Lightweight compositional analysis of metagenomes with FracMinHash and minimum metagenome covers

This manuscript (permalink) was automatically generated from dib-lab/2020-paper-sourmash-gather@b2b79b8 on January 17, 2022.

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 GBMF4551 from the Gordon and Betty Moore Foundation; Grant R01HG007513 from the NIH NHGRI

- **Phillip T. Brooks**
  0000-0003-3987-244X · brooksph · brooksph
  Department of Population Health and Reproduction, UC Davis · Funded by Grant GBMF4551 from the Gordon and Betty Moore Foundation

- **Taylor Reiter**
  0000-0002-7388-421X · taylorreiter · ReiterTaylor
  Graduate Group in Food Science, UC Davis; Department of Population Health and Reproduction, UC Davis · Funded by Grant GBMF4551 from the Gordon and Betty Moore Foundation; Grant R03OD030596 from the NIH Common Fund

- **N. Tessa Pierce-Ward**
  0000-0002-2942-5331 · bluegenes · saltyscientist
  Department of Population Health and Reproduction, UC Davis · Funded by Grant 1711984 from the NSF; Grant GBMF4551 from the Gordon and Betty Moore Foundation; Grant 2018911 from the NSF

- **Mahmudur Rahman Hera**
  · mahmudhera · HeraMahmudur
  Department of Computer Science and Engineering, Penn State University · Funded by Grant 2029170 from the NSF

- **David Koslicki**
  0000-0002-0640-954X · dkoslicki · DavidKoslicki
  Department of Computer Science and Engineering, Penn State University; Department of Biology, Penn State University; Huck Institutes of the Life Sciences, Penn State University · Funded by Grant 2029170 from the NSF

- **C. Titus Brown**
  0000-0001-6001-2677 · ctb
  Department of Population Health and Reproduction, UC Davis · Funded by Grant GBMF4551 from the Gordon and Betty Moore Foundation; Grant R01HG007513 from the NIH NHGRI; Grant 2018911 from the NSF; Grant R03OD030596 from the NIH Common Fund

Abstract

The identification of reference genomes and taxonomic labels from 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 sizes. We implement FracMinHash in the sourmash software, evaluate its accuracy, and demonstrate large-scale containment searches of metagenomes using 700,000 microbial reference genomes. We next frame shotgun metagenome compositional analysis as the problem of finding a minimum collection of reference genomes that "cover" the known k-mers in a metagenome, a minimum set cover problem. We implement a greedy approximate solution using FracMinHash sketches, and evaluate its accuracy for taxonomic assignment using a CAMI community benchmark. Finally, we show that the minimum metagenome cover can be used to guide the selection of reference genomes for read mapping. sourmash is available as open source software 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 [1,2]. By sampling the genomic content of microbial communities, shotgun metagenomics enables the taxonomic and functional characterization of microbiomes [3,4]. However, this characterization relies critically on the methods and databases used to interpret the sequencing data [5,6,7,8].

Metagenome function and taxonomy are typically inferred from available reference genomes and gene catalogs, via direct genomic alignment [9,10], large-scale protein search [11,12,13], or k-mer matches [14,15]. For many of these methods, the substantial increase in the number of available microbial reference genomes (1.1m in GenBank as of November 2021) presents a significant practical obstacle to comprehensive compositional analyses. Most methods choose representative subsets of available genomic information to analyze; for example, bioBakery 3 provides a database containing 99.2k reference genomes [9]. Scaling metagenome analysis approaches to make use of the rapidly increasing size of GenBank is an active endeavor in the field [16,17].

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 - a "minimum metagenome cover". We use a mod-hash based sketching approach for k-mers to reduce memory requirements [18], and implement a polynomial-time greedy approximation algorithm for the minimum set cover analysis [19].

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 `sourmash` open source software package 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 and overlap estimation for DNA sequencing 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 metagenome cover as reference genomes for read mapping.

FracMinHash sketches support accurate containment operations

We define the *fractional MinHash*, or FracMinHash as follows: for a hash function $h : \Omega \rightarrow [O, H]$, on an input set of hash values $W \subseteq \Omega$ and for any $0 <= s <= H$,

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

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

The FracMinHash is a mix of MinHash and ModHash [18,20]. 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.

Like ModHash (but not MinHash), FracMinHash supports estimation of the containment index:

$$\hat{C}_{\text{frac}}(A, B) := \frac{|\mathbf{FRAC}_S(A) \cap \mathbf{FRAC}_S(B)|}{|\mathbf{FRAC}_S(A)|}.$$

See Methods for details.

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. Beyond accurate containment operations, FracMinHash can be used to estimate evolutionary distance between pairs of sequences undergoing a mutation model, similar to but more accurately than the MinHash derived method in [20]. See [21] for these and other analytical details.

A FracMinHash implementation accurately estimates containment between sets of different sizes

We compare the FracMinHash method, implemented in the sourmash software [22], to *Containment MinHash* [23] and Mash Screen (*Containment Score*) [24] for containment queries in data from the `podar mock` community, a mock bacterial and archaeal community where the reference genomes are largely known [25]; see also Table 1, row 2. This data set has been used in several methods evaluations [24,26,27,28].

