The Journal of Open Source Software

Finch: a tool adding dynamic abundance filtering to genomic MinHashing

Roderick Bovee[1] **and Nick Greenfield**[1]

1 One Codex, San Francisco, California, USA

DOI: 10.21105/joss.00505

Software

- Review
- Repository
- Archive

Submitted: 24 August 2017
Published: 01 February 2018

Summary

MinHash (Broder 1997) is a document similarity estimation technique that has been applied to problems in genomics including sequence search, phylogenetic reconstruction (Ondov et al. 2016; Brown and Irber 2016), and evaluating outbreaks of hospital acquired infections (HAIs) (Sim et al. 2017). We developed the `finch-rs` library (https://github.com/onecodex/finch-rs) and `finch` command line tool for creating, filtering, and manipulating MinHash sketches from genomics data, including both FASTA sequence files and FASTQ raw read data from next-generation sequencing (NGS) instruments. We extend existing MinHash schemes for genomics data with two major additions: (1) calculation of abundances (i.e., minmer counts) during the generation of the MinHash sketches; and (2) adaptive correction of biases introduced due to variable sequencing depths. These features greatly improve the utility of MinHashing when applied directly to raw read data (i.e., FASTQ files) and allows more robust estimation between both isolates and complex metagenomic samples.

Finch and similar genomic MinHashing software works by breaking sequence data up into k-length nucleotide or amino acid subsequences ("k-mers"), computing a hash of each k-mer, and then taking the n lowest hash values. Collectively these n smallest values ("minmers") comprise a "sketch" of the input sample. By default, previous MinHash implementations for genomics data work by creating sketches from *all* k-mers from an input genomic dataset (though the original Mash tool does enable filtering out k-mers that appear only once using a Bloom filter (Ondov et al. 2016)). While this works well for high-quality sequences such as genome assemblies (i.e., FASTA files), it quickly becomes problematic when working with raw FASTQ data where errors from NGS instruments can lead to a far larger number of *unique* observed k-mers than are truly present biologically. Similarly, this also leads to the inclusion of sequencing errors and k-mers from minor community members when comparing complex, mixed genomic samples (i.e., microbiome samples). In both cases, non-representative k-mers (either direct products of sequencing error or low abundance organisms) come to dominate sketches and confound inter-sample distance estimates.

We address this filtering challenge by "over-sketching" the input genomic data. First, we create a sketch substantially larger than the desired final size (n), tracking the abundances of each k-mer in the sketch, and using the abundances in the large sketch to determine a dynamic filtering threshold. We also track how often each k-mer is seen in its forward versus its reverse orientation. These two metrics allow us to both: (1) estimate the empirical sequencing error in the sample and remove k-mers that may not be biologically present; and (2) remove k-mers that exhibit unbalanced forward and reverse orientation ratios. The former addresses the challenges of comparing data sequenced to varying depths (or samples of varying natural complexity) while the latter can correct for errors that may stem from measurement artifacts such as reads that include adapters and barcode sequences. By removing these error k-mers from a final, reduced sketch (size n), `finch`

more robustly estimates distances between sets of both isolates and complex metagenomic samples.

Finch is written in the Rust programming language (Matsakis and Klock 2014) which reduces programming errors through static type checking, allows greater control over performance, and easily integrates into higher-level languages such as Python or R. We have integrated Finch into our One Codex platform (S. S. Minot, Krumm, and Greenfield 2015) and are using it to power clustering and similarity search features.

References

Broder, Andrei Z. 1997. "On the Resemblance and Containment of Documents." In *Proceedings. Compression and Complexity of Sequences 1997 (Cat. No.97TB100171)*, 21–29. Institute of Electrical; Electronics Engineers (IEEE). https://doi.org/10.1109/sequen.1997.666900.

Brown, C. Titus, and Luiz Irber. 2016. "Sourmash: A Library for Minhash Sketching of Dna." *The Journal of Open Source Software* 1 (5). The Open Journal. https://doi.org/10.21105/joss.00027.

Matsakis, Nicholas D., and Felix S. Klock II. 2014. "The Rust Language." *Ada Lett.* 34 (3). ACM:103–4. https://doi.org/10.1145/2692956.2663188.

Minot, Samuel S, Niklas Krumm, and Nicholas B Greenfield. 2015. "One Codex: A Sensitive and Accurate Data Platform for Genomic Microbial Identification." *bioRxiv*. Cold Spring Harbor Laboratory. https://doi.org/10.1101/027607.

Ondov, Brian D., Todd J. Treangen, Pall Melsted, Adam B. Mallonee, Nicholas H. Bergman, Sergey Koren, and Adam M. Phillippy. 2016. "Mash: Fast Genome and Metagenome Distance Estimation Using Minhash." *Genome Biology* 17 (1). Springer Nature. https://doi.org/10.1186/s13059-016-0997-x.

Sim, James Heng Chiak, Cynthia Truong, Samuel S. Minot, Nick Greenfield, Indre Budvytiene, Akshar Lohith, Victoria Anikst, Nader Pourmand, and Niaz Banaei. 2017. "Determining the Cause of Recurrent Clostridium Difficile Infection Using Whole Genome Sequencing." *Diagnostic Microbiology and Infectious Disease* 87 (1). Elsevier:11–16. https://doi.org/10.1016/j.diagmicrobio.2016.09.023.