SKE: Ultra fast simultaneous K-mer counting for multiple values of k

Eric Pauley[1], Raunaq Malhotra[1], Guillaume Rizk[3], Paul Medvedev[1], Rayan Chikhi[2], Raj Acharya[1]
[1]Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802
[2]CNRS Bioinformatics, University of Lille 1, France
[3]IRISA, Rennes, France

K-mer counting is an essential pre-processing step in a number of bioinformatics applications, including de novo assembly using De Bruijn graphs, repeats detection, and multiple sequence alignment. For a given collection of strings or reads, k-mer counting involves computing the number of times every unique substring of length k occurs in all the strings. Additionally, as DNA is double stranded, the counts of a k-mer and its reverse complement are combined together and reported as counts for the canonical k-mer (lexicographically smaller of the two k-mers). However, when working with RNA-seq data, non-canonical counts can be valuable too. It is now possible to obtain a large number of reads (millions to billions) in a single next-generation sequencing run which necessitates the availability of fast and efficient k-mer counting tools [1,2,3,4,5]. Most applications perform k-mer counting for a number of distinct values of k. For example, De Bruijn graph assemblers try a number of k-values for de novo assembly. Currently, a k-mer counting tool such as KMC2[2] or DSK[1] is run multiple times for computing k-mer counts for multiple values of k, which requires accessing the large dataset of reads every time. However, given counts of a k-mer, it is possible to derive the counts of n-mers, where n<k, from the counts of k-mers with minimal disk-IO overhead.

We propose Suffix K-mer Extrapolation, SKE, an algorithm which takes existing non-canonical k-mer counts of size k and computes counts of any n-mers where n<k. Non-canonical counts for length k are obtained using an existing tool such as DSK. The non-canonical counts for an n-mer can be computed from k-mer counts by truncating all k-mers to size n, and combining the counts of same n-mers. The only additional counts that are missed come from suffixes of sizes less than k from each read. We compute the counts for suffixes of lengths less than k separately and combine them with the non-canonical k-mer counts.

The suffix of length (k-1) base pairs (referred as (k-1)-read suffix) is extracted from each read and stored in partitions on disk. Each partition is iterated and n-mers are extracted starting at each index of the (k-1)-read suffix and ending at the final base pair. These n-mers are sorted, combined, and written to disk. A second step of algorithm takes existing non-canonical k-mer counts and the counts of n-mers sorted in the partitions, merges the sorted partitions from both files into a continuous stream of k-mers, then truncates every k-mer down to each requested size n-mer, combining duplicate counts before saving. By counting n-mers from (k-1)-read suffixes in addition to k-mer counts, we obtain correct counts for any length. Whereas existing solutions such as DSK [1] or KMC2 [2] have computational complexity $O(x*bp)$ where x is the number of n-mer sizes to compute and bp is the number of base pairs in the input reads, we achieve complexity $O(bp + x*d)$ where d is the number of distinct k-mers in the largest size counted. Because a significant portion of the time is spent merging (k-1)-read suffix counts with k-mer counts, which takes time linearly proportional to the number of sequences, SKE becomes more efficient as read length increases and error rate goes down.

We have benchmarked our algorithm on reads of both E.coli DNA (read length 151, 1.2GB FASTA, Ecoli_DH10B_110721) and human DNA (read length 250, 250GB FASTA, HG002_NA24385). SKE performed counting faster than KMC2 on human and E.coli reads. For E.coli, all tests were performed on a quad-core server node with 2GB RAM and disk capable of 102MB/s sequential write. Because SKE uses partitioning similar to DSK it benefits greatly from faster disk IO. Counts were performed for k-mer lengths 8-31 using DSK, KMC2, and SKE. SKE took 322s to perform counts for all k-values, including the initial 31-length counting time using DSK. KMC2 took 657s, and DSK took 1139s. The growth rate for all algorithms is linear with number of k-values being counted, though SKE has a higher constant time requirement. Human counting was performed on an equivalent machine with memory limit changed to 8GB. For DSK and KMC2 counts were performed for lengths 8,15,20,25,31, and were used to interpolate the times required for lengths 8-31. These times were then added to get the total estimated time. SKE took 3217 min including 31-length DSK counting to perform all counts from 8-31, KMC2 took 3481 min, and DSK took 21880 min. SKE shows substantial improvement in counting speed over existing solutions and demonstrates the utility of algorithms targeted at multi k-value counting.

The source for SKE can be found at https://github.com/ericpauley/ske

The current work was supported by NSF grants: 1421908,1533797,1356529,1439057, and 1453527.

References

[1]Rizk G, Lavenier D, Chikhi R: DSK: k-mer counting with very low memory usage. Bioinformatics. 2013, 29 (5): 652-653. 10.1093/bioinformatics/btt020.
[2]Deorowicz, Sebastian, et al. "KMC 2: Fast and resource-frugal k-mer counting." *Bioinformatics* 31.10 (2015): 1569-1576.
[3]Marçais, Guillaume, and Carl Kingsford. "A fast, lock-free approach for efficient parallel counting of occurrences of k-mers." *Bioinformatics* 27.6 (2011): 764-770.
[4]Melsted, Pall, and Jonathan K. Pritchard. "Efficient counting of k-mers in DNA sequences using a bloom filter." *BMC bioinformatics* 12.1 (2011): 1.
[5]Deorowicz, Sebastian, Agnieszka Debudaj-Grabysz, and Szymon Grabowski. "Disk-based k-mer counting on a PC." *BMC bioinformatics* 14.1 (2013): 1.