AliCo: a new efficient representation for SAM files

Idoia Ochoa[†], Hongyi Li[†], Florian Baumgarte[*],
Charles Hergenrother[γ,α], Jan Voges[*,α] and Mikel Hernaez[α]

[†]Electrical and Computer Engineering, University of Illinois, Urbana, IL, USA
[*]Institut für Informationsverarbeitung, Leibniz University, Hannover, Germany
[γ]Computer Science, University of Notre Dame, Notre Dame, IN, USA
[α]Carl R. Woese Institute for Genomic Biology, University of Illinois, Urbana, IL, USA
idoia@illinois.edu, hli106@illinois.edu, baumflo@tnt.uni-hannover.de,
chergenr@nd.edu, voges@tnt.uni-hannover.de, mhernaez@illinois.edu

Abstract

As genome sequencing continues to become more cost-effective and affordable, more raw and aligned genomic files are expected to be generated in future years. In addition, due to the increase in the throughput of sequencing machines, the size of these files is significantly growing. In particular, aligned files (e.g., SAM/BAM) are used for further processing of the data, and hence efficient representation of these files is a pressing need. In this work we present AliCo, a new compression method tailored to the aligned data represented in the SAM format. We demonstrate through simulations on existing datasets that AliCo outperforms in compression ratio, on average, the state-of-the-art compressors for SAM files, achieving more than 85% reduction in size when operating in its lossless mode. AliCo also supports a variety of modes for lossy compression of the quality scores, including for the first time the recently proposed lossy compressor CALQ, which uses information from the aligned reads to adjust the level of quantization for each location of the genome (achieving more than 10× compression gains in high-coverage datasets). AliCo also supports optional compression of the reference sequence used for compression, hence guaranteeing exact reconstruction of the compressed data. Finally, AliCo allows to stream the data as it is being compressed, as well as to decompress the data as it is being received, potentially providing significant time savings. AliCo can be accessed at: https://github.com/iochoa/alico

Introduction

Next Generation Sequencing (NGS) is becoming extremely cost-effective, with the cost for sequencing a human genome at around $1,000 [1]. Genome sequencing is therefore becoming more affordable and accessible, which has allowed the generation of massive projects such as the UK10K project [2] or the Personal Genomes Project [3]. As a result, massive amounts of raw and aligned genomic files are being generated at a fast rate [4]. In addition, there has been an improvement in the throughput of the sequencing machines, which directly translates into raw (e.g., FASTQ) and aligned (e.g., SAM/BAM) genomic files of increasing size.

This, together with the fact that the genomic files need to be transmitted, stored, and analyzed, calls for efficient representations of the information contained in them. In particular, after alignment of the raw data to a reference sequence, the aligned data represented in the SAM format [5] generally becomes the input to the downstream analyses. In addition, the size of the SAM file is always larger than that of the

2375-0359/19/$31.00 ©2019 IEEE
DOI 10.1109/DCC.2019.00017

corresponding raw file, being generally twice its size. Hence, in this paper we focus on designing a compression method tailored to the aligned data represented in the SAM format.

The SAM file is a tab-delimited human readable file (see Figure 1 for a snapshot of a SAM file). It contains a varying number of optional header lines that start with symbol @, followed by the alignment information for each read, with one line per read. For each read, the SAM format specifies the following compulsory fields: QNAME, FLAG, RNAME, POS, MAPQ, CIGAR, RNEXT, PNEXT, TLEN, SEQ and QUAL. QNAME, SEQ and QUAL correspond to the raw information, that is, the read identifier, the read, and the quality scores, respectively. FLAG, RNAME, POS and CIGAR have information related to the alignment, such as if the read maps to the reversed complement of the reference (in FLAG), the chromosome to which it maps to (RNAME), the position (POS), and information about the mismatching information (CIGAR). Finally, RNEXT, PNEXT, and TLEN refer to information about the paired read, if available, and MAPQ indicates the mapping quality. Optionally, these mandatory fields can be followed by a varying number of auxiliary fields, as in the example of Figure 1. If present, the number and type of the auxiliary fields can vary between reads, and the order in which they appear may not be the same. For a comprehensive overview of the format, we refer the reader to [5].

```
@HD     VN:1.4   SO:coordinate
@SQ     SN:1     LN:249250621
HS2_098:215  99   1   101  2  30M  =  228  30   TAACCCTAACCCTAACCCGAACCCTAACCC
AAC=AA?)A?EFJB67EGG37E))FFGHHH  X0:i:1  X1:i:333  XG:i:0  AM:i:0  NM:i:1
HS2_098:104  16   1   141  2  30M  =  256  30   CTGGAACACCTAGGCGGTGTAAAACTCTAG
CCC?EF?))?FFEE76EEEEFEAAFFFFFF  SM:i:17 XM:i:6  XO:i:2  XT:A:M
```

