SOFTWARE TOOL ARTICLE

Large-scale sequence comparisons with *sourmash* [version 1; peer review: 2 approved]

N. Tessa Pierce *, Luiz Irber *, Taylor Reiter*, Phillip Brooks *, C. Titus Brown

Department of Population Health and Reproduction, University of California, Davis, Davis, California, 95616, USA

* Equal contributors

v1 **First published:** 04 Jul 2019, **8**:1006 (https://doi.org/10.12688/f1000research.19675.1)
Latest published: 04 Jul 2019, **8**:1006 (https://doi.org/10.12688/f1000research.19675.1)

Abstract

The sourmash software package uses MinHash-based sketching to create "signatures", compressed representations of DNA, RNA, and protein sequences, that can be stored, searched, explored, and taxonomically annotated. sourmash signatures can be used to estimate sequence similarity between very large data sets quickly and in low memory, and can be used to search large databases of genomes for matches to query genomes and metagenomes. sourmash is implemented in C++, Rust, and Python, and is freely available under the BSD license at http://github.com/dib-lab/sourmash.

Keywords

sequence analysis, MinHash, k-mer, sourmash, bioinformatics

This article is included in the Python Collection collection.

Open Peer Review

Reviewer Status ✓ ✓

	Invited Reviewers	
	1	**2**
version 1 published 04 Jul 2019	✓ report	✓ report

1 **Brad Solomon**, Johns Hopkins University, Baltimore, USA

2 **Rayan Chikhi**, Institut Pasteur, Paris, France

Any reports and responses or comments on the article can be found at the end of the article.

Corresponding author: C. Titus Brown (ctbrown@ucdavis.edu)

Author roles: Pierce NT: Formal Analysis, Investigation, Software, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Irber L**: Conceptualization, Formal Analysis, Methodology, Software, Validation, Writing – Review & Editing; **Reiter T**: Formal Analysis, Investigation, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Brooks P**: Formal Analysis, Investigation, Methodology, Software, Validation, Writing – Review & Editing; **Brown CT**: Conceptualization, Formal Analysis, Funding Acquisition, Investigation, Methodology, Project Administration, Resources, Software, Supervision, Validation, Visualization, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: This work is funded in part by the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative [GBMF4551 to CTB]. NTP was supported by a National Science Foundation Postdoctoral Fellowship in Biology [1711984].
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

How to cite this article: Pierce NT, Irber L, Reiter T *et al.* **Large-scale sequence comparisons with *sourmash* [version 1; peer review: 2 approved]** F1000Research 2019, **8**:1006 (https://doi.org/10.12688/f1000research.19675.1)

First published: 04 Jul 2019, **8**:1006 (https://doi.org/10.12688/f1000research.19675.1)

Introduction

Bioinformatic analyses rely on sequence comparison for many applications, including variant analysis, taxonomic classification and functional annotation. As the Sequence Read Archive now contains over 20 Petabases of data[1], there is great need for methods to quickly and efficiently conduct similarity searches on a massive scale. MinHash techniques[2] utilize random sampling of k-mer content to generate small subsets known as "sketches" such that Jaccard similarity (the intersection over the union) of two sequence data sets remains approximately equal to their true Jaccard similarity[2,3]. The many-fold size reductions gained via MinHash opens the door to extremely large scale searches.

While the initial k-mer MinHash implementation focused on enabling Jaccard similarity comparisons[3], it has since been modified and extended to enable k-mer abundance comparisons[4], decrease runtime and memory requirements[5], and work on streaming input data[6]. Furthermore, as Jaccard similarity is impacted by the relative size of the sets being compared, containment searches[2,7,8] have been developed to enable detection of a small set within a larger set, such as a genome within a metagenome.

Here we present version 2.0 of sourmash[9], a Python library for building and utilizing MinHash sketches of DNA, RNA, and protein data. sourmash incorporates and extends standard MinHash techniques for sequencing data, with a particular focus towards enabling efficient containment queries using large databases. This is accomplished with two modifications: (1) building sketches via a modulo approach[2], and (2) implementing a modified Sequence Bloom Tree[10] to enable both similarity and containment searches. In most cases, these features enable sourmash database comparisons in sub-linear time.

Standard genomic MinHash techniques, first implemented in Ondov BD *et al.*[3], retain the minimum *n* k-mer hashes as a representative subset. sourmash extends these methods by incorporating a user-defined "scaled" factor to build sourmash sketches via a modulo approach, rather than the standard bottom-hash approach[2]. Sketches built with this approach retain the same fraction, rather than number, of k-mer hashes, compressing both large and small datasets at the same rate.

This enables comparisons between datasets of disparate sizes but can sacrifice some of the memory and storage benefits of standard MinHash techniques, as the signature size scales with the number of unique k-mers rather than remaining fixed[8]. In sourmash, use of the "scaled" factor enables user modification of the trade-off between search precision and sketch size, with the caveat that searches and comparisons can only be conducted using signatures generated with identical "scaled" values (downsampled at the same rate).

To enable large-scale database searches using these signatures, sourmash implements a modified Sequence Bloom Tree (SBT), the SBT-MinHash (SBTMH), that allows both similarity (sourmash search) and containment (sourmash gather) searches for taxonomic exploration and classification. Notably, Jaccard similarity searches using this modified SBT require storage of the cardinality of the smallest MinHash below each node in order to properly bound similarity. sourmash also implements a second database format, "LCA", for in-memory search when sufficient RAM is available or database size is tractable. The LCA format can be leveraged to return additional information, such as taxonomic lineage.

In addition to these modifications, sourmash implements k-mer abundance tracking[4] within signatures to allow abundance comparisons across datasets and facilitate metagenome, metatranscriptome, and transcriptome analyses, and is compatible with streaming approaches. The sourmash library is implemented in C++, Rust[11], and Python, and can be accessed both via command line and Python API. The code is available under the BSD license at http://github.com/dib-lab/sourmash.

Implementation

`sourmash` provides a user-friendly, extensible platform for MinHash signature generation and manipulation for DNA, RNA, and protein data. Sourmash is designed to facilitate containment queries for taxonomic exploration and identification while maintaining all functionality available via standard genomic MinHash techniques.

sourmash Signatures

`sourmash` modifies standard genomic MinHash techniques in two ways. First, `sourmash` scales the number of retained hashes to better represent and compare datasets of varying size and complexity. Second, `sourmash` optionally tracks the abundance of each retained hash, to better represent data of metagenomic and transcriptomic origin and allow abundance comparisons.

Scaling. `sourmash` implements a method inspired by modulo sketches[2] to dynamically scale hash subset retention size (*n*). When using scaled signatures, users provide a scaling factor (*s*) that divides the hash space into *s* equal bands, retaining hashes within the minimum band as the sketch. These scaled signatures can be converted to standard bottom-hash signatures, if the subset retention size *n* is equal to or smaller than the number of hashes in the scaled signature. `sourmash` provides a signature utility, downsample, to convert sketches when possible. Finally, to maintain compatibility with sketches generated by other programs such as Mash[3], `sourmash` generates standard bottom-hash MinHash sketches if users specify the hash subset size *n* rather than scaling factor.

