Method

Efficient storage of high throughput DNA sequencing data using reference-based compression

Markus Hsi-Yang Fritz, Rasko Leinonen, Guy Cochrane, and Ewan Birney[1]

European Molecular Biology Laboratory's European Bioinformatics Institute (EMBL-EBI), Wellcome Trust Genome Campus, Hinxton, Cambridgeshire CB10 1SD, United Kingdom

Data storage costs have become an appreciable proportion of total cost in the creation and analysis of DNA sequence data. Of particular concern is that the rate of increase in DNA sequencing is significantly outstripping the rate of increase in disk storage capacity. In this paper we present a new reference-based compression method that efficiently compresses DNA sequences for storage. Our approach works for resequencing experiments that target well-studied genomes. We align new sequences to a reference genome and then encode the differences between the new sequence and the reference genome for storage. Our compression method is most efficient when we allow controlled loss of data in the saving of quality information and unaligned sequences. With this new compression method we observe exponential efficiency gains as read lengths increase, and the magnitude of this efficiency gain can be controlled by changing the amount of quality information stored. Our compression method is tunable: The storage of quality scores and unaligned sequences may be adjusted for different experiments to conserve information or to minimize storage costs, and provides one opportunity to address the threat that increasing DNA sequence volumes will overcome our ability to store the sequences.

[Supplemental material is available for this article.]

DNA sequencing has had a major impact on life sciences since the wide scale adoption of the Sanger sequencing method (Sanger et al. 1977). With the advent of array-based pyrosequencing in 2005 (Margulies et al. 2005), followed rapidly by new sequencing-by-synthesis and sequencing-by-ligation techniques, there has been an exponential increase in the generation of DNA sequence data (Stein 2010). Critically, these newer technologies are developing at a much greater rate than was seen for the older technologies, with the current doubling time for DNA sequence output—as measured by output per unit cost—of around 5 mo (Stein 2010). In practical terms this has provided a 1000-fold drop in sequencing costs since 1990 and has made economically possible an increasing number of large data projects. These include the 1000 Genomes Project (The 1000 Genomes Project Consortium 2010), the International Cancer Genome Project (ICGC 2010), and a variety of projects that use DNA sequences as an assay platform, such as the ENCODE project (ENCODE Project Consortium 2004).

There are many challenges in handling this "next generation" of sequence data, from the highly fragmented nature of the shorter reads generated by the new technologies, to storage, analysis, and computational requirements for such large data volumes. The new technologies typically create large direct storage costs as a result of the imaging data that require post-sequencing processing and analysis. These storage costs have sometimes exceeded reagent costs. However, this large excess storage requirement is primarily for "unprocessed" data and large-scale facilities have streamlined their pipelines to discard images at the earliest possible stage and use standard compression methods for the "processed" data in order to reduce storage requirements and associated costs. For example, the Sequence Read Archive (SRA) (Shumway, Cochrane, and Sugawara, 2010), run by the INSDC partners (http://www.insdc.org) as a raw data repository for data from new sequencing platforms, has adopted more efficient compression routines to deliver a final on-disk footprint some 200- to 500-fold reduced relative to that of raw image-containing sequence data.

However, the absolute data volume, whether stored near to the sequencing machines or compressed in archives, is of less concern than the consistent growth in sequencing capacity. Prior to 2005 the rate of increase in sequencing capacity (doubling time around 18 mo) was close to the rate of increase in disk storage capacity (doubling time between 18 and 24 mo) on a per unit cost basis. This meant that, even with conservative engineering techniques, stable budgeting allowed production sequencing facilities and archival databases to match the increase in sequencing rate with the required storage hardware. The drop in sequencing capacity doubling time without a concomitant decrease in storage costs means either that there must be a progressive reduction in the fraction of sequence data that are stored data or, alternatively, a progressive increase in storage budgets. The latter seems an unattractive and unlikely solution.

A number of scientific projects and activities have similar "large data" components, albeit with differing parameters and doubling times. One such high-profile project is the Large Hadron Collider (LHC) at CERN, which will generate an estimated 15 petabytes of data per year when fully active (http://public.web.cern.ch/public/en/LHC/Computing-en.html). LHC scientists are concerned only with a very small subset of "interesting" events recorded against an overwhelming backdrop of unimportant physical events. By processing the LHC data stream in real-time and discarding all uninteresting measurements as noise, vastly reduced data volumes can be achieved for long-term storage. Medical and scientific image processing provide a different large-data set processing example. In many fields, images, and in particular video images, can be collected in almost unlimited quantities, and the application of selective data compression, sensitive to scientific decisions about the value of data components, is essential. A key concept in the compression of images is "controlled loss of precision" in which different compression routines lose precision of the image appropriate for the desired reuse.

[1]**Corresponding author.**
E-mail birney@ebi.ac.uk.
Article published online before print. Article, supplemental material, and publication date are at http://www.genome.org/cgi/doi/10.1101/gr.114819.110. Freely available online through the *Genome Research* Open Access option.

