Bioinformatics, 33(4), 2017, 514–521
doi: 10.1093/bioinformatics/btw683
Advance Access Publication Date: 5 December 2016
Original Paper

OXFORD

Sequence analysis

Fast motif matching revisited: high-order PWMs, SNPs and indels

Janne H. Korhonen[1,2,3,*], Kimmo Palin[4], Jussi Taipale[5] and Esko Ukkonen[2,3]

[1]School of Computer Science, Reykjavík University, Reykjavík, Iceland, [2]Helsinki Institute for Information Technology HIIT, Helsinki, Finland, [3]Department of Computer Science, [4]Genome-Scale Biology Research Program, Research Programs Unit and [5]Department of Biosciences and Nutrition, Karolinska Institutet, Genome Scale Biology Program, Faculty of Medicine, University of Helsinki, Helsinki, Finland

*To whom correspondence should be addressed

Associate Editor: Alfonso Valencia

Received on June 22, 2016; revised on September 25, 2016; editorial decision on October 21, 2016; accepted on October 27, 2016

Abstract

Motivation: While the position weight matrix (PWM) is the most popular model for sequence motifs, there is growing evidence of the usefulness of more advanced models such as first-order Markov representations, and such models are also becoming available in well-known motif databases. There has been lots of research of how to learn these models from training data but the problem of predicting putative sites of the learned motifs by matching the model against new sequences has been given less attention. Moreover, motif site analysis is often concerned about how different variants in the sequence affect the sites. So far, though, the corresponding efficient software tools for motif matching have been lacking.

Results: We develop fast motif matching algorithms for the aforementioned tasks. First, we formalize a framework based on *high-order position weight matrices* for generic representation of motif models with dinucleotide or general q-mer dependencies, and adapt fast PWM matching algorithms to the high-order PWM framework. Second, we show how to incorporate different types of *sequence variants*, such as SNPs and indels, and their combined effects into efficient PWM matching workflows. Benchmark results show that our algorithms perform well in practice on genome-sized sequence sets and are for multiple motif search much faster than the basic sliding window algorithm.

Availability and Implementation: Implementations are available as a part of the MOODS software package under the GNU General Public License v3.0 and the Biopython license (http://www.cs.helsinki.fi/group/pssmfind).

Contact:janne.h.korhonen@gmail.com

1 Introduction

Position weight matrices (Gribskov *et al.*, 1987; Henikoff *et al.*, 1990; Stormo *et al.*, 1982) are a simple model of signals in sequences called *motifs* that are used in particular in the context of biological sequence data to model e.g. transcription factor binding sites in DNA. Specifically, the idea is to model a sequence motif of length m over alphabet Σ as an $m \times |\Sigma|$ matrix M such that the the fit between the model M and a sequence $u = u_1 u_2 \dots u_m$ is measured by the *match score*

$$M[u] = M(1, u_1) + M(2, u_2) + \cdots + M(m, u_m)\,.$$

The weights in the matrix M may be interpreted in various ways, for example, as describing the binding energy between a transcription factor and DNA, or as log-likelihood presentation of a sequence of

© The Author 2016. Published by Oxford University Press. All rights reserved. For Permissions, please e-mail: journals.permissions@oup.com

Downloaded from https://academic.oup.com/bioinformatics/article/33/4/514/2726114 by guest on 21 August 2022

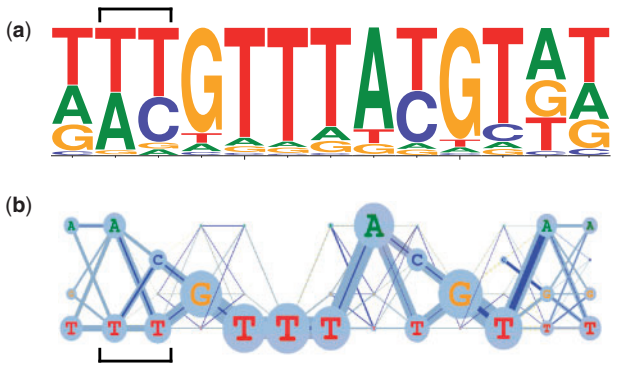


Fig. 1. Example motif from our test set, modelled as (**a**) a 0-order PWM (PWM logo) and (**b**) a 1st-order Markov model (a 'riverlake' logo (Morgunova *et al.*, 2015)). A first-order Markov model can capture dinucleotide preferences of motifs that cannot be represented with a 0-order PWM (Morgunova *et al.*, 2015; Nitta *et al.*, 2015). For example, the riverlake logo (b) shows that dinucleotides AT and TC are preferred over AC and TT at the second and third positions, and that this preference is not and cannot be represented by the PWM model (a)

independent multinomially distributed random variables (Stormo, 2000).

From a computational perspective, there are two main tasks involved in the use of PWMs as motif model. First is the *learning* task, where one wants to derive the PWM weights from empirical data representing the motif of interest. The second is the task of *predicting the sites* for a motif, where one wants to find the occurrence sites of a given motif represented by a PWM in a longer sequence, such as a chromosome or a complete genome. Formally, for a longer sequence $s = s_1 s_2 \dots s_n$, we want to find sequence positions j where the match score exceeds some threshold T, that is, we have $M[s_j s_{j+1} \dots s_{j+m-1}] \geq T$. This *position weight matrix matching problem* is obviously a weighted variant of the basic string matching (key-word matching) problem. Using techniques from string matching algorithms, the PWM matching problem can be solved significantly faster than by using the brute-force sliding window algorithm that evaluates the match score explicitly at each sequence position (see Section 2).

1.1 High-order PWM matching

In this work, we develop motif matching algorithms for high-order PWMs. There has been recent interest towards models for transcription factor binding site motifs that are more complicated than the basic PWM (Weirauch *et al.*, 2013). An obvious generalization of PWMs is to include *dinucleotide* effects into the model, that is, dependencies between adjacent positions (see Fig. 1), either by directly formulating the model in terms of additive dinucleotide effects or by modelling the signal as a 1st-order Markov model. Indeed, many authors have suggested models of these types, studied the corresponding learning problem and evaluated their methods on empirical biological data (Jolma *et al.*, 2013; Kulakovskiy *et al.*, 2013; Mathelier and Wasserman, 2013; Zhao *et al.*, 2012). Moreover, dinucleotide models are starting to appear in databases such as JASPAR (Mathelier *et al.*, 2016) and HOCOMOCO (Kulakovskiy *et al.*, 2016). In validation tests, models with dinucleotide effects are often found better than standard PWMs, and accounting for dinucleotides is important for recognition of some transcription factor binding sites (Berger *et al.*, 2006; Man and Stormo, 2001; Nitta *et al.*, 2015; Siebert and Soeding, 2016; Zhao *et al.*, 2012). It is also worth noting that various other ways to generalize upon the PWM framework have been suggested (see Section 2).

