Sourmash Branchwater Enables Lightweight Petabyte-Scale Sequence Search

This manuscript (permalink) was automatically generated from dib-lab/2022-paper-branchwater-software@2d21ffd on October 27, 2022.

Authors

- **Luiz Irber**
  0000-0003-4371-9659 · luizirber · luizirber
  Graduate Group in Computer Science, UC Davis; Department of Population Health and Reproduction, UC Davis · Funded by Grant GBMF4551 from the Gordon and Betty Moore Foundation; Grant R01HG007513 from the NIH NHGRI

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

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

✉ — Correspondence possible via GitHub Issues

Abstract

We introduce branchwater, a flexible and fast petabase-scale search for the 800,000 public metagenomes presently in the NCBI Sequence Read Archive. Our search is based on the FracMinHash k-mer sketching technique and can search all public metagenomes with 1000 query genoems in approximately 24 hours using 30 GB of RAM and 32 threads. Branchwater is a Rust-based multithreading front-end built on top of the sourmash library. We provide biological use cases, discuss design and performance considerations, and provide benchmarks for a variety of different parameters.

Introduction

The Introduction should provide context as to why the software tool was developed and what need it addresses. It is good scholarly practice to mention previously developed tools that address similar needs, and why the current tool is needed.

Substantial growth in publicly available nucleotide sequencing data (DNA and RNA) has occurred over the last decade, driven by decreases in sequencing costs. In particular the Sequence Read Archive now has over XXX PB of data as of YY date, with YYY PB of data public. Shotgun metagenomes, generated by random sequencing of mixtures of microbes sampled from a microbiome are a particularly interesting resource.

Shotgun metagenome data sets are often large (100s of MBs to 10s of GB) and can be highly complex, with environmental samples containing genomic data that can be attributed to thousands or more species. In the past decade, hundreds of thousands of new bacterial and archaeal genomes have been isolated from public metagenomes, and several entirely new branches of life have been discovered (large bacteriophage, hug et al, etc.)

Beyond their initial use, these data sets form an incredibly rich resource for contextualizing novel sequencing data and for synthesis research on a myriad of large-scale genomic questions ranging from basic evolutionary processes to disease associations and pathogenicity tracking (Table X). However, comprehensive discovery of relevant data sets is challenging. Metadata for these data sets is typically geared towards the submitting researcher's major findings and study questions, and moreover cannot possibly describe the full contents of the data. Furthermore, metadata provided at the time of submission can be incomplete or inconsistent, rendering systematic data set discovery intractable.

Content-based search is a promising alternative strategy to finding data sets in archives. By searching with genomic content of interest, content-based search can recover datasets containing relevant species or genes of interest regardless of their associated metadata. However, search of unassembled sequence is critical to ensure unbiased and comprehensive recovery of relevant datasets. Assembly techniques are designed to produce consensus reference sequences useful for consistent comparisons across genotypes, often collapsing sequence variation in the process. In addition, reassembly and reanalysis of existing data using different parameters or newer methods often yields different results. Content-based search of unassembled metagenomes can bypass these issues and facilitate consistent downstream analysis across data sets that may have been initially generated to answer a range of disparate biological questions, and been first analyzed over a range of years and with myriad techniques.

A number of approaches have been developed to enable content-based search of single-organism genomic and RNAseq data. Methods that enable rapid, large-scale search across hundreds of thousands of data sets typically leverage biological sketching techniques and probabilistic data structures to reduce the effective search space [1,2,3]. However, these approaches do not readily translate to datasets with unknown levels of sequence diversity, the defining feature of metagenomic datasets.

Recent extensive search across viral datasets ... comprehensive but time-consuming and costly, intractable for independent researchers [4].

Below, we introduce Branchwater, an SRA-scale querying system that uses containment searches based on FracMinHash sketching to search all public metagenome data sets in the SRA in

approximately 24 hours on commodity hardware with 1-1000 query genomes. Branchwater uses the Rust library underlying the sourmash implementation of FracMinHash to execute massively parallel searches of a presketched digest of the SRA [5,6].

