Title

ChIPWig: a random access-enabling lossless and lossy compression method for ChIP-seq data.

Journal

Bioinformatics, 34(6)

Authors

Milenkovic, Olgica
Ravanmehr, Vida
Kim, Minji
et al.

Publication Date

2018-03-15

DOI

10.1093/bioinformatics/btx685

Peer reviewed

Bioinformatics, 34(6), 2018, 911–919
doi: 10.1093/bioinformatics/btx685
Advance Access Publication Date: 26 October 2017
Original Paper

OXFORD

Genome analysis

ChIPWig: a random access-enabling lossless and lossy compression method for ChIP-seq data

Vida Ravanmehr[1], Minji Kim[1], Zhiying Wang[2] and Olgica Milenković[1,*]

[1]Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA and [2]The Henry Samueli School of Engineering, Center for Pervasive Communications and Computing (CPCC), University of California, Irvine, CA 92697, USA

*To whom correspondence should be addressed.
Associate Editor: Bonnie Berger

Received on April 14, 2017; revised on October 19, 2017; editorial decision on October 22, 2017; accepted on October 25, 2017

Abstract

Motivation: Chromatin immunoprecipitation sequencing (ChIP-seq) experiments are inexpensive and time-efficient, and result in massive datasets that introduce significant storage and maintenance challenges. To address the resulting Big Data problems, we propose a lossless and lossy compression framework specifically designed for ChIP-seq Wig data, termed ChIPWig. ChIPWig enables random access, summary statistics lookups and it is based on the asymptotic theory of optimal point density design for nonuniform quantizers.
Results: We tested the ChIPWig compressor on 10 ChIP-seq datasets generated by the ENCODE consortium. On average, lossless ChIPWig reduced the file sizes to merely 6% of the original, and offered 6-fold compression rate improvement compared to bigWig. The lossy feature further reduced file sizes 2-fold compared to the lossless mode, with little or no effects on peak calling and motif discovery using specialized NarrowPeaks methods. The compression and decompression speed rates are of the order of 0.2 sec/MB using general purpose computers.
Availability and implementation: The source code and binaries are freely available for download at https://github.com/vidarmehr/ChIPWig-v2, implemented in C++.
Contact: milenkov@illinois.edu
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

ChIP-seq files contain information gathered using inexpensive ChIP-seq technologies designed to analyze the interactions between protein and DNA by combining chromatin immunoprecipitation and next generation sequencing methods. Consequently, in many emerging -omic data repositories, ChIP-seq files constitute a significant fraction of stored information: ChIP-seq data have been curated by projects such as the ENCODE project (Consortium *et al.*, 2004), hosted at the UCSC Genome Browser, various National Institutes of Health (NIH) programs, the Roadmap Epigenomics (Bernstein *et al.*, 2010) and the Cistrome projects (Liu *et al.*, 2011). Methods for downstream analysis of ChIP-seq data have also been reported in a vast volume of the bioinformatics literature, and they include peak calling and motif finding (Bailey *et al.*, 2013; Kuan *et al.*, 2011; Machanick and Bailey, 2011; Nakato and Shirahige, 2016).

ChIP-seq experiments generate raw reads in FASTQ format that require large storage space and are unsuitable for visualization. Consequently, most FASTQ files are accompanied by Wig files that provide summary information contained in the reads that needs to be visualized, such as read density. Furthermore, the sizes of ChIP-seq files in Bed and Wig format, available from ENCODE, Gene Expression Omnibus, or Galaxy repositories, often exceed 1 GB. Given that they are accessed and downloaded with high frequency, it becomes hard to avoid storage and communication bottlenecks. To enable efficient maintenance and organization of these and other genomic databases, specialized compression methods have been developed that can significantly reduce the size of the sequencing datasets (Cao *et al.*, 2007; Hoang and Sung, 2014; Kent *et al.*, 2010; Pinho *et al.*, 2011; Tabus and Korodi, 2008; Yu *et al.*, 2015). For Wig files, two of the most frequently used compression techniques

are bigWig and bigBed (Kent *et al.*, 2010). BigWig essentially uses classical gzip compression and leads to moderate reductions in file sizes. A more efficient compression method for RNA-seq and ChIP-seq Wig files, termed cWig (Hoang and Sung, 2014), was recently proposed, but appears to require special compilers and is currently not publicly available.

Recently, some of the authors proposed a compression method for RNA-seq Wig files termed smallWig (Wang *et al.*, 2016) that represents the state-of-the-art method in the field, offering significantly improved compression rates when compared to bigWig, cWig and gzip. SmallWig enables fast queries from the compressed files in addition to access to summary statistics features paralleling those offered by the UCSC Genome Browser. At the core of the smallWig algorithm are two separate encoding pipelines for location and expression tracks via differential and run-length encoding, and accompanying arithmetic compaction. Although smallWig allows for a large gain in terms of compression rate and compression/decompression time for RNA-seq Wig files, it is not suitable for *direct* use on ChIP-seq data due to different properties and forms of the data tracks. In RNA-seq files, the first column of the data lines represents chromosome locations that appear as consecutive positive integers, while in ChIP-seq Wig files, the chromosome locations in the first column are positive integers that are not necessarily consecutive. Furthermore, in ChIP-seq files, the average read densities may have high variance both globally and locally; by contrast, in RNA-seq Wig files, the processed expression values are mostly *locally* smooth. This is best illustrated by the histograms of two typical ChIP-seq and RNA-seq files, shown in Supplementary Figure S2. As a result, the smoothness feature of RNA-seq expression tracks makes them suitable candidates for run-length encoding, which may not be optimal for ChIP-seq data. Most importantly, smallWig is only designed to operate in a lossless mode, which may not be needed for highly noisy data or data used for visualization such as ChIP-seq Wig data. ChIP-seq data is typically processed through sequential peak calling followed by motif discovery, which makes it possible to exactly characterize the effects of lossy data quantization on the performance of two these inference algorithms. Thus, in many applications it may be desirable to perform lossy compression of ChIP-seq data, as it significantly reduces file sizes while preserving information relevant for downstream processing. No lossy data compression methods for ChIP-seq Wig or other visualization files are currently available.

