Bioinformatics, 2016, 1–9
doi: 10.1093/bioinformatics/btw385
Advance Access Publication Date: 26 June 2016
Original Paper

OXFORD

Sequence analysis

GeneCodeq: quality score compression and improved genotyping using a Bayesian framework

Daniel L. Greenfield[1,*], Oliver Stegle[2] and Alban Rrustemi[1]

[1]PetaGene, Ideaspace, 3 Charles Babbage Rd, Cambridge CB3 0GT, UK and [2]European Molecular Biology Laboratory, European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SQ, UK

*To whom correspondence should be addressed.
Associate Editor: Inanc Birol

Received on October 8, 2015; revised on May 31, 2016; accepted on June 15, 2016

Abstract

Motivation: The exponential reduction in cost of genome sequencing has resulted in a rapid growth of genomic data. Most of the entropy of short read data lies not in the sequence of read bases themselves but in their Quality Scores—the confidence measurement that each base has been sequenced correctly. Lossless compression methods are now close to their theoretical limits and hence there is a need for lossy methods that further reduce the complexity of these data without impacting downstream analyses.

Results: We here propose *GeneCodeq*, a Bayesian method inspired by coding theory for adjusting quality scores to improve the compressibility of quality scores without adversely impacting genotyping accuracy. Our model leverages a corpus of *k*-mers to reduce the entropy of the quality scores and thereby the compressibility of these data (in FASTQ or SAM/BAM/CRAM files), resulting in compression ratios that significantly exceeds those of other methods. Our approach can also be combined with existing lossy compression schemes to further reduce entropy and allows the user to specify a reference panel of expected sequence variations to improve the model accuracy. In addition to extensive empirical evaluation, we also derive novel theoretical insights that explain the empirical performance and pitfalls of corpus-based quality score compression schemes in general. Finally, we show that as a positive side effect of compression, the model can lead to improved genotyping accuracy.

Availability and implementation: *GeneCodeq* is available at: github.com/genecodeq/eval

Contact: dan@petagene.com

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Over the past decade, unprecedented advances in next generation sequencing (NGS) technologies have led to a dramatic reduction in sequencing cost (Wetterstrand, 2015) and much faster data production rates (Illumina, 2014). These technological advances are fostering increasingly wide-ranging applications in biotechnology, public healthcare (Berg *et al.*, 2011) and personalized medicine (Fernald *et al.*, 2011). Furthermore, as genomic sequencing data has grown exponentially, they have outpaced advances in some information technologies that they indirectly rely on: computing power and storage (Berger *et al.*, 2013; Stephens *et al.*, 2015). In particular, genome sequencing results in a large storage footprint for each genome. Thus, storing and transferring raw sequencing information is becoming prohibitively expensive (Baker, 2010) and effective compression schemes of raw sequencing data are indispensable.

The majority of NGS data output consists of read sequence information whereby each nucleotide base is associated with a sequence confidence level (also referred to as quality score), produced by the base caller (Das and Vikalo, 2012). The Phred scale (Ewing and Green, 1998) ($Q = -10\log_{10}P$) encodes with integer Quality Score Q an estimate of the probability P that the base has been called incorrectly. This information is used both for the quality control of raw read data and for downstream processing, including genome assembly, read mapping and genotyping (DePristo *et al.*, 2011; Li, 2011).

In compressed genomic datasets, quality score values take up the dominating share. For example, when compared to the read sequence data, quality scores of Illumina reads take at least 2.3× more storage, although this can be an even higher ratio when using more aggressive sequence compression (Benoit *et al.*, 2015; Cox *et al.*, 2012; Grabowski *et al.*, 2015). Quality scores are more difficult to compress due to a larger alphabet (63–94 in original form) and intrinsically have a higher entropy (Yu *et al.*, 2014). With lossless compression algorithms and entropy encoders reaching their theoretical limits and delivering only moderate compression ratios (Bonfield and Mahoney, 2013), there is a growing interest to develop lossy compression schemes to improve compressibility further. Quantizing quality scores (i.e. reducing the alphabet size) is the most basic approach to improve compressibility in a lossy manner.

One such approach of reducing all quality values to eight levels (bins) (Illumina, 2011) has become a widely used standard for the Illumina platform and is enabled by default on the most recent machines (Illumina, 2014). Another approach called *P-Block* (Cánovas *et al.*, 2014) involves local quantization so that a representative quality score replaces a contiguous set of quality scores that are within a fixed distance of the representative score. Similarly the *R-Block* (Cánovas *et al.*, 2014) scheme replaces contiguous quality scores that are within a fixed relative distance of a representative score. Other lossy approaches improve compressibility and preserve higher fidelity by minimizing a distortion metric such as mean-squared-error or L1-based errors (*Qualcomp* and *QVZ*) (Malysa *et al.*, 2015; Ochoa *et al.*, 2013). However, adoption of lossy compression schemes for quality scores has been slow due to concerns about adverse effects on downstream analyses, in particular genotyping accuracy (Fritz *et al.*, 2011). However, there are also reports that compression schemes such as *P-Block*, *R-Block*, *QVZ* and *Qualcomp* can, under some circumstances, lead to a slight improvement in genotyping accuracy.

A number of more recent approaches utilize the sequence data itself to guide the quality score compression. *Quartz* achieves this using a reference corpus built from frequent 32-mers across reads from individuals sequenced in the 1000 Genomes Project (Yu *et al.*, 2014, 2015). Read base pairs that match any one 32-mer in the corpus (up to one allowed mismatch per 32-mer) have their quality score set to a fixed high value. This 'sparsification' of quality scores reduces entropy, thus improving quality score compressibility. A different approach, *Leon*, (Benoit *et al.*, 2015) utilizes the dataset itself for building its set of *k*-mers, and to generate a reference probabilistic *de Brujin Graph*. In this case, bases in a read that have enough highly frequent *k*-mers covering it within the dataset are set to a fixed high-value quality score. Both methods were reported to improve genotyping accuracy.

In this article we present *GeneCodeq*, a lossy compression scheme that is inspired by coding theory (Ash, 1965; Barg, 1993) and Bayesian inference. Uniquely, *GeneCodeq* uses a statistical approach to objectively reason about the compressibility of quality scores. Briefly, our model estimates the posterior probability of a sequencing error given the evidence of the full read, including quality scores, together with information from a reference corpus. As a result, the posterior estimates of most quality scores are boosted above a saturation point of the Phred scheme, corresponding to very high confidence. This approach results both in a significant reduction in entropy and better genotyping accuracy when compared to existing methods.

