FQSqueezer: *k*-mer-based compression of sequencing data

Abstract

Modern sequencing instruments create a massive amount of data that must be processed. the advanced data compression of FASTQ files has gotten a lot of attention. Current algorithms, on the other side, are still flawed, and the best tools create very large archives. FQSqueezer is a novel data compression algorithm capable of processing single- and paired-end reads of varying lengths. It is based on the well-known prediction by partial matching and dynamicMarkov coder algorithms from the world of general-purpose compressors. The compression ratios are often tens of percent higher than what state-of-the-art software can provide. Large memory and time requirements are disadvantages of the proposed system.

Introduction

Genome sequencing has progressed to the point that it is now a mature technology with various medical applications. Illumina's instruments generate the vast majority of available data at a low cost. The sequenced reads are short but of high quality. The PacBio or Oxford Nanopore 3rd generation instruments can produce much longer readings, but at a much lower throughput and with much lower efficiency.

In this article, we propose a novel compression algorithm for FASTQ files. The main novelty is in the compression of DNA bases, for quality scores and read identifiers we follow similar strategies. Our algorithm, FQSqueezer, make use of the ideas from the prediction by partial matching (PPM) and dynamic Markov coder (DMC) general-purpose methods. the PPM-like strategy to sequencing reads would be, very hard and likely unsuccessful. There are four main reasons for that. First, PPM algorithm should construct a dictionary of all already seen strings of length up to some threshold. Second, the PPM algorithms often need many accesses to the main memory. Third, sequenced data contain errors that should be corrected. Fourth, the PPM algorithms usually learn slowly.

To address these issues, we created a collection of fixed-k dictionaries for k-mers (k-symbol long substrings) We use a much more aggressive approach to estimating the likelihood of symbols appearing. As a result, compression is greatly improved. Finally, we perform some kind of error correction when storing k-mers in dictionaries.

Proposed ideas are that longer (assembled) fragments are of much better quality than reads, so small differences between very similar fragments are usually due to variations between organisms. In sequencing datasets, we usually work with reads originating from a single genome and differences are in majority due to sequencing errors. A compressor of sequencing data should be designed in a different way than a compressor of assembled genomes or its parts. We greatly developed the previously listed ideas and suggested new approaches in this article. The most significant are the organizing of large dictionaries and the design of PPM-like probabilistic estimation. We designed the entire FASTQ compressor system.

The main asset of FQSqueezer is its compression ratio. It is a few times slower than the mentioned competitors in compression and much slower in decompression. Our tool has, however, also some drawbacks in terms of speed and memory usage.

Related work

> the storage and transfer of sequenced (and mapped) reads will consume a lot of money and could be a dominant factor in total costs related to sequencing. Therefore,

there is a lot of research was made to overcome this problem.

The first step was application of gzip, a general-purpose compressor. about 3-fold reduction of files was remarkable. The main problem of gzip is that it was designed mainly for textual data, or more precisely, for data with textual-like redundancy types.

The next step was an invention of specialized algorithms taking into account types of redundancies specific for FASTQ files. The details of the proposed algorithms were different, but in general the authors tried (with some exceptions) to compress the reads locally, i.e., not looking for the large-scale relations between the reads.

After that, the authors introduced a variant of the Burrows–Wheeler transform to find overlaps between reads. For human reads with 40-fold coverage they were able to spend about 0.5 bits per base, which was a significant improvement.

In the following years, other researchers explored the concept of using minimizers, short lexicographically smallest, substrings of sequences, to find reads from close regions. The key observation was that if two reads originate from the close regions of a genome their minimizers are usually the same.

The recent experiments suggest that reduction of quality score resolution has very little impact on the quality of variant calling.it was shown that even more aggressive reduction to just two quality values can be justified, at least in some situations. Moreover, there are several algorithms like QVZ and Crumble that perform advanced analysis of quality scores to preserve only the most important information.