Published online 18 December 2017 *Nucleic Acids Research, 2018, Vol. 46, No. 5* ***e29***
doi: 10.1093/nar/gkx1251

THiCweed: fast, sensitive detection of sequence features by clustering big datasets

Ankit Agrawal[1], Snehal V. Sambare[1], Leelavati Narlikar[2] and Rahul Siddharthan[1,*]

[1]Computational Biology Group, The Institute of Mathematical Sciences (HBNI), Chennai 600113, Tamil Nadu, India and [2]Chemical Engineering and Process Development Division, CSIR-National Chemical Laboratory, Pune 411008, Maharashtra, India

Received June 09, 2017; Revised November 28, 2017; Editorial Decision November 30, 2017; Accepted December 01, 2017

ABSTRACT

We present THiCweed, a new approach to analyzing transcription factor binding data from high-throughput chromatin immunoprecipitation-sequencing (ChIP-seq) experiments. THiCweed clusters bound regions based on sequence similarity using a divisive hierarchical clustering approach based on sequence similarity within sliding windows, while exploring both strands. ThiCweed is specially geared toward data containing mixtures of motifs, which present a challenge to traditional motif-finders. Our implementation is significantly faster than standard motif-finding programs, able to process 30 000 peaks in 1–2 h, on a single CPU core of a desktop computer. On synthetic data containing mixtures of motifs it is as accurate or more accurate than all other tested programs. THiCweed performs best with large 'window' sizes ($\geq$50 bp), much longer than typical binding sites (7–15 bp). On real data it successfully recovers literature motifs, but also uncovers complex sequence characteristics in flanking DNA, variant motifs and secondary motifs even when they occur in <5% of the input, all of which appear biologically relevant. We also find recurring sequence patterns across diverse ChIP-seq datasets, possibly related to chromatin architecture and looping. THiCweed thus goes beyond traditional motif finding to give new insights into genomic transcription factor-binding complexity.

INTRODUCTION

Chromatin immunoprecipitation with sequencing (ChIP-seq) (1) is a widely used assay for determining transcription factor-binding sites (TFBS) *in vivo*. By crosslinking the *in vivo* DNA–protein complexes using formaldehyde, sonicating to break the DNA, precipitating the protein of interest using a specific antibody, reversing the crosslinks, sequencing the DNA fragments and mapping them to a reference genome, a genome-wide map of TFBS with a resolution of 100–200 bp can be obtained. Newer variants like ChIP-exo (2) and ChIP-nexus (3), which promise even higher resolution, are gaining popularity. Typically these assays yield hundreds or, in large genomes, thousands to hundreds of thousands of binding sites per factor per cell type (4,5).

TFBS are generally characterized by short conserved patterns or 'motifs' in the DNA sequence, commonly represented by 'position weight matrices' (PWMs) (6,7), a probabilistic representation where each position within a binding site is described by an independent categorical distribution over the 4 nucleotides. A key bioinformatic task is to identify these motifs, but *ab initio* motif detection using traditional tools such as MEME (8) and Gibbs samplers such as AlignACE (9,10) and PhyloGibbs (11,12) is a challenge on such large datasets. Additionally, it is common for factors to interact with DNA via co-factors and not directly, which means a mixture of different motifs may be found in the ChIP-seq data.

A previous program by one of us, MuMoD (13), was targeted at the second of these problems: it simultaneously and sensitively finds multiple motifs in a given dataset. Other programs such as Chipmunk (14–16), Meme-Chip (17) and Weeder (18,19) find successive motifs sequentially, masking previously identified sites or sequences to find the next motif.

The program we describe here, THiCweed, offers both speed and accuracy in finding multiple motifs in large datasets. It does not require prior information on the number of motifs or the lengths of the motif, since its approach is based on clustering rather than traditional motif finding, and the clustering is based on stringent statistical criteria. On synthetic data, we show that it outperforms all current alternatives greatly on speed and is close to the best current alternative in terms of accuracy. On real genomic data, it reveals an unusual complexity in the structure of sequence motifs, in particular in internal dependencies and in flanking sequence extending far beyond the core motif.

*To whom correspondence should be addressed. Tel: +91 44 2254 3204; Fax: +91 44 2254 1586; Email: rsidd@imsc.res.in
Present address: Snehal V. Sambare. Bioinformatics and Computational Biology program, George Mason University, Manassas, VA 20110, USA.

MATERIALS AND METHODS

There are two components to our approach:

(i) First is an efficient method of divisive hierarchical clustering. Starting with one large cluster, we split it in two clusters (or three, the third consisting of poor matches to either cluster). The scoring is described below, and is based on the likelihood ratio of a sequence belonging to one or the other cluster, done iteratively starting from an initial heuristic split. We then split each new cluster into two (or three) further clusters; and proceed until no further splits are possible. For each split, we apply stringent statistical criteria to accept or reject the split. Further optimizations are described in 'Algorithm'.

(ii) During this clustering process, we include shifts and reverse complements of individual sequences to find optimal clusters. This is implemented by considering fixed-sized 'windows' of length W, one window within each sequence. Sequences may have variable length; we permit up to half the window to lie outside the sequence, with the missing nucleotides scored as `N`'s, so that for each sequence of length L, $2L$ configurations (L window positions and two orientations) are considered and the optimal window chosen. The default choice of W is one-third the median sequence length, that is, much longer than a typical TF motif. whose positioning and orientation is sampled. This, it turns out, constitutes an effective and fast implementation of an *ab initio* motif finder on large ChIP-seq datasets, in addition to detecting the variations in motif and sequence context alluded to in the previous point.
THiCweed can also be used on sequences that have been previously aligned by a 'feature' (motif) to discover additional motifs/complexities, by disabling shifts and reverse complements, similar to the program No Promoter Left Behind (20,21), but we do not discuss this use here.

Our divisive clustering is in contrast to typical (agglomerative) hierarchical clustering, where individual data points are formed into clusters, requiring $O(N^3)$ or at best $O(N^2 \log N)$ time for N data points. We call our approach 'Top–down hierarchical clustering'; and since its purpose is to weed out 'signals' in ChIP-seq peaks, we call the program 'THiCweed'. (We considered 'THC-weed' but it may confuse search engines.)

Algorithm

Top–down hierarchical clustering. The algorithm and a typical run through it are portrayed in Figure 1 and described below. We first take the simpler case of input data that has been pre-aligned with all sequences of the same length, where we do not consider shifts and reverse complements of sequences. The steps are as follows:

(i) Initialize with one cluster containing all sequences.
(ii) Split every current cluster C (initially just one cluster), into two clusters C_1 and C_2, using scoring and significance criteria described below. Sequences not consistently clustering with either C_1 or C_2 (as described below) are concatenated into a third cluster C_p. In each round, all these unclustered sequences from each division are concatenated into one cluster.
(iii) After every two iterations of step (2), if the current state has more than two clusters, reassign the poor-scoring sequences (sequences whose likelihoods in their current cluster are low) to the 'best' available cluster.
(iv) Repeat from (2), until no new clusters are formed and no reassignments are made.