21:734–740 © 2011 by Cold Spring Harbor Laboratory Press; ISSN 1088-9051/11; www.genome.org

There are three approaches to dealing with the growth in sequences submitted to the public databases: (1) A add storage; (2) throw away some data ("triage"); and (3) compress the stored data. These three are not mutually exclusive. A number of arguments for not storing all sequence data have been put forward. These include "don't store the data, just store the physical DNA sample," "discard older data," "discard data from samples that can be regenerated," and "don't store the raw data, store only the analysis output, such as the differences." The last parallels the approaches taken by the LHC and image processing work in that it saves only informative data.

What is common to most of these suggestions is the implied ability to resequence any sample at any given point of time, thus obviating the need to store the available sequence data electronically. Given the large scale of international projects, it is clear that at least in the short term the large volumes of data generated by distributed production centers will have to be kept available and ready for aggregation and analysis. One might argue that after analysis, the raw data should no longer be stored electronically. However, many of these projects run for several years and, even after they have been completed, further follow-up projects arise that require easy access to the data. In addition, the exponential decrease in storage costs means that the cost for storing any particular data set is heavily weighted to the early years of its storage. Given the large worldwide investment in variation and cancer sequencing, it seems inappropriate to limit the reanalysis of sequencing data over the forthcoming decade for the sake of data storage costs.

An additional major concern is the feasibility of storing or reacquiring samples. Many clinical samples have low DNA content, meaning that the samples cannot be stored and freely distributed in the future. Where samples are nonrenewable, as it is for many cancer and other human samples, future availability of sequence data may only be guaranteed through permanent electronic archives. Even when samples are renewable, the cost of acquiring them can be considerable, especially for clinical samples. Finally, long-term storage and distribution of DNA samples worldwide is a complex operation, with considerable costs in physical storage, shipping, and end-point sequencing.

Another option for dealing with increasing sequencing data is compression. In its natural representation as a string of characters, a DNA sequence can be compressed using generic approaches, for which a rich literature exists. Beyond this, though, compressibility of DNA sequence can take advantage of certain biological characteristics, such as, for example, repeat content and relationship to existing sequence. Prior work on DNA sequence compression, reviewed in Matsumoto et al. (2000) has yielded DNACompress, exploiting redundancy within DNA sequences due to approximate repeats (Chen et al. 2002), and DNAzip (Christley et al. 2009), introducing the important idea of only storing differences to a reference sequence, but in this case for storing an entire, assembled genome as a series of differences.

In this paper we describe a new and more efficient method for raw DNA sequence data storage, reference-based compression, that nevertheless does have some aspects of "triage" in that some information is lost. This is closest to the "don't store the raw data, store only the analysis output, such as the differences" intuition cited above, but changes the challenge from appropriate scientific analysis to appropriate compression. We do not aim to perform a certain type of analysis under our scheme, nor do we constrain ourselves to current resources, such as any one current reference genome. A key feature of reference-based compression is that its performance improves exponentially with respect to areas of growth in DNA sequencing technology, so it can mitigate the mismatch between DNA sequencing capacity and disk technology growth rates as well as reducing the absolute quantity of stored data. Efficient reference-based compression requires a controlled loss of precision in terms of the sequencing information stored. Usefully, the trade-off between loss of precision and storage efficiency cost can be quantified and presented to the scientific community for discussion of where it is most appropriate to maintain higher precision. Our goal is to bring about a scenario where, for any reasonable sustained improvement in DNA sequencing technology, a constant cost storage solution is available with a well understood loss of precision appropriate for future reuse.

Results

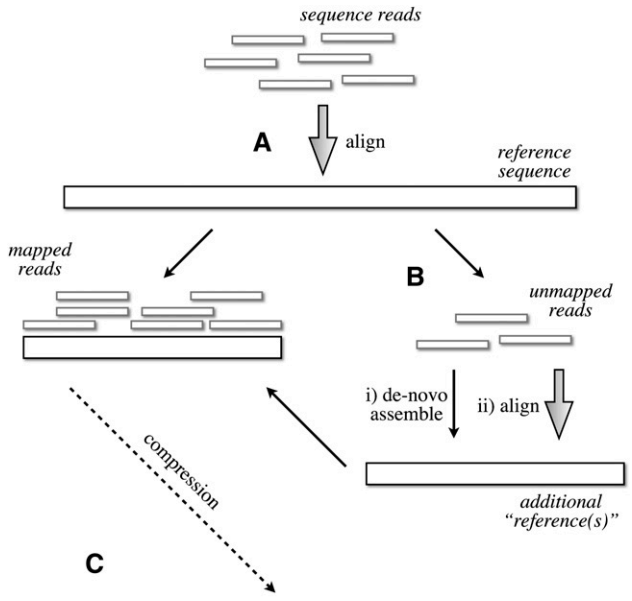
Lossless reference-based compression of DNA sequence

The basis of our method is the efficient storage of information that is identical or near-identical to input "reference" sequences. The key feature of our method is that new sequences identical to the reference have minimal impact on storage regardless of their length or depth of sequencing coverage. Additionally, we use the reference genome strictly as a compression framework, and do not require any biological correctness for the reference. We first focus on lossless compression of the base sequence, then discuss information that cannot easily be represented by mapping to the reference genome (such as unaligned reads), and finally cover appropriate controlled loss of precision on continuous information (such as qualities). The overall outline of our method is shown in Figure 1.

