Lossy Compression of Quality Scores in Genomic Data

Rodrigo Cánovas, Alistair Moffat, and Andrew Turpin
NICTA Victoria Research Laboratory,
Department of Computing and Information Systems,
The University of Melbourne, Victoria 3010

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Motivation: Next-generation sequencing technologies are revolutionizing medicine. Data from sequencing technologies is typically represented as a string of bases, an associated sequence of per-base quality scores, and other meta-data; and in aggregate can require a very large amount of space. The quality scores show how accurate the bases are with respect to the sequencing process, that is, how confident the sequencer is of having called them correctly, and are the largest component in data sets in which they are retained. Previous research has examined how to store sequences of bases effectively; here we add to that knowledge by examining methods for compressing quality scores. The quality values originate in a continuous domain, and so if a fidelity criterion is introduced, it is possible to introduce flexibility in the way these values are represented, allowing lossy compression over the quality score data.

Results: We present existing compression options for quality score data, and then introduce two new lossy techniques. Experiments measuring the trade-off between compression ratio and information loss are reported, including quantifying the effect of lossy representations on a downstream application that carries out Single Nucleotide Polymorphism and insert/deletion detection. The new methods are demonstrably superior to other techniques when assessed against the spectrum of possible tradeoffs between storage required and fidelity of representation.

Availability: An implementation of the methods described here is available at `https://github.com/rcanovas/libCSAM`.

Contact: rcanovas@student.unimelb.edu.au

Supplementary Information: Experimental results on a second file are reported.

1 INTRODUCTION

Genome sequencing methods have evolved at the same astonishing rate as computing technology. Next-generation devices parallelize the process, producing billions of sequences (*reads*) (Ansorge, 2009; Church, 2006; Mardis, 2008; Myllykangas *et al.*, 2012), and generating file sizes potentially in the terabyte range, at costs that are decreasing on a yearly basis. Each read is a fragment of data extracted from the processing of a single genome, represented as a string of *bases*. As well, a number of meta-data fields are associated with each read. Some of these fields are more expensive to store than the sequence of bases.

The mechanics of effectively storing and querying this information are fundamental to the field of bioinformatics (Giancarlo *et al.*, 2009). Several standard formats to store genome data have been adopted, each aiming to be easy to manipulate and parse using text-processing tools such as Perl and Python. The most common are the FASTA and FASTQ formats (Cock *et al.*, 2010); and the SAM/BAM formats (Li *et al.*, 2009).

Table 1. Example quality scores using *Phred* $+ 33$.

Letter	Quality value	Estimated error probability
(	40	20%
7	55	0.6%
F	70	0.02%
U	85	0.0006%
d	100	0.00002%

1.1 Quality Scores

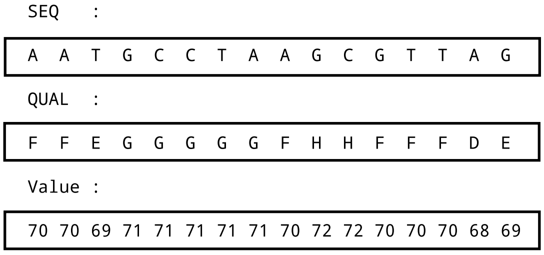
We consider the problem of effectively representing the quality scores of a genome sequence. Quality scores are typically stored as a *Phred-scaled* base error probability (Ewing and Green, 1998), calculated as $\lfloor -10 \cdot \log_{10} Pr\{\textit{base is wrong}\} \rfloor$. These are then typically scaled (*Phred* $+ 33$ or *Phred* $+ 64$) so that they can be represented as printable ASCII characters. Table 1 shows examples for *Phred* $+ 33$, the scheme assumed in examples in this paper.

The probability that a base in a given position is wrong can be computed in different ways, and depends on the hardware and software that are used in the sequencer. Figure 1 shows an example of a short sequence of bases together with the corresponding quantized quality assessments. The higher the quality value, the lower the probability that the base at that position has been incorrectly called. For example, in Figure 1, the first nucleotide of the sequence has a quality value equal to 70 (the symbol “F” in the ASCII alphabet), which means that there is an estimated probability of approximately $10^{-(70-33)/10} \approx 0.02\%$ that the corresponding “A” has been incorrectly sequenced.

1.2 Compression

We consider two different compression modalities: *lossless* and *lossy*. Each has advantages and disadvantages, depending on the importance of the information that is stored, and on the eventual use of the data.

