1 Background introduction for some classic encoding techniques

**Huffman coding:** Huffman coding was developed by David A. Huffman. It is a popular entropy encoding algorithm used for lossless data compression. The key is to encode a source symbol (such as a character in a file) by constructing a variable-length code table in a particular way based on the estimated probability of occurrence for each possible value of the source symbol.

Huffman code expresses the most common source symbols using shorter strings of bits than that are used for less common source symbols. The resulting code is known as “prefix-free codes”, which means the bit string representing some particular symbol never serves a prefix of the bit string to represent any other symbol. Although Huffman's original algorithm is optimal for a symbol-by-symbol coding (i.e., a stream of unrelated symbols) with a known input probability distribution, it has no optimal guarantee when the symbol-by-symbol restriction is dropped, or when the probability mass functions are unknown or violate the independent and identically distributed (i.i.d.) condition.

**Run-length Encoding:** Run-length encoding (RLE) is usually used for the sequences, for which the same data value occurs consecutively. It stores the runs of data as a single data value and count, rather than the original run. It performs well for compressing files that contain many runs of same data value.

**Delta Encoding:** Delta encoding is a scheme that stores or transmits data in the form of differences between sequential data rather than complete files. It is also known as data differencing. It can significantly reduce data redundancy when encoding file where there is a small difference between consecutive data.

**Dictionary coding - LZW:** Dictionary coding, also known as a substitution coding, is a class of lossless data compression algorithms, which operate by searching for matches between the text to be compressed and a set of strings contained in a data structure (called the 'dictionary') maintained by the encoder. When the encoder finds such a match, it substitutes a reference to the string’s position in the data structure.

LZW is a representative of dictionary coding published by Welch in 1984 as an improved implementation.
of the LZ78 algorithm published by Lempel and Ziv in 1978. At each stage in compression, LZW gathers input bytes into a sequence until the next character would make a sequence for which there is no code yet in the dictionary. The code for the sequence (without that character) is added to the output, and a new code (for the sequence with that character) is added to the dictionary.

**Prediction by partial matching:** Prediction by partial matching (PPM) is an adaptive statistical data compression technique based on context modeling and prediction. PPM models use a set of previous symbols in the uncompressed symbol stream to predict the next symbol in the stream. PPM reduces predictions to symbol rankings. The number of previous symbols, \( n \), determines the order of the PPM model. If no prediction can be made based on all \( n \) context symbols, a prediction is attempted with \( n-1 \) symbols. This process is repeated until a match is found or no more symbols remain in context.

## 2 Detailed description of encoding techniques for fields other than 'Sequence' and 'Quality value'

**QNAME:** This field corresponds to the query template name. Many reads exhibit a common long subsequence in this field. We only store the number of each query name and encode the difference between the query's local position and its existing readID if it finds an identical query name in the existing list. Otherwise, we store the flag ‘0’ to indicate that it introduces a different query name. Since the identical query sequences are close to one another, we encode these small values with nonuniform distribution by Huffman coding.

**FLAG:** The range of the flag is from 0 to 65535. It contains small integer values fitting within a single byte or two with frequent repetitions. Since small values have high frequency, we use one byte to represent the value from 0 to 255 and three bytes, i.e., \( x/256 \) and \( x\%256 \) to represent other values. Then, we apply Huffman coding to encode the transformed values.

**RNAME:** This field corresponds to the reference sequence name. The majority of the query reads share the identical reference name over the entire BAM/SAM file. We label all appearing reference names and encode the numbers using Run-Length Encoding (RLE).

**POS:** This field is 1-based leftmost POSition/coordinate of clipped sequence and ranges from 0 to \( 2^{2^{k-1}} \) with an increasing trend. We apply the Delta Encoding (i.e., \( \Delta \) coding) followed by Huffman coding.
MAPQ: This field is the mapping quality (phred-scaled) ranging from 0 to 255. There are many consequent repetitions. We apply Huffman coding that provides better compression efficiency than RLE.

CIGAR: This field contains values of alpha-numeric type of any length, and also exhibits long runs of the same value. We apply the Lempel-Ziv-Welch (LZW) coding method for this field.

MRNM: This field refers to the mate reference sequence name, where most positions turn out to be ‘=’, which means the mate’s reference sequence is the same as this alignment’s, or ‘*’ if there is no mate. We use RLE for this field.

MPOS: This field is 1-based offset into the forward reference strand where the leftmost character of the mate’s alignment occurs. Offset is 0 if there is no mate. Value in this field ranges from 0 to $2^{29} - 1$ like ‘POS’ but without a similar increasing trend. We apply the Delta Encoding here but include the sign (i.e., ‘+’ or ‘-’) flag, and then use Huffman coding to process the transformed data combined with the string of ‘MRNM’.

TLEN: This field refers to the observed template size. The value is negative if the mate’s alignment occurs upstream of this alignment and 0 if there is no mate. It ranges from $-2^{29} + 1$ to $2^{29} - 1$. Although the distribution of ‘TLEN’ is irregular and there seems little relationship with other fields, we find that the value of $\text{abs}(\text{TLEN} - (\text{MPOS} - \text{POS}))$ is confined to a limited alphabet (See Figure S1 in Appendix) after transformation. We apply the Huffman coding combined with the value of ‘POS’, ‘MPOS’ and ‘MRNM’ to the transformed value.

OPTIONAL FIELDS: The optional fields are tab-separated and each read shares a similar format. In our experiments, we used the bzip2 tool, which works well for those identical descriptions.