To store information efficiently using a known reference sequence as a compression framework, we need to take advantage of the fact that most reads in a resequencing run match the sequence perfectly or near-perfectly. In our approach, we take a mapping of the reads against the reference sequence (Fig. 1B) and summarize both mapping properties and deviations from the reference in an efficient manner (Fig. 1C). Much of the efficiency of our method relies on appropriate use of Golomb codes (Golomb 1966) that are now a standard compression technique for storing integer values.

The method works as follows:

1. We store the lookup position of each read on the reference sequence as the integer position on this compression framework. The lengths of reads are compressed using Huffman coding (Huffman 1952). For constant read length across a file, we store the length once.
2. We assume that read order does not have any meaning for any given data set and reorder the reads with respect to the integer position determined in (1), allowing efficient relative encoding of the positions by storing the differences between successive values instead of the absolute values. Given coverage and read length, a Golomb code is chosen that is parameterized on the expected distance between reads.
3. Any variation from the reference is stored as an offset relative to the integer position from (1), along with the base identities (for substitutions and insertions) or the length (for deletions). Offset values, again, are assigned a Golomb code parameterized on the distance between successive variation positions.
4. The read pair information is stored as an unsigned rank offset from the positionally lower read to the positionally higher read, with three bits present to indicate relative orientation of reads and the strands from which they were sequenced. Again, Golomb encoding, parameterized on expected rank offsets, is used.



Position	Strand	Substitutions	Insertions	Deletions
4	+	4-G	*none*	5-3
6	+	*none*	*none*	*none*
7	+	*none*	8-AT 4-C	*none*

Figure 1. Schematic of the compression technique. (*A*) Reads are first aligned to an established reference. (*B*) Unaligned reads are then pooled to create a specific "compression framework" for this data set. (*C*) The base pair information is then stored using specific offsets of reads on the reference, with substitutions, insertions, or deletions encoded in separate data structures.

Figure 2 shows the behavior of this compression technique for simulated data sets at a variety of different read lengths, error rates, and coverage, for both unpaired and paired data sets. The worst case scenario uses 0.66 bits/base pair, and the best case uses 0.02 bits/base pair. These compare very favorably with efficient bzip2 compression for raw base DNA of about 1 bit/base pair, and are considerably more efficient than BAM-based storage (Li et al. 2009). We observe an exponential increase in efficiency of storage with higher read lengths (at any given coverage). Similarly, there is a progressive increase in efficiency in storage as coverage increases between 1× and 30×, but this is eventually saturated. At different error, coverage, and read length costs, the different components take differing proportions of the storage. Figure 3 shows the breakdown in storage costs for three parameterizations, and shows that the relative proportion of read information to variant information changes considerably for different coverage and error rate scenarios.

Table 1 shows two real data sets with their alignable portion compressed using this method, while Supplemental Table 1 shows error rates for the same two data sets. The compression rates are close to the simulated data sets, and are between fivefold to 54-fold smaller than compressed FASTA or BAM, respectively. Even if one distributed the reference sequence along with the data set each time, the reference bases contribute a small amount to the total storage. A key question is whether to keep soft clipped information or not. Clipping is the process of removing the sequencing chemistry adaptors from the sequence. Soft clipping is the usual standard for sequencing in which the adaptors are identified, but not removed explicitly, allowing future analysis routines to change the clip point. Although soft clipped DNA information might add a small percentage of bases onto each read, for this compression method it creates an edit structure, and thus a storage overhead, on nearly every read. Interestingly in the bacterial set, because the soft clipped reads are of constant length, the soft clipped data compresses better than hard clipped. For the human 1000 genomes case, with a mixture of read lengths and considerable variation in soft clip length, the hard clipped sequences compress better.

Efficient compression frameworks for unmapped reads

While mapped sequence is efficiently stored using the reference sequence compression framework, unmapped sequences cannot be compressed in this manner and may dominate storage cost. Usually between 10% and 40% of reads remain unmapped to traditional references; the exact figure depends upon the experimental and mapping strategies. Furthermore, for some experiments, such as short RNA sequencing, the unmapped proportion can rise to 60%–70% of reads. For the highest efficiency of compression, we aim to develop a compression framework in which as many reads as possible can be placed. Here the distinction between a biologically correct reference sequence and a "useful" compression framework becomes more pronounced. Critically, a compression framework need not be biologically correct or interpretable; it simply needs to provide efficient compression.