The availability of relatively lightweight content-based search of SRA metagenomes addresses many of use cases (Table 1). Some of these use cases have already been explored with Branchwater: Viehweger et al. used Branchwater to discover a metagenomic sample containing *Klebsiella pneumonia* that was subsequently included in an outbreak analysis, and Lumian et al. (2022) conducted a biogeographical study on five newly generated cyanobacterial genomes from Antarctic samples.

Thoughts and questions:

- do we mention CMash and mash screen? Same challenges as serratus and intended for single sample analysis. not intended for large scale search but rather s
- For discussion: content-based search, including MAGs and things without marker genes.
- stress lightweight resource usage
- note that a key requirement of Branchwater was that we search unassembled.
- discuss thresholds ...somewhere.
- list breakdown of current data sets in paper
- remember “hit to lead” stuff

Table 1: Biological use cases for petabase scale sequence search of metagenomes

Use case	Description	Enabled by branchwater
Biogeography of genomes	Describe and characterize biogeographical distribution of species; identify sampling locations	Yes
Outbreak tracking	Trace pathogen spread via public data	Yes, for genomes > 10kb
Pangenome expansion	Expand and explore composition of strains, species and genus levele pangenomes (including SAGs and MAGs)	Yes
SRA metadata reannotation	Content-based validation and reannotation of SRA metagenomes	Yes
Private database access and search	Provide privacy-enabled search of large, access-restricted databases	Yes
Post-processing and cleaning MAGs	Evaluating contig-level presence and abundance across multiple metagenomes	Yes, for contigs > 10kb
Exploring breadth of plasmids etc.	Evaluating range and prevalence of laterally transferred genomic elements	Yes, for plasmids > 10b
Exploring host range of species for regulatory evaluation		Yes, for genomes > 10kb
Search for small viruses		No
Searching for specific functional genes		No
Detecting novel classes and orders		No

NOTE: Small viral pangenome query comment/Luiz. Association studies?

edit table here

Background: FracMinHash and sourmash

FracMinHash is a bottom-sketch version of ModHash that supports accurate estimation of overlap and containment between two sequencing sets [6]. In brief, FracMinHash is a lossy compression approach that represents data sets using a "fractional" sketch containing 1/S of the original k-mers. Sketches support estimation of overlap, bidirectional containment, and Jaccard similarity between two data sets. Unlike other common sketching techniques such as MinHash [7] and HyperLogLog [8], FracMinHash supports these operations between two data sets of different sizes, and unlike mash screen and CMash does not require the original data sets [9,10]. In exchange, FracMinHash sketches are essentially unbounded in size.

The open-source sourmash software provides a mature and well-documented command-line interface to FracMinHash, along with Python and Rust APIs for loading and using FracMinHash sketches [5,11]. The Python layer provides a larger number of user experience conveniences on top of the performant Rust layer. However, despite the thread safety of the underlying Rust code, the CLI and Python library still operate in single-threaded mode, which limits the utility of sourmash for very large scale operations. Refactoring the sourmash CLI and Python libraries to take advantage of thread safety is a substantial and ongoing effort; we chose here to develop a dedicated CLI in Rust instead.

There are several features of FracMinHash and sourmash that limit their utility for specific use cases. In particular, the default scaled parameters used in sourmash do not work well for comparing or detecting genomes smaller than 10kb in size. Nor can divergent genomes be found; based on k-mer containment to ANI conversion [12], we find that sourmash works well for finding matches to genomes within about 90% ANI of the query, but not further. Finally, FracMinHash was developed for shotgun data sets and different parameters would be required for targeted sequencing data such as amplicon data sets. Some of these limitations are intrinsic to FracMinHash, and others may be overcome in the future by parameter tuning and further research.

Petabase scale search represents a specific technical challenge to sourmash

The primary design focus for the sourmash CLI has been on searching and comparing many genome-sized sketches, where for typical parameters there are between 1000 and 10,000 hashes in each sketch. The software provides a variety of in-memory and on-disk data structures for organizing sketches in this size range and can search hundreds of thousands of genome sketches with a single query in minutes in a single thread on an SSD laptop; more complex algorithms such as the min-set-cov described in XXX can take a few hours but are still acceptably performant on real-world data.

