decomposition of the mock metagenome from [12] (Table 1, row 2), into constituent genome matches. The high rank (early) matches reflect large and/or mostly-covered genomes with high containment, while later matches reflect smaller genomes, lower-covered genomes, and/or genomes with substantial overlap with earlier matches. Where there are overlaps between genomes, shared common k-mers are "claimed" by higher rank matches and only k-mer content specific to the later genome is used to identify the lower rank matches. For example, genomes from two strains of *Shewanella baltica* present in the mock metagenome in Figure 2 have an approximately 50% overlap in k-mer content, and these shared k-mers are claimed by *Shewanella baltica* OS223 (compare *Shewanella baltica* OS223, rank 8, with *Shewanella baltica* OS185, rank 33; the difference between the red circles and green triangles for *S. baltica* OS185 represents the k-mers claimed by *S. baltica* OS223). (CTB: maybe indicate or highlight these genomes in the figure?)

For this mock metagenome, 205m (54.8%) of 375m k-mers were found in GenBank. The remaining 169m (45.2%) k-mers had no matches, and represent either k-mers introduced by sequencing errors or unknown k-mers from real community members.

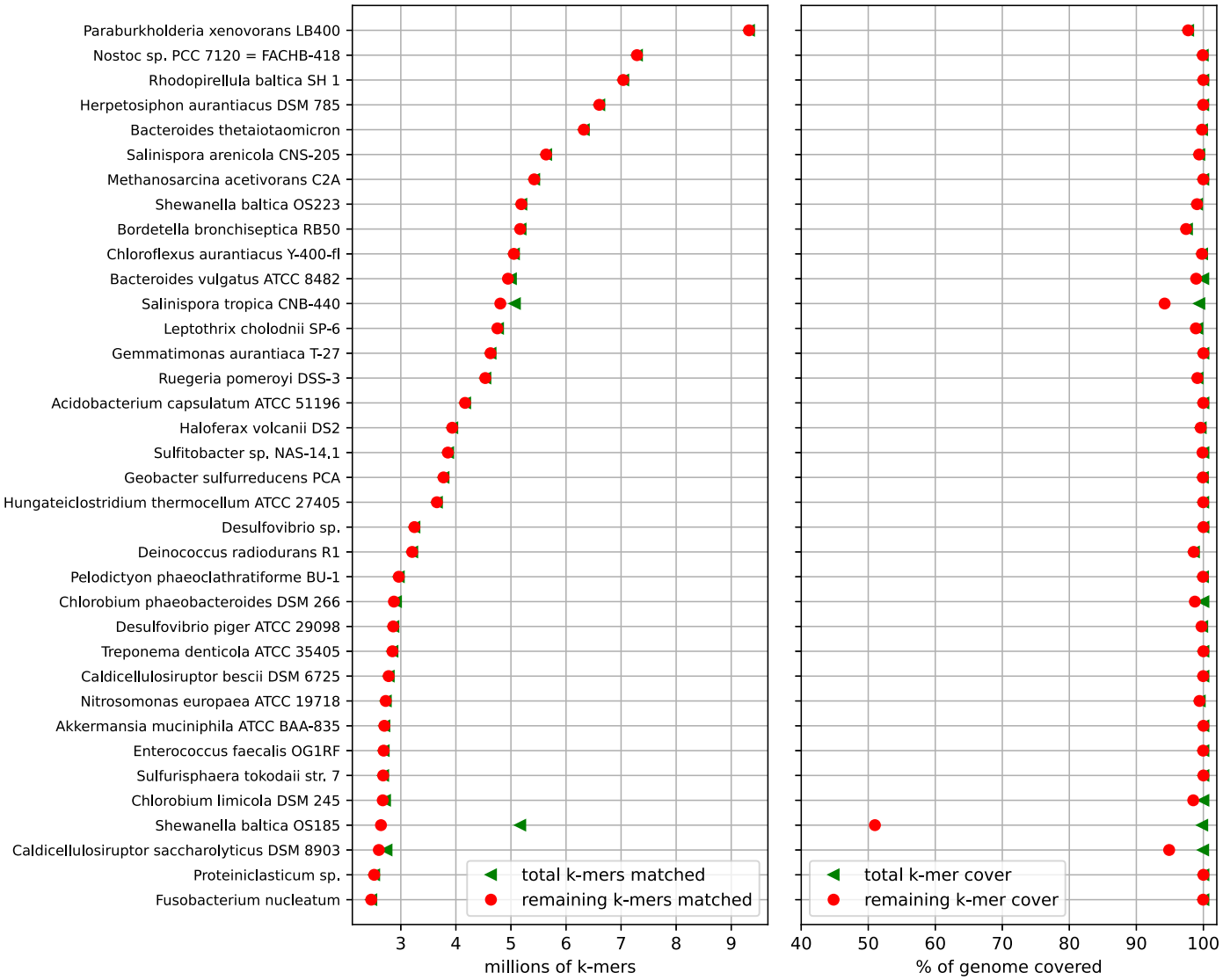

Figure 2: K-mer decomposition of a metagenome into constituent genomes. A rank ordering by remaining containment for the first 36 genomes from the minimum set cover of the `podar mock` synthetic metagenome from [12], calculated against a database containing 700,000 genomes from GenBank. The Y axis is labeled with the NCBI-designed name of the genome. In the left plot, the X axis represents the estimated number of k-mers shared between each genome and the metagenome. The red circles indicate the number of matching k-mers that were not matched at previous ranks, while the green triangle symbols indicate all matching k-mers. In the right plot, the X axis represents the estimated k-mer coverage of that genome. The red circles indicate the percentage of the genome covered by k-mers remaining at that rank, while the green triangle symbols indicate total coverage with all k-mers in the metagenome, including those already assigned at previous ranks.

