Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

OXFORD

Genome Analysis

Genome-wide search of nucleosome patterns using visual analytics

Rodrigo Santamaría [1,*], Roberto Therón [1], Laura Durán [2], Alicia García [2], Sara González [2], Mar Sánchez [2] and Francisco Antequera [2]

[1] Departmento de Informática y Automática, Universidad de Salamanca, Salamanca, 37008, Spain. and
[2] Instituto de Biología Funcional y Genómica, CSIC-USAL, Salamanca, 37007, Spain.

* To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: The Burrows-Wheeler Transform (BWT) is widely used for the fast alignment of high-throughput sequence data. This method also has potential applications in other areas of bioinformatics, and it can be specially useful for the fast searching of patterns on coverage data from different sources.
Results: We present a nucleosome pattern search method that converts levels of nucleosomal occupancy to a sequence-like format to which BWT searches can be applied. The method is embedded in a nucleosome map browser, "Nucleosee", an interactive visual tool specifically designed to enhance BWT searches, giving them context and making them suitable for visual discourse analysis of the results. The proposed method is fast, flexible and sufficiently generic for the exploration of data in a broad and interactive way.
Availability: The proposed algorithm and visual browser are available for testing at http://cpg3.der.usal.es/nucleosee. The source code and installation packages are also available at https://github.com/rodrigoSantamaria/nucleosee
Contact: rodri@usal.es
Supplementary information: Supplementary files are available at *Bioinformatics* online. A supplementary video tutorial is available at: http://vis.usal.es/rodrigo/nucleosee/nucleosee.mp4

1 Introduction

The Burrows-Wheeler Transform (BWT; Burrows and Wheeler, 1994; Adjeroh *et al.*, 2002) is a method used to compress data in a way that permits fast searches. Briefly, BWT takes a sequence of characters and computes all cyclic rotations over the sequence to generate a matrix of sequences. After sorting the rows alphabetically, the last column of the matrix is the BWT. Such a transformation has two useful properties: firstly, similar patterns are grouped together; and secondly, it permits retrieval of the original sequence allowing fast identification of similar sequences without losing the connection to the raw data. More than a decade after its appearance, BWT approaches to next-generation sequencing (NGS) alignments have been developed, driven by the need for faster methods of analysing the increasing amount of available genomic data (Li and Durbin, 2009; Langmead and Salzberg, 2012; Kim *et al.*, 2015).

High-throughput techniques are also used to determine sequence coverage, allowing nucleotide-resolution analysis of transcriptomes (Wang, 2009) along with other applications such as nucleosome mapping (Jiang and Pugh, 2009), and area where several computational resources are available (Teif, 2016).

High-throughput sequencing has also triggered the development of interactive visualisation tools (Yardimci and Noble, 2017; Kerpedjiev *et al.*, 2017; Lekschas *et al.*, 2017) focused on viewing large-scale data, but, in general, without incorporating pattern search techniques. Nucleotide-resolution or nucleotide-based pattern searches remain a challenge, both to expand searches beyond gene or transcript annotations, and to provide fast and reliable results that can be corroborated against the original data.

In the following sections we describe an approach that integrates a novel BWT-based search method into a genome browser especially designed for visualisation of results.

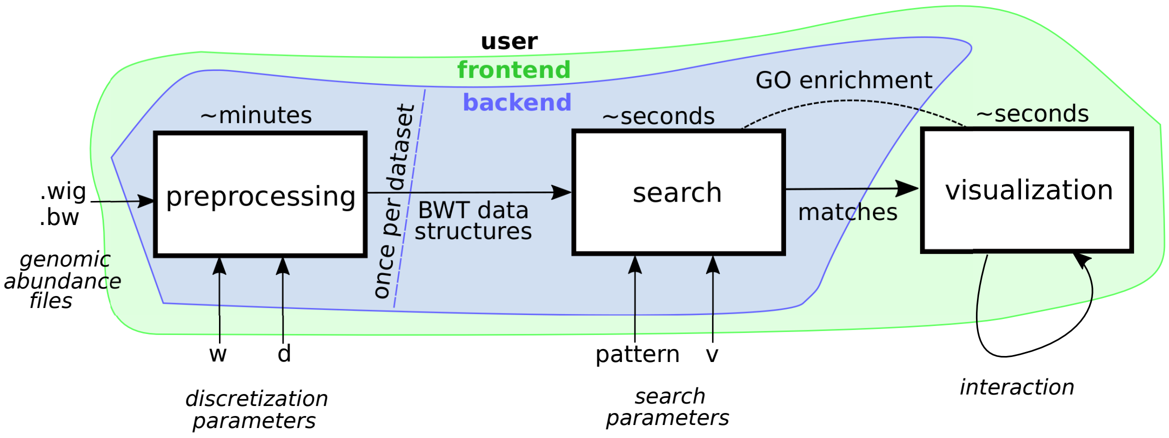

Fig. 1. System architecture. The three basic phases: 1) pre-processing, which is time intensive but performed just once per dataset; 2) search for patterns based on the BWT data structures generated, fast and flexible; and 3) browse for search matches and enrichment, which allows visualisation of results, original data and annotations.

2 System and methods

The general architecture of Nucleosee involves three main phases: pre-processing, search and visualisation (Fig. 1). In this section we will detail each of these phases, with a more technical description included in the Algorithm and Implementation sections.

2.1 Pre-processing

The first procedure necessary to be able to apply BWT to high-throughput numerical (i.e. coverage) data is to transform the data to a 'character' sequence. The advantage of such a transformation is twofold: firstly, it reduces the dimensionality of the data by binning continuous numerical ranges into a defined number of percentile ranges; secondly, the percentile ranges can then be treated as a character sequence that can be transformed by BWT for fast searching.