Consider a generalization of the PWM model where we represent a motif of length m as a $(m-1) \times |\Sigma|^2$ matrix M so that the the fit between the model M and a sequence $u = u_1 u_2 \dots u_m$ is measured as

$$M[u] = \sum_{i=1}^{m-1} M(i, u_i u_{i+1}) .$$

This is a natural additive scoring scheme to address dinucleotide effects in PWM-like models, used, for example, by Kulakovskiy *et al.* (2013). However, some other dinucleotide models are not represented in terms of additive weights, such as the transcription factor flexible models in the JASPAR database (Mathelier *et al.*, 2016). To unify and generalize the representation of this class of models, we give in Section 3 a framework based on *high-order position weight matrices* as a generic model for motifs with dinucleotide dependencies and, more generally, with q-mer dependencies ($q = 0$ corresponds to the standard PWM). The framework uniformly represents different motif models as additive scoring matrices, including, as we show, probabilistic models such as Markov models of order 1 and higher. Using the framework, we obtain a natural formulation for the corresponding motif occurrence prediction task as *a high-order PWM matching problem*.

Section 4 gives the main result of this paper. We generalize the fast position weight matrix matching algorithms of Pizzi *et al.* (2011) to high-order PWM matching, which gives us almost-linear time matching that remains efficient even for large motif sets and genome-scale sequence data.

Overall, we obtain a flexible and computationally efficient framework for motif occurrence detection that can support various existing motif models, such as the additive matrix models of Zhao *et al.* (2012) and Kulakovskiy *et al.* (2013) and the 1st-order Markov models of Jolma *et al.* (2013) and Mathelier and Wasserman (2013). We finally stress that we do not introduce new models for transcription factor binding site motifs, but rather an efficient framework and algorithms to predict occurrence sites for some existing and future motif models.

1.2 Sequence variants: SNPs, insertions and deletions

As an additional contribution, we consider motif matching for target sequences that are annotated with *variants*, representing biological features such as single-nucleotide polymorphisms (SNPs), insertions and deletions. It is known, for example, that SNPs affecting binding motifs of transcription factors can alter affinity, and be a significant mechanism in the development of cancer (Tuupanen *et al.*, 2009). Thus, we are interested in identifying variants and combinations of variants that create new matches between the target sequence and motif models.

We consider this problem in a framework in which a contiguous substring $s_i s_{i+1} \dots s_j$ of the target sequence s is replaced with another string u of any length; this allows us to model SNPs, insertions and deletions as well as more general sequence variants (see Fig. 2). We present a simple algorithm that computes, for a PWM or a high-order PWM, all significant matches with the target sequence where one or more of the variants affects the match score (Section 5).

1.3 Implementation and experiments

We have implemented the algorithms presented in this paper as a part of the MOODS software suite (Korhonen *et al.*, 2009), freely available online (http://www.cs.helsinki.fi/group/pssmfind/). The implementations have been written in C++ with performance on large-scale data-sets in mind, and include Python interfaces (Generated using SWIG (http://www.swig.org)) for integration into more complicated workflows.

Downloaded from https://academic.oup.com/bioinformatics/article/33/4/514/2726114 by guest on 21 August 2022

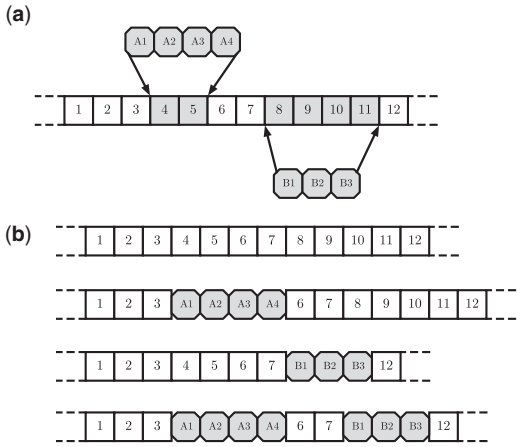


Fig. 2. (**a**) A sequence annotated with variants. (**b**) Possible modified sequences generated by applying the variants to the original sequences

We present benchmarks for our implementations in Section 6, demonstrating that the algorithms also perform well in practice. As an illustrative example, finding all matches for a set of 78 first-order PWMs from a chromosome-sized sequence with nearly two million variants (human chromosome 22 with 1 811 391 variants (The 1000 Genomes Project Consortium, 2015)) corresponding to P-value 10^{-3} took 3 min and 18 s on a standard laptop computer, not including the time needed for reading the inputs.

2 Related work

2.1 Transcription factor binding site models

Rapidly increasing sequence data has made it feasible to learn motif models that are more complex than the basic PWM. The focus in these developments has been on the model specification, learning and validation; see, for example, the recent DREAM5 TF-DNA Motif Recognition Challenge (Weirauch *et al.*, 2013). In addition to the dinucleotide models that form the basis for the present work, there have been proposals with more global dependencies (Mordelet *et al.*, 2013; Santolini *et al.*, 2014; Sharon *et al.*, 2008; Siddharthan, 2010) and models based on position-invariant scoring of longer q-mers (Annala *et al.*, 2011). However, from the perspective of matching algorithms, models with global or long-distance statistical dependencies are more difficult to deal with; compare the present work with e.g. Giaquinta *et al.* (2013; 2014).

2.2 Position weight matrix matching algorithms

Research into fast algorithms for PWM matching is generally based on exploiting *thresholding*, that is, instead of using the sliding window approach to explicitly evaluate the PWM match score at each position, the algorithms attempt to identify positions where the score exceeds a given threshold without explicit computation. This research can be roughly divided into two main branches; first of these is *index-based algorithms*, where the idea is to preprocess the input sequences to an index structure such as suffix tree (Dorohonceanu and Nevill-Manning, 2000) or suffix array (Beckstette *et al.*, 2004) in order to facilitate sub-linear time motif matching. On the other hand, *online algorithms* rely on a linear scan over the input sequence, but usually speed up the search using ideas from classical string matching algorithmics (Liefooghe *et al.*, 2009; Pizzi *et al.*, 2011; Salmela and Tarhio, 2007; Wu *et al.*, 2000). The review by Pizzi and Ukkonen (2008) gives a more thorough overview of various ideas suggested in this context. From a more practical perspective, these fast algorithms have been implemented in software suites such as PoSSuMSearch (Beckstette *et al.*, 2004) and MOODS (Korhonen *et al.*, 2009), demonstrating that carefully designed algorithms can give significant advantages over the sliding window approach.