Figure 1: Snapshot of a SAM file showing two header lines followed by the information of two aligned reads (one per line).

Compression of aligned data represented in the SAM format has been studied in the past, and as such, several specialized compression algorithms have been proposed in the literature. We refer the reader to [6] for a thoughtful review, in addition to CARGO [7], CSAM [8], TSC [9], and GeneComp [10], which were presented after publication of [6]. Among the proposed algorithms for compression of SAM files, BAM [5], CRAM [11, 12], and DeeZ [13] deserve special attention.

In particular, the BAM format was introduced at the same time as the SAM format, and it has been since then the de-facto choice for storing SAM files. However, more than being a specialized compressor, it is a binarized SAM file compressed with general-purpose compression schemes. CRAM, on the other hand, is a specialized compressor for SAM files that is experiencing a wider adoption. It is generally referred to as the CRAM format. There have been several versions of the CRAM format. The first version was published in 2011 [11], followed by a more specialized version published in 2014 [12], named CRAM v3, with CRAM v4 being currently in the works [14]. In what follows we chose to use CRAM v3 as it is the most stable version

and the default configuration on the HTSlib package. CRAM v3 achieves significant compression gains with respect to BAM. In addition, several tools for downstream analyses accept it as input (generally in addition to the BAM format). Finally, DeeZ was presented as a specialized compressor for SAM files, and it competes directly with CRAM v3 in terms of compression ratio and speed [6].

Both CRAM v3 and DeeZ focus mainly on the compression of the reads and the quality scores, as these data take most of the space of the compressed file. In particular, reads are represented by specifying the chromosome and position to which they map to, together with the mismatch information. Note that given the reference sequence used for alignment, this information suffices to reconstruct the read. Regarding the quality scores, both CRAM v3 and DeeZ support lossless and lossy compression (note that lossy compression has proven to reduce storage costs while keeping a performance on variant calling comparable to that of using the original quality scores [15]). Nevertheless, the manner in which CRAM v3 and DeeZ compress the read alignment information and the quality scores differ significantly, leading to different compression performances.

In particular, CRAM v3 compresses the alignment information of each read separately, without explicitly exploiting the fact that reads mapping to neighbouring positions in the reference tend to share common alignment patterns. DeeZ, on the other hand, generates a consensus sequence based on the reads that map to a specific locus, and then encodes the differences of the consensus sequence with the reference sequence only once. In the event of reads differing from the consensus sequence, the corresponding differences are encoded separately. For lossy compression of quality scores, both CRAM v3 and DeeZ perform a transformation to reduce their alphabet, CRAM v3 based on Illumina's proposed binning, and DeeZ based on the empirical computed frequencies for each value. Another significant difference is that CRAM v3, after separating the data by type, compresses them using a variety of different entropy coders [12]. On the contrary, DeeZ uses a unique compression method for each field in the SAM file. Finally, CRAM v3 may not be lossless, as some fields are modified during compression and therefore are not reconstructed exactly at the time of decompression [6]. In terms of random access, both DeeZ and CRAM v3 provide it as an option by compressing the data in blocks.