2 Methods

GeneCodeq uses a basic statistical model for mutations and sequencing errors within the Bayesian framework. These processes are not modelled in full detail. Instead, this simplified generative model lends itself to an effective algorithm to estimate the probability of a sequencing error (see Fig. 1 for an overview). During sequencing, the sample genome is fragmented and individual reads are randomly selected and sequenced. If the set of all possible fragments was known a priori, these could be regarded as codewords transmitted over a noisy channel (sequencing). Given an observed sequence read, standard principles from coding theory could be applied to infer the likelihoods for each codeword and thereby accurately estimate the probability of sequencing error for each base. In real sequencing, we do not have access to the set of true fragments. Instead, *GeneCodeq* uses the reference genome as a source of possible read fragments of a defined length k (source k-mers). The mismatch between the reference and the true sample genome is then modelled as an additional source of error in the transmission process. Importantly, this additional error probability tends to be low as most reads can be well explained by the reference corpus. Hence, a reference corpus is a suitable proxy for the true sequence and again principles from coding theory can be used to estimate the posterior probability of a sequencing error.

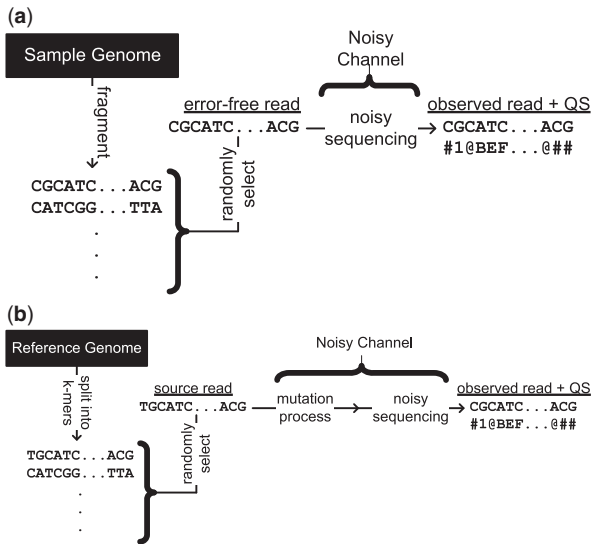

Fig. 1. Illustration of our approach to model sequencing as a noisy transmission problem. (**a**) The source genome is observed. The source genome is uniformly randomly fragmented into a set of *k*-mers. A *k*-mer (sequence read) is selected at random for transmission over the noisy channel (the sequencer), resulting in an observed read and corresponding quality scores. The posterior probability of a sequencing error can be estimated for each base, given the read, quality scores and the set of possible source *k*-mers. (**b**) The source genome is not observed. The reference genome is used as a proxy for the sample genome, and split into a set of all possible *k*-mers (a corpus). The noisy channel now introduces both mutation and sequencing errors. The posterior probability of sequencing error can be calculated as in (**a**) but also taking into account the probability of mutation in the noisy channel

When applying this approach to NGS, we consider both the sequence of the received read as well as the quality scores for each base. We use these additional data as additional evidence for the likelihood of a sequencing error. This estimate is then refined in the light of all the remaining evidence, which includes the sequence context of the entire read and its quality scores as well as the read *k*-mer corpus, which is derived from the reference.

Relationship to coding theory

The model that underlies *GeneCodeq* is related to coding theory. In the general case of transmitting binary data over a noisy medium, forward Error Correction (Ash, 1965) can be used to correct multiple bit errors. The key idea is that outgoing data is encoded using a dictionary of carefully constructed binary strings called codewords, ideally accounting for the specifics of the noisy channel. When a particular codeword is transmitted, the original transmitted codeword can be recovered with high probability, even in the presence of bit errors. The key component to enable accurate decoding is to define a dictionary of error tolerant codewords. For example, if the Hamming distance is used to determine the likelihood that the signal on the other end corresponds to one codeword versus another, it is desirable to define a dictionary where individual codewords have maximal pairwise Hamming distances to each other (Ash, 1965).

In NGS we also have the objective to identity errors and recover the true source sequence. However, unlike applications in Coding Theory, the codewords are not constructed but instead defined by a corpus, derived from e.g. a reference or sample genome (Fig. 1). Nonetheless, we can use the analytical framework of Coding Theory to solve the decoding task. Although this approach could also be used to uncover the most likely true sequence, we are not interested in error correction. Instead, *GeneCodeq* uses the noisy channel model to estimate the likelihood of sequencing error and in particular to identify bases where errors are unlikely. Intuitively, the ability to reject a sequencing error is strongly affected by the Hamming Distance between the source *k*-mer and its Hamming neighbourhood in the dictionary. Those *k*-mers which have high Hamming Distance to all other *k*-mers are easier to identify, which in turn informs the probability of a sequence error.

2.1 GeneCodeq model

GeneCodeq assumes that an unknown Sample Genome (Z) is generated from a known Reference Genome (C) with per base mutation rate m. We use a simple mutational model that assumes that each mutation event is independent and identically distributed, with a value for m informed by GWAS studies (see Supplementary Materials for more details). It is well known that mutation rates vary considerably across the genome. More complex mutational models could be straightforwardly incorporated in the model, and this would be an area worthy of future exploration. Using the reference genome, we generate a corpus of N k-mers that forms a dictionary of codewords $C = (C_1, \ldots, C_N)$ with alphabet size four (i.e. A, C, G or T), where $C_{i,j}$ denotes base j of the ith codeword. The corresponding unobserved sequences in the sample genome is denoted as Z_i, where the base $Z_{i,j}$ differs from $C_{i,j}$ due to mutations.

We purposefully model each read independently from other reads, which simplifies the overall model and is consistent with the notation of independent sampling of reads (or fragments) from the source genome. Hence, the model can be fully described when considering a single observed read. Let k denote the total number of bases in the observed read R with R_j denoting the jth base. Additionally, we observe information about the sequence uncertainty for each base $\epsilon = (\epsilon_1, \ldots, \epsilon_k)$, which are derived from the corresponding quality scores.

