Genome Analysis

Genome-wide search of nucleosome patterns using visual analytics

Rodrigo Santamaría¹,*, Roberto Therón¹, Laura Durán², Alicia García², Sara González², Mar Sánchez² and Francisco Antequera²

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

*To whom correspondence should be addressed.

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: rodrig@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.
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 \( A = a_0, \ldots, a_n \) be the original sequence of numbers. Let \( w \) be the window size and \( d \) the number of discrete levels. The discretised sequence \( S = s_0, \ldots, s_m \) can be obtained with Eq. 1:

\[
  s_k = p_d\left(\sum_{i=1}^{k+1}w_i\right), \quad k = 0, \ldots, m
\]

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).

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).

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 \( ab\overline{c}ba \) pattern with Nucleosee should generate a genome-wide map of nucleosomal occupancy. To test this possibility, we allowed...
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

\[
\begin{align*}
\text{procedure } & bwtSearch(bwt, fo, sa, pattern, v) \\
\text{text: } & \text{pre-processed genome as discretised levels} \\
bwt: & \text{BWT sorted list of cyclic rotations of text} \\
& \text{for first occurrence of each symbol on bwt} \\
sa: & \text{integer array with the genomic positions of the BWT suffixes} \\
pattern: & \text{character string to search} \\
v: & \text{integer with the number of variations from pattern} \\
\end{align*}
\]

\[
\begin{align*}
\text{step} & \leftarrow |\text{pattern}|/(v + 1) \triangleright \text{minimal exact length for } v \text{ mismatches} \\
\text{matches} & \leftarrow \{\} \\
\text{for } i = 0, i \leq |\text{pattern}|; i \text{ increases by step } & \text{do} \\
\text{seed} & \leftarrow \text{pattern}[i: \text{step}] \\
\text{seed_matches} & \leftarrow \{\} \triangleright \text{search for seeds of minimal exact length} \\
\text{top} & \leftarrow 0 \\
\text{bottom} & \leftarrow |\text{bwt}| - 1 \\
\text{while } \text{top} \leq \text{bottom} & \triangleright \text{BWT exact search} \\
& \text{if } \text{seed} \text{ has more symbols then} \\
& \text{symbol} & \leftarrow \text{next letter in seed} \\
& \text{for } s \text{ in seed} & \text{times symbol occurs in bwt[top]} \\
& \text{top} & \leftarrow \text{top} + s \\
& \text{bottom} & \leftarrow \text{bottom} + s \\
& \text{else} & \text{add sa[\text{top} : \text{bottom} + 1] to seed_matches} \\
\text{for } pos \text{ in seed_matches do} & \triangleright \text{seed extension} \\
& \text{matches} & \leftarrow \text{mismatches of seed in text}[pos : pos + |\text{pattern}|] \\
& \text{if } \text{matches} \leq v & \text{then add pos to matches} \text{ end if} \\
\end{align*}
\]

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 $\text{abcba}(2)$ patterns with two allowed variations. DANPOS results for a height cut-off of 1 (red), and $S.\text{pombe}$ nucleosomal occupancy as detected by MNase analysis (green) (González et al., 2016) (section of chromosome 1, between 2,425,000 and 2,435,000).

Fig. 5. Visual analytics design of the NucleoSee browser on three levels. The narrative zoom search for three nucleosomes matching exactly the pattern $\text{abcba}^*3$ is shown (in red). Notice that, for example, the higher nucleosome in blue right before the match responds to pattern $\text{aacca}$ instead of $\text{abcba}$ 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). This website also 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 (goda 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 $S.\text{pombe}$, 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
Nucleosee is a tool that can be used to search for nucleosome patterns in genomes. It is designed to allow users to search for nucleosome patterns (nucleosomes) with high efficiency and accuracy. The tool is capable of searching for nucleosome patterns across multiple datasets and can handle large genomes with memory consumption usually being a bottleneck. The tool can perform exact searches, allowing users to identify nucleosome patterns with high precision.

Nucleosee includes a feature that allows users to search for isolated nucleosomes, which are nucleosomes that are not flanked by other nucleosomes. This feature can be useful for identifying regions of the genome where nucleosome positions are not conserved.

One example of a possible search is for isolated nucleosomes, which are 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 (\(abcba+a^5(6)\)) that are delocalised in the mutant strain \(hal1\Delta\ (b^25)\), allowing three single-character deviations from the pattern (\(v = 3\)). 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 \(2.6 \times 10^5\)). 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 \(972\ b\) and \(hal1\Delta\) 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 at 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 wta*a5 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 pf3 gene between the two biological replicates (Fig. 10, notice the shaded area in the curve).
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 Nucleos 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 Competitividad (Ref. BFU2014-52143-P).

References


Fig. 10. Higher nucleosomal occupancy of the pfl3 (a) and ubi4 (b) genes on the S. pombe hta1Δ mutant relative to 972h– cells. Variation between duplicates is unusually high on a region of the pfl3 gene in the hta1Δ mutant.