Lossless compression is the most common approach. It ensures that the decompressed data is exactly the same as the original,



```
SEQ   :
A A T G C C T A A G C G T T A G
QUAL  :
F F E G G G G G F H H F F F D E
Value :
70 70 69 71 71 71 71 71 70 72 72 70 70 70 68 69
```

Fig. 1: Example of a sequence and quality components. The value of each quality score is also shown.

meaning that the compressed version must contain sufficient information for the decompresser to generate a byte stream identical to the one that was input to the compressor. *Lossy* compression mechanisms store an approximate representation of the input, trading loss in fidelity of reproduction for enhanced compression effectiveness. Lossy compression is typically only applied to signals that were originally continuous, for which the digital form of the signal already incorporates an element of quantization-generated approximation, such as image and sound data.

If lossy compression is being considered, the maximum degree of information loss that can be tolerated needs to be specified as an input to the compressor. That requirement presupposes that a measure of loss has been defined, taking as its inputs two data sequences of the type being represented, and providing (using some meaningful unit) a numeric score that summarizes the degree to which they differ. Table 2 shows some of the measures that can be applied to pairs of ℓ-element strictly-positive sequences (Cha, 2007). In all of them, if the two sequences are identical, the score is low (zero, or, in the case of Max:Min Distance, one); and when the two sequences are different, the measure is higher. For example, for the two $\ell = 6$-element sequences $X = \langle 70, 69, 70, 71, 69, 73\rangle$ and $Y = \langle 70, 70, 70, 70, 70, 73\rangle$, the Mean Manhattan Distance is 0.5. The measures shown in Table 2 will be used in Sections 3 and 4 to evaluate, and in some cases control, the accuracy of lossy compression regimes for quality score sequences.

Table 2. Examples of fidelity measures between strictly positive numeric vectors X and Y of length ℓ (adapted from Cha (2007)). Scores of zero indicate that X and Y are identical except for the Max:Min measure, where one is the minimum value.

Measure	Function
Mean Manhattan Distance(X, Y)	$\frac{1}{\ell}\sum_{i=1}^{\ell} \lvert X_i - Y_i \rvert$
Max:Min Distance(X, Y)	$\max_{1\le i\le \ell} \frac{\max(X_i, Y_i)}{\min(X_i, Y_i)}$
Mean Squared Error(X, Y)	$\frac{1}{\ell}\sum_{i=1}^{\ell} (X_i - Y_i)^2$
Chebyshev Distance(X, Y)	$\max_{1\le i\le \ell} \lvert X_i - Y_i \rvert$
Soergel Distance(X, Y)	$\frac{\sum_{i=1}^{\ell} \lvert X_i - Y_i \rvert}{\sum_{i=1}^{\ell} \max(X_i, Y_i)}$
Lorentzian Distance(X, Y)	$\sum_{i=1}^{\ell} \log_2(1 + \lvert X_i - Y_i \rvert)$

Table 3. Examples of Binary, Unary, Gamma, and Golomb codes for the integers 0 to 6, assuming that values $x \ge 0$ are to be coded. The Binary code shown here uses 3 bits per integer (and hence has an upper limit of $x = 7$); the Golomb code is constructed using $b = 3$.

Integer	Binary	Unary	Gamma	Golomb
0	000	0	0	0 0
1	001	10	10 0	0 10
2	010	110	10 1	0 11
3	011	1110	110 00	10 0
4	100	11110	110 01	10 10
5	101	111110	110 10	10 11
6	110	1111110	110 11	110 0

1.3 Bit Encodings

Once the set of symbols to be coded has been determined, the compressor *encodes* each symbol into a bit sequence. A fixed-length binary code or variable-length prefix-free code is used to represent the symbols contained in the data, with the choice dependent on the probability distribution that arises over the set of symbols.

The simplest variable-length code is Unary, which represents a non-negative number x as x 1-bits, followed by a 0-bit. Unary has the advantage of being able to represent arbitrarily large values, but requires as many bits as the value being coded. The Elias Gamma code is more complex, but also more effective, with a value x represented as the concatenation of the length of the binary representation of x in Unary, and the binary representation of $x + 1$, omitting its most significant bit (Elias, 1975). The Golomb code also consists of a Unary/Binary combination, but makes use of a parameter b, with the unary component storing the quotient $q = \lfloor x/b \rfloor$, and a minimal-length binary component storing the remainder $r = x - b \cdot q$. The parameter b is chosen according to the characteristics of the set of x values to be stored. Table 3 shows examples of these codes, covering the first 6 integers. Note that the 3-bit Binary code is limited to inputs $0 \le x < 8$, and assumes that all of the symbols have equal probability; the Unary, Gamma, and Golomb codes assume that the symbol probabilities are non-increasing, and place no upper limit on the values x that can be represented. If the symbol probabilities are not non-increasing as the distance from the origin increases, a permutation table can be used, so that monotonicity can be maintained. Moffat and Turpin (2002) describe all of these mechanisms in detail.

