Published on Feb 21, 2023 by Arcadia Science **DOI**: 10.57844/arcadia-1896-ke33

A new R package, sourmashconsumr, for analyzing and visualizing the outputs of sourmash

The sourmash Python package produces many outputs that describe the content and similarity of sequencing data. We developed a new R package, sourmashconsumr, that lets a wider range of users easily load, analyze, and visualize those outputs in R.

Contributors (A-Z)

Adair L. Borges, Seemay Chou, Rachel J. Dutton, Megan L. Hochstrasser, Elizabeth A. McDaniel, Taylor Reiter, Emily C.P. Weiss

Version 4 · Mar 31, 2025

Purpose

Bioinformatics tools are written in many software languages and produce varied outputs. This is in part because different software languages excel at different tasks — for example, Python is good at text parsing, while R is good at statistics and visualization. The variation in the tool space creates barriers to use due to the required language-specific knowledge.

Sourmash is a Python package that facilitates quick insights into sequencing data through rapid comparisons. To take advantage of visualization and statistical analysis tools, it is often helpful to analyze the outputs of this tool in R. We built sourmashconsumr to make this easier in our own work and for others comparing large amounts of sequencing data, doing metagenomics, or doing other sequencing quality control. This package provides a series of parsing functions to bring the sourmash output files into R and features visualization and analysis functions commonly used to interpret sequencing data.

- This pub is part of the **project**, “Useful computing at Arcadia.” Visit the project narrative for more background and context.
- The sourmashconsumr R **package** is available in this GitHub repository. **Documentation** for the package is available here.
- All **code associated with figures and validation** is available in this GitHub repository.

Why it matters

We often use sequencing data to help answer biological questions and generate new hypotheses. One of the goals of Arcadia’s software team is to empower biologists to analyze their own data, and one way to achieve that is by packaging code that accomplishes routine tasks so that it’s easier for diverse scientists to jump into their own data.

In this pub, we’re sharing a tool that allows anyone working with sequencing data to analyze sourmash outputs in R, a language that excels at statistical analysis and visualization. We thought it would be especially helpful to include a standard set of go-to visualizations that we wouldn’t need to build from scratch every time we analyze a new dataset. With this package, we anticipate that scientists with a wider range of computational literacy will be able to directly play with their data, giving them more ownership and creativity.

The problem

Some biological computing tools, statistical methods, or visualizations are only available as R packages **[1][2][3]**. To access these methods, we need to load and parse biological data into the appropriate, often tool-specific, format.

Sourmash is a Python package that quickly compares potentially very large sets of DNA and protein sequences **[4]**. This functionality can be used to, for example, cluster transcriptomes **[5]** or genomes **[6]**, to identify the taxonomy of new isolate or metagenome-assembled genomes **[7]**, or to determine the taxonomic composition of a new metagenome sequence by comparing it against a database of reference genomes **[8]**. Sourmash outputs text files in JSON and CSV format that contain information about the sequences themselves or about the similarity between a sequence and other sequences. These text files can be used for many downstream applications, including visualization of pairwise sample similarity **[5]** or taxonomic composition **[8]**, machine learning **[9]**, differential abundance analysis **[10]**, or to estimate pangenomes **[11]**.

The process of transforming the output of a sourmash command into a downstream analysis requires a substantial amount of code, much of which can be standardized between use cases (e.g., parsing). The sourmash Python API provides this functionality in the Python language, thereby lowering the bar for analysis of these outputs in Python. We sought a similar code base that would let us analyze sourmash outputs in R.

Our solution

We wrote an R package, sourmashconsumr, which provides parsing, visualization, and analysis functions to operate on the outputs of the sourmash Python package in R.

The resource

Below, we describe the main functionalities encoded in the sourmashconsumr R package (Figure 1). In addition to this overview, the package itself contains function documentation and a vignette that demonstrates how to use the code.

The sourmashconsumr R **package** is available at this GitHub repository (DOI: 10.5281/zenodo.7591833) under an MIT license. All **code associated with figures and validation** is available in this GitHub repository (DOI: 10.5281/zenodo.7591845).

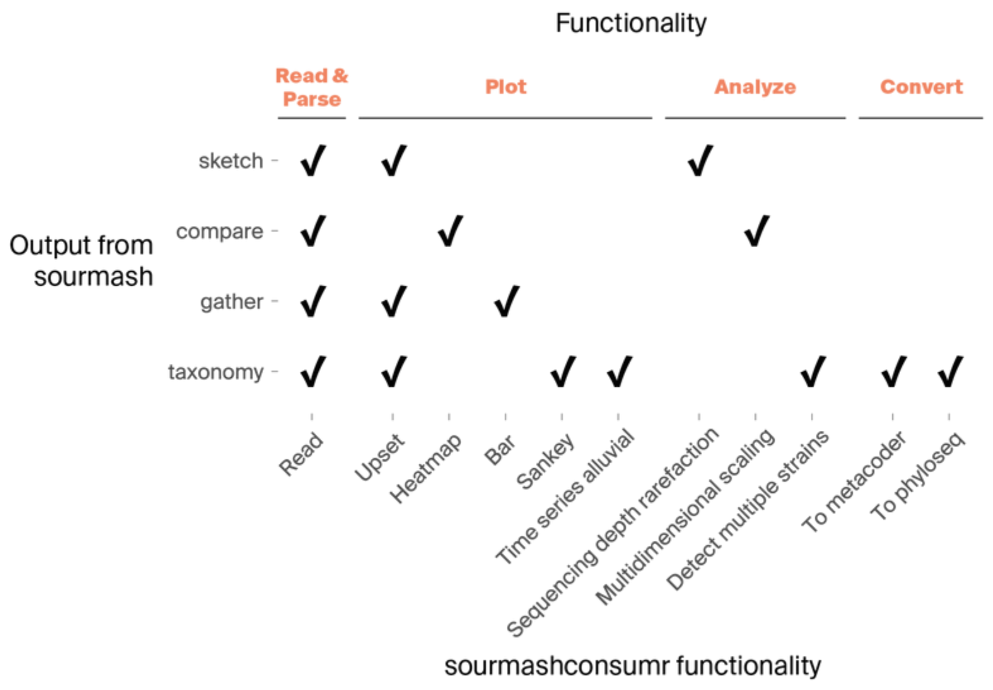

Figure 1

Overview of functionality encoded by the sourmashconsumr R package.

The sourmashconsumr R package operates on text files output by specific commands in the sourmash Python package (y-axis). It works with the output of `sourmash sketch` (JSON), `compare` (CSV), `gather` (CSV), and `taxonomy annotate` (CSV). At a high level, the functions in the sourmashconsumr package either read and parse, plot, analyze, or convert the outputs of these commands. The x-axis summarizes the functionality encoded by the sourmashconsumr package, and the check marks designate which sourmash output files the functionality applies to.

Reading sourmash outputs into R

The sourmashconsumr package provides parsing functions to read the output of `sourmash sketch`, `compare`, `gather`, and `taxonomy annotate` into R as tidy data frames. These function names begin with `read*`. Each function takes the path to one or many files or URLs and returns a single data frame. These data frames can then be used to further analyze or visualize the input data using other code or R packages.

Visualizing the outputs of sourmash

The majority of functions in the sourmashconsumr package implement common visualizations for each of the four output types that the `read*` functions parse. These function names begin with `plot*`. When possible, we encode visualizations as ggplot2 objects so that the user can add additional layers to control the plot aesthetics **[12]**.

Below, we show some of the outputs of these functions using the datasets built into sourmashconsumr: six stool microbiome shotgun metagenome samples from the Integrated Human Microbiome Project Inflammatory Bowel Disease cohort **[13]**, a longitudinal survey of the emergence of inflammatory bowel disease (IBD). The samples in the example data are all starting samples taken from different individuals. All individuals were symptomatic at this initial time point, but three individuals were diagnosed with Crohn’s disease (CD), a type of IBD, by the end of the year, and three individuals were not (Non-IBD).

SRA accession number	Group
SRR5936131	CD
SRR5947006	CD
SRR5935765	CD
SRR5936197	Non-IBD
SRR5946923	Non-IBD
SRR5946920	Non-IBD

Table 1

Example dataset distributed with the sourmashconsumr package.

