JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 25, Number 0, 2018
© Mary Ann Liebert, Inc.
Pp. 1–11
DOI: 10.1089/cmb.2018.0065

Research Article

A Two-Level Scheme for Quality Score Compression

JAN VOGES[1] ALI FOTOUHI[2] JÖRN OSTERMANN[1] and MUHAMMED OĞUZHAN KÜLEKCI[3]

ABSTRACT

Previous studies on quality score compression can be classified into two main lines: lossy schemes and lossless schemes. Lossy schemes enable a better management of computational resources. Thus, in practice, and for preliminary analyses, bioinformaticians may prefer to work with a lossy quality score representation. However, the original quality scores might be required for a deeper analysis of the data. Hence, it might be necessary to keep them; in addition to lossy compression this requires lossless compression as well. We developed a space-efficient hierarchical representation of quality scores, QScomp, which allows the users to work with lossy quality scores in routine analysis, without sacrificing the capability of reaching the original quality scores when further investigations are required. Each quality score is represented by a tuple through a novel decomposition. The first and second dimensions of these tuples are separately compressed such that the first-level compression is a lossy scheme. The compressed information of the second dimension allows the users to extract the original quality scores. Experiments on real data reveal that the downstream analysis with the lossy part—spending only 0.49 bits per quality score on average—shows a competitive performance, and that the total space usage with the inclusion of the compressed second dimension is comparable to the performance of competing lossless schemes.

Keywords: quality score compression, variant calling, genomic data management, lossless data compression, lossy data compression, high-throughput sequencing.

1. INTRODUCTION

Sequencing data produced by high-throughput sequencing machines are typically stored in the FASTQ format (Cock et al., 2010). Due to the growing volumes of sequencing data, processing, transmission, and storage of the FASTQ files becomes challenging. Therefore, the compression of data stored in FASTQ files has been receiving great interest in the last years (Numanagić et al., 2016). Compact representations of the data do not only help during storage and transmission by decreasing the required disk space or by enabling the possibility to better manage the available bandwidth, but also help during the analysis of the huge data volumes when the applied compression schemes support functionality such as random access over the compressed data directly. That dimension, namely compressive genomics, has been proposed and discussed in previous studies (Loh et al., 2012; Berger et al., 2016).

[1] Institut für Informationsverarbeitung, Leibniz Universität Hannover, Hannover, Germany.
[2] Electronics and Communication Engineering Department, Istanbul Technical University, Istanbul, Turkey.
[3] Informatics Institute, Istanbul Technical University, Istanbul, Turkey.

FASTQ files include four lines per read. The first and the third line, beginning with the @ and + symbols, respectively, indicate the read identifier and an optional description. The second line lists the read-out nucleotides. For each nucleotide in the second line, a corresponding quality score (QS) Q is stored in the fourth line. The quality scores indicate the accuracy of the base calling by $Q = -10 \cdot \log_{10} P$, where P is the error probability of the base-calling process (Ewing and Green, 1998).

So far, efforts in compressing raw sequencing data stored in FASTQ files have been focusing on compressing the nucleotide sequences, quality scores, and read identifiers separately. This approach yields a better performance than jointly compressing the different streams since these streams have divergent statistical properties. Previous studies on quality score compression can be further separated into two categories: lossy schemes and lossless schemes. The lossy methods achieve much better compression ratios by sacrificing some information. This is done by reducing the alphabet size of the quality scores according to specific quantization methods. Although these lossy approaches help a lot in terms of storage and transmission of the data, the original values might still be required for further analyses (Van der Auwera et al., 2013).

The daily practice in sequencing data analysis starts with regular routines. In further steps of the analysis, deeper investigations are performed on the reads that are mapped to regions of interest detected by these regular routines. Quantized quality scores may work well during the initial processing unless the incorporated quantization does impact further steps significantly. Thus, when the target regions regarding the tested hypothesis become clear, necessity to access the original quality scores of the selected reads may become unavoidable during further downstream analyses. Yet another reason to keep the original values stems from the underlying thought that the original quality scores might be required by new methods in the future. Specifically, in large population genomics projects, the owners of the data may prefer lossless compression techniques. Thus, an approach would be preferable where the users have the choice to work effectively in the first stage with quality scores represented with a lossy scheme, but at the same time have the choice to reach the original values in following analysis steps.

Motivated by this demand, we explore in this study a two-level approach for the compact representation of the quality scores. By using a novel decomposition scheme $\mathcal{D}$, we represent each quality score Q with a tuple $\mathcal{D}(Q) \rightarrow \langle q_1, q_2 \rangle$. The compression of the q_1 values constitutes the first compression level, and compressing the q_2 values creates the second level, where the q_1 values determine the context during the compression of the q_2 sequence. The first level is the lossy representation of the quality scores Q. Thus, working with this level corresponds to a lossy scheme. Given q_1 and q_2, the inverse decomposition $\mathcal{D}^{-1}$ yields the original quality scores by $Q \leftarrow \mathcal{D}^{-1}(q_1, q_2)$. This way, we preserve the capability to extract the original values. With such a two-level approach, both lossy compression and lossless compression of the quality scores can be achieved hierarchically. In the scope of this article, we evaluate the lossy layer in terms of its effect on downstream analyses. The space occupied by the first level and the second levels is expected to be competitive to previously proposed lossless schemes.

2. PREVIOUS STUDIES

In a FASTQ file the alphabet for the nucleotides (i.e., A, C, G, T, and N) is usually much smaller than that of the quality scores, which typically stem from an alphabet of size 40 or 41 (Cock et al., 2010). Thus, quality scores at full resolution are, in general, more difficult to compress. Therefore, the overall success of compressing an input FASTQ file depends more on the representation of the quality scores than on the compression of the nucleotide sequences.

Lossless compression techniques focus on detecting regularities in quality score streams (Wan et al., 2012). For instance, some of the quality scores are likely to be more frequent than others, or several biases may appear in some positions of the reads due to the underlying sequencing technology. Remember that a compression scheme can be viewed as a two-step process, where the first phase is to devise a context model describing the data, and the second phase is to encode the data that are represented with that model using an entropy coder. General-purpose FASTQ compressors mainly differ in their context modeling approaches. The DSRC scheme defines three models for quality score streams, and represents a given quality score sequence according to its best-fitting model (Deorowicz and Grabowski, 2011). SCALCE (Hach et al., 2012) and Quip (Jones et al., 2012) make use of a single standard order-3 context model, and encode every quality score according to its three immediate predecessors. Fastqz (Bonfield and Mahoney, 2013) applies a more complex scheme that uses relations in the near predecessors to define the context of the current quality score.