1.4 Subsequent Applications

One use of sequence data is to identify locations of variation from a specified reference sequence. These variations can assist with the determination of the genotype for each individual at each site, allowing, for example, the study of diseases (Nielsen *et al.*, 2011). The process of finding variations is know as *Single Nucleotide Polymorphism* (*SNP*) and *insert/deletion* (*indel*) detection. The normal output of this process is a set of SNPs and indels, stored in a standard format such as *Variant Call Format*

(VCF) (Danecek *et al.*, 2011). Each line in a VCF file represents a different variation, storing its position and allele in the reference genome, plus the bases related to the variation, a computed quality score, and extra information about the variation. We use comparisons between generated VCF files as an applied evaluation of the effect that lossy compression of quality scores has on sequence data use.

2 PREVIOUS APPROACHES

The SLIMGENE tool presented by Kozanitis *et al.* (2010) was one of the first methods for independent compression of quality scores. Kozanitis *et al.* give two compression methods, both based on the observation that neighboring quality values are, in general, close to each other. Their first method encodes the differences between consecutive quality values using a Huffman code; while the second approach inserts a Markov model prior to the Huffman code. Deorowicz and Grabowski (2011) observed that the quality sequences were quasi-random with mild dependence on score position; that some sequences finish with several "#" characters; and also that there can be strong local correlations within individual records. In their proposal a bit flag is stored per sequence to indicate whether or not the sequence is prefixed by a string of hashes. The hash prefix is removed and the rest of the sequence is stored using a Huffman code. They also proposed storing a modified version of the quality sequence, where equal consecutive values are stored runs. For example, the sequence "FFEGGGGGFHHFFFDE" shown in Figure 1 would be transformed to "FEGFHFDE", plus a list $1, 0, 4, 0, 1, 2, 0, 0$ of run-lengths. The new sequence and the vector of run-lengths are then stored using Huffman codes.

Wan *et al.* (2012) examine several compression methods for compressing the quality field, and propose a new mechanism in which quality scores are transformed into a bin number, with each bin representing an interval. They present three lossy transformations: UniBinning, Truncating, and LogBinning. The UniBinning approach creates uniform splits across the error probability distribution; that is, if the parameter is four (and two-bit Binary codes will be used), the first bin spans all quality values corresponding to error probabilities between 0 percent and 25 percent. The Truncating transformation separates all the quality values independently, creating a special bin for the *l* largest values, with *l* a user define parameter. The LogBinning transformation is similar to UniBinning, but with the bins formed according to the logarithms of the probabilities, by constructing equal-sized segments according to quality values.

After applying a binning transformation, Wan *et al.* form groups of k mapped quality values into a block, and prefix the block with a header containing some local information, depending on the lossless method used. Their MinShifting transformation stores the minimum quality value $q_{\min}$ of each block, and the k values q of the block are each replaced by $q - q_{\min}$ before being coded. The FreqOrdering transformation permutes all the quality values in each block according to their frequency, so that the most commonly-occurring values are assigned the shortest codes when Gamma or Golomb are employed. The third option considered by Wan *et al.* is GapTranslating, in which all the values of a block are changed to the difference-values presented by Kozanitis *et al.* Finally, after a lossy transformation and then a lossless mapping, Wan *et al.* employ one of the coding regimes summarized in Section 1.3.

Ochoa *et al.* (2013) describe the QualComp tool, which compresses quality scores based on the assumption that they are drawn from a Gaussian distribution (Lapidoth, 1997) parameterized by the position of the quality value in the quality read, assuming that there is a mean and standard deviation for each position $i \in [1, \ell]$ in the read. The mean and covariance matrix of the quality scores are extracted from the entire input, and singular value decomposition (SVD) is applied to the covariance matrix to recover standard deviations to be used as parameters of the Gaussian. Once those parameters are decided, they are stored for use by the decoder, and employed to derive the number of bits used to encode each quality value, by minimizing the mean square error (MSE) of a rate distortion problem. This optimization problem is parameterized by a user input which is the total number of output bits the quality scores can consume, expressed as a rate r. The solution to the problem specifies, for position i in a read, the number of bits, ρ_i, that should be used to encode the quality value in position i. For each quality value, if it is to be coded in ρ_i bits, the Gaussian for that position is split into 2^{ρ_i} regions of equal probability, and a bit pattern selected to indicate the region in which the quality value lies. The decoder reads in the Gaussian parameters and r, performs the optimization to recover the ρ_i values, and then emits a representative corresponding to the region specified by each codeword.

Other approaches compress the quality scores depending on information of the read sequences. Tembe *et al.* (2010) describe a simple lossless compression technique over the sequence and the quality fields, combining all distinct pairs of genomic base and respective quality value, and then using a Huffman code. That is, the DNA sequence and quality scores of an alignment are represented by a joint sequence of codewords, one per each $\langle \textit{base}, \textit{quality} \rangle$ pair. Janin *et al.* (2013) present a lossy compression scheme based on the premise that, if a base in a read can be predicted by the context of the read, then the base and the respective quality need not be stored. To compute predictions they use the Burrows-Wheeler transform (Burrows and Wheeler, 1994) and a longest common prefix array, together with a user-defined minimum context length.