The user may specify a maximum number of desired clusters, and if the number of clusters at the end is greater than this, a dendrogram of current clusters is constructed and closest leaves are joined until the number of clusters is sufficiently reduced.

Scoring. Only windowed portions of sequences are scored. Let the window length be W. Consider a cluster C with N sequence windows in it, $S^1, S^2, \ldots, S^N$. The probability of seeing this data if all these windows were drawn from the same PWM model is:

$$P(C) = \prod_{i=1}^{W} \frac{\prod_\alpha \Gamma(n_{i\alpha} + c)\Gamma(4c)}{\Gamma(\sum_\alpha n_{i\alpha} + 4c)\Gamma(c)^4} \quad (1)$$

where, $n_{i\alpha}$ is the number of times nucleotide α appears in column i, and c is a pseudocount (0.5 by default). If the cluster contains a single sequence, this expression reduces to $\left(\frac{1}{4}\right)^W$.

The likelihood that a sequence window S is sampled from the same PWM as sequences in a cluster C that contains N seqs is:

$$P(S|C) = \frac{P(S.C)}{P(C)} = \prod_{i=1}^{W} \frac{n_{i S_i} + c}{N + 4c} \quad (2)$$

where, S_i indicates the i'th nucleotide in sequence window S and $n_{i S_i}$ is the number of occurrences of that nucleotide at position i in the cluster.

When splitting a cluster, an initial split is made by ranking each sequence by its likelihood of belonging to that cluster, and moving the 'best' 25% to another cluster. Then sequences are selected in random order, removed from their current cluster and re-assigned to the more likely cluster, considering all possible window choices (position and orientation) within the sequence during the reassignment, until no further reassignments are made.

The significance of the split is assessed using two criteria. First, we demand that the log ratio of the likelihoods of the two clusters, to the likelihood of the unsplit cluster, as calculated from Equation 1, exceed a threshold, calculated from the log likelihood ratio (LLR) of two columns being cleanly separated in nucleotide composition. That is, suppose the two clusters consisted of random sequences, and were split on a single position—say, one cluster contained only `A` or `C` in that position, the other only `G` or `T`—while the nucleotides at all other positions are evenly distributed. This is not a significant split (it is always possible to do this, or better, for any cluster). Call the log likelihood ratio in this case L_1. However, if the clusters differed in this manner in two positions—one cluster contained only `A` or `C` in

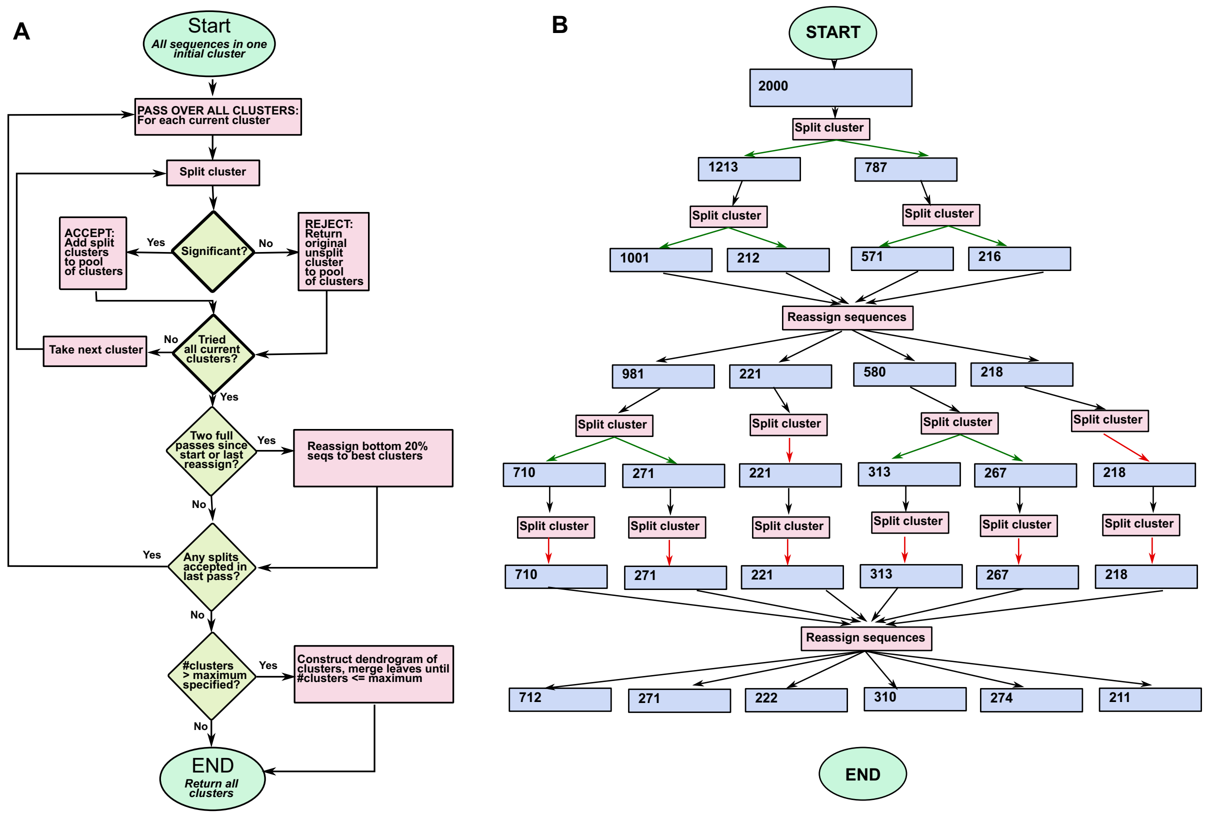
A
Start
All sequences in one initial cluster
PASS OVER ALL CLUSTERS: For each current cluster
Split cluster
Significant?
Yes
No
ACCEPT: Add split clusters to pool of clusters
REJECT: Return original unsplit cluster to pool of clusters
Tried all current clusters?
No
Take next cluster
Yes
Two full passes since start or last reassign?
Yes
Reassign bottom 20% seqs to best clusters
No
Any splits accepted in last pass?
Yes
No
#clusters > maximum specified?
Yes
Construct dendrogram of clusters, merge leaves until #clusters <= maximum
No
END
Return all clusters
B
START
2000
Split cluster
1213
787
Split cluster
Split cluster
1001
212
571
216
Reassign sequences
981
221
580
218
Split cluster
Split cluster
Split cluster
Split cluster
710
271
221
313
267
218
Split cluster
Split cluster
Split cluster
Split cluster
Split cluster
Split cluster
710
271
221
313
267
218
Reassign sequences
712
271
222
310
274
211
END

Figure 1. (**A**) Flowchart for the hierarchical clustering algorithm. The initialization is with all sequences in one cluster. At every pass, an attempt is made to split every current cluster. Splits are accepted or rejected based on significance. Every two passes, a reassignment of low-scoring sequences to the best available cluster is made. When a pass has ended with no splits being made, the program terminates returning the current clusters. (**B**) A possible run for an input of 2000 sequences. The blue boxes represent cluster sizes, green arrows from 'Split Cluster' boxes indicate successful splits and red arrows indicate unsuccessful splits. Each horizontal row of 'split cluster' boxes represents one pass.

those two positions, the other only G or T—this would be significant. Call the log likelihood ratio of this split L_2. We demand the LLR of the split performed be equal to at least $L_1 + T(L_2 - L_1)$ where, T is a parameter set to 0.4 by default (Supplementary Data). L_1 and L_2 can be calculated quickly using Equation 1.