Minimum metagenome covers can accurately estimate taxonomic composition

We evaluated the accuracy of min-set-cov for metagenome decomposition using benchmarks from the Critical Assessment of Metagenome Interpretation (CAMI) [19], a community-driven initiative for reproducibly benchmarking metagenomic methods. We used the mouse gut metagenome dataset [20], in which a simulated mouse gut metagenome (*MGM*) was derived from 791 bacterial and archaeal genomes, representing 8 phyla, 18 classes, 26 orders, 50 families, 157 genera, and 549 species. 64 samples were generated with *CAMISIM*, with 91.8 genomes present on each sample on average. Each sample is 5 GB in size, and both short-read (Illumina) and long-read (PacBio) simulated sequencing data is available. (CTB: check citations / content of latest actual CAMI pub, https://www.biorxiv.org/content/10.1101/2021.07.12.451567v1)

Since min-set-cov yields only a collection of genomes rather than a species list, we generated a taxonomic profile from a given metagenome cover through the following procedure. For each genome match, we noted the species designation in the NCBI taxonomy for that genome. Then, we calculated the fraction of the genome remaining in the metagenome after k-mers belonging to higher-rank genomes have been removed (red circles in Figure 2 (a)). We used this fraction to weight the contribution of the genome's species designation towards the metagenome taxonomy. This procedure produces an estimate of that species' taxonomic contribution to the metagenome, normalized by the genome size.

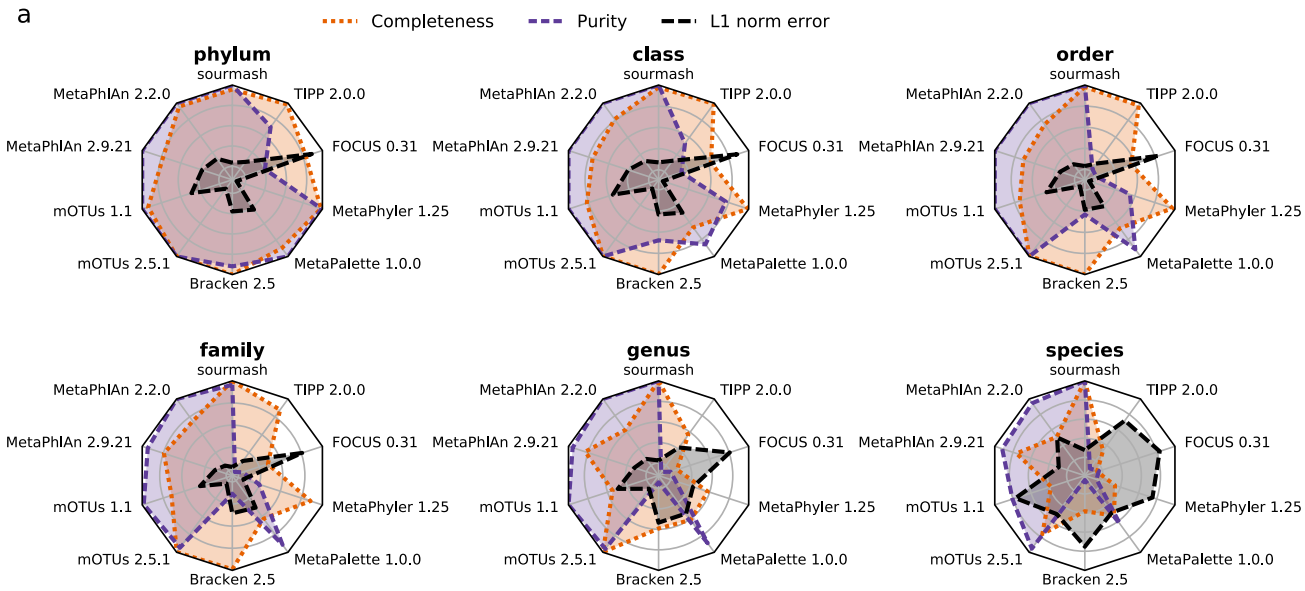

Figure 3: Comparison per taxonomic rank of methods in terms of completeness, purity (1% filtered), and L1 norm.

C	Completeness	Purity (1% filtered)	L1 norm error	Sum of scores
1st	sourmash (247)	sourmash (179)	mOTUs 2.5.1 (789)	sourmash (1262)
2nd	mOTUs 2.5.1 (416)	MetaPhlAn 2.2.0 (241)	sourmash (836)	mOTUs 2.5.1 (1887)
3rd	Bracken 2.5 (1008)	mOTUs 1.1 (631)	MetaPhlAn 2.9.21 (1401)	MetaPhlAn 2.2.0 (3527)
4th	MetaPhyler 1.25 (1298)	mOTUs 2.5.1 (682)	MetaPhlAn 2.2.0 (1497)	MetaPhlAn 2.9.21 (4349)
5th	TIPP 2.0.0 (1424)	MetaPhlAn 2.9.21 (789)	MetaPhyler 1.25 (1586)	MetaPhyler 1.25 (5148)
6th	MetaPhlAn 2.2.0 (1789)	MetaPalette 1.0.0 (1182)	mOTUs 1.1 (2317)	mOTUs 1.1 (5253)
7th	MetaPhlAn 2.9.21 (2159)	MetaPhyler 1.25 (2264)	TIPP 2.0.0 (2361)	MetaPalette 1.0.0 (5989)
8th	mOTUs 1.1 (2305)	Bracken 2.5 (2881)	MetaPalette 1.0.0 (2390)	Bracken 2.5 (6574)
9th	MetaPalette 1.0.0 (2417)	TIPP 2.0.0 (3361)	Bracken 2.5 (2685)	TIPP 2.0.0 (7146)
10th	FOCUS 0.31 (3424)	FOCUS 0.31 (3764)	FOCUS 0.31 (3894)	FOCUS 0.31 (11082)

Figure 4: Methods rankings and scores obtained for the different metrics over all samples and taxonomic ranks. For score calculation, all metrics were weighted equally.