Such data discretisation is performed as follows. Let $A = a_0, \cdots, a_n$ be the original sequence of numbers. Let w be the window size and d the number of discrete levels. The discretised sequence $S = s_0, \cdots, s_m$ can be obtained with Eq. 1:

$$s_k = p_d\left(\frac{\sum_{i=kw}^{(k+1)w} a_i}{w}\right), \text{for } k = 0, \cdots, m \qquad (1)$$

Where $p_d(x)$ assigns a character to x depending on the percentile range it falls into, having taken d equally sized ranges into account (Fig. 2).

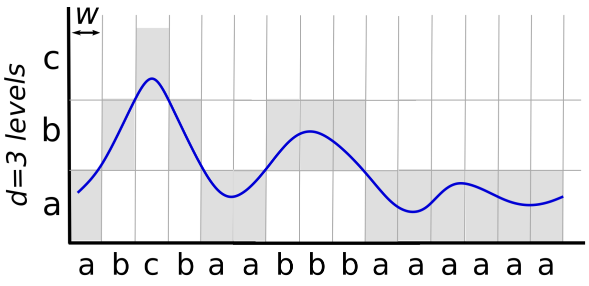

Fig. 2. Example of pre-processing. Numerical data are discretised into $d = 3$ percentile ranges or bins (a, b, and c) on a window w basis. The resulting sequence can be searched for patterns such as well positioned nucleosomes (abcba) or nucleosome depletion (aaaaa) using BWT.

2.2 Search

Search is implemented as an approximate pattern matching algorithm on BWT data, which is described in Section 3. The method provides flexibility on up to v character variations with respect to the search pattern, and also permits filtering by genomic features (genes, intergenic regions, exons or untranslated regions).

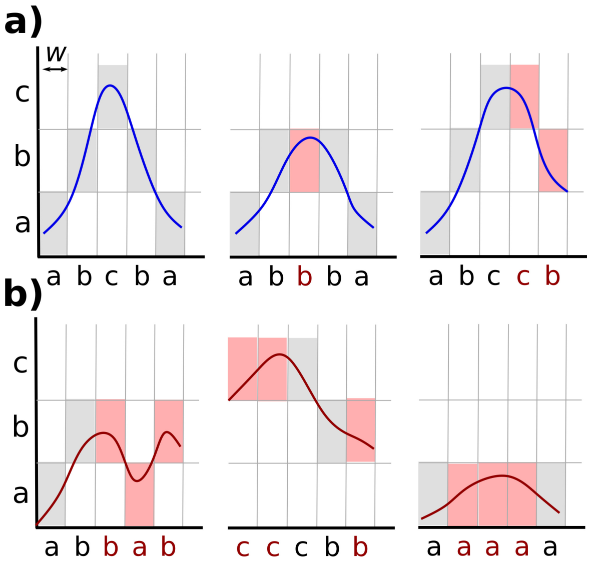

Fig. 3. a) Examples of nucleosome peaks characterised by the pattern abcba(2), which allows up to two variations relative to the abcba pattern (top). b) Examples of peaks that will not be reported in the search.

We tested the recovery power of our method using the genomic nucleosome occupancy map generated by micrococcal nuclease of the yeast *Schizosaccharomyces pombe*. The dataset included processed data of two biological replicates carried out on the *972 h-* strain, which is commonly used as a wild type reference (González *et al.*, 2016) (accession number GSE84910). Given that mononucleosomal DNA wrapped around the histone core spans 147 bps (Luger et al. 1997) and that the level of occupancy is usually symmetrical relative to the central (dyad) position, a search for the abcba pattern with Nucleosee should generate a genome-wide map of nucleosomal occupancy. To test this possibility, we allowed

$v = 2$ variations on the searched pattern, or `abcba(2)`, to identify well-positioned nucleosomes that might not exactly fit the `abcba` pattern (Fig. 3, Sup. Data 1).

We compared the results of Nucleosee with those generated by the widely used nucleosome search algorithm DANPOS (Chen *et al.*, 2013) (Sup. Data 2). We also searched for the same pattern `abcba(2)` on chemical nucleosome maps (Moyle-Heyrman *et al.*, 2013) and compared them with their reported unique nucleosomes. Although Nucleosee searches are designed for generic patterns, it is capable of recovering most of the reported nucleosomes by these two methods, with high sensitivity for sound margins of error. For example, Nucleosee is able to retrieve DANPOS nucleosomes with a sensitivity of 87.9% and a specificity of 68.5% for a margin of error of 40bps, which relates to the error due to the discretisation resolution (see Sup. Methods and Sup. Figs. 1 and 2)

Finally, our search design permits differential pattern search, by defining two same-length pattern searches for two distinct datasets, an performing the intersection, union or difference between the two resulting match sets. The design also permits differential agnostic or exploratory searches (without a specific predetermined pattern). To do so, the discretised genome is searched for windows of a defined length that differ in more than n characters between two samples.

2.3 Visualisation

High-throughput data and pattern matches are visualised on a genomic browser designed to search, analyse and confirm results from the original data. Based on this premise, the visual analytics design focuses on pattern search narrative rather than on the traditional topological pan-and-zoom interfaces (Kent *et al.*, 2002; Nicol *et al.*, 2009; Skinner *et al.*, 2009). This design unfolds into three levels parallel to the concept of semantic zoom (that is, adding different visual features based on the scale of the visualisation; see, for example, Westesson *et al.*, 2013) towards what we can call a 'narrative zoom' (see Fig. 5).