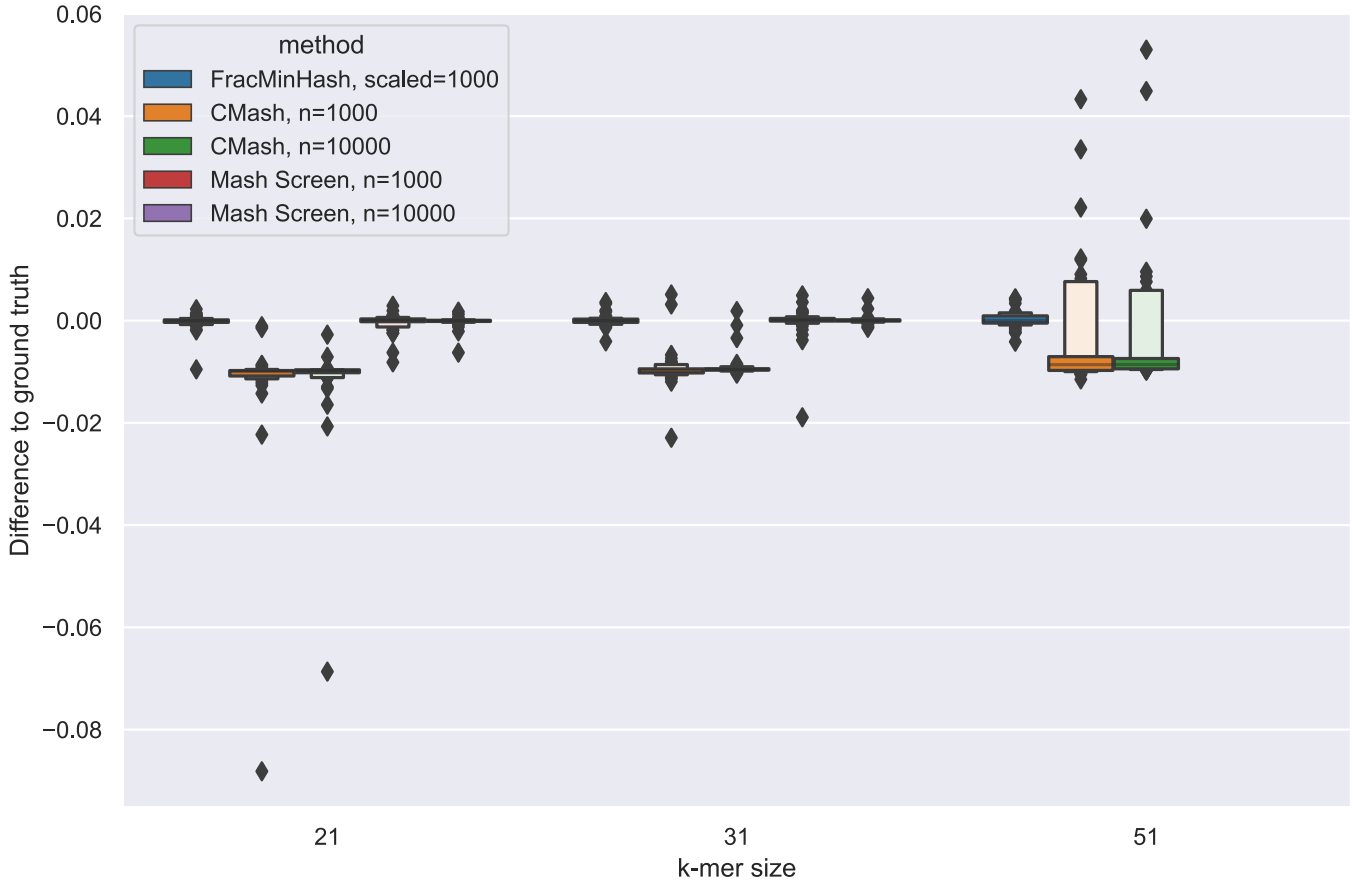

Figure 1: Letter-value plot [29] of the differences from containment estimate to ground truth (exact). Each method is evaluated for $k = \{21, 31, 51\}$, except for `Mash` with $k = 51$, which is unsupported.

Figure 1 shows containment analysis of genomes in this metagenome, with low-coverage and contaminant genomes (as described in [28] and [24]) removed from the database. All methods are within 1% of the exact containment on average (Figure 1), with `CMash` consistently underestimating the containment. `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 $n = 1000$. The sourmash sketch sizes varied between 431 hashes and 9540 hashes, with a median of 2741 hashes.

FracMinHash can be used 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? We wish to find the smallest set $\{G_n\}$ of genomes in D such that, for the k-mer decomposition $k()$,

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

This is a *minimum set covering* problem, for which there is a polynomial-time approximation [19]:

1. Initialize $C \leftarrow \emptyset$
2. While $k(M) \cap k(D) \setminus \bigcup_{G \in C}(k(M) \cap k(G))$ is nonempty:

3. $C \leftarrow C \bigcup \left\{ \arg\max_{G \in D} |k(G) \cup (k(M) \cap k(D) \setminus \bigcup_{G \in C}(k(M) \cup k(G)))| \right\}$
4. return C

This greedy algorithm iteratively chooses reference genomes from D in order of largest remaining overlap with M, where overlap is in terms of number of k-mers. This results in a progressive classification of the known k-mers in the metagenome to specific genomes.[1] Note it is classically known that this greedy heuristic results in a logarithmic approximation factor to the optimal set cover solution [19]. This algorithm is implemented as sourmash `gather`.

In Figure 2, we show an example of this progressive classification of 31-mers by matching GenBank genome for `podar mock`. The matching genomes are provided in the order found by the greedy algorithm, i.e. by overlap with remaining k-mers in the metagenome. The high rank (early) matches reflect large and/or mostly-covered genomes with high containment, while later matches reflect genomes that share fewer k-mers with the remaining set of k-mers in the metagenome - smaller genomes, less-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 find lower rank matches.

As one example of metagenome k-mers shared with multiple matches, genomes from two strains of *Shewanella baltica* are present in the mock metagenome. These genomes overlap in k-mer content by approximately 50%, and these shared k-mers are first claimed by *Shewanella baltica* OS223 – compare *S. baltica* OS223, rank 8, with *S. baltica* OS185, rank 33 in Figure 2. Here the difference between the green triangles (all matched k-mers) and red circles (min-set-cov matched k-mers) for *S. baltica* OS185 represents the k-mers claimed by *S. baltica* OS223 .

For this mock metagenome, 205m (54.8%) of 375m k-mers were found in GenBank (see Table 1, row 2). The remaining 169m (45.2%) k-mers had no matches, and represent either k-mers introduced by sequencing errors or k-mers from real but unknown community members.