We introduce a compression technique for ChIP-seq data, termed ChIPWig, which may be executed both in a lossless and lossy mode. The algorithm employs delta encoding, run-length encoding and arithmetic encoding akin to smallWig, but in a fundamentally different manner. In particular, ChIPWig aims to smooth out the variance in the average read densities and hence performs a transform step on the relevant data track. Furthermore, it uses specialized coding methods for the location sequence. Unlike smallWig, ChIPWig offers a lossy compression feature through uniform and nonuniform quantization. Uniform quantization ensures best compression results, as it significantly smoothes out the data which may then be efficiently compressed using runlength coding. Unfortunately, and as expected, uniform quantization degrades peak calling accuracy due to the same described smoothing properties. Nonuniform quantization leads to significant compressed file size reduction when compared to lossless methods, and it does not change the visual quality of the data or the output of peak calling and motif finding algorithms, as it performs smoothing in a specialized fashion controlled by the data distribution.

To illustrate the utility of our lossless and lossy compression methods, we tested the lossless and lossy mode of the algorithm on numerous ChIP-seq files from the ENCODE project. We include detailed information on 10 ChIP-seq files from the ENCODE project and compared the lossless ChIPWig compression rate to those of bigWig, cWig, gzip and the original Wig files. Our results show that lossless ChIPWig compression rates, on average, outperform those of bigWig by 6-fold, gzip 4.5-fold and cWig 2-fold. Lossless ChIPWig also enables random query of different chromosome regions by partitioning the data tracks into blocks of suitable size. In the random access mode, compression is performed on each block individually, trading-off compression rate for ease of targeted access. ChIPWig, unlike bigWig and cWig, also accepts different block sizes. Furthermore, it allows the user to access the summary statistics information (maximum, minimum, average and standard deviation of the average read densities, and peak magnitudes) for each block. Lossy compression is accomplished by first performing uniform or nonuniform quantization of the average read densities, and then executing the lossless ChIPWig procedure. Lossy ChIPWig was tested on the same 10 ChIP-seq files from the ENCODE project, and led to close to 3-fold reductions in file sizes for the case of uniform quantization, and 2-fold reductions in file sizes for the case of nonuniform quantization. For visualization purposes, the nonuniformly quantized files do not introduce any subjective, visible degradation.

In addition, we did a case study on two example ChIP-seq Wig files containing information about chromosomes Y and 11. On these datasets and their uniformly and nonuniformly quantized counterparts, we performed peak calling via NarrowPeaks (Madrigal, 2016) [for the most recent version of the software, the reader is referred to Madrigal and Krajewski (2016)]. Although peak calling is mostly performed on BAM or SAM files, some peak calling methods such as NarrowPeaks allow one to directly operate on Wig files; note that despite its name, NarrowPeaks can identify both narrow and broad peaks. We fed the output of the peak caller into MEME-ChIP (Machanick and Bailey, 2011) for the purpose of identifying binding motif sequences. The results reveal that in contrast to the uniform quantizer, the near-optimal (with respect to mean-square distortion, and for a sufficiently large number of thresholds) nonuniform quantizer did not introduce any performance loss in peak calling and motif finding. In other words, the results produced by NarrowPeaks and MEME-ChIP were identical for the original and nonuniformly quantized files in terms of preserving peak positions and motifs.

The paper is organized as follows. In Section 2, we outline the ChIPWig compression architecture and describe how a ChIP-seq Wig file is processed via delta, run-length and arithmetic encoding. In Section 3, we first describe the implementation of the lossless ChIPWig algorithm and illustrate its performance with respect to compression/decompression rate and execution time. Then, we proceed to introduce the quantization schemes used in the implementation of the lossy ChIPWig algorithm. In Section 4, we describe the results of peak calling and motif analysis of the example unquantized and quantized ChIP-seq dataset. In Section 5, we discuss the features of the algorithm and Section 6 concludes the paper.

2 Materials and methods

2.1 Lossless encoding

The output of a ChIP-seq experiment is sequence information in FASTQ format which is used for downstream analysis that generates

a vast amount of other types of data. A typical pipeline includes read mapping, peak calling and motif discovery, which in turn produce SAM/BAM, Wig and Bed files. There is a storage and computation time trade-off for each of the file types: keeping all files incurs a high cost of storage, while re-running large scale analysis is computationally expensive. A solution is to store raw data in a lossless manner, and highly processed data in a lossy form. In particular, Wig files contain 'processed' counts for protein binding sites, often containing read counts themselves, averaged over a window of 25 bp, or fold-control ratios. Typically, ChIP-seq files are stored in bigWig format which represents a compressed version of the Wig format. One advantage of the Wig files is a simple data structure that can accommodate visualization on genome browsers, which does not require large precision due to low sensitivity of the human eye.

For these reasons, we suggest lossy compression (quantization) of read densities site information in Wig files using scalar quantization schemes that map the large set of average read densities into a significantly smaller one. We investigate both uniform and nonuniform scalar quantization, as described in Section 2.2., coupled with lossless compression of the quantized values which offers further file size reductions.

For simplicity, we start by outlining the lossless ChIP-seq Wig file processing comprising delta and run-length encoding and then proceed to describe the compression technique based on arithmetic encoding. The ChIP-seq Wig files used in our experiments were generated as part of the ENCODE project, where the ChIP-seq files used for visualization are originally stored in bigWig format that has to be converted into a Wig format for subsequent ChIPWig compression.

A sequence is defined with a capital letter and elements of the sequence are denoted by lower case letters. For instance, $A = (a_1, a_2, \ldots, a_N)$ is a sequence with N elements. We write $[j] = \{1, 2, \ldots, j\}$, for any positive integer $j \in N$ and $[j, k] = \{j, j+1, \ldots, k\}$ for any non-negative integers j and k where $j < k$.

The ChIP-seq Wig files comprise the following sequence tracks:

- *The location sequence* $L = (\ell_1, \ell_2, \ldots, \ell_N)$, where N denotes the length of the sequence and $\ell_i \in N$ for all $i \in [N]$ satisfies $\ell_i < \ell_{i+1}$, $i \in [N-1]$. In this sequence, one often has $\ell_{i+1} = \ell_i + s$, where $s > 1$, and $i \in [N]$, except for some skipped locations, for which $\ell_{i+1} > \ell_i + s$. We note that while in most of the tested ChIP-seq data, s is a constant and is equal to 25, in some cases $s = 21$ and $s = 10$ were used as well. We also observed that locations such that $\ell_{i+1} > \ell_i + s$ are exceptions rather than the norm, but still appear significantly often. As an example, in one prototypical ChIP-seq Wig file used as the running example and described in the Results section, the number of locations for which $s > 25$ equals 6 723 719, while the number of locations for which $s = 25$ equals to 73 575 541.
- *The average read density sequence* $C = (c_1, \ldots, c_N)$, where $c_i \in R^+$ for all $i \in [N]$. The c_is indicate the average read density corresponding to the location ℓ_i [Read density refers to the average number of reads in a given window length (the length of the window varies from application to application), whereas coverage refers to the average number of reads mapped to a location in the reference genome.]. The sequences L and C have the same length.