The genome level displays the chromosomes of the analysed organism (three in the case of *S. pombe*) and the number of matches found on each of them. The chromosome level displays the distribution of identified pattern matches along the entire chromosome at a resolution of several thousands of nucleotides per pixel (in the case of chromosome 1 of *S. pombe* in Fig. 5, about 4,000 per pixel). If there is functional enrichment, it is visualized at this level as a word cloud where more enriched terms (those with lower corrected p-values) are larger. If several data samples are loaded, their lines appear overlapped at this level, with different colours. Match locations for the specific searched pattern can be hovered over for detailed inspection at the gene level. Finally, the gene level focuses on a single pattern match, usually on a 1:1 pixel to nucleotide scale, including pan-and-zoom navigation and gene annotations. If several data samples are loaded they appear stacked as separate tracks at this level. If a data sample is comprised of several replicates, the between replicate variation is represented as a shaded area around the mean coverage.

All elements in the visualisation are interactive: chromosomes can be selected at the chromosome level, matches can be hovered over to show the pattern at the gene level, gene ontology (GO) terms can be selected to highlight corresponding matches and gene annotations can be hovered over to check details or clicked to gain focus. The use of a backend supported by BWT fast searches smooths the interaction by minimising computation times. A supplementary video explaining the interactive power for visual discourse with the data is available at http://vis.usal.es/rodrigo/nucleosee/nucleosee.mp4

3 Algorithm

The central algorithm of Nucleosee is the BWT search algorithm (see Algorithm 1). This algorithm is based on an adaptation of the BWMatching algorithm (Compeau and Pevzner, 2014), which requires BWT data structures to operate, in particular:

- The *bwt* matrix for fast retrieval of the pattern matches.
- A *suffix array* to recover the original genomic positions of searches.
- A *first occurrence* array with starting positions of each discrete level on the *bwt*.

Several other minor data structures and methods have been implemented to optimise time performance and reduce memory requirements. All these structures are built on the pre-processing step discussed in Section 2.1.

Algorithm 1 BWT search

```
procedure bwtSearch(bwt, fo, sa, pattern, v)
    text: pre-processed genome as discretised levels
    bwt: BWT sorted list of cyclic rotations of text
    fo: first occurrence of each symbol on bwt
    sa: integer array with the genomic positions of the BWT suffixes
    pattern: character string to search
    v: integer with the number of variations from pattern

    step ← $|pattern|/(v+1)$   ▷ minimal exact length for $v$ mismatches
    matches ← $\{\}$
    for $i = 0; i \leq |pattern|; i$ increases by $step$ do
        seed ← $pattern$[i:i+step]
        seed_matches ← $\{\}$   ▷ search for seeds of minimal exact length
        top ← 0
        bottom ← $|bwt| - 1$
        while top ≤ bottom do   ▷ BWT exact search
            if seed has more symbols then
                symbol ← next letter in seed
                fos ← fo[s]
                top ← fos + #times symbol occurs in bwt[:top]
                bottom ← fos + #times symbol occurs in bwt[:bottom+1] -1
            else
                add $sa[top : bottom + 1]$ to $seed\_matches$
        for $pos$ in seed_matches do   ▷ seed extension
            $mm$ ← mismatches of seed to $text[pos : pos + |pattern|]$
            if $mm \leq v$ then add $pos$ to $matches$ end if
    return $matches$
```

The BWT search algorithm splits the original pattern into sub-patterns (or seeds) that are searched with a BWT exact search. The resulting seed matches are reported if, when extended, they do not deviate from the original search pattern by more than v. Several other algorithms are implemented for data pre-processing, including GO enrichment, functional annotation, data visualisation, storage, concurrency and interaction. All of them are based on well-known existing solutions which we adapted to our needs.

4 Implementation

Nucleosee is implemented as a web browser. The backend is implemented in Python 3 and constitutes the grounds for BWT pre-processing and searching. It makes use of the following libraries:

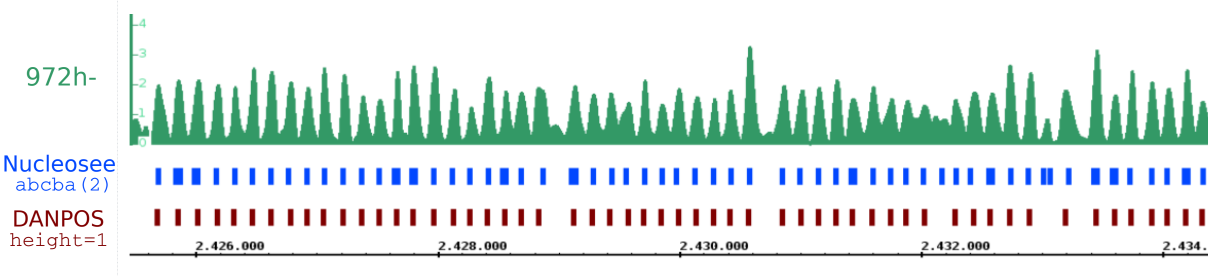

Fig. 4. Peak detection by NucleoSee and DANPOS. NucleoSee reported peaks (blue) for `abcba` patterns with two allowed variations. DANPOS results for a height cut-off of 1 (red), and S. pombe nucleosomal occupancy as detected by MNase analysis (green) (González et al., 2016) (section of chromosome 1, between 2,425,000 and 2,435,000).

