Adaptive reference-free compression of sequence quality scores

Lilian Janin¹, Giovanna Rosone² and Anthony J. Cox¹, *

¹Computational Biology Group, Illumina Cambridge Ltd., Chesterford Research Park, Little Chesterford, Essex CB10 1XL, UK and ²Dipartimento di Matematica e Informatica, University of Palermo, Via Archirafi 34, 90123 Palermo, Italy

ABSTRACT

Motivation: Rapid technological progress in DNA sequencing has stimulated interest in compressing the vast datasets that are now routinely produced. Relatively little attention has been paid to compressing the quality scores that are assigned to each sequence, even though these scores may be harder to compress than the sequences themselves. By aggregating a set of reads into a compressed index, we find that the majority of bases can be predicted from the sequence of bases that are adjacent to them and, hence, are likely to be less informative for variant calling or other applications. The quality scores for such bases are aggressively compressed, leaving a relatively small number at full resolution. As our approach relies directly on redundancy present in the reads, it does not need a reference sequence and is, therefore, applicable to data from metagenomics and de novo experiments as well as to re-sequencing data.

Results: We show that a conservative smoothing strategy affecting 75% of the quality scores above Q2 leads to an overall quality score compression of 1 bit per value with a negligible effect on variant calling. A compression of 0.68 bit per quality value is achieved using a more aggressive smoothing strategy, again with a very small effect on variant calling.

Availability: Code to construct the BWT and LCP-array on large genomic data sets is part of the BEETL library, available as a github repository at git@github.com:BEETL/BEETL.git.

Contact: acox@illumina.com

Received on March 17, 2013; revised on April 22, 2013; accepted on April 30, 2013

1 INTRODUCTION

The raw output of a DNA sequencer is converted by a program known as a base caller into nucleotide bases, each of which is typically assigned a quality score that estimates the probability that the base has been sequenced correctly. Quality scores have long been used to trim the low-quality ends of reads and for accurate consensus sequence determination (Bonfield and Staden, 1995; Marth et al., 1999). More recently, they have enabled more accurate alignments of the shorter sequences produced by ‘next-generation’ technologies, by allowing the aligner to give lower weight to mismatches at less reliable base positions (Li et al., 2008; Smith et al., 2008). Often quality scores are expressed on an integer scale derived from the error probability p via the formula $-10 \log_{10} p$, a scoring scheme named after the Phred base caller (Ewing and Green, 1998) that first used it.

The widely used FASTQ format (Cock et al., 2010) stores the sequence and metadata of a set of DNA reads as ASCII text, together with one-character-per-score quality strings that encode the Phred scores of their bases. The different properties of these three data types have meant that many FASTQ compression methods have treated them as three distinct data streams and applied separate compression strategies to each.

The metadata field tends to be formatted in ways that are specific to the technology that was used to generate the sequence, and some FASTQ compressors have exploited such structure to improve compression. However, there is no global format specification for the metadata; therefore, any universally applicable method for its compression must necessarily be a generic exercise in the compression of ASCII text.

Although standard text compressors, such as gzip (www.gzip.org, Jean-loup Gailly and Mark Adler), do not significantly outperform a naïve 2 bits per base encoding on DNA sequence data, applications such as re-sequencing and de novo assembly typically rely on a 20-fold or more oversampling of the underlying genome, and this redundancy can be exploited to improve compression of the sequences themselves. Reference-based compression tools, such as CRAM (Fritz et al., 2011), encode reads in terms of differences between their sequences, and the sites they align to on a reference sequence. Sorting the reads by the coordinates of these alignments saves most of the overhead of storing their positions and is a convenient ordering for applications, such as SNP calling and visualization.

Despite these advantages, reference-based compression suffers when the reference is incomplete (reads that do not align cannot be compressed), subject to change or not present at all (as in metagenomics), motivating an interest in reference-free compression methods. The tool QUIP (Jones et al., 2012) creates an on-the-fly de novo assembly to perform reference-based compression against, whereas SCALCE (Hach et al., 2012) places similar reads near to each other in a sorted file, facilitating good performance by standard tools such as gzip that operate on a buffer of text at a time.