Lossy compression was considered based on the assumption that the resolution of raw quality scores is much higher than required for accuracy evaluation, and that the tools in the analysis pipelines will not be affected much from a lossy representation. It was proven that this assumption is true, and more than that, actually lossy representations improve the efficiency of downstream analyses in many cases (Yu et al., 2015; Ochoa et al., 2016). The authors (Wan et al., 2012) explored different binning strategies and their effects on the compression efficiency. Besides simple bucketing that uses fixed-length intervals, variable-length intervals inferred through a number of different statistical measures have also been proposed (Cánovas et al., 2014).

Another statistical approach has been introduced with QualComp (Ochoa et al., 2013). QualComp fits a Gaussian distribution to the quality score sequences (i.e., vectors), and provides users with the ability to define the level of acceptable distortion during encoding. According to the specified number of bits to be used per quality score, QualComp performs the optimal alteration of the quality scores such that the mean squared error is minimized according to the precomputed Gaussian model. This idea has been further improved by the more recent QVZ and QVZ 2 compressors (Malysa et al., 2015; Hernaez et al., 2016). Besides the binning and statistical inference approaches, there are other efforts which exploit the information contained in the read-out nucleotide sequences (Janin et al., 2014; Yu et al., 2015; Voges et al., 2017). For example, the Quartz compressor (Yu et al., 2015) sets the quality scores of the most frequent k-mers to a predefined high value with the motivation that if a specific nucleotide sequence is observed many times, then its correctness does not need any further verification from the quality scores. Thus, the quality scores can be set to a fixed value. This way the entropy is reduced and higher compression performance is achieved.

3. PROPOSED METHODS

When an analysis pipeline automatically returns results for a set of reads (stored in a FASTQ file), the analyst usually feels the necessity to perform a verification of these results by investigating the reads together with their associated quality scores. A bioinformatician working on such reads might become suspicious when she observes low-quality scores since those indicate a possible error in the base-calling process, which could have then caused problems in the automatically produced results. Similarly, when quality scores are larger than a threshold, it does not tell much to the analyst in most cases as there appears to be not much practical difference between the 99.999% accuracy with $Q=50$ than 99.9999% with $Q=60$. This difference becomes less and less important as long as the quality scores get higher. On the other side, due to the logarithmic nature of the quality scores, $Q=10$ is quite different from $Q=20$, since the first case implies 90% accuracy, whereas the second indicates 99% accuracy in the base-calling process.

Therefore, it seems that a simple bucketing approach with short intervals for the small quality scores and larger intervals for the higher quality scores might work well in practical analyses. Hence, we propose to decompose a quality score Q into the tuple

$$\mathcal{D}(Q) \to \langle q_1, q_2 \rangle \tag{1}$$

such that

$$q_1 = \text{round}(\sqrt{Q}), \tag{2}$$

$$q_2 = Q - \left(q_1^2 - q_1 + 1\right). \tag{3}$$

Notice that given q_1 and q_2, the inverse decomposition yields the original quality score as

$$Q = \mathcal{D}^{-1}(q_1, q_2) = q_1^2 - q_1 + 1 + q_2. \tag{4}$$

This decomposition is inspired by the representation of integers in an Elias gamma code (Elias, 1975; or its generalization, the Exp-Golomb code, Ostermann et al., 2004). Assume $Q = q_1^2 + c$ with $c = 1 - q_1 + q_2$. If Q is an n-bit binary number, then q_1 is an $n/2$-bit binary number and c lies in the interval $[0, 2b]$. Then q_1 can be encoded using any universal coding. Given q_1, the number of bits necessary to represent c can be determined as $\log_2(2q_1 + 1)$. However, as the scope of this work is the two-level representation of quality

Table 1. An Example Describing the Proposed Representation of Quality Scores

	$(q_1 - 1)$ items						*q_1^2*	*q_1 items*						
Q	30	31	32	33	34	35	**36**	37	38	39	40	41	42	43
q_1	5	6	6	6	6	6	6	6	6	6	6	6	6	7
q_2	9	0	1	2	3	4	5	6	7	8	9	10	11	0

The corresponding squared q_1 value is highlighted in bold.

scores and not the exploration of sophisticated entropy coding schemes, we use the well-known general-purpose compressor bzip2 for the compression of the tuples $\mathcal{D}(Q)$.

Table 1 shows the decomposition of quality scores in the interval [30, 43]. The proposed decomposition creates buckets of length $(2 \cdot q_1)$, where typically $q_1 \in \{6, 7, 8, 9, 10, 11\}$ since in the FASTQ format the quality scores are between 33 and 126 (i.e., in the range of printable ASCII characters). The first $(q_1 - 1)$ of the items in a bucket are promoted to a better quality, whereas the last q_1 are faced with a penalty. Notice that the $(2 \cdot q_1)$ items long bins are relatively short for the smaller q_1 values, which fits to the motivating observation described above.

Without incorporating the q_2 values, the representation of quality scores (only by their corresponding q_1 values) creates a simple lossy scheme. In that sense, a FASTQ file in which all quality scores are changed to their q_1^2 values will exhibit a better compressibility since the alphabet for the quality scores is reduced to at most 6 symbols instead of $94(=126-33+1)$ possible characters. Remember that, in general, the observed number of symbols is around 40 as opposed to the theoretically possible 90+ symbols. Similarly, when the users would like to obtain the capability to retrieve the original scores, then they need to also keep the q_2 sequence. Instead of handling the q_2 sequence as a single stream, which would force the subsequent compressor to assume the most general alphabet for the q_2 sequence, clustering the q_2 values according to their corresponding q_1 values would improve the compression ratio (as the q_1 value in a tuple specifies the exact alphabet for the q_2 values). Thus, for each distinct q_1 value observed in the input FASTQ file, we maintain a separate sequence of q_2 values. Finally, we compress the q_1 values and the multiple q_2 sequences individually. Any general-purpose compressor can be applied. As already mentioned, we prefer to use bzip2. Surely, the users of the proposed system can proceed with different choices at this step.

4. EXPERIMENTAL RESULTS

In this section, we provide experimental results for the evaluation of the proposed compression scheme QScomp. We compare QScomp to three competitors, namely Crumble (https://github.com/jkbonfield/crumble), Quartz (Yu et al., 2015), and QVZ 2 (Hernaez et al., 2016). Table 2 lists the tools, including QScomp, which were selected for the evaluation in this work.