Second, we demand that the splits be reproducible. using the following approach: we perform the split four times with four random initializations. With the resulting four pairs of clusters, we demand that at least three of the six pairwise cluster comparisons that result have an adjusted Rand index (ARI) (22) greater than a threshold r (by default 0.2). An ARI of 1.0 indicates perfect agreement while random clusterings would have ARIs close to zero. If the three pairwise comparisons between the first three splits each exceed r, the fourth split is not performed. If the split is accepted, the three pairs of clusters resulting from the three splits are identified based on majority membership and sequences that failed to be consistently clustered by this criterion (that is, did not cluster in the same way according to this association) are put in a third cluster.

Splits that fail one of these two significant criteria are rejected, that is, the split clusters are joined again and returned to the pool. Both parameters T and r are user adjustable. The default values were chosen based on benchmarks on synthetic data, as discussed in Supplementary Data.

When reassigning sequences (step 3 of the algorithm), we consider the poorest 20% of the sequences (measured by their likelihoods in their current clusters). For each sequence S, we first remove it from its current cluster, then calculate $P(C')$ for each available cluster C where, $C' = C + S$, using the above formula, and add it to the best cluster. In practice, on average 4% and at most about 10% of the sequences considered in this step get reassigned.

Benchmarking: synthetic data

We generated synthetic datasets consisting of sequences of length 100 bp each, with motifs drawn from random PWMs placed within the central 40 bp of these sequences, and otherwise random (each nucleotide having probability 0.25). The PWMs had columns sampled from Dirichlet distributions with uniform hyperparameter c (i.e. each column $\boldsymbol{v}$ denoting the probability distribution over the four bases A, C, G and T, was independently sampled from the distribution $P(\boldsymbol{v}) \propto v_\alpha^{c-1}$). Drawing from a Dirichlet distribution with a low value of c is more likely to result in a probability distribution that is highly skewed, i.e. is different from a uniform 0.25 probability per base. This skewness reduces with increase in c, a high value of c making the motif less distinguishable from background. Five datasets were generated with $c = 0.1$, 0.2, 0.3, 0.4 and 0.5. Each dataset consisted of 20 files, with each file having sequences containing between two and five distinct motifs (one motif per sequence), the motifs drawn from PWMs of a ‘core’ width of 5–10 bp and a tapering ‘flank’ to a full width of 10–20 bp (to reflect what is often in real data, as described below). The core positions were drawn from Dirichlet distributions with the hyperparameter c as described above, while the flanks tapered off rapidly from the core c to a hyperparameter of 20 (essentially a uniformly random vector). The performance of the programs and therefore the conclusions do not change when the flank is omitted (not shown).

Each sequence contained one motif, and each dataset contained motifs drawn from a small number of PWMs. The number and lengths of PWMs were varied across datasets for each c, but the distribution of numbers and lengths was the same for different c’s. Figure 2 shows synthetic motifs for $c = 0.1$, 0.3 and 0.5, all with a core width of 6 bp and a full width of 20 bp.

THiCweed and five other programs (Peak-Motifs (23), MuMoD, Chipmunk, Meme-Chip and Weeder2) were run on these sets, in multiple-motif ZOOPS mode (zero or one occurrences of a motif per sequence). The ‘known’ clustering of the set was the assignment of sequences to PWMs and the ‘predicted’ clustering for each program was the assignment of sequences to predicted motifs. The known and predicted clusters were compared using the ARI, and the results plotted as a function of c. Higher ARI indicates a better match between the clusterings, with 1.0 indicating perfect agreement and 0.0 being the value expected by chance.

Two such datasets are shown here, with dataset 1 containing 1000 sequences per file and dataset 2 containing 5000 sequences per file. The ARIs are averaged over all 20 files for each value of c in each dataset.

Commandline options:

THiCweed: no additional parameters

MuMoD: default parameters were used for the curves marked ‘MuMoD’. For ‘MuMoD(i)’ the true number of motifs was specified.

ChipMunk: in all runs, the correct number of motifs was specified. The length of the motif was given as 7:20.

Weeder: default options, but with a background frequency model derived from synthetic data.

Meme-Chip (meme): dreme was disabled with ‘`-dreme-m 0`’, and the known number of motifs specified with ‘`-meme-nmotifs`’, with default parameters otherwise.

Meme-Chip (dreme): meme was disabled with ‘`-meme-nmotifs 0`’ Peak-Motifs: default parameters were used.

Other notes

Despite the ‘filter’ keyword used in the command line, Chipmunk sometimes predicts multiple motifs per sequence because it searches for matches for predicted motifs in all sequences. For computing the ARI, each sequence was classified to the best-matching motif, as per the score reported by Chipmunk. The same was done for Peak-Motifs. In addition, sequences where no motifs were reported were assigned to an additional cluster.

ENCODE data

Here we used data from the ENCODE project (4,5,24), consisting of ChIP-seq peaks. Narrowpeak files were downloaded from the ENCODE website. Seventy five base pairs flanking sequence was taken about each peak location, and repetitive regions (lowercase sequence in chromosome files downloaded from the UCSC Genome Browser (25), identified using RepeatMasker and Tandem Repeat Finder with period of 12 or less) were rejected for the purposes of this work. The cell types and ENCODE accession numbers for various factors portrayed in Figures 4–7 are as follows, and full output is available on the web server:

Factor	Cell type	Accession number
BATF	GM12878	ENCSR000BGT ENCFF002CGQ
BCL11A	GM12878	ENCSR000BHA ENCFF002CGR
ELK1 (Figure 4:1)	HeLa-S3	ENCSR000ECI ENCFF001VIJ
ELK1 (Figure 4:2)	A549	ENCSR623KNM ENCFF818TAN
FOS	HeLa-S3	ENCSR000EZE ENCFF001VHZ
FOXA1	Ishikawa	ENCSR000BKW ENCFF002CGL
GATA1	erythroblast	ENCSR000EXP ENCFF001VQR
GATA2	K562	ENCSR000EWG ENCFF001VNE
IRF1 (Figure 5)	K562	ENCSR000EGL ENCFF002CWW
IRF1 (Figure 4)	K562	ENCSR000EGT ENCFF001VNN
IRF3	GM12878	ENCSR408JQO ENCFF735DCQ
MAX	HeLa-S3	ENCSR000EZF ENCFF001VIT
MEF2C	GM12878	ENCSR000BNG ENCFF002CHD
MYC	K562	ENCSR000EGS ENCFF002CWF
NFYA	HeLa-S3	ENCSR000DNS ENCFF002CSU
NR2F2	K562	ENCSR000BRS ENCFF002CME
RELA	GM12878	ENCSR000EAG ENCFF001VET
REST	Panc1	ENCSR000BJO ENCFF002CNA
RFX5	HepG2	ENCSR000EEA ENCFF002CUT
SIX5	GM12878	ENCSR000BJE ENCFF002CHU
SP1	K562	ENCSR000BKO ENCFF002CMN
SP2 (Figure 4:1)	H1-hESC	ENCSR000BQG ENCFF002CJL
SP2 (Figure 4:2,3,4)	HepG2	ENCSR000BOU ENCFF002CLC
STAT5A	K562	ENCSR000BRR ENCFF002CMQ
TAL1	K562	ENCSR000EHB ENCFF002CYH
TEAD4	K562	ENCSR000BRK ENCFF002CMT
ZNF143	GM12878	ENCSR000DZL ENCFF002CPW

The ZNF143 clusters were compared with nucleosome positioning data in the same cell type (GM12878) from ENCODE and PhastCons (26) phylogenetic conservation data (with other primates) from the UCSC genome site (25), distances from nearest transcriptional start sites (TSS) and

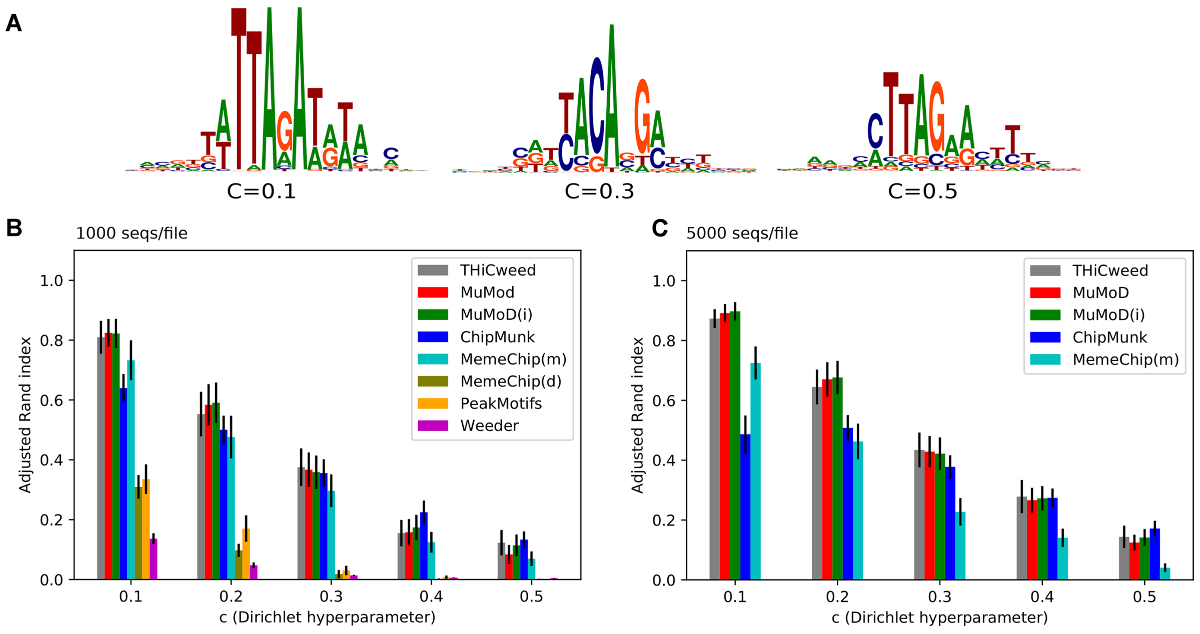

Figure 2. (**A**) Examples of embedded synthetic motifs. In this case all these have core widths of 6 bp and full widths of 20 bp, which are common to corresponding files in all datasets. The PWMs are sampled from different values of *c*, which varies from the indicated value in the core to a large value of 20 at the periphery. This is intended to model the appearance of motifs observed in real data. (**B**) and (**C**): ARI (higher is better) of predicted clustering to known clustering of synthetic datasets, containing motifs drawn from PWMs sampled column wise from Dirichlet distributions with hyperparameter *c*. Error bars in black (standard error from 20 datasets). (B) In the case of 1000 seqs/file, THiCweed is competitive but somewhat inferior on this metric to MuMoD and ChipMunk, and somewhat superior to MemeChip (meme mode). (C) With 5000 seqs/file, comparing the better-performing programs from the previous figure, THiCweed is very close to MuMoD in performance.

DNAse-seq values from ENCODE, using custom python scripts. For TSS, we used the refGene data from the hg19 release on the UCSC genome browser site.

RESULTS

Synthetic data

Results for the two datasets described in 'Materials and Methods' section are plotted in Figure 2 parts A, B and C for $c = 0.1$, 0.2, 0.3, 0.4 and 0.5 (smaller value of *c* corresponds to sharper motifs).

In all cases THiCweed was run with default parameters, and in particular, a 'window size' of 33 bp or one-third the median input sequence length. As noted, it is designed to be run with large window sizes on real genomic data. Also, the stringent criteria for splitting a cluster ensure that spurious clusters are unlikely, so setting the maximum number of clusters helps only marginally (not shown). Since clusters are split according to significance criteria, there is no option to set a minimal amount of clusters.

MuMoD was run both with default parameters ('MuMoD') and with the additional information of number of motifs ('MuMoD(i)'); the latter provides only marginal improvement. Chipmunk (in ChipHorde mode) requires the exact number of motifs to be told to it, which was done in these cases, and the range of lengths of the motif was given. Meme-Chip with its default options run the MEME motif finder on a random subset of the input data, with inferior results. Forcing MEME for the full set improved the results, at a significant cost in running time. For comparison, we also disabled MEME entirely in favor of DREME, a heuristic approach based on regular expressions rather than PWMs. Weeder2 was run with default options but a background model derived from synthetic data, as described in Methods. With 1000 seqs/set, THiCweed is competitive with MuMoD and ChipMunk on this metric.

Only the best performers were tested with 5000 seqs/set. All programs show improved performance here, because the motif strength is maintained the same but background 'noise' reduces as $N^{-\frac{1}{2}}$ with increasing number of sequences *N*. But THiCweed's improvement is sharper: it catches up with MuMoD and is largely superior to ChipMunk.