Streaming compatibility. Scaled signature generation is streaming compatible and provides some advantages over streaming calculation using standard MinHash. As data streams in, standard MinHash replaces hashes based on the minimum hash value to maintain a fixed number of hashes in the signature. In contrast, no hash is ever removed from a scaled signature as more data is received. As a result, for searches of a database using streamed-in data, all prior matches remain valid (although their significance may change as more data is received). This allows us to place algorithmic guarantees on containment searches using streaming data.

Abundance tracking. `sourmash` extends MinHash functionality by implementing abundance tracking of k-mers. k-mer counts are incremented after hashing as each k-mer is added to the hash table. `sourmash` tracks abundance for k-mers in the minimum band and stores this information in the signature. These values accompany the hashes in downstream comparison processes, making signatures better representations of repetitive sequences of metagenomic and transcriptomic origin.

Signatures. MinHash sketches associated with a single sequence file are stored together in a "signature" file, which forms the basis of all `sourmash` comparisons. Signatures may include sketches generated with different *k* sizes or molecule type (nucleotide or protein) and are stored in JSON format to maintain human readability and ensure proper interoperability.

Signatures can only be compared against signatures and databases made from the same parameters (*k* size(s), scaled value, nucleotide or protein level). If signatures differ in their scaled value or size(*n*), the larger signatures can be downsampled to become comparable with smaller signatures using the *signature utilities*, below. `sourmash` also provides 6-frame nucleotide translation to generate protein signatures from nucleotide input if desired.

Signature utilities. `sourmash` provides a number of utilities to facilitate set operations between signatures (`merge`, `intersect`, `extract`, `downsample`, `flatten`, `subtract`, `overlap`), and handling (`describe`, `rename`, `import`, `export`) of `sourmash` signatures. These can be accessed via the `sourmash signature` subcommand.

SBT-MinHash

`sourmash` implements a modified Sequence Bloom Tree (SBT[10]), the SBT-MinHash (SBTMH), which can efficiently capture large volumes of MinHashes (e.g., all microbes in GenBank) and support multiple search regimes that improve on run time of linear searches.

Implementation. The SBTMH is a n-ary tree (binary by default), where leaf nodes are MinHash signatures and internal nodes are Bloom Filters. Each Bloom Filter contains all the values from its children, so the root node contains all the values from all signatures. SBTMH is designed to be extensible such that signatures can be subsequently added without the need for full regeneration. Adding a new signature to SBTMH causes parent nodes up to the root to be updated, but other nodes are not affected.

SBTMH trees can be combined if desired: In the simplest case, adding a new root and updating it with the content of the previous roots is sufficient, and this preserves all node information without changes. As an example, separate indices can be created for each RefSeq subdivision (bacteria, archaea, fungi, etc) and be combined depending on the application (such as an analysis for bacteria + archaea, but not fungi). In practice, this is most useful for updating the SBTMH, as both `search` and `gather` support search over multiple databases without the need for rebuilding a single large database.

Searching SBTMH. Similarity searches start at the root of the SBTMH, and check for query elements present in each internal node. If the similarity does not reach the threshold, the subtree under that node does not need to be searched. If a leaf is reached, it is returned as a match to the query signature. In order to enable similarity (in

addition to containment) searches using this modified SBTMH, nodes store the cardinality of the smallest signature below each node in order to properly bound similarity. The full SBTMH does not need to be imported to RAM during searches, making this method best for rapid searching with minimal memory requirements. However, if sufficient RAM is available, searches of databases (or many signatures) may be completed in memory via an alternate database format (discussed below).

SBTMH utilities. `sourmash` provides several utilities for construction, use, and handling of SBTMH databases. These include `sbt index` to index a collection of signatures as an SBTMH for fast searching, `sbt append` to add signatures, and `sbt combine` to join two SBTMH databases.

LCA database

`sourmash` implements an alternate database format, LCA, to support in-memory queries. This implementation utilizes two named lists to store MinHashed databases: the first containing MinHashes, and the second containing taxonomic information, with both lists named by sample name. This structure facilitates direct look-up of MinHashes, and thus can be leveraged to return additional information, such as taxonomic lineage. The LCA database structure can be prepared using the `sourmash lca index` command.

Assessing sequence similarity

Pairwise comparisons

For sequence comparison, `sourmash compare` reimplements Jaccard sequence similarity comparison to enable comparison between scaled MinHashes. When abundance tracking of k-mers is enabled, `compare` instead calculates the cosine similarity, although we recommend using more accurate approaches for detailed comparisons[6].

Database searches

In addition to conducting pairwise comparisons, two types of database searches are implemented: breadth-first similarity searches (`sourmash search`) and best-first containment searches (`sourmash gather`), which support different biological queries. These searches can be conducted using either database format.

Similarity queries. Breadth-first `sourmash search` can be used to obtain all MinHashes in the SBTMH that are present in the query signature (above a specified threshold). This style of search is streaming-compatible, as the query MinHash can be augmented as the search is occurring.

Containment queries Best-first `sourmash gather` implements a greedy algorithm where the SBTMH is descended on a linear path through a set of internal nodes until the highest containment leaf is reached. The hashes in this leaf are then subtracted from the query MinHash and the process is repeated until the threshold minimum is reached. `sourmash` post-processes similarity statistics after the search such that it reports percent identity and unique identity for each match.

Taxonomy-resolved searches `sourmash` can conduct taxonomy-resolved searches uses the "least common ancestor" approach (as in Kraken[12]), to identify k-mers in a query. From this it can either find a consensus taxonomy between all the k-mers (`sourmash classify`) or it can summarize the mixture of k-mers present in one or more signatures (`sourmash summarize`).

Operation

`sourmash` is a tool for building and utilizing MinHash signatures of DNA, RNA, and protein sequences. A straightforward workflow consists of generating a signature using `sourmash compute`, and comparing it against other signatures or databases of signatures via `sourmash compare, search, gather, lca search`, or `lca gather`. `sourmash` has no particular memory requirements, but does need to hold the largest single sequence in memory while generating a signature. For example, computing a signature from a 100Mb human microbiome sample requires 30MB of RAM, and searching it against a sourmash Genbank signature database takes 1–6 minutes and requires 2–6 GB of RAM, depending on the search type. "LCA" databases are smaller on disk but require more memory to be searched.

Below we provide several use cases to demonstrate the utility of `sourmash` for sequence comparisons, starting with signature generation and proceeding into signature comparisons, tetranucleotide frequency clustering analysis, and taxonomic classification. We primarily demonstrate nucleotide-level applications in this paper; protein-level analyses will be explored further in future work. Additional information and tutorials are available at https://sourmash.readthedocs.io.

Use cases

Installation

`sourmash` is available for both Linux and OSX, and runs under either `Python 2.7.x` or `Python 3.5+`. To install sourmash, we recommend using conda. For these examples, we used sourmash v.2.0.1, installed with conda v 4.6.14.

```
conda install -c conda-forge \
-c bioconda sourmash
```

Alternate installation instructions are available at sourmash.readthedocs.io.