Having defined the sequences L and C which represent the two columns of an ENCODE ChIP-seq Wig file, we are ready to describe the lossless compression techniques applied on the sequences.

Compressing the location sequence L involves the following steps.

1. **Delta encoding** refers to computing the difference sequence $D = (d_1, d_2, \ldots, d_N)$, $d_i \in Z, i \in [N]$, of a sequence. When the underlying sequence equals L, one has $d_i = \ell_i - \ell_{i-1}$.
2. **Run-length encoding** results in two sequences $S = (s_1, \ldots, s_K)$, $R = (r_1, \ldots, r_K)$ obtained by performing run-length encoding on the delta encoded sequence D, where $s_i \in D$ captures the symbol, while r_i equals the number of consecutive appearances (i.e. the runlength) of s_i in D, for all $i \in [K]$.
3. **Arithmetic encoding** is a form of entropy coding that uses the symbol probability distribution to perform variable-length interval parsing that leads to a sequence being represented by an interval. Arithmetic coding often outperforms Huffman entropy coding, as it operates on the sequence as a whole, rather than on individual symbols. The sequences S and R are compressed by an arithmetic encoder.

The compression method for the average read density sequence C involves the following steps.

1. **Scaling** refers to converting the average read density sequence, which consists of non-negative real-valued numbers, into integers. Each $c_i \in C$ is multiplied by a scaling factor 10^E, for some $E \in N$, where E is chosen based on the number of decimals in c_i. This results in a sequence $A = (a_1, \ldots, a_N)$, where the a_is are integers and $i \in [N]$. In most ENCODE ChIP-seq Wig files, the average read densities are represented with at most four decimal digits, resulting in $E = 4$. In lossless ChIPWig, all four decimal digits are retained, while in the lossy implementation, only up to two decimal digits are preserved.
2. **Delta encoding** results in a difference sequence $W = (w_1, \ldots, w_N)$, $w_i \in Z, i \in [N]$ is generated according to $w_i = a_i - a_{i-1}$.
3. **Arithmetic encoding** is applied on the delta encoded sequence W, separately from S and R. Joint compression is not pursued, as extensive analysis of joint track statistics performed for smallWig (Wang *et al.*, 2016) reveals that joint encoding for Wig files in general does not significantly improve performance, but increases computational time.

The block diagram of the compression scheme for the location sequence and the average read density sequence is shown in Figure 1a. The lossless ChIPWig performs scaling, delta encoding and arithmetic encoding on the average read density sequence, while the lossy ChIPWig adds a quantization step to the read density sequence. The quantized data is then processed by scaling, delta encoding and arithmetic encoding. The decompression method of ChIPWig is illustrated in Figure 1b. Both the lossless and lossy ChIPWig perform the same decompression steps.

Figure 2 provides a snapshot of the lossless compression algorithm outputs for a ChIP-seq Wig file. The average read densities were represented with at most three decimal digits. Thus, all values were multiplied by 1000 to obtain a list of integer values.

2.2 Quantization

The exposition in this section is heavily based on definitions, concepts and formulas from classical sources on quantization (Gallager, 2006; Gersho and Gray, 1992). Details regarding density estimation and relevant quantization results are deferred to the Supplementary Material.

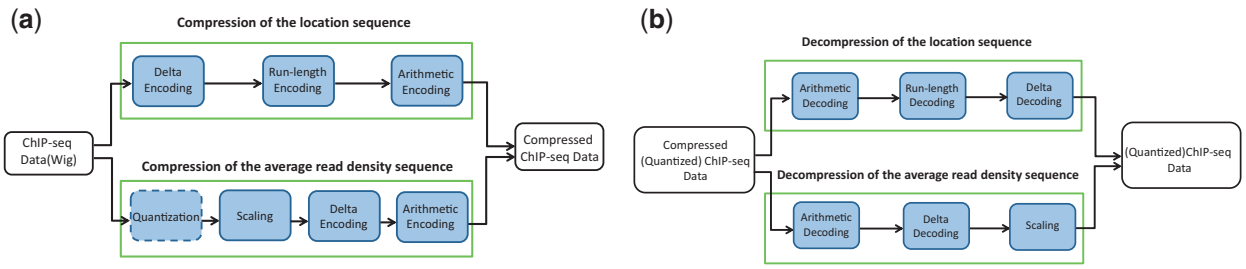

Fig. 1. The block diagram of the ChIPWig compression and decompression algorithms. (**a**) compression model; (**b**) decompression model

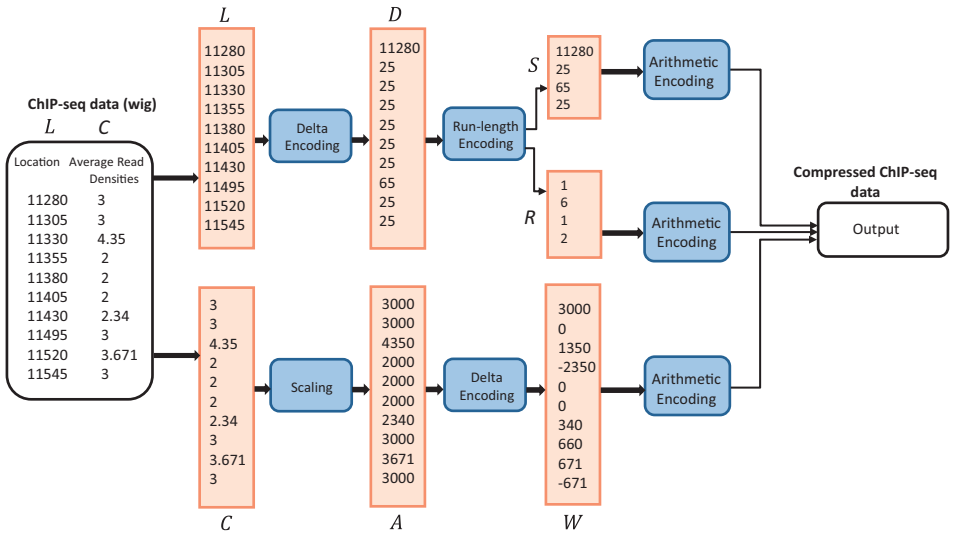

Fig. 2. Snapshot of the processing stages of the lossless ChIPWig compression algorithm on a ChIP-seq Wig file, applied to the sequences *L*, *C*, *D*, *S*, *R*, *A* and *W*

2.2.1 Scalar quantization

A scalar quantizer divides the set of real numbers or a given interval into M subsets (subintervals) $Q_1, \ldots, Q_M$, also called quantization regions. Each quantization region $Q_i = (t_i, t_{i+1})$ contains a chosen representative point $\omega_i \in Q_i$, termed a level. Quantization refers to mapping values in Q_i to the level ω_i, for all $i \in [M]$. The set $\{t_0, t_1, \ldots, t_M\}$ in which (t_i, t_{i+1}) defines the interval Q_i, is called the set of thresholds and the set $\{\omega_0, \omega_1, \ldots, \omega_{M-1}\}$ is called the set of levels. Thus, a scalar quantizer may be viewed as a function $Q(x) : R \to R$ where $Q(x) = \omega_i$ if $x \in Q_i$ (Gallager, 2006).