Note that QScomp is the only tool which truly is able to operate in the lossless and in the lossy mode.

The data sets used to evaluate the performance of the selected compression tools originate from the same individual, namely NA12878. For this individual, the National Institute of Standards and Technology (NIST) released a consensus set of variants, which we used for our analyses (Zook et al., 2016). Note that similar analyses were conducted in other works (Alberti et al., 2016; Ochoa et al., 2016; Voges et al., 2017). The selected data sets are shown in Table 3. For more information on the used data sets we refer the reader to the Supplementary Data.

Moreover, for the evaluation of the proposed compression scheme QScomp, we selected three different variant-calling pipelines. The first pipeline is composed of GATK (Van der Auwera et al., 2013) variant

Table 2. Tools Selected for the Evaluation

Tool name	*Tool version*	*Lossless (Y/N)*	*Lossy (Y/N)*
QScomp	ec5c61b	Y	Y
Crumble	0.5	N	Y
Quartz	0.2.2	N	Y
QVZ 2	d5383c6	Y	Y

Table 3. Data Sets Selected for the Evaluation

ID	*Name*	*Technology*	*Coverage*
H01	ERR174324	Illumina HiSeq 2000	14×
H11	SRR1238539	Ion Torrent	10×
H12	Garvan replicate	Illumina HiSeq X	49×

calling (using the HaplotypeCaller tool) and SNP extraction with subsequent filtering of variants using GATK Vector Quality Score Recalibration (VQSR) with four different filter values. The second pipeline is also composed of GATK variant calling using the HaplotypeCaller tool and SNP extraction, but followed by the more traditional hard filtration of variants instead of VQSR. The third pipeline uses Platypus (Rimmer et al., 2014) for variant calling. For the individual commands and tools and auxiliary files used, we refer the reader to the Supplementary Data.

Each of the mentioned pipelines outputs a set of variants in the VCF file format. Subsequently, each set of variants is compared with the consensus set of variants. We perform this comparison using the tool hap.py (https://github.com/Illumina/hap.py) released by Illumina and adopted by the Global Alliance for Genomics and Health (GA4GH). This benchmarking tool outputs the following values for each comparison:

- True Positives (T.P.): All those variants that are both in the consensus set and in the set of called variants.
- False Positives (F.P.): All those variants that are in the called set of variants but not in the consensus set.
- False Negatives (F.N.): All those variants that are in the consensus set but are not in the set of called variants.
- Non-Assessed Calls: All those variants that fall outside of the consensus regions defined by a BED file.

These values are used to compute the following two metrics:

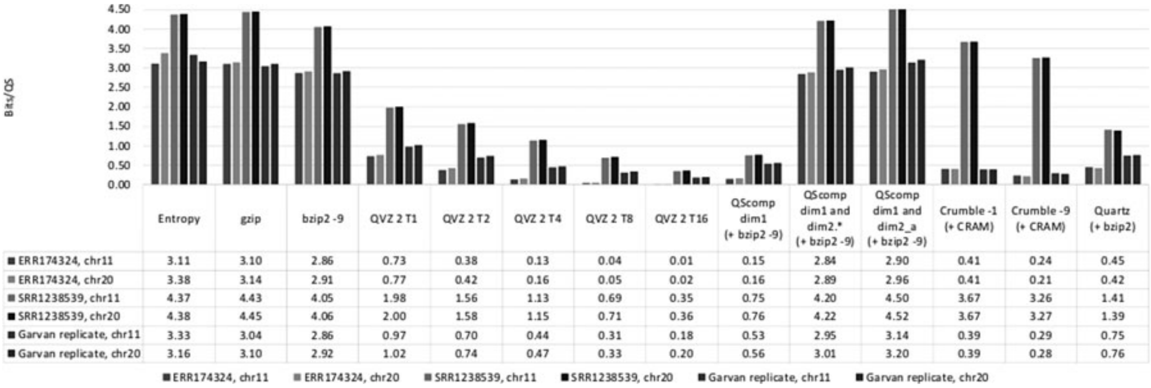
- Recall/Sensitivity: This is the proportion of called variants that are included in the consensus set; that is, $R = {}^{\text{T.P.}}/_{(\text{T.P.}+\text{F.N.})}$,
- Precision: This is the proportion of consensus variants that are called by the variant calling pipeline; that is, $P = {}^{\text{T.P.}}/_{(\text{T.P.}+\text{F.P.})}$.

Finally, we measured the maximum memory usage and the execution time of each tool on each dataset with GNU time.

4.1. *Performance analysis of the proposed scheme*

In this section we first show the compression ratios of all tools and for all datasets from Table 3.

Figure 1 shows the compression results for all tools in bits per quality score. In addition to the compression results for the mentioned tools, we also show the memoryless entropy per original quality score, which is 3.62 bits per quality score, averaged over all data sets. Furthermore, we show the gzip and bzip2 compression results for the raw quality scores, which are 3.54 bits per quality score and 3.27 bits per quality score, also averaged over all data sets.



	Entropy	gzip	bzip2 -9	QVZ 2 T1	QVZ 2 T2	QVZ 2 T4	QVZ 2 T8	QVZ 2 T16	QScomp dim1 (+ bzip2 -9)	QScomp dim1 and dim2.* (+ bzip2 -9)	QScomp dim1 and dim2_a (+ bzip2 -9)	Crumble -1 (+ CRAM)	Crumble -9 (+ CRAM)	Quartz (+ bzip2)
■ ERR174324, chr11	3.11	3.10	2.86	0.73	0.38	0.13	0.04	0.01	0.15	2.84	2.90	0.41	0.24	0.45
■ ERR174324, chr20	3.38	3.14	2.91	0.77	0.42	0.16	0.05	0.02	0.16	2.89	2.96	0.41	0.21	0.42
■ SRR1238539, chr11	4.37	4.43	4.05	1.98	1.56	1.13	0.69	0.35	0.75	4.20	4.50	3.67	3.26	1.41
■ SRR1238539, chr20	4.38	4.45	4.06	2.00	1.58	1.15	0.71	0.36	0.76	4.22	4.52	3.67	3.27	1.39
■ Garvan replicate, chr11	3.33	3.04	2.86	0.97	0.70	0.44	0.31	0.18	0.53	2.95	3.14	0.39	0.29	0.75
■ Garvan replicate, chr20	3.16	3.10	2.92	1.02	0.74	0.47	0.33	0.20	0.56	3.01	3.20	0.39	0.28	0.76