In Figures 3 and 4 we show an updated version of Figure 6 from [20] that includes our method, implemented in the `sourmash` software (CTB: what databases are used?). Here we compare 10 different methods for taxonomic profiling and their characteristics at each taxonomic rank. While previous methods show reduced completeness – the ratio of taxa correctly identified in the ground truth – below the genus level, `sourmash` can reach 88.7% completeness at the species level with the highest purity (the ratio of correctly predicted taxa over all predicted taxa) across all methods: 95.9% when filtering predictions below 1% abundance, and 97% for unfiltered results. `sourmash` also has the lowest L1-norm error, the highest number of true positives and the lowest number of false positives.

Minimum metagenome covers select small subsets of large databases

Table 1: Four metagenomes and their estimated minimum set cover from GenBank.

data set	genomes >= 100k overlap	min-set-cov	% k-mers identified
`zymo mock` (SRR12324253)	405,839	19	47.1%
`podar mock` (SRR606249)	5800	74	54.8%
`gut real` (SRR5650070)	96,423	99	36.0%
`oil well real` (SRR1976948)	1235	135	14.9%

In Table 1, we show the results of running min-set-cov for four metagenomes against GenBank - two mock communities [12,21], one human gut microbiome data set from iHMP [2], and an oil well sample [22]. Our implementation provides estimates for both the *total* number of genomes with substantial overlap to a query genome, and a *minimum list* of genomes that account for k-mers with overlap in the query metagenome (see Methods).

We find many genomes with large overlaps for each metagenome, due to the redundancy of the reference database. For example, `zymo mock` contains a *Salmonella* genome, and there are over 200,000 *Salmonella* genomes that match to it in GenBank. Likewise, `gut real` matches to over 75,000 *E. coli* genomes in GenBank. Since neither `podar mock` nor `oil well real` contain genomes from species with substantial representation in GenBank, they yield many fewer total overlapping genomes.

Regardless of the number of genomes in the database with substantial overlap, the estimated *minimum* collection of genomes is always much smaller than the number of genomes with overlaps. In the cases where the k-mers in the metagenome are mostly identified, this is because of database redundancy: e.g. in the case of `zymo mock`, the min-set-cov algorithm chooses only one *Salmonella* genome from the 200,000+ available. Conversely, in the case of `oil well real`, much of the sample is not identified, suggesting that the small size of the covering set is because much of the sample is not represented in the database.

Minimum metagenome covers provide representative genomes for mapping

Mapping metagenome reads to representative genomes is an important step in many microbiome analysis pipelines, but mapping approaches struggle with large, redundant databases. One specific use for a minimum metagenome cover is to select a small set of representative genomes for mapping. We therefore developed a hybrid selection and mapping pipeline that uses the rank-ordered min-set-cov results to map reads to candidate genomes.

We first map all reads to all genomes in the minimum set cover, and then successively remove reads that map to higher rank genomes from lower rank genomes, and remap the remaining reads. That is, all reads mapped to the rank-1 genome in Figure 2 are removed from the rank-2 genome mapping, and all reads mapping to rank-1 and rank-2 genomes are removed from the rank-3 genome mapping, and so on. This produces results directly analogous to those presented in Figure 2, but for reads rather than k-mers (CTB: provide as Suppl Figure?). Importantly, in this process we only consider genomes identified in the minimum set cover, because it is computationally intractable to map reads to the entire GenBank database. (CTB: check centrifuge.)

Figure 5 compares hash assignment rates and mapping rates for the four evaluation metagenomes in Table 1. Broadly speaking, we see that k-mer-based estimates of metagenome composition align closely with the number of bases covered by mapped reads: the y axis has not been re-scaled, so hash matches and read mapping correspond well. This suggests that the k-mer-based min-set-cov approach effectively selects reference genomes for metagenome read mapping.

For mock metagenomes (panels X and Y), there appears to be a close correspondence between mapping and hash assignment rates, while for actual metagenomes, there is more variation between mapping and hash assignments. Further work is needed to evaluate rates of variation across a larger number of metagenomes.

CTB: update figure to contain all four metagenomes! Fix axis labels, symbols.