Creating a signature

All `sourmash` comparisons work on signatures, compressed representations of biological sequencing data. To create a signature from sequences with abundance tracking:

```
# download the genome
curl -L  https://osf.io/bjh2y/download  \
-o GCF_000005845.2_ASM584v2_genomic.fna.gz

# calculate the signature
sourmash compute -k 21,31,51 \
  --scaled 2000 \
  --track-abundance \
  -o GCF_000005845.2_ASM584v2_genomic.sig  \
  GCF_000005845.2_ASM584v2_genomic.fna.gz
```

Because a signature can contain multiple MinHashes, multiple k-sizes can be specified per a sequence. Only one scaled size can be used.

By default, the name of the file becomes the name of the signature. To name the signature from the first line of the sequencing file, use `--name-from-first`. Although the `--track-abundance` flag is optional, since downstream comparison methods contain the flag `--ignore-abundance` to ignore them, we recommend calculating all signatures with abundance tracking.

To create a signature from protein sequences:

```
# download amino acid sequences
curl -L https://osf.io/y9kra/download \
-o GCF_000146045.2_R64_protein.faa.gz

# calculate the signature
sourmash compute -k 11,21,31 \
    --scaled 2000 \
    --track-abundance \
    -o GCF_000146045.2_R64_protein.sig \
    GCF_000146045.2_R64_protein.faa.gz
```

Signatures can also be made directly from reads. Depending on the downstream use cases, we recommend different preparation methods. When the user aims to `compare` the signature to other signatures, we recommend k-mer trimming the reads before computing the signature. Because `compare` does an all-by-all comparison of signatures, errors in the reads will falsely deflate the similarity metric. We recommend trimming RNA-seq or metagenome reads with `trim-low-abund.py` in the khmer package[13], a dependency of `sourmash`.

```
# download the reads
curl -L -o ERR458584.fq.gz \
     https://osf.io/pfxth/download

# trim the reads
trim-low-abund.py ERR458584.fq.gz \
```

```
     -V -Z 10 -C 3 --gzip -M 3e9 \
     -o  ERR458584.khmer.fq.gz

# calculate the signature from trimmed reads
sourmash compute -k 21,31,51 \
    --scaled 2000 \
    --track-abundance \
    -o ERR458584.khmer.sig \
    ERR458584.khmer.fq.gz
```

When using methods that compare a signature against a database such as `gather` or `search`, k-mer trimming need not be used. These methods use exact matching of hashes in the signature to those in the databases. k-mer trimming could increase false negatives, but results on k-mer trimmed data will more accurately represent the proportions of content in the data.

```
# calculate the signature from raw reads
sourmash compute -k 21,31,51 \
    --scaled 2000 \
    --track-abundance \
    -o ERR458584.sig \
    ERR458584.fq.gz
```

Comparing many signatures

Signatures calculated with abundance tracking enable rapid comparison of sequences where k-mer frequency is variable, and can be leveraged for quality control and summarization methods. For example, principle component analysis (PCA) and multidimensional scaling (MDS) are standard quality control and summarization methods for count data generated during RNA-seq analysis[14]. `sourmash` can be used to build this MDS plot in a reference-free (or assembly-free) manner, using k-mer abundances of the input reads. We also find this useful for comparing other types of RNA sequencing samples (mRNA, ribo-depleted, 3' tag-seq, metatranscriptomes, and transcriptomes).

MDS

Here, we use a set of four *Saccharomyces cerevisiae* RNA-seq samples: replicate wild-type samples and replicate mutant (*SNF2*) samples[15]. To use `sourmash` to build an MDS plot, we first trim the data to remove low abundance k-mers via khmer[13]. We demonstrate the streaming capability of `sourmash` by downloading, k-mer trimming, and calculating a signature with one command. This allows the user to generate signatures without needing to store large files locally.

```
curl -L https://osf.io/pfxth/download \
  | trim-low-abund.py -V -Z 10 \
  -C 3 -M 3e9 -o - -\
  | sourmash compute -k 31 \
  --scaled 2000 --track-abundance \
  -o ERR458584.khmer.sig -
```

The signature will be named from the input filename, in this case –. We can change the name to reflect its contents using the `signature rename` function.

```
sourmash signature rename \
    -k 31 -o ERR458584.khmer-named.sig \
    ERR458584.khmer.sig \
    ERR458584.khmer
```

We can also check that the name has been changed.

```
sourmash signature describe \
    ERR458584.khmer-named.sig
```

Using signatures from four samples, we can compare the files with the `compare` function. Here we download signatures calculated and renamed using the above commands. We output the comparison matrix as a csv for downstream use in other platforms.

```
# download signatures
curl -L -o yeast-sigs.tar.gz  \
    https://osf.io/pk2w5/download

# uncompress the signatures
tar xvf yeast-sigs.tar.gz

# compare the signatures
sourmash compare -k 31 \
  --csv yeast-comp.csv \
  *named.sig
```

We then import the `compare` similarity matrix into R (v3.4.1) to produce an MDS plot with wild-type samples (ERR459011, ERR459102) in yellow and mutant samples (ERR458584, ERR458829) in blue.

```
# Read data into R
comp_mat <- read.csv("yeast-comp.csv")

# Set row labels
rownames(comp_mat) <- colnames(comp_mat)

# Transform for plotting
comp_mat <- as.matrix(comp_mat)

# Make an MDS plot
fit <- dist(comp_mat)
fit <- cmdscale(fit)
plot(fit[ , 2] ~ fit[ , 1],
     xlab = "Dim 1",
     ylab = "Dim 2",
     xlim= c(-.6, .9),
     main = "sourmash Compare MDS")
# add labels to the plot
text(fit[ , 2]~ fit[ , 1],
     labels = row.names(fit),
     pos = 4, font = 1,
     data = fit,
     col = c("blue", "blue",
             "orange", "orange"))
```

For comparison, we also produced an MDS plot using a more traditional approach, utilizing Salmon (v0.11.3)[16] to quantify abundance relative to an *S. cerevisiae* reference, and edgeR (v3.22.5)[17] to build an MDS plot (Figure 1; code available online at https://osf.io/97rt4/).

Tetranucleotide Frequency Clustering

We can also use `sourmash` with abundance tracking for tetranucleotide frequency clustering. Tetranucleotide usage is species-specific, with strongest conservation in DNA coding regions[18]. This is often used in metagenomics as one method to "bin" assembled contiguous sequences together that are from the same species[19]. Recently, tetranucleotide frequency clustering using `sourmash` was used to detect microbial contamination in the domesticated olive genome[20]. Here we reimplement this approach using 100 of the 11,038 scaffolds in the draft genome. We calculate the signature using a k-mer size of 4, use all 4-mers, and track abundance. Then we use `sourmash compare` to calculate the similarity between each scaffold. (The `--singleton` flag calculates a signature for each sequence in the fasta file.)