2.3 SNPs and PWM matching

The analysis of sequence variants in the context of motif analysis has been mostly focused on identification of regulatory SNPs (rSNPs), that is, SNPs that affect the transcription factor DNA-binding affinities (Andersen *et al.*, 2008; Macintyre *et al.*, 2010; Riva, 2012; Thomas-Chollier *et al.*, 2011; Zuo *et al.*, 2015). The existing tools typically analyze a single SNP at a time, computing PWM scores for both alleles and performing appropriate statistical testing to determine whether the effect of the SNP is significant. The statistical testing generally makes these tools computationally demanding; e.g. Zuo *et al.* (2015) report running time of over 7 min for 26 100 SNPs and 10 motifs even when using parallel processing. By contrast, the approach presented in this paper supports analysing joint effects of multiple sequence variantions of more general form, but does not enable statistical testing, making it more appropriate for *discovery* of potentially significant sequence variants in very large datasets.

3 High-order PWMs

In this section, we first define the notion of high-order PWM, which is sufficient to capture additive models, and then show how the high-order PWMs can be used as a representation for probabilistic models, such as the 1st-order Markov models.

3.1 Basic framework

We define a *qth-order position weight matrix* of length m in alphabet Σ as a real-valued matrix $M = (M(i,u))$ of size $(m-q) \times |\Sigma|^{q+1}$. The columns of M are thought to be indexed with $(q+1)$-tuples of symbols $u \in \Sigma^{q+1}$, that is, by $(q+1)$-mers. For biological applications we generally use the DNA alphabet $\Sigma = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$.

As with standard PWMs, the similarity between a qth-order PWM M and a string $u = u_1 \dots u_m$ is measured by the *match score*, defined as

$$M[u] = \sum_{i=1}^{m-q} M(i, u_i \dots u_{i+q}) . \quad (1)$$

Here the key distinction from standard PWMs is that the column scores depend on partially overlapping $(q+1)$-mers of u instead of the individual characters.

3.2 Probabilistic models as high-order PWMs

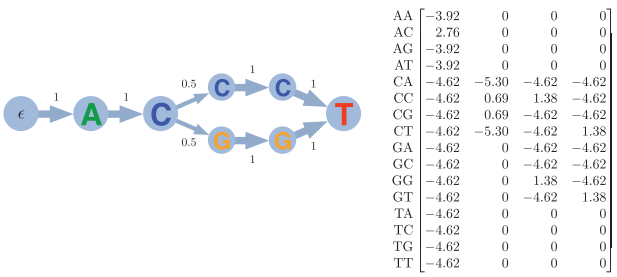
A probabilistic motif model Φ of length m models a signal as a sequence of random variables $X = X_1 X_2 \dots X_m$ over alphabet Σ, assigning a probability $P_\Phi(X = u)$ for each $u \in \Sigma^m$. We say that a probabilistic motif model Φ is *order-q decomposable* if there are functions $\Phi_i : \Sigma^{q+1} \to R$ such that the probability that Φ generates a string $u = u_1 u_2 \dots u_m$ can be written as

$$P_\Phi(X = u) = \prod_{i=1}^{m-q} \Phi_i(u_i u_{i+1} \dots u_{i+q}) .$$

That is, the probability decomposes in terms of the length-$(q+1)$ substrings of u.

We can now define a high-order PWM representation for an order-q decomposable motif model Φ as follows. Assume that we have a fixed background distribution ϕ over strings of length m, and that ϕ is also order-q decomposable; for example, ϕ can be simply

Downloaded from https://academic.oup.com/bioinformatics/article/33/4/514/2726114 by guest on 21 August 2022



AA	−3.92	0	0	0
AC	2.76	0	0	0
AG	−3.92	0	0	0
AT	−3.92	0	0	0
CA	−4.62	−5.30	−4.62	−4.62
CC	−4.62	0.69	1.38	−4.62
CG	−4.62	0.69	−4.62	−4.62
CT	−4.62	−5.30	−4.62	1.38
GA	−4.62	0	−4.62	−4.62
GC	−4.62	0	−4.62	−4.62
GG	−4.62	0	1.38	−4.62
GT	−4.62	0	−4.62	1.38
TA	−4.62	0	0	0
TC	−4.62	0	0	0
TG	−4.62	0	0	0
TT	−4.62	0	0	0

Fig. 3. A example of 1st-order Markov model representing a sample that consists of sequences ACCCT and ACGGT, and a 1st-order PWM representation of the model computed using (2); missing transitions have been assigned a small probability to account for missing data. In particular, the dependence between positions 3 and 4 cannot be represented by a 0-order model

the uniform distribution over Σ^m. The qth-order PWM representation for Φ is now an $(m-q)\times|\Sigma|^{q+1}$ matrix with entries

$$M(i,u)=\log\frac{\Phi_i(u)}{\phi_i(u)}\,,\quad 1\le i\le m-q\,,\quad u\in\Sigma^{q+1}\,. \tag{2}$$

See Figure 3 for an example of this derivation for a simple 1st-order Markov model.

Note that the assumption that the background distribution ϕ is also order-q decomposable is made purely for technical reasons, as we have to split the background effect between the $m-q$ columns of the high-order PWM. For example, for the position-independent multinomial background model defined by probabilities $P_\phi(a)$ for $a\in\Sigma$, we can set $\phi_1(u)=\prod_{i=1}^{q+1}P_\phi(u_i)$ and $\phi_i(u)=P_\phi(u_{q+1})$ for $i\ge 2$, and for the uniform background distribution, we can set $\phi_i(u)=((m-q)|\Sigma|^m)^{-1}$ for all $i=1,2,\ldots,m-q$ and $u\in\Sigma^{q+1}$

3.3 Matching and threshold selection

The *matching problem* for qth-order PWMs is defined in a natural way; we are given qth-order PWM M, a sequence $s=s_1s_2\ldots s_n$, and a threshold T, and the task is to find all positions i such that $M[s_is_{i+1}\ldots s_{i+m-1}]\ge T$. Note that the basic *sliding window algorithm* for PWMs, corresponding to the direct evaluation of (1) in each position of s, can be adapted to high-order PWMs to get a simple $O(nm)$ time algorithm for this task. However, this is fairly slow for large datasets, especially when dealing with large number of motifs.

