Efficient counting of k-mers in DNA sequences using a Bloom filter

Páll Melsted Jonathan K. Pritchard

doi:10.1186/1471-2105-12-333

Abstract

The paper presents a new lightweight method for identifying non-unique k-mers in a DNA sequence, using Bloom filter – a space-efficient probabilistic data structure, which implicitly stores all observed k-mers. According to the authors, proposed approach is applicable to any algorithm based on k-mer counts.

k-mer counting is an important step in many bioinformatics applications: genome assemblers, read error correctors, seed-based read aligners etc. A number of efficient methods have been developed to facilitate this task, for example: *Jellyfish* [1] uses a highly optimized lock-free hash table for k-mers up to 31 bases in length; *DSK* (disk streaming of k-mers) [2] – streams partitioned k-mer hash table from disk, thus counting k-mers only in a fixed amount of memory.

The paper suggests yet another way to reduce memory usage, when counting k-mers. The idea is to use Bloom filter, a probabilistic set-like data structure, which doesn't store it's elements explicitly, hashing them with a fixed number of hash functions into a bit array. *Probabilistic* part is due to hashing: multiple different elements might hash to the same bit in a bit array, resulting in false positives during membership query. Error rate can be controlled by choosing an appropriate number of hash functions and size of the bit array.

Proposed approach is to traverse input sequences two times: first, to fill up a Bloom filter with all observed k-mers, then, to count frequencies of all non-unique k-mers, using a hash table.

Authors provide a proof-of-concept implementation, called *BFCounter*, and compare its memory consumption and running time on whole genome data to the aforementioned Jellyfish and a naive k-mer counter, based on Google's *sparsehash* hash table. The benchmark shows, that BFCounter runs a bit slower than Jellyfish, but uses less memory than both Jellyfish and naive k-mer counter. It's worth pointing, that comparisons were performed on a single dataset with no replicates, which doesn't allow one to draw any statistically valid conclusions from the provided results.

Overall, the idea of using Bloom filter for NGS datasets looks intriguing, but applicability of the proposed approach remains questionable. For instance, even though genome assemblers do include k-mer counting step, de Bruijn graph, used by most assemblers, requires significantly greater amount of memory (one notable exception is *Minia* [1] – a genome assembler, which uses Bloom filter to encode de Bruijn graph). A related class of bioinformatics tools, error correctors, actually depend on non-unique k-mers, because with high probability they are exactly the k-mers to be corrected.

[1] `http://minia.genouest.org/`

References

1. G. Marcais and C. Kingsford. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764–770, Mar 2011.

2. G. Rizk, D. Lavenier, and R. Chikhi. DSK: k-mer counting with very low memory usage. *Bioinformatics*, Feb 2013.