The reason for poor performance of Peak-Motifs seems to be its prediction of a very large number of motifs that are minor variations of one another. While it is hard to judge the relevance of this for real data, in the case of synthetic data these are certainly spurious and THiCweed's statistical criteria for splitting help it avoid this problem.

Running times: synthetic data

Figure 3A shows running times of all the programs tested, except Peak-Motifs, for synthetic input data consisting of 200, 400, 600, 800 and 1000 sequences, each 1000-bp long and containing two different motifs, each of length 10 sampled with Dirichlet parameter 0.2, in 60:40 proportion. Meme-Chip in MEME mode is an outlier: though its performance in accuracy is not very far behind other programs (Figure 2, its running time would seem to disqualify it from realistic datasets (and indeed it disables MEME by default for sequence sets larger than about 600 × 100 bp). It appears that, of the other programs, Chipmunk and Meme-

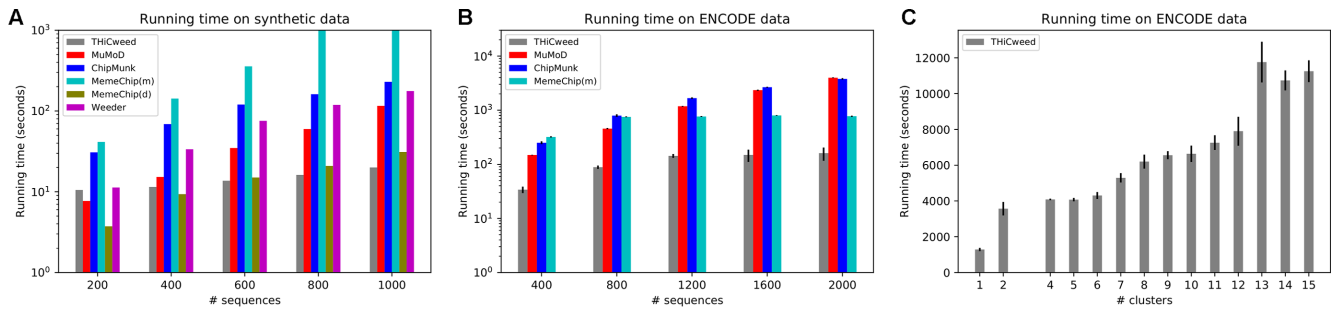

Figure 3. Running time of various programs as the size of the dataset varies for (**A**) synthetic data and (**B**) data from the ENCODE project. (**C**) THiCweed's performance on real data varies significantly with the complexity of the sequence features. Nevertheless, it remains on average much faster than other programs (Peak-Motifs was not tested but it is the fastest in this comparison).

Chip (Dreme mode) have runtimes increasing roughly linearly with data size; MuMoD and Weeder running times increase superlinearly; and THiCweed's increase is somewhat sublinear. The figure also discusses running time on genomic data from ENCODE, discussed below.

ChIP-seq data from the ENCODE project

Running on actual genomic data yields a variety of different results depending on the factor being examined and the size of the dataset.

THiCweed has no prior knowledge of the number of different motif clusters, but by default reports a maximum of 15. In some cases far fewer are reported. Because of the statistical criteria on splitting clusters that we use, described in 'Materials and Methods' section, we believe that large numbers of clusters, if produced, are statistically significant, but THiCweed can recluster the output into smaller numbers of clusters for ease of visualization and this is done in some cases here. Also, it works with window sizes much larger than typical motif lengths that one considers; here we used 50 bp. We compare the discovered motifs to previously reported motifs from JASPAR (27,28); the THiCweed website also includes comparisons to motifs from HocoMoco (29) and FactorBook (30).

Ubiquitous 'zinger' motifs. Hunt and Wasserman (31) observed that certain TF motifs occur repeatedly in different ChIP-seq datasets, which they termed 'zingers'. In particular they identified CTCF-like, JUN-like, ETS-like and THAP11-like motifs in multiple datasets. We see all of these in our analysis of ENCODE data too (for example, the THAP11-like and CTCF motifs occur in Figure 7, but several other motifs appear across multiple experiments. Figure 4 shows examples that resemble IRF1, SP2, GATA1, NFYB, REST and a novel motif that we could not identify. Of these, SP2 and the novel motif are roughly as ubiquitous as CTCF. Both frequently co-occur with CTCF and the SP2-like motif tends to be concentrated near TSS (an example is in Figure 7). We suspect a role for these in chromatin organization, a topic to be explored in future work.

Also noteworthy is the appearance of a secondary motif in multiple cases for the GATA-like and NFYB-like motifs; and the variable spacing of the REST-like motif. The canonical motif has two halves, TCAGCACC and GGACAG, separated by 2 nt. But we pick up variants, previously described in (32), with longer spacing (8 and 9 bp here). Such widely spaced motifs cause problems for conventional motif-finders, but are readily picked up in our approach.

Examples of THiCweed output. Figure 5 shows four examples of motif output. In some cases the output has been reclustered and filtered for compactness of viewing; complete results for these and many more factors are available on the THiCweed website.

We make the following observations:

(i) Zinger motifs are widespread here. The SP1-like motif that we documented above occurs in IRF1 and NFYA. The unidentified motif in the previous section appears in REST and FOXA1. CTCF occurs in NFYA and FOXA1. ETS-like occurs in IRF1.
(ii) The canonical motif for IRF1 occurs in two clusters, one of which has an additional poly-T tail.
(iii) Similarly, the canonical motif for NFYA appears in three clusters, one of which also exhibits a weak secondary motif to the left.
(iv) The canonical REST motif occurs as a closely spaced dimer (fourth cluster), partial closely spaced dimer (fifth cluster), monomer (third cluster) and a widely spaced dimer (second cluster). All of these variants also occur in THiCweed output for SP2 (Figure 6) suggesting an interaction between SP2 and REST. The widely spaced dimer is not picked up by other motif finders.

A much larger collection of THiCweed output on ENCODE factors is available on the website. Features similar to those noted above are ubiquitous.