In this paper we propose AliCo, a new method for ALIgned data COmpression that incorporates some novel features with respect to BAM, CRAM v3, and DeeZ. In particular, in addition to using specialized models for each data type within the SAM file, AliCo supports optional lossy compression of the quality scores based on both QVZ [16] and CALQ [17], and also supports storing the reference sequence used for compression as part of the compressed file. Note that whereas QVZ has been previously supported in full SAM compressors (e.g., in GeneComp [10]), it is the first time a full SAM compressor incorporates CALQ for lossy compression of quality scores. The main advantage of CALQ comes from the use of the alignment information to tune the level of quantization at each position. Finally, note that incorporating the reference sequence used for compression as part of the compressed output guarantees correct decodability of the compressed data. Not only reference sequences get updated, but also the one used for alignment may differ from the one

used at the time of compression, and hence properly identifying the correct reference sequence needed for decompression may be a challenge. This is specially true for data compressed for long-term storage, as these data is not expected to be accessed constantly. Finally, the proposed compressor AliCo incorporates an SSH framework to facilitate streaming of the data as it is being compressed, hence avoiding having to wait until the compression finishes to transmit the compressed files. This feature also allows the receiving part to start decompressing the file as it receives it. Overall, this can save significant time, especially for large files.

We show through experimental results on real data that the proposed method AliCo in lossless mode achieves significant compression gains with respect to BAM, and moderate gains with respect to DeeZ and CRAM v3. We also show the possible reduction in size achieved when the different lossy modes for the quality scores are employed.

Methods

The information contained in the SAM file for each read (hereafter denoted *record*) is composed of different data types, each with its own characteristics. Therefore, to fully exploit the compressibility of these data, the proposed compressor AliCo uses specialized models for each data type within the SAM file. In addition, these models may depend on information from previously compressed records.

Each record can be divided into the following data types: read identifier, read, quality scores, paired-end read information, mapping quality, flag, and auxiliary fields. In the following we describe the method employed to compress each of them, placing special emphasis on the reads and the quality scores, as they tend to occupy most of the space of the compressed file. We also comment on the compression method for reads that failed to align to the reference sequence but that are included in the SAM file. In addition, we comment on the method used to optionally compress the reference sequence used for compression. No compression is applied to the SAM header section.

Read identifiers: Briefly, the read identifiers are tokenized and then compressed by means of an adaptive arithmetic encoder, which uses a separate model for each token type (tokenizing the read identifiers to aid compression was initially proposed in SamComp [18]). In particular, tokens are classified in the following types: alphabetical, runs of zeros, numerical (with a value representable with 4 bytes), and an extra type for tokens that do not fall into the aforementioned categories. For each token type, we first look for a match with the same token from the previous record. A flag indicating if a match is found is then encoded. If a match is found, we continue to the next token, otherwise we encode the type of token, followed by the information contained in the token. For example, for alphabetical tokens we will encode the length of the token and the string. Additionally, for numerical tokens, if a match is not found, we check if the difference in value with the token from previously compressed read identifier is positive and smaller than 256. If so, we encode this delta rather than the complete value.

Read information: Each read is encoded by specifying the position of the ref-

erence sequence where it maps to, together with the mismatching information, if any. The mismatching information is generally encoded in the CIGAR and auxiliary field MD. However, the CIGAR information might be incorrect in some cases, and the MD field may not be present. In addition, the provided reference sequence may differ from the one used for alignment. For these reasons, and to ensure full recoverability of the data, we follow the method proposed in GeneComp [10], in which the edit distance is computed at the time of compression with the Wagner-Fischer algorithm [19]. The correct CIGAR string is estimated as well, and a flag indicating if it coincides with the original one is encoded. If not, the original CIGAR string is separately encoded. Finally, to encode the position and mismatch information, we use specialized models that make use of the information from previously compressed reads. The rationale is that reads that map to the same region of the genome will share most of the mismatching information, especially as the coverage increases. Hence, using previously compressed reads should give us a good estimate for it.

Quality scores: Quality scores can be either losslessly compressed (by means of QVZ), or lossily compressed either with QVZ or CALQ. The main difference between these two methods is that CALQ uses the alignment information to infer the most appropriate quantizers at each locus, whereas QVZ only uses the information provided by the quality scores themselves. Next we briefly describe both methods.

QVZ models the quality scores per position, assuming a Markov model of order 1. First, the empirical probabilities of the quality scores are computed in a subset of the data. Then, based on these statistics and a user input parameter that specifies the level of lossiness, the encoder designs the tailored quantizers to be used at each position. Note that at each position the number of quantizers is equal to the number of quantized values in the preceding position (to make use of the order-1 Markov model). Once the quantizers are computed, they are used to quantize the quality scores, which are further encoded with an adaptive arithmetic encoder that uses a different model per position and previously quantized value.