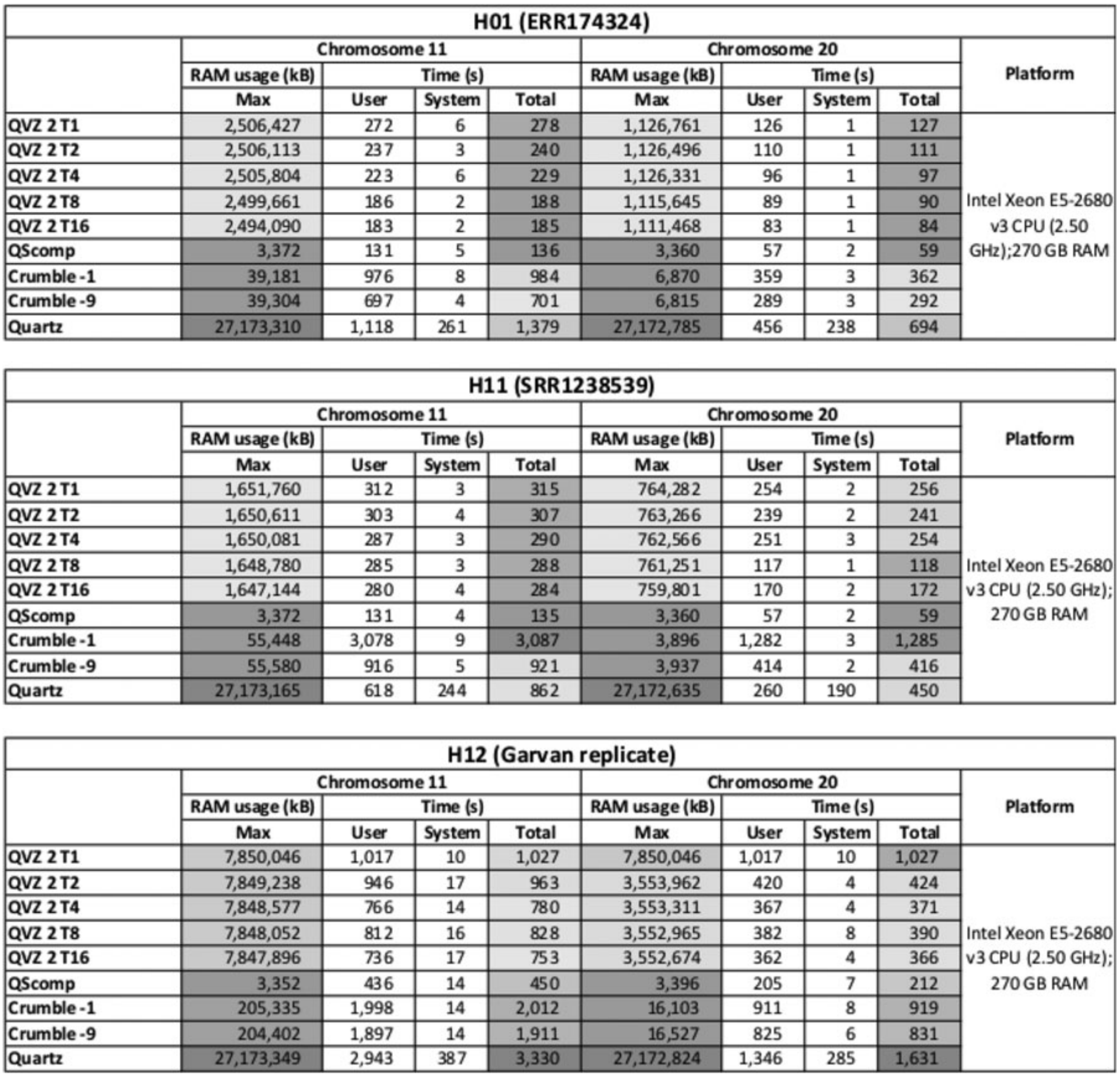
FIG. 1. Compression ratios results.

As shown in Figure 1, the lossy quality score representation obtained using QScomp with subsequent bzip2 compression (i.e., ‘‘QScomp dim1 (+ bzip2 –9)’’) yields 0.49 bits per quality score on average. This result is comparable to the results obtained with QVZ 2 when a target mean squared error (MSE) of 8 (i.e., ‘‘QVZ 2 T8’’) is specified, which yields 0.35 bits per quality score on average.

We can observe from the figure that the lossless quality score representation of QScomp with subsequent bzip2 compression (i.e., ‘‘QScomp dim1 and dim2.* (+ bzip2 –9)’’) is capable of delivering 3.35 bits per quality score, which is slightly below the entropy, as expected. The two-level scheme of QScomp with conditional compression of the second level with respect to first level is slightly superior to just compressing the quality scores with gzip, and comparable to compressing the quality scores with bzip2. Thus, QScomp does not sacrifice the lossless compression performance, while combining the lossless and lossy compression through its unique two-level scheme. We finally show in Figure 1 the results of compressing the joint single sequence of q_2 values (i.e., ‘‘QScomp dim1 and dim2_a (+ bzip –9)’’). This experiment yields 3.53 bits per quality score. These results suggest that the proposed separate compression of multiple q_2 sequences is superior to just compressing the q_2 residues as a single stream.

Furthermore, we measured the maximum memory usage and the execution time of each tool with GNU time 1.7.

The complete performance results for all tools and datasets are shown in Figure 2.