The selection of the threshold T can be done in the same way as for PWMs; a standard approach is to use P-values to control the confidence of the found matches. That is, given a P-value P^*, we select a threshold T such that the probability for a string u generated by a background distribution ϕ having score $M[u]\ge T$ is P^*. This threshold can be computed using a well-known dynamic programming algorithm, similarly as with 0-order PWMs (Staden, 1989; Wu *et al.*, 2000).

4 Fast high-order PWM matching

4.1 Definitions and preliminaries

4.1.1 Submatrices

For a qth-order PWM M of length m and $1\le j<k\le m$, we define the submatrix $M_{j\ldots k}$ as a qth-order PWM of length $(k-j+1)$ with entries

$$M_{j\ldots k}(i,a_1\ldots a_{q+1})=M(j+i-1,a_1\ldots a_{q+1})$$

for $i=1,2,\ldots,k-j-q+1$ and $a_1,\ldots,a_{q+1}\in\Sigma$.

4.1.2 Computing maximum scores

For a qth-order PWM M, we will often need to know the maximum possible score M can give to a string $u\in\Sigma^m$, that is, $\max_{u\in\Sigma^m}M[u]$. While for the case of $q=0$, corresponding to standard PWMs, this score can be simply computed by taking the maximum score from each column, there are interdependencies between positions in qth-order PWM that make things more complicated for $q\ge 1$.

To compute the maximum score for a qth-order PWM, we use a simple dynamic programming algorithm. For all $u=u_1\ldots u_q\in\Sigma^q$, let

$$\begin{aligned}M^*(1,u)&=\max_{a\in\Sigma}M(1,au_1\ldots u_q)\,,\\ M^*(i,u)&=\max_{a\in\Sigma}[M^*(i-1,au_1\ldots u_{q-1})+M(i,au_1\ldots u_q)]\,,\end{aligned} \tag{3}$$

where $i=2,3,\ldots,m-q$. The maximum score M can give to a string of length m is now

$$\max_{u\in\Sigma^q}M^*(m-q,u)\,.$$

This value can be computed in time $O(|\Sigma|^{q+1}(m-q))$ using (3).

4.2 Fast matching via filtering

4.2.1 Filtering via exact string matching

A basic observation is that given a qth-order PWM M and a threshold T, these completely determine all length m strings with match score at least T. Thus, qth-order PWM matching reduces to multi-pattern exact string matching; we can simply generate the set

$$\Lambda=\{u\in\Sigma^m : M[u]\ge T\}\,,$$

and use a fast multi-matrix string matching algorithm to find all occurrences of strings in Λ from an input sequence.

However, for long matrices M this can become impractical, as the size of set Λ can grow exponentially in m. To circumvent this problem, we use *filtration via multi-pattern string matching*, similarly to Pizzi *et al.* (2011); note here that our filtering scheme is lossless, i.e. no occurrences passing the threshold will be lost:

1. We first select a *window* of some fixed width ℓ and starting at some position j; the idea is to select the starting position j such that the substring $u_ju_{j+1}\ldots u_{j+\ell-1}$ determines most of the score $M[u]$ for any $u\in\Sigma^m$.
2. We then construct a *filter set* for M, defined as

$$\Gamma=\{u\in\Sigma^\ell : M[vuw]\ge T \text{ for some } v\in\Sigma^{j-1},\ w\in\Sigma^{m-\ell-j+1}\}.$$

   Intuitively, the filter set contains all strings that may occur in the selected window in a high-scoring string.
3. Then use an exact multi-pattern string matching algorithm to find in the input sequence s all occurrences of strings in Γ, in time linear in $|s|$.
4. For each position i of s where some string from Γ occurs, check whether

$$M[s_{i-j+1}\ldots s_{i-j+m}]\ge T\,.$$

We next discuss each of these steps in detail.

Downloaded from https://academic.oup.com/bioinformatics/article/33/4/514/2726114 by guest on 21 August 2022

4.2.2 Window selection

For qth-order PWM M and background distribution P_ϕ, we define the *expected loss* of the score as

$$L(M) = E[M^* - M[u]] = M^* - E[M[u]]\,,$$

where $u \in \Sigma^m$ is generated from the background distribution P_ϕ and $M^* = \max_{u\in\Sigma^m} M[u]$. Note that M^* can be evaluated in time $O(|\Sigma|^{q+1}m)$ using (3), and by the linearity of expectation

$$E[M[u]] = \sum_{v\in\Sigma^{q+1}} P_\phi(v)\cdot M[v]\,.$$

Thus, $L(M)$ can be evaluated in time $O(|\Sigma|^{q+1}m)$ for fixed M.

To select a filtration window for our algorithm, we first fix the window width ℓ and the background distribution P_ϕ. The parameter ℓ is ideally chosen to be as large as is feasible on the current hardware—the results of Pizzi *et al.* (2011) indicate that $\ell = 7$ or $\ell = 8$ is a good choice on modern hardware, as this means the data structures representing the filter set fit mostly into the fast cache memory—while P_ϕ can be estimated from the input sequence. We select the window of length ℓ by picking the submatrix $M_{j\ldots j+\ell-1}$ that maximizes the expected loss, that is, the window position is selected to be

$$j = \mathrm{argmax}_i L(M_{i\ldots i+\ell-1})\,.$$

This window selection can be implemented to run in time $O(\Sigma^{q+1}(m-\ell)m)$.

4.2.3 Filter set construction

Once we have fixed the window position j, we want to enumerate all strings in the set Γ as defined above. Unlike in the 0-order case, the score inside the filtering window is not completely independent of the score outside the window, but rather we have to account for the interaction occurring at the q positions at the both edges of the filter. To this end, for $v \in \Sigma^q$, we define the *maximum prefix score*

$$P(v) = \max_{u\in\Sigma^{j-1}} M_{1\ldots j+q}[uv]$$

and the *maximum suffix score*

$$R(v) = \max_{u\in\Sigma^{m-\ell-j-1}} M_{j+\ell\ldots m}[vu]\,.$$

Note that both of these can be evaluated for all v in time $O(|\Sigma|^{q+1}m)$ using (3); for $R(v)$, we simply reverse the index ordering.