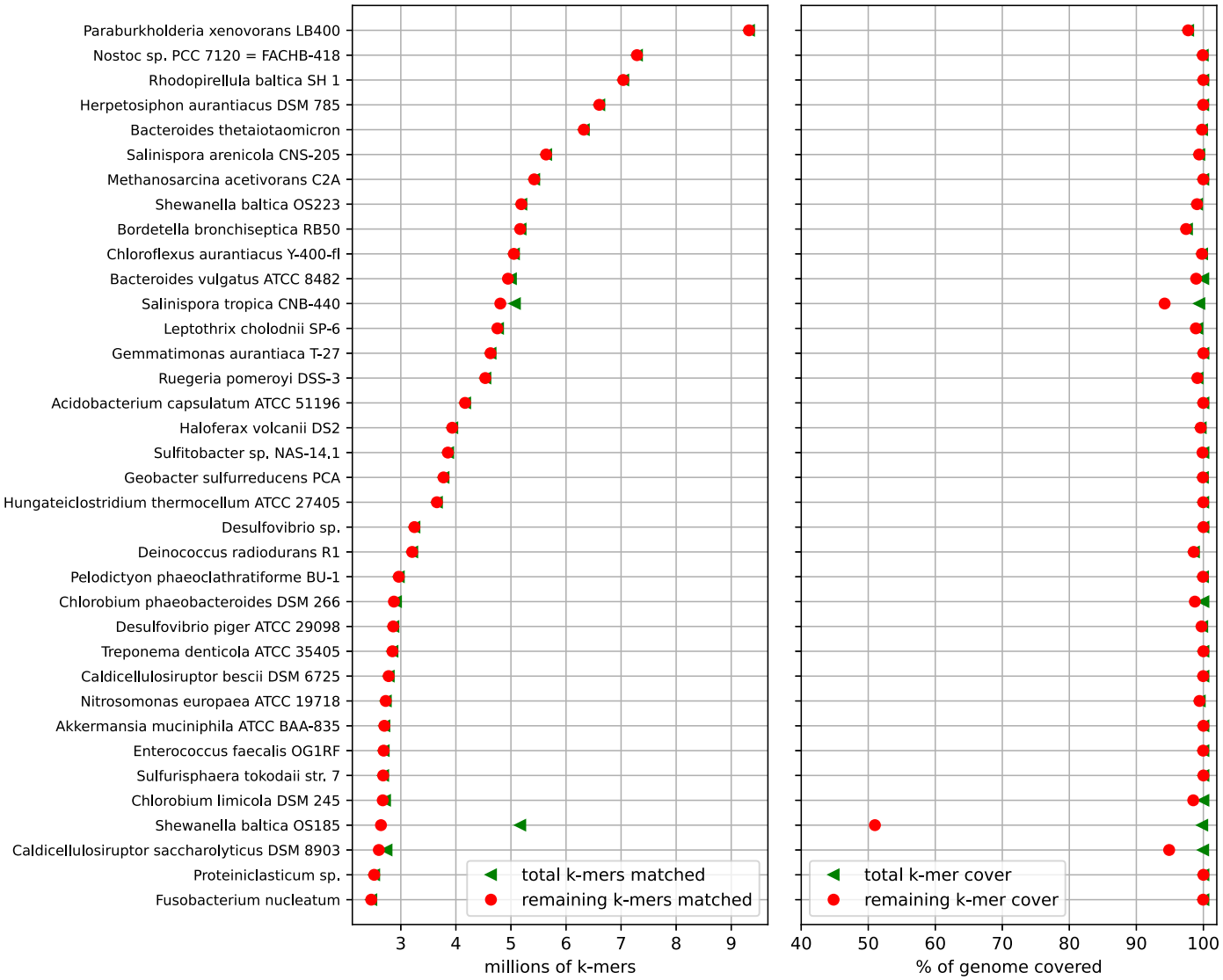

Figure 2: K-mer decomposition of a metagenome into constituent genomes. A rank ordering by remaining containment for the first 36 genomes from the minimum metagenome cover of the `podar mock` synthetic metagenome [25], calculated using 700,000 genomes from GenBank with scaled=2000, k=31. The Y axis is labeled with the NCBI-designated 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 triangles indicate overlap between the genome and the entire 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), a community-driven initiative for reproducibly benchmarking metagenomic methods [30]. We used the mouse gut metagenome dataset [31], 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. Sixty-four samples were generated with *CAMISIM*, with 91.8 genomes present in 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.

Since min-set-cov yields only a collection of genomes, this collection must be converted into a taxonomy with relative abundances for benchmarking with CAMI. We developed the following

procedure for generating a taxonomic profile from a given metagenome cover. For each genome match, we note the species designation in the NCBI taxonomy for that genome. Then, we calculate the fraction of the genome remaining in the metagenome after k-mers belonging to higher-rank genomes have been removed (i.e. red circles in Figure 2 (a)). We multiply this fraction by the median abundance of the hashes in the sketch to weight the contribution of the genome's species designation to the metagenome taxonomy. This procedure produces an estimate of that species' taxonomic contribution to the metagenome, normalized by the genome size.

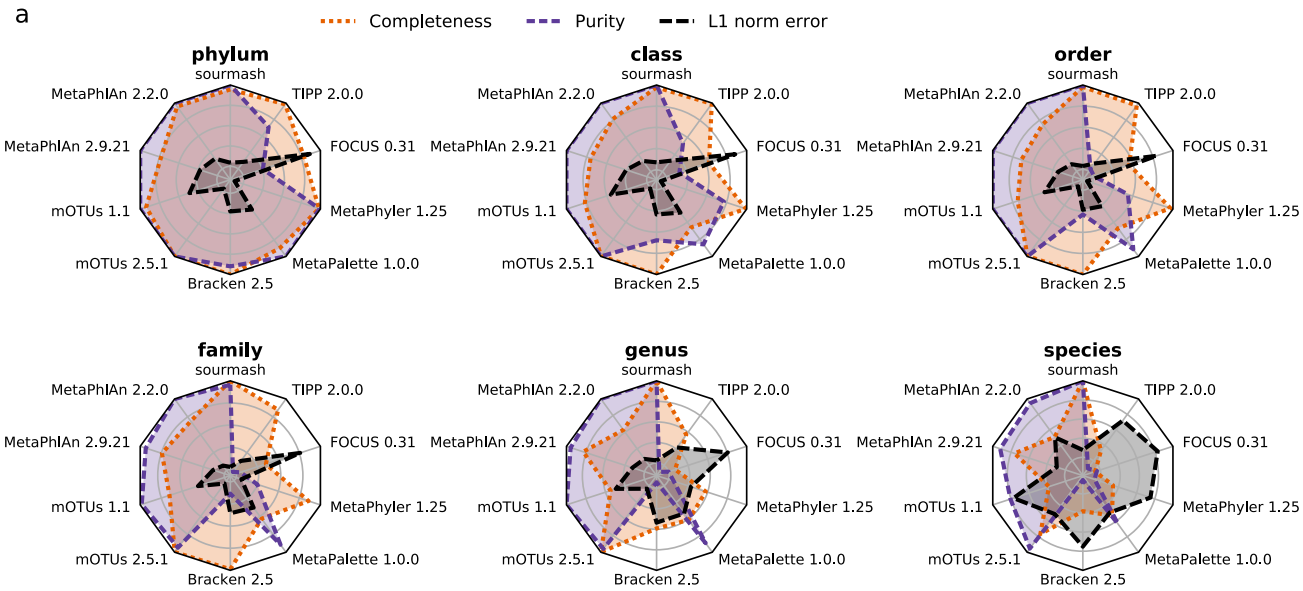

Figure 3: Comparison per taxonomic rank of methods in terms of completeness, purity (1% filtered), and L1 norm.

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. A scaled value of 2000 and a k-mer size of 31 was used.

In Figures 3 and 4 we show an updated version of Figure 6 from [31] that includes our method, implemented in the `sourmash` software and benchmarked using OPAL [32]. The minimum metagenome cover was calculated against the Jan 8, 2019 snapshot of RefSeq provided by the CAMI project. 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 second 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 the number of genomes in the estimated minimum metagenome cover from GenBank, with scaled=2000 and k=31. Overlap and % 31-mers identified are estimated from FracMinHash sketch size.

data set	genomes >= 100k 31-mer overlap	size of min-set-cov	% 31-mers identified
`zymo mock`	405,839	19	47.1%
`podar mock`	5,800	74	54.8%
`gut real`	96,423	99	36.0%
`oil well real`	1,235	135	14.9%

In Table 1, we show the minimum metagenome cover for four metagenomes against GenBank - two mock communities [25,33], a human gut microbiome data set from iHMP [3], and an oil well sample [34]. Our implementation provides estimates for both the *total* number of genomes with substantial overlap to a query genome, and the minimum set of genomes that account for k-mers with overlap in the query metagenome. Note that only matches estimated to have more than 100,000 overlapping k-mers are shown (see Methods for details).

