ANNUAL REVIEWS

Annual Review of Biomedical Data Science

Genomic Data Compression

Mikel Hernaez,[1] Dmitri Pavlichin,[2] Tsachy Weissman,[2] and Idoia Ochoa[3]

[1]Carl R. Woese Institute for Genomic Biology, University of Illinois at Urbana–Champaign, Urbana, Illinois 61801, USA; email: mhernaez@illinois.edu

[2]Department of Electrical Engineering, Stanford University, Stanford, California 94305, USA

[3]Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Urbana, Illinois 61801, USA

Annu. Rev. Biomed. Data Sci. 2019. 2:19–37

First published as a Review in Advance on March 18, 2019

The *Annual Review of Biomedical Data Science* is online at biodatasci.annualreviews.org

https://doi.org/10.1146/annurev-biodatasci-072018-021229

Keywords

genomic data, compression, storage

Abstract

Recently, there has been growing interest in genome sequencing, driven by advances in sequencing technology, in terms of both efficiency and affordability. These developments have allowed many to envision whole-genome sequencing as an invaluable tool for both personalized medical care and public health. As a result, increasingly large and ubiquitous genomic data sets are being generated. This poses a significant challenge for the storage and transmission of these data. Already, it is more expensive to store genomic data for a decade than it is to obtain the data in the first place. This situation calls for efficient representations of genomic information. In this review, we emphasize the need for designing specialized compressors tailored to genomic data and describe the main solutions already proposed. We also give general guidelines for storing these data and conclude with our thoughts on the future of genomic formats and compressors.

1. INTRODUCTION

In the year 2000, US President Bill Clinton declared the success of the Human Genome Project (1), calling it "the most important, most wondrous map ever produced by humankind" (2). This was at the end of a project that took almost 13 years to complete and cost $3 billion (around $1 per base pair).

Fortunately, sequencing cost has drastically decreased in recent years. While in 2004 the cost of sequencing a whole human genome was around $20 million, in 2008 it dropped to $1 million, and in 2015 to a mere $1,000 (see **Figure 1**). This decrease in cost, together with advances in sequencing technology, has allowed the field of personalized medicine to rapidly develop, enabling the design of individualized paths to help patients mitigate risks, prevent disease, and treat it effectively when it occurs. As a result, massive amounts of genomic data are being generated. At the current rate of growth (sequencing data are doubling approximately every seven months), more than an exabyte of sequencing data per year will be produced, approaching zettabytes by 2025 (3).

Often these data are unique, in that the samples are not available for resequencing. Moreover, the tools that are used to process and analyze the data improve over time, and thus it will likely be beneficial to revisit and reanalyze the data in the future. For these reasons among others, long-term storage with convenient retrieval is required. In addition, the acquisition of the data is highly distributed, which demands a large bandwidth to transmit and access large quantities of information through the network. This situation calls for state-of-the-art, efficient compressed representations of massive biological data sets, which not only can alleviate the storage requirements but also can facilitate the exchange and dissemination of these data. In addition, compression schemes exploit the redundancy and the underlying structure of the data; hence, the specialized compressors for genomic data can discover hidden structure, which can then be used for analysis or can give insight into the nature of the genome.

This undertaking is of paramount importance, as the storage and acquisition of the data are becoming a major bottleneck, as evidenced by the recent flourishing of cloud-based solutions

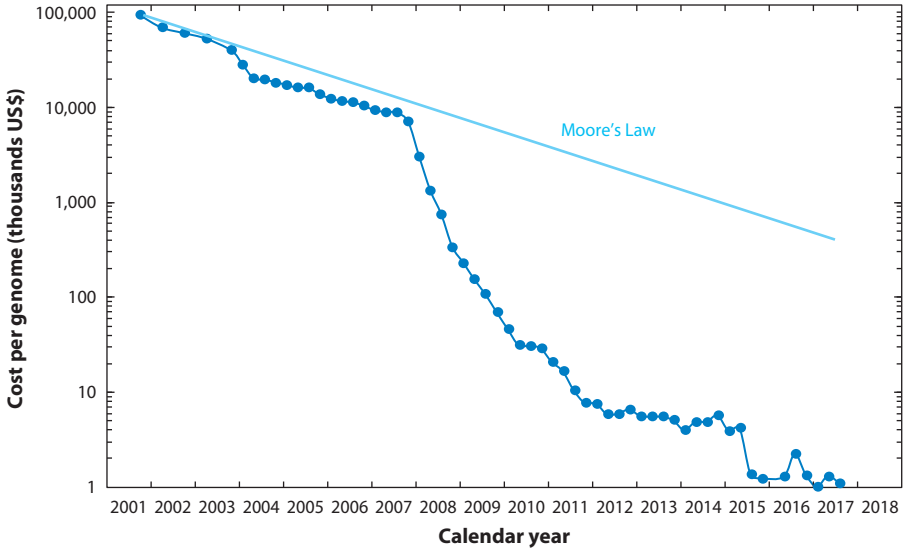

Figure 1

Overview of the decrease in sequencing cost. As can be observed, the rate of this price drop is surpassing Moore's law. Data from Reference 4.

Figure 2

Short-read next-generation sequencing (NGS) technologies require a library preparation of the DNA sample, which is preceded by a DNA preprocessing step that includes cutting the DNA into small fragments. These are then used as input into the sequencing machine, which performs the sequencing in parallel.

enabling processing data directly in the cloud. For example, companies such as DNAnexus (**http://www.dnanexus.com**), GenoSpace (**http://www.genospace.com**), Genome Cloud (**http://www.genome-cloud.com**), and Google Genomics (**https://cloud.google.com/genomics**), to name a few, offer solutions to perform genome analysis in the cloud.

Most of the genomic data being stored and analyzed to date comprise sequencing data produced by next-generation sequencing (NGS) technologies, with short-read technology being the most prominent. These technologies require some level of DNA processing and library preparation suitable for sequencing (see **Figure 2**). As a consequence, instead of the whole genome sequence, NGS short-read technologies produce a collection of millions of small fragments termed reads, which can be thought of as strings randomly sampled from the sequenced genome (see **Figure 3**). For most technologies, the length of these reads is of a few hundred base pairs, which is generally significantly smaller than the genome itself (e.g., a human genome is composed of around 3×10^9 base pairs).

To account for the possible mistakes in the readout of the sequencing signal, sequencers generate, in addition to the reads (i.e., the nucleotide sequences), quality scores that reflect the level of confidence in the readout of each base. The raw sequencing data are therefore mainly composed of the reads and the quality scores and are stored in the widely accepted FASTQ format. Today, the latest sequencing systems can sequence in a single run the equivalent to 48 whole human genomes at 30× coverage. This accounts for 20 billion paired-end reads of length 2×150, or equivalently, 6 TB of FASTQ files (5).

The raw sequencing data typically undergo the analysis pipeline depicted in **Figure 4**. First, the reads contained in the FASTQ file are aligned to a reference genome. In brief, the alignment process infers, for each read, the corresponding location in the reference sequence from which that read was generated (or that no such region exists). In addition to the mapping location, the

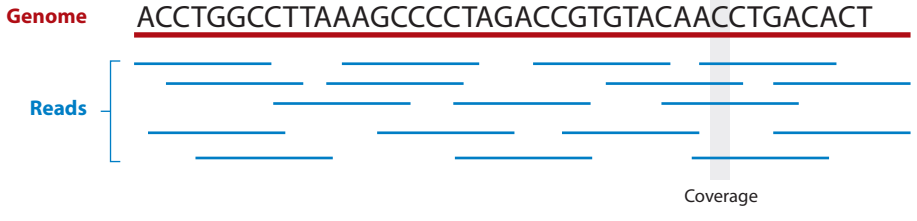

Figure 3

The reads output by the sequencing machine (*blue*) correspond to fragments of the genome. The coverage indicates the number of reads on average that were generated from a random location of the genome.