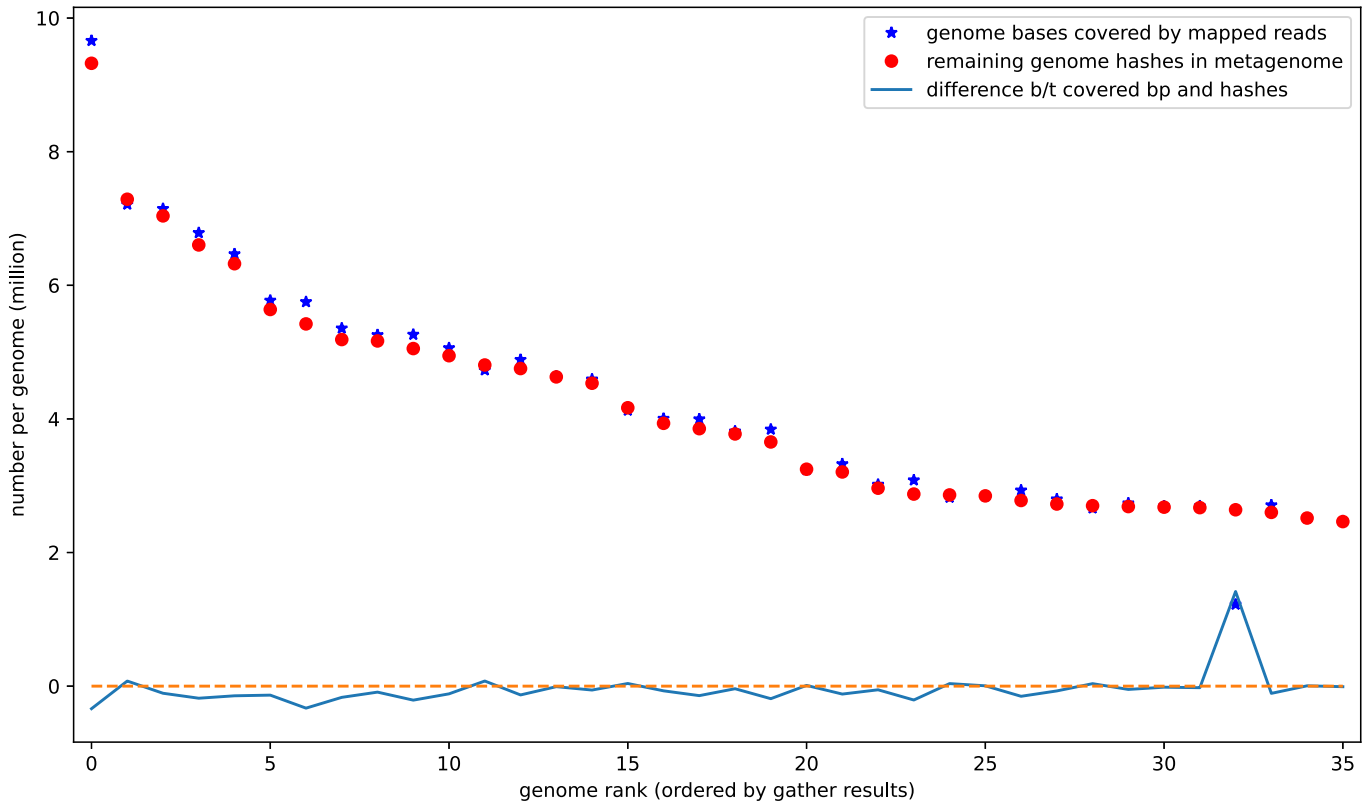

Figure 5: Hash-based k-mer decomposition of a metagenome into constituent genomes compares well to bases covered by read mapping. (CTB: add more description here.) The reference genomes are rank ordered along the x axis (as in Figure 2), based on the largest number of hashes from the metagenome specific to that genome; hence the number of hashes classified for each genome (red circles) is monotonically decreasing. The y axis shows number of hashes (k-mers) classified to this genome (red circles) or total number of bases in the reference covered by mapped reads (blue stars); the numbers have not been rescaled. Decreases in mapping (green peaks) occur for genomes which are not exact matches to the genomes of the organisms used to build the mock community; for example, in plot (a), the peak at rank 33, *S. baltica OS185* represents reads that were preferentially mapped to *S. baltica OS223*, rank 8. [15,17].

Discussion

Below, we discuss the features and drawbacks of using FracMinHash and minimum metagenome covers to analyze metagenome datasets.

(CTB: probably want to talk a bit about long reads below, too.)

Scaled MinHash provides efficient containment queries for large data sets.

FracMinHash is a derivative of ModHash that uses the bottom hashing concept from MinHash to support containment operations. In brief, all elements in the set to be sketched are hashed, and any hash values below a certain fixed boundary value are kept for the sketch. This fixed boundary value is determined by the desired accuracy for the sketch operations.

Intuitively, FracMinHash performs a density sampling at a rate of 1 k-mer per s distinct k-mers seen, where s is the size of the hash space divided by the boundary value used in creating the sketch. This is a type of lossy compression, with a fixed compression ratio of s: for values of s used here ($s \approx 1000$), k-mer sets are reduced in cardinality by 1000-fold.

Unlike MinHash, FracMinHash supports containment estimation between sets of very different sizes, and here we demonstrate that it can be used efficiently and effectively for compositional analysis of shotgun metagenome data sets with k-mers. In particular, FracMinHash is competitive in accuracy with extant MinHash-based techniques for containment analysis, while also supporting Jaccard similarity.

We note that the FracMinHash technique has been used under a number of different names, including FracMinHash [23)] (cite Luiz thesis), universe minimizers [24], Shasta markers [25], and mincode syncmers [26]. The name FracMinHash was coined by Kristoffer Sahlin in an online discussion on Twitter [27] and chosen by discussants as the least ambiguous option. We use it here accordingly.

FracMinHash offers several conveniences over MinHash. No hash is ever removed from a FracMinHash_ sketch during construction; thus sketches grow proportionally to the number of distinct k-mers in the sampled data set, but *also* support many operations - including all of the operations used here - without needing to revisit the original data set. This is in contrast to MinHash, which requires auxiliary data structures for many operations - most especially, containment operations (cite CMash and mash screen). Thus FracMinHash sketches serve as compressed indices for the original content for a much broader range of operations than MinHash.

Because FracMinHash sketches collect all hash values below a fixed threshold, they also support streaming analysis of sketches: any operations that used a previously selected value can be cached and updated with newly arriving values. ModHash has similar properties, but this is not the case for MinHash: after n values are selected any displacement caused by new data can invalidate previous calculations.

FracMinHash also directly supports the addition and subtraction of hash values from a sketch, allowing post-processing and filtering without revisiting the original data set. This includes unions and intersections. Although possible for MinHash, in practice this requires oversampling (using a larger n) to account for possibly having less than n values after filtering, e.g. see the approach taken in Finch [28].

When the multiplicity of hashes in the original data is retained, FracMinHash sketches can be filtered on abundance. This allows removing low-abundance values, as implemented in Finch [28]. Filtering values that only appear once was implemented in Mash by using a Bloom Filter and only adding values after they were seen once, with later versions also implementing an extra counter array to keep track of counts for each value in the MinHash. These operations can be done in FracMinHash without auxiliary data structures.

Another useful operation available on FracMinHash sketches is *downsampling*: the contiguous value range for FracMinHash sketches allows deriving MinHash sketches from FracMinHash sketches whenever the number of hashes in the FracMinHash sketch is equal to or greater than n, as long as the same hashing scheme is used. Likewise, MinHash sketches can be converted to FracMinHash sketches when the maximum hash value in the MinHash sketch is larger than s.

Finally, because FracMinHash sketches are simply collections of hashes, any existing k-mer indexing approaches can be applied to sketches to support fast search with both similarity and containment estimators; several index types, including Sequence Bloom Trees and reverse indices, are provided in the sourmash software.

In exchange for these many conveniences, FracMinHash sketches have limited sensitivity for small data sets where the k-mer cardinality of the data set $\approx s$, and are only bounded in size by H/s (typically quite large, $\approx 2e16$). The limited sensitivity of sketches may affect the sensitivity of gene- and viral genome-sized queries, but at $s = 1000$ we see comparable accuracy and sketch size to MinHash for bacterial genome comparisons (Figure 1).