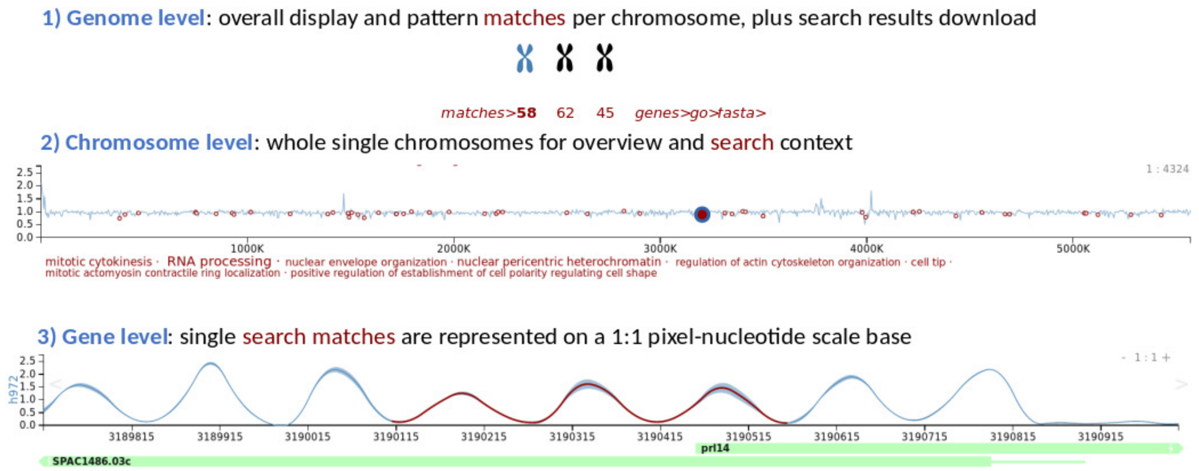
1) Genome level: overall display and pattern matches per chromosome, plus search results download
matches>58 62 45 genes>go>fasta>
2) Chromosome level: whole single chromosomes for overview and search context
1 : 4324
2.5
2.0
1.5
1.0
0.5
0.0
1000K
2000K
3000K
4000K
5000K
mitotic cytokinesis · RNA processing · nuclear envelope organization · nuclear pericentric heterochromatin · regulation of actin cytoskeleton organization · cell tip ·
mitotic actomyosin contractile ring localization · positive regulation of establishment of cell polarity regulating cell shape
3) Gene level: single search matches are represented on a 1:1 pixel-nucleotide scale base
- 1:1 +
h972
2.5
2.0
1.5
1.0
0.5
0.0
3189815
3189915
3190015
3190115
3190215
3190315
3190415
3190515
3190615
3190715
3190815
3190915
prl14
SPAC1486.03c

Fig. 5. Visual analytics design of the Nucleosee browser on three levels. The narrative zoom search for three nucleosomes matching exactly the pattern $abcba * 3$ is shown (in red). Notice that, for example, the higher nucleosome in blue right before the match responds to pattern $aacca$ instead of $abcba$ and therefore is not included in an overlapping nucleosome triplet.

- `pickle` for pre-processed data management.
- `numpy` for numerically complex computations.
- `fisher` for statistical enrichment.
- `flask` to provide searching and pre-processing as web services.

The frontend is implemented in Javascript and comprises the genomic browser and its interface. It makes use of the following libraries:

- `D3.js` for visual elements and interface.
- `bootstrap` for css style definition.
- `jquery` for communication with the web services.

The source code is distributed free with a GPL 3.0 license and is available at https://github.com/rodrigoSantamaria/nucleosee. This website also offers a Docker container with a ready to run stable version of Nucleosee on a Debian OS with Python 3.

A Nucleosee genomic browser has been set up for testing with pre-processed data for examples discussed in this paper (http://cpg3.der.usal.es/nucleosee). It includes a Help document explaining how to use the tool (http://cpg3.der.usal.es/nucleosee/help.pdf) and a video tutorial (http://vis.usal.es/rodrigo/nucleosee/nucleosee.mp4) showing the visual interaction for several of the cases discussed in this paper.

4.1 Genome annotation and GO enrichment

Genome annotation data is retrieved from the latest versions of the public repositories corresponding to each organism (gff or gff3 files). Gene ontology (GO) terms (Ashburner *et al.*, 2000) are taken from obo files at http://geneontology.org and annotations are taken from the latest versions of the corresponding organisms' public repositories (goa files). Gene ontology enrichment is implemented by a Fisher's exact test based on Python's `fisher` library, discarding electronically inferred annotations (IEA) and terms with less than five or more than 500 annotated genes. A false discovery rate (FDR) multiple hypothesis tests' correction is applied, with a threshold for the corrected p-value of 0.01. All these parameters (correction method, p-value threshold and discard options) are configurable on the backend web services.

4.2 Time performance

One of the most important advantages of BWT searches is time performance. Search times of around one second permit integration of searches into visual interfaces for a seamless dialogue between the analyst and the data. All performance tests described below were performed on a personal computer with a 2.4 GHz Intel Core 2 Duo processor and 16 GB RAM running a Debian 9.5 (Stretch) operating system.

To determine the effect of pattern length on time performance, we averaged the time performance of 20 searches for pattern lengths from two to 20 characters. Patterns were composed with random characters to avoid pattern composition biases.

To determine the effect of pattern flexibility, we averaged the time performance of 20 searches allowing variations in zero to three characters, for five-length random-character patterns. These tests were performed using two wildtye replicates of genome-wide maps of nucleosomal occupancy of *S. pombe*, pre-processed with $w = 30$ and $d = 3$.

