BIOINFORMATICS APPLICATIONS NOTE
Vol. 30 no. 18 2014, pages 2676–2677
doi:10.1093/bioinformatics/btu362

Sequence analysis

Advance Access publication May 28, 2014

CODOC: efficient access, analysis and compression of depth of coverage signals

Niko Popitsch
Center for Integrative Bioinformatics Vienna (CIBIV), Max F Perutz Laboratories, University of Vienna and Medical University of Vienna, Dr. Bohrgasse 9, 1030 Vienna, Austria

Associate Editor: John Hancock

ABSTRACT

Summary: Current data formats for the representation of depth of coverage data (DOC), a central resource for interpreting, filtering or detecting novel features in high-throughput sequencing datasets, were primarily designed for visualization purposes. This limits their applicability in stand-alone analyses of these data, mainly owing to inaccurate representation or mediocre data compression. CODOC is a novel data format and comprehensive application programming interface for efficient representation, access and analysis of DOC data. CODOC compresses these data ~4–32× better than the best current comparable method by exploiting specific data characteristics while at the same time enabling more-exact signal recovery for lossy compression and very fast query answering times.

Availability and implementation: Java source code and binaries are freely available for non-commercial use at http://purl.org/bgraph/codoc.

Contact: niko.popitsch@univie.ac.at

Supplementary information: Supplementary data and usage examples are available at *Bioinformatics* online.

Received on March 16, 2014; revised on April 30, 2014; accepted on May 21, 2014

1 INTRODUCTION

Depth of coverage (DOC) data are one-dimensional discrete genomic signals describing the number of reads from a high-throughput sequencing (HTS) experiment covering each position in a reference genome. DOC is determined by traversing short-read alignments column-wise and counting the number of mapped reads overlapping with each individual position (i.e. its coverage). DOC is a primary source of information for the detection of structural variants where it is assumed that a gain/loss of genetic material results in increased/decreased coverage. The same fundamental assumption is exploited in RNA sequencing where estimated expression rates are calculated from normalized coverage signals. DOC also plays a central role in, e.g. identifying novel transcripts, characterizing the alternative splicing landscape of a sample or comparing multiple SNP-calling datasets where it is required to determine what genomic regions are sufficiently covered in all considered datasets to be comparable among each other (Ameur *et al.,* 2011; Duan *et al.,* 2013; Meynert *et al.,* 2013; Teo *et al.,* 2012).

Current data formats—Broad's tiled data file format, TDF, and UCSC's BigWig format (Kent *et al.,* 2010; Thorvaldsdóttir *et al.*, 2013)—were optimized for efficient display of DOC in genomic browsers but not for their stand-alone analysis. Consequently, they are not optimized for strong data compression or result in very lossy signal reconstruction, which makes them unfavorable for storing, processing and sharing large amounts of such data. Therefore, we developed CODOC, a novel data format and application programming interface (API) especially focusing on good data compression and exact signal reconstruction.

2 COMPRESSION ALGORITHM

CODOC supports lossy and lossless data compression. Generally, DOC data are compressed by run-length encoding (RLE), exploiting several HTS data characteristics to get longer and thus overall fewer run-lengths, which finally requires less storage space. This is achieved by using quantization intervals ('corridors') that define an upper and a lower bound for coverage values. A corridor interval is calculated from an initial coverage value cov_i at a genomic position $pos_i = (chr_i, off_i)$, consisting of a chromosomal identifier and an offset value, and possibly additional parameters. Starting from pos_i, CODOC iterates over subsequent positions $pos_{i+1}, pos_{i+2}, \ldots$ until it finds a value cov_e at position pos_e that lies outside of the corridor. Next, a new codeword consisting of this chromosomal position, the coverage right before the corridor was left (cov_{e-1}) and the current coverage cov_e is created. This latter value then acts as the initial coverage of the next corridor. By this, CODOC creates codewords only at positions showing a substantial signal change and stores exact coverage values and signal slopes at such 'key positions'.

Quantization corridors. CODOC supports several different and even user-defined quantization corridor schemas, including uniform and non-uniform alternatives (Supplementary Material). An example for the latter is our proposed default corridor schema (Fig. 1).

$C_{def}(p, cov_i) = [Round(cov_i \times (1-p)),\ Round(cov_i \times (1+p))]$ with $p \in [0, 1]$ being a parameter to control compression 'lossyness'. In this schema, the corridor height shrinks with low and grows with high initial coverage values. Consequently, the signal in low-coverage regions can be reconstructed more accurately, as already small changes will make it leave the (narrower) corridors, which results in more fine-grained sampling in such areas. Note that $p = 0$ actually results in lossless data compression, while larger values for p compress the data in an increasingly lossy manner.