Our strategy is to pool unmapped reads from a variety of "similar" experiments, e.g., those using the same individual or species, and then discover runs of sequences that are present across the experiments. These sequences then serve as a secondary compression framework upon which we can provide efficient storage. This is similar but not identical to the task of finding the "correct" assembled sequence from the fragmented reads from genome shotgun sequencing. Here, we have used a large-scale de Bruijn graph framework, Cortex (Z Iqbal and M Caccamo, in prep.), to find regions of contiguous sequence in unaligned reads. From a sample of unaligned reads from the NA12878 genome, which is the most sequenced human individual from the 1000 Genomes panel, we observe that a novel 45 MB of contiguous sequences could be generated, to which 14% of the previously unaligned reads could be mapped. The reads compressed to 0.26 bits/base if stored without read pair information. Finally, we matched the remaining unaligned reads to both a further human sequence (the Venter sequence) and all bacterial and viral sequences (assuming potential laboratory or sample contamination), with the understanding that the proportion of reads that mapped at this step could be compressed against these further references. In this step, an additional 3% of reads could be compressed at a similar compression rate. However, this left a considerable number of reads (83%) that cannot be easily compressed. Compressing the raw sequence of these reads yields a compression rate of 2.12 bits/base, somewhat higher than average sequencing reads. Furthermore, the reads do not show dense, repeated kmer frequencies; 57% of 21 mers were unique in the data set and <1% showed >10 frequencies. This strongly suggests that there is not a systematic source of

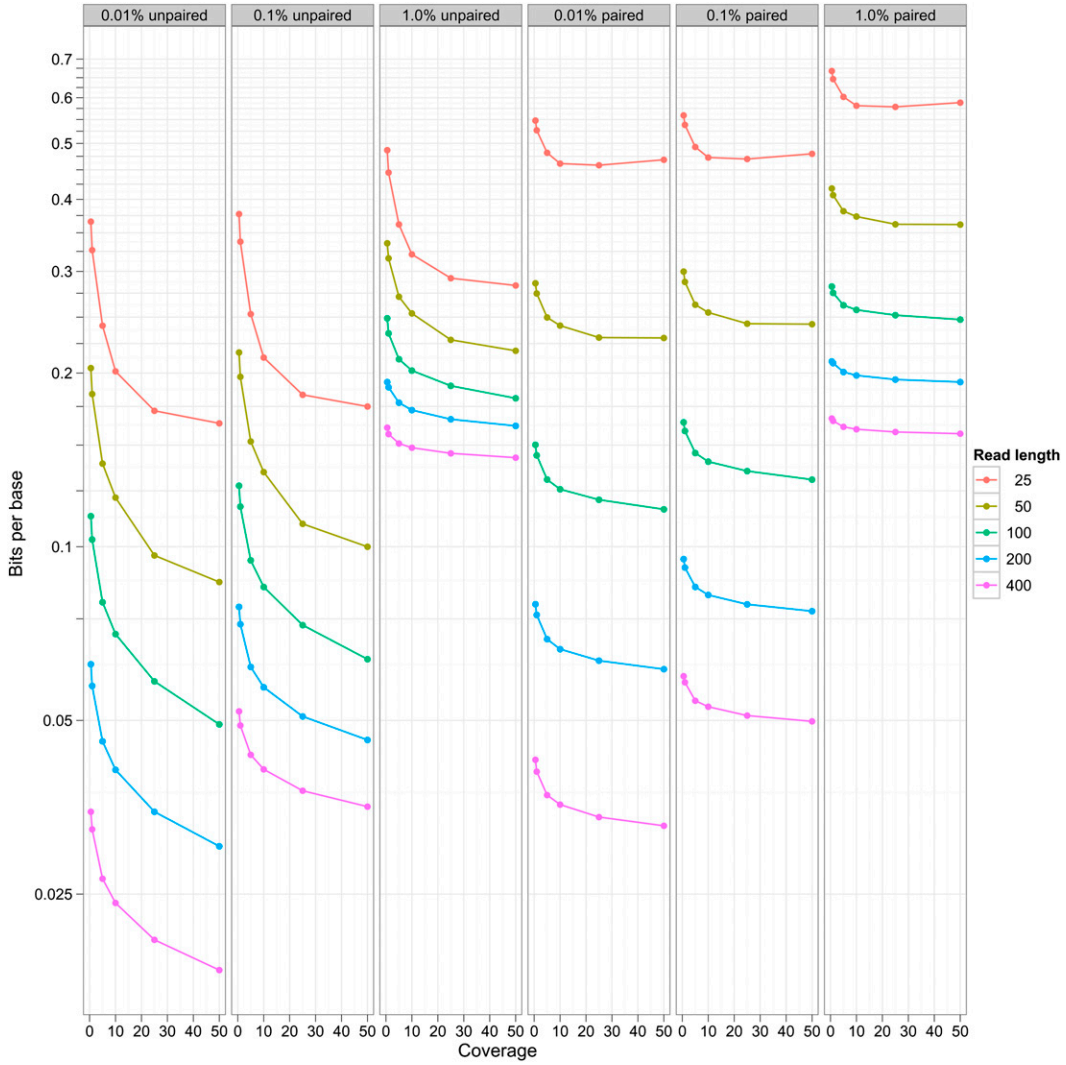


Figure 2. Compression efficiency for simulated data sets. The plot shows storage of DNA sequence expressed as a bits/base stored on the *y*-axis (log scale) vs. coverage of data sets (*x*-axis) for different read lengths (the different colors) after reference-based compression. The different columns indicate different simulated error rates (0.01%, 0.1%, 1.0%). The *left* three panels show this for unpaired data, the *right* three for paired data.

sequence at high coverage in the remaining unaligned sequences. We comment on how these unalignable sequences may be treated in the Discussion section below.