Branchwater faces very different parameters in searching 800,000 metagenomes. Many of these data sets are extremely large, slow to read from disk, and individually require substantial memory to load. Multiple queries may be used to search each metagenome as well, making this a quadratic search.

One solution we tried initially was a scatter-gather approach based on a cluster-aware workflow engine (in this case, snakemake [**snakemake?**]). The overhead on workflow coordination and executing shell commands was prohibitive for our initial implementation, so we pursued a purpose-built multithreaded solution instead.

Methods

The Methods should include a subsection on Implementation describing how the tool works and any relevant technical details required for implementation; and a subsection on Operation, which should include the minimal system requirements needed to run the software and an overview of the workflow.

Sketching the Sequence Read Archive

We determined the accessions of all publicly available shotgun metagenomic via the query string `"METAGENOMIC"[Source] NOT amplicon[All Fields]` at the NCBI Sequence Read Archive Web site, https://www.ncbi.nlm.nih.gov/sra. We then downloaded all runs for all accessions and streamed them into `sourmash sketch dna` with parameters `-p k=21,31,51,scaled=1000,abund`. The resulting sourmash signature files were saved as individual gzipped JSON files (each containing 3 sketches), one file for each input run.

The resulting catalog contains 767,277 metagenome data sets as of YYY, with the large majority being from human-associated microbiomes (Table 2). The size of all sketches together is 7.5 TB, representing ZZZ PB of original files. The average sketch file size is 9.7 MB, and the median is 570kb. The largest 10,000 data sets comprise 30% of the total sketch sizes.

Table 2: The 10 largest categories of metagenome data set types in the Sequence Read Archive, as of March 2022.

"Scientific Name" provided by submitter	distinct data sets
human gut metagenome	162187
metagenome	57048
gut metagenome	47244
human metagenome	36438
soil metagenome	35323
mouse gut metagenome	26482
human skin metagenome	25700
Homo sapiens	21020
marine metagenome	14400
human oral metagenome	14235

Implementation of multithreaded search

The `sra_search` program is built in Rust on top of the sourmash library for loading and comparing sketches. It implements the following steps:

1. Loads all query sketches into memory from a list of files.
2. Loads the list of filenames containing subject sketches to search.
3. In a Rust closure function executed in parallel for each subject sketch filename,
    a. loads the subject sketch from the file;
    b. for each query, determines the estimated overlap between query and subject;
    c. reports overlaps above a user-specified threshold.
    d. releases all per-metagenome resources

Downsampling of sketches to higher scaled values is performed dynamically, after load (if requested). Results are reported back to a separate "writer" thread via a threadsafe multi-producer, single-consumer FIFO queue. We use the rayon `par_iter` function to execute the closures in parallel.

This approach leverages the core features of sourmash to efficiently keep queries in memory and batch-process metagenome sketches without storing them all in memory. The approach also takes advantage of the effective immutability of queries, which can be shared without data races by multiple processing threads.

Executing sra_search at the command line

`sra_search` takes in search parameters as well as two text files, one containing a list of query file paths and one containing a list of subject file paths. Upon execution, it reports the number of query sketches loaded and the number of subject file paths found, and then begins the search. It progressively reports the number of sketches searched in blocks of 10000, and outputs matches to a CSV File.

We typically run `sra_search` in a snakemake workflow, which manages environment variables and input/output files.

Performance and scaling analysis

`sra_search` scales linearly in memory and time with queries and # threads.

A variety of simple benchmarks will presumably show:

- Speed increases linearly with number of threads
- Memory scales linearly: sum(queries) + sum(threads*average size of sketches)
- What happens w/biggest sketches?
- Complexity is n(query) * n(subject), with subject loading being the dominant practical time. More complex indexing and query foo could be done but it's fast enough and the code is simple.

Do this on ~1000-10,000 sketches. Make repo, do benchmarks.

Do a benchmark of a complete query against all.

`sra_search` is largely I/O bound, with substantial input requirements; this is particularly clear from the YYY% slowdown from loading the 10,000 biggest sketches.

Post-search validation etc. Testing.

It is straightforward to use sourmash CLI to query the metagenome data sets to double check magsearch results. This is usually only internal technical validation since magsearch is built on the same code that sourmash uses, but is a recommended first step because the sourmash UX is better and the output is richer (e.g. weighted abundances,e tc.)