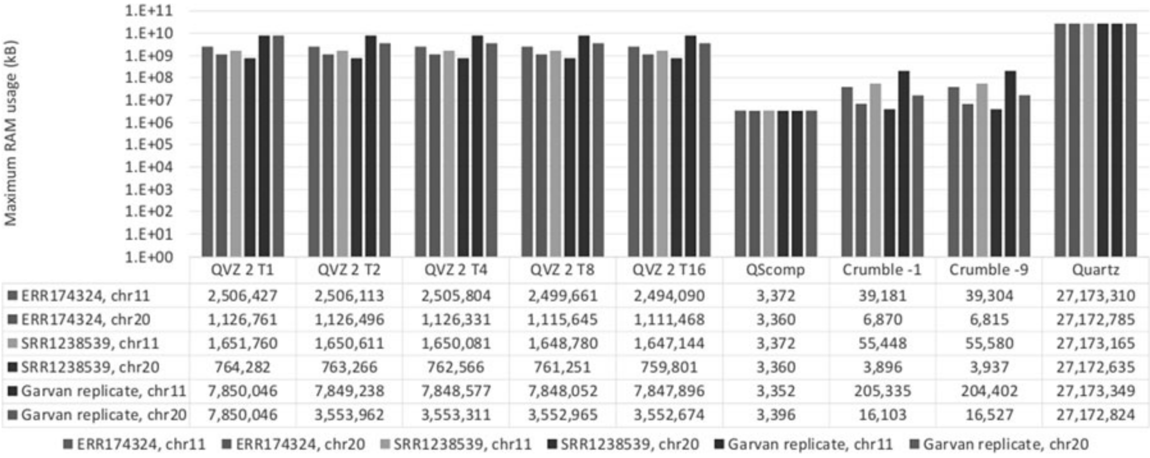


H01 (ERR174324)									
	Chromosome 11				Chromosome 20				Platform
	RAM usage (kB)	Time (s)			RAM usage (kB)	Time (s)			
	Max	User	System	Total	Max	User	System	Total	
QVZ 2 T1	2,506,427	272	6	278	1,126,761	126	1	127	Intel Xeon E5-2680 v3 CPU (2.50 GHz);270 GB RAM
QVZ 2 T2	2,506,113	237	3	240	1,126,496	110	1	111	
QVZ 2 T4	2,505,804	223	6	229	1,126,331	96	1	97	
QVZ 2 T8	2,499,661	186	2	188	1,115,645	89	1	90	
QVZ 2 T16	2,494,090	183	2	185	1,111,468	83	1	84	
QScomp	3,372	131	5	136	3,360	57	2	59	
Crumble -1	39,181	976	8	984	6,870	359	3	362	
Crumble -9	39,304	697	4	701	6,815	289	3	292	
Quartz	27,173,310	1,118	261	1,379	27,172,785	456	238	694	

H11 (SRR1238539)									
	Chromosome 11				Chromosome 20				Platform
	RAM usage (kB)	Time (s)			RAM usage (kB)	Time (s)			
	Max	User	System	Total	Max	User	System	Total	
QVZ 2 T1	1,651,760	312	3	315	764,282	254	2	256	Intel Xeon E5-2680 v3 CPU (2.50 GHz); 270 GB RAM
QVZ 2 T2	1,650,611	303	4	307	763,266	239	2	241	
QVZ 2 T4	1,650,081	287	3	290	762,566	251	3	254	
QVZ 2 T8	1,648,780	285	3	288	761,251	117	1	118	
QVZ 2 T16	1,647,144	280	4	284	759,801	170	2	172	
QScomp	3,372	131	4	135	3,360	57	2	59	
Crumble -1	55,448	3,078	9	3,087	3,896	1,282	3	1,285	
Crumble -9	55,580	916	5	921	3,937	414	2	416	
Quartz	27,173,165	618	244	862	27,172,635	260	190	450	

H12 (Garvan replicate)									
	Chromosome 11				Chromosome 20				Platform
	RAM usage (kB)	Time (s)			RAM usage (kB)	Time (s)			
	Max	User	System	Total	Max	User	System	Total	
QVZ 2 T1	7,850,046	1,017	10	1,027	7,850,046	1,017	10	1,027	Intel Xeon E5-2680 v3 CPU (2.50 GHz); 270 GB RAM
QVZ 2 T2	7,849,238	946	17	963	3,553,962	420	4	424	
QVZ 2 T4	7,848,577	766	14	780	3,553,311	367	4	371	
QVZ 2 T8	7,848,052	812	16	828	3,552,965	382	8	390	
QVZ 2 T16	7,847,896	736	17	753	3,552,674	362	4	366	
QScomp	3,352	436	14	450	3,396	205	7	212	
Crumble -1	205,335	1,998	14	2,012	16,103	911	8	919	
Crumble -9	204,402	1,897	14	1,911	16,527	825	6	831	
Quartz	27,173,349	2,943	387	3,330	27,172,824	1,346	285	1,631	

FIG. 2. Performance measurements results.



	QVZ 2 T1	QVZ 2 T2	QVZ 2 T4	QVZ 2 T8	QVZ 2 T16	QScomp	Crumble -1	Crumble -9	Quartz
■ ERR174324, chr11	2,506,427	2,506,113	2,505,804	2,499,661	2,494,090	3,372	39,181	39,304	27,173,310
■ ERR174324, chr20	1,126,761	1,126,496	1,126,331	1,115,645	1,111,468	3,360	6,870	6,815	27,172,785
■ SRR1238539, chr11	1,651,760	1,650,611	1,650,081	1,648,780	1,647,144	3,372	55,448	55,580	27,173,165
■ SRR1238539, chr20	764,282	763,266	762,566	761,251	759,801	3,360	3,896	3,937	27,172,635
■ Garvan replicate, chr11	7,850,046	7,849,238	7,848,577	7,848,052	7,847,896	3,352	205,335	204,402	27,173,349
■ Garvan replicate, chr20	7,850,046	3,553,962	3,553,311	3,552,965	3,552,674	3,396	16,103	16,527	27,172,824

FIG. 3. Maximum RAM usage results.

The maximum RAM usage results for all tools and datasets are shown in Figure 3. Note that we applied a logarithmic scaling to the *Y*-axis.

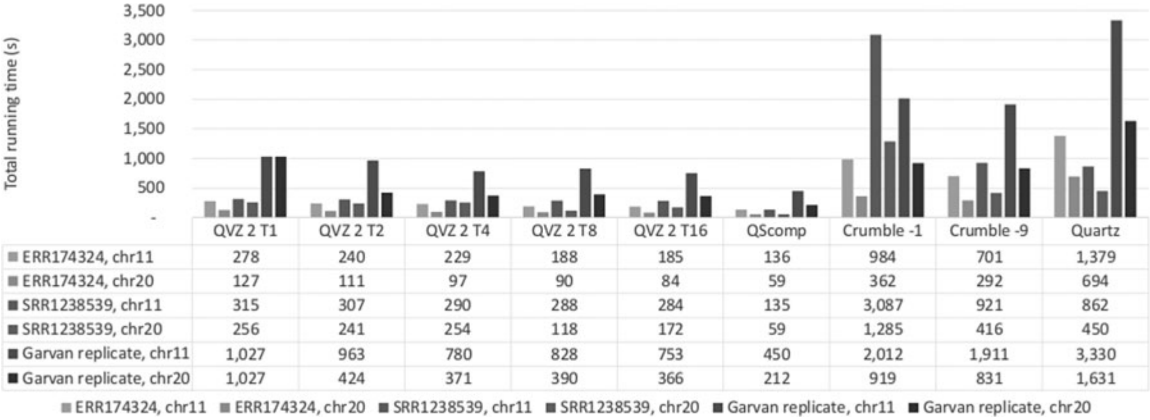
The running times for all tools and datasets are shown in Figure 4.