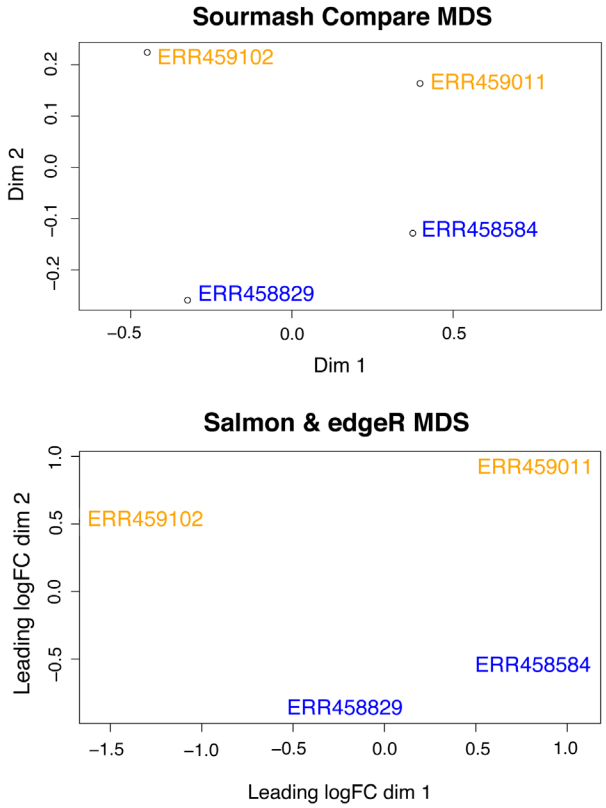

Figure 1. The MDS plots produced from the reference-free `sourmash compare` similarity matrix and the transcript quantification analysis (salmon and edgeR) are similar. Wild-type *S. cerevisiae* samples (ERR459011, ERR459102) are in yellow and mutant samples (ERR458584, ERR458829) in blue.

```
# download the subsampled genome
curl -L https://osf.io/xusfa/download \
    -o Oe6.scaffolds.sub.fa.gz

# calculate a signature for each scaffold
sourmash compute -k 4 \
    --scaled 1 \
    --track-abundance \
    --singleton \
    -o Oe6.scaffolds.sub.sig \
    Oe6.scaffolds.sub.fa.gz

# pairwise compare all scaffolds
sourmash compare -k 4 \
    -o Oe6.scaffolds.sub.comp \
    Oe6.scaffolds.sub.sig
```

Although `sourmash compare` supports export to a csv file, `sourmash` also has a built in visualization function, `plot`. We will use this to visualize scaffold similarity.

```
sourmash plot --labels \
    --vmin .4 \
    Oe6.scaffolds.sub.comp
```

In Figure 2, we see that there is high similarity between 98 of the scaffolds, but that Oe6_s01156 and Oe6_s01003 are outliers with tetranucleotide frequency similarity around 40% to olive scaffolds. These two scaffolds are contaminants[20].

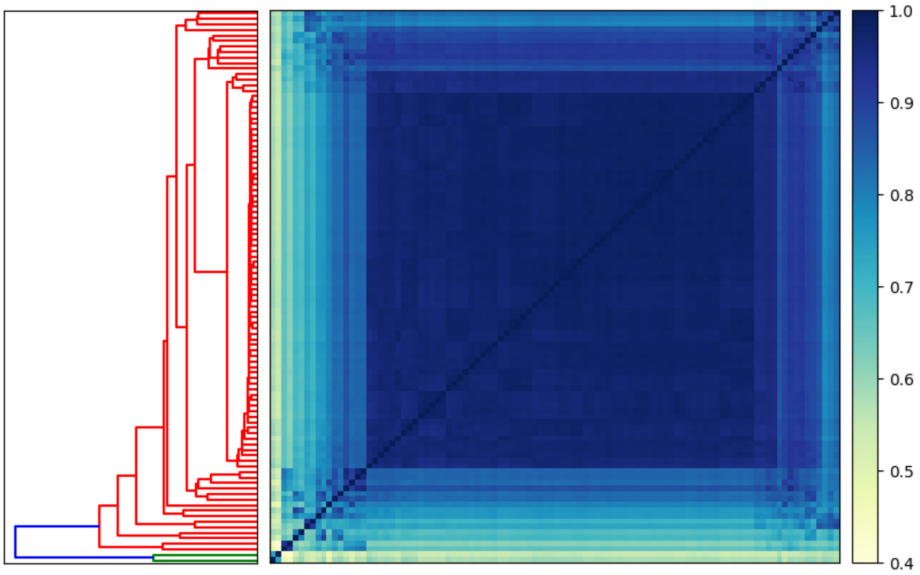

Figure 2. Heatmap and dendrogram generated using `sourmash` signatures built from scaffolds in the domesticated olive genome. Two scaffolds are outliers when using tetranucleotide frequency to calculate similarity (highlighted in green on the dendrogram).

Comparisons to detect outliers

MinHash comparisons are useful for outlier detection. Below we compare 50 genomes that contain the word "*Escherichia coli*." We have pre-calculated the signatures for each of these genomes. We then use the `plot` function to visualize our comparison.

```
# download the signatures into a folder
mkdir escherichia-sigs
cd escherichia-sigs

curl -L https://osf.io/pc76j/download \
    -o escherichia-sigs.tar.gz

# decompress the signatures
tar xzf escherichia-sigs.tar.gz
rm escherichia-sigs.tar.gz

cd ..

# pairwise compare the signatures
sourmash compare -k 31 \
    -o ecoli.comp \
    escherichia-sigs/*sig

# plot the comparison
sourmash plot --labels \
    ecoli.comp
```

We see that the minimum similarity in the matrix is 0%. If all 50 signatures were from the same species, we would expect to observe higher minimum similarity at a k-mer size of 31. When we look closely, we see one signature has 0% similarity with all other signatures because it is a phage (Figure 3).

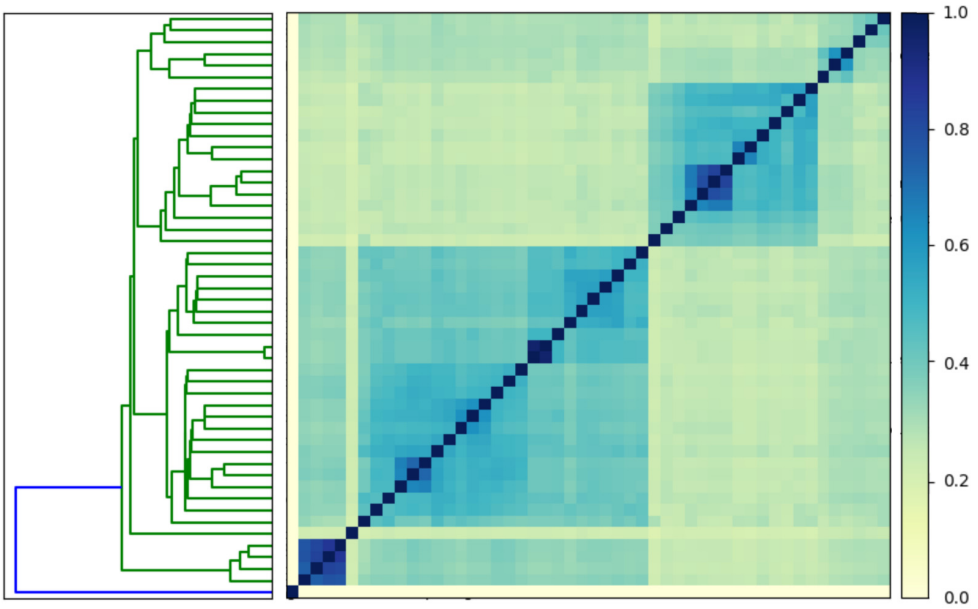

Figure 3. Heatmap and dendrogram generated using `sourmash` signatures built from 50 genomes that contained the word "*Escherichia coli*". One signature is an outlier (highlighted in blue on the dendrogram).