External information. Users may pass lists of genetic variants (e.g. VCF files) to our compressor, which considers these variant positions as 'key positions' without storing corresponding codewords, as the respective information (including variant

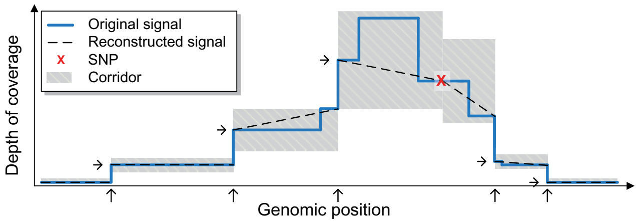

Fig. 1. Proposed non-uniform quantization schema. Original and reconstructed DOC signals are shown as bold blue and dashed black lines, respectively. Horizontal and vertical arrows denote key positions and their coverage values that initiate new quantization corridors (depicted as dashed, gray boxes) and codewords. In this schema, the height of a corridor is a percentage of its initial coverage value. The red X marks a SNP that was read together with its DOC value from a VCF file. This external information results in a new quantization corridor but no new codeword. Pseudocode describing the compression algorithm can be found in the Supplementary Material

coverages) can be read from these external lists at decompression time (Fig. 1). This further reduces the number of required codewords while at the same time preserving (near-) exact DOC values at/close to variant positions.

Blocked encoding and compression. Codewords are converted to byte representations using different encoders: chromosome identifiers are stored by standard RLE, position offsets by differential Golomb/Rice encoding (Golomb, 1966), and coverage values are stored unencoded. Eventually, all encoded bytes are additionally compressed by a general-purpose compression algorithm (gzip or bzip2). The data are then split into blocks (BITs) that contain a configurable maximum of codewords (100 000 by default), which enables efficient random access at decompression time.

Decompression. Signal reconstruction is done by linear interpolation at neighboring key positions. Decompression starts by constructing an in-memory interval tree (Cormen *et al.,* 2001) of BITs and their byte offsets. When a particular genomic position is queried, CODOC consults this index and loads the BIT containing the queried position from disk, finds the key positions and their coverage values by binary search and reports the interpolated result. CODOC caches accessed BITs using a least-recently used strategy to exploit spatial locality of queries to close genomic positions.

3 RESULTS AND DISCUSSION

In our evaluation, CODOC outperforms BigWig (~50–70×), lossless TDF (~4–32×) and lossy TDF (~6–14×) considerably with respect to compression ratios (Supplementary Material). In lossy mode, our method provides better signal reconstruction in low-coverage regions at the cost of less accuracy for higher coverage values when compared with lossy TDF. This is by design, as we consider exact reconstruction of low-coverage (especially uncovered) regions as more important because most thresholds (e.g. for SNP filtering) are also established in this range. CODOC also extracts/compresses data considerably faster (2–5× for lossless modes) than TDF, being slower only for some lossy configurations. Our API answered random-access queries to uncached regions in ~0.1–0.4 s and queries to cached BITs in the sub-millisecond range.

API functions. In addition to good data compression, our Java API supports efficient random and sequential access to compressed DOC data and provides several useful access methods that distinguish it from other toolsets, thereby facilitating the development of novel data analysis tools. Notable functions include strand-specific DOC extraction, querying of regions above/below given coverage thresholds, calculation of minimum/average coverage per region of interest, signal rescaling (normalization) or pairwise combination of multiple DOC signals (e.g. for calculating average, difference or minimum signals). CODOC also enables users to choose from various quantization strategies (which allows balancing of signal reconstruction accuracy and required storage space) or to implement own corridor functions. One special API feature is that it reports strict upper and lower error bounds for returned interpolated values based on the quantization corridor at this position. This enables applications to estimate potential errors and react respectively.

Conclusions. DOC helps to answer central questions of many HTS data analyses: How abundant is a certain transcript in my data? What genomic regions are sufficiently covered/accessible for certain analyses? What sequenced genomic regions in my data are amplified/deleted? CODOC helps researchers to answer these questions in a fast and accurate way. It is, to the best of our knowledge, the first open data format and API that was specially designed for the computational analysis of such data; by this it complements current formats developed with highly specific application scenarios (e.g. display in a genomic browser) in mind. Besides its comprehensive API functionalities, it is particularly the efficient compression of CODOC that proves beneficial when dealing with the enormous amounts of data produced by current HTS technologies.

ACKNOWLEDGEMENTS

The author thanks Stefan Leitich for helpful discussions about audio compression methods and several CIBIV colleagues and the reviewers for critically reading the manuscript.

Funding: This work was supported by the CIBIV household budget.

Conflict of interest: none declared.

REFERENCES

Ameur,A. *et al.* (2011) Total rna sequencing reveals nascent transcription and widespread co-transcriptional splicing in the human brain. *Nat. Struct. Mol. Biol.*, **18**, 1435–1440.

Cormen,T.H. *et al.* (2001) *Introduction to Algorithms*. 2nd edn. MIT Press, Cambridge, Massachusetts, USA.

Duan,J. *et al.* (2013) Comparative studies of copy number variation detection methods for next-generation sequencing technologies. *PLoS One*, **8**, e59128.

Golomb,S. (1966) Run-length encodings. *IEEE Trans. Inf. Theory*, **12**, 399–401.

Kent,W.J. *et al.* (2010) Bigwig and bigbed: enabling browsing of large distributed datasets. *Bioinformatics*, **26**, 2204–2207.

Meynert,A.M. *et al.* (2013) Quantifying single nucleotide variant detection sensitivity in exome sequencing. *BMC Bioinformatics*, **14**, 195.

Teo,S.M. *et al.* (2012) Statistical challenges associated with detecting copy number variations with next-generation sequencing. *Bioinformatics*, **28**, 2711–2718.

Thorvaldsdóttir,H. *et al.* (2013) Integrative genomics viewer (IGV): high-performance genomics data visualization and exploration. *Brief Bioinform.*, **14**, 178–192.