Let $X = (X_1, X_2, \ldots, X_N)$ be a sequence of i.i.d. random variables with individual probability density function $f_{X_i}(x)$ or some adequate distribution function, whenever the variables are discrete. Assume that the variables are quantized to $Y = (Y_1, Y_2, \ldots, Y_M)$ and let $Q(X)$ be the quantizer function used in the process. Then, the *second order distortion* is defined as follows:

$$D_2 = E\left[(X - Q(X))^2\right] = \int_{-\infty}^{+\infty} (X - Q(X))^2 f_X(x) dx \qquad (1)$$

Since the second order distortion measures the difference between the original data and quantized data, the goal is to design a quantizer function that minimizes D_2, given X and $f_X(x)$. The second order distortion is also called the mean-squared distortion or the mean-squared error (MSE). This distortion may be generalized to arbitrary orders m, $m \geq 1$, as described in the Supplementary Material. The choice of the distortion function may be governed by the downstream processing tasks performed on the quantized data. For example, if one only seeks very accurate quantization for certain ranges of values, a distortion with value of m larger than two may be desirable. However, we only consider the case that $m = 2$, as this choice is adequate for our implementation and describe optimal quantizer derivations for $m \geq 2$ in the Supplementary Material. We make use of both uniform and nonuniform quantizers and state closed-form formulas the MSE in both cases, which are known from the quantization literature. The formulas are used on the estimated data distributions, as outlined in the Supplementary Material.

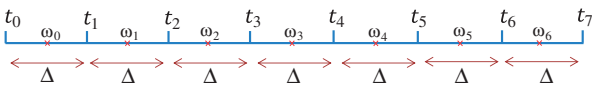

Fig. 3. An example of a uniform quantizer with seven quantization levels, where $\Delta = \frac{t_7 - t_0}{7}$ and $\omega_i = \frac{t_i + t_{i+1}}{2}$ for $i \in [0, 6]$

A. Uniform scalar quantization. In uniform scalar quantization, each quantization region Q_i has the same length $|Q_i| = \Delta$, provided that the underlying distribution has finite support. Each value $x \in Q_i = (t_i, t_{i+1})$ is mapped to the midpoint $\omega_i = \frac{t_i + t_{i+1}}{2}$. The uniform quantization function takes the form:

$$Q(x) = \Delta \times \lfloor \frac{x}{\Delta} \rfloor + \frac{\Delta}{2}. \qquad (2)$$

An example of a uniform quantizer with $M = 7$ quantization levels is shown in Figure 3. When the data is supported on an infinite interval, one of the quantization intervals has to be of infinite length, and the corresponding level has to be chosen with care. Given that all

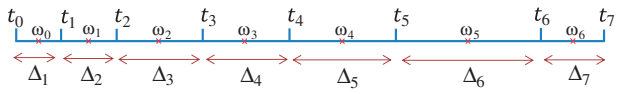

Fig. 4. An example of a nonuniform quantizer with seven quantization levels, where $\Delta_i = t_i - t_{i-1}$ for $i \in [1,7]$ and $\omega_i = \frac{t_i + t_{i+1}}{2}$ for $i \in [0,6]$

our data is supported in a finite interval, we do not discuss this scenario in detail.

It is known that for a uniform quantizer with M quantization levels between a and b and $\Delta = \frac{b-a}{M}$, one has

$$\text{MSE} = \frac{\Delta^2}{12}. \tag{3}$$

The proof for the general mth order distortion result is retraced in the Supplementary Material.

B. Nonuniform scalar quantization. In nonuniform quantization, the lengths of quantization regions are not necessarily equal. Each value $x \in Q_i = (t_i, t_{i+1})$ is mapped to either $\omega_i = \frac{t_i + t_{i+1}}{2}$, or another carefully selected level. Here, we only pursue the former level implementation. An example of a nonuniform quantizer with seven quantization levels is given in Figure 4, where the Δ_is, for $i \in [1,7]$, are the length of the quantization regions.

To design a quantization scheme with M quantization levels, when M is large, one usually refers to *asymptotic quantization theory* (Gallager, 2006). The *point density* of a quantizer is defined as a function $\lambda(x)$ such that for any a and b, $a < b$,

$$\int_a^b \lambda(x)dx \cong \frac{\text{number of levels between } a \text{ and } b}{M}, \tag{4}$$

where $\lambda(x) \geq 0$ and $\int_{-\infty}^{+\infty} \lambda(x)dx = 1$.

It can be shown that for a nonuniform quantizer with M quantization levels, the MSE depends on the probability density function $f_X(x)$ and the point density function $\lambda(x)$ as:

$$\text{MSE} \cong \frac{1}{12M^2} \int_{-\infty}^{+\infty} \frac{f_X(x)}{\lambda^2(x)} dx. \tag{5}$$

The point density that minimizes the MSE reads as

$$\lambda(x) = \alpha^{-\frac{1}{3}} f_X^{\frac{1}{3}}(x), \tag{6}$$

where the constant α is obtained from $\int_{-\infty}^{\infty} \lambda(x)dx = 1$.

The extension of this result for the general mth order distortion is outlined in the Supplementary Material.

Thus, given $f_X(x)$, the constant α and the density $\lambda(x)$ may be obtained from $\int_{-\infty}^{\infty} \lambda(x)dx = 1$ and (6), respectively. Furthermore, for a given number of quantization levels M, using (4), the number of quantization levels between any two values in the support of the distribution, v_1 and v_2, $v_1 < v_2$ can be obtained. Hence, asymptotically optimal (with respect to the MSE and for a large number of levels) quantizer design may be guided by the point density, as we illustrate on ChIP-seq Wig file data in the Supplementary Material.