Classifying signatures

The `search` and `gather` functions allow the user to classify the contents of a signature by comparing it to a database of signatures. Prepared databases for microbial genomes in RefSeq and GenBank are available at https://sourmash.readthedocs.io/en/latest/databases.html. However, it is also simple to create a custom database with signatures.

Below we make a database that contains 50 *Escherichia coli* genomes.

```
mkdir escherichia-sigs
cd escherichia-sigs

curl -L https://osf.io/pc76j/download \
    -o escherichia-sigs.tar.gz

tar xzf escherichia-sigs.tar.gz
rm escherichia-sigs.tar.gz

cd ..

sourmash index -k 31 ecolidb \
    escherichia-sigs/*.sig
```

This database can be queried with `search` and `gather` using any signature calculated with a k-size of 31.

For example, below we download a small set of k-mer trimmed *Escherichia coli* reads and generate a signature with k=31.

```
curl -L -o ecoli-reads.khmer.fq.gz \
    https://osf.io/26xm9/download \

sourmash compute -k 31
    --scaled 2000 \
     ecoli-reads.khmer.fq.gz \
    -o  ecoli-reads.sig
```

Then, we search the 50-genome database created above.

```
sourmash search -k 31 \
     ecoli-reads.sig ecolidb \
     --containment
49 matches;   showing first 3:
similarity     match
----------     --------
 65.4%         NZ_JMGW01000001.1 Escherichia coli 1-176-05_S4_C2 e117605 ...
 64.9%         NZ_GG774190.1 Escherichia coli MS 196-1 Scfld2538,  whole  ...
 63.7%         NZ_JMGU01000001.1 Escherichia coli 2-011-08_S3_C2 e201108 ...
```

Breadth-first `sourmash search` finds all matches in the SBTMH that are present in the query signature (above a specified threshold).

Now try the same search using `sourmash gather`.

```
sourmash gather -k 31 \
    ecoli-reads.sig ecolidb

loaded query: ecoli_ref -5m.khmer.fq.gz ... (k=31, DNA)
loaded 1 databases.

overlap     p_query  p_match
---------    -------  -------
4.1 Mbp       65.4%     83.5%
NZ_JMGW01000001.1 Escherichia coli 1-...
2.4 Mbp        2.7%    3.3%
NZ_GG749254.1 Escherichia coli FVEC14...
3.4 Mbp        1.4%     1.7%
NZ_MOGK01000001.1 Escherichia coli st...
3.1 Mbp        0.6%     0.7%
NZ_LVOV01000001.1 Escherichia coli st...
3.1 Mbp        0.3%     0.4%
NZ_MIWP01000001.1 Escherichia coli st...
3.0 Mbp        0.3%     0.4%
NZ_APWY01000001.1 Escherichia coli 17...
3.5 Mbp        0.2%     0.2%
NZ_JNLZ01000001.1 Escherichia coli 3-...
4.0 Mbp        0.2%     0.2%
NZ_GG774190.1 Escherichia coli MS 196...
2.3 Mbp        0.1%     0.2%
NZ_KB732756 .1 Escherichia coli KTE66...
2.0 Mbp        0.1%     0.1%
NZ_BBUW01000001.1 Escherichia coli O1...
2.3 Mbp        0.1%     0.1%
NZ_MOZX01000101.1 Escherichia coli st...
2.3 Mbp        0.1%     0.1%
NZ_JSMW01000001.1 Escherichia coli st...
4.0 Mbp        0.1%     0.1%
NZ_JMGU01000001.1 Escherichia coli 2-...
2.0 Mbp        0.0%     0.0%
NZ_MOZC01000010.1 Escherichia coli st...
2.1 Mbp        0.0%     0.0%
NZ_MKJG01000001.1 Escherichia coli st...
1.8 Mbp        0.0%     0.0%
NZ_AEKA01000453.1 Escherichia sp. TW1...
2.6 Mbp        0.0%     0.0%
```

```
NZ_LEAD01000071.1 Escherichia coli st...
3.5 Mbp        0.0%     0.0%
NZ_MIWF01000001.1 Escherichia coli st...

found 18 matches total;
the recovered matches hit 71.5% of the query
```

Best-first `sourmash gather` finds the best match first, e.g. here the first *E. coli* genome has an 83% match to 65.4% of our query signature. The hashes that matched (65.4% of the query) are then subtracted, and the database is queried with the remaining hashes (34.6% of original query). This process is repeated until the threshold is reached. `sourmash` post-processes similarity statistics after the search such that it reports percent identity and unique identity for each match.

`sourmash gather` is also useful for rapid metagenome decomposition. Below we calculate a signature of a metagenome using raw reads, and then use `gather` to perform a best-first search against all microbial genomes in Genbank. This approach was recently used to classify unknown genomes in a "mock" metagenome[21]. The mock community was made to contain 64 genomes, but additional genomic material was inadvertently added prior to sequencing. Below we will use `gather` to investigate the content in the mock metagenome that did not map to the 64 reference genomes. For details on how this signature was created, please see Awad *et al.*[22]. Note that the GenBank database is approximately 7.8 Gb compressed, and 50 Gb decompressed. Searches of the current Gen-Bank database run fastest if allowed to use 11 Gb of RAM.

```
# download the signature
curl -L -o unmapped-qc-to-ref.fq.sig \
     https://osf.io//download \

# download the gather k 31 Genbank database
curl -L -o genbank-d2-k31.tar.gz \
     https:// s3-us-west-2.amazonaws.com/
     sourmash-databases/2018-03-29/
     genbank-d2-k31.tar.gz

# run gather
sourmash gather -k 31 \
     -o unmapped-qc-to-ref.csv \
     unmapped-qc-to-ref.fq.sig \
     genbank-d2-k31
```