Storage of continuous values related to base quality

So far we have focused on the development of lossless base sequence compression. Also important for a number of applications of sequencing data is per-base continuous value information. While previously this was commonly raw intensity data, the recent trend amongst large projects has been to recommend not preserving these data due to the lack of observed reuse of this information, but rather to preserve phred or log-scale base call quality values (Ewing and Green 1998). Although these phred scores are reasonably compressible due to their limited integer range and very biased composition (in our hands, real data sets have about 3.5 bits/quality score after Huffman-based compression), this is still far higher than the lossless base pair compression described above. Instead we have implemented a different scheme in which all the quality scores of positions showing variation are stored and, in addition, a user-defined percentage of quality positions that are identical to reference are stored. In both cases, the qualities are then further compressed using a Huffman-based code.

A scientific decision regarding the scope and best use of this "quality budget" for any given data set can then ensue. For example, it may be sensible to have all reads mapping to known copy number variation regions or sites with ambiguous SNP calls associated with rich quality information. For this paper we have chosen a simple scheme to store the lowest set of qualities of bases that are identical to the reference. This scientific decision about how to use the quality budget will be specific to at least classes of data, and potentially specific data sets. By allowing the quality budget to be variable, this compression can place more value on some data sets than others, and there can be a progressive reduction of quality budgets over time to mitigate the risk of discarding important differences in quality values early on in any practical implementation process.

Figure 4 shows the overall storage cost at different quality budgets and different read lengths. Unsurprisingly, decreasing the

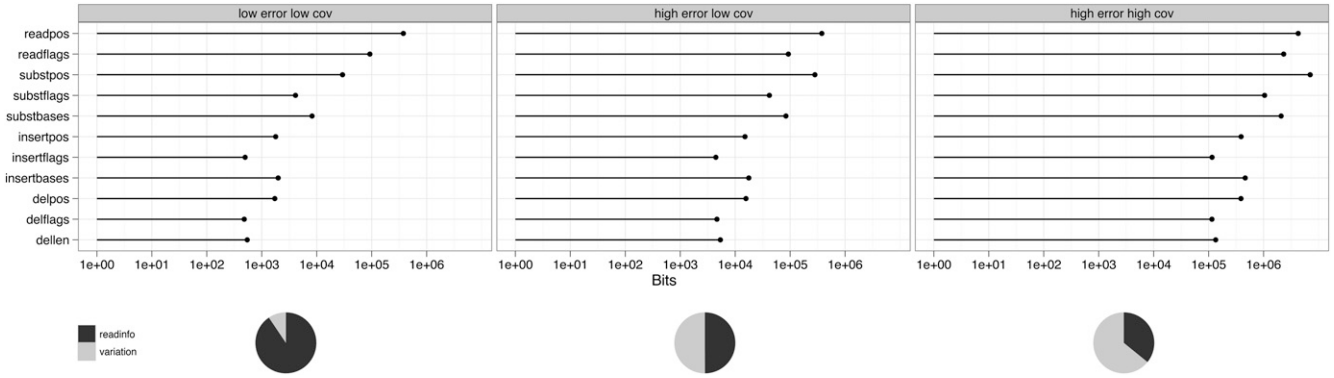


Figure 3. Storage components for three parameterizations of simulated data: 0.1% error and 1× coverage (*left* panel), 1% error and 1× coverage (*middle*), and 1% error and 25× coverage (*right*). *readpos* and *readflags* is the storage of the read positions and read flags (strand, exact match), respectively. Variation storage for substitutions (*subst*), insertions (*insert*), and deletions (*del*) is split into positional information (*pos*), flags (*flags*), and bases (*bases*, for substitutions and insertions) or length (*len*, for deletions). The pie charts show overall storage requirements, where *readinfo* sums over read positions and read flags, and *variation* is the sum over all variation storage components.

quality budget leads to more compressible data. In addition, the budget level affects the efficiency of compression at increasing read lengths; i.e., for a lower quality budget, not only is there an absolute shift in the compressibility of the data, but also the data becomes proportionally more compressible at longer lengths. This is because much of the benefit of the compression occurs when a read matches identically to the reference at high quality, and so the compression is driven by the proportion of reads that require no additional data structures. As our current simulations assume a random distribution of where quality information is used, any heterogeneity in errors or decisions on where to use quality information will improve the compressibility of the data. Table 1 shows the impact of using a 2% quality budget on storage on two real data sets with a 10- to 30-fold compression to compressed FASTQ or BAM, respectively.

Discussion

Although the challenge of storing DNA sequencing information is manageable in the near term, it is important for the biological community to consider the implications of sustained technology improvements in DNA sequencing and to plan for better data compression methods—potentially with controlled loss of precision—before there is a critical mismatch between data generation and storage. We present here a novel compression method that creates a more explicit balance between storage cost and the precision at which data is stored. This compression framework enables us to evaluate the impact of discarding different components of the sequencing information and thereby to make informed choices that choose precision at the expense of storage costs, loss of precision to reduce storage costs, or some value in between.