Another widely used generic compression tool bzip2 (www.bzip.org, Julian Seward) exemplifies Burrows–Wheeler transform (BWT) compression: text is split into 900 kb blocks, and the BWT of each block is computed. The BWT is a reversible permutation of the text that acts as a compression booster for the pipeline of standard compression steps that bzip2 subsequently applies. In Cox et al. (2012a), two of the present authors showed that although bzip2 performs comparably with gzip on DNA sequence reads, the compression achieved by BWT-based methods improves by >3-fold if the BWT of the entire read set is built, as
this captures redundancy between reads that were widely spaced in the original file. It was shown that compression can be further boosted by pre-sorting the reads or applying an implicit sorting strategy while the BWT is being built, enabling compression of better than 0.5 bits per base to be achieved.

Lossless approaches to the compression of quality scores have exploited empirical relationships between the scores assigned to bases within a read: for instance, Illumina quality scores tend to be monotone decreasing along a read with a decrease in scores between adjacent bases that is usually small. An overreliance on such observations potentially ties a compression scheme to a given sequencing technology and makes it sensitive to changes in sequencing protocol. Moreover, it is likely that Illumina quality scores at their full resolution contain a proportion of random noise that is impossible to compress: striking evidence for this is given by Table 4 in Bonfield and Mahoney (2013), which shows multiple entrants to the SequenceSqueeze competition for FASTQ compression achieving similar lossless compressions of ~2.94 bits per score on a test dataset, but that no entrants were able to improve on this figure.

It is undesirable that when compressed, the quality scores should take up several times more space than the sequences themselves; therefore, we are led to consider compressing them in a *lossy* way. Kozanitis et al. (2010) found that a global reduction in the resolution of the scores from 40 to 8 values (thus permitting each score to be stored in 3 bits) had no significant impact on the quality of variant calls, whereas strategies for global re-quantization of quality scores were studied in more detail by Wan et al. (2012).

However, treating all scores in the same way ignores the fact that most of them could likely be reduced in resolution or even discarded entirely with little impact on our ultimate goal of ensuring that analyses performed with the reduced scores closely reflect the results obtained from the original data. In a human re-sequencing context, for example, if a large coverage of high-quality bases unanimously supports a homozygous match to the re-sequencing context, for example, if a large coverage of high-quality bases is discarded entirely with little impact on our ultimate goal of making the output more compressible even by simple compressors (for instance see Restivo and Rosone, 2011).

A way to generalize the BWT to a set $S = \{s_1, s_2, \ldots, s_N\}$ of strings is simply to apply distinct end markers $\mathcal{S}_i$ to each $s_i$ such that $s_1 < \ldots < s_N < c$ for any $c \in \Sigma$. For a single string, the permutation $s \mapsto \text{BWT}(s)$ provides a relation to the suffix array of $s$, which is defined by applying the same permutation to the integers $0, \ldots, |s| - 1$, so as to arrange the starting positions of the suffixes of $s$ into lexicographical order. The BWT of a collection is related to its *generalized suffix array* in an analogous way. Formally, the GSA is defined such that $\text{GSA}(S[j])$ gives the position of the $j$th smallest suffix of the strings in $S$, which is encoded as a pair $(t, i)$ denoting that the suffix starts at position $i$ of $s_i$. In particular, if $\text{GSA}(S[j]) = (t, i)$ then $\text{BWT}(S[j]) = s_i[t \bmod |s_i|]$.

Now we suppose the elements of $S$ are DNA sequences that are accompanied by strings $Q = \{q_1, q_2, \ldots, q_N\}$ such that the symbol $q_i[j]$ encodes the quality score of $s_i[j]$. The alphabet used by $Q$ and the means of encoding are not relevant at this point. We apply the same permutation $S \mapsto \text{BWT}(S)$ to $Q$ (which we emphasize is not the same as computing the BWT of $Q$ itself) to obtain a string $Q'V$ such that $Q'[j]$ encodes the quality score associated with the symbol $\text{BWT}(S)[j]$.