The output to the terminal begins:

```
loaded query: unmapped-qc-to-ref.fq... (k=31, DNA)
downsampling query from scaled=10000 to 10000
loaded 1 databases.

overlap     p_query p_match
---------   ------- -------
1.6 Mbp        1.1%   19.9%
BA000019.2 Nostoc sp. PCC 7120 DNA, c...
1.2 Mbp        0.8%   50.8%
LN831027.1 Fusobacterium nucleatum su...
1.2 Mbp        0.8%   31.0%
CP001957.1 Haloferax volcanii DS2 pla...
1.1 Mbp        0.8%   16.7%
BX119912.1 Rhodopirellula baltica SH ...
1.0 Mbp        0.7%   29.0%
CH959311.1 Sulfitobacter sp. EE-36 sc ...
0.8 Mbp        0.6%   37.3%
AP008226.1 Thermus thermophilus HB8 g...
0.8 Mbp        0.6%   56.7%
```

```
CP001941.1 Aciduliprofundum boonei T4...
0.8 Mbp        0.5%   23.3%
FOVK01000036.1 Proteiniclasticum rumi...
0.7 Mbp        0.5%   15.0%
CP000031.2 Ruegeria pomeroyi DSS-3, c...
0.7 Mbp        0.5%   11.3%
CP000875.1 Herpetosiphon aurantiacus ...
0.6 Mbp        0.4%   22.9%
BA000023.2 Sulfolobus tokodaiistr.  7...
0.6 Mbp        0.4%   13.6%
AP009153.1 Gemmatimonas aurantiaca T-...
```

We see that 20.1% of k-mers match 82 genomes in GenBank. The majority of matches are to genomes present in the mock community. However, some species like *Proteiniclasticum ruminis* were not members of the mock community. These results also highlight how `sourmash gather` behaves with inexact matches such as strain variants. For example, we see two matches between *P. ruminis* strains among all matches. This likely indicates that a *P. ruminis* strain that has not been sequenced before is in our sample, and that it shares more k-mers of size 31 in common with one strain than the other. (See Brown CT *et al.*[23] for further analysis of this strain.)

`sourmash gather` and `search` also support custom databases. Using a custom database with `sourmash gather`, we can identify the dominant contamination in the domesticated olive genome[20]. Below, we will use a database containing all fungal genomes in NCBI. We will then use the streaming compatibility of `sourmash` to download and calculate the signature. Lastly, we will search the olive genome against the fungal genomes using gather.

```
# download the fungal database
curl -L -o fungi-genomic.tar.gz \
    https://osf.io/7yzc4/download

# decompress the database
tar xf fungi-genomic.tar.gz

# download the olive genome
# calculate the signature
curl -L https://osf.io/k9358/download \
  | zcat \
  | sourmash compute -k 31 \
  --scaled 2000 --track-abundance \
  -o Oe6.scaffolds.sig -

# perform gather
sourmash gather -k 31 \
    --scaled 2000 \
    -o Oe6.scaffolds.csv \
    Oe6.scaffolds.sig \
    fungi-k31
```

Using gather, we see two matches both within the genus *Aureobasidium*. This is the dominant contaminant within the genome[20].

```
loaded query: Oe6.scaffolds.fa... (k=31, DNA)
loaded 1 databases.


overlap     p_query p_match avg_abund
---------   ------- ------- ---------
140.0 kbp      0.0%    1.0%       1.2
LVWM01000001.1 Aureobasidium pullulan...
68.0 kbp       0.0%    0.1%       1.0
```

```
MSDY01000045.1 Aureobasidium sp. FSWF...
found less than 30.0 kbp in common. => exiting

found 2 matches total;
the recovered matches hit 0.0% of the query
```

Conclusions

The `sourmash` package provides a collection of tools to conduct sequence comparisons and taxonomic classification, and makes comparison against large-scale databases such as GenBank and SRA tractable on laptops. `sourmash` signatures are small and irreversible, which means they can be used to facilitate pre-publication data sharing that may help improve classification databases and facilitate comparisons among similar datasets.

Data availability

Underlying data

Open Science Framework: sourmash-use-cases. https://doi.org/10.17605/OSF.IO/KESH2

This project contains the following underlying data:

- data-files
  - ecoli-reads.khmer.fq.gz (Small set of k-mer trimmed *Escherichia coli* reads)
  - ERR458584.fq.gz (*Saccharomyces cerevisiae* SRA Record ERR458584 SNF2 mutant,[15])
  - Oe6.scaffolds.fa.gz (domesticated olive (*Olea europaea*) genome[20])
  - Oe6.scaffolds.sub.fa.gz (Subsampled set of scaffolds from the domesticated olive (*Olea europaea*) genome[20])
  - yeast_ktrimmed.tar (kmer-trimmed *Saccharomyces cerevisiae* reads[15])
- index-files
  - escherichia-sigs.tar.gz (Sourmash signatures of 50 randomly selected *Escherichia coli* genomes)
  - fungi-genomic.tar.gz (Sourmash signature database of all fungal genomes in NCBI as of 12/2018)
- signature-files
  - GCF_000005845.2_ASM584v2_genomic.fna.gz (*Escherichia coli* genome str. K-12 substr. MG1655)
  - GCF_000146045.2_R64_genomic.fna.gz (*Saccharomyces cerevisiae* S288C genome)
  - GCF_000146045.2_R64_protein.faa.gz (*Saccharomyces cerevisiae* S288C protein sequence)

Extended data

Open Science Framework: sourmash-use-cases. https://doi.org/10.17605/OSF.IO/KESH2

This project contains the following extended data:

- yeast-mds.txt (Code to generate MDS plots via Salmon and edgeR)

Software availability

Source code available from: https://github.com/ dib-lab/sourmash/

Archived source code at time of publication: http://doi.org/10.5281/zenodo.3240653

Licence: 3-Clause BSD License

Grant information

This work is funded in part by the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative [GBMF4551 to CTB]. NTP was supported by a National Science Foundation Postdoctoral Fellowship in Biology [1711984].