The filter set Γ for window position j is now

$$\Gamma = \{u \in \Sigma^\ell : P(u_1\ldots u_q) + M_{j\ldots j+\ell-1}[u] + R(u_{\ell-q}\ldots u_\ell) \geq T\}\,. \tag{4}$$

Assuming the prefix and suffix scores are precomputed, then Γ can be constructed in time $O(|\Sigma|^\ell \ell)$ by enumerating all strings $u \in \Sigma^\ell$ and checking for each of them if the condition in (4) is satisfied.

4.2.4 Filtering

To find occurrences of strings $u \in \Gamma$ in the input sequence $s = s_1 s_2 \ldots s_n$, it suffices to use any linear-time multi-pattern string matching algorithm. We use a simple finite state automaton similarly to Pizzi *et al.* (2011); we construct an automaton with an explicit state for each $u \in \Sigma^\ell$, and the automaton takes a transition from state $u_1u_2\ldots u_\ell$ to state $u_2u_2\ldots u_\ell a$ when reading symbol $a \in \Sigma$. For each state u, we record whether $u \in \Gamma$ and the corresponding score $M_{j\ldots j+\ell-1}[u]$.

In cases where the size of the filter set Γ is significantly smaller than $|\Sigma|^\ell$, it might be beneficial to use more space-efficient algorithms such as the Aho-Corasick algorithm (Aho and Corasick, 1975). However, the experimental results of Pizzi *et al.* (2011) suggest that this does not confer much practical benefit, especially in the multi-matrix generalization we discuss below.

4.2.4 Lookahead for off-filter scoring

For each match $s_i s_{i+1}\ldots s_{i+\ell-1} \in \Gamma$, we can verify whether or not this match can be expended to a match of full M by explicitly computing the scores for flanking regions, that is, $M_{1\ldots j}[s_{i-j+1}\ldots s_i]$ and $M_{\ell-1\ldots m}[s_{i+\ell-1}\ldots s_{i-j+m}]$, and checking if

$$M[s_{i-j+1}\ldots s_{i-j+m}] \geq T\,.$$

This can be done in time $O(m-\ell)$, assuming we stored the score from the filtering window in the scanning automaton.

4.2.5 Multi-matrix version

Our algorithm also generalizes in a straightforward way to the case where we are given multiple qth-order PWMs $M_1, M_2, \ldots, M_k$, and we want to find matches for all of the matrices simultaneously. That is, we preprocess each matrix as above, constructing the corresponding filter sets $\Gamma_1, \Gamma_2, \ldots, \Gamma_k$. However, we do the filtration step only once, finding all matches for strings in the set $\cup_{i=1}^k \Gamma_i$; when constructing the finite automaton for filtration, we additionally store information on which of filter sets $\Gamma_1, \Gamma_2, \ldots \Gamma_k$ each strings $u \in \Sigma^\ell$ belongs to.

In practice, this can speed up the search significantly for large k, as we then have to do the $O(n)$ filtration phase only once. Moreover, we note that the matrices $M_1, M_2, \ldots, M_k$ can have different lengths m and orders q, and we can still combine the filtrations steps, as the preprocessing can be done on per-matrix basis.

5 Variants in sequences

5.1 Definitions and preliminaries

5.1.1 Variants

Let Σ be a fixed alphabet and let $s \in \Sigma^n$ be a sequence of length n. In the following, we write $s_{i\ldots j}$ for the substring $s_i s_{i+2}\ldots s_{j-1}$; if $j \leq i$, we mean the empty substring, and for $i \leq 0$ or $j \geq n+2$, we interpret these as if we had $i = 1$ and $j = n+1$ respectively.

A *variant for* s is a string $\mu \in \Sigma^*$, associated with a starting position $b(\mu)$ and end position $e(\mu)$ such that $1 \leq b(\mu) \leq e(\mu) \leq n+1$. We say that s *modified by* μ, denoted by s/μ, is the sequence obtained by replacing the (possibly empty) substring $s_{b(\mu)\ldots e(\mu)}$ of s with the string μ, that is,

$$s/\mu = s_{1\ldots b(\mu)}\mu s_{e(\mu)\ldots n+1}\,.$$

5.1.2 Compatibility

For variants μ and λ, we write $\mu < \lambda$ if $e(\mu) < b(\lambda)$ or $e(\mu) = b(\lambda)$ and either $b(\mu) < e(\mu)$ or $b(\lambda) < e(\lambda)$. Intuitively, this means that both μ and λ can be applied to s simultaneously without ambiguity; we say that μ and λ are *compatible* if $\mu < \lambda$ or $\lambda < \mu$. For compatible variants $\mu^1 < \mu^2 < \cdots \mu^k$, we extend the notion of applying the variants to s in an obvious way; that is, we simultaneously replace substring $s_{b(\mu^i)\ldots e(\mu^i)}$ of s with μ^i for each $i = 1, 2, \ldots, k$. We denote the resulting string by $s/\mu^1, \ldots, \mu^k$.

5.1.3 Matching problem with variants

Let M be a qth-order PWM of length m, and let us fix a threshold T. Given sequence $s = s_1 s_2 \ldots s_n$ and a set of variants $X = \{\mu^1, \mu^2, \ldots, \mu^k\}$

Downloaded from https://academic.oup.com/bioinformatics/article/33/4/514/2726114 by guest on 21 August 2022

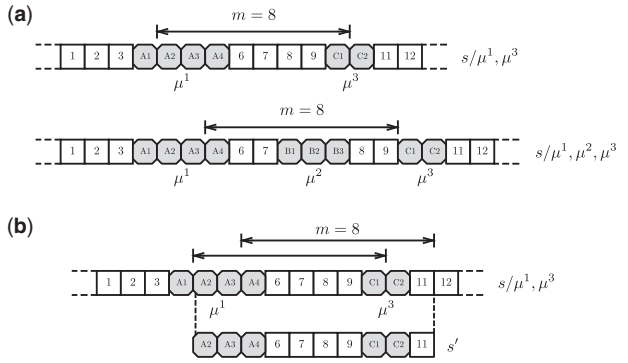


Fig. 4. (**a**) A PWM M of length $m=8$ can have a modified match for $\{\mu^1, \mu^3\}$, but not for $\{\mu^1, \mu^2, \mu^3\}$, as the PWM cannot overlap all three of the variants at the same time. (**b**) A match for M and $\{\mu^1, \mu^3\}$ must overlap the last position of μ^1 and the first position of μ^3, which restricts the search to substring s' of $s/\mu^1, \mu^3$

to s, we say that a *modified match for M and $Y \subseteq X$* is a substring u of s/Y of length m such that (a) $M[u] \geq T$ and (b) u overlaps all of the replacement strings μ for $\mu \in Y$. In case of empty replacement string, i.e. a deletion, we take this requirement to mean that the match must include at least one position before and after the deleted substring. In the *variant matching problem*, we are given a high-order PWM M, threshold T, sequence s and the set of variants X as above, and the task is to find all modified matches in s for M and all subsets of compatible variants $Y \subseteq X$.