We also consider CRAM format files (Fritz *et al.*, 2011). These files are generated using CRAMTOOLS, a suite of Java programs and APIs that compress DNA sequence and quality data stored in SAM/BAM format, making use of an external reference genome. Sequences are stored as a difference relative to the external reference, and hence the external reference genome must be provided each time compression or decompression is undertaken. Several lossy options for storing quality scores are provided in the CRAMTOOLS suite. We explored three combinations: Preserve, Bin-Preserve, and Match-Bin-Preserve. The Preserve mode only stores quality scores for reads whose mapping quality scores are higher than a user defined threshold. The other two approaches are variations of Preserve, in which the prefix Bin means that the quality scores are stored using 8 bins like Wan *et al.*'s LogBinning; and prefix Match indicates that only quality scores of bases that match with the reference are retained.

Algorithm 1 : Inputs *quality*$[0 \ldots \ell - 1]$ and a threshold p

```
Representatives ← [ ]
RunLengths ← [ ]
block ← [ ]
block_len ← 0
for $i = 0 \to \ell - 1$ do
    if FulfillCriteria(block ⊕ quality[i], p) then
        block ← block ⊕ quality[i]
        block_len ← block_len + 1
    else
        rep ← CalculateRepresentative(block)
        Representatives ← Representatives ⊕ rep
        RunLengths ← RunLengths ⊕ block_len
        block ← quality[i]
        block_len ← 1
    end if
end for
Representatives ← Representatives ⊕ rep
RunLengths ← RunLengths ⊕ block_len
return Representatives and RunLengths
```

3 ALGORITHM

We now present a lossy compression approach based on localized properties of the quality scores, following the lead established by Kozanitis *et al.* (2010). The new algorithms represent quality scores by separating them into blocks of variable size, where all the values contained in each block comply with a chosen parameter p according to some measure criterion. For each block we store its length and a *representative value*. The representative value also depends on which measure is used. For example, if Mean Manhattan Distance (see Table 2) is used as the fidelity measure, blocks must be formed such that none of the values in each block are more than p from the corresponding representative value. That is, one possible configuration allows blocks to span a range of up to $2p+1$ different quality scores, with the mid-point of each block's range taken as the representative value. Generalizing from this starting point, the use of other measures, or other values of p, results in different blocks being formed, and hence different amounts of space being required for the set of quality scores, with different degrees of approximation to the original values.

Algorithm 1 describes the proposed process, with "[ ]" used to represent the empty list, and with the " $\oplus$ " operator used to append a single value to a list of the same type of object. To determine the block boundaries, a left-to-right greedy search is undertaken, seeking to make each block as long as possible, consistent with the constraints imposed by the chosen fidelity criterion and the chosen fidelity limit p. Method *FulfillCriteria* determines if it is possible to include the next quality value in the current block, or whether it is necessary to start a new block; and method *CalculateRepresentative* computes the best representative value for a given block, given that at least one valid representative exists. For example, if the measurement metric being used is Mean Manhattan Distance, then *FulfillCriteria*$(qvals, p)$ is implemented by computing

$$\left(\max_i \textit{qvals}[i]\right) - \left(\min_i \textit{qvals}[i]\right) \leq 2p\,,$$

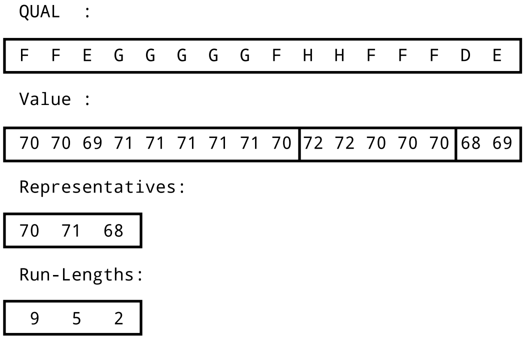


Fig. 2: Quality values being formed into blocks consistent with a Mean Manhattan Distance guarantee (the approach used in the P-Block transformation) when $p = 1$.

where the range of i is $0 \leq i < |\textit{qvals}|$. Within a set that meets this criterion, the value

$$\textit{rep} = \frac{(\max_i \textit{qvals}[i]) + (\min_i \textit{qvals}[i])}{2}$$

guarantees that the Mean Manhattan Distance for the whole sequence will be not greater than p. For any given block there may also be other integer values *rep*$'$ for which

$$\max_i |\textit{qvals}[i] - \textit{rep}'| \leq p\,,$$

and in that case, method *CalculateRepresentative* is free to use a secondary criterion, such as minimizing the average distance between the selected representative and the block's values.