Our objective is to infer the posterior probability of sequencing error at each base in order to then adjust the corresponding quality scores. To achieve this, we start with a model that assumes that we know which particular source codeword C_w, for $w \in (1, \ldots, N)$ gave rise to the observed read R. We denote the presence of a sequence error with Boolean variable E_j (i.e. where R_j is not identical to the true sample sequence $Z_{w,j}$). In this model, we represent the mutational error as $\Pr(Z_{w,j}|C_{w,j}, m)$ and the error process of sequencing as $\Pr(R_j|Z_{w,j}, E_j)$. The joint probability of the observed read and the true uncorrupted sample sequence factorize, which reflects the assumption that mutations and sequence errors are independent:

$$\Pr(R, Z_w, E|C_w, m, \epsilon) = \prod_{j=1}^{k} \Pr(Z_{w,j}|C_{w,j}, m)\Pr(R_j|Z_{w,j}, E_j)\Pr(E_j|\epsilon_j). \tag{1}$$

The prior probability of a sequencing error based on the quality score ϵ_j is given by the Bernoulli distribution $\Pr(E_j = true|\epsilon_j) = \epsilon_j$. Note that while the distribution for mutational errors is identical and independent across base pairs, the sequencing error is independent across base pairs but not identically distributed, as the model accounts for base pair specific differences in quality scores ϵ_j.

The derivation in Equation (1) assumes that the true source codeword C_w is known. In principal, one would need to marginalize over all possible codewords. In order to retain computational efficiency, we use a greedy search method to limit the space of codewords to be considered (see Section 2.3). For more details and illustration diagrams on this model, please consult the section on GeneCodeq Methodology in the Supplementary Material.

Unlike *Leon*, our approach does not attempt to identify the true sequence variants Z_w in the process. Although the integration of variant calling and compression of quality scores may have practical advantages, there is the risk of circularity when reproducing the compressed read data.

When it comes to sequencing errors, we also assume the errors on one base are independent from other bases, given their observed quality scores. This assumption is accurate if the quality scores internally reflect the known biases of read error, which for some platforms such as Illumina tend to accumulate at reads ends. In principle, it would be straightforward to include more complex error models that account for the relative biases of specific sequencing platforms.

2.2 Statistical details

We here provide the core statistical elements. For a complete derivation, please consult the Supplementary Material.

We again start by assuming that true source codeword C_w for the read R was known. For brevity, we will denote this source codeword by $S = C_w$.

As mentioned, we are not interested in explicitly recovering the true sample genome Z_w and hence, we collapse both the mutational error and the sequencing error process, resulting in the following conditional distribution, which factorizes across base pairs:

$$\Pr(R|S) = \prod_j \Pr(R_j|S_j) \text{with a per base pair distribution} \tag{2}$$

$$\Pr(R_j|S_j) = \begin{cases} (1-m)(1-\epsilon_j) + \frac{1}{3}m\epsilon_j & \text{if } R_j = S_j \\ \frac{1}{3}(1-m)\epsilon_j + \frac{1}{3}m(1-\epsilon_j) + \frac{2}{9}m\epsilon_j & \text{if } R_j \neq S_j. \end{cases} \tag{3}$$

Given a particular R_j, ϵ_j and S_j, the probability of sequencing error E_j follows from Equation (3) as (see derivation in Supplementary Material):

$$\Pr(E_j|R_j,\,\epsilon_j,\,S_j) \;=\; \begin{cases} \dfrac{m\epsilon_j}{3-3m-3\epsilon_j+4m\epsilon_j} & \text{if } R_j = S_j \\ \dfrac{\epsilon_j(3-m)}{3\epsilon_j+3m-4m\epsilon_j} & \text{if } R_j \neq S_j \end{cases}. \quad (4)$$

Then since S is unknown, we can determine the probability across all possible $S \in (C_1, \ldots, C_N)$ by utilizing the rest of the read and sequencing error rates (i.e. quality scores):

$$\Pr(E_j|R,\,\epsilon,\,C) = \sum_{i=1}^{N} \Pr(E_j|R_j,\,\epsilon_j,\,S_j = C_{ij})\Pr(S = C_i|R,\,\epsilon,\,C). \quad (5)$$

The posterior probability of the source codeword originating from C_i then follows from Bayes theorem:

$$\Pr(S = C_i|R,\,\epsilon,\,C) = \frac{\Pr(R|S = C_i,\,\epsilon,\,C)\Pr(S = C_i|\epsilon,\,C)}{\Pr(R|\epsilon,\,C)}. \quad (6)$$

It is worth noting that the posterior probability of assigning the read to a particular source codeword accounts for the full information within the read and does not factorize across base pairs. Assuming that reads can originate from any codeword in the corpus with equal probability, then $\Pr(S = C_i|\epsilon,\,C) = \frac{1}{N}$. Please see the Supplementary Material for how this approach appropriately handles the case of duplicate k-mers/codewords, as are typically found in real genomes, by allowing duplicate codewords in the corpus. Expanding the denominator in Equation (6) and eliminating common terms, this yields:

$$\Pr(S = C_i|R,\,\epsilon,\,C) = \frac{\Pr(R|S = C_i,\,\epsilon,\,C)}{\sum_b \Pr(R|S = C_b,\,\epsilon,\,C)}. \quad (7)$$

Combining Equation (5) with Equation (7) gives the posterior probability of sequencing error that can be calculated from all the knowns R, ϵ and C:

$$\Pr(E_j|R,\,\epsilon,\,C) = \frac{\sum_i \Pr(E_j|R_j,\,\epsilon_j,\,S_j = C_{ij})\Pr(R|S = C_i,\,\epsilon,\,C)}{\sum_b \Pr(R|S = C_b,\,\epsilon,\,C)}. \quad (8)$$

With ~3.2 billion codewords, this calculation is resource intensive when completed by brute force, making it rather impractical. However, an upper-bound estimate can be found much faster by recognizing that the contribution of reference codewords decreases exponentially according to their Hamming distance from the read. Let L be the set of local indices s.t.

