SOFTWARE TOOL ARTICLE

Large-scale sequence comparisons with sourmash [version 1; peer review: awaiting peer review]

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

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

id Equal contributors

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.

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.

Copyright: © 2019 Pierce NT et al. This is an open access article distributed under the terms of the Creative Commons Attribution Licence, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Pierce NT, Irber L, Reiter T et al. Large-scale sequence comparisons with sourmash [version 1; peer review: awaiting peer review] 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, there is great need for methods to quickly and efficiently conduct similarity searches on a massive scale. MinHash techniques 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. 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, it has since been modified and extended to enable k-mer abundance comparisons, decrease runtime and memory requirements, and work on streaming input data. Furthermore, as Jaccard similarity is impacted by the relative size of the sets being compared, containment searches 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, 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, and (2) implementing a modified Sequence Bloom Tree 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., 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. 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. 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 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, 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\(^8\)), 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.

**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\(^1\)), 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](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\(^3\), 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 \
```
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.

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. 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. To use sourmash to build an MDS plot, we first trim the data to remove low abundance k-mers via khmer. 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.

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.

We can also check that the name has been changed.
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.

```bash
# 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.

```r
# 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,
     col = c("blue", "blue",
             "orange", "orange"))
```

For comparison, we also produced an MDS plot using a more traditional approach, utilizing Salmon (v0.11.3)\(^7\) to quantify abundance relative to an *S. cerevisiae* reference, and edgeR (v3.22.5)\(^7\) 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\(^8\). This is often used in metagenomics as one method to “bin” assembled contiguous sequences together that are from the same species\(^8\). Recently, tetranucleotide frequency clustering using `sourmash` was used to detect microbial contamination in the domesticated olive genome\(^8\). 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.)
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 \n  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. 

**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.
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:

<table>
<thead>
<tr>
<th>similarity</th>
<th>match</th>
</tr>
</thead>
<tbody>
<tr>
<td>65.4%</td>
<td>NZ_JMGW01000001.1 Escherichia coli 1-176-05_S4_C2 e17605 ...</td>
</tr>
<tr>
<td>64.9%</td>
<td>NZ_GG774190.1 Escherichia coli MS 196-1 Scf1d2538, whole ...</td>
</tr>
<tr>
<td>63.7%</td>
<td>NZ_JMGU01000001.1 Escherichia coli 2-011-08_S3_C2 e201108 ...</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>overlap</th>
<th>p_query</th>
<th>p_match</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Mbp</td>
<td>65.4%</td>
<td>83.5%</td>
</tr>
<tr>
<td>NZ_JMGW01000001.1 Escherichia coli 1-...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4 Mbp</td>
<td>2.7%</td>
<td>3.3%</td>
</tr>
<tr>
<td>NZ_GG749254.1 Escherichia coli FVEC14...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4 Mbp</td>
<td>1.4%</td>
<td>1.7%</td>
</tr>
<tr>
<td>NZ_MOGK01000001.1 Escherichia coli st...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1 Mbp</td>
<td>0.6%</td>
<td>0.7%</td>
</tr>
<tr>
<td>NZ_LVOV01000001.1 Escherichia coli st...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1 Mbp</td>
<td>0.3%</td>
<td>0.4%</td>
</tr>
<tr>
<td>NZ_MIWP01000001.1 Escherichia coli st...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.0 Mbp</td>
<td>0.3%</td>
<td>0.4%</td>
</tr>
<tr>
<td>NZ_APWY01000001.1 Escherichia coli 17...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.5 Mbp</td>
<td>0.2%</td>
<td>0.2%</td>
</tr>
<tr>
<td>NZ_JNLI01000001.1 Escherichia coli 3-...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.0 Mbp</td>
<td>0.2%</td>
<td>0.2%</td>
</tr>
<tr>
<td>NZ_GG774190.1 Escherichia coli MS 196...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3 Mbp</td>
<td>0.1%</td>
<td>0.2%</td>
</tr>
<tr>
<td>NZ_KB732756 .1 Escherichia coli KTE66...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.0 Mbp</td>
<td>0.1%</td>
<td>0.1%</td>
</tr>
<tr>
<td>NZ_BBUW01000001.1 Escherichia coli Ol...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3 Mbp</td>
<td>0.1%</td>
<td>0.1%</td>
</tr>
<tr>
<td>NZ_MOZX01000101.1 Escherichia coli st...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3 Mbp</td>
<td>0.1%</td>
<td>0.1%</td>
</tr>
<tr>
<td>NZ_JSMW01000001.1 Escherichia coli st...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.0 Mbp</td>
<td>0.1%</td>
<td>0.1%</td>
</tr>
<tr>
<td>NZ_JMGU01000001.1 Escherichia coli 2-...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.0 Mbp</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>NZ_MOZC01000001.1 Escherichia coli st...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1 Mbp</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>NZ_MKJG01000001.1 Escherichia coli st...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.8 Mbp</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>NZ_AEKA01000453.1 Escherichia sp. TW1...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.6 Mbp</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>
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\(^1\). 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.*\(^2\). Note that the GenBank database is approximately 7.8 Gb compressed, and 50 Gb decompressed. Searches of the current GenBank database run fastest if allowed to use 11 Gb of RAM.

```bash
# 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.
```

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

---

\(^1\) F1000Research 2019, 8:1006 Last updated: 04 JUL 2019

---
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.*\textsuperscript{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\textsuperscript{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.

```bash
# 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\textsuperscript{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
```
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](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)
  - Oe6.scaffolds.fa.gz (domesticated olive (*Olea europaea*) genome)
  - Oe6.scaffolds.sub.fa.gz (Subsampled set of scaffolds from the domesticated olive (*Olea europaea*) genome)
  - yeast_ktrimmed.tar (kmer-trimmed *Saccharomyces cerevisiae* reads)

- **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](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)

Data are available under the terms of the Creative Commons Zero “No rights reserved” data waiver (CC0 1.0 Public domain dedication).

Software availability

Source code available from: [https://github.com/dib-lab/sourmash/](https://github.com/dib-lab/sourmash/)


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

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com