FracMinHash generally and Branchwater specifically have been validated in a more scientific sense primarily by mapping reads. This is discussed further below.

Sra_search is inexpensive and supports exploratory queries

Estimate cost of a run. Compare to serratus - cloud compute, data download. Serratus is probably cheaper than $20k now but still expensive.

Discussion

Making large collections of sequencing data easy to search by content is an open problem, and approaches that work for smaller collections rarely scale well, even for current database sizes. New methods that take advantage of specific particularities of the query and desired answer can help bridge the gap between more general methods by allowing filtering large databases, resulting in more manageable subsets that can be used efficiently with current methods. FracMinHash sketches allow calculating similarity and containment between datasets with-out the need to access the original datasets. Because only a fraction of the original data need to be stored, they are good basic components in the implementation of systems that allow searching large collections of datasets.

This has been used in two papers so far - Lumian et al, Viehgewer et al.

We expect more use cases, and more elaborate use cases, to emerge over the next few years. The low cost of search is particularly enabling for exploratory efforts, although the sheer size of the underlying data needed for branchwater continues to present obstacles. We are currently focusing on realtime API access to search methods.

There are several scientific limitations to overcome as well. The current search approach has limited sensitivity to divergent sequence beyond the genus level, and cannot find smaller matches. These are topics for future research and development.

Following up on Branchwater results

Many Branchwater use cases are intended for early-stage hypothesis generation and refinement, i.e. branchwater implements the first part of a “hit to lead” pipeline. Hence Branchwater operates at an early stage in conceptual and concrete workflows. The initial steps immediately after executing Branchwater are (1) choosing a threshold at which to filter results, (2) evaluating the overall results by type of metagenome retrieved, and (3) retrieving the data underlying the matches.

The first analysis step taken is typically picking a more stringent threshold. The default Branchwater threshold is set to 0.01 containment - any metagenome that contains more than 1% of the k-mers in any query is reported. This is done because searches are exhaustive - every query is searched against every subject - and so there is no extra cost beyond the minimal space required to save the results.

Thresholds are typically chosen based on the use case and the expected distribution of the annotated metagenome type (`ScientificName` from the SRA Runinfo database). We have provided a simple script that imports the SRA metadata and summarizes the Branchwater results at the provided threshold (example output).

After filtering, many paths can be taken. A plethora of general purpose bioinformatics tools exist for working with the data from individual metagenomes.

We have built two custom tools in concert with sourmash and Branchwater, genome-grist and spacegraphcats. Genome-grist performs an entirely automated reference-based characterization of individual metagenomes that combines sourmash gather / minimum metagenome cover with mapping; it is described in Irber et al. [6] and was used in Lumian et al. Given that it does download all the data and maps all the reads, it is still relatively lightweight.

spacegraphcats is an assembly-graph based investigative tool for metagenomes that retrieves graph neighborhoods from metagenome assembly graphs for the purpose of investigating strain variation

[13]. It was used in Reiter et al., and Lumian et al. (phormidium paper). It is much heavier weight than genome-grist because it relies on a compact De Bruijn graph, which is expensive to build for many metagenomes.

Design alternatives

Brancwhater is a simple and effective implementation that is easy to analyze algorithmically and supports a number of use cases. However, many improvements are possible: FracMinHash analyses are based on comparing collections of 64-bit integers, and there are many effective tools and approaches for organizing and searching such collections more efficiently than is presently done in Branchwater.

One area for particular improvement is storing and loading sketches more efficiently. The current JSON-based format is convenient for debugging and multi-language interoperability but is extremely inefficient, to put it mildly. Moreover, each file currently contains three k-mer sketches (one per k-mer size), which means approximately 3 times as much data is loaded per query than is actually used. FracMinHash also could support fractional loading, i.e. decreased resolution by loading only the bottom portion of the sketch; this would enable must faster searches albeit at lower resolution. This is not yet supported by the underlying sourmash library.