$$\forall i \in L,\, |C_i - R| \,<\, B$$

$$\forall i \notin L,\, |C_i - R| \geq B$$

for some Hamming Distance B (see Methodology in Supplementary Material). This means we separate codewords into local and background codewords according to distance, with the bulk of codewords being in the background set. Because the contribution of each background codeword is small, we can then try to estimate the contribution of background codewords without directly calculating it. Let β be an upper-bound estimate, such that:

$$\beta \geq \sum_{i \notin L} \Pr(R|S = C_i,\,\epsilon,\,C).$$

Let μ denote the probability of a base change (whether due to mutation or sequencing). To obtain an upper-bound estimate β for the background contribution, we assume each of the background codewords is at distance $\boldsymbol{B}$, which produces the worst-case contribution. The average probability of these codewords $\Pr(R|S = C_i)$ is then scaled to encompass all N codewords in the corpus, and not just a subset of background codewords, leading to a value that is typically a very large overestimate and thus conservative in practice (see Supplementary Material):

$$\beta = N\mu^B(1-\mu)^{k-B} \geq \sum_{i \notin L} \Pr(R|S = C_i). \quad (9)$$

The above assumes that μ, the probability of a base change (whether due to mutation or sequencing), is constant for each base position. A more accurate calculation uses a per base probability μ_i where the B mismatches are uniformly distributed over the bases in the read. An approach based on dynamic programming is used to calculate a more accurate background contribution β, and this is what is used for the results presented with this work (see section *Calculating the value of* β in Supplementary Material).

Therefore, an upper-bound estimate of each base's read error can be represented with:

$$\Pr(E_j|R,\,\epsilon,\,C) \;\leq\; \frac{\sum_{i \in L} \Pr(E_j|R_j,\,\epsilon_j,\,S_j = C_{ij})\Pr(R|S = C_i,\,\epsilon,\,C) + \phi_j}{\sum_{b \in L} \Pr(R|S = C_b,\,\epsilon,\,C)}, \quad (10)$$

where $\phi_j \equiv \frac{\epsilon_j(3-m)}{3\epsilon_j+3m-4m\epsilon_j}\beta$.

Let P_j denote the RHS of the inequality in Equation (10). This upper bound P_j yields a lower-bound for the associated quality score for each base, which we encode using the Phred scheme:

$$Q_j = -10\log_{10}P_j,$$

where Q_j is the corresponding Phred quality score for upper bound error probability P_j.

This lower bound is dependent on the depth of the codeword search, where a deeper search can result in a higher value for the bound. In the limiting case, where the search covers all codewords, this bound becomes equal to the actual quality score for the posterior probability. In other words, the usage of the lower bound leads to a conservative approach. The posterior probability of a sequencing error is by construction biased upwards and hence the boosting of quality scores is conservative. In practice, this means that increased depth of the codeword search may lead to additional compressibility at the price of higher computational cost. However, the approximate nature of the algorithm will not falsely boost quality score values. Furthermore, the inequality in Equation (10) is only informative for replacing the observed quality score with a higher value, and is uninformative for replacing it with a lower value.

Note that the use of Hamming-distance for the search means that the presence of an indel in a read may result in large Hamming distances from the reference corpus, yielding no results up to a reasonable search distance. In this case, *GeneCodeq* will be unable to calculate a better posterior probability of sequencing error, leaving the read untouched.

In principle, the approach could potentially be extended by future work to use edit-distance instead, which enables the accounting of insertions and deletions from both mutations as well as sequencing errors. Further details of how such an approach might work is explored in the *Edit Distance* section of the Supplementary Material.

2.3 Codeword search

Rather than calculate the contribution of every single possible codeword in the corpus, the strategy adopted is to search for all the closest codewords to the read, and use a conservative approximation for the likelihood of all the other remaining (background) codewords. Thus, unlike read-alignment against a reference, we are not reconstructing where the read can be positioned in the reference genome, but rather in the distribution of likelihoods for the highest likelihood matches to reference codewords.

To find all codewords with up to M mismatches from the read, the read is divided into $M+1$ slots (see Fig. 2). Based on the Pigeonhole principle, for any codeword up to Hamming distance M away, there must be at least one slot which does not contain a mismatch (i.e. is an exact match). All slots are searched for all matching codewords. If a particular slot, for example, is a n-mer (where $n<k$), a search is made to find all codewords that contain that particular n-mer (this is the Slot LU operation in Fig. 2). The union of searches across the slots is then guaranteed to contain at least all those codewords within the desired Hamming distance M, however it can also contain codewords that are greater than this distance. Filtering is used to discard codewords that are greater than distance M, yielding the desired search results.

The per-slot search can be done by indexing the reference sequence/corpus according to overlapping n-mers (e.g. if the corpus contains ACGGCTAC at some position 1004 then a 6-mer index for that would store position 1004 at index ACGGCT and position 1005 at index CGGCTA and position 1006 at index GGCTAC). For each slot, the set of possible matching codewords is then easily determined by looking up the index for match positions in the corpus. It can be noted that the larger the slot width, the more specific the slot search (Slot LU in Fig. 2), and the fewer the possible candidate codewords that need to be examined.

A basic mechanism for conducting this search is to generate an index of all possible matches for a fixed slot width of 12 bases. Searching with a larger slot width, such as 15 can then be conducted by taking the intersection of the results of two overlapping searches at width 12. Smaller slot width searches would allow greater search distance, enabling further quality scores boosting, but at a computational cost.

For further details of how the Codeword Search is implemented, please see Supplementary Material.

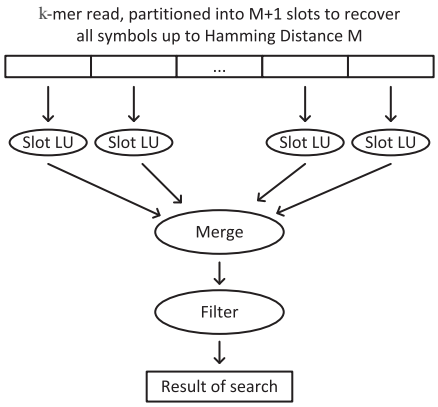

Fig. 2. Codeword search in GeneCodeq. A codeword search of up to Hamming Distance M first divides the *read* into $M+1$ slots, each of which are looked up in an index of the corpus (Slot LU) to find all matches to that slot. The results are merged into a union of candidate matches, and filtered according to distance to return only those up to distance M

2.4 Implementation details and preprocessing

2.4.1 Read preprocessing

Reads are known to frequently have poor quality regions at the head or tail of the read. *GeneCodeq* operates on a trimmed version of the read. The preprocessing step trims the head and tail of the read for quality scores less than or equal to the threshold value (default of 10). This potentially shorter read is thus guaranteed to begin and end with a quality score above the threshold.