To determine the effect of genome size, we selected high-throughput data from four organisms: *Saccharomyces cerevisiae*, *Caenorhabditis*

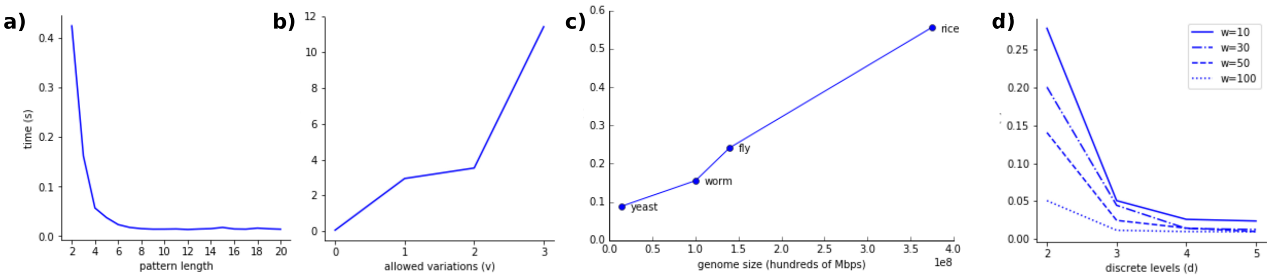

Fig. 6. Time performance for different searches. a) Performance across different pattern sizes (v=0). b) Performance across different variation values for a pattern of size five. c) Performance across different genome sizes. d) Performance for different pre-processing scenarios.

elegans, *Drosophila melanogaster* and *Oryza sativa* taken from publicly available wig files (GEO accession numbers GSM585199, GSM2417781, GSM883798 and GSE81436 respectively), which were pre-processed with $w = 30$ and $d = 3$. Time performance was averaged from 20 searches for five-length, random-character patterns.

To determine the effect of the pre-processing parameters, we pre-processed the above cited *S. pombe* maps with ranges for d (2 to 5 discrete levels) and w (10, 30, 50 and 100 bp windows). Then we performed exact searches ($v = 0$) as in the pattern length case, for the different pre-processed data.

The results of the time performance tests (Fig. 6) show that search of genomes in the order of 10 Mbps and pattern sizes larger than two took well below one second provided that the allowed number of variations was zero (see Fig. 6a). Patterns of length below six have higher search times because they become ubiquitous in the genome and hard to pack by BWT. Variations larger than $v = 0$ severely affected the time performance, although performance remained acceptable while $v < 3$ (Fig. 6b). This is because patterns are divided into seeds (sub-patterns) that are then searched based on v (see Algorithm 1). Depending on the pattern size and v, small-sized seeds might compromise the performance and increase searching times as seen in Figs. 6a and 7b. Searches of a few hundred bps (for example, patterns of length five with a discretisation window $w \sim O(10)$ and $v = 0$) on genomes up to 0.5 Gbps were also generally below one second as expected by the linear complexity of BWT searches (Fig. 6c). For larger genomes, memory consumption is usually a bottleneck, considering personal computers typically have 8 or 16GB RAM, but this can be overcome by the installation of the backend on a more powerful server. To increase the number of discrete levels improves time performance, as it permits early discard of larger sections of the genome (see Fig. 6d), although gains are small if $d >= 3$. Window size (w) effect on time performance is negligible, except if $d = 2$, were it become a parameter more relevant that the number of discretisation levels.

5 Discussion

5.1 Pattern Search

In addition to the fast and reliable identification of well positioned nucleosomes (Fig. 4), Nucleosee main goal is to allow search for pattern definitions. For example, 600 bp long regions depleted of nucleosomes could be identified by searching for the `a*20(1)` pattern in the *972 h-* strain, which reports 206 occurrences, of which 137 partially overlap genes and 67 map fully inside them. The chromosome level shows that enrichment identifies six of these genes as related to the *plasma membrane*, three of which (*ght6*, *ght8* and *SPCC794.04c*) are closely located in the initial section of chromosome 3 (Fig. 7.)

Another example of a possible search is for isolated nucleosomes, defined as nucleosomes flanked by 150 bp long nucleosome-depleted regions (NDRs). In this case, the BWT search pattern would be `a*5+abcba+a*5(1)`. Nucleosee identifies 74 regions with this pattern in the genome, which are mostly located between genes. Chromosome level visualisation of chromosome 3 reveals that two pairs of occurrences are very close, flanking the *wtf13* and *wtf23* repetitive elements (Fig. 8). Further inspection of the *wtf* family (20 genes in *S. pombe*, all of which are chromosome 3) shows that at least half of the genes are flanked by isolated nucleosomes at one or both ends (see Sup. Video).

We have not tested Nucleosee on larger genomes (e.g. mammals) because available maps do not show a regular distribution of nucleosomes along the genomes. This could represent the real situation in vivo or could be a consequence of the lower sequencing coverage due to the large size of their genomes. This means that discretisation and search for specific patterns with Nucleosee would not yield results as clear as in yeasts.

5.2 Differential search based on other patterns and genomic features

Nucleosee can also identify pattern differences between two datasets. It is possible, for example, to search for well-positioned nucleosomes in a strain that have become misplaced in another. To illustrate this, we searched for regions encompassing five consecutive well-positioned nucleosomes in the *972 h- S. pombe* strain (`abcba*5`) that are delocalised in the mutant strain *hta1*Δ (`b*25`), allowing three single-character deviations from the patterns ($v = 3$). The search reported two matches corresponding to genes *nap1* and *yap18* (Fig. 9).