CALQ 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, i.e., locus. Broadly described, CALQ first infers the "genotype uncertainty" for each genomic locus l given all bases and quality scores of all reads that overlap locus l. This is done using a statistical genotyping model. Within CALQ, the genotype uncertainty can be regarded as a metric that measures the likeliness that a unique genotype is the correct one. This genotype uncertainty is then used to compute a quantizer index per locus which us used to select a quantizer. Quantizers can in particular be drawn from a set of 7 pre-computed uniform quantizers with 2 to 8 representative values. Specifically, quantizers with low resolution (i.e., few representative values) will be chosen for loci at which there is enough evidence for a specific genotype, and vice versa. Thus, the compressibility of the quality scores associated to each locus l will be driven by the uncertainty on the genotype at that particular locus. After some further processing, the quantizer indexes and the actual quantized quality scores are compressed with arithmetic encoders using pre-computed probabilities.

Auxiliary fields: Auxiliary fields, if present, may occupy a significant part of the SAM file. Therefore, we use a specialized method to compress them (recall that

they are specified as TAG:TYPE:VALUE). We first create a list of the most common auxiliary fields by scanning a subset of the data. We also hard-coded a Look Up Table (LUT) containing the main TAG:TYPEs found in practice. Given an auxiliary field, we first check if it is contained in the created list, and send a flag. If found, we send the pointer, using as context the previously compressed TAG:TYPE. If not found, we check if the TAG:TYPE is contained in the LUT, and send a separate flag. If it is, we send a pointer following the same approach as before. If not contained, we encode it, and add it to the LUT (as the same auxiliary fields tend to appear in multiple records). Finally, the VALUE is encoded using as context the TAG:TYPE, as there is a strong correlation in the values sharing the same TAG:TYPE.

Paired-end read information: The paired-end read information is contained in the fields RNEXT, PNEXT, and TLEN. We compress each of them independently, and in the same manner as proposed in GeneComp [10]. Specifically, for RNEXT, we first indicate if the values "=" or "*" occurred, as their frequency of appearance is high as compared to other values. Otherwise we encode the specific value by means of an arithmetic encoder. For PNEXT and TLEN, we first indicate if the value is "0", due again to its high frequency. If not, the PNEXT value is encoded as the distance with the value in POS, whereas the value of TLEN is directly encoded (both by means of an adaptive arithmetic encoder).

Mapping quality, flag and unmapped reads: The mapping quality and the flag are simply encoded with an adaptive arithmetic encoder that uses a model based on the frequencies of previously observed values. Unmapped reads are encoded with the general compressor gzip, except when the RNAME is specified, in which case we use the method described above for aligned reads.

Reference sequence: Several specialized compressors for aligned sequences have been proposed in the literature, such as XM [20], DNA-COMPACT [21], and GeCo [22]. These methods, although achieving significant compression ratios, tend to be very slow. In AliCo, however, since the size of the reference sequence is generally minimal when compared to the size of the SAM file, we chose to trade-off compression performance for speed. Hence we encode the reference sequence by means of an arithmetic encoder that uses an adaptive order-2 Markov model.

Finally, AliCo supports **streaming** of data as it compresses, as well as decompression before the entire compressed file is received at the receiver. The streaming is based on the SSH protocol, via the *scp* command. AliCo makes use of the APIs provided by the *libssh* library (https://www.libssh.org). The current implementation works by splitting the compressed data in blocks of 500 KB. Each block is then sequentially transmitted independently to the decoder. Note that this means that data can be received in random order; a locking-style synchronization mechanism is used at the decoding process to guarantee that data is decoded in the correct order.

Results and Discussion

In order to assess the performance of the proposed compression method AliCo, we compared it with the state-of-the-art compressors BAM [5], CRAM v3 (via htslib) [12], and DeeZ [13], the latter two without the random access option. Note that

BAM does not use a reference sequence for compression. We selected three short-read whole genome sequencing (WGS) datasets from the MPEG-G Genomic Information Database (`https://github.com/voges/mpeg-g-gidb`) [23] for our study: file NA12878_S1.sam from item 02 (H. Sapiens), item 05 (H. Sapiens), and item 16 (*E. coli* DH10B strain). See Table 1 for more details. All experiments were carried out on a Linux machine with 20 Intel Xeon CPU E5-2698's at 2.20 GHz, and 528 GB of RAM.