The *longest common prefix array* (denoted by LCP) of a collection $S$ of strings stores the length of the longest common prefixes between two consecutive suffixes of $S$ in the lexicographic order. For every $j = 1, \ldots, n - 1$, if $\text{GSA}(S[j - 1]) = (p_1, p_2)$ and $\text{GSA}(S[j]) = (q_1, q_2)$, LCP$(S[j])$ is the length of the longest common prefix of suffixes starting at positions $p_1$ and $q_1$ of the words $s_{p_2}$ and $s_{q_2}$, respectively. We set
LCP(S)[0] = 0. Note that we do not need to compute explicitly the generalized suffix array to obtain the BWT and LCP. Methods suitable for computing BWT(S) and LCP(S) where S is a large collection of DNA sequences and without the use of the GSA of S were given in Bauer et al., (2011, 2012, 2013), and it is straightforward to adapt them to compute QV at the same time.

2.2 Smoothing strategy
A crucial consequence of the definition of the BWT is that the symbols associated with suffixes that begin with some string w will form a contiguous substring in BWT(S), we call this the w-interval. If all symbols in the w-interval take the same value c, then every occurrence of w in S is preceded by c; seeing w in a read predicts that c will come before it.

For a fixed length k, a linear scan through the LCP array allows us to identify LCP-intervals, which are maximal intervals [i, j] that satisfy LCP[i] ≥ k for i ≤ r ≤ j and whose associated suffixes, therefore, share at least the first k bases. We set thresholds on the minimum length of the predicting context k and the minimum number of times j - i + 1 it must occur in S. If these are both exceeded and the symbols in BWT(S)[i, j] are all the same, then we smooth the corresponding quality scores in QV(S)[i, j].

As each score in QV(S)[i, j] implies an error probability for its associated base, one way to do the smoothing would be to take the mean of these error rates across QV(S)[i, j] and convert this to a score with which we replace all scores in QV(S)[i, j] (which we note is not the same as taking the mean of the scores). However, better compression is obtained by replacing the smoothed scores with the score implied by the mean error rate of all bases whose scores are smoothed. Moreover, we empirically observed that almost as good results are achieved just by smoothing all quality scores in QV that are associated with runs of a given symbol in BWT(S) whose lengths exceed a threshold, which has the practical benefit of not needing the LCP array.

2.3 Measuring the information loss because of smoothing
The probability distribution of a randomly chosen symbol from a string s is \( p(s) = (n_1/|s|, \ldots, n_{|\Sigma|}/|s|) \), where \( n_1, \ldots, n_{|\Sigma|} \) count the occurrences of the symbols of \( \Sigma \) in s. Applying the Shannon entropy transformation \( H: (p_1, \ldots, p_n) \rightarrow -\sum_i p_i \log p_i \) to this distribution (Shannon, 1948) yields

\[
H_0(s) = H(p(s)) = \sum_{i \in \Sigma} \frac{n_i}{|s|} \log \frac{|s|}{n_i}
\]

a quantity known as the zero-order empirical entropy of s (we assume 0 log 0 = 0 and adopt the convention that all logarithms are taken to be the base 2).

Let \( b(w, s) \) be the string formed by concatenating the symbols that immediately precede the occurrences of some substring \( w \) of s. For a positive integer k, we define the kth order empirical entropy of s as

\[
H_k(s) = \frac{1}{|s|} \sum_{w \in \Sigma^k} |b(w, s)| H_0(b(w, s))
\]

It can be thought of as the mean entropy across all symbols of s when a context of k bases is taken into consideration. The computations of \( H_0(s) \) and \( H_k(s) \) for a collection S are identical to the single-string case. We note here the connection with BWT(S) and LCP(S); the strings \( b(w, S) \) form disjoint substrings in BWT(S) whose order in the BWT matches the lexicographic order of their associated k-mers \( w \) (see also Manzini, 2001). The coordinates of the strings \( b(w, S) \) in S are precisely the LCP-intervals of length k, which we have observed can be computed by a single pass through LCP(S).