3 Results

We start by describing the features of the lossless ChIPWig method both in the standard mode and random query mode and present a set of results pertaining to compression rates, compression and decompression times for 10 test files downloaded from the ENCODE hg19 browser. The files were selected randomly among ChIP-seq files of sufficient large size. We then proceed to present the same result for the lossy mode, which we accompany with a case study of downstream peak calling and motif finding on quantized data.

Table 1. Names and sizes (in MB) of the 10 ChIP-Seq files used in the compression experiments, retrieved from the ENCODE hg19 browser

File Number	File Size	File Name
1	959	wgEncodeBroadHistoneA549H3k09acEtoh02Sig
2	1000	wgEncodeBroadHistoneA549CtcfEtoh02Sig
3	938	wgEncodeBroadHistoneA549CtcfDex100nmSig
4	1460	wgEncodeBroadHistoneA549ControlEtoh02Sig
5	1210	wgEncodeBroadHistoneA549H3k04me3Etoh02Sig
6	925	wgEncodeBroadHistoneA549H3k27acEtoh02Sig
7	1090	wgEncodeBroadHistoneA549H3k27me3Dex100nmSig
8	781	wgEncodeBroadHistoneA549H3k79me2Dex100nmSig
9	1020	wgEncodeBroadHistoneCd20ro01794Ezh239875Sig
10	1210	wgEncodeBroadHistoneDnd41Ezh239875Sig

3.1 Lossless ChIPWig

As explained in Section 2, we use arithmetic encoders/decoders to process the sequences S, R and W. The arithmetic encoding that is used in our implementation is the same as the one used in smallWig (Wang *et al.*, 2016). This arithmetic compression algorithm is based on range coding (Martin, 1979) and some techniques from the rangemapper by Polar (http://ezcodesample.com/reanatomy.html?Source=To+article+and+ source+code).

The random access function is implemented through *block-wise* encoding, in which a ChIP-seq Wig file is divided into blocks of fixed length b where $b = 2^k$, $k = 12, \ldots, 18$, and where each block is processed and encoded separately. The encoded blocks are merged into one file which forms the compressed file. When the random query mode is active, a summary statistics of each block is stored in the compressed file. The summary statistics includes: (i) the minimum average read density in the block, (ii) the maximum average read density in the block, (iii) the mean of average read densities in the block and (iv) the standard deviation of the average read densities in the block. We also store two more values for each block that are not usually recorded in Wig files. These values represent the minimum jump and maximum jump in read density, which are of relevance for identifying relevant peak regions, and are defined as follows. Each local maxima in the average read density is identified with its value and the average densities appearing before and after the maxima. A jump is defined as the difference between the maxima and the average read densities immediately preceding and following the maxima. The minimum jump and the maximum jump are the smallest and largest jump within each individual block.

For our experiments, we used 10 ChIP-seq data files in bigWig format from the ENCODE hg19 browser listed in Table 1 which were converted to Wig format.

Figure 5 shows the compression rates achieved by the ChIPWig, compared to the rates of bigWig and gzip with 3 compression levels 1, 6 and 9. The compression rate equals the ratio of the compressed file size and the uncompressed file size. ChIPWig offers 6-fold rate improvement compared to bigWig and almost 4.5-fold improvement (on average) compared to gzip. Using gzip, the compression rate slightly decreases as the compression level increases. For compression with random queries with block size 65 536, ChIPWig offers a

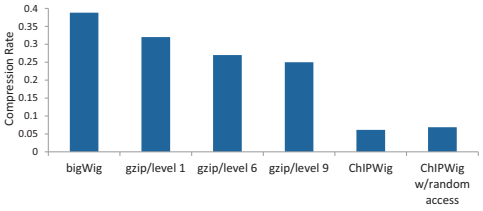

Fig. 5. Compression rates achievable by gzip with compression levels 1, 6 and 9, bigWig, and ChIPWig algorithms both in the standard mode and the random query mode with block size 65 536. Results represent the average over 10 sample ENCODE files

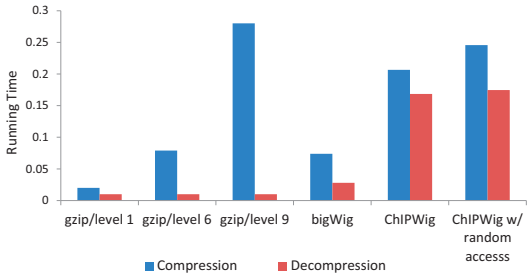

Fig. 6. Compression and decompression times of the gzip, bigWig and ChIPWig algorithms in standard mode and the random query mode with block size 65 536. Compression via gzip was performed using three different compression levels, 1, 6 and 9. The compression and decompression times are expressed in seconds per MB of the original Wig file size. All the results were averaged over the example 10 sample files from ENCODE hg19

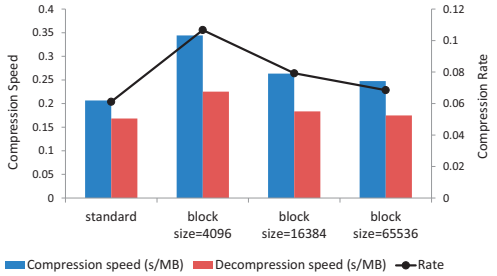

Fig. 7. Compression rate, compression time and decompression time for different block sizes. The label 'standard' indicates that the whole sequence is compressed as a single block. The compression/decompression time is expressed in seconds per MB in the original file. The *y*-label is used for both the rate and the speed (s/MB). All the results were obtained by averaging over the selected 10 sample ENCODE files

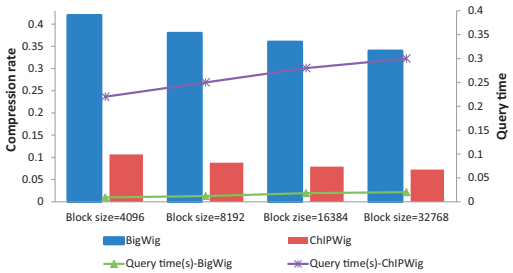

Fig. 8. Average Compression rate and average decompression time of a query of length 1000 with ChIPWig and BigWig using different block sizes. All the results were obtained by averaging over the selected 10 sample ENCODE files

5.5-fold rate improvements compared to bigWig. In order to compare the size of the compressed files with ChIPWig and cWig, we compared the average size of the ChIP-seq Wig files that we obtained with those reported in the paper introducing cWig, and found that our method offers almost 2-fold decrease in file size compared to cWig. We had to resort to this type of comparison as cWig is no longer available online.