We find many genomes with 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 precisely 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 [16,17]. One specific use for a minimum metagenome cover could be 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 metagenome reads to the first ranked genome in the minimum metagenome cover, and then remove successfully mapped reads from the metagenome. Remaining unmapped reads are then mapped to the second rank genome, and this then continues until all genomes have been used. 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. This approach is implemented in the automated workflow package `genome-grist`; see Methods for details.

Figure 5 compares k-mer 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 agree closely with the number of bases covered by mapped reads: the Y axis has not been re-scaled, so k-mer matches and read mapping coverage correspond well. This suggests that the k-mer-based min-set-cov approach effectively selects reference genomes for metagenome read mapping.

For mock metagenomes (Figure 5 (A) and (B)), there is a close correspondence between mapping and k-mer coverage, while for real metagenomes (Figure 5 (C) and (D)), mapping coverage tends to be higher. This may be because the mock metagenomes are largely constructed from strains with known genomes, so most 31-mers match exactly, while the gut and oil well metagenomes contain a number of strains where only species (and not strain) genomes are present in the database, and so mapping performs better. Further work is needed to evaluate rates of variation across a larger number of metagenomes.

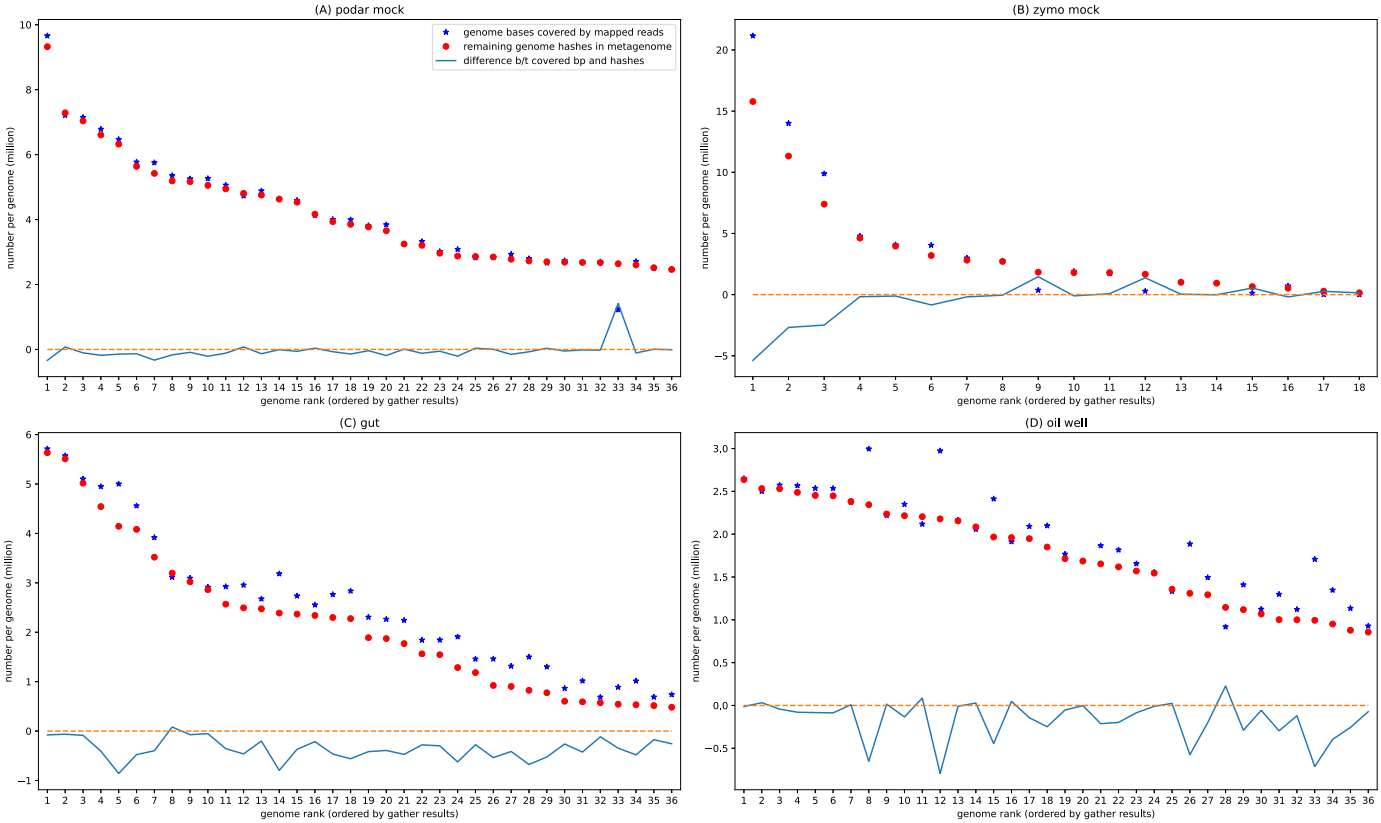

Figure 5: Hash-based k-mer decomposition of a metagenome into constituent genomes compares well to bases covered by read mapping. Plots for each of four metagenomes showing estimated k-mer overlap per genome, along with bases covered by read mapping, for the first 36 genomes in the minimum metagenome cover. The reference genomes are rank ordered along the X axis (as in the Y axis for 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 estimated number of 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 (peaks in blue lines) occur for genomes which are not exact matches to the genomes of the organisms used to build the mock community; for example, in (A), the peak at rank 33 of `podar mock` is for *S. baltica OS185*, and represents reads that were preferentially mapped to *S. baltica OS223*, rank 8.

Discussion

Below, we discuss the use of FracMinHash and minimum metagenome covers to analyze metagenome datasets.

FracMinHash 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: 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, with clear space/time constraint tradeoffs.

Intuitively, FracMinHash can be viewed as performing density sampling at a rate of 1 k-mer per s distinct k-mers seen, where s is used to define a boundary value $\frac{H}{s}$ for the bottom sketch. FracMinHash can also be viewed as 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. In addition, FracMinHash can be used to obtain point estimates of and confidence intervals around mutation rates and evolutionary distances; see [21] for these and other analytical results.