Our method takes advantage of a number of standard compression techniques on a specific data structure, namely alignment of reads to a reference sequence, and relatively standard compression techniques. It is quite possible that other compression techniques could provide even more efficiency. For example, the observation that once one has decided to store quality information on a read, the quality information for a second base is less expensive to store than the first, may be leveraged in some way. However, our method already achieves 10- to 30-fold better compression than standard approaches with real data, and this is expected to rise to 100-fold better with longer read lengths. In particular, our compression technique becomes more efficient for longer read lengths, and the increase in efficiency is a function of the "quality budget" we allow for each data set. As increase in read length accounts for about half of the gains in productivity of the DNA sequence technology, this suggests we can set a quality budget level that allows a matching of the increase of DNA sequence output to the increase of disk storage capacity, providing a sustainable model for electronic DNA sequence storage. As much of the potential increase in output in the so-called "third generation" technologies is currently around read length, our compression system is well

Table 1. Compression efficiencies for two real data sets.

Compression method	NA12878 chrom20	*Pseudomonas syringae* pathovar syringae B728a
Raw FASTQ	19.96	21.04
Raw FASTA	11.45	12.37
Bzip2 FASTQ	6.64	4.77
Bzip2 FASTA	1.84	1.66
Bzip2 Raw sequence	1.09	0.99
BAM	17.48	7.02
Bzip2 BAM	17.55	7.03
Hard clipped, reference-based, sequence only	0.32 (0.37)	0.21 (0.28)
Bzip2 hard clipped, reference-based, sequence only	0.30 (0.35)	0.16 (0.22)
Soft clipped, reference-based, sequence only	0.41 (0.46)	0.19 (0.25)
Bzip2 soft clipped, reference-based, sequence only	0.37 (0.42)	0.16 (0.22)
Hard clipped, reference-based, 2% quality budget	0.59 (0.64)	0.47 (0.53)
Bzip2 hard clipped, reference-based, 2% quality budget	0.56 (0.61)	0.39 (0.45)
Soft clipped, reference-based, 2% quality budget	0.79 (0.84)	0.44 (0.50)
Bzip2 soft clipped, reference-based, 2% quality budget	0.69 (0.74)	0.39 (0.45)

Left column shows data format, middle and right columns show the size of each of two data sets, one human and one bacterial (see Methods for sources), in bits/base. Under all conditions our reference-based compression method is significantly more efficient than standard compression techniques. The bracketed numbers shows bits/base when a bzip2-compressed copy of the reference sequence is stored with the data set.

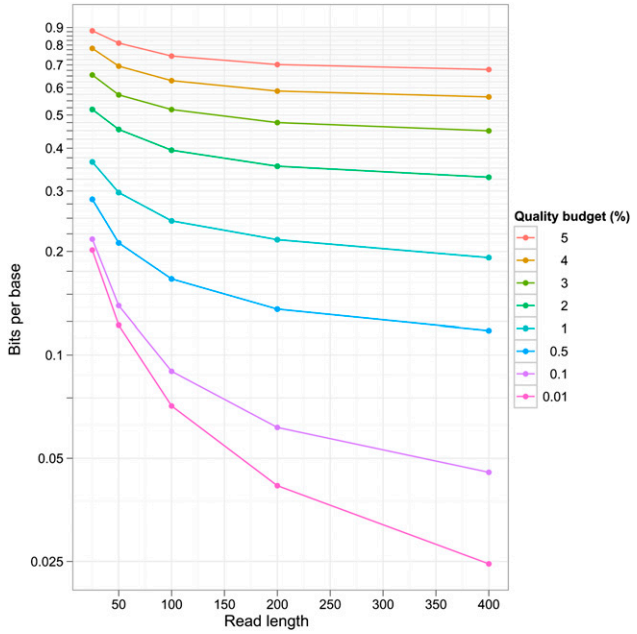


Figure 4. Storage costs for different quality budgets. The plot shows the change in storage cost (expressed as bits/base, including quality information, *y*-axis) with respect to read length for different quality budgets for a fixed coverage (10×) simulated data set. Note that not only do lower quality budgets compress better, but also the compression efficiency improves proportionally more at lower quality budgets for higher read lengths. Quality budgets are the percentage of base pairs in the data set for which quality scores are retained.

placed to increase in efficiency in step with improvements in sequence technology throughput. However, it is certainly possible that single molecule technology will have higher or at the very least, different, error profiles, and it may be a more complex landscape in which to work out an optimal "quality budget."

Recently Daily et al. (2010) published a similar compression scheme for DNA sequences. Their general approach, as well, is using a reference sequence as lookup key and storing position, length, and edit structure of reads, applying relative encoding and efficient integer codes. While these authors do explore different integer codes, they don't consider the handling of unaligned data, paired-end reads, and quality scores. For a high coverage, human data set, they achieve a similar compression rate (~0.35 bits per base) to our method, although they don't consider indels at all and restrict to a maximum of two substitutions per read.