As the BWT searches are connected to the visual interface which integrates genome annotations and GO enrichment, the user can see that the GO term *DNA replication-independent nucleosome assembly* is enriched (one out of the seven genes annotated with this term -*nap1*- appears in our search with a corrected p-value of 2.6×10^3). Although the enrichment is weak, this option allows the user to identify this term as possibly relevant but weak and perform a search without losing context for the remaining six genes in the term. Some of these genes' profiles also indicated nucleosome misplacement, but such misplacement did not match the original pattern search (for example, *hip1* and *hip4*; see Fig. 9, bottom). This way, the search algorithm, the data visualisation, the integrated annotations and the automatic enrichment collaborate to assist the analyst with the discursive process (see Sup. Video for the full analysis flow).

5.3 Differential agnostic search

To illustrate differential agnostic searches we searched for windows 1 kbps long with differences on at least 700 bps between the *972 h-* and *hta1*Δ strains. Taking into account the BWT pre-processing (Fig. 2), this means changes on 23 out of 33 bins of 30 nucleotides. Twenty-eight regions were identified that fit this definition, all of them corresponding to gene regions where the nucleosome position is fuzzy on both samples but occupancy is higher on one of them (Fig. 10). Again, direct comparison of the results

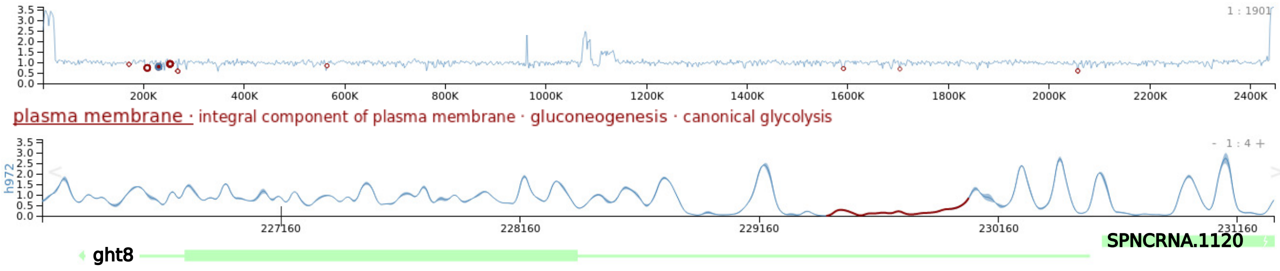

Fig. 7. 600 bps length NDRs partially overlapping with genes on chromosome 3. Five occurrences are on the first section of the chromosome, three of which are annotated with the GO term plasma membrane. One of these genes, ght8, is selected and visualised in the gene level below. Genes are shown in green and the pattern found by Nucleosee is highlighted in red.

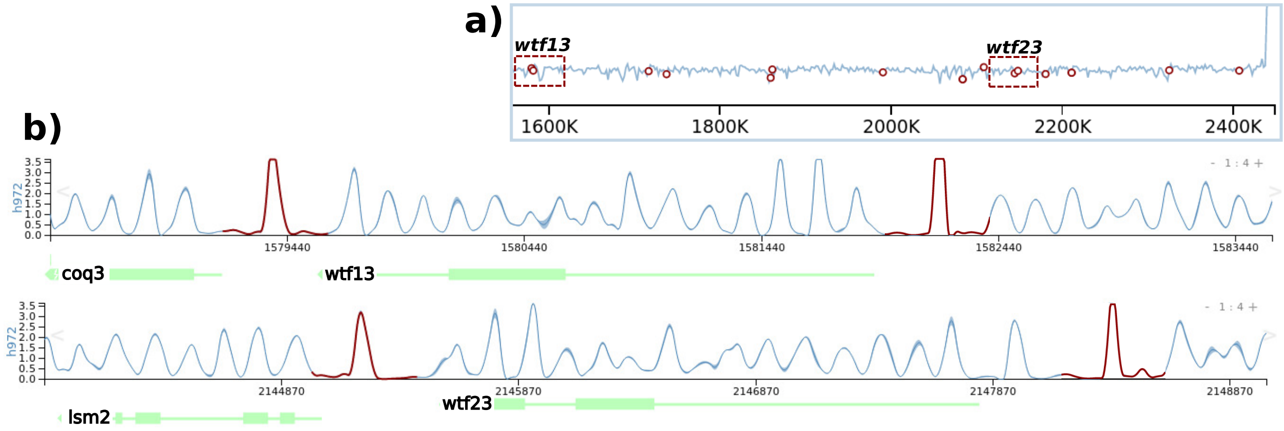

Fig. 8. ’Isolated nucleosome’ patterns flank wtf genes. a) Chromosome 3 track detail reveals close pattern occurrences. b) Gene level tracks for wtf13 and wtf23 reveal the pattern is present at both sides of these elements.