We can view \( H_0(b(w, S)) \) as the Shannon entropy of the distribution \( p(b(w, S)) \) obtained by assuming each symbol of \( b(w, S) \) is exactly known (so that e.g. a ‘G’ corresponds to a distribution \( p(A, C, G, T) = (0, 0, 1, 0) \)) and then computing the mean of these distributions across all symbols of \( b(w, S) \). With this in mind, we can generalize \( H_0(b(w, S)) \) to imprecisely known symbols by replacing these exact symbol-level distributions with the ones that are implied by the quality values that are associated with the elements of \( b(w, S) \). For example, a Q20 ‘G’ receives a distribution \( p(A, C, G, T) = (0.01/3, 0.01/3, 0.99, 0.01/3) \) — we assume the three error bases are equiprobable. Taking the mean of these gives a new ‘quality-aware’ distribution \( q(b(w, S), Q) \) for the symbol expected to precede \( w \), from which we can compute modified entropy \( H_0(b(w, S), Q) = H(q(b(w, S), Q)) \) via the Shannon formula. This in turn implies a generalization of the kth order empirical entropy:

\[
H_k(S, Q) = \frac{1}{|S|} \sum_{w \in \Sigma^k} |b(w, S)| H_0(b(w, S), Q)
= \frac{1}{|S|} \sum_{w \in \Sigma^k} |b(w, S)| H(q(b(w, s), Q)).
\]

We give two ways to describe the effect of smoothing Q to obtain a new set of scores \( Q' \). First, the improvement in compression is measured by comparing the size of the files produced when standard compression tools are applied to the BWT-permuted quality scores \( QV \) and \( QV' \). Second, the information loss is quantified by the relative entropy (or Kullback-Liebler divergence), which measures the information loss when a distribution \( p' = (p'_1, \ldots, p'_n) \) is used to approximate a distribution \( p = (p_1, \ldots, p_n) \) and is defined by

\[
\text{RE}(p|p') = \sum_i p_i \log \frac{p_i}{p'_i}.
\]

In much the same way as \( H_k(S) \) is defined from \( H_0(S) \), we may define the kth order empirical relative entropy \( \text{RE}(Q||Q') \) as (see also Epifanio et al., 2011)

\[
\frac{1}{|S|} \sum_{w \in \Sigma} |b(w, S)| \text{RE}(q(b(w, S), Q)||q(b(w, S), Q'))
\]

Similar to \( H_k(S) \), a single pass through LCP(S) allows us to compute \( \text{RE}(Q||Q') \) by enumerating the w-intervals associated with each k-mer: this time, we need the w-intervals in BWT(S) plus the corresponding scores in \( QV \) and \( QV' \) to compute the terms \( \text{RE}(q(b(w, S), Q)||q(b(w, S), Q')) \).

Given two smoothings \( Q \rightarrow Q' \) and \( Q \rightarrow Q'' \), the smaller of \( \text{RE}(Q||Q') \) and \( \text{RE}(Q||Q'') \) suggests the smallest loss in information content. It can be verified that \( \text{RE}(Q||Q') \geq 0 \) and that \( \text{RE}(Q||Q') \) is only zero when the distributions \( q(b(w, S), Q) \) and \( q(b(w, S), Q') \) are identical for all k-mers \( w \) in S.