2.4.2 Search parameters

A fixed slot width W lets one search a maximum distance $\lfloor \frac{n}{W} - 1 \rfloor$ for read length n. One strategy is to always search to this distance, however it can be faster to try searching with a lower distance first. A simple heuristic for balancing between speed and depth was used in *GeneCodeq*, which is described in detail in the Supplementary Material.

2.4.3 Quality score quantization

If the lower-bound estimate of the quality score (i.e. boosted quality score), given by the posterior probability calculation, is higher than the original quality score, then any intermediate value between the two quality scores is also a better estimate of the probability of sequencing error than the original quality score is. Boosted quality scores may be improved to such an extent that they represent negligible error. Such quality scores can be constrained to a maximum saturation value S (e.g. Phred value 40). Those quality scores that are boosted from a value x to a non-saturation value $y<S$ may instead of recording the value y, use a quantized value $y' = Q(y)$ (e.g. based on the Illumina 8-bin quantization values) provided $y' > x$. This means that the resultant quality scores are conservatively quantized with the boosting, leading to higher compression ratios. For example, if a quality score is boosted from 32 to 43, instead of recording 43, we quantize to 40. In the results here we used the Illumina 8-bin quantization levels, so that whenever a quality score was improved enough to exceed a quantization level, it would be given the highest quantized score instead. This is consistent with our primary aim of improving the compressibility of quality scores, while preserving genotyping accuracy.

2.4.4 Analysis pipeline

The following steps comprise the analysis for each read:

1. trim head and tail of read to remove bases with low quality scores
2. conduct a neighbour search of codewords for the remaining read at a suitable Hamming distance
3. use Equation (10) to determine the posterior probabilities of sequencing error for each base
4. calculate a new quality score per base, quantizing based on Illumina 8-bin quantization levels
5. for each base, if the new quality score is better than original quality score, replace the quality score with the new quality score

2.5 Accounting for known sequence variants in the reference corpus

We can add support for variants on top of a reference corpus by allowing them to be matched directly in addition to the original reference codeword. For example, if a read has a mismatch against the closest codewords from the corpus and any of those mismatches are covered by known variants, then those variants are also temporarily

treated as part of the corpus for the purposes of that read and are added to the search result.

Variant calling tools report many spurious variants (false positives) that may simply be the result of sequencing errors or other uncertainties/problems in the pipeline. To guard against these, we only include common variants that are present with a minimum variant frequency of h in the reference population, set to try to exclude 99.99% of spurious variants. See Supplementary Material for more details.

2.6 Note on indels

Reads with indel variants (base insertion or deletion) that are not present in the corpus are likely to result in large Hamming distances to the rest of the corpus. This makes it unlikely for such reads to be affected by *GeneCodeq*, preserving their original quality scores. For future work, using Edit distance instead of Hamming distance as a metric thus presents an opportunity for further gains in compression. It could allow *GeneCodeq* to also improve genotyping accuracy and compression on sequence data that has a higher indel error rate such as those from Ion Torrent and Pacific Biotech sequencers. Further details of how such an approach might work is explored in the Supplementary Material.

2.7 Analysis of quartz

The *Quartz* approach uses a corpus of 32-mers in order to sparsify the quality scores of matching 32-mers within each read. Although there are similarities to *GeneCodeq*, *Quartz* is based on a heuristic approach and hence the assumptions made are less transparent. Using the same modelling framework as used to derive *GeneCodeq*, one can provide a better assessment of *Quartz* to see under what circumstances it can improve compressibility while preserving genotyping accuracy, and under what circumstances it can degrade genotyping accuracy and fail to improve compressibility. See Supplementary Material for more details.

3 Results

We've used sequences from NA12878 for evaluation purposes due to availability of high quality trio-validated SNP calls to validate genotyping accuracy. Unmapped raw reads were obtained for two datasets:

- SRR622461, a 1000 Genomes Project (1000 Genomes Project Consortium, 2012) dataset with 18.3 gigabases at 5× coverage.
- NA12878J, a public dataset from the Garvan Institute with 122.6 gigabases at 30× coverage (see Supplementary Materials for details).

Resulting variant calls from genotyping were compared to the Illumina Platinum set (http://www.illumina.com/platinumgenomes), which served as a gold standard. In the Supplementary Material we also provide results for additional datasets.

The raw FASTQ files were processed with *GeneCodeq*, *Quartz* (Yu *et al.*, 2015), *R-Block* (Cánovas *et al.*, 2014), *P-Block* (Cánovas *et al.*, 2014), *QVZ* (Malysa *et al.*, 2015), *Qualcomp* (Ochoa *et al.*, 2013), *Leon* (Benoit *et al.*, 2015) and *Illumina 8-bin* quantization (Illumina, 2011). We used the GATK best practices workflow and the samtools recommended workflow, with and without Base-Quality-Score-Recalibration (BQSR). In addition to running *GeneCodeq* in its default mode, we also explored its behaviour with a reference-only corpus (see Section 3.2 for more details), as well as in combination with other approaches—*Illumina 8-bin*, *P-Block* and *R-Block* (see Section 3.4 for more details). For comparison, Quartz was also run with a reference-only corpus, and in combination with *Illumina 8-bin* quantization.

We found that the BQSR stage produces a greater variable change to genotyping accuracy than most of the lossy compression algorithms do alone (pre-BQSR). This makes it difficult to separate and properly assess the impact of lossy compression alone on post-BQSR results. On the lossless dataset, applying BQSR resulted in a significant drop in genotyping accuracy. Combinations of BQSR with lossy compression also resulted in lower genotyping accuracy versus the original across almost all methods. However, since BQSR had a large variable impact, sometimes the lossy compressed post-BQSR results appeared better than the lossless post-BQSR results, skewing relative comparisons. There are strong indications that this is due to flaws with BQSR itself rather than due to the lossy compression. A detailed discussion, included additional results, can be found in the Supplementary Material.

The improvements with *GeneCodeq* are seen across each combination of workflows, for brevity the main results shown here are with the GATK best practices workflow excluding BQSR. Similar results are seen with the samtools recommended workflow, which can be found in the Supplementary Material, along with BQSR results.

