Sourmash MAGsearch Enables Lightweight Petabyte-Scale Sequence Search

This manuscript (permalink) was automatically generated from dib-lab/2022-paper-magsearch-software@a5a18c4 on October 11, 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.

A particularly intriguing sort of data is metagenomes, shotgun (random) sequencing of mixtures of microbes from host-associated and environmental samples. These datasets are often large, and can be highly complex, with environmental samples containing genomic sequence from thousands or more species. These samples often contain many microbial species unknown at the time of their deposition as well. The SRA currently contains about 800,000 of these data sets as of ZZ date, and these datasets are an as-yet largely untapped source of information on microbial life. In particular, they are ripe for reuse, since they typically contain culture-independent random samples of an entire microbial community, even when they were collected for a specific purpose.

Mining SRA metagenomes for biological information takes many forms, including reference-based and de novo techniques. De novo techniques focus on isolating new genomes of certain types from large collections. Reference based techniques concentrate instead on identifying new strains and species organisms related to known species using genome references; this has been used on a large scale with serratus. However, all of these techniques are relatively heavyweight and in particular rely on analyzing the original data, which is time consuming and expensive. The diversity of these data prevent approaches developed for single-organism data such as BIGSI from being used. (Also mention raptor? Or others?)

The growth in k-mer sketching approaches that support containment analysis, i.e. looking for small overlaps between two sequencing data sets, provides an opportunity to enable fast queries on a much reduced index of the original data. In particular, the FracMinHash approach supports robust overlap and containment analysis on sketches, and the sourmash software implementation provides a mature CLI and Python/Rust library for generating and investigating FracMinHash sketches.

Here, we describe a fast search tool, MAGsearch, that makes use of precomputed sourmash FracMinHash sketches to search the entire metagenomic SRA in under 24 hours in 50 GB of RAM and 32 threads, i.e. on commodity hardware, 1-1000 microbial genomes at a time. With the present SRA digest, this tool has high sensitivity and specificity for detecting 10kb of larger overlaps, and can also be used to detect larger matches at average nucleotide identities of 90% or lower, i.e. genus level matches.

We developed a new tool for the specific use case of searching the SRA because the general purpose sourmash command-line tool and Python library was not performant enough to search.

Biological use cases for MAGsearch

The availability of relatively lightweight search of SRA metagenomes has opened up a number of use cases, including biogeographical analysis of genomes and pathogen outbreak analysis. And, because sketches can be generated separately and this approach searches only a subset of the information that does not require revealing the full sample data, it can also be applied to making private metagenome databases available for search, e.g. for commercial collections of metagenomes.

features of current MAGsearch: * content-based search, including MAGs and things without marker genes. * lightweight resource usage

- 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 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.

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

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.

Wort and the SRA digest

Over time we (Luiz :)) has sketched the entire microbial SRA (put query here) at DNA k=21,31,51, scaled=1000 with abundance tracking. The metagenome portion of this is approximately 13 TB in size. Sketches are CC0. Discuss availability. Describe number, maximum size, mode of sketches.

Use case: MAGsearch

Given this collection of sketches, collaborators immediately wanted to search it with isolate genome sequences and metagenome-assembled genomes (MAGs). Two early use cases were biogeography and outbreak detection. In both cases we were confronted with the desire to search for 5-100 queries,

MAGsearch represents a specific technical challenge to sourmash

The primary design focus for 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. 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.

One possible solution is a parallel workflow / scatter-gather. Overhead on shell commands and workflow coordination was significant. Instead luiz wrote rust thing described below.

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.

Implementation of sra_search - rust; uses sourmash core.

It loads all the queries into memory and then uses threads to iterate across the provided collection of search sketches.

From luiz thesis: it is possible to leverage the core features to prototype a large scale search method in Rust that can efficiently keep queries in memory and process batches of the metagenome signatures without needing to store them in memory first, loading on demand and releasing resources after they are queried. This approach also benefits from queries being effectively immutable, and so can be easily shared without data races by multiple processing threads.

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.

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.

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.

Immediate validation etc.

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 addressed below.

Practically running MAGsearch

Usually run in snakemake workflow, which sets RAYON threads etc. Here are the repos.

Discussion

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 - (could be moved to discussion)

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.

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.

Lack of auxiliary data structures is also a feature… allows us to update collection quickly.

Conclusion

From luiz thesis: Making large collections of sequencing data searchable 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. Scaled MinHash 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, (controlled with the scaled parameter) they are good basic components in the implementation of systems that allow searching large collections of datasets.

We provide flexible large-scale fast search, together with some simple downstream summarization tools and a more mature (but slower) investigative ecosystem. This supports and enables a wide range of use cases that explore public data, ranging from biomedical to ecological to technical.

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

We expect more use cases to emerge quickly. 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.

Data availability statement.

References