The smoothing scheme we gave in Section 2.2 can be understood in this context. If all of \( b(w, S) \) support the same base call, then it must be true that \( q(b(w, S), Q) \) assigns some probability \( p \) to that call and probability \( 1 - p/3 \) to the remaining bases. Replacing all the quality scores in \( b(w, S) \) with the score corresponding to \( p \) will create a smoothed set of scores \( Q' \) for which \( q(b(w, S), Q') = q(b(w, S), Q') \); hence, \( \text{RE}(Q'||Q') = 0 \). In practice, the integer nature of the Phred scoring scheme means the value used to overwrite the smoothed scores will typically be a slight approximation to the score that corresponds to \( p \). However, we found that \( p \) did not vary much between different intervals \( b(w, S) \), and that small changes to its value had little effect on results. We, therefore, chose to improve compression by replacing all smoothed scores with a globally chosen replacement value.
3 RESULTS

3.1 Parameter sweep

Using Caenorhabditis elegans data allowed us to study the effect of compression on variant calls made using whole-genome shotgun sequences from a diploid genome, while permitting more extensive parameter sweeps than would be tractable for human data. We chose a dataset SRR065390_1 comprising 33,808,546 reads of length 100 (33.6× coverage of the genome) that has been previously studied by the Sequence Squeeze entrants. For our computational pipeline, we chose bwa [bwa version 0.6.1 with parameters -t 12 -q 15 (Li and Durbin, 2009)] followed by GATK [GATK version 1.6 (DePristo et al., 2011)]. Only the variants marked as ‘PASS’ by GATK (using UnifiedGenotyper and Variant Quality Score Recalibration) are considered.

We treat the results of this pipeline on uncompressed data as ‘ground truth’ (i.e. we do not try to distinguish false positive or false negative calls in the uncompressed data, as we would on a simulated dataset), as we wish results derived from compressed data to reflect the original data as closely as possible. We chose to measure the proportion of reads that were differentially mapped and the proportion of variant calls that were different. We calculated sensitivity and specificity values and combined the two into an F-statistic.

Qualities are distributed as shown in Figure 1. We notice an overrepresentation of Q2 scores (8.3%) because of end-of-reads trimming. A negligible number of ’N’ bases (0.07%) are also associated to Q0 scores. We verified that better results were obtained when these Q0 and Q2 scores were conserved, and all the results presented later in the text keep these two Q scores unchanged. To gain an idea of ‘worst case scenario’ behaviour, we first computed the mean estimate error rate by converting each quality score (ignoring Q0 and Q2) to an error probability, taking the mean of these values and converting back to Phred score (which we note is not the same as taking the mean of the quality scores). We obtained a mean Q score of 28.36 (as a comparison, the mean of the Q scores is 32.74). We then replaced every Q score by this mean value and reran our pipeline. This caused 15% of variant calls to change (2.3% missing calls and 12.7% new variants calls), also characterized by the worst-case F-score of 92.8%.

We then compressed the quality scores using various values for the two parameters of our quality smoothing algorithm: LCP cut threshold and minimum stretch length. All stretches (in BWT-space) of same BWT letter longer than (or equal to) the minimum stretch length, and for which the LCP value stays above (or equal to) the LCP cut threshold gets ‘smoothed’, i.e. its associated qualities get reset to a constant quality score value (except for Q0 and Q2, which stay at their current values but are still allowed to contribute to stretches).

In theory, as each set of parameters leads to different numbers of quality values being smoothed, the average Q score that should be used as a replacement should not be constant across runs: to strictly follow the Q score definition associated with error rates, we recalculated for each run the Q score that would lead to the same mean error rate and used it as a replacement. However, we noticed that it only varied between 28.36 and 30.46, and that its effect on variant calling was negligible. For this reason, the results presented below are using a fixed replacement Q score of 29 for all runs.

Table 1 summarizes the results obtained for a sweep of parameters. In this table, the results of the ‘LCP-free’ smoothing appears when ‘LCP cut threshold = 0’. For each combination of parameters, we compressed the BWT-ordered smoothed qualities using bzip2 and 7-Zip’s PPMd algorithm (7-Zip with parameters -m0 = PPMd; http://www.7-zip.org). The 7-Zip compression was always better and is the one reported in the table, in bits per quality value, in BWT-space (column 3) and in read-space (column 4). We also calculated F-scores (column 5) based on the number of false-positive and false-negative variant calls.