5.2 Finding modified matches

5.2.1 Restricting variant combinations

As a basic observation, we note that variants have to be sufficiently close together for there to be a match that overlaps them both. That is, for two variants $\mu < \lambda$, there can be a modified match for M and $\{\mu, \lambda\}$ only if we have $b(\lambda) - e(\mu) \leq m - 2$; otherwise, M is too short to overlap both.

More generally, for a set $X = \{\mu^1, \mu^2, \ldots, \mu^k\}$ of compatible variants with $\mu^1 < \mu^2 < \cdots < \mu^k$, let $b(Y)$ denote the position of last symbol of μ^1 and $e(Y)$ the position of first symbol of μ^k in $s/\mu^1, \ldots, \mu^k$. Any modified match for M and Y in s must clearly overlap these positions in $s/\mu^1, \ldots, \mu^k$, so M can have modified matches for Y only if $e(Y) - b(Y) \leq m - 2$; we call variant sets satisfying this condition *feasible*. Moreover, this implies that the only possible starting positions for a such match in $s/\mu^1, \ldots, \mu^k$ are $e(Y) - m + 1, e(Y) - m + 2, \ldots, b(Y) - 1, b(Y)$. That is, the modified matches in s for M and $\{\mu^1, \ldots, \mu^k\}$ are exactly the matches of M in sequence $(s/\mu^1, \ldots, \mu^k)_{e(Y)-m+1 \ldots b(Y)+m}$ (see Fig. 4(b)).

5.2.2 Branch and scan

Based on the above observations, our basic strategy for the problem is as follows. For any subset $Y \subseteq X$ of variants, we check if Y is feasible, that is, $e(Y) - b(Y) \leq m - 2$. If this is the case, we generate the substring $(s/Y)_{j-m+1 \ldots i+m}$ and find all matches for M in that string using an arbitrary matching algorithm.

However, we do not want to explicitly check all possible $2^{|X|}$ subsets of X, so we use a simple branch-and-bound approach to restrict the search space. Assume that variants X are $\mu^1, \mu^2, \ldots, \mu^k$ and that they are sorted so that $b(\mu^i) \leq b(\mu^j)$ if $i < j$. With this ordering fixed, we say that $Y' \subseteq X$ is an *extension* of $Y \subseteq X$ if (a) $Y' \supseteq Y$ and (b) $i < j$ for any $\mu^i \in Y$ and $\mu^j \in Y' \backslash Y$, that is, all variants in $Y' \backslash Y$ come after variants in Y in the global ordering. Clearly, if Y is not feasible, then all of its extensions are also not feasible.

We now use a simple algorithm that, given a set of variants $Y \subseteq X$, recursively walks through extensions Y' of Y and finds the corresponding modified matches; the recursive search is stopped if a non-feasible variant set Y is encountered. Initially, the algorithm is called for each singleton variant set $Y = \{\mu^j\}$ for $j = 1, 2, \ldots, k$, with the algorithm proceeding as follows:

1. Generate sequence $s' = (s/Y)_{e(Y)-m+1 \ldots b(Y)+m}$ and find all matches for M from s'.
2. Go through the variants $\mu^i \in \{\mu^{\ell+1}, \mu^{\ell+2}, \ldots \mu^k\}$ in order, where μ^ℓ is the last variant in Y:
   a. If μ^i is compatible with μ^ℓ and $e(Y \cup \{\mu^i\}) - b(Y \cup \{\mu^i\}) \leq m - 2$, recursively find all matches for extensions of $Y \cup \{\mu^i\}$.
   b. If $e(Y \cup \{\mu^i\}) - b(Y \cup \{\mu^i\}) > m - 2$, return from recursion ($Y \cup \{\mu^i\}$ is not feasible).

The worst-case running time is still in the order of $2^{|X|}$, but if the variants are sparsely distributed, as is likely the case with real data, the running time will be much smaller. Informally speaking, if no more than K variants occur within a length-m window in s, then the number of variant sets $Y \subseteq X$ that need to be visited by the algorithm is $O(n2^K)$. Moreover, the algorithm can be easily modified to only check sets Y with $|Y| \leq C$ for a constant C.

6 Experimental results

6.1 High-order PWM matching

To evaluate the general performance of our high-order PWM matching framework, we compared the multi-matrix version of our algorithm versus the similar multi-matrix PWM matching algorithm from MOODS, as well as the naive single-matrix PWM matching algorithm that explicitly computes the score at each position. An overview of the results is shown in Figure 5.

The test motifs were obtained by generating both standard PWM and 1st-order PWM representations for 78 putative transcription factor binding sites identified using SELEX (unpublished data). The motif length varied between 7 and 17 as shown in Figure 5(d). As target sequences, we used the 22 autosomal chromosomes from the human genome (assembly GRCh38), obtained from the UCSC Genome Browser database (The Genome Sequencing Consortium, 2001; http://genome.ucsc.edu/). The experiments were performed using a single thread on a 2.53-GHz Intel Xeon processor with 32 GB of memory. We set window width parameter to $\ell = 7$, so the memory usage of the data structures in scanning was very low.

Overall, matching is slightly slower for 1st-order PWMs compared to 0-order PWMs, though never by more than factor 2. Our algorithms are also significantly faster than the naive search; even for thresholds corresponding to $P = 10^{-3}$, extrapolation from running times on single 0-order PWM suggests that running the naive algorithm for the full set of 78 0-order PWMs takes more than 3 times as long as our multi-matrix algorithm for 1st-order PWMs, while for $P = 10^{-5}$ this speed-up factor is over 25.