We call this option – in which Mean Manhattan Distance is used in both *FulfillCriteria* and *CalculateRepresentative* – the P-Block mechanism, and control it with the parameter p. Figure 2 illustrates the application of the P-Block to the example sequence.

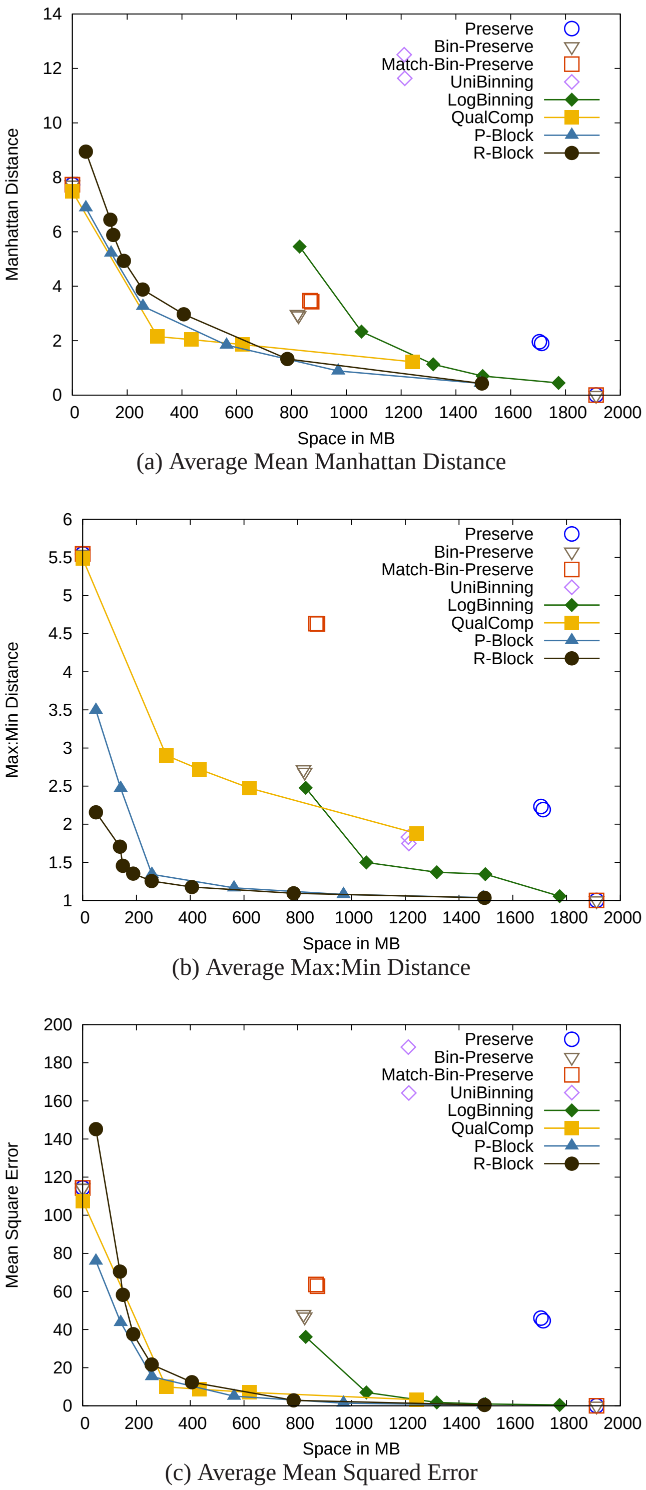
As a second example of how Algorithm 1 might be instantiated, if a given upper bound r must be guaranteed using the Max:Min Distance metric, then *FulfillCriteria*$(qvals, r)$ needs to check if an integral *rep* exists such that $(\max_i \textit{qvals}[i])/\textit{rep} < r$ and $\textit{rep}/(\min_i \textit{qvals}[i]) < r$. This R-Block approach, parameterized by a maximum ratio r, recognizes that it may be preferable to be more precise in representing low quality scores (when the base is assessed as being more likely to be in error) than high ones.

After either of these transforms has been applied to an alignment, the k elements in the *Representatives* array are stored as a minimum and maximum, followed by k values relative to that range, using a Binary code of $\lceil \log_2(\max_k \textit{Representatives}[k] - \min_k \textit{Representatives}[k] + 1) \rceil$ bits per value. The k values in the *RunLengths* array are stored using the Gamma code. When p or r are sufficiently large, only one value is stored for each read, covering every quality value in that alignment.

In related work we have examined compression techniques for the other components of SAM/BAM-format files (Cánovas and Moffat, 2013), including the bases; and added a small index that allows decoding to commence at every 500 th read. The space used to stored the index information is a small fraction of the total space required, and the functionality that it offers is not part of the experimentation that is reported here.