In Figure 6, we present the running time of the ChIPWig compression/decompression schemes, as well as those of BigWig and gzip with 3 compression levels 1, 6 and 9. In general, the running time of ChIPWig is slightly longer, but comparable, to that of bigWig and gzip. However, in one case, compression time of the gzip with compression level 9 is longer than that of ChIPWig, bigWig and gzip with other compression levels. We would like to mention that gzip with compression level 9 has the best compression rate compared to bigWig and other compression levels of the gzip, which is still 4 times greater than that of in ChIPWig. The average compression times in the standard mode of ChIPWig is on average 0.12 sec/MB longer than that of in bigWig and gzip with compression level 6, and the average decompression time is 0.15 and 0.17 sec/MB per MB longer than that of in bigWig and gzip, respectively. To compare the effect of different block sizes used for random query on compression rate and compression/decompression time, we refer the reader to Figure 7. In these experiments, the block sizes ranged from 4096 to 65 536. In general, the results show that as the size of the block increases, the compression/decompression time and compression rate decrease. The ChIPWig algorithm with random access leads to a slight increase in compression rate from 0.06114 in standard mode to 0.1067 in random access mode with the block size 4096. It also leads to a modest increase from 0.2066 to 0.3443 sec/MB in the compression time rate and an increase from 0.1684 to 0.2252 sec/MB for the decompression time rate when compared to the standard mode. We also ran the ChIPWig algorithm in random query mode to determine the average decompression time of a query of length 1000 in compressed ChIP-seq files; the files were compressed by ChIPWig using blocks of sizes 4096, 8192, 16 384 and 32 768. This average decompression time was compared to the average time for a query of the same length (1000) in files compressed by bigWig, using the same block sizes. Figure 8 shows the average query time and the compression rates of the bigWig and ChIPWig methods. The results show that the average query time for queries of length 1000 over all 10 Wig files and 10 queries on each file increases as the block size increases for both bigWig and ChIPWig. The average compression rate of files compressed by bigWig is at least 4 times greater than that of with ChIPWig, while the query time for the bigWig is almost 10 times smaller than ChIPWig. However, we would like to note that ChIPWig also provides a statistical analysis during queries and returns the minimum, maximum, average and standard deviation of average read densities.

3.2 Lossy ChIPWig

In our second round of tests, we also applied both nonuniform and uniform quantization schemes on the 10 tested ChIP-seq Wig files. We first set a threshold τ on the average read density and applied the designed quantization schemes on values that lie on $(0, \tau)$. For values outside of this interval, we used one level as specified in the Supplementary Material. For our analysis, we first assumed that $\tau = 50$ and designed quantizers with $M = 50$ levels. Then, we repeated the simulations by letting τ be 70% of the maximum average read density in the file and let $M = 50$ and $M = 100$. For

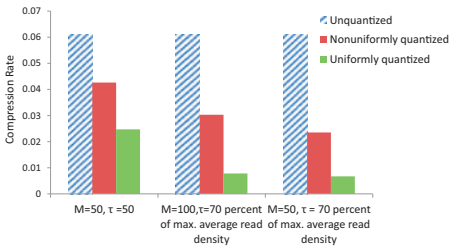

Fig. 9. Compression rates of the 10 sample ChIP-seq Wig files before and after nonuniform and uniform quantization schemes with $M = 100$ and $M = 50$. The threshold τ is set to 50 or 70% of the maximum average read density in each of the files

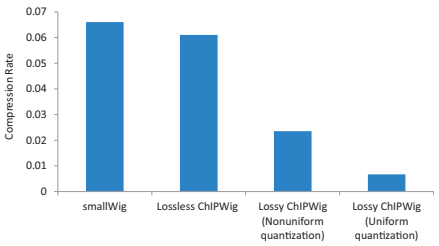

Fig. 10. Compression rates of the 10 sample ChIP-seq Wig files with smallWig and ChIPWig before and after nonuniform and uniform quantization with $M = 50$, and a threshold equals to 70% of the maximum average read density in each of the files

nonuniform quantization, we used an optimized point density for these two choices of the cutoff threshold τ. The average ChIPWig compression rate of the quantized files is shown in Figure 9. Overall, the uniform quantizer has a lower compression rate than the nonuniform quantizer with the same number of quantization levels and threshold. While the average compression rate in the lossless standard mode is almost 0.06, with nonuniform quantization, the average compression rate is reduced to 0.042 when $M = 50$ and $\tau = 50$; for uniform quantization, it is reduced to 0.024 for the same parameters. For the nonuniform quantizer, there is a small increase in the average compression rate from 0.023 for $M = 50$ and τ equals to the 70% of the maximum average read density in the file to 0.042 for $M = 50$ and $\tau = 50$. The results for the uniform quantizer also shows an increase in the average compression compression rate from 0.006 to 0.024, where the former result is obtained for $M = 50$ and τ equal to the 70% of the maximum average read density in the file; the latter is obtained for $M = 50$ and $\tau = 50$. Figure 10 shows the compression rates achieved by the ChIPWig both in the lossless and lossy modes, compared to the rate of smallWig. In order to perform smallWig on ChIP-seq files, we first converted the ChIP-seq files into a format used by smallWig, i.e. we converted ChIP-seq files to Wig files whose location sequence consists of consecutive integer values and then we performed smallWig on the corresponding Wig files. It is seen that even in the lossless compression mode, ChIPWig offers roughly a 10% improvement in compression rate when compared to ChIPWig. Furthermore, lossless ChIPWig coupled with nonuniform quantization offers a compression rate reduction of 65%. We would like to note that the size of the file created by converting a ChIP-seq file to a Wig file in the format that is used by smallWig is almost 30 times bigger than the size of the original ChIP-seq file, which in turn may create data storage issues. Also, we observed that the running time for compression of a ChIP-seq file with smallWig is at least five times slower than ChIPWig which directly compresses ChIP-seq files with different span sizes. For the lossy ChIPWig, we chose the compression rate corresponding to the compression of nonuniformly and uniformly quantized files with $M = 50$ and threshold equals to the 70% of the maximum average read density in each of the ChIP-seq files. In this case, the lossy ChIPWig with nonuniform quantization improves the compression rate by 2.86 fold compared to smallWig and the lossy ChIPWig using uniform quantization has almost 10 fold improvement compared to smallWig.

4 Peak calling and motif analysis

4.1 Peak calling

One of the most critical steps in ChIP-seq data analysis is to identify enriched regions and discover potential binding sites of proteins.