3 Supplementary experimental figures and tables

Table S1: The compression results by bzip2 and the quantization error (MAPE) after k-means clustering on quality values. When $k$ equals 51, it is equivalent to a lossless compression scheme without quantization.

| $k$  | 5  | 10 | 15 | 20 | 30 | 40 | 45 | 51 |
|------|----|----|----|----|----|----|----|----|
| Compressed size | 63.4M | 88.3M | 103M | 106M | 145M | 148M | 149M | 150M |
| MAPE  | 27.49% | 15.49% | 12.72% | 10.74% | 0.617% | 0.0228% | 0.0003% | 0.00% |
Table S2: The statistics of variant callings based on NA12878chrom20 and HG00096chrom11 sequences using SAMTOOLS. It counts the number of recovered (i.e., tp: true positive), lost (i.e., fn: false negative), additional (i.e., fp: false positive) variants and recovered invariants (i.e., tn: true negative). is the number of variants from the set of true positives that changed their genotype classification from homozygous to heterozygous or vice versa.

| Sequence   | Method mode | tp      | fn | fp  | cgt   | tn         |
|------------|-------------|---------|----|-----|-------|------------|
| NA12878    | HUGO        | 49596862| 0  | 0   | 0     | 13368553   |
|            | HUGO _L(30) | 49596489| 373| 0   | 4     | 13368553   |
|            | HUGO _L(20) | 49595983| 879| 252636| 535  | 13115917   |
|            | HUGO _L(10) | 49593533| 3329| 256304| 2086 | 13112249   |
|            | HUGO _L(6)  | 49596033| 829 | 891593| 6529 | 12476960   |
|            | HUGO _L(2)  | 39658419| 9938443| 34020| 2149 | 13334533   |
|            | HUGO _L(1)  | 27431274| 22165584| 3768 | 408  | 13364785   |
|            | CRAM-l      | 49582804| 14058| 1390934| 4551 | 11977619   |
| HG00096    | HUGO        | 125698063| 0 | 0   | 0     | 9248417    |
|            | HUGO _L(30) | 125694505| 3558| 11659| 245  | 9236758    |
|            | HUGO _L(20) | 125692550| 5513| 11263| 751  | 9237154    |
|            | HUGO _L(10) | 125696676| 1387| 97048| 4548 | 9151369    |
|            | HUGO _L(6)  | 125695988| 2057| 185474| 9035 | 9062943    |
|            | HUGO _L(2)  | 118472136| 7225909| 43321| 8162 | 9205096    |
|            | HUGO _L(1)  | 110957968| 14740064| 29034| 3392 | 9219383    |
|            | CRAM-l      | 125675323| 22740| 771309| 13348| 8477108    |
Figure S1: Illustration of the relationship between the fields ‘TLEN’ and ‘POS’, ‘MPOS’. The X-axis and Y-axis represent the position and the value of MPOS-POS, TLEN, TLEN-(MPOS-POS), and abs(TLEN-(MPOS-POS)), respectively. These fields are highly correlated.
Figure S2: The information flow of encoding the alignment results with hg19, where the input file is NA12878.chrom20.ILLUMINA.bwa.CEU.low coverage.20111114.bam.

Figure S3: Illustration of an alignment for mapping fragments to a reference sequence. Note that exact mapped reads are often adjacent or near to each other and their offset positions in the reference sequence are nearby.

4 Available sources of genome sequences and software referenced in the paper

1. hg18, http://hgdownload.cse.ucsc.edu/goldenPath/hg18/chromosomes/;

2. hg19, ftp://ftp.ncbi.nlm.nih.gov/genbank/organisms/Eukaryotes/vertebrates_mammals/Homo_sapiens/GRCh37/Primary_Assembly/assembled_chromosomes/FASTA/;

3. HuRef, alternate assembly http://www.ncbi.nlm.nih.gov/genome/guide/human/release_notes.html#37.1assembly;

4. Genome whose name begins with NA or HG, ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data;

5. SOAP3, http://www.cs.hku.hk/2bwt-tools/soap3/ and http://soap.genomics.org.cn/soap3.html;

6. SAMtools, http://samtools.sourceforge.net/;

7. CRAM, http://www.ebi.ac.uk/ena/about/cram_toolkit;
8. Samcomp, http://sourceforge.net/projects/samcomp/;

9. SRX014899/SRR032209, http://www.cibiv.at/~niko/ngc/download.html;

10. ERX007969/ERR019653, http://www.cibiv.at/~niko/ngc/download.html;

11. mouse (Mus musculus genome mm9, NCBI Build 37),
    http://hgdownload.cse.ucsc.edu/goldenPath/mm9/bigZips/mm9.2bit;

12. NC_000913 (Escherichia coli str. K-12 substr. MG1655 genome),
    ftp://ftp.ncbi.nlm.nih.gov/genomes/Bacteria/Escherichia_coli_K_12_substr__MG1655_uid57779/NC_000913.fna

5 References

1 Huffman D. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE* 1952;40:1098–101.

2 Welch TA. A technique for high-performance data compression. *Computer* 1984;17:8–19.

3 Ziv J, Lempel A. Compression of individual sequences via variable-rate coding. *Information Theory, IEEE Transactions on* 1978;24:530–6.

4 Cleary J, Witten I. Data compression using adaptive coding and partial string matching. *Communications, IEEE Transactions on* 1984;32:396–402.