Signatures

Sourmash signatures contain one or multiple sub-sampled representations of DNA or protein sequences (FracMinHash sketches) **[14][15]**. Each FracMinHash sketch contains hashes (and optionally, hash abundances) that represent a subset of k-mers from the original sequences. The `sourmash sketch` command consistently subsamples k-mers across different sequences, so we can compare (e.g., intersect) sketches to understand sequence similarity. While the command line function `sourmash compare` estimates similarity and containment metrics, sourmashconsumr introduces upset plots that operate directly on sets of signatures read into R with the `read_signature()` function (Figure 2). These plots operate directly on the hashes in the FracMinHash sketch (e.g., subsampled k-mers) and so can help build intuition for the amount of sequence shared between sets of samples. Upset plots can work on any signatures but are most meaningful when applied to sketches built with the same k-mer size and scaled value. In addition to upset plots built from signatures that represent single sequencing libraries, the sourmash command line function `sourmash sig combine` can combine signatures (e.g., from the same group; cases vs. controls) prior to reading and plotting the signatures with sourmashconsumr **[9]**.

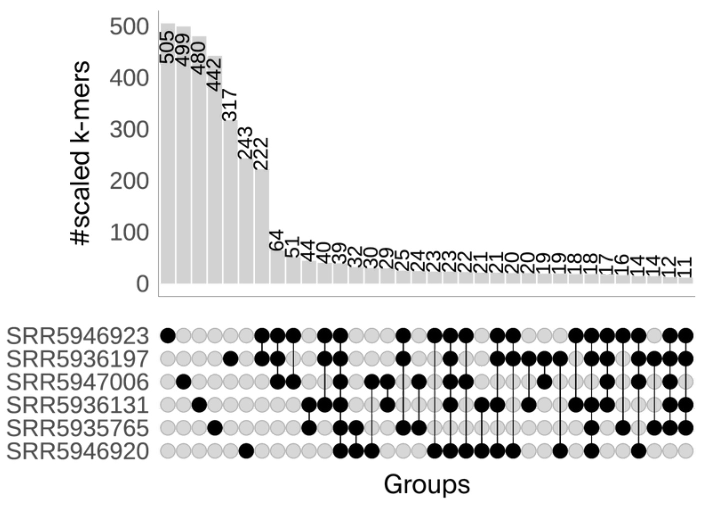

Figure 2

An upset plot built from sourmash signatures representing stool microbiome shotgun metagenomes.

Upset plots are alternatives to venn diagrams that show the intersection size between different groups of samples. In the above plot, we see that the majority of k-mers present in any metagenome are distinct to that metagenome. Groups with fewer than 10 shared k-mers are not displayed.

Compare

`sourmash compare` estimates pairwise similarity or containment between many signatures. The sourmashconsumr package provides functions to plot the CSV output as a clustered heatmap (Figure 3, A) or to process the CSV via multidimensional scaling (MDS) and produce a plot (Figure 3, B).

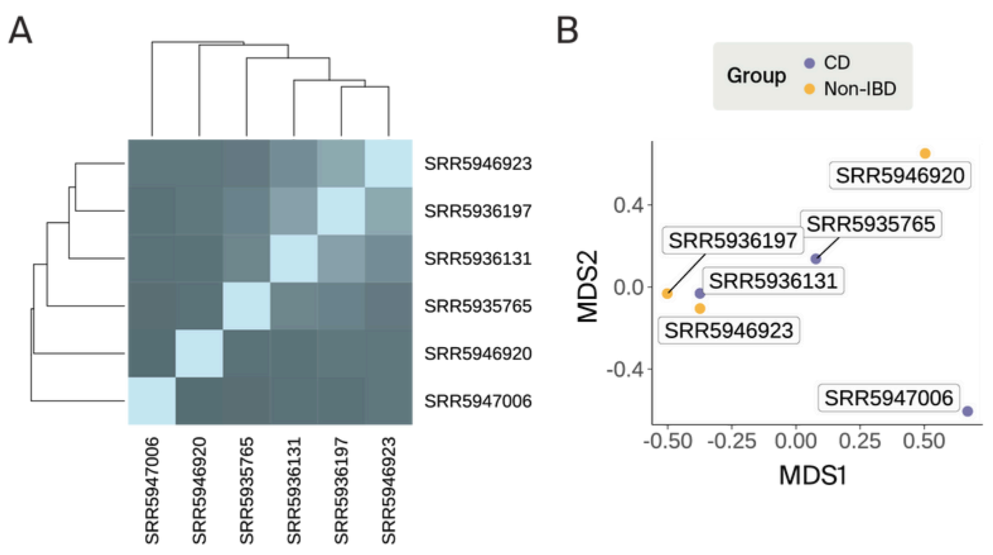

Figure 3

Visualizations that sourmashconsumr produced from the output of `sourmash compare` **.**

(A) Clustered heatmap representing sample similarity. Lighter colors represent higher similarity with the diagonal axis representing one (100% similar).

(B) Multidimensional scaling plot where points that are more similar are grouped more closely together. The groups refer to stool microbiome metagenome samples of individuals with ("CD") and without ("Non-IBD") Crohn's disease.

Gather

`Sourmash gather` compares a query signature against a database of reference signatures and outputs the minimum set of reference matches that contain all of the k-mers in the original query signature **[14]**. `Sourmash gather` is most frequently applied to metagenome profiling where a query metagenome is compared against a database of reference genomes (e.g., GenBank **[16]**, Genome Taxonomy Database **[17]**) to determine which genomes are in the metagenome (see Figure 10 for an additional explanation of the gather algorithm). Recent studies demonstrated that `sourmash gather` provides accurate taxonomic profiles for metagenomes sequenced with both long and short reads **[14][18]**. In sourmashconsumr, the first visualization for the output of `sourmash gather` details the fraction of a sample that is classifiable versus unclassifiable (Figure 4, A). Because we can use multiple databases in one run of `sourmash gather`, we can color the plot based on the database from which the match originated. The second visualization is an upset plot that depicts the intersection of reference genomes identified in each sample (Figure 4, B).

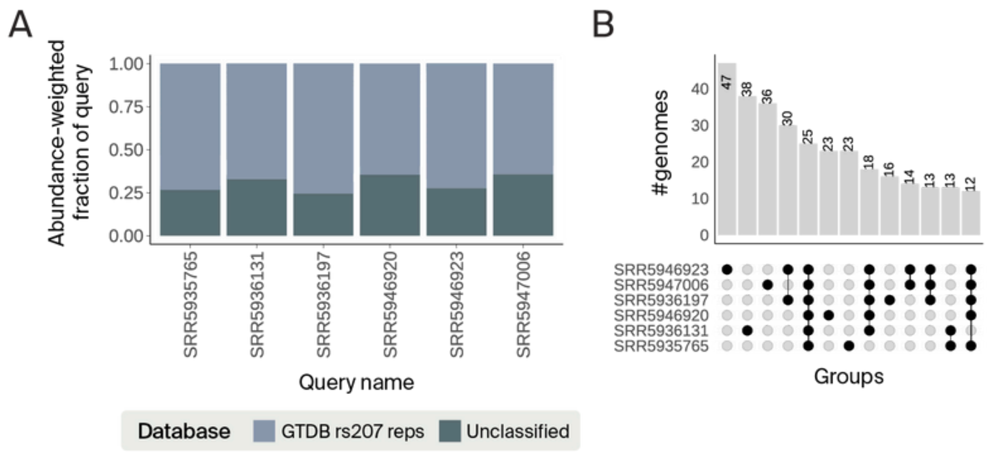

Figure 4

Visualizations that sourmashconsumr produced from the output of `sourmash gather`.

(A) A bar plot summarizing the fraction of each metagenome sample that is classifiable against the GTDB representative database (“GTDB rs207 reps").

(B) An upset plot depicting the genome matches shared by different sets of samples. Groups with fewer than 10 shared genome matches are not displayed. See **Figure 2** for an explanation of upset plots.

Taxonomy annotate

`sourmash taxonomy annotate` adds taxonomic lineages to genome matches that `sourmash gather` produces. This allows us to summarize genome matches up to higher levels of taxonomy than genome or strain. Taking advantage of this, all sourmashconsumr plots that visualize the output of `sourmash taxonomy annotate` allow optional taxonomy agglomeration to any level of taxonomy (domain, phylum, class, order, family, genus, species, and when provided by the taxonomy, strain). The first visualization in sourmashconsumr for the output of `sourmash taxonomy annotate` is a Sankey diagram, which can be applied to one or many samples (Figure 5, A). Sankey diagrams depict the flow from one set of values to another and, when applied to taxonomic profiles of metagenomes, show the taxonomic composition at increasingly finite resolution. The second visualization is an upset plot that depicts the presence of taxonomic lineages across samples (Figure 5, B). Since upset plots are based on the presence or absence of lineages, the counts in the upset barplot do not reflect the abundances of lineages observed at a given taxonomic level, but rather how many lineages are shared between samples at that level. The last visualization is an alluvial plot for time series samples that shows how relative abundances of taxonomic lineages shift over time (Figure 5, C).