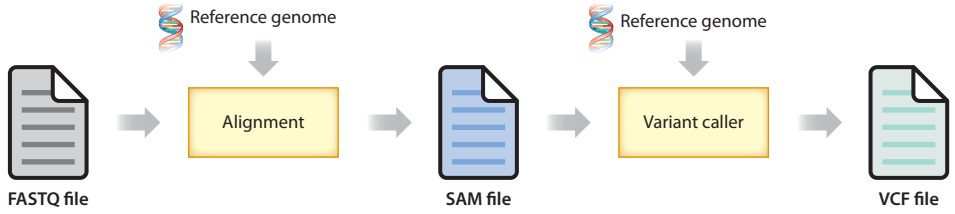

Figure 4

The typical analysis pipeline for genome sequencing. It consists of aligning the reads to a reference sequence, followed by some postprocessing of the aligned file. Then variant calling is performed to find the biological variants with respect to the reference sequence. The input to the pipeline is typically a FASTQ file containing the raw data, and the output is a VCF file containing the found variants. Abbreviations: SAM, sequence alignment map; VCF, variant call format.

alignment also generates the mismatching information, if any, together with some extra fields. This alignment information, together with the original reads and quality scores, is stored in the SAM (sequence alignment map)/BAM (binary alignment map) format (6) (BAM is the binarized and compressed version of the SAM file). These files, which are heavily used by most downstream applications, are extremely large—typically hundreds of gigabytes (for a 30× human genome).

After the alignment, variant calling is performed, which seeks the variants (discrepancies) between the original genome and the reference sequence used for alignment. These variants are normally due to single-nucleotide polymorphisms (SNPs) (i.e., a single nucleotide variation) and insertions and deletions (termed INDELs). The set of called variants, together with some extra information such as the quality of the call, are stored in a VCF (variant call format) file (7). For human genomes, a VCF file may contain around three million variants, most of them due to SNPs (two human genomes are about 99.9% identical). The size of this file is on the order of a gigabyte. With the information contained in the VCF file, the original genome can potentially be reconstructed (this process is called assembly), yielding the assembled genome.

Ideally, it would suffice to store just the VCF file, as it contains all the relevant information regarding the sequenced genome, hence eliminating the need for storing the intermediate FASTQ and SAM files, which are generally prohibitively large. However, as already outlined above, several reasons make this impossible. For example, the alignment tools and variant calling programs keep improving over time, and thus reanalyzing the raw data can yield new variants that were initially undetected. Thus, there is a pressing need for compressing these files to ease the storage and transmission of the data.

Although there are general-purpose compression schemes like gzip (**http://www.gzip.org**), bzip2 (**http://www.bzip.org**), and 7zip (**http://www.7zip.org**) that can be directly applied to any type of genomic data, they do not exploit the particularities of these data, yielding relatively low compression gains (8–10). With this gap in compression ratios in mind, several specialized compression algorithms have been proposed in recent years. These algorithms can be classified into two main categories: (*a*) compression of raw NGS data (namely FASTQ and SAM files) and (*b*) compression of assembled genomes.

In the following review, we revisit the main algorithms proposed in both of these categories and give some guidelines about how the data should be stored. We conclude with thoughts on the future of genomic formats and compressors, including the upcoming standard for efficient genomic information representation, named MPEG-G, which is being developed under the umbrella of ISO (International Organization for Standardization).

```
@SRR0626347 13976/1
TGGAATCAGATGGAATCATCGAATGGTC
+
IIIHIIHABBBAA=2))!!!(!!!((!!
```

Figure 5

Example of a FASTQ file entry corresponding to a read of length 28 and quality scores in the scale Phred+33.

2. COMPRESSION OF RAW SEQUENCING DATA

The raw sequencing data are stored in the FASTQ format, widely accepted as a standard for storing sequencing data. FASTQ files consist of separate entries for each read, each consisting of four lines (see **Figure 5** for an example). The first line contains the identifier, the second line contains the nucleotide sequence (i.e., the read itself), the third line generally contains only the character +, and the last line contains the quality scores (there is a quality score per nucleotide in the read). The quality scores are represented with ASCII characters in the FASTQ file. Specifically, the quality score Q is the integer mapping of P (the probability that the corresponding base call is incorrect), and it is usually represented in the Phred scale (11), given by $\lfloor Q = -10 \log_{10} P \rfloor$. Different NGS technologies use different scales, Phred+33 being the most common one, which corresponds to values of Q in the range from 33 to 73.

Compression of the raw sequencing data therefore involves compression of the identifiers, the reads, and the quality scores. However, there has been more interest in the compression of the reads and the quality scores, as they take most of the space and carry the most relevant information. **Figure 6** shows a time line with the algorithms proposed for compression of entire FASTQ files, as well as those that focus only on the compression of the reads or the quality scores. In what follows we focus our attention to the best performing tools.

Recently, there has been a boost in the compression ratio of the reads by reordering them prior to compression. The reads are stored in the FASTQ file in no particular order; hence, in principle, one can think of them as a set rather than as an ordered list. In addition, the original order is usually irrelevant to later analysis and so is safe to discard. Nevertheless, this can be regarded as a form of lossy compression, since the original data are not recovered exactly. The main gain comes from reordering reads by similarity, such that reads are efficiently represented based on their predecessors. Algorithms that exploit reordering of the reads include, but are not limited to, ORCOM (26), MINCE (34), SCALCE (17), BEETL (16), and HARC (42). Among these, HARC offers the best compression performance, followed by ORCOM. In particular, HARC compresses the nucleotide sequences of *Homo sapiens* data of 13× coverage from 42 GB (uncompressed) to 1.5 GB. As coverage increases, better gains are obtained. For example, 50× data are compressed from 167 GB to 2.9 GB, and 107× data from 349 GB to 4.1 GB (see Reference 42 for details). These results correspond to a 1.7× improvement over ORCOM on average. Finally, FaStore (50) and SPRING (49) are full FASTQ file compressors whose read compressors improve upon ORCOM and HARC, respectively (see the sidebar titled FASTQ File Compressors).

Due to their higher entropy and larger alphabets, quality scores have proven more difficult to compress than the reads. In addition, there is evidence that quality scores are inherently noisy, and downstream applications that use them do so in varying heuristic manners. As a result, lossy compression of quality scores (as opposed to lossless) has emerged as a candidate for boosting compression performance, at a cost of introducing some distortion (i.e., the reconstructed quality scores may differ from the original ones). Dedicated quality value compressors are marked with red boxes in **Figure 6**. The best performing algorithms in this category are QVZ (32, 39), Quartz

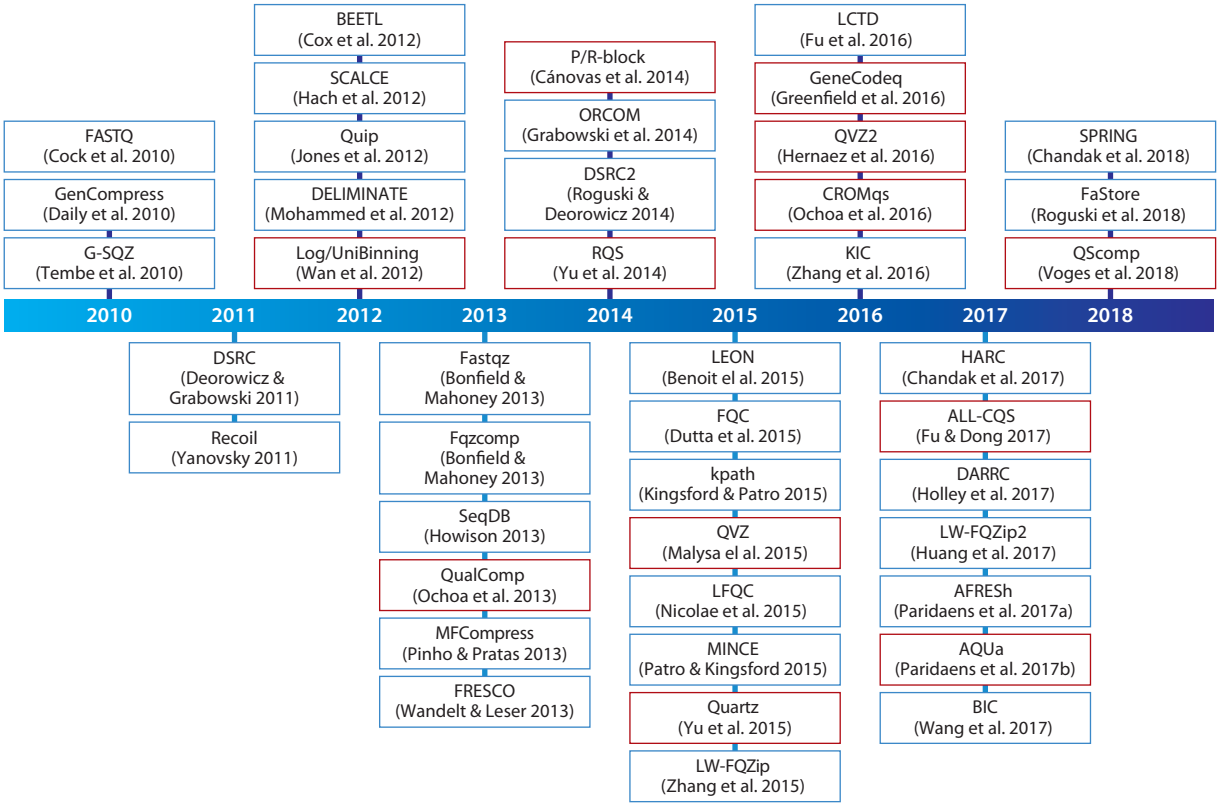

Figure 6

A time line of the proposed methods for compression of FASTQ files. Some of the methods focus on the compression of the unaligned sequences (reads) or the quality scores, rather than on the whole FASTQ file. Red boxes denote dedicated methods for the compression of quality values from unaligned data.