Table 4. Test sequences: quality scores were extracted from two SAM files, `NA12878.chr20.qual`, and `HG01477.chr11.qual`. The columns show the size of the sequence of extracted quality scores as ASCII bytes, the number of reads, the length of each read, and the entropy (H_0, in bits per quality score) of the quality sequence.

File	Size (MB)	Reads	Read length	H_0
`NA12878.chr20.qual`	5,017.98	51,585,658	101	3.384
`HG01477.chr11.qual`	617.22	6,407,925	100	3.937



(a) Average Mean Manhattan Distance

(b) Average Max:Min Distance

(c) Average Mean Squared Error

Fig. 3: Fidelity versus space trade-offs for the quality sequence `NA12878.chr20.qual`. In the case of P-Block and R-Block, low values of the parameter give points in the top left; as the parameter is increased, the distance measure decreases, and the space increases.

4 IMPLEMENTATION

We compare the quality scores of the original file against the quality scores in a reconstructed file using different approaches and a range of fidelity measures (see Table 2). The fidelity of a mechanism is computed by averaging all of the per-read scores generated for that measure. The quality scores associated with a version of the *NA12878* genome are the primary test file used. In particular, we work with reads that were associated with chromosome 20[1]. To further validate our results, we also take genome *HG01477* from the 1000 Genomes Project[2], using the reads that were aligned with chromosome 11; those results are presented in the Supplementary Material. Table 4 gives details of two files of sequence quality values. As part of the processing pipeline used in the experimentation, we also made use of reference chromosomes `chr11` and `chr20`[3].

The implementations explored include Wan *et al.*'s LogBinning and UniBinning; the QualComp mechanisms; and our P-Block and R-Block approaches. We also include the CRAMTOOLS lossy models, in recognition of their extensive use in data repositories such as the *1000 Genomes Project*. To establish a benchmark for lossless compression, results for GZip –9 are also included. Figure 3a shows the various tradeoffs measured when fidelity is assessed using Mean Manhattan Distance; Figure 3b shows the same tradeoff, but with the vertical scale changed to Max:Min Distance; and Figure 3c likewise, but with fidelity measured using Mean Squared Error.

Each of the implementations provides a tradeoff "knob" that adjusts the balance between fidelity and compression rate. The graphs are plotted using QualComp rates $r = \{0, 0.5, 1, 2\}$; P-Block parameters $p = \{1, 2, 4, 8, 16, 32\}$; R-Block ratios $r = 1 + \{0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4\}$; UniBinning thresholds $b = \{80, 100\}$; and LogBinning thresholds $b = \{5, 10, 20, 30, 40, 60\}$. For CRAM, lower bound mapping qualities of 40, 50, and 60 were used with the Preserve, Bin-Preserve and Match-Bin-Preserve modes.

The three components of Figure 3 show that the QualComp, P-Block, and R-Block approaches outperform the other methods. The new approaches are particularly-well suited to the Max:Min Distance, as was the intention; while Ochoa *et al.*'s QualComp

[1] That is, we extracted the quality score fields from the file `NA12878.Hi.WGS.bwa.cleaned.recal.hg19.20.sam` fetched from `https://usegalaxy.org/library_common/browse_library?show_deleted=False&cntrller=library&use_panels=False&id=f9ba60baa2e6ba6d`.

[2] `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/`.

[3] Accessed from `http://hgdownload.cse.ucsc.edu/goldenPath/hg19/chromosomes/`.

offers good performance when fidelity is quantified using Mean Squared Error. Worth noting is that QualComp offers a further modality, allowing the quality file to be separated into c clusters and QualComp applied to each cluster separately. The results obtained via this approach did not result in any appreciable difference in performance.

We also analyze the effect of lossy compression of quality scores on a downstream application. For each lossy file generated we compute its VCF (see Section 1.4) and compare it with the VCF generated from the original file. To compare two VCF files we use the methodology of Ochoa *et al.* (2013), and define a *true positive* (*TP*) as a variation that is found in both VCF files; a *true negative* as a location at which neither file records a variation; a *false negative* (*FN*) as a variation that is only found in the VCF of the lossless file; and a *false positive* (*FP*) as a variation that is found only in the VCF of the lossy file. The TP values are further separated into *half* and *equal* TP, where equality is registered if the same variation is found in both VCF files, and half is registered if a variation is found in the same position in both files, but the type of variation is not exactly the same. In all our experiments the amount of “half TP” was no higher than 0.004 per cent of the total TP value, and they were included in the FP count without affecting the results.