Dataset	Species	SAM size [MB]	Number of Chromosomes	Number of Reads	Length of Reads	Coverage
Item 02	*H. sapiens*	589,083	24	1,582,771,014	101	52.3x
Item 05	*H. sapiens*	21,059	29	56,463,236	100	2.3x
Item 16	*E. coli*	5,579	1	13,175,679	150	447.8x

Table 1: Main characteristics of the datasets used for testing.

Lossless compression results

The lossless compression performances of BAM, CRAM v3, DeeZ, and the proposed method AliCo are summarized in Table 2. For each of the considered datasets, we show the size of the original SAM file, as well as the compressed size achieved by the aforementioned methods. We also specify the compression ratio (CR), computed as "size of compressed file" $\times 100/$"size of SAM file". Therefore, the smallest the CR the better in terms of compression performance. As can be observed from the table, BAM and CRAM v3 are outperformed in all cases by both DeeZ and AliCo, with AliCo offering better compression ratio for items 05 and 16 (with a corresponding gain of 6.1% and 2.1%, respectively). For item 2, the AliCo and DeeZ perform similarly. Overall, AliCo demonstrates an average compression ratio of 13.37%.

AliCo can optionally store the reference sequence used for compression as part of the compressed file. For the *H. sapiens* data (items 02 and 05), we use reference sequence hg19.fa, which occupies 3,199 MB uncompressed, and for the *E. coli* dataset (item 16), we employ reference sequence EcoliDH10B.fa, of 4.75 MB. AliCo can compress these two sequences to 687 MB and 1.15 MB, respectively, increasing the compressed files by 1.08% and 0.14%, respectively.

Dataset	SAM	BAM		CRAM v3		DeeZ		AliCo		Gain
	Size [MB]	Size [MB]	CR [%]	Size [MB]	CR [%]	Size [MB]	CR [%]	Size [MB]	CR [%]	[%]
Item 02	589,083	121,709	20.66	105,115	17.84	62,950	**10.69**	63,513	10.78	-0.8
Item 05	21,059	6,188	29.39	3,296	15.65	3,527	16.75%	3,094	**14.70**	6.1
Item 16	5,579	1,372	24.61	1,204	21.60	834	14.96%	817	**14.65**	2.1

Table 2: Compression performance of BAM, CRAM v3, DeeZ and AliCo when operating on lossless mode (the results of AliCo do not include compression of the reference sequence). Best compression ratios (CR) are highlighted in bold. Gain is computed as: 1 - (AliCo CR value)/(min CR value of competitors).

The corresponding compression and decompression speeds are outlined in Table 3. As can be inferred from the table, AliCo needs more time to compress and decompress than BAM, CRAM v3 and DeeZ. Among the latter, BAM offers the best decompression times, followed by CRAM v3, whereas CRAM v3 is faster than BAM at compression. DeeZ is outperformed by both BAM and CRAM v3. Note that BAM requires fewer computations as it utilizes *gzip* along with some preprocessing [10]. In addition, DeeZ runs faster than AliCo because the default mode of its implementation supports four threads that run in parallel. The current implementation of AliCo, however, is single-threaded. Having multiple threads could significantly improve the running times, as each chromosome could be compressed (and decompressed) in parallel. Note also that streaming could start before compression is completed. Similarly, decompression can start as packets are received.

Dataset	BAM		CRAM v3		DeeZ		AliCo	
	CS	DS	CS	DS	CS	DS	CS	DS
Item 02	33.62	200.37	40.91	62.14	32.09	28.21	3.70	4.83
Item 05	25.07	269.99	98.04	155.30	25.07	23.40	3.51	6.05
Item 16	33.21	290.57	52.53	74.39	28.35	26.49	0.87	6.20

Table 3: Compression speed (CS) and decompression speed (DS), in MB/s, of the considered compressors BAM, CRAM v3, DeeZ, and AliCo when operating in lossless mode.