Currently the data files are organized as flat files in a single directory on a single network file system. There are a variety of practical ways to speed up the search by distributing sketch files across multiple nodes, but this is logistically challenging. In particular, our typical usage involves running branchwater once every few weeks on our HPC, which does not have sufficient local storage to distribute the data sets. In addition, the speed savings from distributing sketches across nodes is unlikely to be rewarding enough to offset the maintenance requirements for a distributed collection of 13 TB of sketches. Future work could include implementation of an automated distribution system, although careful evaluation of the maintenance and update requirements would be needed.

We could also create a simple pre-filter for each file using a data structure with one-sided error. For example, we could create a Bloom filter for each sketch that could be used to estimate containment prior to loading the full sketch file. However, for some potentially common use cases such as queries with many matches, this could add significant I/O without speeding up the actual search.

Building an inverted index that maps hashes to data sets could also enable rapid queries. Two challenges here are the scale of the catalog and the number of data sets; the total number of hashes present in our metagenome catalog is XYZ, across nearly 800,000 data sets.

Despite these many opportunities for optimization, we note that there is a significant benefit to the simplicit of our current approach. In particular, providing the sketches in individual files organized by accession makes it straightforward to access individual sketches by a distinct ID and quickly update the overall metagenome catalog. This is particularly valuable since the sourmash Python package provides a flexible suite of tools for inspecting and manipulating individual metagenome sketches. The lack of auxiliary data structures also avoids expensive load and synchronization steps when adding new datasets. These features are important for downstream user investigation as well as maintainability and correctness, which are important considerations in any scientific software workflow.

Conclusion

We provide a flexible and fast petabase-scale search based on FracMinHash, together with some simple downstream summarization tools and an increasingly mature (but much slower) investigative ecosystem. This supports and enables a wide range of interesting use cases that take advantage of public data; these use cases range from biomedical to ecological to technical (Table 1).

Data availability statement:

All of the original data underlying the Branchwater database is available from the NCBI Sequence Read Archive. A current catalog of the SRA accessions is provided (here). The sketch collection is 7 TB and is available upon request.

References

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

2. **Ultrafast search of all deposited bacterial and viral genomic data**
Phelim Bradley, Henk C den Bakker, Eduardo PC Rocha, Gil McVean, Zamin Iqbal
Nature Biotechnology (2019-02) https://doi.org/gf4x49
DOI: 10.1038/s41587-018-0010-1 · PMID: 30718882 · PMCID: PMC6420049

3. **Improved Search of Large Transcriptomic Sequencing Databases Using Split Sequence Bloom Trees**
Brad Solomon, Carl Kingsford
Journal of Computational Biology (2018-07) https://doi.org/gdxhgz
DOI: 10.1089/cmb.2017.0265 · PMID: 29641248 · PMCID: PMC6067102

4. **Petabase-scale sequence alignment catalyses viral discovery**
Robert C Edgar, Jeff Taylor, Victor Lin, Tomer Altman, Pierre Barbera, Dmitry Meleshko, Dan Lohr, Gherman Novakovsky, Benjamin Buchfink, Basem Al-Shayeb, … Artem Babaian
Nature (2022-01-26) https://doi.org/gn9xsd
DOI: 10.1038/s41586-021-04332-2 · PMID: 35082445

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

6. **Lightweight compositional analysis of metagenomes with FracMinHash and minimum metagenome covers**
Luiz Irber, Phillip T Brooks, Taylor Reiter, NTessa Pierce-Ward, Mahmudur Rahman Hera, David Koslicki, CTitus Brown
Cold Spring Harbor Laboratory (2022-01-12) https://doi.org/gn34zt
DOI: 10.1101/2022.01.11.475838

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

8. **Dashing: fast and accurate genomic distances with HyperLogLog**
Daniel N Baker, Ben Langmead
Genome Biology (2019-12) https://doi.org/ggkmjc
DOI: 10.1186/s13059-019-1875-0 · PMID: 31801633 · PMCID: PMC6892282

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

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

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

12. **Debiasing FracMinHash and deriving confidence intervals for mutation rates across a wide range of evolutionary distances**
Mahmudur Rahman Hera, NTessa Pierce-Ward, David Koslicki
Cold Spring Harbor Laboratory (2022-01-12) https://doi.org/gn342h
DOI: 10.1101/2022.01.11.475870

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