We notice that minimum stretch length is the parameter having the most influence on both the compression rate and the fidelity of variant calling. A minimum stretch length of 10, meaning that 10 consecutive identical BWT letters must share the same following k-mer (whose length depends on the LCP cut threshold) to get smoothed, already leads to 76% of the qualities getting replaced (excluding Q0 and Q2). In this situation, the quality string in BWT-space gets compressed at 1.28 bits per quality value, in read-space (i.e. same as the original FASTQ) at 1 bit per base. The effect on variant calling is minimal, as the F-score of 99% corresponds to 30 variants getting missed (i.e. false negatives) from the original 3208 called and 40 variants being created (i.e. false positives). However, in all the cases, the false-negative and false-positive calls are present in the opposite variants file but did not pass filter because of a quality slightly below the required threshold.

The LCP cut threshold has marginal effect on both the compression rate and the variant calling. In fact, it has unpredictable effect on variant calling, where reduction of the threshold sometimes leads to slightly better and sometimes to slightly worse results. An interesting observation is that reducing the LCP cut threshold to zero, i.e. when the LCP value is not used at all, does not affect the results as negatively as we expected: in this situation, any stretch of consecutive identical BWT letters gets smoothed even if the letters do not share a common following k-mer (as the minimal length of this k-mer is zero). This is explained by the relatively low number of occurrences of such stretches: Table 1 column 3 reveals that only 0.2% of the bases get affected by a change of LCP cut threshold from 5 to 0 in the most interesting case where min stretch length is 5.

Another interesting observation is the better compression obtained in read-space than in BWT-space. This is explained by the...
tendency for bases to stay constant from one cycle to the next (a property observable in read-space), but do not have any reason to stay constant across reads sharing the same suffix. This makes compression in BWT-space more difficult.

We also wished to verify that randomly smoothing qualities had a more detrimental effect on variant calling. Based on the distribution of smoothed stretch lengths obtained with LCP cut threshold \( \frac{1}{5} \) and minimal stretch length \( \frac{1}{5} \), we smoothed random stretches of the original dataset in such a way as to achieve the same final distribution. After running our computational pipeline on this randomly smoothed dataset, we obtained the statistics shown in Table 2: same compressibility in BWT-space, but much worse in read-space, and worse variant calling F-score. The lower compressibility in read space is explained by the fact that the qualities being replaced are now distributed less consecutively than before in read space: our smoothing strategy, even though applied in BWT space, is occurring at specific places that are permuted into consecutive positions in read space.

Instead, the smoothing of random BWT stretches does not get permuted into consecutive positions in read space and leads to a worse compression rate. It can be noted that some compression still happens because 85.9% of the qualities, which had various values before smoothing, are reduced to a single value and, therefore, become more compressible.

Finally, Figure 2 depicts the \( k \)th order relative entropy between the full-resolution quality scores and a subset of the compression schemes mentioned in Tables 1 and 2. We would expect the worst-case information loss to occur when all quality scores are replaced by a constant value and the ‘c0s1’ curve behaves accordingly, exhibiting the highest relative entropy. The ‘c5s5’ and ‘c5s10’ curves correspond to the ‘stretch length 5’ and ‘stretch length 10’ entries in Table 1 for an LCP cut threshold of 5. As we would expect, the more aggressive of the two schemes ‘c5s5’ has higher relative entropy with respect to the original scores, suggesting a greater loss of information. The curve for ‘c5s5’ contrasts with the ‘c5s5 (rand)’ curve for the randomly smoothed dataset compared with it in Table 2. Although Table 2 shows nearly identical compression for the two datasets, Figure 2 reveals that randomly smoothing the quality scores has caused a much greater loss of information.