(36) and P/R-block (25). In addition, QVZ is the only algorithm in this category that also provides lossless compression. It requires significantly less memory than Quartz, and it is able to operate at any point in the rate distortion curve, making it the preferred compressor for quality scores in state-of-the-art FASTQ file compressors. QVZ models the quality scores as a Markov model of order one and generates a codebook composed of several quantizers optimized for the computed empirical statistics. In particular, there are as many quantizers as positions in the read, and each quality score is quantized using as context the previous quantized quality score within the same read. This design choice comes from the realizations that quality scores are highly correlated with neighboring positions, and that quality scores tend to deteriorate at the end of the read. Finally, the quantized quality scores are encoded by means of an adaptive arithmetic encoder, using the appropriate models. By tuning the lossyness of the algorithm, QVZ is able to operate everywhere from lossless to almost zero bits per quality score.

Traditionally, lossy compressors have been analyzed in terms of their rate distortion performances. Such an analysis provides a yardstick for comparing lossy compressors of quality scores that is oblivious to the multitude of downstream applications, which use the quality scores in different ways. However, the data compressed are used for biological inference. Researchers are thus interested in understanding the effect that the distortion introduced in the quality scores has on

FASTQ FILE COMPRESSORS

Among complete FASTQ file compressors, FaStore (50) and SPRING (49) are currently the state of the art, with the latter offering better compression performance, as well as more compression modes. Although both algorithms deliver their best performance when the reads are reordered, SPRING also supports lossless compression (hence allowing the original ordering of the reads to be recovered). FaStore employs an improved version of ORCOM (26) for compression of the reads, whereas SPRING is based on HARC (42). For quality score compression, FaStore supports QVZ (32) as well as binary thresholding and Illumina's proposed binning, in which quality scores are quantized to eight values. SPRING supports the same lossy modes, with the only difference being that once the quality scores are quantized (with any of the above methods), the quantized values are encoded by means of BSC (block sorting compression; **https://github.com/IlyaGrebnov/libbsc**) rather than by an arithmetic encoder. Finally, both algorithms support discarding the identifiers at the time of compression and storing only the information pertaining to the pairing information. When losslessly compressed, the identifiers are tokenized, and each token is compressed based on the previous identifier (to exploit the redundancy across them) and encoded by means of an adaptive arithmetic encoder.

Both algorithms recommend discarding the original order of the reads, applying quantization to the quality scores, and retaining only the pairing information of the identifiers. The reason is that this is generally the information needed by most of the genomic applications such as alignment, assembly, or variant calling. In this mode of operation, FaStore and SPRING compress 195 GB of 25× whole-genome human FASTQ data from Illumina's NovaSeq™ 6000 sequencer to 10 GB and 5.6 GB, respectively. A similar file but with 100× coverage is compressed from 787 GB to 20 GB by SPRING, around 1.4 times smaller than the FASTQ compressor FaStore. SPRING achieves this improvement while using comparable computational resources. For example, the 25× FASTQ file is compressed in about 2.5 h and decompressed in less than 1 h, using 30 GB of memory to compress and 12 GB to decompress. FaStore is slightly faster, needing slightly less than 2 h to compress and about 15 min to decompress (in the same machine), and it uses about 40 GB for both compression and decompression. Finally, in lossless mode, SPRING compresses the above-mentioned data sets from 195 GB and 787 GB (uncompressed) to 7 GB and 26 GB, respectively.

the subsequent analysis performed on the data, rather than a more generic measure of distortion, such as rate distortion.

With this in mind, there have been recent efforts and interest in obtaining a methodology to analyze how lossy compression of quality scores affects the output of variant calling, one of the most widely used downstream applications in practice. Not surprisingly, recent studies have shown that lossy compression can significantly alleviate storage requirements while maintaining a variant calling performance comparable—and sometimes superior—to the performance achieved using the uncompressed data (see Reference 52 and the references therein). For example, QVZ operating at less than one bit per quality score does not degrade the variant calling performance, but offers more than 60% in savings as compared to lossless (39). This phenomenon can be explained by the fact that the data are noisy, and current variant callers do not use the data in an optimal manner. Following this trend, in their latest machines, Illumina is currently delivering quality scores with an alphabet of just 8 or even 4 values (as opposed to 40).

3. COMPRESSION OF ALIGNED SEQUENCING DATA

Aligned sequencing data are stored in the widely accepted SAM format (6). The SAM format is a tab-delimited text format consisting of a header section, which is optional, and an alignment section. If present, the header must appear prior to the alignment section. Each alignment line

```
@RG ID:NA12878 SM:NA12878
@PG ID:bwa PN:bwa VN:0.6.1-r104-tpx
HSQ04:1020 117 chr1 11 35 34M = 1 0 CGTGAGTGGTTAATAGGGTGATAGACCTGTGATC
DDDDDDDDDEEDDCDDDEEEEEEEEEDFFFHHHHJ RG:Z:NA12878 NM:i:2
HSQ04:134 153 chr1 45 37 34M = 1 0 GATCACAGGTCTATCACCCTATTAACCACTCACG
EEEDDDDDDCCEDDB?DDCEEDCB>DCCDBBADD RG:Z:NA12878 XT:A:U NM:i:1 AM:i:0
HSQ04:2073 117 chr1 99 41 34M = 1 0 GAGCTCCCGTGAGTGGTTAATAGGGTGATAGACC
CDDDDDDDDCCC?DDDEEDEEDEFFDCFEHHHGH RG:Z:NA12878 XT:A:U
```

Figure 7

First lines of a SAM (sequence alignment map) file. After the header lines (those starting with @) comes the alignment information for the reads, one per line.

has 11 mandatory fields and an arbitrary number of auxiliary fields. The mandatory fields include, among others, the read (i.e., the sequence of nucleotides), the sequence of quality scores, and the position where the read maps in the reference genome. For a comprehensive overview of the format, we refer the reader to Reference 6. A snapshot of a SAM file is shown in **Figure 7**.

Compression of SAM files is especially pressing, as they are both the largest files (up to terabytes) and the ones generally used in downstream applications. Moreover, the size of these files conveys the major bottleneck for the transmission and handling of NGS data among researchers and institutions; therefore, good compression algorithms are of primal need.

Currently these files are stored in the BAM (6) or CRAM format (53, 54). BAM was released at the same time as its uncompressed counterpart, SAM. However, more than a specialized compression algorithm, BAM is a binarization of the SAM file that uses general-purpose compression schemes as its compression engine. CRAM, on the other hand, is a specialized compressor that is experiencing a wider adoption. CRAM combines a reference-based compression of sequence data with a data format that is directly available for computational use in downstream applications. The first version of CRAM was published in 2011 (53), followed by a more specialized version published in 2014 (54), coined CRAMv3. CRAMv4 is currently being developed (J. Bonefield, personal communication).

Since the first version of CRAM appeared, much effort has been devoted to designing compression schemes for these data (see **Figure 8** and the sidebar titled Compression of SAM Files).