Notes from DK: do these belong in this section?

- The variance of the estimate of C(A,B)=|AB| / |A| appears to also depend on |A|, which was somewhat surprising
- The "fixed k-size" problem (which might be able to be overcome with the prefix-lookup data structure, if one sacrifices some accuracy)

Minimum set covers can be used for accurate compositional analysis of metagenomes.

Many metagenome content analysis approaches use reference genomes to interpret the metagenome content, but most such approaches rely on select a list of reduced-redundancy genomes from a much larger database (e.g. bioBakery 3 selects approximately 100,000 genomes [4]). Here, we do this reduction automatically by searching the complete database to retrieve a *minimum* set of reference genomes necessary to account for all k-mers shared between the metagenome and the database. We show that this can be resolved efficiently for real-world data sets; implementing a greedy min-set-cov approximation algorithm on top of FracMinHash, we provide an approach that readily scales to 700,000 genomes on current hardware (performance in appendix). We show that in practice this procedure reduces the number of genomes under consideration to ≈ 100 for several mock and real metagenomes.

The development of a small list of relevant genomes is particularly useful for large reference databases containing many redundant genomes; for example, in Table 1, we show that for one mock and one real community, we select minimum metagenome covers of 19 and 99 genomes for metagenomes that contain matches to 406k and 96k GenBank genomes total.

The min-set-cov approach for assigning genomes to metagenomes using k-mers differs substantially from extant k-mer and mapping-based approaches for identifying relevant genomes. LCA-based approaches such as Kraken label individual k-mers based on taxonomic lineages in a database, and then use the resulting database of annotated k-mers to assign taxonomy to reads. Mapping- and homology-based approaches such as Diamond use read mapping to genomes or read alignment to gene sequences in order to assign taxonomy and function (cite). These approaches typically focus on assigning *individual* k-mers or reads. In contrast, here we analyze the entire collection of k-mers and assign them *in aggregate* to the *best* genome match, and then repeat until no matches remain.

The resulting metagenome cover can then be used as inputs for further analysis, including both taxonomic content analysis and read mapping. For taxonomic analyses, we find that this approach is competitive with other current approaches and has several additional conveniences (discussed in detail below). The comparison of hash-based estimation of containment to mapping results in Figure 5 suggests that this approach is an accurate proxy for systematic mapping.

Our implementation of the min-set-cov algorithm in sourmash also readily supports custom reference databases as well as updating minimum set covers with the addition of new reference genomes. When updating set covers with new reference genomes, the first stage of calculating overlaps can be updated with the new genomes (Column 2 of Table 1), while the actual calculation of the minimum set cover must be redone each time.

Minimum set cover approaches may provide opportunities beyond those discussed here. For example, read- and contig-based analysis, and analysis and presentation of alignments, can be potentially simplified with this approach.

Minimum metagenome covers support accurate and flexible taxonomic assignment

We can build a taxonomic classifier on top of minimum set covers for metagenomes by reporting the taxonomies of the constituent genomes, weighted by distinct overlap and aggregated at the relevant taxonomic level using an LCA approach. Our CAMI-based taxonomic benchmarking shows that this approach is competitive for all metrics across all taxonomic levels (Figures 3 and 4).

One convenient feature of this approach to taxonomic analysis is that new or changed taxonomies can be readily incorporated by assigning them directly to genome identifiers; the majority of the computational work is involved in finding the reference genomes, which can have assignments in different taxonomic frameworks. For example, sourmash already supports GTDB [29] natively, and will also support the emerging LINS framework [30]. sourmash can also readily incorporate updates to taxonomies, e.g. the frequent updates to the NCBI taxonomy, without requiring expensive reanalysis of the primary metagenome data or the min-set-cov computation.

Interestingly, the framing of taxonomic classification as a minimum set cover problem may also avoid the loss of taxonomic resolution that affects k-mer- and read-based approaches on large databases [31]; this is because we apply LCA *after* reads and k-mers have been assigned to individual genomes, and choose entire *genomes* based on a greedy best-match-first approach. This minimizes the impact of individual k-mers that may be common to a genus or family, or were mis-assigned as a result of contamination.

Finally, as the underlying min-set-cov implementation supports custom databases, it is straightforward to support *taxonomic* analysis using custom databases and/or custom taxonomic assignments. sourmash natively supports this functionality.

The minimum set cover approach is reference dependent

The min-set-cov approach is reference-based, and hence is entirely dependent on the reference database. In particular, in many cases the exact reference strains present in the metagenome will not be present in the database. This manifests in two ways in Figure 5. First, there is a systematic mismatch between the hash content and the mapping content (green line), because mapping software is more permissive in the face of small variants than k-mer-based exact matching. Moreover, many of the lower rank genomes in the plot are from the same species but different *strains* as the

higher ranked genomes, suggesting that strain-specific portions of the reference are being utilized for matching at lower ranks. In reality, there will usually be a different mixture of strains in the metagenome than in the reference database. Methods for updating references from metagenome data sets may provide an opportunity for generating metagenome-specific references [32].

The approach presented here chooses arbitrarily between matches with equivalent numbers of contained k-mers. There are specific genomic circumstances where this approach could usefully be refined with additional criteria. For example, if a phage genome is present in the reference database, and is also present within one or more genomes in the database, it may desirable to select the match with the highest Jaccard *similarity* in order to select the phage genome directly. This is algorithmically straightforward to implement.

In light of the strong reference dependence of the min-set-cov approach together with the insensitivity of the FracMinHash technique, it may be useful to explore alternate methods of summarizing the list of overlapping genomes, that is, *all* the genomes in column 2 of Table 1. For example, a hierarchical approach could be taken to first identify the full list of overlapping genomes, followed by a higher resolution approach to identify a specific subset of matching genomes.

Opportunities for future improvement of min-set-cov

There are a number of immediate opportunities for future improvement of the min-set-cov approach.