### Table 1. Statistics after quality smoothing

<table>
<thead>
<tr>
<th>LCP cut threshold</th>
<th>Min stretch length</th>
<th>%Q3 + cut</th>
<th>BWT-space compress (bits/Q)</th>
<th>Read-space compress (bits/Q)</th>
<th>Variant calling F-score (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncut</td>
<td>10</td>
<td>0</td>
<td>2.51</td>
<td>1.67</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>76.7</td>
<td>1.28</td>
<td>1.00</td>
<td>99.1</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>76.8</td>
<td>1.28</td>
<td>0.99</td>
<td>99.1</td>
</tr>
<tr>
<td>0(^c)</td>
<td>10</td>
<td>76.8</td>
<td>1.28</td>
<td>0.99</td>
<td>98.8</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>85.9</td>
<td>1.06</td>
<td>0.68</td>
<td>97.8</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>86.1</td>
<td>1.06</td>
<td>0.68</td>
<td>97.9</td>
</tr>
<tr>
<td>0(^c)</td>
<td>5</td>
<td>86.1</td>
<td>1.06</td>
<td>0.68</td>
<td>97.7</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>96.9</td>
<td>0.50</td>
<td>0.20</td>
<td>92.3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>99.3</td>
<td>0.39</td>
<td>0.11</td>
<td>92.9</td>
</tr>
<tr>
<td>0(^c,d)</td>
<td>1</td>
<td>100</td>
<td>0.37</td>
<td>0.06</td>
<td>92.8</td>
</tr>
</tbody>
</table>

\(^a\)Compression in BWT-space in bits per quality value. \(^b\)Compression in read space (i.e. same as FastQ file) in bits per quality value. \(^c\)LCP cut threshold = 0 does not use LCP. \(^d\)All cut except Q0 and Q2, which are kept intact in all cases.

### Table 2. Comparison with random smoothing of same stretch lengths distribution

<table>
<thead>
<tr>
<th>Smoothing strategy</th>
<th>BWT-space compression (bits/Q)</th>
<th>Read-space compression (bits/Q)</th>
<th>Variant calling F-score (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original cut5 stretch5</td>
<td>1.06</td>
<td>0.68</td>
<td>97.8</td>
</tr>
<tr>
<td>Random same dist.</td>
<td>1.05</td>
<td>1.09</td>
<td>94.8</td>
</tr>
</tbody>
</table>

Fig. 2. The \( k \)th order empirical relative entropy between the original and compressed quality scores of SRR065390_1, for various quality-score compression schemes

3.2 Details and re-quantization

The F-scores presented in the previous section, although good at summarizing the overall impact of compression on variant calling, are abstracting away some important details. In this section, we use QQ plots to show the correlation between variant call qualities (QUAL field reported by GATK) with and without compression. These QQ plots are also highlighting some
interesting differences between our BWT-based compression strategy and the re-quantization strategy (reduction from 40 to 8 quality scores) at similar compression and F-score rates.

Considering our smoothing strategy with parameters (cut threshold = 5, min stretch length = 10), which was leading to a compression rate of 1.28 bits/Q in BWT space and 0.99 bit/Q in read space and was associated to a F-score of 99.1%, Figure 3 shows for each variant called its quality before (x-axis) and after compression (y-axis). We distinguish three classes of variants: true positives (TP) are those called as 'passing filter' (as defined by GATK) before and after compression; false positives (FP) are those passing filter after compression but not before; false negatives (FN) are those passing filter before compression but not after. FP and FN calls, although passing filter in only one of the two GATK runs, always happen to be present in both call sets, allowing us to plot their quality values.

Figure 3 shows 36 FP, 25 FN and 3183 TP calls. We notice that the majority of stray points are FP calls whose quality has been enhanced by our algorithm. In fact, all the qualities considered in this QQ plot, before and after compression, are above the filter threshold of quality 30 (the x- and y-axis start at Q = 30). The reason for calls not passing filter in the original dataset is most often because of the LowQD filter, but a direct link to the aligner’s mapping quality of reads or number thereof has not been established.