We have focused our efforts on compressing whole genome shotgun information. This is because such data will be the largest component of sequence archive growth for the next decade, mainly because of the high sample numbers per cancer (~500), high coverage requirements (30× on tumor and normal samples), and high scientific interest (at least 10 declared projects so far) in cancer resequencing (ICGC 2010). In addition, whole genome sequencing for medical genetics purposes is likely to be the next largest segment of growth. Although there may be a large number of RNA-seq, ChIP-seq, or other more basic biology focused experiments, they currently occupy <20% of the SRA archive by bases, and are likely to occupy progressively less due to the growth in cancer and medical sequencing. Our method can also be applied to RNA-seq and ChIP-seq data, though careful attention must be paid to such aspects as unaligned data. Taking RNA-seq data, for example, even if the current transcript sequences augmented the reference sequence, thus capturing known splice junction reads efficiently, it is expected that new RNA-seq experiments would discover new junction reads potentially at low coverage. Other data types, such as shotgun data for de novo genome assembly in previously unstudied organisms and environmental sequencing projects aimed at understanding genomics at the community level, will bring further challenges in relation to unaligned reads.

Our reference-based method provides efficient lossless compression for reads that align to the reference sequence with few, or no, differences, as efficiently as 0.02 bits/base. This corresponds to most people's intuition that one needs to "store only the analysis output, such as the differences" of a sequencing run. However, there are still critical decisions about what information to discard. One is to what extent unaligned reads should be discarded. With an individual- or species-level creation of specific "compression frameworks" an additional 15% of unaligned reads can be efficiently stored. A critical scientific question is whether the remaining unaligned reads should be stored or discarded, and if they should be stored, can some other compression approach be used to decrease the storage cost. It seems likely that this decision would be different for different data sets; for example, cancer genomes compared to their normal controls. A further critical decision is how large the "quality" budget should be, and how this budget should be distributed. A practical approach is to adjust the quality budget to balance the disk improvement rate with improvements in DNA sequencing technologies. As with unaligned reads, such decisions should ideally be made at the level of the data set and, as we need to control the overall storage cost, we might increase the quality budget on some samples (such as tumors) at the expense of other samples (such as their matched normals). Importantly, this framework is flexible with regard to quality budgets, so early implementations can be generous in their quality budget while discussions with analyst and user communities about the impact of different quality budgets in production pipelines are undertaken.

In this scheme, the quality budget is bounded by the number of differences between reads and the reference genome to ensure the correct base sequence is returned; in turn, this difference rate is likely to be dominated by sequencing error rates rather than biologically interesting phenomena. In scenarios where there is a large systematic biological difference (e.g., a very high density of SNPs), one can imagine providing a condensed edit structure for the reference sequence to improve compressibility. Achieving quality budgets lower than the sequencing error rate will be more challenging. Potentially one could imagine preprocessing reads to recognize "clear" sequencing errors, by a combination of the weight of evidence from other reads and quality values, and "fix" such errors to be more consistent. However, such manipulations are a more fundamental change of the experimental data, and would require considerable discussion of the consequence for downstream analysis before implementation.

We have implemented our algorithms in a prototype written in the Python programming language. The source files are available at http://www.ebi.ac.uk/~markus/mzip. This prototype works with BAM-based input, but it has been deliberately written to experiment with different component compression schemes; as such its run time is slow, with around 2 h taken for compression (see Methods); however, we are confident that this can be improved at least fivefold, if not more. The current on-disk format in our prototype is both streamable and indexable by genomic position, so slice functionality, as provided in BAM format associated tools (Li et al. 2009), would be feasible. Practical implementation of these ideas will be more productive inside of specific tool kits, such as BAM or SRA, and

we are actively working on this integration. There are other complex practical aspects for using this scheme, such as who is responsible for providing the reference alignment, how reference sequences are stored and verified in different locations, and how additional assembled references would be computed. Again, a practical implementation would have to fit into the current data flow from sequencing groups into the archive sites. However, many of these questions are relevant to current discussions on how to handle alignment submission in general. By asking for submission as an alignment, itself a very common part of local analysis, some of the complex compute stages would be distributed to the submitters rather than the central resources. In summary, although there are many individual technical details to resolve before this compression routine becomes part of production processes, there is no critical piece that is not solvable with appropriate effort.

DNA sequence has become the first molecular data for which the cost of storage has become a significant proportion of the overall cost of generation and analysis. In common with other technologies with a high storage to generation cost, in particular imaging, we must consider carefully what information, and at what level of precision, is worth storing. As with imaging, intelligent decisions about the precision of the information stored allow efficient, loss compression algorithms that retain key components of the information for future analysis. By creating efficient compression algorithms we can at least postpone any harder decision about whether specific data sets should be deleted in their entirety, and at best not limit future analysis of DNA data based on constraints on current storage budgets. It seems likely that similar decisions will eventually be necessary for other biological data, from proteomics to metabolomics, as well as for the compression algorithms used for biological and clinical images.