Variant calls that resulted from the datasets and produced using these workflows, were ranked by their confidence levels (quality of the variant call) to generate receiver operating characteristic (ROC) curves. This approach avoids choosing a specific quality threshold and hence is well suited to compare alternative methods (see also Yu *et al.*, 2015 where a similar approach is used). However, if two ROC curves have different domains (i.e. a different set of variants for the x-axis) then they are not directly comparable to one another. For the results here and in the supplementary material, the ROC curves and area under the curve (AUC) were calculated with a common domain. As additional quality metrics, we also report precision, recall and F-score metrics. See Supplementary Material for more details on the evaluation approach for all these metrics.

To estimate the impact in compressibility, both the raw and modified quality scores were compressed with *bzip2* (http://www.bzip.org), unless a custom entropy coder was already integrated into the lossy compression (i.e. *QVZ*, *Qualcomp* and *Leon*). We chose *bzip2* because under its default options it consistently yielded superior compression numbers compared to *7zip* (http://www.7-zip.org) and *gzip* (http://www.gzip.org). We note, however, that *7zip* in its PPMd mode can perform even better, and these results are included in the Supplementary Material. This includes significantly better *7zip* re-compressed results for *Leon* which ordinarily uses the poorer performing *zlib* for its internal compression.

Table 1 reports quality score compression rates and genotyping accuracy metrics for these approaches. Figure 3 shows a scatter plot summarizing the best results in Table 1, depicting the genotyping accuracy and compressibility.

3.1 Non-corpus based approaches

The non-corpus based approaches with *Illumina 8-bin*, *P-Block*, *R-Block*, *QVZ* and *Qualcomp* consistently resulted in poorer genotyping performance compared to the raw data. For each of the parameterizable approaches *P-Block*, *R-Block*, *QVZ* and *Qualcomp*, we observed that at higher compression ratios, the genotyping AUC and F-scores degraded, demonstrating a tradeoff between compression and genotyping accuracy. With these non-corpus approaches, we note that AUC is typically lower than when using the raw

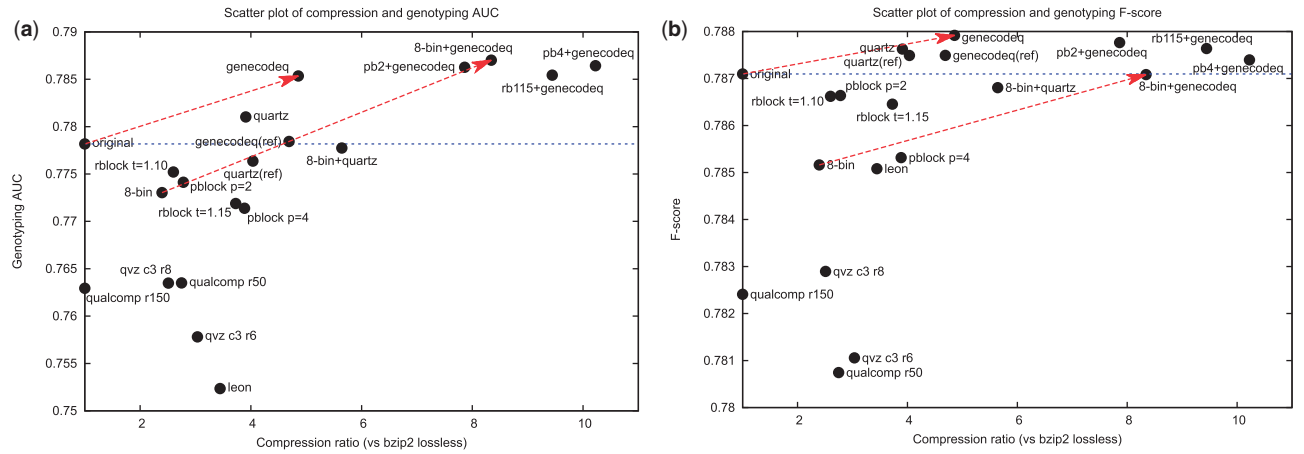

Fig. 3. Performance evaluation for a selection of methods applied to the SRR622461 benchmark dataset. Scatter plots show the tradeoffs between genotyping accuracy and compression ratio. To represent genotyping accuracy, the left plot considers ROC AUC whereas the right plot shows F-score (using the Illumina Platinum set of variants of NA12878 as a gold standard). The horizontal dotted lines correspond to the genotyping accuracy of the raw dataset, with methods above this threshold improving genotyping accuracy. Arrows correspond to the result of applying *GeneCodeq* compression after a particular version of the dataset (raw original dataset or 8-bin Illumina quantized). Notably, while the quantized versions of the dataset results in reduced genotyping accuracy, this loss is recovered when applying *GeneCodeq*, both when considering AUC or F-score metrics. Note PB2, PB4 and RB115 denote P-Block at $p=2$, at $p=4$, and R-Block at $t=1.15$ respectively. When comparing compression ratios, AUC and F-score, higher values are better. For full results, please see Table 1

Table 1. Compression approaches for SRR622461: bits per QS (using *bzip2*) versus genotyping accuracy metrics

Approach	Bits/QS	AUC	F-score	Prec	Recall
Original	1.360	0.7782	0.7871	0.9325	0.6809
Illumina 8-bin	0.568	0.7730	0.7852	0.9341	0.6772
GC (GeneCodeq)	0.280	0.7853	0.7879	0.9323	0.6822
GC w. ref corpus	0.290	0.7784	0.7875	0.9323	0.6816
8-bin + GC	0.163	0.7870	0.7871	0.9339	0.6802
P-Block $p=2$ + GC	0.173	0.7863	0.7878	0.9323	0.6820
P-Block $p=4$ + GC	0.133	0.7864	0.7874	0.9327	0.6813
R-Block 1.15 + GC	0.144	0.7854	0.7876	0.9322	0.6819
Quartz	0.348	0.7810	0.7876	0.9308	0.6826
Quartz (ref)	0.337	0.7764	0.7875	0.9310	0.6823
8-bin + Quartz	0.241	0.7777	0.7868	0.9320	0.6807
P-Block $p=2$	0.523	0.7752	0.7866	0.9325	0.6802
P-Block $p=4$	0.350	0.7714	0.7853	0.9329	0.6781
R-Block $t=1.10$	0.489	0.7741	0.7866	0.9325	0.6802
R-Block $t=1.15$	0.365	0.7719	0.7865	0.9324	0.6800
Leon *	0.395	0.7523	0.7851	0.9332	0.6775
QVZ 0.6 *	0.448	0.7578	0.7811	0.9156	0.6810
QVZ 0.8 *	0.542	0.7635	0.7829	0.9207	0.6810
Qualcomp r50 *	0.495	0.7635	0.7807	0.9155	0.6805
Qualcomp r150	1.360	0.7629	0.7824	0.9192	0.6811
QVZ 0.6 + GC	0.252	0.7669	0.7818	0.9152	0.6824
Qualcomp r50 + GC	0.354	0.7734	0.7812	0.9147	0.6817