Table 2. The first five peaks identified by the NarrowPeaks when applied to chrY ChIP-seq file

Index	Chr.	Start	End	Width	Strand	Peak Position	Score
1	ChrY	2649426	2649625	200	*	2649426	2
2	ChrY	2649701	2650500	800	*	2650051	12
3	ChrY	2650676	2651150	475	*	2650751	3
4	ChrY	2653751	2653975	225	*	2653776	3
5	ChrY	2654326	2654500	175	*	2654326	2

Note: The symbol '*' denotes that the strand information is not used. The same values were recovered using optimal nonuniform quantization techniques with only 50 levels.

Generally, a peak is called when the number of covering reads exceeds a predetermined threshold, or when the enriched region is statistically significantly compared to the background peaks (Bailey *et al.*, 2013; Nakato and Shirahige, 2016; Steinhauser *et al.*, 2016). Among frequently used peak calling algorithms are MACS (Zhang *et al.*, 2008), SPP (Kharchenko *et al.*, 2008) and NarrowPeaks (Mateos *et al.*, 2015). NarrowPeaks operates on ChIP-seq data in Wig and bigWig formats (Mateos *et al.*, 2015) and recovers both narrow and broad peaks. We hence focus our attention on analyzing the performance of this peak caller on lossless and quantized Wig files.

To see how quantization affects peak calling, we use the first file in Table 1, 'wgEncodeBroadHistoneA549H3k09acEtoh02Sig' as our running example and focus on the locations and average read densities reported for chromosome Y; for simplicity, we refer to this file as the chrY ChIP-seq file. NarrowPeaks is applied with the default parameters both on the chrY ChIP-seq file and the corresponding (non) uniformly quantized files. NarrowPeaks returns the list of peaks with the start position, end position and width of the peak, as well as the peak position and its score. As an illustration, the first five peaks of the chrY ChIP-seq file identified by NarrowPeaks are listed in Table 2.

The compression rates of the uniformly and nonuniformly quantized chrY ChIP-seq file are shown in Figure 11. As expected, the uniform quantizer has a lower compression rate compared to the nonuniform quantizer. Both uniform and nonuniform quantizer have the lowest compression rate when $M = 50$ and $\tau = 307$. The compression rate for the nonuniform quantizer shows a slight increase from 0.032 for $M = 50$, $\tau = 307$ to 0.045 for $M = 50, \tau = 50$ and the compression rate for the uniform quantizer has also an increase from 0.012 to 0.025 for the same parameters as the nonuniform quantizer.

To evaluate the accuracy of the peak positions/scores found based on the quantized files, we calculated the Pearson correlation

coefficient (The Pearson correlation coefficient between two random variables equals the ratio of the covariance of the variables and the product of their standard deviations. This correlation measure is used to assess the degree of linear dependence between the variables.) between the corresponding random variables for the unquantized file and the quantized files with $M = 50$ and $\tau = 50$. The Pearson correlation coefficient between two random variables equals the ratio of the covariance of the variables and the product of their standard deviations. This correlation measure is used to assess the degree of linear dependence between the variables. Details of the derivations may be found in the Supplementary Material. We found that the correlation coefficient of the peak positions and scores between the unquantized file and the nonuniformly and uniformly quantized file are 1 and 0.8663, respectively. Hence, there exists a linear relationship that perfectly describes the peak positions/scores of the original and nonuniformly quantized data. A snippet of peak positions and scores for the running example is given in Table 2. More detailed results are found in the Supplementary Material.

4.2 Motif analysis

A widely accepted assumption is that many transcription factors exhibit DNA binding motifs so as to successfully locate themselves in the genome. Thus, discovering DNA motifs from the peaks is another step in the downstream analysis of ChIP-seq data. Although approaches may vary by software packages, the general workflow involves finding overrepresented sequences among a set of sequences covering the peak positions (Park, 2009). MEME (Bailey *et al.*, 2006) is the most popular tool for accomplishing this task, and it is based on the Expectation-Maximization (EM) algorithm. The authors of MEME have also developed MEME-ChIP (Machanick and Bailey, 2011) to accommodate large datasets. To demonstrate that quantization has little or no effects on the motif downstream analysis, we discovered motifs from the peaks identified in the previous testing stage. For each of the seven generated peak files (one original unquantized file, three uniformly quantized and three nonuniformly quantized files), we ran MEME-ChIP in default mode as described in what follows. We first computed the center of the peak regions by taking the floor [median(start, end)], and extended the sequence by 150 base pairs in both directions. Using the hg19 reference genome, we fetched fasta files containing sequences of 301 bps and ran MEME-ChIP under default setting. Figure 12 shows the three most significant (sorted by E-values) motif logos discovered by MEME-ChIP for the unquantized file. Of our six quantized files, three nonuniformly quantized files discovered the exact same motifs as the original file, verifying that near-optimal nonuniform quantization has no effects on downstream analysis. The remaining uniformly quantized files led to the discovery of similar, but different motifs. We listed the most significant motifs of the files obtained from uniform quantization in the Supplementary Material.

We also selected another chromosome at random from the remaining 22 chromosomes, chromosome 11, which is an average-length chromosome. In the text, we refer to the corresponding data file as the chr11 ChIP-seq file. We performed both nonuniform and uniform quantization of the file, with parameters $\tau = 50$, $M = 50$ and $\tau = 70\%$ of the maximum average read density, with parameters $M = 50$ and $M = 100$. We found that the number of peaks identified for the chr11 ChIP-seq file and three nonuniformly quantized files are equal. Running MEME-ChIP on the peaks of the nonuniformly quantized files produced the same most significant motif logos as those obtained from the unquantized file, which confirms that there is no performance degradation in the peak calling and motif discovery pipeline using near-optimal nonuniform quantization. The motif logos are shown in Figure 13. We listed the first five peaks of each file along with the most significant motifs of the peaks of uniformly quantized files in the Supplementary Material. As for the other chromosome, uniform quantization resulted in a significant change of identified motifs.