Comparison with other programs. Figure 6 compares the output of THiCweed with three other programs. All programs pick up the main motif (though with varying numbers of instances). All also pick up the REST motif, but only THiCweed picks up the widely spaced version in one piece. THiCweed also seems to reveal a larger surrounding sequence context in many cases, notably for the SP1-like motif which generally occurs in a CG-rich background. Peak-Motifs identifies a very large number of motifs, most of which appear to be minor variations of the main motif. This may explain the poor performance of Peak-Motifs on

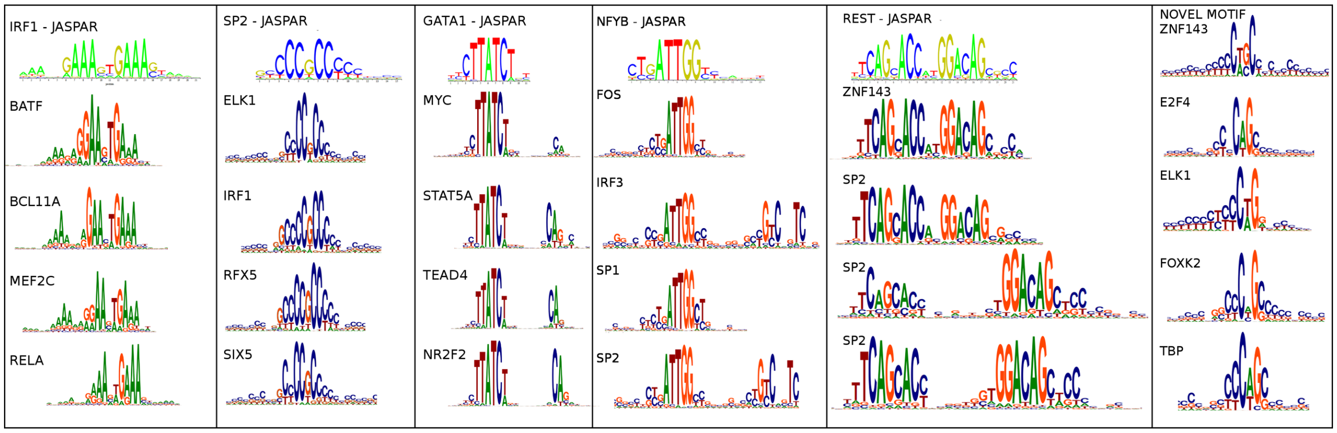

Figure 4. Motifs that occur across multiple chip-seq datasets, in addition to zinger motifs identified in (31). The factor for which the motif is the canonical motif according to JASPAR is indicated at the top of each column, together with the JASPAR sequence logo. Below are datasets for various other TFs where THiCweed finds the same motif.

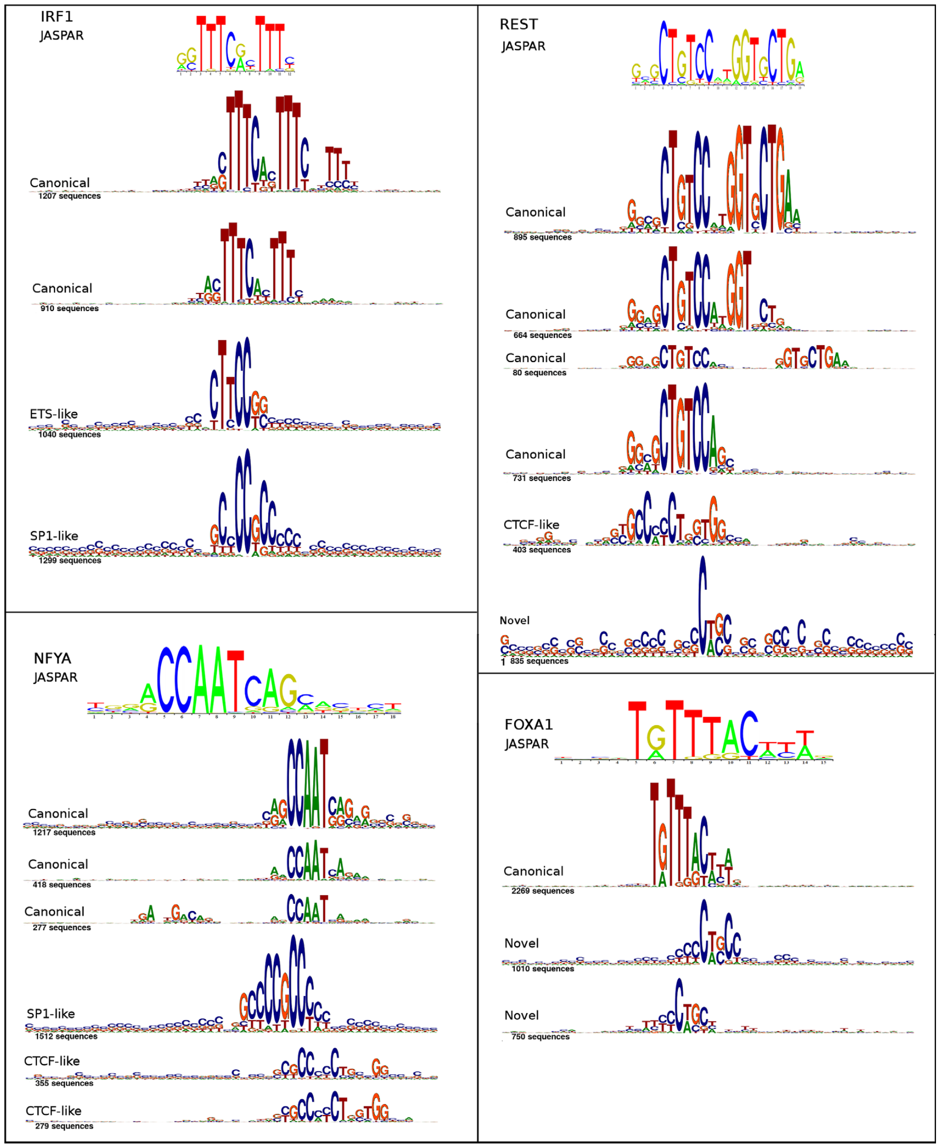

Figure 5. Sample THiCweed output on four ChIP-seq datasets: IRF1 (5543 peaks), NFYA (4497 peaks), REST (3998 peaks) and FOXA1 (4029 peaks). Not all output clusters are shown here. The full output is available on the THiCweed website.