AUC is the area under the curve of a ROC plot. The lowest bits/QS is highlighted. Where genotyping metrics are preserved or improved compared to the original, they are highlighted. Where both AUC and F-score genotyping metrics are improved, the approach is also highlighted. Note * denotes custom compression, '(ref)' denotes the use of a reference corpus with no variants. GC denotes GeneCodeq. The 8-bin denotes Illumina 8-binning. Qualcomp was run with one cluster. Genotyping was performed using the Broad Institute recommended pipeline excluding BQSR. Higher AUC, Precision, Recall and F-score measures are better. Lower bits/QS are better.

uncompressed data, whereas other metrics such as Precision for *Illumina 8-bin*, *P-Block*, *R-Block* and Recall for *QVZ* and *Qualcomp* attest a marginal improvement. This suggests that these approaches are at a different tradeoff point between Precision and Recall along the curve, rather than resulting in an actual improvement. This supports using AUC and F-score, that account for these tradeoffs, as the main comparison metrics.

3.2 Impact of the reference corpus on quartz and GeneCodeq

An important experimental consideration is the specific reference corpus that is used for evaluation. For example, *Quartz* utilizes a corpus that is built from reads extracted across samples from the 1000 Genomes Project. It implicitly includes multiple known variants within that corpus that a variant calling pipeline may not be aware of. This extra information can directly improve the actual genotyping accuracy. It is not so surprising that a particular variant has improved calling characteristics when that variant is found to be present in its corpus and its sequencing uncertainty is consequently reduced.

In order to assess the impact of additional variant information on genotyping accuracy and compression, we generated a new corpus for *Quartz* that excludes all variant information. This was built from the set of all 32-mers in the Human Reference Genome (from valid human chromosomes in hs37d5 (Reference Assemblies at https://browser.cghub.ucsc.edu/help/assemblies/). The results can then be compared to *GeneCodeq* when the model is also run with a corpus derived from the same Human Reference Genome, excluding any variant data. As can be seen in Table 1 there is a minor improvement in the ROC AUC for *GeneCodeq (ref)* compared with the raw sequence data (AUC 0.7784 versus 0.7782), whereas *Quartz (ref)* resulted in reduced genotyping accuracy (AUC 0.7764 versus 0.7782). The corresponding ROC curves can be found in Supplementary material. Both approaches result in almost identical improvements in F-score. Thus given the same corpus information, *GeneCodeq* appears to perform slightly better than *Quartz* in genotyping accuracy. *GeneCodeq* also performed a factor of 1.16 better in compressibility than *Quartz* (Table 1).

3.3 Impact of known variants on quartz and GeneCodeq

To explore the benefits of incorporating known sequence variants, we built a corpus of variants from the 1000 Genomes Project (see

Supplementary Material for details) and present the comparison here. Generally, we observed that both *Quartz* and *GeneCodeq* improved genotyping accuracy compared to the raw data when including sequencing variants (Fig. 3). Additionally, we observed that *GeneCodeq* exhibits a larger improvement when compared to *Quartz*.

When variants are included as part of the corpus, we see that *Quartz* and *GeneCodeq* both improved in their AUC, and marginally improved in F-score compared to the reference-only corpus. The change in AUC in both cases is significantly more than when applying the reference-only corpus. This suggests that for both approaches, *Quartz* and *GeneCodeq*, the main factor that contributes to improved genotyping accuracy is information about known sequence variants.

With this dataset, only the *Quartz* and *GeneCodeq* approaches result in better genotyping accuracy, and of these *GeneCodeq* has both superior compression and genotyping accuracy. Looking just at the compression numbers overall, we see that compression for *GeneCodeq* is a factor of 1.24 better than for *Quartz*.

3.4 Combining *GeneCodeq* with non-corpus lossy compression

Since *GeneCodeq* leaves some quality scores untouched, the bulk of the remaining entropy lies in those bases or reads where there is insufficient evidence to saturate them, and indeed some reads are completely untouched as a result. This means that the principle of *GeneCodeq* is complementary to most (non-corpus based) lossy compression schemes, suggesting that combinations of different schemes may offer further improvements.

When applying *GeneCodeq* after Illumina 8-binning (*il8b* + *GeneCodeq*), we observe a markedly low entropy of 0.163 bits per quality score under *bzip2* while retaining a higher AUC and F-score compared to the raw dataset. We also explored combining *Quartz* with Illumina 8-binning, which although improved *Quartz* compression, degraded both AUC and F-score metrics to be worse than that of the raw sequence data.

When combining *GeneCodeq* with the *P-Block* scheme, we see that improvements in AUC and F-score can be retained at even higher compression ratios, with compression of 10:1 for the combination of *P-Block* with $p = 4$ followed by *GeneCodeq*. This combination is a factor of 2.6 better in bits/QS compared to *Quartz* while still having superior AUC, F-score, Precision and Recall values compared to the raw dataset.

3.5 Full sequence data compression

It is instructive to see how well the full sequence data (i.e. both sequence data and quality scores) is compressed using standard sequence compression approaches as a result of this reduction in quality-score entropy. Taking the datasets referred above, and looking at file sizes of standard compression formats for sequence data (namely gzip, BAM and CRAM) we can explore the impact of *GeneCodeq* on these complete datasets. Table 2 shows file sizes for these different approaches, and we observe that the combination of CRAM with *GeneCodeq* is a factor of 3.6–4.6 times smaller than the gzip-compressed FASTQ files of the raw dataset, and under half the size of the CRAM file for the raw dataset. The combination of *GeneCodeq* with *P-Block* demonstrates how further size reduction can be realized.

3.6 Throughput and memory consumption

Single-threaded throughput on an Intel® i7-4790K CPU was 5.57MiB/s with up to 24GiB of memory consumption on a sample human FASTQ file. Since reads are processed independently, the overall throughput can be trivially improved by using a multi-threaded implementation of the *GeneCodeq* algorithm without further memory consumption. For further details please consult the *Throughput and Memory consumption* section of the Supplementary Material.