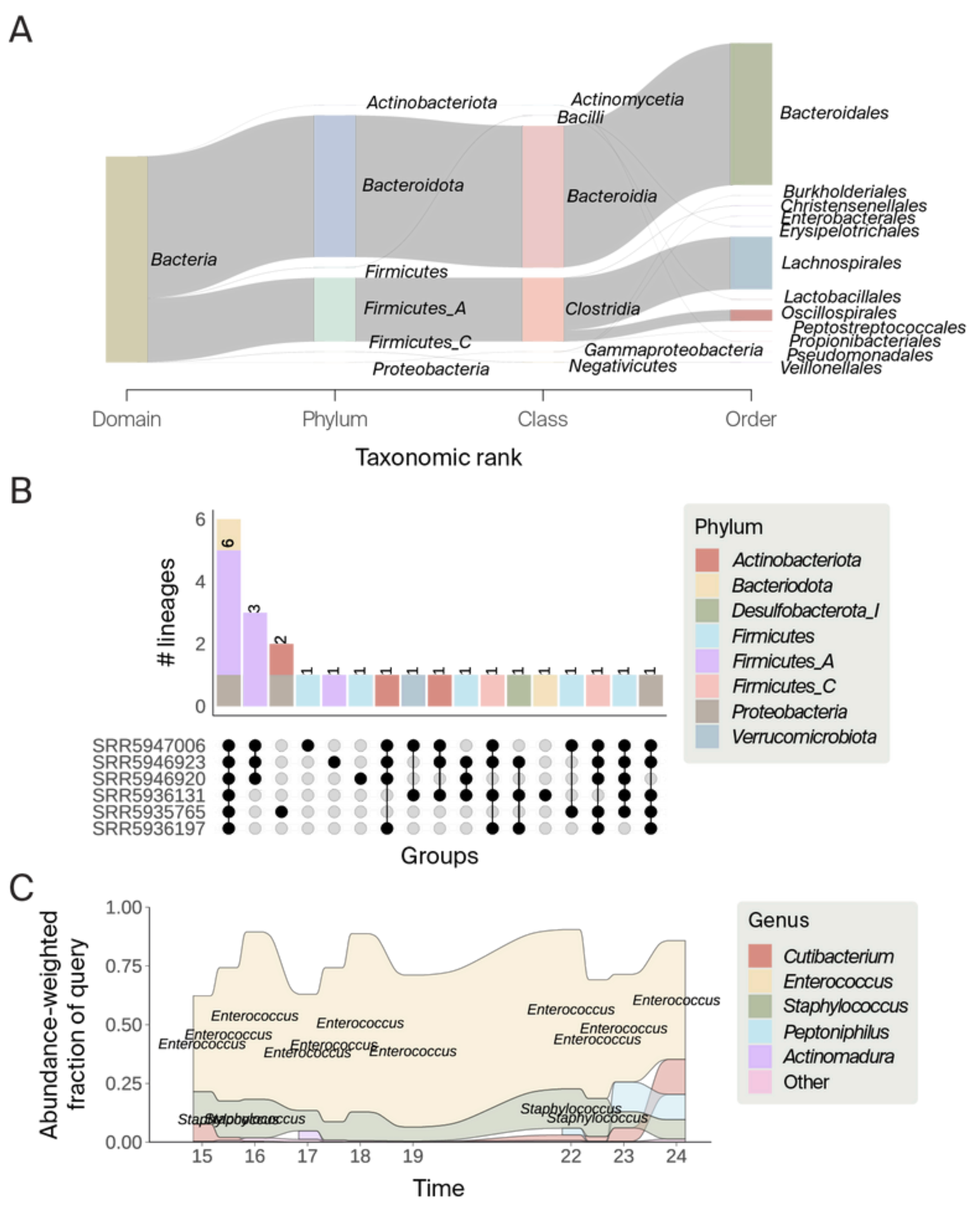

Figure 5

Visualizations sourmashconsumr produced from the output of `sourmash taxonomy annotate`.

(A) A Sankey diagram produced from one stool microbiome metagenome from an individual with Crohn's disease, SRR5935765. We performed taxonomic agglomeration at the order level. While colors help draw attention to delineations between arms of the diagram, they do not encode information.

(B) An upset plot depicting the shared taxonomic lineages between different sets of stool microbiome metagenome samples of individuals with and without Crohn's disease. We performed taxonomic agglomeration at the order level and colored the plot by the phylum of the shared lineages. See **Figure 2** for an explanation of upset plots.

(C) A time series alluvial plot depicting the change in taxonomic lineages over time in a time course of infant stool microbiome metagenomes (see the “New functionality” section below for a description of the dataset). We performed taxonomic agglomeration at the genus level. Each ribbon represents the fraction of the metagenome attributable to that taxonomic lineage at a given time. Lineages are labeled by name on the plot and by fill color. Only lineages that comprised at least 0.15 of the sample at a given time are labeled. Lineages that are low-abundance throughout the entire time course are grouped into “Other.”

Converting sourmash outputs into objects compatible with other R libraries

Many popular biological computing R packages require data to be in a specific format to enable the functions in those packages to work upon the data. Sourmashconsumr provides parsing functions to transform the output of `sourmash taxonomy annotate` into phyloseq **[2]** and metacoder **[3]** objects. After conversion to these formats, users can use the functions in those packages on sourmash results (Figure 6).

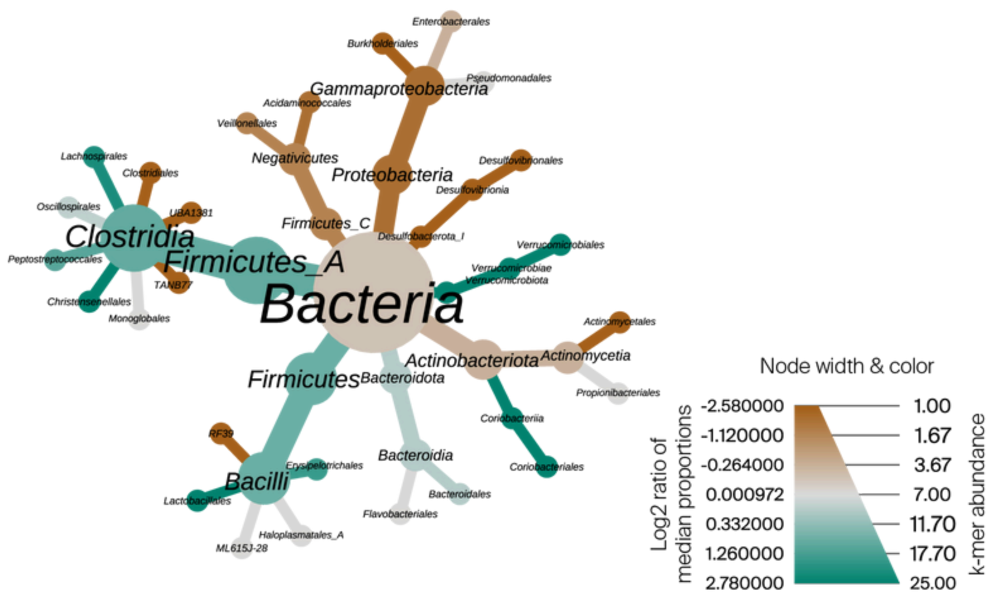

Figure 6

sourmashconsumr provides functions to convert the output of `sourmash taxonomy annotate` **into other package ecosystems in R.**

Metacoder heat tree plot representing the order-level taxonomic lineages present in six stool microbiome metagenomes from individuals with (n = 3) and without (n = 3) Crohn's disease. The nodes are labeled with taxonomic lineage assigned with `sourmash gather` and `sourmash taxonomy annotate` by comparing the metagenome samples against the Genome Taxonomy Database. Node size reflects the k-mer abundance of each lineage across all samples. Node colors represent log2 ratio for median proportions when comparing order-level k-mer abundance between individuals with and without Crohn's disease as calculated by the metacoder function `compare_groups()`. Green nodes are more abundant in individuals with Crohn's disease.