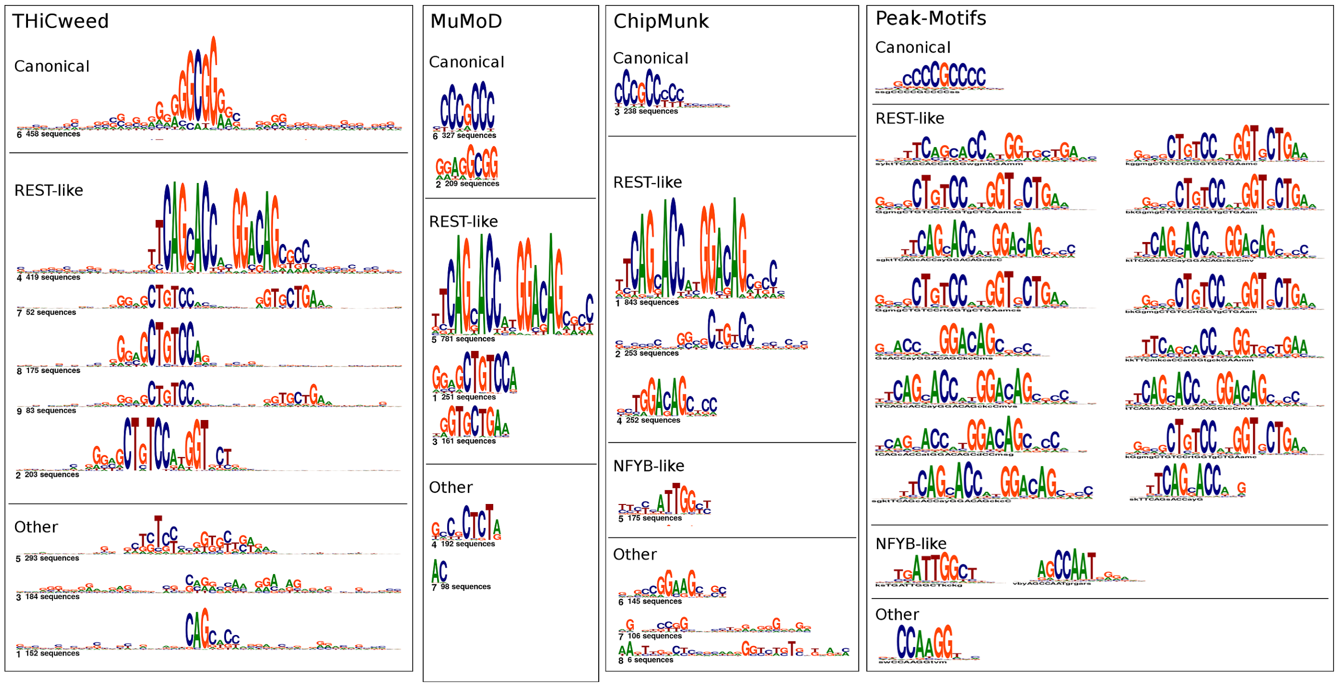

Figure 6. Comparison of clustering of 2019 peaks for SP2 by THiCweed, with motifs found by three other programs.

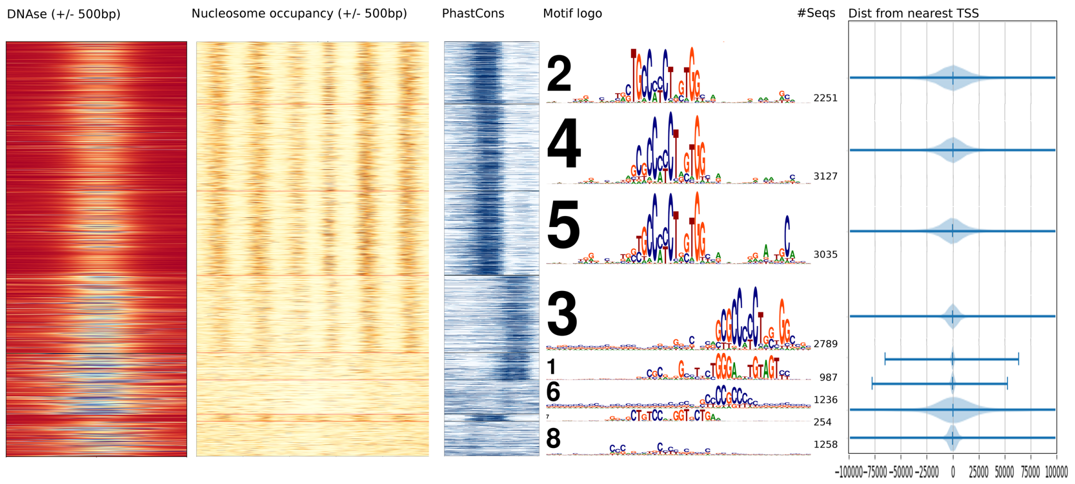

Figure 7. Comparison of sequence clusters of 14 937 ZNF143 ChIP-seq peaks with DNAse-seq values (color scale: blue = open, red = closed), nucleosome occupancy (color scale: white = 0, brown = 5+), PhastCons conservation score (color scale: white = 0, dark blue = 1) and distance from nearest TSS, suggesting connections between the motif structure in different sequence clusters, biological function, evolutionary conservation pressure, nucleosome positioning and open/closed chromatin.

our synthetic benchmark: the ARI would penalize breaking up clusters into smaller clusters.

Biological relevance of these clusters. We typically find several different motifs, variants of a motif and a few apparently uninformative clusters in THiCweed runs. Biological significance to these are suggested on comparing other genomic features, such as phylogenetic conservation (via PhastCons scores (26) from the UCSC Genome Browser (25)) and nucleosome occupancy and DNAse-seq data (from ENCODE (4)). Figure 7 compares each of eight clusters for ZNF143 with a plot of conservation, nucleosome occupancy (in an extended region of 1000 bp on each side), distance to the nearest TSS and DNAse-seq values. Cluster 6 (SP2-like motif) tends to be concentrated close to TSSs (mostly within 1000 bp—a pattern we see consistently), shows little phylogenetic conservation and no sign of nucleosome positioning. Cluster 1 (a motif resembling THAP11, identified in (31) as a zinger motif), too, is concentrated near TSSs; it too shows little effect in nucleosome positioning, but is strongly conserved. Cluster 7, resembling the REST motif, is spread away from TSSs, is phylogenetically conserved and has an effect on nucleosome positioning (which we observe in other datasets where this motif occurs).

Cluster 8 seems uninformative, but it appears concentrated near the TSSs (within about 5000 bp), which would likely not happen if it consisted only of random unclusterable sequences left over from the other clusters.

The remaining clusters are variants of the CTCF motif; cluster 5 includes the previously documented 'M2' motif. Cluster 3 appears different from other CTCF clusters in that it occurs in a GC-rich background, is more concentrated near TSS (mostly within about 10 000 bp), appears a little less conserved and a little less effective at nucleosome positioning, with more open chromatin as shown by DNAse.

Running times: ENCODE data

Figure 3B shows the results of THiCweed, MuMoD, ChipMunk and Meme-Chip (MEME mode) on real ENCODE data, consisting of 400–2000 random samples from a set of CTCF ChIP-seq peaks (dataset ENCFF001USS). The results are similar to on the synthetic data, except that, somewhat surprisingly, Meme-Chip is faster than MuMoD and ChipMunk on larger datasets.

Figure 3C shows the running time of THiCweed as a function of the number of clusters found, on 92 ChIP-seq datasets each consisting of 27 000–33 000 peaks, across multiple TFs and cell lines. The running time increases with the number of clusters, but somewhat sublinearly. On such realistic ChIP-seq datasets, THiCweed's running time is about two orders of magnitude less than MuMoD, which can take days and is also much faster than all other programs tested. Meme-Chip uses the MEME step on only a small fraction of the input sequences; and Weeder2 learns motifs from a small fraction of the sequences and uses those to analyze the rest (19). THiCweed processes the majority of files of this size in under 2 h, with interesting and biologically relevant results.