We note that the FracMinHash technique has been used under a number of different names, including Scaled MinHash [35,36], universe minimizers [37], Shasta markers [38], and mincode syncmers [39]. The name FracMinHash was coined by Kristoffer Sahlin in an online discussion on Twitter [40] 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 [23,24]. 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 for limited types of 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 fewer than n values after filtering, e.g. see the approach taken in Finch [41].

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 [41]. Filtering

values that only appear once was implemented in Mash by using a Bloom filter and only adding values after they were seen once; later versions also implemented 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 means that MinHash sketches can be extracted from FracMinHash sketches whenever the size of the requested MinHash is less than the size of the FracMinHash sketch. Likewise, MinHash sketches can be losslessly converted to FracMinHash sketches when the maximum hash value in the MinHash sketch is larger than H/s.

Finally, because FracMinHash sketches are simply collections of hashes, existing hash-based 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 [42] 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, which is 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).

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 starting with a list of reduced-redundancy genomes from a much larger database (e.g. bioBakery 3 selects approximately 100,000 genomes [9]), which can reduce sensitivity and precision [17]. Here, we incorporate this reduction into the overall workflow by searching the complete database for 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. 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 [43]. 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 minimum metagenome cover can then be used as part of further analyses, 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, as also seen in Metalign [17].

There is one significant drawback to assigning minimum metagenome covers based on k-mers: because k-mers are not a perfect proxy for mapping (e.g. see Figure 5, blue lines), using k-mers to identify the best genome for *mapping* may sometimes lead to inaccurate assignments. Note that long k-mers are generally more stringent and specific than mapping, so e.g. 51-mer overlaps can be used to identify *some* candidate genomes for mapping, but not *all* candidate genomes will necessarily be found using 51-mer overlaps. The extent and impact of this kind of false negative in the min-set-cov approach remains to be evaluated but is likely to only affect strain- and species-level assignments, since nucleotide similarity measures lose sensitivity across more distant taxonomic ranks [44].

Our implementation of the min-set-cov algorithm in sourmash also readily supports using custom reference databases as well as updating minimum metagenome covers with the addition of new reference genomes. When updating metagenome 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 a *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 analyses, 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 metagenome covers by reporting the taxonomies of the constituent genomes, weighted by distinct overlap and aggregated at the relevant taxonomic levels. Our CAMI-based taxonomic benchmarking shows that this approach is competitive with several extant approaches for all metrics across all taxonomic levels (Figures 3 and 4). This taxonomic accuracy also suggests that minimum metagenome covers themselves are likely to be accurate, since the taxonomic assignment is built solely on the metagenome cover.

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 here is involved in finding the reference genomes, which can have assignments in multiple taxonomic frameworks. For example, sourmash already supports GTDB [45] natively, and will also support the emerging LINS framework [46]. 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 even regenerating the minimum metagenome cover.

Interestingly, this 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 [47]; this is because we incorporate taxonomy *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. This is potentially useful for projects that are generating many new genomes and wish to use them for metagenome analysis. sourmash natively supports this functionality.

Our current implementation of taxonomic assignment in sourmash does not provide read-level assignment. However, it is a straightforward (if computationally expensive) exercise to use the read mapping approach developed in this paper to provide read-level taxonomic assignment along with genome abundance estimation.

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. This may present challenges: for example, in many cases the exact reference strains present in the metagenome will not be present in the database. This manifests in two ways - see Figure 5. First, for real metagenomes, 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 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 is present in the reference database. Methods for updating references from metagenome data sets may provide an opportunity for generating metagenome-specific references [48].

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 choose the phage genome. This is algorithmically straightforward to implement when desired.

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, summarizing *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 using FracMinHash at a low resolution, followed by a higher resolution (but more resource intensive) approach to identify the best 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 our approach may incorrectly choose between very closely related genomes, because the set of subsampled hashes 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. 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 [49] can also be used, with the benefit of reduced storage and memory usage.

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. This may be important for genomes with more repetitive content such as eukaryotic genomes. 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 [50].

Conclusion

The FracMinHash and min-set-cov approaches explored here provide powerful and accurate techniques for analyzing metagenomes, with well defined limitations. We show several immediate applications for both taxonomic and mapping-based analysis of metagenomes. 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 different data structures, alternative approximation algorithms, and additional summarization approaches.

Methods

Analytical analysis of FracMinHash

Given two arbitrary sets A and B which are subsets of a domain Ω, the containment index $C(A, B)$ is defined as $C(A, B) := \frac{|A \cap B|}{|A|}$. Let h be a perfect hash function $h : \Omega \to [0, H]$ for some $H \in \mathbb{R}$. For a *scale factor* s where $0 \leq s \leq 1$, a FracMinHash sketch of a set A is defined as follows:

$$\mathbf{FRAC}_S(A) = \{ h(a) \mid \forall a \in A \text{ s.t. } h(a) \leq Hs \}.$$

The scale factor s is a tunable parameter that can modify the size of the sketch. Using this FracMinHash sketch, we define the FracMinHash estimate of the containment index $\hat{C}_{\text{frac}}(A, B)$ as follows:

$$\hat{C}_{\text{frac}}(A, B) := \frac{|\mathbf{FRAC}_S(A) \cap \mathbf{FRAC}_S(B)|}{|\mathbf{FRAC}_S(A)|}.$$

For notational simplicity, we define $X_A := |\mathbf{FRAC}_S(A)|$. Observe that if one views h as a uniformly distributed random variable, we have that X_A is distributed as a binomial random variable: $X_A \sim \text{Binom}(|A|, s)$. Furthermore, if $A \cap B \neq \emptyset$ where both A and B are non-empty sets, then X_A and X_B are independent when the probability of success is strictly smaller than 1. Using these notations, we compute the expectation of $\hat{C}_{\text{frac}}(A, B)$.

Theorem 1: For $0 < s < 1$, if A and B are two distinct sets such that $A \cap B$ is non-empty,

$$\mathrm{E}\left[\hat{C}_{\mathrm{frac}}(A,B)\mathbb{1}_{|\mathbf{FRAC}_S(A)|>0}\right] = \frac{|A\cap B|}{|A|}\left(1-(1-s)^{|A|}\right).$$

Proof. Using the notation introduced previously, observe that

$$\hat{C}_{\mathrm{frac}}(A,B)\mathbb{1}_{|\mathbf{FRAC}_S(A)|>0} = \frac{X_{A\cap B}}{X_{A\cap B}+X_{A\setminus B}}\mathbb{1}_{X_{A\cap B}+X_{A\setminus B}>0},$$

and that the random variables $X_{A\cap B}$ and $X_{A\setminus B}$ are independent (which follows directly from the fact that $A\cap B$ is non-empty, and because A and B are distinct, $A\setminus B$ is also non-empty). We will use the following fact from standard calculus:

$$\int_0^1 x t^{x+y-1}\,dt = \frac{x}{x+y}\mathbb{1}_{x+y>0}.$$

Then using the moment generating function of the binomial distribution, we have

$$\begin{aligned}\mathrm{E}\left[t^X_{A\cap B}\right] &= (1-s+st)^{|A\cap B|}\\ \mathrm{E}\left[t^X_{A\setminus B}\right] &= (1-s+st)^{|A\setminus B|}.\end{aligned}$$