For comparison with another compression strategy, Figure 4 shows the QQ plot obtained after re-quantization of the original dataset to eight Qscore bins. This led to the same compression rate as the c5s10 strategy (which was in fact chosen for this reason): 1.29 bits/Q in BWT space. The F-score of re-quantization, 98.8%, was also similar to c5s10’s. However, re-quantization achieves this F-score with 27 FP, 51 FN and 3157 TP calls, which is much more biased towards FN, whereas c5s10 (and all our other observed results from this smoothing strategy) was more pronounced towards FP.

Because of the dataset used in this article, we have not reached any conclusion regarding the quality of those FP and FN calls: some FP calls may introduce real false-positive calls, whereas others may reveal some real variants that had not been called in the original dataset. Similarly, re-quantization’s FN calls may be correct pruning of previously incorrect calls, as well as real false negatives. We intend to run the same analysis on a simulated dataset where we will have previous knowledge of the correctness of calls.

Finally, Figure 5 shows the distribution of 108 FP, 44 FN and 3164 TP obtained with c0s5 smoothing. Relatively to c5s10, the number of FP increases faster than the number of FN, and this trend continues when we push the smoothing parameters towards lower min stretch size and higher compression rates. Also, confirming an expected behaviour, the FP calls present after c5s10 smoothing are still present as FP calls after the more compressed c0s5 smoothing.
4 DISCUSSION

This article aims to introduce the general idea of smoothing quality scores based on the BWT and LCP of their associated reads. Section 2.2 describes perhaps the simplest and most conservative approach to this, but the theoretical framework presented in Section 2.3 allows comparison of future more sophisticated quality smoothing strategies. Note that the effect of smoothing or otherwise adjusting the quality scores is to change the weightings of the different nucleotides in the distributions \( X(w, QV) \). We can take a step further and consider adjusting low-probability bases in \( X(w, QV) \) to zero. Our work can thus be extended to a new quality-based view on de novo error correction that may provide an interesting alternative to existing approaches, many of which are based on the counting of k-mers (as surveyed by Yang et al., 2013). Such a strategy could be thought of as a quality-aware extension of the HiTEC algorithm (Ilie et al., 2011), while enjoying the considerable space advantage of being based on a (potentially compressed) BWT instead of a suffix array.

Although it is an advantage that the relative entropy measures the information lost by quality smoothing in an application-neutral way, we also recognize that a single numerical quantity cannot fully model the effect of quality smoothing on the often complex multi-step analysis pipelines that are applied to sequence data. We investigated this by measuring the effect of smoothed quality scores on the results of widely accepted tools for a well-understood application.

Transforming the smoothed scores back into their original reads added a significant boost to the compression already achieved by 7-Zip from exploiting similarities between the quality scores of individual read. It is equally simple to combine our approach with the application to the unsmoothed scores of one of the lossy re-quantization schemes studied by Wan et al. (2012).

However, using our method in this way involves building the BWT (and possibly LCP) of a set of reads and then applying the inverse BWT permutation to the quality scores to obtain a smoothed quality string for each read. The overhead of these tasks may limit its practicality for downstream applications that operate on reads and does not use the potential of the BWT. As well as allowing excellent lossless compression, storing sequences in BWT form also facilitates rapid analysis: the sga (Simpson and Durbin, 2012) and Fermi (Li, 2012) assemblers both operate directly on BWT-based compressed indexes of sets of reads, and we ourselves have shown that similar data structures can be the basis of both RNA-Seq (Cox et al., 2012b) and metagenomic (Ander et al., 2013) analyses. Although the compression achieved on BWT-space reads is less than in read-space (although the difference may be less clear-cut on a dataset where the Q2-masking of read ends is less prevalent or has been switched off), we, therefore, envisage that a key application of our work is to allow quality scores to be used in a BWT-space context while being stored in as compact a manner as the reads themselves.

Conflict of Interest: L.J. and A.J.C. are employees of Illumina Inc., a public company that develops and markets systems for genetic analysis. They receive shares as part of their compensation.

REFERENCES