6.2 Matching with sequence variants

To benchmark our PWM matching on sequence variants, we used our algorithm to find all modified matches for the same 78 1st-order PWMs as above from the human chromosome 22 relative to a set of 1 811 391 variants from the 1000 Genomes Project (The 1000 Genomes Project Consortium, 2015); the thresholds were selected to correspond

Downloaded from https://academic.oup.com/bioinformatics/article/33/4/514/2726114 by guest on 21 August 2022

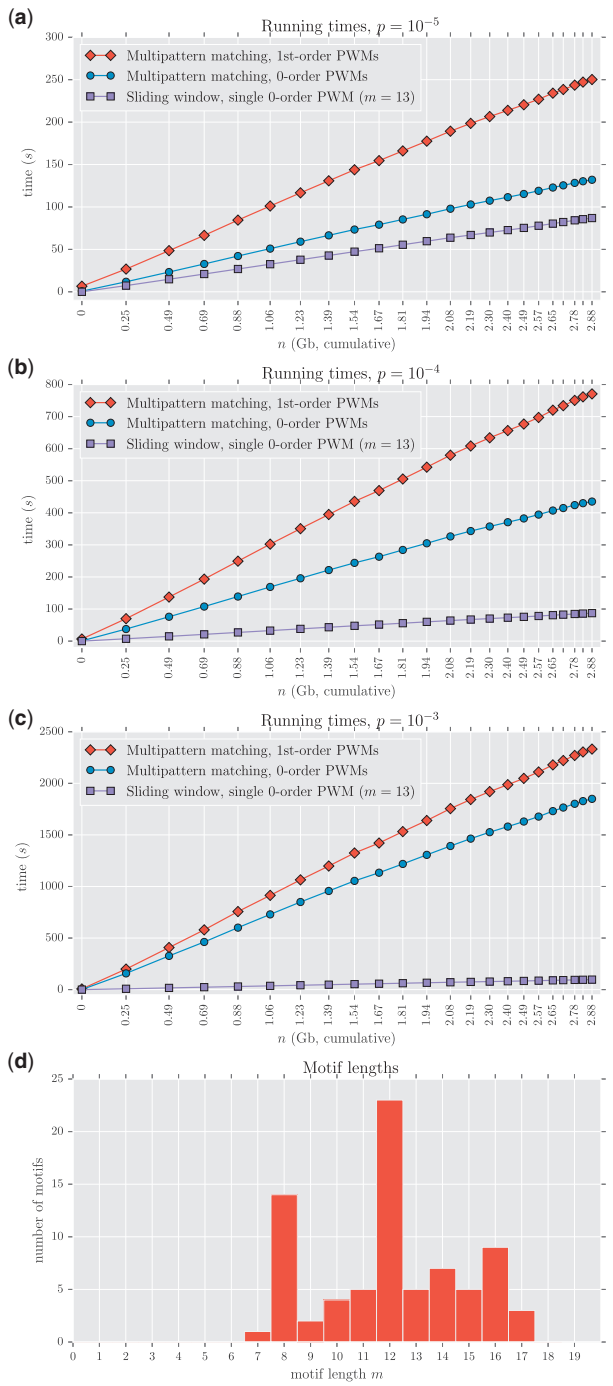


Fig. 5. (**a–c**) Running times for the multi-matrix PWM matching algorithms (78 PWMs) and the sliding window algorithm, i.e. explicit evaluation of the PWM score at each position (one PWM, $m = 13$) over the human genome for three different threshold values. Preprocessing time is included in the time at $n = 0$. Reported times are medians of 10 repeated runs. (**d**) Lengths of the motifs used as test data

to $P = 10^{-3}$. Running the algorithm took 53 seconds on a 2.9-GHz Intel Core i5 processor with 8 GB of memory using a single thread (median of 10 runs, includes preprocessing of the PWMs); this includes the detection of modified matches only, not the detection of the non-modified matches. By comparison, reading the chromosome 22, and the variants from a compressed file, took 208 seconds. However, we note that the used set of variants was fairly sparse, and no matches overlapping more than two variants were found.

7 Conclusions

7.1 Summary of results

We have developed algorithmic tools for motif matching to deal with high-order PWMs and variants in sequences. These algorithms have been implemented as a part of the MOODS software package, and experimental results show that they perform well in practice. Given the toolkit nature of MOODS, we expect that these algorithms should also be readily adaptable as a basis for more complicated sequence analysis workflows. For example, with learning of dinucleotide motif models widely discussed in prior work, we believe that all the pieces are in place for easy transition to dinucleotide models from the zero-order PWMs in predicting binding sites for transcription factors.

7.2 Future work and limitations

As noted in Section 2, a wealth of alternative motif models have been proposed, and hence a major open question is to develop fast motif matching algorithms for models not covered by our high-order PWM framework. A general challenge is to deal with the heterogeneity of the proposals, as the increasingly complicated models do not all fit into a neat combinatorial framework.

Moreover, the approach of this paper may not be the right one for all models. For example, while our algorithm could be extended to models with long-distance dependencies, the locality of our filtering approach might in practice mean that it is no longer very good at separating the occurrences from non-occurrences, and the increased complexity of the preprocessing phase will result in slower running times. One possible research direction is thus the development of more flexible and global filters that can be better adapted to different motif models.

Acknowledgements

We than Jarkko Toivonen and Teemu Kivioja for discussions, and the anonymous reviewers for their valuable comments.

Funding

The research was supported by SYSCOL (a European Union Framework Programme 7 Collaborative Project, 258236). K.P. and J.T. were supported by the Academy of Finland CoE in Cancer Genetics Research (Finnish Center of Excellence Program 2012–2017, 250345), K.P. by NIASC (a NordForsk Nordic Center of Excellence, 62721), J.H.K. by the Icelandic Research Fund (grant 152679-051) and E.U. by a grant VP12014044 from the Leverhulme Trust.

Conflict of Interest: none declared.

References

Aho,A.V. and Corasick,M.J. (1975) Efficient string matching: an aid to bibliographic search. *Commun. ACM*, **18**, 333–340.

Andersen,M.C. *et al.* (2008) In silico detection of sequence variations modifying transcriptional regulation. *PLoS Comput. Biol.*, **4**, 12.

Annala,M. *et al.* (2011) A linear model for transcription factor binding affinity prediction in protein binding microarrays. *PLoS ONE*, **6**, 1–13.

Beckstette,M. *et al.* (2004). PoSSuMsearch: Fast and sensitive matching of position specific scoring matrices using enhanced suffix arrays. In: *German Conference on Bioinformatics*, pp. 53–64.