Implementing min-set-cov on top of FracMinHash means that there is a loss of resolution when choosing between very closely related genomes, because the set of hashes chosen may not discriminate between them. Likewise, the potentially very large size of the sketches may inhibit the application of this approach to very large metagenomes.

These limitations are not intrinsic to min-set-cov, however; any data structure supporting both the *containment* $C(A, B) = \frac{|A \cap B|}{|A|}$ and *remove elements* operations can be used to implement the greedy approximation algorithm (ref algorithm in results section 1). For example, a simple *set* of the k-mer composition of the query supports element removal, and calculating containment can be done with regular set operations. Approximate membership query (AMQ) sketches like the *Counting Quotient Filter* [33] can also be used, with the benefit of reduced storage and memory usage. Moreover, the collection of datasets can be implemented with any data structure that can do containment comparisons with the query data structure.

In turn, this means that limitations of our current implementation, such as insensitivity to small genomes when s is approximately the same as the genome size, may be readily solvable with other sketch types.

There are other opportunities for improving on these initial explorations. The availability of abundance counts for each element in the FracMinHash is not well explored, since the process of *removing elements* from the query does not use them . Both the multiple match as well as the abundance counts issues can benefit from existing solutions taken by other methods, like the *species score* (for disambiguation) and *Expectation-Maximization* (for abundance analysis) approaches from Centrifuge [**kim_centrifuge_2016?**].

Conclusion

The FracMinHash and min-set-cov approaches explored here provide powerful and accurate techniques for analyzing metagenomes, with well defined limitations. The immediate applications for both mapping-based and taxonomic analysis of metagenomes are already interesting. We provide an implementation of these approaches in robust open-source software, together with workflows to enable their practical use on large data sets. The approaches also offer many opportunities for further exploration and improvement with additional sketch types, additional approximation algorithms, and additional summarization approaches. The world is our oyster!

Methods

TBD

References

1. **On the resemblance and containment of documents**
AZ Broder
Institute of Electrical and Electronics Engineers (IEEE) (2002-11-22) https://doi.org/fqk7hr
DOI: 10.1109/sequen.1997.666900

2. **The Integrative Human Microbiome Project**
The Integrative HMP (iHMP) Research Network Consortium
Nature (2019-05-29) https://doi.org/gf3wp9
DOI: 10.1038/s41586-019-1238-8 · PMID: 31142853 · PMCID: PMC6784865

3. **Priorities for the next 10 years of human microbiome research**
Lita Proctor
Nature (2019-05-29) https://doi.org/gnnprk
DOI: 10.1038/d41586-019-01654-0 · PMID: 31142863

4. **Integrating taxonomic, functional, and strain-level profiling of diverse microbial communities with bioBakery 3**
Francesco Beghini, Lauren J McIver, Aitor Blanco-Míguez, Leonard Dubois, Francesco Asnicar, Sagun Maharjan, Ana Mailyan, Paolo Manghi, Matthias Scholz, Andrew Maltez Thomas, … Nicola Segata
eLife (2021-05-04) https://doi.org/gkc38n
DOI: 10.7554/elife.65088 · PMID: 33944776 · PMCID: PMC8096432

5. **MEGAN-LR: new algorithms allow accurate binning and easy interactive exploration of metagenomic long reads and contigs**
Daniel H Huson, Benjamin Albrecht, Caner Bağcı, Irina Bessarab, Anna Górska, Dino Jolic, Rohan BH Williams
Biology Direct (2018-04-20) https://doi.org/gnnprp
DOI: 10.1186/s13062-018-0208-7 · PMID: 29678199 · PMCID: PMC5910613

6. **Fast and sensitive taxonomic assignment to metagenomic contigs**
M Mirdita, M Steinegger, F Breitwieser, J Söding, E Levy Karin
Bioinformatics (2021-09-15) https://doi.org/gnnprm
DOI: 10.1093/bioinformatics/btab184 · PMID: 33734313 · PMCID: PMC8479651

7. **eggNOG 5.0: a hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses**
Jaime Huerta-Cepas, Damian Szklarczyk, Davide Heller, Ana Hernández-Plaza, Sofia K Forslund, Helen Cook, Daniel R Mende, Ivica Letunic, Thomas Rattei, Lars J Jensen, … Peer Bork
Nucleic Acids Research (2019-01-08) https://doi.org/gg8bdg
DOI: 10.1093/nar/gky1085 · PMID: 30418610 · PMCID: PMC6324079

8. **Improved metagenomic analysis with Kraken 2**
Derrick E Wood, Jennifer Lu, Ben Langmead
Genome Biology (2019-11-28) https://doi.org/ggfk55
DOI: 10.1186/s13059-019-1891-0 · PMID: 31779668 · PMCID: PMC6883579

9. **Fast and sensitive taxonomic classification for metagenomics with Kaiju**
Peter Menzel, Kim Lee Ng, Anders Krogh
Nature Communications (2016-04-13) https://doi.org/f8h4b6
DOI: 10.1038/ncomms11257 · PMID: 27071849 · PMCID: PMC4833860

10. **IMPROVING MIN HASH VIA THE CONTAINMENT INDEX WITH APPLICATIONS TO METAGENOMIC ANALYSIS**
David Koslicki, Hooman Zabeti
Cold Spring Harbor Laboratory (2017-09-04) https://doi.org/ghvn6z
DOI: 10.1101/184150

11. **Mash Screen: high-throughput sequence containment estimation for genome discovery**
Brian D Ondov, Gabriel J Starrett, Anna Sappington, Aleksandra Kostic, Sergey Koren, Christopher B Buck, Adam M Phillippy
Genome Biology (2019-11-05) https://doi.org/ghtqmb
DOI: 10.1186/s13059-019-1841-x · PMID: 31690338 · PMCID: PMC6833257

12. **Comparative metagenomic and rRNA microbial diversity characterization using archaeal and bacterial synthetic communities**
Migun Shakya, Christopher Quince, James H Campbell, Zamin K Yang, Christopher W Schadt, Mircea Podar
Environmental Microbiology (2013-06) https://doi.org/f42ccr
DOI: 10.1111/1462-2920.12086 · PMID: 23387867 · PMCID: PMC3665634