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

1. **Sequence read archive overview.** 2018.
Reference Source
2. Broder AZ: **On the resemblance and containment of documents.** In *Compression and complexity of sequences 1997. proceedings.* IEEE. 1997; 21–29.
Reference Source
3. Ondov BD, Treangen TJ, Melsted P, *et al.*: **Mash: fast genome and metagenome distance estimation using MinHash.** *Genome Biol.* 2016; **17**(1): 132.
PubMed Abstract | Publisher Full Text | Free Full Text
4. Bovee R, Greenfield N: **Finch: a tool adding dynamic abundance filtering to genomic minhashing.** 2018; **3**(22): 505.
Publisher Full Text
5. Zhao XF: **BinDash, software for fast genome distance estimation on a typical personal laptop.** *Bioinformatics.* 2019; **35**(4): 671–673.
PubMed Abstract | Publisher Full Text
6. Rowe WP, Carrieri AP, Alcon-Giner C, *et al.*: **Streaming histogram sketching for rapid microbiome analytics.** *Microbiome.* 2019; **7**(1): 40.
PubMed Abstract | Publisher Full Text | Free Full Text
7. Koslicki D, Zabeti H: **Improving minhash via the containment index with applications to metagenomic analysis.** *Appl Math Comput.* 2019; **354**: 206–215.
Publisher Full Text
8. **Mash screen: What's in my sequencing run?** 2017.
Reference Source
9. Brown CT, Irber L: **sourmash: a library for MinHash sketching of DNA.** *J Open Source Softw.* 2016; **1**(5): 27.
Publisher Full Text
10. Solomon B, Kingsford C: **Fast search of thousands of short-read sequencing experiments.** *Nat Biotechnol.* 2016; **34**(3): 300–2.
PubMed Abstract | Publisher Full Text | Free Full Text
11. Matsakis ND, Klock FS II: **The rust language.** *Ada Lett.* 2014; **34**(3): 103–104.
Publisher Full Text
12. Wood DE, Salzberg SL: **Kraken: ultrafast metagenomic sequence classification using exact alignments.** *Genome Biol.* 2014; **15**(3): R46.
PubMed Abstract | Publisher Full Text | Free Full Text
13. Crusoe MR, Alameldin HF, Awad S, *et al.*: **The khmer software package: enabling efficient nucleotide sequence analysis [version 1; peer review: 2 approved, 1 approved with reservations].** *F1000Res.* 2015; **4**: 900.
PubMed Abstract | Publisher Full Text | Free Full Text
14. Conesa A, Madrigal P, Tarazona S, *et al.*: **A survey of best practices for RNA-seq data analysis.** *Genome Biol.* 2016; **17**(1): 13.
PubMed Abstract | Publisher Full Text | Free Full Text
15. Schurch NJ, Schofield P, Gierliński M, *et al.*: **How many biological replicates are needed in an RNA-seq experiment and which differential expression tool should you use?** *RNA.* 2016; **22**(6): 839–51.
PubMed Abstract | Publisher Full Text | Free Full Text
16. Patro R, Duggal G, Love MI, *et al.*: **Salmon provides fast and bias-aware quantification of transcript expression.** *Nat Methods.* 2017; **14**(4): 417–419.
PubMed Abstract | Publisher Full Text | Free Full Text
17. Robinson MD, McCarthy DJ, Smyth GK: **edger: a bioconductor package for differential expression analysis of digital gene expression data.** *Bioinformatics.* 2010; **26**(1): 139–140.
PubMed Abstract | Publisher Full Text | Free Full Text
18. Pride DT, Meinersmann RJ, Wassenaar TM, *et al.*: **Evolutionary implications of microbial genome tetranucleotide frequency biases.** *Genome Res.* 2003; **13**(2): 145–158.
PubMed Abstract | Publisher Full Text | Free Full Text
19. Albertsen M, Hugenholtz P, Skarshewski A, *et al.*: **Genome sequences of rare, uncultured bacteria obtained by differential coverage binning of multiple metagenomes.** *Nat Biotechnol.* 2013; **31**(6): 533–538.
PubMed Abstract | Publisher Full Text
20. Reiter T, Brown CT: **Microbial contamination in the genome of the domesticated olive.** 2018.
Publisher Full Text
21. Shakya M, Quince C, Campbell JH, *et al.*: **Comparative metagenomic and rRNA microbial diversity characterization using archaeal and bacterial synthetic communities.** *Environ Microbiol.* 2013; **15**(6): 1882–1899.
PubMed Abstract | Publisher Full Text | Free Full Text
22. Awad S, Irber L, Brown CT: **Evaluating metagenome assembly on a simple defined community with many strain variants.** 2017.
Publisher Full Text
23. Brown CT, Moritz D, O'brien M, *et al.*: **Exploring neighborhoods in large metagenome assembly graphs reveals hidden sequence diversity.** *BioRxiv.* 2019; 462788.
Publisher Full Text

Open Peer Review

Current Peer Review Status: ✔ ✔

Version 1

Reviewer Report 02 September 2019

https://doi.org/10.5256/f1000research.21579.r52218

✔ **Rayan Chikhi**
C3BI USR 3756, CNRS (French National Center for Scientific Research), Institut Pasteur, Paris, France

The authors present sourmash 2, a tool that implements a novel combination of SBTs and MinHashes, which are both fascinating computational concepts; thus, their mix is quite an interesting one. Sourmash 2 enables to perform large-scale sequences-vs-database similarity searches. The article offers a comprehensive guide for many of the software features, with biologically relevant scenarios. This is a useful contribution that is highly relevant to current needs in biology. There are a few technical issues with the current manuscript version that I list below. But otherwise, most of my remarks are for adding some extra perspective. I believe the manuscript can be approved after the technical fixes.

Major remarks:

1. A quick recap of the state of the art in containment search would be helpful. Here you claim to use ‘a modulo approach’. Mash screen and containment minhash use different approaches (see e.g. the blog post of ‘Mash screen’). It would be nice if, in this paper, the usage of the modulo approach was put into perspective compared to those two aforementioned methods.

2. In fact, in the blog post cited as reference 8, Ondov writes that “the modulo approach is problematic for metagenomic applications (e.g. finding a virus in a metagenome).” The problem is indirectly mentioned in the manuscript (“can sacrifice some of the memory and storage benefits of standard MinHash techniques, as the signature size scales with the number of unique k-mers”). It would be neat to get the authors’ comparative perspective here as to why using modulo is the better approach.

3. My main comment would perhaps be the lack of comparison with other software. I do not know if this is a requirement for F1000Research in the “Software Tool Article” category. I suppose that sourmash is the only tool that implements SBT-Minhashes, so of course here there is no competitor in that category. It would however be nice to have some indication on whether sourmash is best-in-class in each of the proposed features (the uses cases), or whether other tools already exist and somehow do a similar job. And, the other way around, which areas where

sourmash is really the only tool capable of doing X in reasonable time. I do not expect a comprehensive benchmark, but some informal indication would already be appreciated.

4. What are roughly the limits of similarity queries? E.g. sequences shorter than X or having identity below Y% have no chances to be reported.

5. A summary of all the features demonstrated in the main text could be helpful. For instance, reading only the introduction, it is not explicit that a natural application of sourmash is outlier/contaminant detection.

Minor remarks:

1. “Sequence Read Archive now contains over 20 Petabases of data1”: seems to be over 30 PB in 2019 according to the plot in reference 1.

2. It is not clear what the ‘LCA’ term stands for in the context of the database format introduced here. Is it the lowest common ancestor?

3. The description of LCA (in section “LCA database”) is imprecise. What does a “named list” mean in this context? A Figure would be helpful to see a small example.

4. A sentence in the manuscript mentions ‘a second database format’. The ‘first format’ is supposedly the SBTMH but it is only implicit that SBTMH is a ‘format’.

5. The introduction mentions a bunch of features implemented in other tools (“Jaccard similarity comparisons, .., k-mer abundance comparisons, decrease runtime and memory requirements, and work on streaming input data.”) Are all of these implemented in sourmash2, or only a subset of them? (It seems to me that most are implemented.)

6. The description of the modulo approach used is imprecise. How is the hash space divided into s equal ‘bands’ (undefined term), precisely? Also, I suppose this somewhat different from the modulo approach proposed by Broder, and clarified in Mash screen’s blog post, but how so?

7. The concept of ‘hash subset retention’ is not well defined. I suppose it is the set of hashes that result from a MinHash computation.

8. Abundance filtering (as in Finch) is not performed in sourmash2, right?

9. Some of the ‘signature utilities’ are self-explanatory. However, what is the difference between ‘intersect’ and ‘overlap’? What is ‘flatten’?

10. Regarding the sentence: “although we recommend using more accurate approaches for detailed comparisons.” To make the paper self-contained, a short explanation would be needed to delineate what sort of concrete use-case(s) is/are meant behind the term ‘detailed comparisons’.

