Sourmash MAGsearch Enables Lightweight Petabyte-Scale Sequence Search

This manuscript (permalink) was automatically generated from dib-lab/2022-paper-magsearch-software@cce83ad on October 25, 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

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. 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 (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 cannot possibly cannot encompass the range of data available within. 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. By searching with genomic content of interest, these approaches 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 obscure sequence variation in order to produce consensus reference sequences useful for consistent comparisons across genotypes. 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 (SBT, BIGSI, etc). 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.

Below, we introduce MAGsearch, 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. MAGsearch uses the Rust library underlying the sourmash implementation of FracMinHash to execute massively parallel searches of a presketched digest of the SRA. This addresses many (but not all) of the use cases in Table XXX.

The availability of relatively lightweight content-based search of SRA metagenomes presents a possible solution for a number of use cases (Table XX). Some of these use cases have already been explored with MAGsearch: Viehweger et al. used MAGsearch 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
- (re)emphsaize unassembled, no marker genes
- figure out how to talk about things it can't do - in Table 1, and also caption, and also discussion.
- For discussion: content-based search, including MAGs and things without marker genes.
- stress lightweight resource usage

Table 1 contents to have:

- Biogeography of genomes
  - describe and characterize biogeographical distribution of species and genus;
  - identify potential sampling locations;
  - (Jessica paper)
- Outbreak tracking
  - trace history of pathogen evolution from public data
  - monitor for future outbreaks
  - (Adrian paper)
- Expanding / exploring strain, species, and genus composition for further genomic analysis
  - Newly isolated species and genera
  - SAGs
- Content-based exploration and reannotation of SRA contents vs metadata
  - Host contamination
  - IBD stuff/discovering gut microbiome
- Making other large-scale databases accessible
  - differential privacy
  - commercial privacy
- Post-processing and cleaning MAGs
- Discovering matches in newly public samples; notification service
- Exploring spread of AMR
- Regulatory evaluation of probiotics

Small viral pangenome query comment/Luiz. Association studies?

Anti use cases: small virus search a la serratus. Searching for new members of class or order. Searching for specific functional genes.

Background: FracMinHash and sourmash

FracMinHash is a bottom-sketch version of ModHash that supports accurate estimation of overlap and containment between two sequencing sets (gather paper). In brief, FracMinHash is a lossy compression approach that reduces data sets in size by a scaled factor S. Sketches support estimation of overlap, bidirectional containment, and Jaccard similarity between two data sets. Unlike other common sketching techniques (MinHash, HyperLogLog), FracMinHash supports these operations between two data sets of different sizes, and unlike mash screen and CMash does not require the original data sets.

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. The Python layer provides a larger number of UX conveniences on top of the performant Rust layer. Despite the thread safety of the underlying Rust code, the CLI and Python library operate in single-threaded mode, which limits the utility of sourmash in certain corners of design space.

We note up front that there are several limitations of FracMinHash and sourmash that affect downstream use cases. The typical scaled parameters used below do not work well for genomes < 10kb in size. Nor can divergent genomes be found; based on the k-mer containment to ANI conversion, we find that MAGsearch works well for finding matches to genomes within about 10% ANI of the query. 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 can be overcome by parameter tuning and further research.

MAGsearch 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, e.g. genomes, where for typical parameters above there are ~1k-10k hashes in the sketch. The software provides a variety of in-memory and on-disk data structures for organizing sketches in this size range and can search 100s 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.

This doesn't work at all for searching 800,000 metagenomes. The main challenges are (1) many very large data sets that (2) do not easily fit in memory (3) being queried by 1-1000 query genomes at once. An additional practical challenge was that much of the sourmash CLI and Python library UX is designed for end users and is hence slow and not supportive of batch processing. In particular, the sourmash CLI and Python library is single-threaded.

One possible solution is a parallel workflow / scatter-gather. Overhead on shell commands and workflow coordination was significant, and we decided to implement 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.

Wort and the SRA digest

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` and saved them as individual gzipped JSON files for each input run.

The resulting metagenome data set sizes were XXX TB. (Describe number, sum, average, median, mode of sketch file sizes.)

Implementation of sra_search - rust; uses sourmash core.

The `sra_search` program implements the following steps:

1. Loads the 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

Note that any requested downsampling of sketches is performed dynamically, after load. 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 easily 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 lits 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 along with any matches.

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

Performance - 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.

Sra_search is largely I/O bound. Presumably we could speed it up a bit by distributing sketches to various nodes but in practice this is logistically challenging to coordinate. 13 TB is large. In a cloud environment with fast interconnect other design decisions could be made. But also other query systems could be built (what was I thinking here? :).

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 MAGsearch 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. Here we believe that it's important that the low cost of search means that exploratory efforts can be quickly evaluated. The major obstacle at the moment is that it is not realtime nor can it be run by others without direct command-line access to the 13 TB of data. These are topics for future software engineering development.

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 MAGsearch results

Many MAGsearch use cases are intended for early-stage hypothesis generation and refinement i.e. "hit to lead", and hence MAGsearch is an early stage in conceptual and concrete workflows. Typical immediate concerns after receiving MAGsearch results are (1) what is the right threshold for my results (2) are my results at that threshold valid (3) how do I get my hands on the actual data, not just the sketches.

The first analysis step taken is often picking a threshold. The exact approach taken will vary depending on use case, but many use cases are looking for or expecting specific distributions of ScientificName so we have provided a simple script that imports SRA metadata and summarizes the MAGsearch results at that threshold. (example output)

After that, many paths can be taken.

Most metagenome data sets are Illumina short-read sequencing, and a plethora of general purpose bioinformatics tools exist for mapping and assembling.

Two tools that were developed in concert with sourmash and MAGsearch are 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 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. It

was used in Reiter et al., and Lumian et al. (phormidium paper). It is much heavier weight than genome-grist because it uses assembly graphs.

Design alternatives

Sra search is a simple yet extremely effective initial implementation that supports a number of use cases. As its use increases many improvements are possible.

In practice, FracMinHash consists of comparing collections of 64-bit integers which is in the wheelhouse of computers.

Roads not (yet) taken include:

- Indexing the data sets in some way. Colors present a challenge. (Mastiff)
- Organizing data sets in some way based on content. Clustering and data set organization present a challenge at this scale.
- Putting a Bloom filter in front of each data set, and/or implementing SBT. Unbalanced data sets and additional storage present a challenge.
- Revising on-disk format; progressive loading; etc.

We note that lack of auxiliary data structures is also a feature because it allows us to update the collection quickly.

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 original data is available from SRA
- query for what we search is here (=> zenodo)
- list of data sets we currenlty have indices for is here
- how to get data one at a time via IPFS is here
- bulk data is 13 TB, available upon request/arrangement
- all sketches are CC0

References