Both phyloseq and metacoder objects contain abundance information about taxonomic lineages in the data they represent. To derive this information, sourmashconsumr calculates the abundance-weighted number of matched hashes (subsampled k-mers) for each genome match returned by `sourmash gather`. It uses the metrics `unique_intersect_bp`, `scaled`, and `average_abund` output by `sourmash gather` to infer this using the equation:

$$(unique\_intersect\_bp/scaled) * average\_abund$$

`unique_intersect_bp` is the estimated number of base pairs that would overlap between the query and the genome match. It accounts for gather's greedy best-match-first algorithm and only counts overlaps once (e.g., once a sequence is matched between a query and a genome, the sequence is removed from subsequent rounds of matching; see Figure 10). Dividing this value by the `scaled` value results in the number of distinct k-mers that overlap between the query and the genome match. In the sourmash package, the scaled value controls the fraction of k-mers that are represented in the FracMinHash sketch **[14]**. The most common scaled value is 1000, so approximately 1/1000th of the k-mers in the original sequence become represented in the sketch. Lastly, sourmashconsumr multiplies this number by the average abundance of k-mers observed for a given genome match.

Representing abundances for genomes and lineages this way ensures that only k-mers that sourmash actually counted are represented in the final object, and that these estimates are not falsely inflated, which can impact downstream statistical results. One side effect of this is that phyloseq and metacoder objects will be most meaningful when they are built with samples analyzed with the same scaled value.

New functionality

The sourmash outputs are information-rich and can be used for downstream analyses like machine learning **[9]** or differential abundance analysis **[10]**. Taking advantage of these information sources, we implemented two new functions that work on the outputs of `sourmash sketch` and `sourmash taxonomy annotate`, to derive new insights from sequencing data. The first uses sourmash signatures to estimate sequencing saturation of a run and the second uses the output of `sourmash gather` and `sourmash taxonomy annotate` to detect whether multiple strains of the same species are present in a metagenome or other sequencing sample.

Estimating sequencing saturation using k-mers and accumulation curves

Sample complexity, sequencing strategy (e.g., RNA-seq, whole-genome sequencing), and sequencing depth determine whether a sequencing dataset captures all of the sequences contained in the original sample. It's difficult to determine the appropriate depth at which to sequence a sample without *a priori* knowledge of the sample complexity (e.g., genome size, number of transcripts expressed, microbial community diversity), but sample coverage can impact biological insights **[19]**. While it is often impossible to increase sequencing depth of a sample after initial sequencing efforts, measuring sequencing saturation after the fact highlights whether the sequencing run has captured the full diversity of the sample.

A variety of methods assess sequencing coverage **[20][21][22]**. We implemented an approach in sourmashconsumr that builds an accumulation curve (rarefaction curve) from abundance-weighted sketches and that uses the mean slope of the accumulation curve to estimate sequencing saturation (Figure 7). Using signatures built from raw or trimmed FASTQ reads, the function `from_signatures_to_rarefaction_df()` wraps the vegan `rarecurve()` function to produce a data frame that represents k-mer accumulation as a function of data seen **[23]**. Before building the data frame, we remove k-mers that only occur once in a sample, as these likely represent sequencing errors (when we build sketches from long k-mers, the majority of k-mers in a sketch will be from erroneous sequences given that one sequencing error produces *k* erroneous k-mers **[24]**). However, filtering out k-mers that only occur once per sample means that traditional methods to assess accumulation curve saturation that rely on k-mers that occur once or twice in a dataset can't be applied in this scenario **[25]**. To overcome this, we estimate sequencing saturation by calculating the mean slope of the accumulation curve — samples with higher mean slopes have a lower saturation as the accumulation curve doesn't approach its asymptote. Two of the CD samples are sequenced more deeply than the Non-IBD samples (Figure 7).

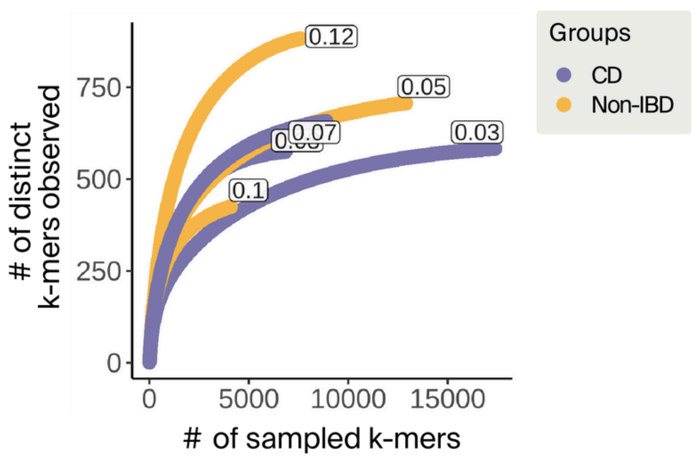

Figure 7

Accumulation curves produced from sourmash signatures.

Each curve represents one of six stool microbiome metagenomes from individuals with (CD, n = 3) and without (Non-IBD, n = 3) Crohn’s disease. Curves are labeled by their estimated mean slope. Curves that saturate and have a lower mean slope represent samples that have higher coverage. Mean slopes are reported as labels at the end of each curve.

To assess the accuracy of this method, we compared this approach against Nonpareil, a software tool that estimates metagenome coverage and sequencing diversity **[21]**. Using a subset of samples shown in Figure 2 of the 2018 Nonpareil publication **[21]**, we compared the sourmashconsumr mean slope of the accumulation curve against the Nonpareil diversity estimate. We removed samples sequenced with single-end runs, samples with multiple SRA run accessions, and run accessions for which data was no longer available, leaving a total of 70 samples from six biomes (Figure 8). Following the Nonpareil recommendations, we first trimmed the metagenome FASTQ files using fastp (`-q 20` , `--length_required 24`) **[26]** before applying Nonpareil (`-T kmer`) to produce diversity estimates and sourmash and sourmashconsumr to produce accumulation curves and mean slope estimates. We also obtained the data underlying Figure 2 in the Nonpareil publication, which contained Nonpareil diversity estimates and 16s rRNA gene OTU Shannon *H'* taxonomic diversity indices (https://gist.github.com/lmrodriguezr/c74684c275aa3e4db57a78e94c4fb7c0). Using linear regression to compare each of these metrics, we found that sourmashconsumr mean slope correlated equally well with 16s rRNA *H'* as Nonpareil diversity estimates (Figure 8). When we instead calculated accumulation curve mean slope with raw FASTQ files instead of trimmed, we found that R^2 estimates did not change for any comparison, indicating that filtering k-mers with abundance one effectively removed errors for this use case, removing the need for a trimming step prior to sequencing coverage estimation. This makes sourmashconsumr's approach a lightweight alternative to existing methods, especially when combined with the streaming capability built into `sourmash sketch`.

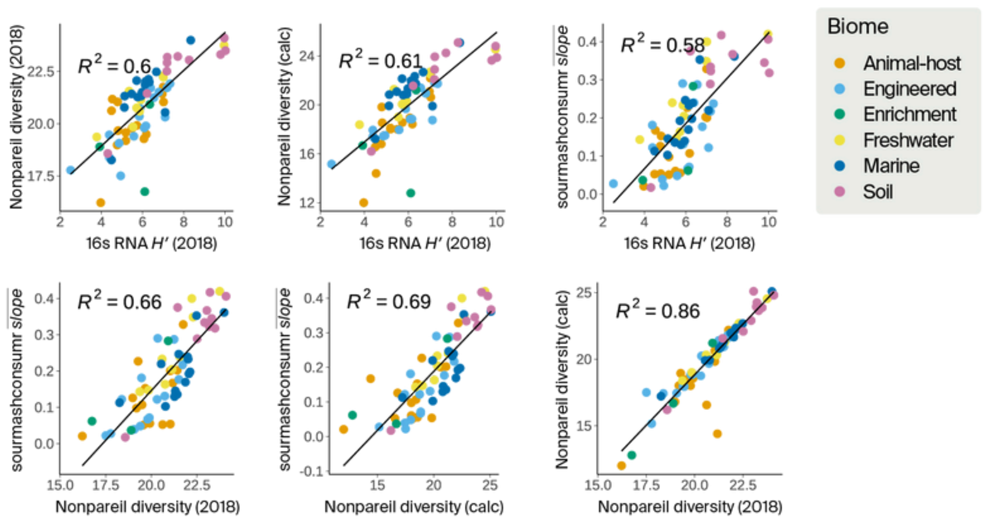

Figure 8

Mean slope of sourmashconsumr accumulation curves is comparable to other sequence diversity estimates.