QScomp exhibits the least RAM usage of all tools, with 3.4 MB on average, due to its low algorithmic complexity. The running times of QScomp are comparable to that of the different runs of QVZ 2 and even two orders of magnitude lower than that of Quartz.

4.2. Variant calling results

In this section, we show the results of variant calling with the GATK + VQSR pipeline. For further results obtained from running the other two pipelines, we refer the reader to the Supplementary Data. For the first set of simulations we used the paired-end run ERR174324 of the NA12878 individual. This run was sequenced by Illumina on an Illumina HiSeq 2000 system as part of their Platinum Genomes project. The coverage of this data set is 14 ×. Due to the size of data and following the approach of Ochoa et al., 2016, we consider chromosomes 11 and 20. Furthermore, we averaged the Recall and Precision metrics over the two chromosomes (11 and 20) and the four VQSR filter values ($\theta \in \{90, 99, 99.9, 100\}$), which yield two plots. In what follows, we did the same for the other data sets. Thus, we present in total six plots (i.e., 3 data sets×2 metrics) in this section.

We can observe from Figure 5 that QScomp compresses the quality scores down to 0.16 bits per quality score while the Precision is retained. However, we also observe a slight drop in Recall, compared with the results for the uncompressed data.



	QVZ 2 T1	QVZ 2 T2	QVZ 2 T4	QVZ 2 T8	QVZ 2 T16	QScomp	Crumble -1	Crumble -9	Quartz
■ ERR174324, chr11	278	240	229	188	185	136	984	701	1,379
■ ERR174324, chr20	127	111	97	90	84	59	362	292	694
■ SRR1238539, chr11	315	307	290	288	284	135	3,087	921	862
■ SRR1238539, chr20	256	241	254	118	172	59	1,285	416	450
■ Garvan replicate, chr11	1,027	963	780	828	753	450	2,012	1,911	3,330
■ Garvan replicate, chr20	1,027	424	371	390	366	212	919	831	1,631

FIG. 4. Total running time results.

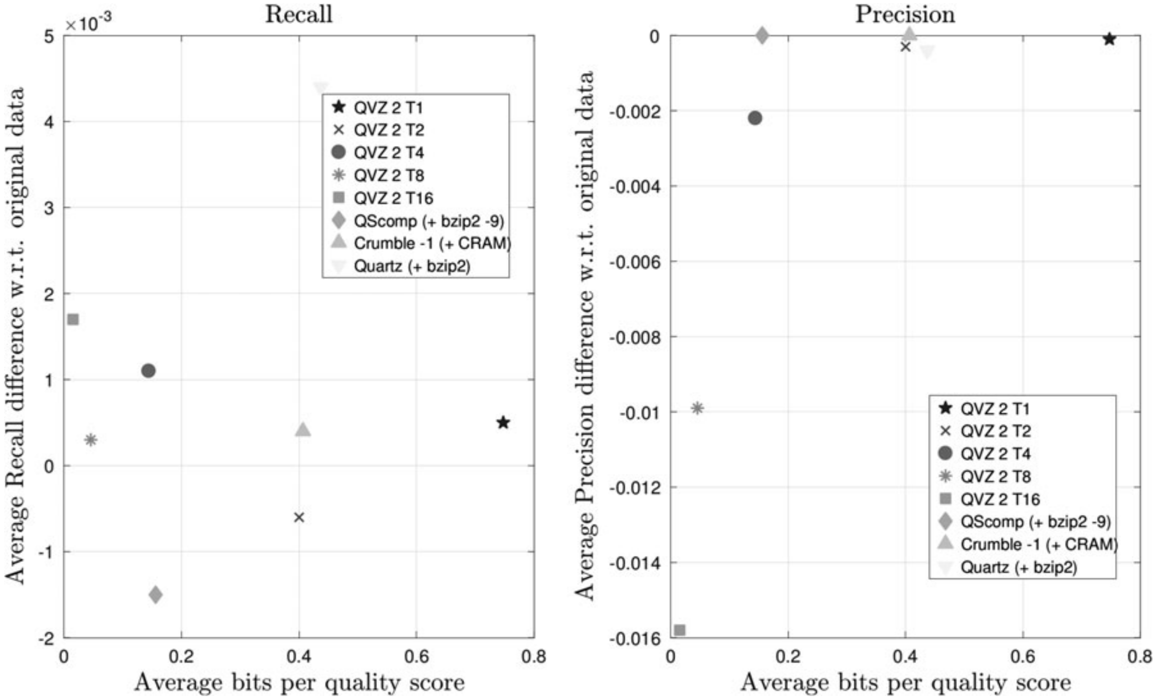


FIG. 5. Recall and Precision results averaged over both chromosomes (11 and 20) and all four VQSR filter values for the Illumina HiSeq 2000 data set (ERR174324) with a coverage of 14×. VQSR, Vector Quality Score Recalibration.

Next, we show the results for the SRR1238539 run on the NA12878 individual for which an Ion Torrent sequencing machine was used. The coverage of this data set is 10×. Again, chromosomes 11 and 20 were considered due to the size of the data. Moreover, the results shown are also the results of averaging over the same four filter values and both chromosomes. Figure 6 shows that QScomp is the worst performer in terms of both Recall and Precision. Since all other tools exhibit a similar performance, we must conclude that the