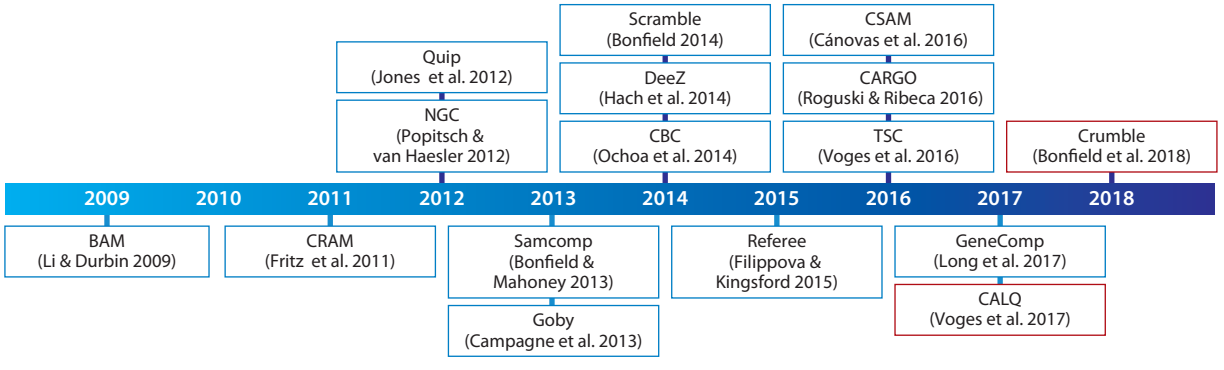

Figure 8

A time line of proposed methods for compression of SAM (sequence alignment map) files. Some of the methods are focused on compression of the aligned sequences (reads) rather than on the whole SAM file. Red boxes denote dedicated methods for the compression of quality values from aligned data.

COMPRESSION OF SAM FILES

Several compressors for SAM files exist in the literature in addition to BAM and CRAMv3 (also known as Scramble), with the main ones being Quip (18), DeeZ (58), and GeneComp (64) (see also **Figure 8**). The main advantage of DeeZ over the other two methods is the support of random access, which allows selective access to the compressed data. This feature may be important, especially as the coverage of the data grows. Most methods support some type of lossy compression of quality values, ranging from simple binning (Scramble) to the more complex QVZ (GeneComp).

Numanagić et al. (2016) offered a comparison of the main existing methods (except for GeneComp) in terms of performance, both in compression ratio and speed (67). For example, a 589-GB SAM file of an *H. sapiens* sequenced with an Illumina HiSeq 2500 at 50× can be further reduced to 121 GB (BAM), 66 GB (CRAMv3), 64 GB (Quip), and 62 GB/53 GB (DeeZ/DeeZ+bzip2). In terms of speed, CRAMv3 is the fastest in both compression and decompression, followed by DeeZ and Quip (BAM is generally faster at decompression). Regarding GeneComp, Long et al. (2017) showed that it offers on average around 10% improvement in lossless compression with respect to DeeZ (64). Finally, the compression time of GeneComp is comparable to that of DeeZ, whereas the decompression is about two times slower.

Compression of the aligned genomic data can be divided into several subproblems of different natures, namely, compression of the aligned reads, compression of the identifiers, compression of the quality scores, and compression of the fields related to the alignment.

As expected, better compression ratios can be achieved when considering aligned reads (in contrast to unaligned ones), as in that case only the differences with respect to the subsequence of the reference to which they aligned need to be stored (8) (this scenario assumes the availability of a reference genome for compression and decompression). These differences, together with the reference sequence, suffice to reconstruct the read exactly. Hence proposed compression methods for aligned reads consist mainly in storing the position where the read maps in the reference, together with the mismatching information. Schemes of this type include, but are not limited to, Samcomp (8), Scramble (54), Quip (18), DeeZ (58), and GeneComp (64). The main differences in performance across these methods come from how well they model the data at hand, and how well each method is able to exploit the redundancy across reads. For example, GeneComp outperforms Samcomp by noticing that reads mapping to the same region (locus) in the genome generally share the same nucleotide differences with respect to the reference, and it uses this information in modeling the data. Another approach to exploiting the redundancy across the aligned reads is to obtain the consensus of the reads mapped to a specific locus and then encode the differences between the consensus contigs and the reference genome. This is the approach used by DeeZ (58). Exact comparison of the compression performance of the reads by the aforementioned methods is somewhat challenging, since they generally compress the entire SAM file and may use different fields to reconstruct the corresponding reads. For example, Samcomp only compresses the necessary information to reconstruct a FASTQ file from the SAM file, and hence it may discard the CIGAR (concise idiosyncratic gapped alignment report) string, whereas methods such as GeneComp compress the CIGAR string even if it is not needed to reconstruct the read itself.

Quality scores from aligned data have the same characteristics as those from the raw data, and hence, they can be compressed with the same tools. However, when dealing with aligned data, lossy compression of quality scores is even more appealing, since when they are losslessly compressed they typically occupy most of the space in the compressed file (up to 70%) (8). The

reason is that the reads can be better compressed, and hence they contribute less space to the compressed file. Even though the lossy compressors designed for the raw data can be seamlessly applied in this scenario, the use of the alignment information has proven to be beneficial in that less degradation is observed in the subsequent variant calling. This is the case with CALQ (65), which introduces an additional dimension to fine-tune quality score quantization in the case of aligned data: The actual quantizer to be applied is chosen per genomic position (or locus). In particular, given the readout nucleotides and quality values, CALQ computes the probabilities of all possible genotypes for every locus covered by aligned reads. Then, using these genotype probability distributions, CALQ computes a quantizer index per locus, which is used to select one of the precomputed quantizers. Quantizers with low resolution will be chosen for loci at which there is enough evidence for a specific genotype, given the readout bases and quality scores, and vice versa. After some further processing, the quantizer indexes and the actual quantized quality scores are compressed using an arithmetic encoder. A similar approach has been recently proposed in Crumble (66).

Compression schemes designed for identifiers of FASTQ reads can be seamlessly applied to the ones present in the SAM file. The main difference is that reads in the SAM file are generally sorted by the position where they map in the reference, and as such, consecutive identifiers tend to be less predictable than in the FASTQ file. Note, however, that this scenario is similar to the case where the reads are reordered in the FASTQ file.

The final pieces to fully reconstruct a SAM file are the fields related to the alignment. These include several compulsory fields such as the mapping quality, the position where the read maps to the reference (POS), or the CIGAR string, which describes some operations needed to reconstruct the read from the reference genome, among others. Some of these fields may be reconstructed from the read and the reference genome, such as the CIGAR string, and others are normally included as part of the compression of the read, such as the POS. Other information generally included as aligned data corresponds to the auxiliary fields, which are represented as TAG:TYPE:VALUE. The number of auxiliary fields present in a SAM file can vary greatly, from none to tens of them.

Compression of these data is done in different ways, from simply applying general compression algorithms, such as gzip, to designing specialized models for each field that are then generally fed into an arithmetic encoder. Other approaches include predicting the value of the corresponding field based on already reconstructed data from the corresponding read, and then sending a flag specifying if the prediction is correct or not (if not, the specific value is compressed). As expected, this approach is valid only for some fields, like the ones related to the mismatching information, such as the CIGAR string and some auxiliary fields.

4. RECOMMENDATION FOR STORAGE OF RAW AND ALIGNED SEQUENCING DATA

As discussed in the above sections, several algorithms have been proposed for the compression of the raw and aligned sequencing data. Generally these algorithms come with several options, allowing the user to choose between different compression modes. These modes vary in terms of compression ratio, compression and decompression speed, memory usage, and, more importantly, what information is preserved in the reconstructed file. Navigating the different options may be overwhelming, especially for users who are either not familiar with the compression setting or not sure about the future use of the data being compressed. With that in mind, in this section we give some guidelines about what we believe are best practices when it comes to the compression and storage of raw and aligned genomic data.

One of the main questions to ask is whether both the FASTQ and the corresponding SAM file need to be stored. The SAM file can always be reconstructed from the FASTQ file as long as the same alignment program, version, and reference sequence are used. Similarly, the FASTQ file can be reconstructed from the SAM file, although in this case the original order of the reads may not be recoverable (depending on whether the identifiers are kept lossless and whether they were sorted alphabetically in the original FASTQ file). Additionally, the unaligned reads must be contained in the SAM file, since otherwise they would be unrecoverable. Another caveat when it comes to reproducibility of the results is that, as shown in References 68 and 50, a different read order in the FASTQ file can produce downstream results that differ from the ones using the original order. The reason is that current tools used for alignment and variant calling may be affected by this ordering. For example, Firtina & Alkan observed a difference of more than 1% in the variants called by the GATK best practice pipeline (68). Hence, when deciding which of the two files to keep, or if both, one needs to think first about whether one file suffices to reconstruct the other.