Scatter plots comparing different sequence diversity metrics. 16s rRNA *H'* represents 16s rRNA gene OTU Shannon *H'* taxonomic diversity indices, sourmashconsumer s l o p e ‾ \overline{slope} slope represents mean slope estimates from accumulation curves, and Nonpareil diversity represents Nonpareil diversity estimates. Axes labeled with "(2018)" represent data published in Figure 2 of Rodriguez-R et al., while axes labeled "(calc)" are re-calculated for this study. Points are colored by the biome from which the metagenome sample originated. Linear regression R^2 values appear in the upper-left corner of each plot; all values are significant ($p < 0.001$).

While lightweight, the downside of this approach is that it produces fewer metrics in comparison to something like Nonpareil — for example, Nonpareil estimates percent coverage, diversity, and required sequencing effort to achieve full coverage. The accumulation curve mean slope estimate is closest to the Nonpareil percent coverage estimate (Figure 9).

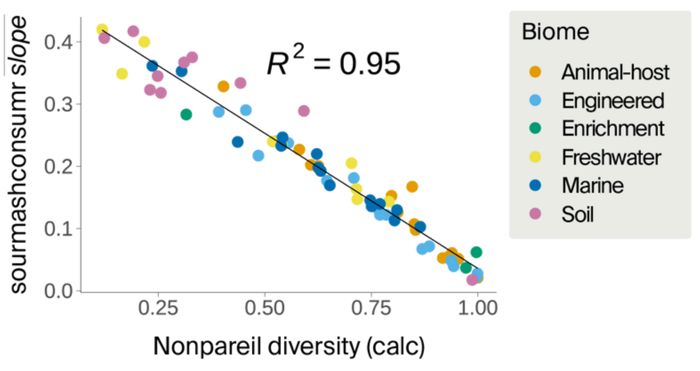

Figure 9

Mean slope estimates from sourmashconsumr accumulation curves are most similar to Nonpareil coverage estimates.

Scatter plot comparing Nonpareil coverage estimates against sourmashconsumr mean slope estimates. See **Figure 8** for axis, legend, and annotation descriptions.

Detecting whether multiple strains of the same species are present in a single metagenome

Microbial communities are made up of many different organisms. These organisms can be closely or distantly related. When organisms are different species, it is often easy to distinguish these sequences as belonging to different genomes because the sequences are sufficiently different (especially in indicators like tetranucleotide frequency **[27]**, average read depth **[28]**, and marker gene sequences **[29]**). However, when the organisms are different strains of the same species, it is often more difficult to resolve strain-level genomes, especially in the face of sequencing error and other noise. Yet, identifying strain-level differences within or between metagenomes can be important, as individual strains often have phenotypic differences of ecological or anthropogenic significance **[30]**. Many computational methods have recently been suggested to address this problem **[31][32]**. Taking advantage of metrics that `sourmash gather` generates, we developed an approach to detect when multiple strains of the same species are present in a single metagenome.

The CSV output of `sourmash gather` reports the genomes in a database that contain some fraction of the metagenome query. It also reports the fraction of each of those genomes that overlaps with the metagenome query. Our approach uses the fraction of each matched genome to detect whether multiple strains of the same species are likely present in the metagenome. The reason this works is because the `sourmash gather` algorithm is a greedy, best-match-first approach that provides the minimum set of genomes in the reference database needed to contain all of the k-mers in the metagenome **[14]** (Figure 10). This means that even if many genomes contain a portion of the k-mers in the metagenome, only the one genome that contains the largest fraction of overlap will be returned (Figure 10).

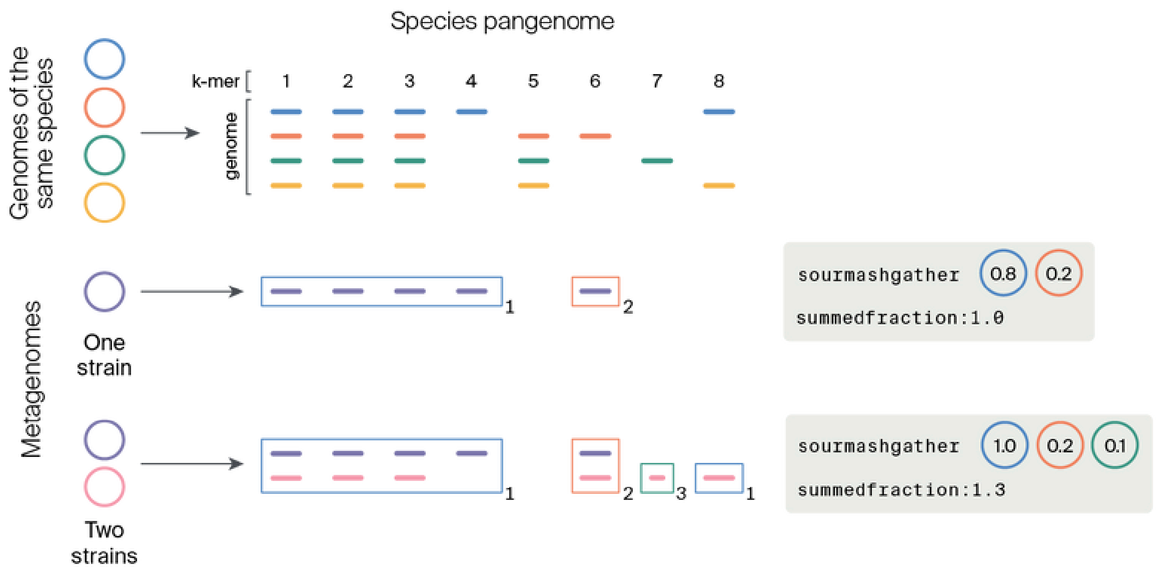

Figure 10

Conceptual overview of how the greedy algorithm sourmash gather works and how this impacts the detection of multiple strains for a given species in a metagenome.

Genomes of the same species represent reference genomes in the database used to run `sourmash gather` . Each genome is represented in the database as a sketch of k-mers. Some k-mers are shared by all genomes of the same species in the database. Other k-mers are only present in some genomes. `sourmash gather` works by first finding the best match between the k-mers in a metagenome and k-mers in a database.

In the example where only one genome is present (in purple), 0.8 (80%) of the k-mers overlap with the blue genome so it is returned first. These k-mers are then subtracted and the search is repeated. The orange genome is the next best match with 0.2 (20%) of the k-mers overlapping, so it is returned second. The number next to the box that surrounds the k-mers that matched a given genome demarcates the order the genome match is reported in. The fraction of the reference genome found in the metagenome is one of the metrics reported in the `sourmash gather` output. If only one strain (genome) of the species is present in a metagenome, the summed fraction of all of the genomes matched should not exceed ~1. When multiple strains are present, the greedy algorithm works the same as illustrated above. This means that the first match found by `sourmash gather` may account for k-mers in both strains. However, after summing the fraction of all matches, if the metagenome was sequenced to sufficient depth, the summed fraction should exceed ~1.

After this fraction is matched, `sourmash gather` will search any leftover fraction for that genome against the database again, and another genome of that species may be returned as a match for that fraction. This means that as long as genome sizes are approximately consistent across a species, the fraction of matched genomes for a given species should sum to approximately 1 if there is only one strain present in the metagenome. When we ran `sourmash gather` against GenBank on single genomes that were not in GenBank, the real value we observed for the sum of the fraction of matched genomes ranged between 0.046 (for genomes that only had ~genus-level relatives in the database) and 1.04 (for genomes that had many close matches in the database). Using this information, we empirically selected 1.1 (e.g., 110%) as the threshold for multiple strains of a single species being present in a metagenome. Thus, our approach parses the output of `sourmash gather` and `sourmash taxonomy annotate` to determine species that 1) had multiple matched genomes and 2) had a greater fraction of matched genomes than we expect. The function then plots these species (Figure 11) and returns a data frame of the species that are hypothesized to have multiple strains.