Table 2. Whole Genome Compression (inclusive of both sequence data and quality scores)

Dataset	FASTQ	FASTQ.gz	BAM	CRAM v2
SRR622461	39.9GiB	8.65GiB	11.0GiB	4.90GiB
w. GeneCodeq	39.9GiB	5.58GiB	7.58GiB	2.39GiB
w. PB4+GeneCodeq	39.9GiB	5.15GiB	7.14GiB	2.00GiB
NA12878J	270GiB	73.7GiB	67.3GiB	36.8GiB
w. GeneCodeq	270GiB	50.5GiB	38.7GiB	15.9GiB

We see significant compression benefits when applying GeneCodeq for gzip-compressed FASTQ, BAM and CRAM formats. Note that the BAM and CRAM formats also include some additional tags from alignment and indel realignment.

4 Discussion

We have described a new method for adjusting base quality scores, which is inspired by coding theory. Our approach leverages available evidence under a Bayesian statistical model to infer posterior estimates of sequencing errors using a known reference corpus and a noisy channel transmission model. This approach requires a suitable corpus either derived from a reference or existing sample(s) in order to apply the statistical model. Given such a corpus, we find that this approach yields significantly higher compression and improves overall genotyping accuracy when compared to previous corpus-based compression methods such as *Quartz*, as well as to all other existing lossy compression approaches. Indeed when combined with the *P-Block* lossy compression approach, for the SRR622461 dataset we see quality score compression ratios of 10:1 whilst preserving the genotyping accuracy of the raw dataset. Other combinations with *GeneCodeq* may also be interesting to explore in future.

Notably, even on datasets that have been *Illumina 8-bin* quantized (also see Supplementary Material), we find that applying *GeneCodeq* can result in improved AUC, Precision and F-score results indicating better genotyping accuracy compared to the raw dataset. This means that *GeneCodeq* is well suited to process and possibly enhance sequencing datasets generated by machines such as Illumina's *HiSeq X* that use this quantization by default. However, we note that, unsurprisingly, the largest gains in genotyping accuracy arise as a result of utilizing a corpus of known variants extracted from the 1000 Genomes Project (carefully excluding the test samples from the corpus set). When used with a corpus containing only reference genome information without variants, we note that genotyping accuracy may still be improved, albeit only slightly, compared to the raw dataset. This means that while the approach preserves genotyping accuracy for variants not in the corpus (such as rare variants), it works significantly better when it knows about potential variants.

The Supplementary Material includes results from other datasets and genotyping pipelines also demonstrating considerable gains in compression whilst preserving genotyping accuracy with *GeneCodeq*.

Acknowledgements

The authors wish to thank the anonymous reviewers for their helpful feedback, as well as Idoia Orchoa-Alvarez and Mikel Hernaez for their assistance

and discussions on variant calling pipelines, and Yun William Yu for his help with reproducing results from *Quartz*.

Funding

This research was conducted under a R&D Smart Grant from Innovate UK.

Conflict of Interest: Daniel Greenfield and Alban Rrustemi were employees of Fonleap Ltd during this research.

References

1000 Genomes Project Consortium (2012) An integrated map of genetic variation from 1,092 human genomes. *Nature*, **491**, 56–65.

Ash,R. (1965). *Information Theory*. *Interscience Tracts in Pure and Applied Mathematics*. Interscience Publishers, New York.

Baker,M. (2010) Next-generation sequencing: adjusting to data overload. *Nat. Methods*, **7**, 495–499.

Barg,A. (1993) At the dawn of the theory of codes. *Math. Intell.*, **15**, 20–26.

Benoit,G. *et al.* (2015) Reference-free compression of high throughput sequencing data with a probabilistic de bruijn graph. *BMC Bioinformatics*, **16**, 288.

Berg,J.S. *et al.* (2011) Deploying whole genome sequencing in clinical practice and public health: meeting the challenge one bin at a time. *Genet. Med.*, **13**, 499–504.

Berger,B. *et al.* (2013) Computational solutions for omics data. *Nat. Rev. Genet.*, **14**, 333–346.

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

Cánovas,R. *et al.* (2014) Lossy compression of quality scores in genomic data. *Bioinformatics*, **30**, 2130–2136.

Cox,A.J. *et al.* (2012) Large-scale compression of genomic sequence databases with the burrows–wheeler transform. *Bioinformatics*, **28**, 1415–1419.

Das,S. and Vikalo,H. (2012) Onlinecall: fast online parameter estimation and base calling for illumina's next-generation sequencing. *Bioinformatics*, **28**, 1677–1683.

DePristo,M.A. *et al.* (2011) A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nat. Genet.*, **43**, 491–498.

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

Fernald,G.H. *et al.* (2011) Bioinformatics challenges for personalized medicine. *Bioinformatics*, **27**, 1741–1748.

Fritz,M.H.Y. *et al.* (2011) Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Res.*, **21**, 734–740.

Grabowski,S. *et al.* (2015) Disk-based compression of data from genome sequencing. *Bioinformatics*, **31**, 1389–1395.

Illumina (2011). Quality scores for next-generation sequencing. *Technical report*. Illumina Inc.[Please provide publisher location for the reference Illumina 2011, 2014.]

Illumina (2014). HiSeq X product sheet, Illumina Inc.

Li,H. (2011) A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics*, **27**, 2987–2993.

Malysa,G. *et al.* (2015) QVZ: lossy compression of quality values. *Bioinformatics*, **31**, 3122.

Ochoa,I. *et al.* (2013) QualComp: a new lossy compressor for quality scores based on rate distortion theory. *BMC Bioinformatics*, **14**, 187.

Stephens,Z.D. *et al.* (2015) Big data: astronomical or genomical? *PLoS Biol.*, **13**, e1002195.

Wetterstrand,K.A. (2015) DNA sequencing costs: data from the NHGRI Genome Sequencing Program (GSP),www.genome.gov/sequencingcosts.

Yu,Y.W. *et al.* (2014). Traversing the k-mer landscape of NGS read datasets for quality score sparsification. In Sharan,R. (ed). *Research in Computational Molecular Biology*. Springer, pp. 385–399

Yu,Y.W. *et al.* (2015) Quality score compression improves genotyping accuracy. *Nat. Biotechnol.*, **33**, 240–243.