Another point to take into account is how the data will be used. For example, SAM files are the ones used by most downstream analysis, and therefore, if the data have not been analyzed yet, it would be advisable to keep the SAM file, since otherwise the alignment would need to be run every time, which generally takes a prohibitively long time (especially when dealing with human data). For archival purposes, on the other hand, we would advise storing the raw data, given that its compressed representation is generally smaller than that of the corresponding SAM file, and the latter can always be reconstructed.

Once it is decided which files need to be compressed, the next step is to decide which mode of operation to use for compression. Is the full precision of the quality scores needed? Is there any information from the identifiers needed other than the pairing information? Is the original order of the FASTQ file important? The answers to these questions will mainly depend on the future use of the data being compressed. For example, deciding whether lossy compression of the quality scores is advisable will depend on whether the employed downstream applications use them and, even if they do, whether their performance gets affected by some loss in precision.

Other aspects to take into account when deciding on the mode of operation are the memory usage, the running times of each of the modes, and the final compressed size. Nevertheless, based on our own experience, and as a general guideline, we recommend storing only one of the files (either the FASTQ or the SAM file), retaining only the pairing information of the identifiers, discarding the original order of the reads, and applying some controlled loss in the compression of the quality scores. For the latter, it has been shown that a rate of one bit per quality score has no meaningful effect on variant calling while yielding substantial storage savings (52, 69). Finally, we recommend performing the above-mentioned compression-saving steps prior to the analysis to ensure full reproducibility of the results.

5. COMPRESSION OF WHOLE GENOMES

As sequencing technologies advance, more genomes are expected to be sequenced and assembled in the near future. Hence there is a need for compression of genomes that is guaranteed to perform well simultaneously on different species, from bacteria to humans, and that can ease their transmission, dissemination, and analysis.

Several compression algorithms for assembled genomes have been proposed in the last two decades. These can be divided into two main categories, those that use a reference genome to aid compression, and those that do not. Not surprisingly, algorithms in the former category achieve

better compression than those in the latter. The reason is that reference-based compressors can exploit the similarities across genomes of the same species.

Interest in reference-based compression started to rise in 2009 with the publication of DNAzip (70) and the proposal from Brandon et al. (71). With DNAzip, Christley et al. (70) compressed the genome of James Watson (JW) to a mere 4 MB based on the mapping from the JW genome to a human reference, using a public database of the most common SNPs in humans. Pavlichin et al. (72) improved the DNAzip approach by entropy-encoding the distances between adjacent SNPs. The distribution of these inter-SNP distances was parametrically fit to a double power law distribution (qualitatively similar to the distribution of mobile phone call durations, file sizes on a hard drive, and the number of friends on Facebook). It is an open question what evolutionary dynamics produced the observed power law distribution [several authors (73, 74) have suggested a role for long DNA duplication events].

These two proposals rely on a database of SNPs [dbSNP (75)] and further assume that the mapping from the target to the reference is given, which limits their practicability. Nevertheless, they set a high performance benchmark for whole–human genome compression.

For that reason, state-of-the-art reference-based compressors use only a reference genome (usually of the same species as the genome to be compressed, although not necessarily) as side information. HiRGC (76), iDoComp (77), GeCo (78) and GDC (14, 79) are currently the best-performing algorithms, with HiRGC and iDoComp being slightly ahead of the others in terms of overall compression ratio when the used reference is from the same species as the genome to be compressed (76, 80). In particular, they can compress a human genome of 3.1 GB to around 6 MB. The main advantage comes from a good modeling of the differences encountered between two genomes of the same species. This modeling is then fed into the compression mechanism, which exploits it to achieve better compression ratios (76, 77). This demonstrates the importance of modeling for compression.

In terms of compression/decompression speed, GDC and HiRGC achieve impressive results, followed by iDoComp and GeCo, the latter being significantly slower. For example, iDoComp, GDC, and HiRGC employ generally less than 10 s for both compression and decompression of small genomes (e.g., *Saccharomyces cerevisiae*, with 12.1 million base pairs), and about 2 min for human genomes.

Note that the above discussion focuses on compression of a single genome (given a reference), rather than a collection of them. Algorithms in this category can be seamlessly applied to the latter case by selecting just one genome and using it as the reference for the remaining ones. However, better compression is expected from algorithms specifically designed to deal with a collection of genomes. This is the case with FRESCO (24), GDC-ultra (14), and GDC 2 (79), the latter offering the best compression ratios, about four times better than the other two. In particular, GDC 2 is able to compress a collection of 1,092 human genomes, which occupies 6.7 TB uncompressed, to a mere 700 MB (with a speed of 200 MB/s). This corresponds to a reduction of 9,500 in size, which translates to an average of less than 700 KB per genome. The key to high compression is to look for similarities across the whole collection, not just against one reference sequence, hence exploiting the high redundancy existing in the data.

6. COMPRESSION OF GENOMIC VARIANTS

Large-scale projects involving the study of thousands of samples are becoming increasingly common, with some famous initiatives being the 1,000 Genomes Project (1KGP) (81) and the 100,000 Genomes Project (**https://www.genomicsengland.co.uk/the-100000-genomes-project-by-numbers/**). These projects deliver data in the form of VCF files (7), that is, each

sample is represented in the form of the variants identified with respect to a reference sequence, instead of the corresponding assembled genome. In addition, the scale of current projects such as the Haplotype Reference Consortium (HRC) (82) or the Exome Aggregation Consortium (83) is growing by an order of magnitude. For example, the VCF files of the HRC occupy 4.3 TB and consist of almost 65,000 sequences.

These files are intensively searched to retrieve the variants (the genotype) of a given sample, as well as to retrieve all samples containing a specific variant or set of variants. Hence, solutions for efficient representation of these data should offer random access capabilities. In addition, much larger databases are expected in the near future, and they should be scalable with the number of samples.

One of the first attempts at compression of these files was TGC (thousands genome compressor) (84), which was able to compress a collection of 1,092 human genomes (comprising a total of 40 million variants) to about 400 MB, which corresponds to less than 400 KB per genome on average. Although TGC is currently the best algorithm in terms of compression ratio, it does not support queries on the data, limiting its practicality. GTRAC (genotype random access compressor) (85) was proposed as an extension of TGC, offering random access in both the samples and the variants dimensions. In particular, the genotype of a sample can be retrieved in about a second, and the samples containing a specific variant in tens of milliseconds. The cost of fast query retrieval is that GTRAC's compressed size is about two times that of TGC. After GTRAC, other solutions providing some type of random access were proposed, such as GQT (86), BGT (87), the SeqArray library (88), and GTC (89). Among those, GTC is considered the current state-of-the-art algorithm for compressing the genotype information contained in VCF files, since it offers the best compression performance while providing selective access to the compressed data over both the samples and the variants dimensions. In particular, GTC is able to compress the collection of 1,092 human genomes to 600 MB. Similarly, a collection of 2,500 human genotypes from 1KGP phase 3 containing about 85 million variants was compressed from 853 GB (uncompressed VCF file) to 1.8 GB, and a collection of about 27,000 human genotypes from the HRC was compressed from 4.3 TB to a mere 4 GB, which corresponds to less than 150 KB per genotype. Finally, queries for single variants are done in milliseconds, and for single samples in a few seconds.

As expected, increasing the number of samples to be compressed decreases the average size needed to represent a single sample (and therefore a genome). An interesting question that arises is, What is the ultimate limit of genome compressibility? For example, assuming that there are 80 novel mutations from parents to child (at the upper range of estimates, reviewed in Reference 90), the space to encode the positions of these variations—with both parents' genomes available as a reference—is at most[1] $G * H(80/G) = 266$ bytes, where $G \approx 3 * 10^9$ is the human genome length and $H(p) = -p\log_2(p) - (1-p)\log_2(1-p)$ is the binary entropy function. Assuming all of these mutations are substitutions, it takes $\log_2(3) * 80 = 16$ bytes (since each base pair can change to one of three other possibilities) to describe the mutated nucleotides. There is also on average one recombination event per parent chromosome (91), so assuming a recombination probability of $1/L$ at each position in a chromosome of length L, an upper bound to describe the positions of the recombination events is $2\sum_{i=1}^{23} L_i * H(1/L_i) = 163$ bytes, where L_i is the length of chromosome i and the factor of 2 accounts for both parents. Finally, it takes 23 bits per parent, or about 6 bytes for both parents, to determine which of two homologous chromosomes is passed onto the child. The above numbers add up to about 450 bytes per human genome encoded with respect to both parents' genomes.

[1]This assumes that the 80 novel mutations are uniformly randomly positioned in the genome. Deviations from this assumption would reduce the space required, since the uniform distribution maximizes entropy on a finite set.