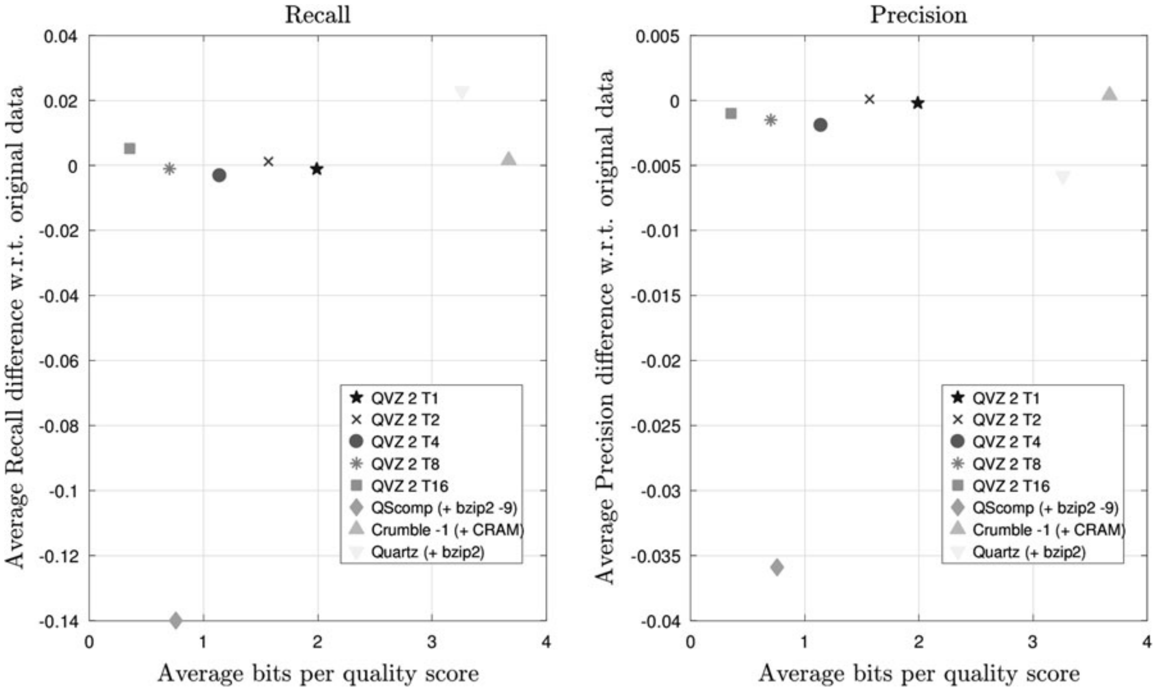


FIG. 6. Recall and Precision results averaged over both chromosomes (11 and 20) and all four VQSR filter values for the Ion Torrent data set (SRR1238539) with a coverage of 10×.

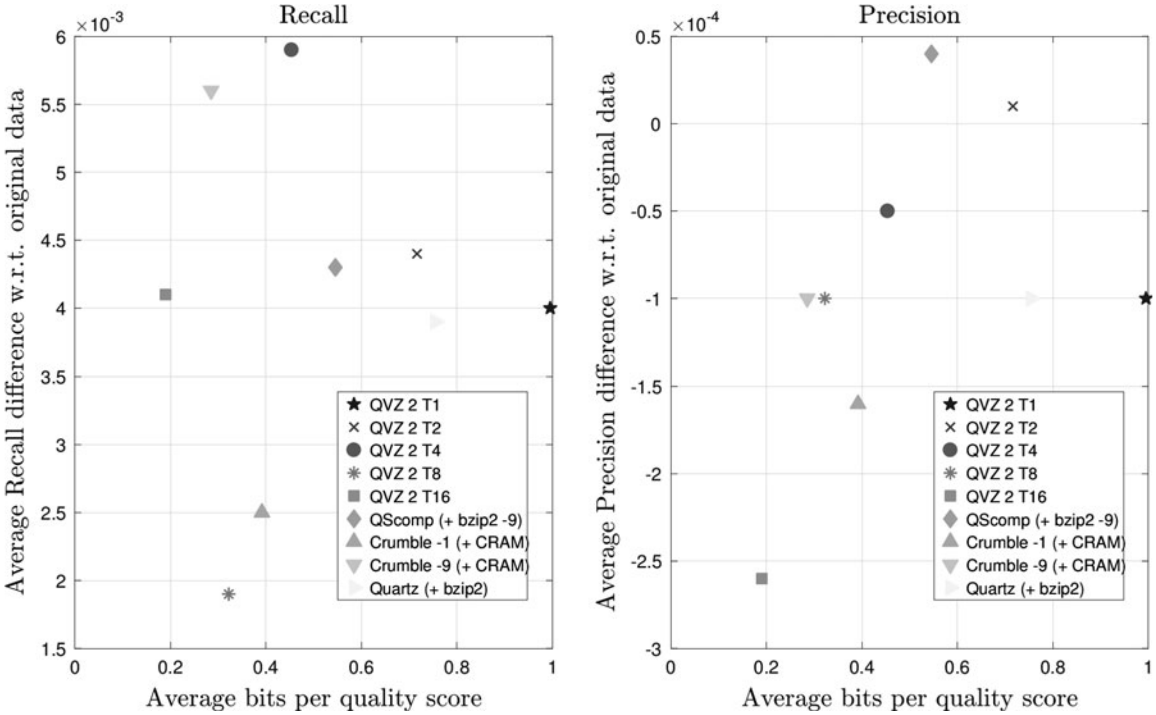


FIG. 7. Recall and Precision results averaged over both chromosomes (11 and 20) and all four VQSR filter values for the Illumina HiSeq X data set (Garvan replicate) with a coverage of 49×.

assumptions used for the construction of the binning scheme implemented in QScomp do not seem to hold for the quality score statistics produced by Ion Torrent sequencing machines.

Finally, we used the first replicate of the sample data set generated by the Garvan Institute from the Coriell Cell Repository NA12878 reference cell line. These data were sequenced on a single lane of an Illumina HiSeq X machine. The coverage of this data set is 49×. These results are shown in Figure 7. In terms of Recall and Precision, QScomp exhibits a similar performance as for the data set ERR174323, which is shown in Figure 5. Again, the Precision is retained. However, for this data set, a better Recall can be observed for all tools, including QScomp. Due to the high coverage of this data set, the competing tools are able to spend less bits per quality score than QScomp. Nevertheless, QScomp compresses the quality scores down to 0.55 bits per quality score, yielding a compression factor of 5.9 with respect to the entropy of the uncompressed data.

5. CONCLUSIONS

We presented a hierarchical quality score compression scheme, which represents the quality scores in two levels. The first level maps each quality score to its nearest square integer, and the second level encodes the distance of the original quality score to its mapped value. The impact of the lossy representation of quality scores on downstream analyses was investigated using three different variant calling pipelines. For data produced by Illumina sequencing machines, the downstream analysis results are competitive to the results obtained with competing lossy quality score compressors. Here, the Precision is retained, while a slight drop in Recall was observed. When this lossy level is accompanied by the second level, we observe that the compression ratio is around the entropy of the original quality scores. This shows that the suggested method to represent each quality score by a tuple does not have a negative effect on the lossless compression ratio performance.