Berger,M.F. *et al.* (2006) Compact, universal DNA microarrays to comprehensively determine transcription-factor binding site specificities. *Nat. Biotechnol.*, **24**, 1429–1435.

Downloaded from https://academic.oup.com/bioinformatics/article/33/4/514/2726114 by guest on 21 August 2022

Dorohonceanu,B. and Nevill-Manning,C.G. (2000). Accelerating protein classification using suffix trees. In: *8th International Conference on Intelligent Systems for Molecular Biology (ISMB 2000)*, pp. 128–133.

Giaquinta,E. *et al*. (2013) Fast matching of transcription factor motifs using generalized position weight matrix models. *J. Comput. Biol*., **20**, 621–630.

Giaquinta,E. *et al*. (2014) Motif matching using gapped patterns. *Theor. Comput. Sci*., **548**, 1–13.

Gribskov,M. *et al*. (1987) Profile analysis: detection of distantly related proteins. *Proc. Natl. Acad. Sci. U. S. A*., **84**, 4355–4358.

Henikoff,S. *et al*. (1990) Finding protein similarities with nucleotide sequence databases. *Methods Enzymol*., **183**, 111–132.

Jolma,A. *et al*. (2013) DNA-binding specificities of human transcription factors. *Cell*, **152**, 327–339.

Korhonen,J. *et al*. (2009) MOODS: fast search for position weight matrix matches in DNA sequences. *Bioinformatics*, **25**, 3181–3182.

Kulakovskiy,I. *et al*. (2013) From binding motifs in ChIP-Seq data to improved models of transcription factor binding sites. *J. Bioinf. Comput. Biol*., **11**

Kulakovskiy,I.V. *et al*. (2016) HOCOMOCO: expansion and enhancement of the collection of transcription factor binding sites models. *Nucleic Acids Res*., **44**, D116–D125.

Liefooghe,A. *et al*. (2009). Self-overlapping occurrences and Knuth-Morris-Pratt algorithm for weighted matching. In: *3rd International Conference Language and Automata Theory and Applications (LATA 2009)*, pp. 481–492. Springer.

Macintyre,G. *et al*. (2010) is-rSNP: a novel technique for in silico regulatory SNP detection. *Bioinformatics*, **26**, i524–i530.

Man,T.K. and Stormo,G.D. (2001) Non-independence of mnt repressor–operator interaction determined by a new quantitative multiple fluorescence relative affinity (QuMFRA) assay. *Nucleic Acids Res*., **29**, 2471–2478.

Mathelier,A. and Wasserman,W.W. (2013) The next generation of transcription factor binding site prediction. *PLoS Comput. Biol*., **9**

Mathelier,A. *et al*. (2016) Jaspar 2016: a major expansion and update of the open-access database of transcription factor binding profiles. *Nucleic Acids Res*., **44**, D116–D125.

Mordelet,F. *et al*. (2013) Stability selection for regression-based models of transcription factor–DNA binding specificity. *Bioinformatics*, **29**, i117–i125.

Morgunova,E. *et al*. (2015) Structural insights into the DNA-binding specificity of E2F family transcription factors. *Nat. Commun*., **6**.

Nitta,K.R. *et al*. (2015) Conservation of transcription factor binding specificities across 600 million years of bilateria evolution. *eLife*, **4**

Pizzi,C. and Ukkonen,E. (2008) Fast profile matching algorithms – a survey. *Theor. Comput. Sci*., **395**, 137–157.

Pizzi,C. *et al*. (2011) Finding significant matches of position weight matrices in linear time. *IEEE/ACM Trans. Comput. Biol. Bioinf*., **8**, 69–79.

Riva,A. (2012) Large-scale computational identification of regulatory SNPs with rSNP-MAPPER. *BMC Genomics*, **13**, (Suppl 4), S7.

Salmela,L. and Tarhio,J. (2007). Algorithms for weighted matching. In: *14th International Symposium on String Processing and Information Retrieval (SPIRE 2007)*, pp. 276–286. Springer.

Santolini,M. *et al*. (2014) A general pairwise interaction model provides an accurate description of *in vivo* transcription factor binding sites. *PLoS ONE*, **9**, e99015.

Sharon,E. *et al*. (2008) A feature-based approach to modeling protein-DNA interactions. *PLoS Comput. Biol*., **4**, e1000154.

Siddharthan,R. (2010) Dinucleotide weight matrices for predicting transcription factor binding sites: Generalizing the position weight matrix. *PLoS ONE*, **5**

Siebert,M. and Soeding,J. (2016) Bayesian markov models consistently outperform PWMs at predicting motifs in nucleotide sequences. *Nucleic Acids Res*., gkw521.

Staden,R. (1989) Methods for calculating the probabilities of finding patterns in sequences. *Comput. Appl. Biosci. (CABIOS)*, **5**, 89–96.

Stormo,G.D. (2000) DNA binding sites: representation and discovery. *Bioinformatics*, **16**, 16–23.

Stormo,G.D. *et al*. (1982) Use of the 'perceptron' algorithm to distinguish translational initiation sites in *E. coli*. *Nucleic Acids Res*., **10**, 2997–3011.

The 1000 Genomes Project Consortium. (2015) A global reference for human genetic variation. *Nature*, **526**, 68–74.

The Genome Sequencing Consortium. (2001) Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.

Thomas-Chollier,M. *et al*. (2011) Transcription factor binding predictions using TRAP for the analysis of ChIP-seq data and regulatory SNPs. *BMC Genomics*, **6**, 1754–2189.

Tuupanen,S. *et al*. (2009) The common colorectal cancer predisposition SNP rs6983267 at chromosome 8q24 confers potential to enhanced Wnt signaling. *Nat. Genet*., **41**, 885–890.

Weirauch,M.T. *et al*. (2013) Evaluation of methods for modeling transcription factor sequence specificity. *Nat. Biotechnol*., **31**, 126–134.

Wu,T.D. *et al*. (2000) Fast probabilistic analysis of sequence function using scoring matrices. *Bioinformatics*, **16**, 233–244.

Zhao,Y. *et al*. (2012) Improved models for transcription factor binding site identification using nonindependent interactions. *Genetics*, **191**, 781–790.

Zuo,C. *et al*. (2015) atSNP: transcription factor binding affinity testing for regulatory SNP detection. *Bioinformatics*, **31**, 3353–3355.

Downloaded from https://academic.oup.com/bioinformatics/article/33/4/514/2726114 by guest on 21 August 2022