13. **Omega: an Overlap-graph de novo Assembler for Metagenomics**
Bahlul Haider, Tae-Hyuk Ahn, Brian Bushnell, Juanjuan Chai, Alex Copeland, Chongle Pan
Bioinformatics (2014-10) https://doi.org/f6kt42
DOI: 10.1093/bioinformatics/btu395 · PMID: 24947750

14. **metaSPAdes: a new versatile metagenomic assembler**
Sergey Nurk, Dmitry Meleshko, Anton Korobeynikov, Pavel A Pevzner
Genome Research (2017-05) https://doi.org/f97jkv
DOI: 10.1101/gr.213959.116 · PMID: 28298430 · PMCID: PMC5411777

15. **Evaluating Metagenome Assembly on a Simple Defined Community with Many Strain Variants**
Sherine Awad, Luiz Irber, CTitus Brown
Cold Spring Harbor Laboratory (2017-07-03) https://doi.org/ghvn6x
DOI: 10.1101/155358

16. **Letter-Value Plots: Boxplots for Large Data**
Heike Hofmann, Hadley Wickham, Karen Kafadar
Journal of Computational and Graphical Statistics (2017-07-11) https://doi.org/gf38v7
DOI: 10.1080/10618600.2017.1305277

17. **Mash: fast genome and metagenome distance estimation using MinHash**
Brian D Ondov, Todd J Treangen, Páll Melsted, Adam B Mallonee, Nicholas H Bergman, Sergey Koren, Adam M Phillippy
Genome Biology (2016-06-20) https://doi.org/gfx74q
DOI: 10.1186/s13059-016-0997-x · PMID: 27323842 · PMCID: PMC4915045

18. **Greedy Set-Cover Algorithms**
Neal E Young
Springer Science and Business Media LLC (2008) https://doi.org/fjhztp
DOI: 10.1007/978-0-387-30162-4_175

19. **Critical Assessment of Metagenome Interpretation—a benchmark of metagenomics software**
Alexander Sczyrba, Peter Hofmann, Peter Belmann, David Koslicki, Stefan Janssen, Johannes Dröge, Ivan Gregor, Stephan Majda, Jessika Fiedler, Eik Dahms, … Alice C McHardy

Nature Methods (2017-10-02) https://doi.org/gbzspt
DOI: 10.1038/nmeth.4458 · PMID: 28967888 · PMCID: PMC5903868

20. **Tutorial: assessing metagenomics software with the CAMI benchmarking toolkit**
Fernando Meyer, Till-Robin Lesker, David Koslicki, Adrian Fritz, Alexey Gurevich, Aaron E Darling, Alexander Sczyrba, Andreas Bremges, Alice C McHardy
Nature Protocols (2021-03-01) https://doi.org/gh77rh
DOI: 10.1038/s41596-020-00480-3 · PMID: 33649565

21. **ZymoBIOMICS Microbial Community Standards**
ZYMO RESEARCH
https://www.zymoresearch.com/collections/zymobiomics-microbial-community-standards

22. **Genome-Resolved Metagenomic Analysis Reveals Roles for Candidate Phyla and Other Microbial Community Members in Biogeochemical Transformations in Oil Reservoirs**
Ping Hu, Lauren Tom, Andrea Singh, Brian C Thomas, Brett J Baker, Yvette M Piceno, Gary L Andersen, Jillian F Banfield
mBio (2016-03-02) https://doi.org/f8j5xr
DOI: 10.1128/mbio.01669-15 · PMID: 26787827 · PMCID: PMC4725000

23. **Large-scale sequence comparisons with sourmash**
NTessa Pierce, Luiz Irber, Taylor Reiter, Phillip Brooks, CTitus Brown
F1000Research (2019-07-04) https://doi.org/gf9v84
DOI: 10.12688/f1000research.19675.1 · PMID: 31508216 · PMCID: PMC6720031

24. **Minimizer-space de Bruijn graphs: Whole-genome assembly of long reads in minutes on a personal computer**
Barış Ekim, Bonnie Berger, Rayan Chikhi
Cell Systems (2021-10) https://doi.org/gmtsjc
DOI: 10.1016/j.cels.2021.08.009 · PMID: 34525345 · PMCID: PMC8562525

25. **Nanopore sequencing and the Shasta toolkit enable efficient de novo assembly of eleven human genomes**
Kishwar Shafin, Trevor Pesout, Ryan Lorig-Roach, Marina Haukness, Hugh E Olsen, Colleen Bosworth, Joel Armstrong, Kristof Tigyi, Nicholas Maurer, Sergey Koren, … Benedict Paten
Nature Biotechnology (2020-05-04) https://doi.org/ggvsn4
DOI: 10.1038/s41587-020-0503-6 · PMID: 32686750 · PMCID: PMC7483855

26. **Syncmers are more sensitive than minimizers for selecting conserved k -mers in biological sequences**
Robert Edgar
PeerJ (2021-02-05) https://doi.org/gm7pzp
DOI: 10.7717/peerj.10805 · PMID: 33604186 · PMCID: PMC7869670

27. **https://twitter.com/krsahlin/status/1463169988689285125**
Twitter
https://twitter.com/krsahlin/status/1463169988689285125

28. **Finch: a tool adding dynamic abundance filtering to genomic MinHashing**
Roderick Bovee, Nick Greenfield
The Journal of Open Source Software (2018-02-01) https://doi.org/gm85dx
DOI: 10.21105/joss.00505

29. **GTDB: an ongoing census of bacterial and archaeal diversity through a phylogenetically consistent, rank normalized and complete genome-based taxonomy**

Donovan H Parks, Maria Chuvochina, Christian Rinke, Aaron J Mussig, Pierre-Alain Chaumeil, Philip Hugenholtz
Nucleic Acids Research (2021-09-14) https://doi.org/gm97d8
DOI: 10.1093/nar/gkab776 · PMID: 34520557

