Low memory, fast, specific, sensitive, multi-reference sequence classification using Bloom filter maps

Justin Chu, Sarah Yeo, Ben Vandervalk, Golnaz Jahesh, Hamid Mohamadi, Chen Yang, Shaun Jackman, Rene Warren, Inanc Birol
Canada's Michael Smith Genome Sciences Centre, British Columbia Cancer Agency, Vancouver, BC V5Z 4S6, Canada. Contact: cjustin@bcgsc.ca

Background

Sequence classification is traditionally performed by alignments of sequencing reads onto a reference sequence set. Although alignment methodologies have the potential to map the location of these reads precisely, this information is not a prerequisite for classification and thus perform more computation than is needed. Hash table based methods provide fast access for classification but require a large amount of memory. To address these shortcomings, we previously proposed an efficient classification method, BioBloom Tools (Chu *et al.* 2014), that uses a low memory, probabilistic set membership query data structure called a Bloom filter.

Using a Bloom filter, elements of a sequence, such as k-mers, are queried to determine whether they are or not members of those decomposed from the reference set. The memory and time benefits of the data structure have spurned the development of classification tools such as FACs (Stranneheim *et. al.* 2010) and BioBloom Tools. However, querying for the set of origin between multiple reference sets requires the construction and usage of multiple Bloom filters leading to O(n) time complexity when querying, where n is the numbers of reference sets/filters. Here we present a new Bloom filter based data structure called a Bloom Map that can act as an associative array, storing and querying the identifier to the set of origin of a specific element in O(1) time .

Results

Conceptually, a Bloom map is a simply a Bloom filter with buckets that are larger than a single bit. In our implementation we first construct a normal Bloom filter by hashing our elements and set the corresponding buckets in the bit vector to 1. We then interleave rank information into the bit vector. Then, we fill an ID array the size of the population count of the filled bloom filter. Finally, we hash the elements again, but this time setting elements in the ID array with the corresponding IDs of the reference sets according to their rank in the bloom filter. This saves memory by effectively reducing the space of each empty bucket to a single bit. To query we check the bit vector and then use the rank information to look up the ID array for the identity of the queried element in O(1) time.

We compared our tool against a metagenomic classification tool called Kraken (Wood & Salzburg 2014), and its spaced seed counterpart Seed-Kraken (Brinda, *et. al.* 2015) on the NCBI bacterial database. Though our tool is designed for general purpose classification, we correctly classified the genus of 97% reads compared to to Kraken's 92% and Seed-Kraken 95%, while utilizing less memory (61 GB RSS + 0GB pre-cache) than both Kraken (73GB RSS + 66GB pre-cache) and Seed-Kraken (71GB RSS + 64GB pre-cache) on a read simulated dataset. The simulated dataset was constructed using dwgsim and on the genomes used in the bacterial database mimicking ~1mil 2x150bp Illumina reads. To investigate specificity we introduced 50282 random sequence reads into our simulated read set; Seed-Kraken incorrectly assigned a single random read to a genus, but both Kraken and our method managed to not

assign any random sequences to a genus. On 8 cores our tool took <2 minutes to run on our simulated 1mil bp dataset.

Hash collisions are dealt with by using logic such as a majority hit rule in addition to assigning heavily colliding reference IDs a mutual collision ID. Other features of our implementation is the utilization of a recursive rolling hash called ntHash for speed, as well as using complementary spaced seeds patterns instead of the traditional use of multiple hash functions to improve both sensitivity and specificity. Unlike Kraken our k-mer/seed sizes do not affect the memory of our method, which gives BBT the potential to reach a higher specificity by using longer seed k-mer/seed sizes.

Conclusions

Sequence classification to a set of known reference sequences has many applications in contamination screening, pathogen detection, metagenomics, and preprocessing for targeted assembly from shotgun sequence data. Here we present an efficient low memory alternative to hash tables for general purpose, multi-reference sequence classification with broad applications, including taxonomic characterization of bio-organisms from metagenomics samples.