From these values the *precision* and *recall* of the processed sequence can be computed, given by $\mathit{Precision} = \mathit{TP}/(\mathit{TP} + \mathit{FP})$ and $\mathit{Recall} = \mathit{TP}/(\mathit{TP} + \mathit{FN})$ respectively; they in turn can be combined into a single statistic known as an *F-Score* by combining them as $\mathit{F\text{-}Score} = 2 \cdot \mathit{Precision} \cdot \mathit{Recall}/(\mathit{Precision} + \mathit{Recall})$. Table 5 provides precision and recall scores for a selected range of methods applied to `NA12878.chr20.qual`, together with the F-Scores computed from them; and Figure 4 plots F-Score as a function of compressed size, clearly illustrating the superiority of the new approaches. Similar results were obtained using the `HG01477.chr11.qual` file, included in the Supplementary Material.

As Figure 4 and Table 5 illustrate, our two approaches outperform other techniques: that is, for a given level of precision or recall, the P-Block and R-Block methods allow more compact storage of quality values than do other mechanisms. For example, if 99.0 per cent in both precision and recall is regarded as being a minimum threshold for having high-confidence in the outcomes, then of the methods compared, P-Block with $p = 4$ generates the most compact representation. The 563 MB required for the quality values in that configuration is less than one third of the space required by the lossless GZip approach. Figure 4 and Table 5 also include an additional point, labeled OneQual, showing what would happen if no quality scores at all were stored, and every quality value was assumed to be the average quality score from the input sequence.

Note that the different methods all have comparable encoding and decoding speeds, and we do not compare them on that basis.

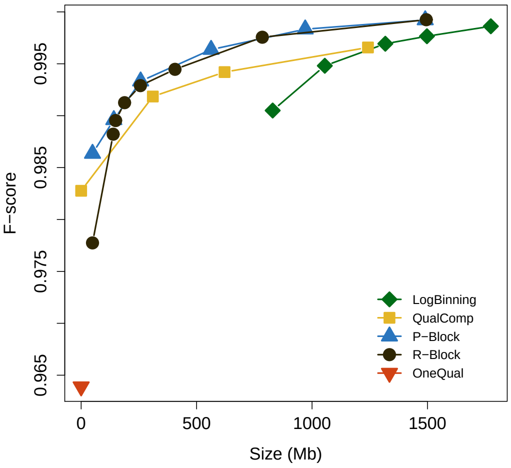


Fig. 4: Effect of lossy compression techniques on the quality of VCF computation for `NA12878.chr20.qual`. Each line represents a different lossy compression method; higher lines represent more desirable methods.

5 DISCUSSION

We have described and measured the performance of a range of lossy compression techniques for quality scores. The QualComp approach, and our two new methods, offer superior trade-off options when fidelity is assessed according to the measures in Table 2. Our new approaches outperformed QualComp in the Max:Min Distance criterion – a measure that is well-suited, we believe, to bioinformatics applications.

Our experiments also quantified the extent to which use of lossy compression affected the performance of a typical downstream application of genetic data, and demonstrate that variation detection can still be reliably carried out, even with relatively compact storage of the quality scores.

The approaches described here make use of sequentially-greedy generation of blocks. One interesting question that we have not yet examined is whether mechanisms for globally optimal block construction will make a measurable difference in overall outcomes. Another area that we have not yet fully explored is the coding mechanisms used for the block representatives and for the block lengths; it may be that tailored codes (rather than Binary and Gamma) can offer further space savings once the particular characteristics of the data streams are taken into account. We are also interested in quantifying the effect that lossy compression has on other downstream applications of genetic sequencing data.

ACKNOWLEDGMENT

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Center of Excellence program. This work was also funded by the Australian Research Council under the Future Fellowship scheme. We thank Idoia Ochoa-Alvarez for her assistance with QualComp; Vadim Zalunin for helping with the CRAMTOOLS usage; and Wei Shi and Jan Schröder for sharing their knowledge of the area. Finally, we thank the anonymous referees for their careful input.

Table 5. Recall and precision percentages for VCF outputs generated after different lossy methods are applied to the quality scores in the file `NA12878.chr20.qual`. The rows are ordered by increasing compressed size; scores greater than 99% are highlighted.

Method	Size (MB)	Variations Found	Precision	Recall	F-Score
OneQual, $q =$ "?" $= 63$	0.0	83,859	97.7	95.2	96.4
R-Block, $\theta = 4.2$	138.7	86,971	98.3	**99.3**	98.8
P-Block, $p = 16$	141.5	87,172	98.3	**99.6**	**99.0**
R-Block, $\theta = 1.4$	256.6	87,038	98.7	**99.9**	**99.3**
QualComp, $r = 0.5$	310.5	86,757	98.8	**99.6**	**99.2**
P-Block, $p = 4$	562.8	86,495	**99.4**	**99.9**	**99.6**
QualComp, $r = 1$	621.1	86,543	**99.1**	**99.7**	**99.4**
R-Block, $\theta = 1.10$	784.6	86,353	**99.6**	**99.9**	**99.8**
CRAM Bin-Preserve, 50	821.4	86,593	98.9	**99.5**	**99.2**
LogBinning, $b = 10$	829.5	86,952	98.5	**99.6**	**99.1**
CRAM Match-Bin-Preserve, 50	867.6	88,008	97.2	**99.4**	98.2
P-Block, $p = 2$	970.4	86,195	**99.8**	**99.9**	**99.8**
LogBinning, $b = 20$	1,055.3	86,506	**99.2**	**99.7**	**99.5**
UniBinning, $b = 100$	1,213.3	86,830	98.2	**99.0**	98.6
QualComp, $r = 2$	1,242.2	86,355	**99.5**	**99.8**	**99.7**
CRAM, Preserve 50	1,704.8	86,636	**99.0**	**99.7**	**99.3**
LogBinning, $b = 60$	1,774.4	86,211	**99.8**	**99.9**	**99.9**
GZip –9 (lossless)	1,901.0	86,062	**100.0**	**100.0**	**100.0**