To validate this approach, we first applied our method to a synthetic metagenome dataset representing 40 microbial species **[33]**. The Critical Assessment of Metagenome Interpretation (CAMI) challenge generated synthetic metagenomes by simulating reads from a set of source genomes **[33]**. The "CAMI low" dataset mimics Illumina short-read sequencing with small insert sizes for a community with low diversity. Twenty-two genomes are from distinct strains while 18 are from strain variants (**Table 2**). We built a genome and taxonomy database from the CAMI low source genomes and ran `sourmash gather` and `sourmash taxonomy annotate` on the CAMI low metagenome. After reading the results into R using the function `read_taxonomy_annotate()`, we used the `from_taxonomy_annotate_to_multi_strains()` function to detect species for which there were likely multiple strains. Our method recovered the correct number of strains for each species in CAMI low without detecting additional species with multiple strains (Figure 11).

Species	Total source strains	Real strains	Evolved strains	Sum of fraction of genomes matched within species
Anaeroplasma bactoclasticum	4	1	3	1.65
Hydrotalea sandarakina	6	1	5	1.71
Paracoccus denitrificans	3	3	0	1.28
Unidentified *Bacillales* species	5	1	4	1.75

Table 2

Source genomes used to synthesize the CAMI low metagenome.

“Total source strains” refers to the total number of genomes used to synthesize the metagenome reads. “Real strains” are genomes that were sequenced from real organisms while “evolved strains” are computationally mutated from a real strain. The “sum of fraction of genomes matched within species” is the sum of the `f_match` metric output by `sourmash gather` and summarized by the `from_taxonomy_annotate_to_multistrains()` function.

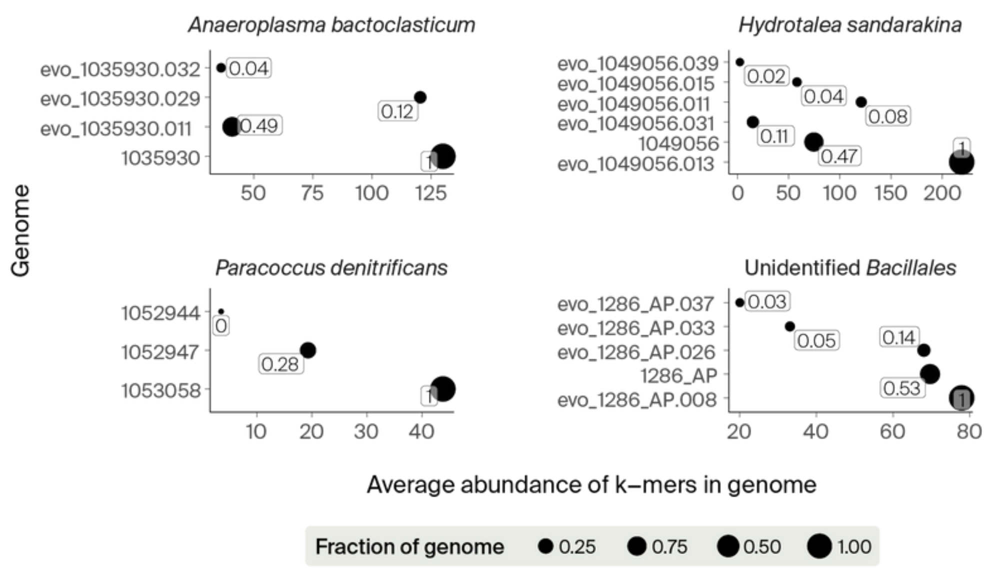

Figure 11

The `from_taxonomy_annotate_to_multi_strains()` **function in sourmashconsumr predicted the correct number of strains for each species with multiple strains in the synthetic CAMI low dataset**.

Each facet depicts a species that had more than one strain. Each dot on a plot represents one genome match identified by `sourmash gather` and the genome names appear on the y-axis. The dot size encodes the fraction of the genome that was matched, as do the dot labels positioned next to each point. The x-axis represents the average abundance of k-mers for each genome match. Note the x-axis varies between each facet.

Having demonstrated the ability of this approach to identify species with multiple strains in a synthetic community, we next tested whether this approach worked with real metagenomes. We used an infant stool metagenome time series that was previously demonstrated to have multiple strains of *Staphylococcus epidermidis* in nine samples during a sampling course across 11 time points **[34]**. Our approach detected strain variation at six of these times (Figure 12), meaning it produced accurate results for 8 of 11 samples (73%). Using coverage estimates for each *S. epidermidis* strain reported in **[34]**, we suspect that insufficient sequencing depth led to failed detection of multiple strains for the three samples that were missed (**Table 3**).

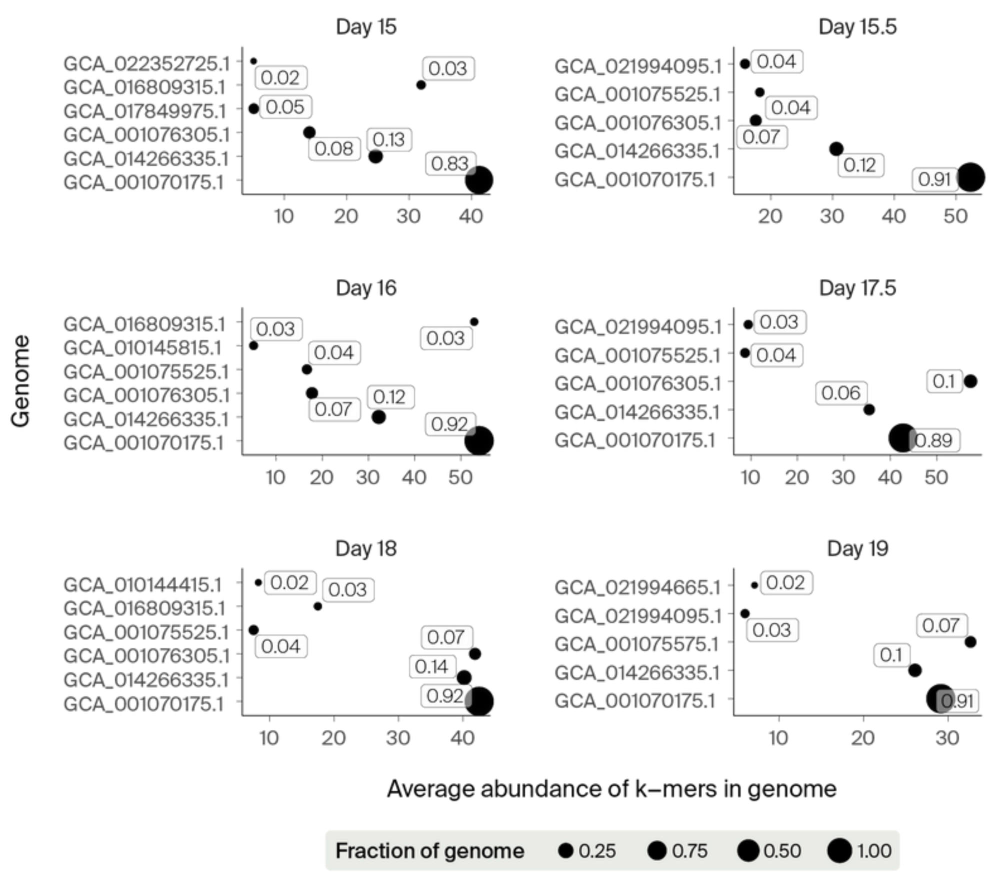

Figure 12

sourmashconsumr predicted the presence of multiple strains of *Staphylococcus epidermidis* at six time points in an infant stool metagenome time series.

Each facet depicts a metagenome and is labeled by the day the sample was collected. All genome accessions (y-axis) represent *S. epidermidis* genomes. See **Figure 11** for further description of the plot.

Sample day	Strain 1 coverage	Strain 3 coverage	Strain 4 coverage	Total coverage	sourmashconsum
Day 15	57	20	1	78	yes
Day 15.5	58	17	4	79	yes
Day 16	65	18	5	88	yes
Day 17	0	53	0	53	no

Sample day	Strain 1 coverage	Strain 3 coverage	Strain 4 coverage	Total coverage	sourmashconsum
Day 17.5	6	35	3	44	yes
Day 18	18	40	4	62	yes
Day 19	6	23	0	29	yes
Day 22	7	22	5	34	no
Day 22.5	1	40	1	42	no
Day 23	5	8	3	16	no
Day 24	0	28	0	28	no

Table 3. Strain coverage and prediction for *S. epidermidis* in an infant stool metagenome time series.

Strain labels are inherited from **[34]**. Normalized coverage estimates from Figure 3 in **[34]** are accurate within five units. “sourmashconsumr” refers to whether the `from_taxonomy_annotate_to_multistrains()` function predicted the presence of multiple strains while “Accurate” refers to whether the sourmashconsumr prediction was accurate or not.