As shown above, current work focuses on compression of genotypes only. Future extensions should include compression of the metadata (variant annotations) stored in the VCF files, such as the genotype quality, the read depth, or the allele frequency, among others (for a detailed description, see **https://samtools.github.io/hts-specs/VCFv4.2.pdf**). This information may be important for analysis, for example, to filter the variants based on quality. Quantization of these values could be possible, but it should be done guaranteeing no significant degradation of the output of the applications that use these data.

7. THE FUTURE OF OMICS DATA FORMATS AND COMPRESSION

As demonstrated above, the current situation calls for methods that can efficiently represent the genomic data. As such, the research community has devoted a significant effort to designing specialized compressors for the raw and aligned sequencing data, as well as the assembled genomes (including their representation in the form of variants with respect to a reference genome). Furthermore, compression can speed up the analysis of different computational biology methods (92–94). For example, Loh et al. (92) showed how redundancy in biological data can be exploited by compressing sequence data in a way that allows direct computation on the compressed domain, and Yu et al. (93) added further structure with fractal dimension, introducing a new framework for accelerated similarity searches.

In spite of this, genomic data are still largely stored using compression algorithms that perform poorly in practice, such as gzip for FASTQ files and BAM for SAM files (although the CRAM format is starting to see some widespread adoption). The reason seems to be the availability of multiple algorithms, making the choice difficult, and the lack of performance guarantees, as most of the state-of-the-art compressors are not guaranteed to be maintained long-term. We believe that the latter poses the largest barrier for adoption of novel genomic data compression technologies.

Motivated by this demand, MPEG (Moving Picture Experts Group), an ISO working group that has developed many generations of successful standards that have transformed the world of media from analog to digital, is working with ISO Technical Committee 276/Working Group 5, integrators of biological data workflows, to produce MPEG-G, a new open standard for genomic information representation. The standard is expected to aid significantly in the compression, storage, transmission, and processing of raw and aligned sequencing data (95).

Besides the assurance of compression ratios close to those of the state-of-the art compressors (the technologies proposed by these compressors are part of the MPEG-G specifications, although the first full implementations of MPEG-G codecs are yet to come), the MPEG-G standard will support features associated with sequencing data that are currently partially or not supported by existing formats. Notable use cases to be addressed by MPEG-G include (*a*) selective access to compressed data according to several criteria, (*b*) genomic studies aggregation, (*c*) enforcement of privacy rules, (*d*) selective encryption of sequencing data and metadata, (*e*) annotation and linkage of genomic segments, and (*f*) incremental update of sequencing data and metadata, among others. Furthermore, interoperability and integration with existing genomic information processing pipelines will be enabled by supporting conversion from/to the FASTQ/SAM/BAM file formats. However, a broad adoption of MPEG-G would lead to downstream applications working directly over MPEG-G files, hence improving the efficiency of queries that these applications perform on files containing genomic data. Finally, given its ISO standard designation, it is guaranteed that a file compressed according to the MPEG-G specifications will be accessible for life. However, with respect to royalties, the terms and conditions of the format remain to be decided. Nevertheless, to facilitate the broad adoption of MPEG-G, we advocate for a royalty-free license for research purposes.

As we advance in our understanding of biology, more weight will be put not only on genomic data but also on other types of omics data, such as proteomics, metabolomics, epigenomics, or transcriptomics, to name a few. As an example, we are already seeing an increase in the size and volume of mass spectrometry data sets from proteomics and metabolomics studies (96). In particular, the MassIVE (Mass Spectrometry Interactive Virtual Environment; **https://massive.ucsd.edu/ProteoSAFe/static/massive.jsp**) repository contains more than two million files with 123 TB of storage, and the PRIDE repository (97, 98) contains around 7,000 projects and 74,000 assays. Similarly, there has been a surge of large-scale projects centered around methylation data, such as MethylomeDB (99), DiseaseMeth (100), and MethBase (101). The ENCODE database, for example, contains more than 800 project assays that are related to DNA methylation, amounting to roughly 10% of the total assays and occupying around 78 TB of space (**https://www.encodeproject.org/matrix/? type=Experiment**).

Hence, similar to what happened several years ago in the field of genomics, other omics disciplines are experiencing a need for specialized compression schemes that can cope with the growth of their specific data. As such, several compression schemes tailored to different types of omics data have recently been proposed, such as MassComp (102), MS-Numpress (103), METHCOMP (104), ChIPWig (105), and smallWig (106), among others. However, as in the case of genomics, general compression approaches such as gzip are continuing to be used in lieu of better-performing compressors. We believe the reason is again the lack of standardized formats tailored to a specific type of omics data that address the identified needs and that are made compatible with the corresponding downstream analyses.

DISCLOSURE STATEMENT

Mikel Hernaez, Idoia Ochoa, and Tsachy Weissman have contributed to the development of the MPEG-G standard.

ACKNOWLEDGMENTS

This work was partially funded by grant numbers 2018-182798 and 2018-182799 from the Chan–Zuckerberg Initiative DAF (donor-advised fund), an advised fund from the Silicon Valley Community Foundation (SVCF), and an SRI (special research initiative) grant from the University of Illinois at Urbana–Champaign.

LITERATURE CITED

1. Lander ES, Linton LM, Birren B, Nusbaum C, Zody MC, et al. 2001. Initial sequencing and analysis of the human genome. *Nature* 409:860–921
2. White House Off. Press Secr. 2000. *Remarks made by the President, Prime Minister Tony Blair of England (via satellite), Dr. Francis Collins, Director of the National Human Genome Research Institute, and Dr. Craig Venter, President and Chief Scientific Officer, Celera Genomics Corporation, on the completion of the first survey of the entire Human Genome Project.* Press Release, June 26, White House Off. Press Sec., Washington, DC
3. Stephens ZD, Lee SY, Faghri F, Campbell RH, Zhai C, et al. 2015. Big data: Astronomical or genomical? *PLOS Biol.* 13:e1002195
4. Wetterstrand KA. 2018. *Data sequencing costs: data from the NHGRI Genome Sequencing Program (GSP).* Tech. Rep., Natl. Human Genome Res. Inst., Bethesda, MD, updated 25 April
5. Illumina. 2019. *NovaSeq™ 6000 sequencing system*. Specif. Sheet, Illumina, San Diego, CA, accessed Jan 2. **https://www.illumina.com/content/dam/illumina-marketing/documents/products/datasheets/novaseq-6000-system-specification-sheet-770-2016-025.pdf**

6. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, et al. 2009. The Sequence Alignment/Map format and SAMtools. *Bioinformatics* 25:2078–79
7. Danecek P, Auton A, Abecasis G, Albers CA, Banks E, et al. 2011. The variant call format and VCFtools. *Bioinformatics* 27:2156–58
8. Bonfield JK, Mahoney MV. 2013. Compression of FASTQ and SAM format sequencing data. *PLOS ONE* 8:e59190
9. Zhu Z, Zhang Y, Ji Z, He S, Yang X. 2013. High-throughput DNA sequence data compression. *Brief. Bioinform.* 16:bbt087
10. Deorowicz S, Grabowski S. 2013. Data compression for sequencing data. *Algorithms Mol. Biol.* 8:25
11. Cock PJ, Fields CJ, Goto N, Heuer ML, Rice PM. 2010. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Res.* 38:1767–71
12. Daily K, Rigor P, Christley S, Xie X, Baldi P. 2010. Data structures and compression algorithms for high-throughput sequencing technologies. *BMC Bioinform.* 11:514
13. Tembe W, Lowey J, Suh E. 2010. G-SQZ: compact encoding of genomic sequence and quality data. *Bioinformatics* 26:2192–94
14. Deorowicz S, Grabowski S. 2011. Robust relative compression of genomes with random access. *Bioinformatics* 27:2979–86
15. Yanovsky V. 2011. ReCoil—an algorithm for compression of extremely large datasets of DNA data. *Algorithms Mol. Biol.* 6:23
16. Cox AJ, Bauer MJ, Jakobi T, Rosone G. 2012. Large-scale compression of genomic sequence databases with the Burrows–Wheeler transform. *Bioinformatics* 28:1415–19
17. Hach F, Numanagić I, Alkan C, Sahinalp SC. 2012. SCALCE: boosting sequence compression algorithms using locally consistent encoding. *Bioinformatics* 28:3051–57
18. Jones DC, Ruzzo WL, Peng X, Katze MG. 2012. Compression of next-generation sequencing reads aided by highly efficient de novo assembly. *Nucleic Acids Res.* 40:e171
19. Mohammed MH, Dutta A, Bose T, Chadaram S, Mande SS. 2012. DELIMINATE—a fast and efficient method for loss-less compression of genomic sequences: sequence analysis. *Bioinformatics* 28:2527–29
20. Wan R, Anh VN, Asai K. 2012. Transformations for the compression of FASTQ quality scores of next-generation sequencing data. *Bioinformatics* 28:628–35
21. Howison M. 2013. High-throughput compression of FASTQ data with SeqDB. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 10:213–18
22. Ochoa I, Asnani H, Bharadia D, Chowdhury M, Weissman T, Yona G. 2013. QualComp: a new lossy compressor for quality scores based on rate distortion theory. *BMC Bioinform.* 14:187
23. Pinho AJ, Pratas D. 2013. MFCompress: a compression tool for FASTA and multi-FASTA data. *Bioinformatics* 30:117–18
24. Wandelt S, Leser U. 2013. FRESCO: referential compression of highly similar sequences. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 10:1275–88
25. Cánovas R, Moffat A, Turpin A. 2014. Lossy compression of quality scores in genomic data. *Bioinformatics* 30:2130–36
26. Grabowski S, Deorowicz S, Roguski Ł. 2014. Disk-based compression of data from genome sequencing. *Bioinformatics* 31:1389–95
27. Roguski Ł, Deorowicz S. 2014. DSRC2 industry-oriented compression of FASTQ files. *Bioinformatics* 30:2213–15
28. Yu YW, Yorukoglu D, Berger B. 2014. Traversing the *k*-mer landscape of NGS read datasets for quality score sparsification. *Res. Comput. Mol. Biol.* 8394:385–99
29. Benoit G, Lemaitre C, Lavenier D, Drezen E, Dayris T, et al. 2015. Reference-free compression of high throughput sequencing data with a probabilistic de Bruijn graph. *BMC Bioinform.* 16:288
30. Dutta A, Haque MM, Bose T, Reddy CV, Mande SS. 2015. FQC: a novel approach for efficient compression, archival, and dissemination of fastq datasets. *J. Bioinform. Comput. Biol.* 13:1541003
31. Kingsford C, Patro R. 2015. Reference-based compression of short-read sequences using path encoding. *Bioinformatics* 31:1920–28
32. Malysa G, Hernaez M, Ochoa I, Rao M, Ganesan K, Weissman T. 2015. QVZ: lossy compression of quality values. *Bioinformatics* 31(19):3122–29

33. Nicolae M, Pathak S, Rajasekaran S. 2015. LFQC: a lossless compression algorithm for FASTQ files. *Bioinformatics* 31:3276–81
34. Patro R, Kingsford C. 2015. Data-dependent bucketing improves reference-free compression of sequencing reads. *Bioinformatics* 31:2770–77
35. Zhang Y, Li L, Yang Y, Yang X, He S, Zhu Z. 2015. Light-weight reference-based compression of FASTQ data. *BMC Bioinform.* 16:188
36. Yu YW, Yorukoglu D, Peng J, Berger B. 2015. Quality score compression improves genotyping accuracy. *Nat. Biotechnol.* 33:240–43
37. Fu J, Ma Y, Ke B, Dong S. 2016. LCTD: a lossless compression tool of FASTQ file based on transformation of original file distribution. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, ed. T Tian, Y Wang, Q Jiang, X Hu, Y Liu, et al., pp. 864–69. New York: IEEE
38. Greenfield DL, Stegle O, Rrustemi A. 2016. GeneCodeq: quality score compression and improved genotyping using a Bayesian framework. *Bioinformatics* 32:3124–32
39. Hernaez M, Ochoa I, Weissman T. 2016. A cluster-based approach to compression of quality scores. In *2016 Data Compression Conference (DCC) Proceedings*, ed. A Bilgin, MW Marcellin, J Serra-Sagrista, JA Storer, pp. 261–70. Los Alamitos, CA: IEEE Comput. Soc.
40. Ochoa I, No A, Hernaez M, Weissman T. 2016. CROMqs: An infinitesimal successive refinement lossy compressor for the quality scores. In *2016 IEEE Information Theory Workshop (ITW)*, pp. 121–25. New York: IEEE
41. Zhang Y, Patel K, Endrawis T, Bowers A, Sun Y. 2016. A FASTQ compressor based on integer-mapped k-mer indexing for biologist. *Gene* 579:75–81
42. Chandak S, Tatwawadi K, Weissman T. 2017. Compression of genomic sequencing reads via hash-based reordering: algorithm and analysis. *Bioinformatics* 34:558–67
43. Fu J, Dong S. 2017. ALL-CQS: adaptive locality-based lossy compression of quality scores. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, ed. X Hu, Y Gong, C-R Shyu, D Korkin, Y Bromberg, et al., pp. 353–59. New York: IEEE
44. Holley G, Wittler R, Stoye J, Hach F. 2017. Dynamic alignment-free and reference-free read compression. In *21st Annual International Conference on Research in Computational Molecular Biology (RECOMB 2017)*, ed. SC Sahinalp, pp. 50–65. Cham, Switz.: Springer
45. Huang ZA, Wen Z, Deng Q, Chu Y, Sun Y, Zhu Z. 2017. LW-FQZip 2: a parallelized reference-based compression of FASTQ files. *BMC Bioinform.* 18:179
46. Paridaens T, Van Wallendael G, De Neve W, Lambert P. 2017a. AFRESh: an adaptive framework for compression of reads and assembled sequences with random access functionality. *Bioinformatics* 33:1464–72
47. Paridaens T, Van Wallendael G, De Neve W, Lambert P. 2017b. AQUa: an adaptive framework for compression of sequencing quality scores with random access functionality. *Bioinformatics* 34:425–33
48. Wang R, Bai Y, Cheng Q, Zang T, Wang Y. 2017. A bucket index correction based method for compression of genomic sequencing data. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, ed. X Hu, Y Gong, C-R Shyu, D Korkin, Y Bromberg, et al., pp. 634–37. New York: IEEE
49. Chandak S, Tatwawadi K, Ochoa I, Hernaez M, Weissman T. 2018. SPRING: a next-generation compressor for FASTQ data. *Bioinformatics*. In press
50. Roguski Ł, Ochoa I, Hernaez M, Deorowicz S. 2018. FaStore–a space-saving solution for raw sequencing data. *Bioinformatics* 34:2748–56
51. Voges J, Fotouhi A, Ostermann J, Külekci MO. 2018. A two-level scheme for quality score compression. *J. Comput. Biol.* 25(10):1141–51
52. Ochoa I, Hernaez M, Goldfeder R, Weissman T, Ashley E. 2016. Effect of lossy compression of quality scores on variant calling. *Brief. Bioinform.* 18(2):183–94
53. Fritz MHY, Leinonen R, Cochrane G, Birney E. 2011. Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Res.* 21:734–40
54. Bonfield JK. 2014. The Scramble conversion tool. *Bioinformatics* 30:2818–19
55. Li H, Durbin R. 2009. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics* 25:1754–60