30. **A Proposal for a Genome Similarity-Based Taxonomy for Plant-Pathogenic Bacteria that Is Sufficiently Precise to Reflect Phylogeny, Host Range, and Outbreak Affiliation Applied to Pseudomonas syringae sensu lato as a Proof of Concept**
Boris A Vinatzer, Alexandra J Weisberg, Caroline L Monteil, Haitham A Elmarakeby, Samuel K Sheppard, Lenwood S Heath
Phytopathology® (2017-01) https://doi.org/gg78hd
DOI: 10.1094/phyto-07-16-0252-r · PMID: 27552324

31. **RefSeq database growth influences the accuracy of k-mer-based lowest common ancestor species identification**
Daniel J Nasko, Sergey Koren, Adam M Phillippy, Todd J Treangen
Genome Biology (2018-10-30) https://doi.org/ggc9db
DOI: 10.1186/s13059-018-1554-6 · PMID: 30373669 · PMCID: PMC6206640

32. **Exploring neighborhoods in large metagenome assembly graphs using spacegraphcats reveals hidden sequence diversity**
CTitus Brown, Dominik Moritz, Michael P O'Brien, Felix Reidl, Taylor Reiter, Blair D Sullivan
Genome Biology (2020-07-06) https://doi.org/d4bb
DOI: 10.1186/s13059-020-02066-4 · PMID: 32631445 · PMCID: PMC7336657

33. **A General-Purpose Counting Filter**
Prashant Pandey, Michael A Bender, Rob Johnson, Rob Patro
Association for Computing Machinery (ACM) (2017-05-09) https://doi.org/gg29n9
DOI: 10.1145/3035918.3035963

Scaled MinHash sketches support efficient indexing for large-scale containment queries

CTB: Additional points to raise:

- in-memory representation of sketches may be too big (!!), goal here is on disk storage/low minimum memory for "extremely large data" situation.
- Also/in addition, want ability to do incremental loading of things.
- Note we are not talking here about situations where the indices themselves are too big to download.
- I think rename LCA to revindex. Or make up a new name.

We provide two index data structures for rapid estimation of containment in large databases. The first, the MinHash Bloom Tree (MHBT), is a specialization of the Sequence Bloom Tree [**solomon_fast_2016?**], and implements a k-mer aggregative method with explicit representation of datasets based on hierarchical indices. The second is LCA, an inverted index into sketches, a color-aggregative method with implicit representation of the sketches.

We evaluated the MHBT and LCA databases by constructing and searching a GenBank snapshot from July 18, 2020, containing 725,331 assembled genomes (5,282 Archaea, 673,414 Bacteria, 6,601 Fungi 933 Protozoa and 39,101 Viral). MHBT indices were built with $scaled = 1000$, and LCA indices used $scaled = 10000$. Table 2 shows the indexing results for the LCA index, and Table 3 for the MHBT index.

Table 2: Results for LCA indexing, with $scaled = 10000$ and $k = 21$.

Domain	Runtime (s)	Memory (MB)	Size (MB)
Viral	57	33	2
Archaea	58	30	5
Protozoa	231	3	17
Fungi	999	3	65
Bacteria	12,717	857	446

Table 3: Results for MHBT indexing, with $scaled = 1000$, $k = 21$ and internal nodes (Bloom Filters) using 10000 slots for storage.

Domain	Runtime (s)	Memory (MB)	Size (MB)
Viral	126	326	77
Archaea	111	217	100
Protozoa	206	753	302
Fungi	1,161	3,364	1,585
Bacteria	32,576	47,445	24,639

Index sizes are more affected by the number of genomes inserted than the individual *Scaled MinHash* sizes. Despite Protozoan and Fungal *Scaled MinHash* sketches being larger individually, the Bacterial indices are an order of magnitude larger for both indices since they contain two orders of magnitude more genomes.

Comparing between LCA and MHBT index sizes must account for their different scaled parameters, but as shown in Chapter 1 a *Scaled MinHash* with $scaled = 1000$ when downsampled to $scaled = 10000$ is expected to be ten times smaller. Even so, MHBT indices are more than ten times larger than their LCA counterparts, since they store extra caching information (the internal nodes) to avoid loading all the data to memory during search. LCA indices also contain extra data (the list of datasets containing a hash), but this is lower than the storage requirements for the MHBT internal nodes.

We next executed similarity searches on each database using appropriate queries for each domain. All queries were selected from the relevant domain and queried against both MHBT ($scaled = 1000$) and LCA ($scaled = 10000$), for $k = 21$.

Table 4: Running time in seconds for similarity search using LCA ($scaled = 10000$) and MHBT ($scaled = 1000$) indices.

	Viral	Archaea	Protozoa	Fungi	Bacteria
LCA	1.06	1.42	5.40	26.92	231.26
SBT	1.32	3.77	43.51	244.77	3,185.88

Table 5: Memory consumption in megabytes for similarity search using LCA ($scaled = 10000$) and MHBT ($scaled = 1000$) indices.

	Viral	Archaea	Protozoa	Fungi	Bacteria
LCA	223	240	798	3,274	20,926

	Viral	Archaea	Protozoa	Fungi	Bacteria
SBT	163	125	332	1,656	2,290

Table 4 shows running time for both indices. For small indices (Viral and Archaea) the LCA running time is dominated by loading the index in memory, but for larger indices the cost is amortized due to the faster running times. This situation is clearer for the Bacteria indices, where the LCA search completes in 3 minutes and 51 seconds, while the SBT search takes 54 minutes.

When comparing memory consumption, the situation is reversed. Table 5 shows how the LCA index consistently uses twice the memory for all domains, but for larger indices like Bacteria it uses as much as 10 times the memory as the MHBT index for the same data.

For both runtime and memory consumption, it is worth pointing that the LCA index is a tenth of the data indexed by the MHBT. This highlights the trade-off between speed and memory consumption for both approaches, especially for larger indices.

Notes: * new genomes can be added quickly to SBT.

---

1. In our current implementation in `sourmash`, when equivalent matches are available for a given rank, a match is chosen at random. This is an implementation decision that is not intrinsic to the algorithm itself.↩