In addition to the presence of multiple strains for *S. epidermidis*, our method detected multiple strains for *Enterococcus faecalis*, a species that wasn’t reported to have strain variation in the original publication **[34]** (Figure 13).

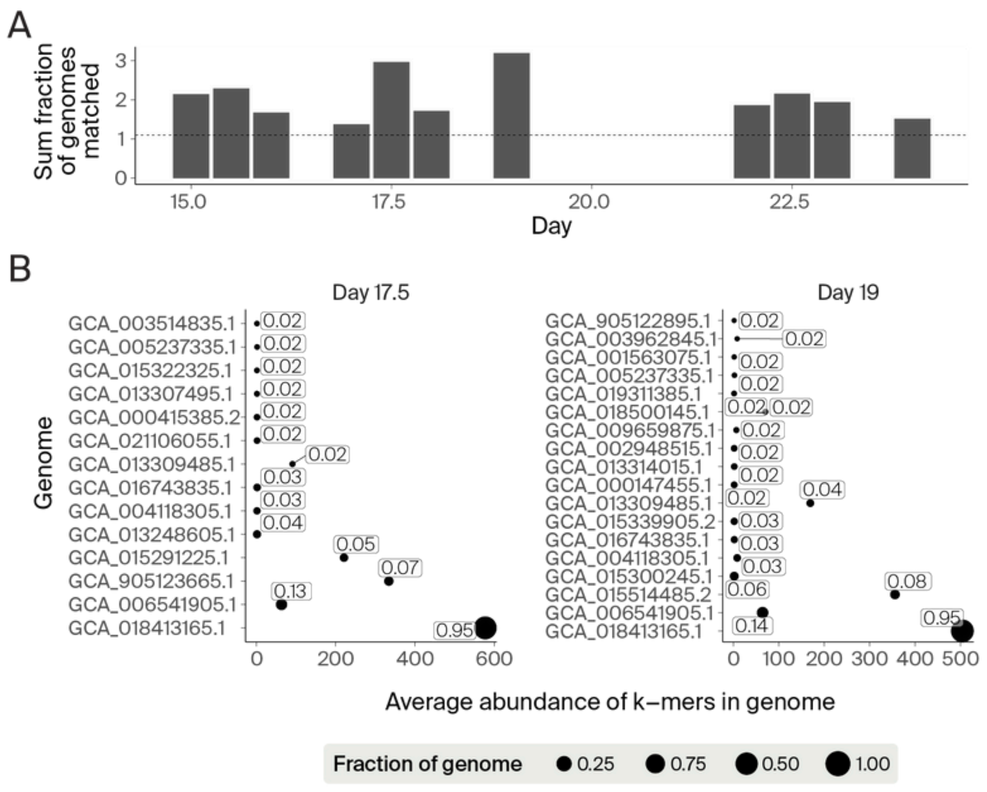

Figure 13

Multiple strains of *Enterococcus faecalis* in all eleven samples of the time series of an infant stool metagenome.

(A) The sum fraction of all *E. faecalis* genomes matched exceeded the empiric threshold of 1.1, indicating that multiple strains of *E. faecalis* were likely present throughout the infant stool microbiome metagenome time course.

(B) *E. faecalis* genomes matched for the two samples that had the most observed sum fraction of *E. faecalis* genomes matched. See **Figure 11** for a description of the plot.

Our approach has limitations tied to the size and quality of the reference database used during `sourmash gather` , and to a lesser extent, the quality of the taxonomy used to label the lineage of each genome match. First, this approach can only detect the presence of multiple strains when there are multiple representative genomes for a given species in the taxonomy. If there are one or no representatives for that species in the database, then this approach will always fail to detect multiple strains. Similarly, the genomes in the reference database must represent the genome in the metagenome, meaning if the genome in the metagenome contains pangenomic elements that aren't in any genome in the reference database, this approach won't quantify those fractions of the genome. Given these two limitations, this approach will be most successful for organisms or environments that are well represented in genome databases (e.g., GenBank **[16]**, Genome Taxonomy Database **[17]**) such as the gut microbiome or *Escherichia coli*. The quality of the genomes in the reference database may also impact this method. If genomes in the reference database are contaminated, this may lead to false inflation of genome match metrics which could skew the results. Similarly, if the taxonomic lineages group improperly or distribute unrelated or related lineages, respectively, this could skew the results. Using a database like GTDB that uses sequence similarity to group genomes into lineages may ameliorate this issue at the expense of genome recall due to the smaller size of GTDB compared to other databases like GenBank.

The success of this strategy will also depend on sequencing depth. Because this approach is tied to coverage of genomes detected in a metagenome, if the sequencing depth is not sufficient to cover the genomes present in the community, this method will fail to detect multiple strains even if they are present.

Finally, while sourmashconsumr can detect the presence of multiple strains of a given species, it does not estimate the number of strains that are present. We attempted to cluster genomes by average k-mer abundance to estimate the number of strains present, but realized that average k-mer abundance was likely not sufficient to infer this information because of the greedy nature of the `sourmash gather` algorithm. That is, because the first genome match may combine genomic elements from multiple strains, the average abundance will not be for genome sequences for one strain, but for multiple strains. To estimate the number of strains present, it may be possible to map the metagenome reads against the matched genomes and use an expectation maximization algorithm such as those used to estimate transcript abundance in RNA-seq quantification **[35]**. This remains an area for future research.

Key takeaways

The functions in the sourmashconsumr R package expose the outputs of the sourmash Python package to analysis and visualization in R. After running sourmash on sequencing data, the sourmashconsumr package provides default parsing, analysis, and visualization functions to help interpret these outputs and to leverage them for additional statistical or machine learning analyses. We hope this package will help a broader range of researchers to gain insights from their sequencing data.

> The sourmashconsumr R **package** is available at this GitHub repository (DOI: 10.5281/zenodo.7591833) under an MIT license. All **code associated with figures and validation** is available in this GitHub repository (DOI: 10.5281/zenodo.7591845).

Next steps

We developed the sourmashconsumr package openly on GitHub following the R packages guide. We plan to continue to extend functionality in the package as needed. We also plan to submit the package to CRAN and to build a conda package.

We'd love to hear your feedback. What function are you most excited to use? What functions do you think are missing or would you like to see?

References

1 Martin BD, Witten D, Willis AD. (2020). Modeling microbial abundances and dysbiosis with beta-binomial regression. https://doi.org/10.1214/19-aoas1283

2 McMurdie PJ, Holmes S. (2013). phyloseq: An R Package for Reproducible Interactive Analysis and Graphics of Microbiome Census Data. https://doi.org/10.1371/journal.pone.0061217

3 Foster ZSL, Sharpton TJ, Grünwald NJ. (2017). Metacoder: An R package for visualization and manipulation of community taxonomic diversity data. https://doi.org/10.1371/journal.pcbi.1005404

4 Titus Brown C, Irber L. (2016). sourmash: a library for MinHash sketching of DNA. https://doi.org/10.21105/joss.00027

5 Gingrich AA, Reiter TE, Judge SJ, York D, Yanagisawa M, Razmara A, Sturgill I, Basmaci UN, Brady RV, Stoffel K, Murphy WJ, Rebhun RB, Brown CT, Canter RJ. (2021). Comparative Immunogenomics of Canine Natural Killer Cells as Immunotherapy Target. https://doi.org/10.3389/fimmu.2021.670309

6 Tian L, Mazloom R, Heath LS, Vinatzer BA. (2021). LINflow: a computational pipeline that combines an alignment-free with an alignment-based method to accelerate generation of similarity matrices for prokaryotic genomes. https://doi.org/10.7717/peerj.10906

7 Stewart RD, Auffret MD, Warr A, Walker AW, Roehe R, Watson M. (2019). Compendium of 4,941 rumen metagenome-assembled genomes for rumen microbiome biology and enzyme discovery. https://doi.org/10.1038/s41587-019-0202-3

8 Patin NV, Goodwin KD. (2022). Long-Read Sequencing Improves Recovery of Picoeukaryotic Genomes and Zooplankton Marker Genes from Marine Metagenomes. https://doi.org/10.1128/msystems.00595-22

9 Reiter TE, Irber L, Gingrich AA, Haynes D, Pierce-Ward NT, Brooks PT, Mizutani Y, Moritz D, Reidl F, Willis AD, Sullivan BD, Brown CT. (2022). Meta-analysis of metagenomes via machine learning and assembly graphs reveals strain switches in Crohn’s disease. https://doi.org/10.1101/2022.06.30.498290