We also know by continuity that

$$\begin{aligned}\mathrm{E}\left[X_{A\cap B}\,t^{X_{A\cap B}-1}\right] &= \frac{d}{dt}(1-s+st)^{|A\cap B|}\\ &= |A\cap B|s(1-s+st)^{|A\cap B|-1}.\end{aligned}$$

Using these observations, we can then finally calculate that

$$\begin{aligned}\mathrm{E}\left[\frac{X_{A\cap B}}{X_{A\cap B}+X_{A\setminus B}}\mathbb{1}_{X_{A\cap B}+X_{A\setminus B}>0},\right] &= \mathrm{E}\left[\int_0^1 X_{A\cap B}\,t^{X_{A\cap B}+X_{A\setminus B}-1}\,dt\right]\\ &= \int_0^1 \mathrm{E}\left[X_{A\cap B}\,t^{X_{A\cap B}+X_{A\setminus B}-1}\,dt\right]\\ &= \int_0^1 \mathrm{E}\left[X_{A\cap B}\,t^{X_{A\cap B}-1}\right]\mathrm{E}\left[t^X_{A\setminus B}\right]\,dt\\ &= |A\cap B|\int_0^1 (1-s+st)^{|A\cap B|+|A\setminus B|-1}\,dt\\ &= \frac{|A\cap B|(1-s+st)^{|A|}}{|A|}\Bigg|_{t=0}^{t=1}\\ &= \frac{|A\cap B|}{|A|}\left(1-(1-s)^{|A|}\right),\end{aligned}$$

using Fubini's theorem and independence.

In light of Theorem 1, we note that $\hat{C}_{\text{frac}}(A, B)$ is *not* an unbiased estimate of $C(A, B)$. This may explain the observations in [36] that show suboptimal performance for short sequences (e.g. viruses). However, for sufficiently large $|A|$ and s, the bias factor $\left(1 - (1 - s)^{|A|}\right)$ is sufficiently close to 1.

Hence we can define:

$$C_{\text{frac}}(A, B) = \frac{|A \cap B|}{|A|} \left(1 - (1 - s)^{|A|}\right)^{-1}$$

which will have expectation

$$\mathrm{E}[C_{\text{frac}}(A, B)] = \frac{|A \cap B|}{|A|}$$

by Theorem 1.

Implementation of FracMinHash and min-set-cov

We provide implementations of FracMinHash and min-set-cov in the software package `sourmash`, which is implemented in Python and Rust and developed under the BSD license [22]. FracMinHash sketches were created for DNA sequence inputs using the `sourmash sketch dna` command with the `scaled` parameter. Minimum metagenome covers were generated using `sourmash gather` with the sketched metagenome as query against a collection of one or more sketched genomes.

sourmash is available at github.com/sourmash-bio/sourmash. The results in this paper were generated with sourmash v4.2.3.

Comparison between CMash, mash screen, and Scaled MinHash.

Experiments use $k = \{21, 31, 51\}$ (except for Mash, which only supports $k \leq 32$). For Mash and CMash they were run with $n = \{1000, 10000\}$ to evaluate the containment estimates when using larger sketches with sizes comparable to the FracMinHash sketches with $scaled = 1000$. The truth set is calculated using an exact k-mer counter implemented with a *HashSet* data structure in the Rust programming language [51]. The sourmash results were generated with `sourmash search --containment`.

For *Mash Screen* the ratio of hashes matched by total hashes is used instead of the *Containment Score*, since the latter uses a k-mer survival process modeled as a Poisson process first introduced in [52] and later used in the *Mash distance* [20] and *Containment score* [24] formulations.

GenBank database sketching and searches

Minimum metagenome covers were calculated using a microbial genome subset of GenBank (July 2020, 725,339 genomes) using a scaled factor of 2000 and a k-mer size of 31. Sketches for all genomes and metagenomes were calculated with `sourmash sketch dna -p scaled=2000,k=31`. The minimum metagenome covers were calculated using all genomes sharing 50 hashes with the metagenome (that is, an estimated overlap of 100,000 k-mers) with `sourmash gather --threshold-bp 1e5`. Overlapping sketches were saved with `--save-prefetch` and matches were saved with `--save-matches`.

The GenBank database used is 24 GB in size and is available for download through the sourmash project [53].

Taxonomy

The CAMI evaluations were run with the sourmash CAMI pipeline [54] against the Jan 8, 2019 RefSeq snapshot provided by CAMI. This pipeline generated Open-community Profiling Assessment (OPAL) compatible output [30]. This output was then processed with the standard CAMI tools.

Read mapping and hybrid mapping pipeline

Metagenome reads were mapped to reference genomes using minimap2 v2.17 [55] with short single-end read mapping mode (`-x sr`).

The hybrid selection and mapping pipeline using the rank-ordered min-set-cov results was implemented in the `subtract_gather.py` script in the genome-grist package [56].

The complete workflow, from metagenome download to taxonomic analysis and iterative mapping, is implemented in the genome-grist package. genome-grist uses snakemake [57] to define and execute a workflow that combines sourmash sketching, metagenome cover calculation, and taxonomic analysis with metagenome download from the SRA, genome download from GenBank, and read mapping. We used genome-grist v0.7.4 [58] to generate the results in this paper; see `conf-paper.yml` in the pipeline repository.

genome-grist relies on matplotlib [59], Jupyter Notebook [60], numpy [61], pandas [62], papermill, samtools [63], bedtools [64], fastp [65], khmer [66], screed [67], seqtk [68], and sra-tools [69]. These tools are all installed and managed in snakemake via conda [70] and bioconda [71]. genome-grist itself is developed under the BSD 3-clause open source license, and is available at github.com/dib-lab/genome-grist/.

Intermediate data products and figure generation

All figures were generated using the Jupyter Notebooks from v0.1 of the github.com/dib-lab/2021-paper-sourmash-gather-pipeline repository [72]. This repository also contains the intermediate data products necessary for figure generation.