REFERENCES

Ansorge, W. (2009). Next-generation DNA sequencing techniques. *New Biotechnology*, **25**(4), 195–203.

Burrows, M. and Wheeler, D. (1994). A block-sorting lossless data compression algorithm. Technical Report SRC-RR 124, Digital Equipment Corporation Systems Research Center.

Cánovas, R. and Moffat, A. (2013). Practical compression for multi-alignment genomic files. In *Proc. 36th Australasian Computer Science Conf.*, pages 51–60, Adelaide, Australia.

Cha, S. (2007). Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, **1**(4), 300–307.

Church, G. M. (2006). Genomes for all. *Scientific American*, **294**(1), 46–54.

Cock, P. A., Fields, C. J., Goto, N., Heuer, M. L., and Rice, P. M. (2010). The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, **38**(6), 1767–1771.

Danecek, P., Auton, A., Abecasis, G., Albers, C. A., Banks, E., DePristo, M. A., Handsaker, R. E., Lunter, G., Marth, G. T., Sherry, S. T., McVean, G., Durbin, R., and 1000 Genomes Project Analysis Group (2011). The variant call format and VCFtools. *Bioinformatics*, **27**(15), 2156–2158.

Deorowicz, S. and Grabowski, S. (2011). Compression of DNA sequence reads in FASTQ format. *Bioinformatics*, **27**(6), 860–862.

Elias, P. (1975). Universal codeword sets and representations of the integers. *Information Theory, IEEE Transactions on*, **21**(2), 194 – 203.

Ewing, B. and Green, P. (1998). Base-calling of automated sequencer traces using Phred.II. Error probabilities. *Genome Research*, **8**(3), 186–194.

Fritz, M. H., Leinonen, R., Cochrane, G., and Birney, E. (2011). Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Research*, **21**(5), 734–740.

Giancarlo, R., Scaturro, D., and Utro, F. (2009). Textual data compression in computational biology: A synopsis. *Bioinformatics*, **25**(13), 1575–1586.

Janin, L., Rosone, G., and Cox, A. J. (2013). Adaptive reference-free compression of sequence quality scores. *Bioinformatics*. To appear.

Kozanitis, C., Saunders, C., Kruglyak, S., Bafna, V., and Varghese, G. (2010). Compressing genomic sequence fragments using SLIMGENE. In *Proceedings of the 14th Annual international conference on Research in Computational Molecular Biology*, RECOMB'10, pages 310–324.

Lapidoth, A. (1997). On the role of mismatch in rate distortion theory. *IEEE Transactions on Information Theory*, **43**(1), 38–47.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., and Durbin, R. (2009). The sequence alignment/map format and SAMtools. *Bioinformatics*, **25**(16), 2078–9.

Mardis, E. R. (2008). Next-generation DNA sequencing methods. *Annual Review of Genomics and Human Genetics*, **9**(1), 387–402.

Moffat, A. and Turpin, A. (2002). *Compression and Coding Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA.

Myllykangas, S., Buenrostro, J., and Ji, H. P. (2012). Overview of sequencing technology platforms. In *Bioinformatics for High Throughput Sequencing*, pages 11–25. Springer New York.

Nielsen, R., Paul, J. S., Albrechtsen, A., and Song, Y. S. (2011). Genotype and SNP calling from next-generation sequencing data. *Nature Reviews Genetics*, **12**(6), 443–451.

Ochoa, I., Asnani, H., Bharadia, D., Chowdhury, M., Weissman, T., and Yona, G. (2013). QualComp: a new lossy compressor for quality scores based on rate distortion theory. *BMC Bioinformatics*, **14**(1), 187.

Tembe, W., Lowey, J., and Suh, E. (2010). G-SQZ: compact encoding of genomic sequence and quality data. *Bioinformatics*, **26**(17), 2192–2194.

Wan, R., Anh, V. N., and Asai, K. (2012). Transformations for the compression of FASTQ quality scores of next-generation sequencing data. *Bioinformatics*, **28**(5), 628–635.