56. Popitsch N, von Haeseler A. 2012. NGC: lossless and lossy compression of aligned high-throughput sequencing data. *Nucleic Acids Res.* 41:e27
57. Campagne F, Dorff KC, Chambwe N, Robinson JT, Mesirov JP. 2013. Compression of structured high-throughput sequencing data. *PLOS ONE* 8:e79871
58. Hach F, Numanagic I, Sahinalp SC. 2014. DeeZ: reference-based compression by local assembly. *Nat. Methods* 11:1082–84
59. Ochoa I, Hernaez M, Weissman T. 2014. Aligned genomic data compression via improved modeling. *J. Bioinform. Comput. Biol.* 12(6):1442002
60. Filippova D, Kingsford C. 2015. Rapid, separable compression enables fast analyses of sequence alignments. In *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics*, pp. 194–201. New York: Assoc. Comput. Mach.
61. Cánovas R, Moffat A, Turpin A. 2016. CSAM: compressed SAM format. *Bioinformatics* 32:3709–16
62. Roguski Ł, Ribeca P. 2016. CARGO: effective format-free compressed storage of genomic information. *Nucleic Acids Res.* 44:e114
63. Voges J, Munderloh M, Ostermann J. 2016. Predictive coding of aligned next-generation sequencing data. In *2016 Data Compression Conference (DCC) Proceedings*, ed. A Bilgin, MW Marcellin, J Serra-Sagrista, JA Storer, pp. 241–50. Los Alamitos, CA: IEEE Comput. Soc.
64. Long R, Hernaez M, Ochoa I, Weissman T. 2017. GeneComp, a new reference-based compressor for SAM files. In *2017 Data Compression Conference (DCC) Proceedings*, ed. A Bilgin, MW Marcellin, J Serra-Sagrista, JA Storer, pp. 330–39. Los Alamitos, CA: IEEE Comput. Soc.
65. Voges J, Ostermann J, Hernaez M. 2017. CALQ: compression of quality values of aligned sequencing data. *Bioinformatics* 34:1650–58
66. Bonfield JK, McCarthy SA, Durbin R. 2018. Crumble: reference free lossy compression of sequence quality values. *Bioinformatics* 35:337–39
67. Numanagić I, Bonfield JK, Hach F, Voges J, Ostermann J, et al. 2016. Comparison of high-throughput sequencing data compression tools. *Nat. Methods* 13:1005
68. Firtina C, Alkan C. 2016. On genomic repeats and reproducibility. *Bioinformatics* 32:2243–47
69. Alberti C, Daniels N, Hernaez M, Voges J, Goldfeder RL, et al. 2016. An evaluation framework for lossy compression of genome sequencing quality values. In *2016 Data Compression Conference (DCC) Proceedings*, ed. A Bilgin, MW Marcellin, J Serra-Sagrista, JA Storer, pp. 221–30. Los Alamitos, CA: IEEE Comput. Soc.
70. Christley S, Lu Y, Li C, Xie X. 2009. Human genomes as email attachments. *Bioinformatics* 25:274–75
71. Brandon MC, Wallace DC, Baldi P. 2009. Data structures and compression algorithms for genomic sequence data. *Bioinformatics* 25:1731–38
72. Pavlichin DS, Weissman T, Yona G. 2013. The human genome contracts again. *Bioinformatics* 29(17):2199–202
73. Polychronopoulos D, Sellis D, Almirantis Y. 2014. Conserved noncoding elements follow power-law-like distributions in several genomes as a result of genome dynamics. *PLOS ONE* 9:e95437
74. Buldyrev SV. 2006. Power law correlations in DNA sequences. In *Power Laws, Scale-Free Networks and Genome Biology*, ed. E Koonin, YI Wolf, G Karev, pp. 123–64. Boston: Springer
75. Sherry ST, Ward MH, Kholodov M, Baker J, Phan L, et al. 2001. dbSNP: the NCBI database of genetic variation. *Nucleic Acids Res.* 29:308–11
76. Liu Y, Peng H, Wong L, Li J. 2017. High-speed and high-ratio referential genome compression. *Bioinformatics* 33:3364–72
77. Ochoa I, Hernaez M, Weissman T. 2014. iDoComp: a compression scheme for assembled genomes. *Bioinformatics* 31(5):626–33
78. Pratas D, Pinho AJ, Ferreira PJ. 2016. Efficient compression of genomic sequences. In *2016 Data Compression Conference (DCC) Proceedings*, ed. A Bilgin, MW Marcellin, J Serra-Sagrista, JA Storer, pp. 231–40. Los Alamitos, CA: IEEE Comput. Soc.
79. Deorowicz S, Danek A, Niemiec M. 2015. GDC 2: compression of large collections of genomes. *Sci. Rep.* 5:11565
80. Hosseini M, Pratas D, Pinho AJ. 2016. A survey on data compression methods for biological sequences. *Information* 7:56

81. Sudmant PH, Rausch T, Gardner EJ, Handsaker RE, Abyzov A, et al. 2015. An integrated map of structural variation in 2,504 human genomes. *Nature* 526:75–81
82. McCarthy S, Das S, Kretzschmar W, Delaneau O, Wood AR, et al. 2016. A reference panel of 64,976 haplotypes for genotype imputation. *Nat. Genet.* 48:1279–83
83. Lek M, Karczewski KJ, Minikel EV, Samocha KE, Banks E, et al. 2016. Analysis of protein-coding genetic variation in 60,706 humans. *Nature* 536:285–91
84. Deorowicz S, Danek A, Grabowski S. 2013. Genome compression: a novel approach for large collections. *Bioinformatics* 29:2572–78
85. Tatwawadi K, Hernaez M, Ochoa I, Weissman T. 2016. GTRAC: fast retrieval from compressed collections of genomic variants. *Bioinformatics* 32:i479–86
86. Layer RM, Kindlon N, Karczewski KJ, Exome Aggreg. Consort., Quinlan AR. 2015. Efficient genotype compression and analysis of large genetic-variation data sets. *Nat. Methods* 13:63–65
87. Li H. 2015. BGT: efficient and flexible genotype query across many samples. *Bioinformatics* 32:590–92
88. Zheng X, Gogarten SM, Lawrence M, Stilp A, Conomos MP, et al. 2017. SeqArray—a storage-efficient high-performance data format for WGS variant calls. *Bioinformatics* 33:2251–57
89. Danek A, Deorowicz S. 2018. GTC: how to maintain huge genotype collections in a compressed form. *Bioinformatics* 34:1834–40
90. Acuna-Hidalgo R, Veltman JA, Hoischen A. 2016. New insights into the generation and role of de novo mutations in health and disease. *Genome Biol.* 17:241
91. Dumont BL, Payseur BA. 2007. Evolution of the genomic rate of recombination in mammals. *Evolution* 62:276–94
92. Loh PR, Baym M, Berger B. 2012. Compressive genomics. *Nat. Biotechnol.* 30:627–30
93. Yu YW, Daniels NM, Danko DC, Berger B. 2015. Entropy-scaling search of massive biological data. *Cell Syst.* 1:130–40
94. Yorukoglu D, Yu YW, Peng J, Berger B. 2016. Compressive mapping for next-generation sequencing. *Nat. Biotechnol.* 34:374–76
95. Alberti C, Paridaens T, Voges J, Naro D, Ahmad JJ, et al. 2018. An introduction to MPEG-G, the new ISO standard for genomic information representation. bioRxiv 426353. **https://doi.org/10.1101/426353**
96. Csordas A, Ovelleiro D, Wang R, Foster JM, Ríos D, et al. 2012. PRIDE: quality control in a proteomics data repository. *Database* 2012:bas004
97. Martens L, Hermjakob H, Jones P, Adamski M, Taylor C, et al. 2005. PRIDE: the proteomics identifications database. *Proteomics* 5:3537–45
98. Jones P, Côté RG, Martens L, Quinn AF, Taylor CF, et al. 2006. PRIDE: a public repository of protein and peptide identifications for the proteomics community. *Nucleic Acids Res.* 34:D659–63
99. Rigden DJ, Fernández XM. 2017. The 2018 *Nucleic Acids Research* database issue and the online molecular biology database collection. *Nucleic Acids Res.* 46:D1–7
100. Lv J, Liu H, Su J, Wu X, Liu H, et al. 2011. DiseaseMeth: a human disease methylation database. *Nucleic Acids Res.* 40:D1030–35
101. Song Q, Decato B, Hong EE, Zhou M, Fang F, et al. 2013. A reference methylome database and analysis pipeline to facilitate integrative and comparative epigenomics. *PLOS ONE* 8:e81148
102. Yang R, Chen X, Ochoa I. 2019. MassComp, a lossless compressor for mass spectrometry data. bioRxiv 542894. **https://doi.org/10.1101/542894**
103. Teleman J, Dowsey AW, Gonzalez-Galarza FF, Perkins S, Pratt B, et al. 2014. Numerical compression schemes for proteomics mass spectrometry data. *Mol. Cell. Proteom.* 13:1537–42
104. Peng J, Milenkovic O, Ochoa I. 2018. METHCOMP: a special purpose compression platform for DNA methylation data. *Bioinformatics* 34(15):2654–56
105. Ravanmehr V, Kim M, Wang Z, Milenković O. 2017. ChIPWig: a random access-enabling lossless and lossy compression method for ChIP-seq data. *Bioinformatics* 34:911–19
106. Wang Z, Weissman T, Milenkovic O. 2015. smallWig: parallel compression of RNA-seq WIG files. *Bioinformatics* 32:173–80

Annual Review of Biomedical Data Science

Volume 2, 2019

Contents

Errata

An online log of corrections to *Annual Review of Biomedical Data Science* articles may be found at http://www.annualreviews.org/errata/biodatasci