11. The “similarity queries” and “containment queries” sections could benefit from at least one use-case example per query. This is to illustrate the two sections, which are a bit obscure without examples. (I realize that use cases are given later in the codes examples, so perhaps a

forward-reference could work, albeit less elegant.) A proposal for similarity queries: 'find all genomes in the SBTMH (leaves) that are similar to a query genome'.

12. Awkward formulation of that part of the sentence: [..] 'using a k-mer size of 4, use all 4-mers, and track abundance.' (although I understood that the k-mer size was 4 and the 'use all k-mers' refers to a scaling factor of 1)

13. The "Tetranucleotide Frequency Clustering" section is quite nice. It should be emphasized however this isn't really a minhash sketch: all 4-mers are considered.

14. Regarding the sentence "We see that the minimum similarity in the matrix is 0%", how is that seen? visual inspection of ecoli.comp.matrix.png?

15. Regarding the sentence "This process is repeated until the threshold is reached.": I forgot.. which threshold?

16. Regarding the sentence "We see that 20.1% of k-mers match 82 genomes in GenBank.": how is this seen? Also "we see two matches between P. ruminis strains among all matches." In the output above that text, I see only one match. (I could not test that section due to the missing download URL.)

Regarding the software commands:

- As an important note, one cannot easily copy-paste the command lines as short dashes (-) are converted to long dashes (–). Nevertheless, I still automatically replaced all the dashes and tested all command lines. I'll report any problem below.

- Extra '\' at the end of the command: "curl –L –o ecoli–reads.khmer.fq.gz https://osf.io/26xm9/download \".

- Missing url in command (and also extra '\' at the end): "[..] unmapped–qc–to–ref.fq.sig https://osf.io//download \" Thus I could not test this part.

- The streaming operation at the beginning of the MDS section overwrites the file 'ERR458584.khmer.sig' produced before, perhaps make a note of that.

- The url "curl –L –o genbank–d2–k31.tar.gz \
  https:// s3–us–west–2.amazonaws.com/
  sourmash–databases/2018–03–29/
  genbank–d2–k31.tar.gz" has extra new lines

- In the command "sourmash index –k 31 ecolidb \
  escherichia–sigs /*.sig", a space is wrongly inserted after "escherichia-sigs"

- And also, in the command that follows, one of the '\' is extra and another '\' is missing.

- The SBT path inside the file 'fungi-k31.sbt.json' is wrongly hardcoded. Also, when fixing it, I get "WARNING: this is an old index version, please run `sourmash migrate` to update it." Although it did end up working.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Algorithms and data structures for sequence bioinformatics

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Report 27 August 2019

https://doi.org/10.5256/f1000research.21579.r52588

Brad Solomon
Department of Computer Science, Whiting School of Engineering, Johns Hopkins University, Baltimore, MD, USA

This manuscript presents an improved software library, sourmash 2.0, for the efficient construction and analysis of MinHash sketches of genomic and proteomic sequence data. The primary innovations of sourmash 2.0 are the novel applications and modified implementations of several existing sketching and indexing methods. In particular, the two main advancements are (1) a ‘modulo approach’ to sketch construction and the development of a modified Sequence Bloom Tree (SBT) index over MinHash sketches. In addition, unlike conventional MinHash methods, sourmash 2.0 can also track the abundance of retained hash elements, allowing some degree of abundance comparison and abundance estimation of genomic datasets.

Multiple use cases for sourmash 2.0 are provided including sketch construction, similarity comparisons,

containment querying, classification, and metagenome decomposition. The manuscript is very thorough in describing sketch construction with example run commands for full-length genomes, proteomes, as well as raw reads. The ease of use is further demonstrated with a single line command that, while involving multiple pipes, is capable of constructing a signature through a download data stream. Similarly, several uses of sketch comparisons are demonstrated including visualization and outlier detection. Lastly the use of the SBT MinHash (SBTMH) is demonstrated for classification or containment of arbitrary sequence queries. The download instructions and example run commands are fully functional and the input datasets are reasonably sized for toy examples (most on the order of <100 MB).

Excluding the description of the modulo approach of sketch construction, the manuscript itself is technically sound. The topic is high impact -- sketching approaches are increasingly popular solutions to a multitude of research topics in computational biology. Many of these potential use cases are demonstrated successfully in the manuscript. That said, there are numerous existing solutions to each of the use cases presented in the manuscript and little to no attempt was made to provide benchmarking information or to demonstrate the improvements sourmash 2.0 has over its competitors. While sourmash 2.0 has an ease of use that will undoubtedly facilitate use, it does not adequately demonstrate an improved capacity for analysis over these other tools. This is primarily a suggestion for improvement as, based on the F1000 Research guidelines, the manuscript is correct and valid in its current state.

Major Comments:

1. The ‘modulo approach’ for sketch construction, despite being one of the main innovations of the method, is particularly unclear in the manuscript. The cited literature (Broder 1997) describes an approach that sub-samples hash values based on a modulo factor to address the inherent weakness of a Minhash in a mixture of several distinct components. However the description of the sourmash implementation instead describes splitting the hash space into ‘equal bands’ and selecting only the minimum band. As the existing modulo approach has no guarantees on equal-sized (or even equal-fraction as the manuscript claims elsewhere) sub-sampling, this appears to be a novel and significant contribution to the field. However there are no details that explain (1) how the hash space is divided, (2) how the minimum band is selected, and (3) how downsampling is performed.

2. Sourmash 2.0 is motivated by “a particular focus towards enabling efficient containment queries using large databases”. However the manuscript does not include any true comparisons about sourmash’s performance against existing tools, alternative approaches, or benchmarking information for even conventionally sized datasets. This greatly limits the potential impact of sourmash given there are many competing sketch strategies and an even larger range of available implementations.

   While it is unreasonable to expect a full review of the available methods, the inclusion of even a single ‘large-scale’ dataset in the test set or use cases would go a long way towards demonstrating the scalability of sourmash. Selecting a biologically relevant subset from a public genomic repository such as the NIH SRA, TCGA, or GTEx (to name just a few) would alleviate the need to host such a dataset while allowing large-scale reproducibility and benchmarking.

Minor Comments:

1. The ‘Salmon & edgeR’ MDS plot in Figure 1 does not have points associated with the text labels. As there is no consistency in the placement of labels versus nodes in the first plot (Sourmash Compare MDS), even approximate values are difficult to determine.

2. The commands listed in the manuscript are using the wrong character ('–' vs '-'). None of them run properly without manual adjustments or retyping. I imagine this is a formatting issue more than a coding one but it would make reproducing the results a lot simpler if it was resolved.

Other:

1. The inclusion of limited abundance information is a particularly interesting improvement over standard MinHash sketches. The manuscript suggests that the abundance tracking can play a significant role when 'comparing many signatures' but there is no concrete claim to assess. While outside the scope or focus of this work, a follow-up piece which explores the theoretical or practical impact of systematically sub-sampled counting information would be potentially high impact.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Computational Biology, Algorithms and Data Structures, Genomics

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.