To further analyze the proposed compressor AliCo, in Table 4 we show its compression performance on the main fields of the SAM file for items 05 and 16. In particular, we look at the QNAME, the information needed to reconstruct the reads, and the quality scores. We also show the percentage that each of these fields occupy in the original and compressed files. As expected, the reads and quality scores together occupy more than half of the space of the uncompressed files. Nevertheless, AliCo can achieve much lower compression ratios on reads, due to the smaller alphabet set for reads and the fact that we use a reference sequence to aid compression. As such, the information of the reads occupy less than 4% of the compressed file, whereas the compressed quality scores occupy more than 60% for both datasets. This showcases the benefits of lossy compression on the quality scores to significantly reduce the footprint of the genomic data. Lossy compression of the QNAMEs, when possible, could also provide some savings in space. Note however that this option is currently not supported by AliCo. In the next subsection we explore the lossy options for the quality scores supported by DeeZ and AliCo in more detail.

Lossy compression results

The results of applying lossy compression to the quality scores are summarized in Table 5. We consider DeeZ and AliCo, the latter run with both QVZ and CALQ options. For DeeZ and AliCo-QVZ, we chose the parameters (l and c) so as to achieve around 1 bit/QS, as this value has been shown to be a good trade-off between compression gain and performance on variant calling [15]. This correspond to a compression ratio of around 15% when compared to the original quality scores. AliCo, when run with

	Item 5					Item 16				
Category	SAM	(%)	AliCo	(%)	CR [%]	SAM	(%)	AliCo	(%)	CR [%]
QNAME	1,835	8.71	272	8.79	14.83	247	4.42	55	6.73	22.32
Reads	5,703	27.08	121	3.91	2.12	1,990	35.66	31	3.79	1.56
Quality Scores	5,703	27.08	1,884	60.89	33.03	1,990	35.66	636	77.85	31.98

Table 4: Compression performance of AliCo on some individual fields on items 05 and 16. *SAM* and *AliCo* refer to the sizes before and after lossless compression, and the (%) entries are the percentages these fields occupy in the original and compressed files, respectively. CR indicates the ratio between the compressed and the original size.

CALQ, can achieve better compression ratios, specially as the coverage increases. For example, for item 02, with coverage 52.3×, the compression ratio decreases to 2.33%. This decreased is even more pronounce for item 16 (CR of 0.017%), as expected, since the coverage is 447.8×. Note that as the coverage increases, the information from the bases at each locus may suffice to make a decision on the presence or absence of a variant, and hence the quality scores lose importance. For item 16, this corresponds to quality scores occupying 0.35 MB rather than 636 MB (see Table 4), hence reducing the final compressed file from 817 MB to a mere 182 MB.

Dataset	DeeZ -l 40			AliCo-QVZ -c 0.50			AliCo-CALQ		
	QS CR	bit/QS	Overall CR	QS CR	bit/QS	Overall CR	QS CR	bit/QS	Overall CR
Item 02	14.23%	1.15	45.65%	13.69%	1.11	60.89%	2.33%	0.19	10.38%
Item 05	17.04%	1.38	36.08%	17.70%	1.43	53.5%	14.11%	1.14	42.72%
Item 16	16.90%	1.36	47.16%	16.84%	1.36	52.65%	0.017%	0.0014	0.055%

Table 5: Results for lossy compression in bits per quality score [bit/QS]. *QS CR* represents the ratio between the size of the compressed and original quality scores, whereas *Overall CR* represents the ratio between the size of the compressed QS in lossy and lossless mode.

Conclusion

We have presented AliCo, a new compressor specialized for aligned genomic data represented in the SAM format. On average, AliCo exhibits a compression performance superior to the state-of-the-art compressors, reducing the SAM file size by more than 85%. In addition, AliCo supports novel desirable features, such as optional compression of the reference sequence used for compression, hence guaranteeing perfect decodability at the time of decompression, and streaming capabilities that allow to transmit the data before compression is finished. Similarly, the compressed data can be decoded as it is being received, hence saving significant time. AliCo is also the first SAM compressor to support, in addition to QVZ, the lossy compressor CALQ for the quality scores. The main advantage of CALQ comes from the use of the aligned data to decide how much precision is needed for the quality scores at each locus. As the coverage increases, the gain becomes more apparent. For example, we show that using CALQ reduces the final compressed size by more than 77% when compared to lossless, for a dataset with more than 400× coverage. Future improvements of the proposed method AliCo include support for random access, as well as parallelization to speed up the running time.