10 West KA, Yin X, Rutherford EM, Wee B, Choi J, Chrisman BS, Dunlap KL, Hannibal RL, Hartono W, Lin M, Raack E, Sabino K, Wu Y, Wall DP, David MM, Dabbagh K, DeSantis TZ, Iwai S. (2022). Multi-angle meta-analysis of the gut microbiome in Autism Spectrum Disorder: a step toward understanding patient subgroups. https://doi.org/10.1038/s41598-022-21327-9

11 Reiter TE, Pierce-Ward NT, Irber L, Botvinnik OB, Brown CT. (2022). Protein k-mers enable assembly-free microbial metapangenomics. https://doi.org/10.1101/2022.06.27.497795

12 Wickham H, Averick M, Bryan J, Chang W, McGowan L, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen T, Miller E, Bache S, Müller K, Ooms J, Robinson D, Seidel D, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H. (2019). Welcome to the Tidyverse. https://doi.org/10.21105/joss.01686

13 Lloyd-Price J, Arze C, Ananthakrishnan AN, Schirmer M, Avila-Pacheco J, Poon TW, Andrews E, Ajami NJ, Bonham KS, Brislawn CJ, Casero D, Courtney H, Gonzalez A, Graeber TG, Hall AB, Lake K, Landers CJ, Mallick H, Plichta DR, Prasad M, Rahnavard G, Sauk J, Shungin D, Vázquez-Baeza Y, White RA III, Bishai J, Bullock K, Deik A, Dennis C, Kaplan JL, Khalili H, McIver LJ, Moran CJ, Nguyen L, Pierce KA, Schwager R, Sirota-Madi A, Stevens BW, Tan W, ten Hoeve JJ,

Weingart G, Wilson RG, Yajnik V, Braun J, Denson LA, Jansson JK, Knight R, Kugathasan S, McGovern DPB, Petrosino JF, Stappenbeck TS, Winter HS, Clish CB, Franzosa EA, Vlamakis H, Xavier RJ, Huttenhower C. (2019). Multi-omics of the gut microbial ecosystem in inflammatory bowel diseases. https://doi.org/10.1038/s41586-019-1237-9

14 Irber L, Brooks PT, Reiter T, Pierce-Ward NT, Hera MR, Koslicki D, Brown CT. (2022). Lightweight compositional analysis of metagenomes with FracMinHash and minimum metagenome covers. https://doi.org/10.1101/2022.01.11.475838

15 Pierce NT, Irber L, Reiter T, Brooks P, Brown CT. (2019). Large-scale sequence comparisons with sourmash. https://doi.org/10.12688/f1000research.19675.1

16 Sayers EW, Cavanaugh M, Clark K, Pruitt KD, Sherry ST, Yankie L, Karsch-Mizrachi I. (2022). GenBank 2023 update. https://doi.org/10.1093/nar/gkac1012

17 Parks DH, Chuvochina M, Rinke C, Mussig AJ, Chaumeil P-A, Hugenholtz P. (2021). GTDB: an ongoing census of bacterial and archaeal diversity through a phylogenetically consistent, rank normalized and complete genome-based taxonomy. https://doi.org/10.1093/nar/gkab776

18 Portik DM, Brown CT, Pierce-Ward NT. (2022). Evaluation of taxonomic classification and profiling methods for long-read shotgun metagenomic sequencing datasets. https://doi.org/10.1186/s12859-022-05103-0

19 Merrill BD, Carter MM, Olm MR, Dahan D, Tripathi S, Spencer SP, Yu B, Jain S, Neff N, Jha AR, Sonnenburg ED, Sonnenburg JL. (2022). Ultra-deep Sequencing of Hadza Hunter-Gatherers Recovers Vanishing Gut Microbes. https://doi.org/10.1101/2022.03.30.486478

20 Rodriguez-R LM, Konstantinidis KT. (2013). Nonpareil: a redundancy-based approach to assess the level of coverage in metagenomic datasets. https://doi.org/10.1093/bioinformatics/btt584

21 Rodriguez-R LM, Gunturu S, Tiedje JM, Cole JR, Konstantinidis KT. (2018). Nonpareil 3: Fast Estimation of Metagenomic Coverage and Sequence Diversity. https://doi.org/10.1128/msystems.00039-18

22 Robinson DG, Storey JD. (2014). subSeq: Determining Appropriate Sequencing Depth Through Efficient Read Subsampling. https://doi.org/10.1093/bioinformatics/btu552

23 Dixon P. (2003). VEGAN, a package of R functions for community ecology. https://doi.org/10.1111/j.1654-1103.2003.tb02228.x

24 Melsted P, Pritchard JK. (2011). Efficient counting of k-mers in DNA sequences using a bloom filter. https://doi.org/10.1186/1471-2105-12-333

25 Dove ADM, Cribb TH. (2006). Species accumulation curves and their applications in parasite ecology. https://doi.org/10.1016/j.pt.2006.09.008

26 Chen S, Zhou Y, Chen Y, Gu J. (2018). fastp: an ultra-fast all-in-one FASTQ preprocessor. https://doi.org/10.1093/bioinformatics/bty560

27 Teeling H, Meyerdierks A, Bauer M, Amann R, Glöckner FO. (2004). Application of tetranucleotide frequencies for the assignment of genomic fragments. https://doi.org/10.1111/j.1462-2920.2004.00624.x

28 Tyson GW, Chapman J, Hugenholtz P, Allen EE, Ram RJ, Richardson PM, Solovyev VV, Rubin EM, Rokhsar DS, Banfield JF. (2004). Community structure and metabolism through reconstruction of microbial genomes from the environment. https://doi.org/10.1038/nature02340

29 Lin H-H, Liao Y-C. (2016). Accurate binning of metagenomic contigs via automated clustering sequences using information of genomic signatures and marker genes. https://doi.org/10.1038/srep24175

30 Van Rossum T, Ferretti P, Maistrenko OM, Bork P. (2020). Diversity within species: interpreting strains in microbiomes. https://doi.org/10.1038/s41579-020-0368-1

31 Quince C, Nurk S, Raguideau S, James R, Soyer OS, Summers JK, Limasset A, Eren AM, Chikhi R, Darling AE. (2021). STRONG: metagenomics strain resolution on assembly graphs. https://doi.org/10.1186/s13059-021-02419-7

32 Olm MR, Crits-Christoph A, Bouma-Gregson K, Firek BA, Morowitz MJ, Banfield JF. (2021). inStrain profiles population microdiversity from metagenomic data and sensitively detects shared microbial strains. https://doi.org/10.1038/s41587-020-00797-0

33 Sczyrba A, Hofmann P, Belmann P, Koslicki D, Janssen S, Dröge J, Gregor I, Majda S, Fiedler J, Dahms E, Bremges A, Fritz A, Garrido-Oter R, Jørgensen TS, Shapiro N, Blood PD, Gurevich A, Bai Y, Turaev D, DeMaere MZ, Chikhi R, Nagarajan N, Quince C, Meyer F, Balvočiūtė M, Hansen LH, Sørensen SJ, Chia BKH, Denis B, Froula JL, Wang Z, Egan R, Don Kang D, Cook JJ, Deltel C, Beckstette M, Lemaitre C, Peterlongo P, Rizk G, Lavenier D, Wu Y-W, Singer SW, Jain C, Strous M, Klingenberg H, Meinicke P, Barton MD, Lingner T, Lin H-H, Liao Y-C, Silva GGZ, Cuevas DA, Edwards RA, Saha S, Piro VC, Renard BY, Pop M, Klenk H-P, Göker M, Kyrpides NC, Woyke T, Vorholt JA, Schulze-Lefert P, Rubin EM, Darling AE, Rattei T, McHardy AC. (2017). Critical Assessment of

Metagenome Interpretation—a benchmark of metagenomics software.
https://doi.org/10.1038/nmeth.4458

34 Sharon I, Morowitz MJ, Thomas BC, Costello EK, Relman DA, Banfield JF. (2012). Time series community genomics analysis reveals rapid shifts in bacterial species, strains, and phage during infant gut colonization.
https://doi.org/10.1101/gr.142315.112

35 Patro R, Duggal G, Love MI, Irizarry RA, Kingsford C. (2017). Salmon provides fast and bias-aware quantification of transcript expression.
https://doi.org/10.1038/nmeth.4197