Methods

Compression

For every read we store its starting position with respect to the reference, its strand, and a flag indicating whether the read matches to the reference perfectly. Positions are relative encoded (i.e., the difference between successive positions is stored rather than the absolute value) and stored as a Golomb code, resulting in a variable length code. Strand and match flags require one bit each. In the case of a nonperfect match, we store a list of variations. Every variation is stored as its position on the read, the variation type (substitution, insertion, deletion), and additional information (the base change in the case of a substitution, the inserted bases, or the length of the deletion). Again, positions are relative and Golomb encoded. The variation type (substitution, insertion, or deletion) is encoded in 1 or 2 bits. Given the reference base, any substitution to A, C, G, T, or N (other than the reference base) can be encoded in 2 bits. For quality budgets, where identical bases are present we store a bit indicating whether the base is the same as the reference, optionally followed by an encoding of the change of base as explained before. Inserted bases are encoded in 2 or 3 bits (in truncated binary encoding, similar to the variation type). Deletion lengths are currently Gamma encoded (Elias 1975). If different read lengths are present across the input file, we store the length Huffman-encoded, and store the length values and their counts at the beginning of the file. For paired-end reads, we store a bit indicating whether a read is paired, the strand from which the read was sequenced, the relative orientation of the mate, and the relative offset from the position of the lower to the higher read. For the latter, a Golomb code is used. More details are given in the supplemental material.

The code is available from http://www.ebi.ac.uk/~markus/mzip/.

Files: Table 1

The file NA12878.chrom20.ILLUMINA.bwa.CEU.high_coverage.20100311.bam was downloaded from ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data/NA12878/alignment. The reference genome was obtained from http://hgdownload.cse.ucsc.edu/goldenPath/hg19/chromosomes.

The *Pseudomonas syringae* files ERR005143_1.fastq and ERR005143_2.fastq were downloaded from http://www.ebi.ac.uk/ena/data/view/ERR005143, the reference genome file from http://www.ncbi.nlm.nih.gov/nuccore/NC_007005. The read files were concatenated and mapped to the reference using BWA 0.5.8 (Li and Durbin 2009) with default options.

Running times

Supplemental Table 2 shows the running time for compressing the two experimental data sets. BAM compression is currently implemented as two steps: i) conversion of BAM into lists of read information and variation; and ii) the compression of these data.

Acknowledgments

We thank James Bonfield, Eugene Yaschenko, Richard Durbin, Paul Flicek, Richa Agarwal, and David Lipman for thoughtful comments and discussion. Charles Cook provided invaluable proofreading of the manuscript. The authors were funded by EMBL core funds. The SRA archive is supported by the Wellcome Trust.

References

The 1000 Genomes Project Consortium. 2010. A map of human genome variation from population-scale sequencing. *Nature* **467:** 1061–1073.
Chen X, Li M, Ma B, Tromp J. 2002. DNACompress: Fast and effective DNA sequence compression. *Bioinformatics* **18:** 1696–1698.
Christley S, Lu Y, Li C, Xie X. 2009. Human genomes as email attachments. *Bioinformatics* **25:** 274–275.
Daily K, Rigor P, Christley S, Xie X, Baldi P. 2010. Data structures and compression algorithms for high-throughput sequencing technologies. *BMC Bioinformatics* **11:** 514. doi: 10.1186/1471-2105-11-514.
Elias P. 1975. Universal codeword sets and representations of the integers. *IEEE Trans Inf Theory* **21:** 194–203.
ENCODE Project Consortium. 2004. The ENCODE (ENCyclopedia Of DNA Elements) Project. *Science* **306:** 636–640.
Ewing B, Green P. 1998. Base-calling of automated sequencer traces using phred. II. Error probabilities. *Genome Res* **8:** 186–194.
Golomb SW. 1966. Run-length encodings. *IEEE Trans Inf Theory* **12:** 399–401.
Huffman D. 1952. A method for the construction of minimum redundancy codes. In *Proceedings of the I.R.E.*, Vol. 40, (No. 9) , pp. 1098–1101. Massachusetts Institute of Technology, Cambridge, MA.
ICGC (The International Cancer Genome Consortium). 2010. International network of cancer genome projects. *Nature* **464:** 993–998.
Li H, Durbin R. 2009. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* **25:** 1754–1760.
Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R. 2009. The Sequence Alignment/Map format and SAMtools. *Bioinformatics* **25:** 2078–2079.
Margulies M, Egholm M, Altman WE, Attiya S, Bader JS, Bemben LA, Berka J, Braverman MS, Chen Y-J, Chen Z, et al. 2005. Genome sequencing in microfabricated high-density picolitre reactors. *Nature* **437:** 376–380.
Matsumoto T, Sadakane K, Imai H. 2000. Biological sequence compression algorithms. *Genome Inform Ser Workshop Genome Inform* **11:** 43–52.
Sanger F, Nicklen S, Coulson AR. 1977. DNA sequencing with chain-terminating inhibitors. *Proc Natl Acad Sci* **74:** 5463–5467.
Shumway M, Cochrane G, Sugawara H. 2010. Archiving next generation sequencing data. *Nucleic Acids Res* **38:** D870–D871.
Stein LD. 2010. The case for cloud computing in genome informatics. *Genome Biol* **11:** 207. doi: 10.1186/gb-2010-11-5-207.

Received September 2, 2010; accepted in revised form January 13, 2011.

Efficient storage of high throughput DNA sequencing data using reference-based compression

Markus Hsi-Yang Fritz, Rasko Leinonen, Guy Cochrane, et al.

Genome Res. 2011 21: 734-740 originally published online January 18, 2011
Access the most recent version at doi:10.1101/gr.114819.110