Metagenome data set accessions

The accessions for the metagenome data sets in Table 1 are:

data set	SRA accession
`zymo mock`	SRR12324253
`podar mock`	SRR606249
`gut real`	SRR5650070
`oil well real`	SRR1976948

References

1. **A genomic catalog of Earth's microbiomes**
Stephen Nayfach, Simon Roux, Rekha Seshadri, Daniel Udwary, Neha Varghese, Frederik Schulz, Dongying Wu, David Paez-Espino, I-Min Chen, Marcel Huntemann, … IMG/M Data Consortium
Nature Biotechnology (2020-11-09) https://doi.org/ghjh4b
DOI: 10.1038/s41587-020-0718-6 · PMID: 33169036 · PMCID: PMC8041624

2. **Metagenomic assessment of the global diversity and distribution of bacteria and fungi**
Mohammad Bahram, Tarquin Netherway, Clémence Frioux, Pamela Ferretti, Luis Pedro Coelho, Stefan Geisen, Peer Bork, Falk Hildebrand
Environmental Microbiology (2020-12-02) https://doi.org/gjcw9f
DOI: 10.1111/1462-2920.15314 · PMID: 33185929 · PMCID: PMC7898879

3. **The Integrative Human Microbiome Project** *Nature* (2019-05) https://doi.org/gf3wp9
DOI: 10.1038/s41586-019-1238-8 · PMID: 31142853 · PMCID: PMC6784865

4. **Structure and function of the global ocean microbiome**
Shinichi Sunagawa, Luis Pedro Coelho, Samuel Chaffron, Jens Roat Kultima, Karine Labadie, Guillem Salazar, Bardya Djahanschiri, Georg Zeller, Daniel R Mende, Adriana Alberti, …
Science (2015-05-22) https://doi.org/4tr
DOI: 10.1126/science.1261359 · PMID: 25999513

5. **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

6. **Challenges in benchmarking metagenomic profilers**
Zheng Sun, Shi Huang, Meng Zhang, Qiyun Zhu, Niina Haiminen, Anna Paola Carrieri, Yoshiki Vázquez-Baeza, Laxmi Parida, Ho-Cheol Kim, Rob Knight, Yang-Yu Liu
Nature Methods (2021-05-13) https://doi.org/gj2n7w
DOI: 10.1038/s41592-021-01141-3 · PMID: 33986544 · PMCID: PMC8184642

7. **Critical Assessment of Metagenome Interpretation - the second round of challenges**
F Meyer, A Fritz, Z-L Deng, D Koslicki, A Gurevich, G Robertson, M Alser, D Antipov, F Beghini, D Bertrand, … AC McHardy
Cold Spring Harbor Laboratory (2021-07-12) https://doi.org/gk566x
DOI: 10.1101/2021.07.12.451567

8. **A review of methods and databases for metagenomic classification and assembly**
Florian P Breitwieser, Jennifer Lu, Steven L Salzberg
Briefings in Bioinformatics (2019-07) https://doi.org/gdq95k
DOI: 10.1093/bib/bbx120 · PMID: 29028872 · PMCID: PMC6781581

9. **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

10. **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 (2018-01) https://doi.org/gnnprp
DOI: 10.1186/s13062-018-0208-7 · PMID: 29678199 · PMCID: PMC5910613

11. **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

12. **Microbial abundance, activity and population genomic profiling with mOTUs2**
Alessio Milanese, Daniel R Mende, Lucas Paoli, Guillem Salazar, Hans-Joachim Ruscheweyh, Miguelangel Cuenca, Pascal Hingamp, Renato Alves, Paul I Costea, Luis Pedro Coelho, … Shinichi Sunagawa
Nature Communications (2019-03-04) https://doi.org/gfwktp
DOI: 10.1038/s41467-019-08844-4 · PMID: 30833550 · PMCID: PMC6399450

13. **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 (2018-11-12) https://doi.org/gg8bdg
DOI: 10.1093/nar/gky1085 · PMID: 30418610 · PMCID: PMC6324079

14. **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

15. **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

16. **ganon: precise metagenomics classification against large and up-to-date sets of reference sequences**
Vitor C Piro, Temesgen H Dadi, Enrico Seiler, Knut Reinert, Bernhard Y Renard
Bioinformatics (2020-07) https://doi.org/gnxxz8
DOI: 10.1093/bioinformatics/btaa458 · PMID: 32657362 · PMCID: PMC7355301

17. **Metalign: efficient alignment-based metagenomic profiling via containment min hash**
Nathan LaPierre, Mohammed Alser, Eleazar Eskin, David Koslicki, Serghei Mangul
Genome Biology (2020-09-10) https://doi.org/ghtqrz
DOI: 10.1186/s13059-020-02159-0 · PMID: 32912225 · PMCID: PMC7488264

18. **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

19. **A Greedy Heuristic for the Set-Covering Problem**
V Chvatal
Mathematics of Operations Research (1979-08) https://doi.org/dx2gd2

DOI: 10.1287/moor.4.3.233

20. **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

21. **Debiasing FracMinHash and deriving confidence intervals for mutation rates across a wide range of evolutionary distances**
Mahmudur Rahman Hera, NTessa Pierce-Ward, David Koslicki
Cold Spring Harbor Laboratory (2022-01-14) https://doi.org/gn342h
DOI: 10.1101/2022.01.11.475870

22. **sourmash: a library for MinHash sketching of DNA**
C Titus Brown, Luiz Irber
The Journal of Open Source Software (2016-09-14) https://doi.org/ghdrk5
DOI: 10.21105/joss.00027

23. **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

24. **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

25. **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

26. **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

27. **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

28. **Evaluating Metagenome Assembly on a Simple Defined Community with Many Strain Variants**
Sherine Awad, Luiz Irber, CTitus Brown
Cold Spring Harbor Laboratory (2017-06-25) https://doi.org/ghvn6x
DOI: 10.1101/155358

29. **Letter-Value Plots: Boxplots for Large Data**
Heike Hofmann, Hadley Wickham, Karen Kafadar

Journal of Computational and Graphical Statistics (2017-07-03) https://doi.org/gf38v7
DOI: 10.1080/10618600.2017.1305277

30. **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

31. **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

32. **Assessing taxonomic metagenome profilers with OPAL**
Fernando Meyer, Andreas Bremges, Peter Belmann, Stefan Janssen, Alice C McHardy, David Koslicki
Genome Biology (2019-03-04) https://doi.org/gf9vbw
DOI: 10.1186/s13059-019-1646-y · PMID: 30832730 · PMCID: PMC6398228

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

34. **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

35. **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

36. **luizirber/phd 2020.09.28**
Luiz Irber, CTitus Brown, Gabriel Marcondes
Zenodo (2020-09-28) https://zenodo.org/record/4057151

37. **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

38. **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

39. **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

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

41. **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

42. **Fast search of thousands of short-read sequencing experiments**
Brad Solomon, Carl Kingsford
Nature Biotechnology (2016-02-08) https://doi.org/f8ddk3
DOI: 10.1038/nbt.3442 · PMID: 26854477 · PMCID: PMC4804353

43. **DIAMOND+MEGAN: Fast and Easy Taxonomic and Functional Analysis of Short and Long Microbiome Sequences**
Caner Bağcı, Sascha Patz, Daniel H Huson
Current Protocols (2021-03-03) https://doi.org/gjhfck
DOI: 10.1002/cpz1.59 · PMID: 33656283