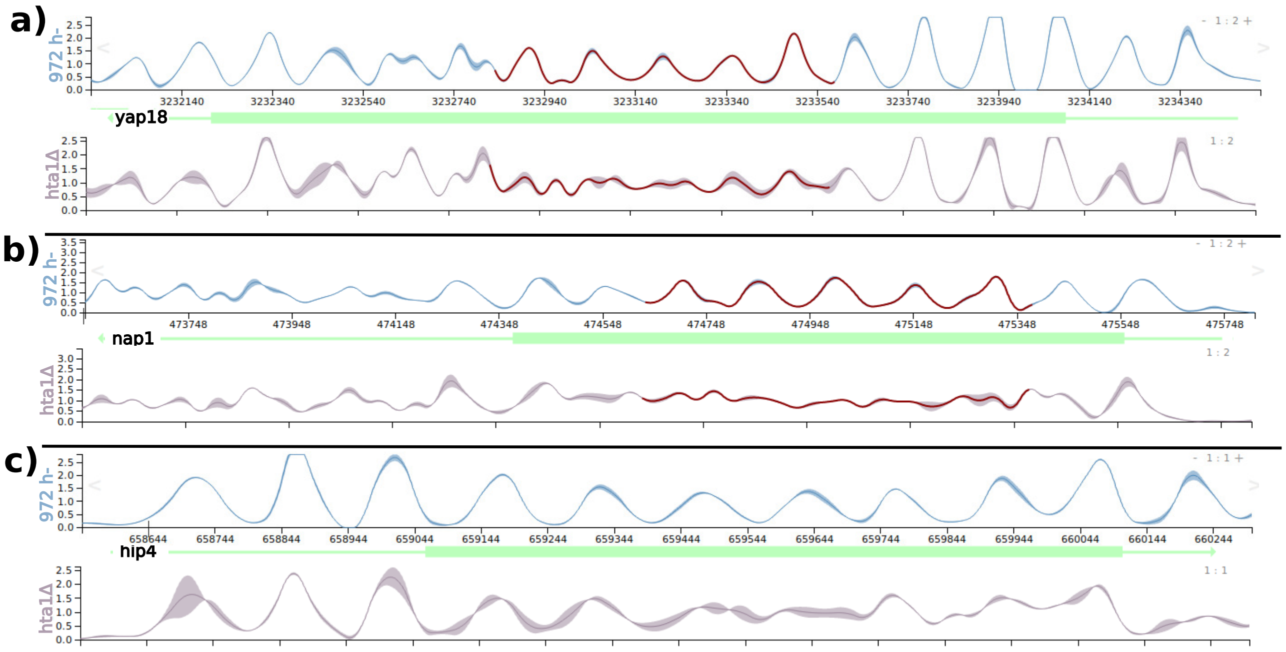

Fig. 9. Loss of positioning of five nucleosomes of the yap18 (a) and nap1 (b) genes on hta1Δ with respect to the 972 h- strain, as detected by BWT searches `abcba*5` and `b*25` ($v = 3$). c) Visual inspection of hip4, also annotated with DNA replication-independent nucleosome assembly, indicates misplaced nucleosomes.

with the original data facilitates further analyses such as the anomalous variation of a region of the *pfl3* gene between the two biological replicates (Fig. 10, notice the shaded area in the curve).

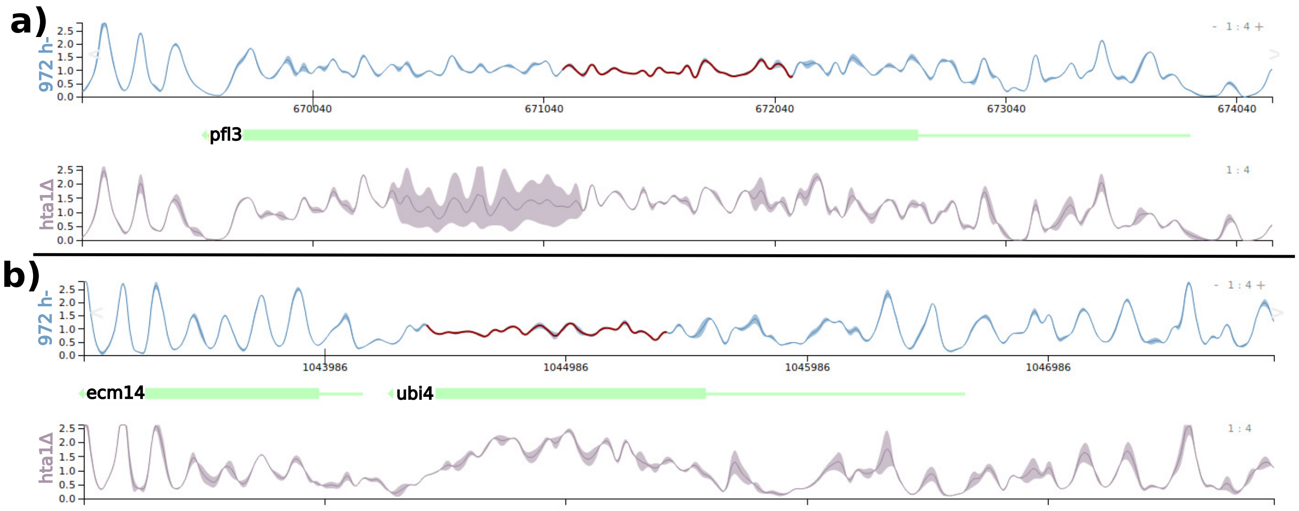

Fig. 10. Higher nucleosomal occupancy of the pfl3 (a) and ubi4 (b) genes on the S. pombe hta1△ mutant relative to 972h- cells. Variation between duplicates is unusually high on a region of the pfl3 gene in the hta1△ mutant.

6 Conclusion

Discretisation has been used on several pattern analysis methods to reduce complexity, taking care that sensitivity is not affected. For example, it has been successfully applied to biclustering algorithms (Prelic *et al.*, 2006), with only two discrete or binary levels. In this case, we detected no significant loss of recovery power in peak retrieval from nucleosome maps compared with a slope-change algorithm (Chen *et al.*, 2013). Pre-processing parameters depend on the data to be analysed. Although a discretisation of 30 bps windows on three percentile ranges is generically sound, variations on defaults can improve sensitivity and performance in specific cases. For example, RNA-seq data of organisms with large genomes could benefit from larger windows (100 to 1,000 bps) and only two percentile ranges.