What is more, we showed that the proposed separate compression of multiple second-level streams is superior to the compression of the second level as a single stream. Hence, the incorporation of other quantization strategies from previous works into the proposed two-level scheme might be a reasonable future research avenue. Besides the compression ratios, the memory consumption and the running times are also important parameters. In this study, with an average of only approximately 3.4 MB, QScomp shows a significant reduction in peak memory usage, and achieved the highest speed in the benchmark.

Previous studies on quality score compression proposed solutions that are either lossless or lossy. Thus, if a user prefers lossy compression, the possibility to extract the original quality scores disappears, and in the reverse case, the user loses the capability to work with lossy quality scores to reduce the necessary computing resources. The QScomp scheme introduced in this study is unique in terms of providing lossless and lossy compression in a single framework by utilizing a hierarchical two-level representation.

In daily practice, we suggest to replace the quality scores in FASTQ files with the proposed first-level values, and to perform initial explorations with this lightweight presentation. The second-level values could for example be stored in an archive, and when deeper investigations are required, the original quality scores could be retrieved.

ACKNOWLEDGMENT

This work has been partially supported by The Scientific and Technological Research Council of Turkey, grant number TÜBİTAK—114E293.

AUTHOR DISCLOSURE STATEMENT

The authors declare that no competing financial interests exist.

REFERENCES

Alberti, C., Daniels, N., Hernaez, M., et al. 2016. An evaluation framework for lossy compression of genome sequencing quality values. 2016 Data Compression Conference (DCC), Snowbird, UT, pp. 221–230.

Berger, B., Daniels, N.M., and Yu, Y.W. 2016. Computational biology in the 21st century: Scaling with compressive algorithms. *Commun. ACM.* 59, 72–80.

Bonfield, J.K., and Mahoney, M.V. 2013. Compression of FASTQ and SAM format sequencing data. *PLoS One.* 8, e59190.

Cánovas, R., Moffat, A., and Turpin, A. 2014. Lossy compression of quality scores in genomic data. *Bioinformatics.* 30, 2130–2136.

Cock, P.J.A., Fields, C.J., Goto, N., et al. 2010. The sanger FASTQ file format for sequences with quality scores, and the solexa/illumina FASTQ variants. *Nucleic Acids Res.* 38, 1767–1771.

Deorowicz, S., and Grabowski, S. 2011. Compression of DNA sequence reads in FASTQ format. *Bioinformatics.* 27, 860–862.

Elias, P. 1975. Universal codeword sets and representations of the integers. *IEEE Trans. Inf. Theory.* 21, 194–203.

Ewing, B., and Green, P. 1998. Base-calling of automated sequencer traces using phred. II. Error probabilities. *Genome Res.* 8, 186–194.

Hach, F., Numanagić, I., Alkan, C., et al. 2012. Scalce: Boosting sequence compression algorithms using locally consistent encoding. *Bioinformatics.* 28, 3051–3057.

Hernaez, M., Ochoa, I., and Weissman, T. 2016. A cluster-based approach to compression of quality scores. 2016 Data Compression Conference (DCC), Snowbird, UT, pp. 261–270.

Janin, L., Rosone, G., and Cox, A.J. 2014. Adaptive reference-free compression of sequence quality scores. *Bioinformatics.* 30, 24–30.

Jones, D.C., Ruzzo, W.L., Peng, X., et al. 2012. Compression of next-generation sequencing reads aided by highly efficient de novo assembly. *Nucleic Acids Res.* 40, e171.

Loh, P.-R., Baym, M., and Berger, B. 2012. Compressive genomics. *Nat. Biotechnol.* 30, 627–630.

Malysa, G., Mikel, H., Ochoa, I., et al. 2015. Qvz: Lossy compression of quality values. *Bioinformatics.* 31, 3122–3129.

Numanagić, I., Bonfield, J.K., Hach, F., et al. 2016. Comparison of high-throughput sequencing data compression tools. *Nat. Methods.* 13, 1005–1008.

Ochoa, I., Asnani, H., Bharadia, D., et al. 2013. Qualcomp: A new lossy compressor for quality scores based on rate distortion theory. *BMC Bioinformatics.* 14, 187.

Ochoa, I., Hernaez, M., Goldfeder, R., et al. 2016. Effect of lossy compression of quality scores on variant calling. *Brief. Bioinform.* 18, 183–194.

Ostermann, J., Bormans, J., List, P., et al. 2004. Video coding with H.264/AVC: Tools, performance, and complexity. *IEEE Circuits Syst. Mag.* 4, 7–28.

Rimmer, A., Phan, H., Mathieson, I., et al. 2014. Integrating mapping-, assembly- and haplotype-based approaches for calling variants in clinical sequencing applications. *Nat. Genet.* 46, 912–918.

Van der Auwera, G.A., Carneiro, M.O., Hartl, C., et al. 2013. From FASTQ data to high-confidence variant calls: The genome analysis toolkit best practices pipeline. *Curr. Protoc. Bioinformatics.* 43, 11.10.1–11.10.33.

Voges, J., Ostermann, J., and Hernaez, M. 2017. CALQ: Compression of quality values of aligned sequencing data. *Bioinformatics.* 34, 1650–1658.

Wan, R., Anh, V.N., and Asai, K. 2012. Transformations for the compression of FASTQ quality scores of next-generation sequencing data. *Bioinformatics*. 28, 628–635.

Yu, Y.W., Yorukoglu, D., Peng, J., et al. 2015. Quality score compression improves genotyping accuracy. *Nat. Biotechnol*. 33, 240–243.

Zook, J.M., Catoe, D., McDaniel, J., et al. 2016. Extensive sequencing of seven human genomes to characterize benchmark reference materials. *Sci. Data.* 3, 160025.

Address correspondence to:
Jan Voges
Leibniz Universität Hannover
Institut für Informationsverarbeitung
Appelstr. 9A
Hannover 30167
Germany

E-mail: voges@tnt.uni-hannover.de

Assoc. Prof. Muhammed Oğuzhan Külekci
Informatics Institute
Istanbul Technical University
Istanbul 34469
Turkey

E-mail: kulekci@itu.edu.tr