44. **MetaPalette: a k -mer Painting Approach for Metagenomic Taxonomic Profiling and Quantification of Novel Strain Variation**
David Koslicki, Daniel Falush
mSystems (2016-06-28) https://doi.org/gg3gbd
DOI: 10.1128/msystems.00020-16 · PMID: 27822531 · PMCID: PMC5069763

45. **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 (2022-01-07) https://doi.org/gm97d8
DOI: 10.1093/nar/gkab776 · PMID: 34520557 · PMCID: PMC8728215

46. **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

47. **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

48. **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

49. **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

50. **Centrifuge: rapid and sensitive classification of metagenomic sequences**
Daehwan Kim, Li Song, Florian P Breitwieser, Steven L Salzberg
Genome Research (2016-12) https://doi.org/f9fnrr
DOI: 10.1101/gr.210641.116 · PMID: 27852649 · PMCID: PMC5131823

51. **The rust language**
Nicholas D Matsakis, Felix S Klock II
ACM SIGAda Ada Letters (2014-11-26) https://doi.org/gntjvb
DOI: 10.1145/2692956.2663188

52. **An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data**
Huan Fan, Anthony R Ives, Yann Surget-Groba, Charles H Cannon
BMC Genomics (2015-07-14) https://doi.org/f7s6tp
DOI: 10.1186/s12864-015-1647-5 · PMID: 26169061 · PMCID: PMC4501066

53. **Welcome to sourmash! — sourmash 4.2.4.dev8+g1c46d7a5.d20220117 documentation**
https://sourmash.readthedocs.io/en/latest/

54. **GitHub - luizirber/2020-cami: Preparing sourmash for CAMI 2 evaluations**
GitHub
https://github.com/luizirber/2020-cami

55. **Minimap2: pairwise alignment for nucleotide sequences**
Heng Li
Bioinformatics (2018-09-15) https://doi.org/gdhbqt
DOI: 10.1093/bioinformatics/bty191 · PMID: 29750242 · PMCID: PMC6137996

56. **GitHub - dib-lab/genome-grist: map Illumina metagenomes to genomes!**
GitHub
https://github.com/dib-lab/genome-grist

57. **Sustainable data analysis with Snakemake**
Felix Mölder, Kim Philipp Jablonski, Brice Letcher, Michael B Hall, Christopher H Tomkins-Tinch, Vanessa Sochat, Jan Forster, Soohyun Lee, Sven O Twardziok, Alexander Kanitz, … Johannes Köster
F1000Research (2021-04-19) https://doi.org/gj76rq
DOI: 10.12688/f1000research.29032.2 · PMID: 34035898 · PMCID: PMC8114187

58. **dib-lab/genome-grist: v0.7.4**
CTitus Brown, Tessa Pierce Ward, Mohamed Abuelanin, Marisa Lim
Zenodo (2021-12-19) https://zenodo.org/record/5792144

59. **Matplotlib: A 2D Graphics Environment**
John D Hunter
Computing in Science & Engineering (2007) https://doi.org/drbjhg
DOI: 10.1109/mcse.2007.55

60. **Jupyter Notebooks – a publishing format for reproducible computational workflows**
Thomas Kluyver, Benjamin Ragan-Kelley, P&#233, Fernando Rez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, ... Jupyter Development Team
Positioning and Power in Academic Publishing: Players, Agents and Agendas (2016)
https://ebooks.iospress.nl/doi/10.3233/978-1-61499-649-1-87
DOI: 10.3233/978-1-61499-649-1-87

61. **Array programming with NumPy**
Charles R Harris, KJarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, ... Travis E Oliphant
Nature (2020-09-16) https://doi.org/ghbzf2
DOI: 10.1038/s41586-020-2649-2 · PMID: 32939066 · PMCID: PMC7759461

62. **pandas-dev/pandas: Pandas 1.4.0rc0**
Jeff Reback, jbrockmendel, Wes McKinney, Joris Van den Bossche, Tom Augspurger, Phillip Cloud, Simon Hawkins, Matthew Roeschke, gfyoung, Sinhrks, ... Skipper Seabold
Zenodo (2022-01-06) https://zenodo.org/record/5824773

63. **The Sequence Alignment/Map format and SAMtools**
H Li, B Handsaker, A Wysoker, T Fennell, J Ruan, N Homer, G Marth, G Abecasis, R Durbin
Bioinformatics (2009-06-08) https://doi.org/ff6426
DOI: 10.1093/bioinformatics/btp352 · PMID: 19505943 · PMCID: PMC2723002

64. **BEDTools: a flexible suite of utilities for comparing genomic features**
Aaron R Quinlan, Ira M Hall
Bioinformatics (2010-03-15) https://doi.org/cmrms3
DOI: 10.1093/bioinformatics/btq033 · PMID: 20110278 · PMCID: PMC2832824

65. **fastp: an ultra-fast all-in-one FASTQ preprocessor**
Shifu Chen, Yanqing Zhou, Yaru Chen, Jia Gu
Bioinformatics (2018-09-01) https://doi.org/gd9mrb
DOI: 10.1093/bioinformatics/bty560 · PMID: 30423086 · PMCID: PMC6129281

66. **khmer release v2.1: software for biological sequence analysis**
Daniel Standage, Ali yari, Lisa J. Cohen, Michael R. Crusoe, Tim Head, Luiz Irber, Shannon EK Joslin, N B. Kingsley, Kevin D. Murray, Russell Neches, ... C Titus Brown
The Journal of Open Source Software (2017-07-03) https://doi.org/gntbpv
DOI: 10.21105/joss.00272

67. **screed - short read sequence utils — screed 1.0 documentation**
https://screed.readthedocs.io/en/v1.0/

68. **GitHub - lh3/seqtk: Toolkit for processing sequences in FASTA/Q formats**
GitHub
https://github.com/lh3/seqtk

69. **GitHub - ncbi/sra-tools: SRA Tools**
GitHub
https://github.com/ncbi/sra-tools

70. **Anaconda Documentation — Anaconda documentation** https://docs.anaconda.com/

71. **Bioconda: sustainable and comprehensive software distribution for the life sciences**
Björn Grüning, Ryan Dale, Andreas Sjödin, Brad A Chapman, Jillian Rowe, Christopher H Tomkins-Tinch, Renan Valieris, Johannes Köster, The Bioconda Team

Nature Methods (2018-07-02) https://doi.org/gd2xzp
DOI: 10.1038/s41592-018-0046-7 · PMID: 29967506

72. **dib-lab/2021-paper-sourmash-gather-pipeline: v0.1**
CTitus Brown
Zenodo (2021-12-20) https://zenodo.org/record/5793387

Appendix

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.↩