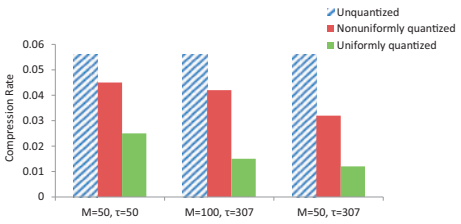

Fig. 11. Compression rates of the chrY ChIP-seq file before and after performing uniform and nonuniform quantizations. The portion of the file allocated to chromosome Y is 5 MB

5 Discussion

The basic design principles supporting ChIPWig compression can be easily modified and used on any other Wig data file. In case that ChIPWig is used to compress RNA-seq Wig files with integer expression values, performing run-length encoding on gene expression values is useful to reduce the size of the compressed file, while scaling is redundant and is to be skipped. In the random access mode, ChIPWig operates similarly to smallWig, and uses simple block-based encoding. ChIPWig accepts a variety of block sizes for encoding which is not the case for cWig that encodes fixed blocks of size 512 and gzip. ChIPWig is also implemented both in the lossless and lossy compression modes, while the smallWig, bigWig, gzip and cWig have been implemented only for the lossless compression. In the Supplementary Material, several examples of nonuniformly

Fig. 12. The most significant motif logos discovered by MEME-ChIP for the original unquantized chrY ChIP-seq file and all three nonuniform quantized files

Fig. 13. The most significant motif logos discovered by MEME-ChIP for the original unquantized chr11 ChIP-seq file and all three nonuniformly quantized files

quantized ChIP tracks are magnified and visualized alongside uncompressed tracks, showing that there is almost no shape distortion which may be of importance for visualization applications. Furthermore, given that nonuniform quantization techniques may be easily designed for any subjective distortion measure and that they tend not to influence peak calling and motif finding in any significant manner, it appears desirable to store these files only in quantized form.

6 Conclusion

We proposed an algorithm termed ChIPWig for compression of ChIP-seq Wig files, operating both in a lossless and lossy mode. ChIPWig has a significantly lower compression rate compared to bigWig and gzip. ChIPWig also has random access feature and provides access to summary statistics of each block. The performance of lossless and lossy ChIPWig on different ChIP-seq Wig files from the ENCODE hg19 browser was evaluated in terms of the compression rate and running time in the standard mode as well as the random access mode. Lossy ChIPWig first performs quantization on the average read density sequence of the ChIP-seq files and then performs lossless ChIPWig. Lossy ChIPWig shows an even lower compression rate compared to the lossless ChIPWig, while still maintaining important features of ChIP-seq data used in peak calling and motif finding.

Funding

The work was supported in part by the National Institutes of Health (NIH) Big Data to Knowledge (BD2K) Targeted Software Development program, under the grant number 5U01CA198943-03, the National Science Foundation (NSF) Emerging Frontiers for Science of Information Center, and NSF Graduate Research Fellowship Program DGE-1144245.

Conflict of Interest: none declared.

References

Bailey,T.L. *et al.* (2006) MEME: discovering and analyzing DNA and protein sequence motifs. *Nucleic Acids Res.*, **34**, W369–W373.

Bailey,T.L. *et al.* (2013) Practical guidelines for the comprehensive analysis of Chip-seq data. *PLoS Comput Biol.*, **9**, e1003326.

Bernstein,B.E. *et al.* (2010) The NIH roadmap epigenomics mapping consortium. *Nat. Biotechnol.*, **28**, 1045–1048.

Cao,M.D. *et al.* (2007) A simple statistical algorithm for biological sequence compression. In: *Data Compression Conference, 2007. IEEE*, pp. 43–52.

Consortium,E.P. *et al.* (2004) The ENCODE (ENCyclopedia of DNA elements) project. *Science*, **306**, 636–640.

Gallager,R. (2006) *Course Materials for 6.450 Principles of Digital Communications I*. MIT OpenCourseWare.

Gersho,A. and Gray,R.M. (1992) Vector quantization I: structure and performance. In: *Vector Quantization and Signal Compression*. The Springer International Series in Engineering and Computer Science (Communications and Information Theory). Vol 159, Springer, Boston, MA, pp. 309–343.

Hoang,D.H. and Sung,W.-K. (2014) CWig: compressed representation of wiggle/bedGraph format. *Bioinformatics*, **30**, 2543–2550.

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

Kharchenko,P.V. *et al.* (2008) Design and analysis of ChIP-seq experiments for DNA-binding proteins. *Nat. Biotechnol.*, **26**, 1351–1359.

Kuan,P.F. *et al.* (2011) A statistical framework for the analysis of ChIP-seq data. *J. Am. Stat. Assoc.*, **106**, 891–903.

Liu,T. *et al.* (2011) Cistrome: an integrative platform for transcriptional regulation studies. *Genome Biol.*, **12**, 1.

Machanick,P. and Bailey,T.L. (2011) MEME-ChIP: motif analysis of large DNA datasets. *Bioinformatics*, **27**, 1696–1697.

Madrigal,P. (2016) An introduction to the narrowpeaks package: Analysis of transcription factor binding ChIP-seq data using functional PCA.

Madrigal,P. and Krajewski,P. (2016) An introduction to the narrowpeaks package: Analysis of transcription factor binding ChIP-seq data using functional PCA.R/Bioconductor package version 1.21.0.

Martin,G.N.N. (1979) Range encoding: an algorithm for removing redundancy from a digitised message. In: *Proc. Institution of Electronic and Radio Engineers International Conference on Video and Data Recording.*

Mateos,J.L. *et al.* (2015) Combinatorial activities of SHORT VEGETATIVE PHASE and FLOWERING LOCUS C define distinct modes of flowering regulation in Arabidopsis. *Genome Biol.*, **16**, 31.

Nakato,R. and Shirahige,K. (2016) Recent advances in ChIP-seq analysis: from quality management to whole-genome annotation. *Brief. Bioinf.*, **18**, 279–290.

Park,P.J. (2009) ChIP–seq: advantages and challenges of a maturing technology. *Nat. Rev. Genet.*, **10**, 669–680.

Pinho,A.J. *et al.* (2011) On the representability of complete genomes by multiple competing finite-context (Markov) models. *PloS One*, **6**, e21588.

Steinhauser,S. *et al.* (2016) A comprehensive comparison of tools for differential ChIP-seq analysis. *Brief. Bioinf.*, **17**, 953–966.

Tabus,I. and Korodi,G. (2008) Genome compression using normalized maximum likelihood models for constrained Markov sources. In: *Information Theory Workshop, 2008, ITW'08. IEEE*, pp. 261–265.

Wang,Z. *et al.* (2016) smallWig: parallel compression of RNA-seq WIG files. *Bioinformatics*, **32**, 173–180.

Yu,Y.W. *et al.* (2015) Entropy-scaling search of massive biological data. *Cell Syst.*, **1**, 130–140.

Zhang,Y. *et al.* (2008) Model-based analysis of ChIP-seq (MACS). *Genome Biol.*, **9**, R137.