The visual interface has been designed to provide a means for the analyst to carry out visual discourse analysis. We adhere to a recent trend in visual analytics (Endert *et al.*, 2014) which is the transition from a 'human in the loop' philosophy to a 'human *is* the loop' approach. That is to say, the analyst's work processes are understood and the visual analytical tools facilitate the interactive process through which analysis is performed. Therefore Nucleosee provides a flexible search for patterns that can be visualised in real time along with the original data, genomic annotations and functional enrichment, seeking to replicate a classical analyst's reasoning-based workflow. The link between the results of analyses and the original data, although it may appear trivial, is lacking in several currently available approaches, despite its value in confirming that findings actually fit the data. We believe that leaving this confirmation to the analyst, as part of a seamless dialogue with data and analysis is key for pattern discovery.

Acknowledgements

We would like to thank Eamonn Maguire for initial discussions on the application of BWT to coverage data, and Alejandro Benito for testing the production tool.

Funding

This work was supported by the Government of Spain, Ministerio de Economía y Competitivad (Ref. BFU2014-52143-P).

References

Adjeroh, D., Zhang, Y., Mukherjee, A., Powell, M., and Bell, T. (2002). DNA sequence compression using the Burrows-Wheeler Transform. In *Proceedings. IEEE Computer Society Bioinformatics Conference*, pages 303–313.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Epping, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., and Sherlock, G. (2000). Gene ontology: tool for the unification of biology. *Nature Genetics*, **25**(1), 25–29.

Burrows, M. and Wheeler, D. J. (1994). A block-sorting lossless data compression algorithm. Technical report, Systems Research Center.

Chen, K., Xi, Y., Pan, X., Li, Z., Kaestner, K., Tyler, J., Dent, S., He, X., and Li, W. (2013). DANPOS: dynamic analysis of nucleosome position and occupancy by sequencing. *Genome research*, **23**(2), 341–51.

Compeau, P. and Pevzner, P. (2014). *Bioinformatics algorithms: an active learning approach*. Active Learning Publishers.

Endert, A., Hossain, M. S., Ramakrishnan, N., North, C., Fiaux, P., and Andrews, C. (2014). The human is the loop: new directions for visual analytics. *Journal of intelligent information systems*, **43**(3), 411–435.

González, S., García, A., Vázquez, E., Serrano, R., Sánchez, M., Quintales, L., and Antequera, F. (2016). Nucleosomal signatures impose nucleosome positioning in coding and noncoding sequences in the genome. *Genome research*, **26**(11), 1532–1543.

Jiang, C. and Pugh, B. F. (2009). Nucleosome positioning and gene regulation: advances through genomics. *Nature Reviews Genetics*, **10**(3), 161–172.

Kent, W. J., Sugnet, C. W., Furey, T. S., Roskin, K. M., Pringle, T. H., Zahler, A. M., and Haussler, D. (2002). The human genome browser at UCSC. *Genome research*, **12**(6), 996–1006.

Kerpedjiev, P., Abdennur, N., Lekschas, F., McCallum, C., Dinkla, K., Strobelt, H., Luber, J. M., Ouellette, S. B., Ahzir, A., Kumar, N., Hwang, J., Alber, B. H., Pfister, H., Mirny, L. A., Park, P. J., and Gehlenborg, N. (2017). HiGlass: Web-based visual comparison and exploration of genome interaction maps. *bioRxiv*, page 121889.

Kim, D., Langmead, B., and Salzberg, S. L. (2015). HISAT: a fast spliced aligner with low memory requirements. *Nature Methods*, **12**(4), 357–360.

Langmead, B. and Salzberg, S. L. (2012). Fast gapped-read alignment with Bowtie 2. *Nature methods*, **9**(4), 357–9.

Lekschas, F., Bach, B., Kerpedjiev, P., Gehlenborg, N., and Pfister, H. (2017). HiPiler: Visual Exploration of Large Genome Interaction Matrices with Interactive Small Multiples. *IEEE Transactions on Visualization and Computer Graphics*, **PP**(99), 1.

Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, **25**(14), 1754–1760.

Moyle-Heyrman, G. and Zaichuk, T. and Xi, L. and Zhang, Q. and Uhlenbeck, O. C. and Holmgren, R. and Widom, J. and Wang, J.-P. (2013). Chemical map of Schizosaccharomyces pombe reveals species-specific features in nucleosome positioning. *Proceedings of the National Academy of Sciences*,**110**(50), 20158–20163.

Nicol, J. W., Helt, G. A., Blanchard, S. G., and Raja, A. (2009). The Integrated Genome Browser: free software for distribution and exploration of genome-scale datasets. *Bioinformatics*, **25**(20), 2730–2731.

Prelic, A., Bleuler, S., Zimmermann, P., Wille, A., Buhlmann, P., Gruissem, W., Hennig, L., Thiele, L., and Zitzler, E. (2006). A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, **22**(9), 1122–1129.

Skinner, M. E., Uzilov, A. V., Stein, L. D., Mungall, C. J., and Holmes, I. H. (2009). JBrowse: a next-generation genome browser. *Genome research*, **19**(9), 1630–8.

Teif, V. B. (2016). Nucleosome positioning: resources and tools online. *Briefings in Bioinformatics*, **17**(5), 745–757.

Wang, Z. (2009). RNA-Seq, a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, pages 1–7.

Westesson, O., Skinner, M., and Holmes, I. (2013). Visualizing next-generation sequencing data with JBrowse. *Briefings in Bioinformatics*, **14**(2), 172–177.

Yardimci, G. G. and Noble, W. S. (2017). Software tools for visualizing Hi-C data. *Genome biology*, **18**(1), 26.