Acknowledgments

This work was partially funded by grant numbers 2018-182798 and 2018-182799 from the Chan Zuckerberg Initiative DAF, and an SRI grant from UIUC.

References

[1] National Human Genome Research Institute, ," `https://www.genome.gov/27541954/dna-sequencing-costs-data/`.

[2] The UK10K Consortium, ," `https://www.uk10k.org/`, 2013.

[3] The Personal Genome Project, ," `https://www.personalgenomes.org/us`, 2012.

[4] Zachary D Stephens, Skylar Y Lee, Faraz Faghri, Roy H Campbell, et al., "Big data: astronomical or genomical?," *PLoS biology*, vol. 13, no. 7, pp. e1002195, 2015.

[5] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, et al., "The sequence alignment/map format and samtools," *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, 2009.

[6] Ibrahim Numanagić, James K Bonfield, Faraz Hach, et al., "Comparison of high-throughput sequencing data compression tools," *nature methods*, vol. 13, no. 12.

[7] Łukasz Roguski and Paolo Ribeca, "Cargo: effective format-free compressed storage of genomic information," *Nucleic acids research*, vol. 44, no. 12, pp. e114–e114, 2016.

[8] Rodrigo Cánovas, Alistair Moffat, and Andrew Turpin, "Csam: Compressed sam format," *Bioinformatics*, vol. 32, no. 24, pp. 3709–3716, 2016.

[9] Jan Voges, Marco Munderloh, and Jörn Ostermann, "Predictive coding of aligned next-generation sequencing data," in *Data Compression Conference (DCC), 2016*.

[10] Reggy Long, Mikel Hernaez, Idoia Ochoa, and Tsachy Weissman, "Genecomp, a new reference-based compressor for sam files," in *Data Compression Conference (DCC)*.

[11] Markus Hsi-Yang Fritz, Rasko Leinonen, et al., "Efficient storage of high throughput dna sequencing data using reference-based compression," *Genome research*, vol. 21.

[12] James K Bonfield, "The scramble conversion tool," *Bioinformatics*, vol. 30, no. 19, pp. 2818, 2014.

[13] Faraz Hach, Ibrahim Numanagic, and S Cenk Sahinalp, "Deez: reference-based compression by local assembly," *Nature methods*, vol. 11, no. 11, pp. 1082, 2014.

[14] James K. Bonfield, ," Personal communication, 2018.

[15] Idoia Ochoa, Mikel Hernaez, Rachel Goldfeder, et al., "Effect of lossy compression of quality scores on variant calling," *Briefings in Bioinformatics*, 2016.

[16] Greg Malysa, Mikel Hernaez, Idoia Ochoa, et al., "Qvz: lossy compression of quality values," *Bioinformatics*, vol. 31, no. 19, pp. 3122–3129, 2015.

[17] Jan Voges, Jörn Ostermann, and Mikel Hernaez, "Calq: compression of quality values of aligned sequencing data," *Bioinformatics*, vol. 34, no. 10, pp. 1650–1658, 2017.

[18] James K Bonfield and Matthew V Mahoney, "Compression of fastq and sam format sequencing data," *PloS one*, vol. 8, no. 3, pp. e59190, 2013.

[19] Robert A Wagner and Michael J Fischer, "The string-to-string correction problem," *Journal of the ACM (JACM)*, vol. 21, no. 1, pp. 168–173, 1974.

[20] Lloyd Allison Chris Mears Minh Duc Cao, Trevor I. Dix, "A simple statistical algorithm for biological sequence compression," pp. 43–52, 2007.

[21] Pinghao Li, Shuang Wang, Jihoon Kim, et al., "Dna-compact: Dna compression based on a pattern-aware contextual modeling technique," *PloS one*, vol. 8, no. 11, 2013.

[22] Paulo J. S. G. Ferreira Diogo Pratas, Armando J. Pinho, "Efficient compression of genomic sequences," pp. 231–240, 2016.

[23] Claudio Alberti, Tom Paridaens, Jan Voges, Daniel Naro, et al., "An introduction to mpeg-g, the new iso standard for genomic information representation," *bioRxiv*, 2018.