DISCUSSION

Motif finding in large datasets produced by ChIP-seq and similar experiments is a qualitatively different problem in complexity from what traditional motif finders are used to handle. Additionally, one could liken the problem of finding rarely occurring motifs to finding a needle in a haystack. We view THiCweed's approach as 'sequence feature analysis' (over large windows) rather than 'motif finding' (detection of short patterns). Our novel clustering algorithm can comfortably handle tens of thousands of sequences at a time, and with significant heterogeneity in motif content. It successfully picks up biologically relevant motifs even when they occur in fewer than 5% of the input sequences, such as the REST-like motif in ZNF143 (cluster 7 in Figure 7). Its large window size enables it to also pick up secondary motifs like the M2 CTCF motif in the ZNF143 data (Figure 7), the widely spaced dimer in SP2 and REST (Figures 5 and 6), and peripheral features such as an overall CG-richness in some motifs (eg CTCF-like cluster 3 in Figure 7). The significance criterion used for splitting, and the differences in biological parameters in Figure 7, suggest that these differences are important and are not artifacts.

Uniquely among the programs we have tested, THiCweed achieves its combination of speed and accuracy without resorting to heuristics in scoring (as DREME and Chipmunk do, using regular expressions and 'seeding' respectively) and without resorting to training on a small subset of the sequences (as Weeder does). THiCweed's clustering algorithm is stochastic, but is essentially similar to an iterated K-means clustering with $K = 2$, with significance criteria to avoid spurious splits. Instead of invoking pairwise distances and calculating a centroid, however, we calculate multinomial likelihoods correctly within the limitations of the PWM assumption. The clustering algorithm and wide-window approach ensures that little or no prior information is required to run the program: significant short motifs can be found inside longer windows by eyeballing, but other relevant sequence features can be picked up too.

A possible shortcoming is that within THiCweed's framework, only one motif occurrence per sequence will be detected (unless two motifs co-occur with a restricted spacing, as in the extended REST motif and the secondary M2 CTCF motif). Sequences that match no dominant motif may end up in a relatively uninformative cluster such as cluster 8 in Figure 7. One may ask whether, in clusters that do not match the canonical motif, the motif nevertheless occurs elsewhere in some of the peaks in additional to the non-canonical motif in the cluster. We checked for this possibility exhaustively using FIMO (33), with a q-value threshold of 10^{-3} and found that, out of 93 TFs that have a PWM in JASPAR, only 25 reported any sequences at all that were not clustered with canonical motif matched but that nevertheless showed hits for the JASPAR motif. These too showed matches in very few cases; the exceptions were SP2 and EGR1, both of which have GC rich canonical motifs, which reported matches in about 15 and 14%, respectively, of such sequences. It would therefore seem that the 'missing' of canonical motifs because of occurrence of other strong motifs within ChIP-seq peaks is not a common concern in practice.

At the moment, THiCweed runs on a single CPU core, but significant speedups are possible by parallelization.

In cases where there is a profusion of similar but slightly different motif patterns as well as an occurrence of many different motifs (as in the ZNF143/CTCF case), it appears that the differences may have biological significance, as reflected by nucleosome positioning and phylogenetic conservation. We plan to explore this, and the significance of some of the novel zinger motifs, further in a future work.

AVAILABILITY

The software is open source and available for download at http://www.imsc.res.in/~rsidd/thicweed/ under the two-clause BSD license. An online web server is also available, linked on the above page, and can be used for modest-sized jobs.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENTS

We thank Vishaka Datta and Arvind Shankar for useful discussions. The plots in Figure 7 were generated with a script

written by Arvind Shankar for a different project which will be reported elsewhere.

Author contributions: A.A., L.N. and R.S. conceived the basic algorithm. A.A. implemented an early prototype. R.S. implemented the current version in the Julia language. A.A., L.N. and R.S. contributed to various refinements in development. S.V.S. and R.S. implemented the web server. All authors contributed to the benchmarks described here and to the analysis of ENCODE chip-seq data. All authors read and approved the final manuscript.

FUNDING

Wellcome Trust-DBT India Alliance Early Career Fellowship 500188/Z/11/Z (to L.N.); Department of Atomic Energy, Government of India (PRISM 12th plan project) (to R,S., S.V.S.). Funding for open access charge: Institute of Mathematical Sciences, Chennai.

Conflict of interest statement. None declared.

REFERENCES

1. Johnson,D.S., Mortazavi,A., Myers,R.M. and Wold,B. (2007) Genome-wide mapping of in vivo protein-DNA interactions. *Science*, **316**, 1497–1502.
2. Rhee,H.S. and Pugh,B.F. (2011) Comprehensive genome-wide protein-DNA interactions detected at single-nucleotide resolution. *Cell*, **147**, 1408–1419.
3. He,Q., Johnston,J. and Zeitlinger,J. (2015) ChIP-nexus enables improved detection of in vivo transcription factor binding footprints. *Nat. Biotechnol.*, **33**, 395–401.
4. ENCODE Project Consortium (2012) An integrated encyclopedia of DNA elements in the human genome. *Nature*, **489**, 57–74.
5. Sloan,C.A., Chan,E.T., Davidson,J.M., Malladi,V.S., Strattan,J.S., Hitz,B.C., Gabdank,I., Narayanan,A.K., Ho,M., Lee,B.T. *et al.* (2016) ENCODE data at the ENCODE portal. *Nucleic Acids Res.*, **44**, D726–D732.
6. Stormo,G.D. and Hartzell,G.W. (1989) Identifying protein-binding sites from unaligned DNA fragments. *Proc. Natl. Acad. Sci. U.S.A.*, **86**, 1183–1187.
7. Hertz,G.Z., Hartzell,G.W. and Stormo,G.D. (1990) Identification of consensus patterns in unaligned DNA sequences known to be functionally related. *Comput. Appl. Biosci.*, **6**, 81–92.
8. Bailey,T.L. and Elkan,C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **2**, 28–36.
9. Roth,F.P., Hughes,J.D., Estep,P.W. and Church,G.M. (1998) Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nat. Biotechnol.*, **16**, 939–945.
10. Hughes,J.D., Estep,P.W., Tavazoie,S. and Church,G.M. (2000) Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. *J. Mol. Biol.*, **296**, 1205–1214.
11. Siddharthan,R., Siggia,E.D. and van Nimwegen,E. (2005) PhyloGibbs: A Gibbs sampling motif finder that incorporates phylogeny. *PLoS Comput. Biol.*, **1**, e67.
12. Siddharthan,R. (2008) PhyloGibbs-MP: module prediction and discriminative motif-finding by Gibbs sampling. *PLoS Comput. Biol.*, **4**, e1000156.
13. Narlikar,L. (2013) MuMoD: a Bayesian approach to detect multiple modes of protein-DNA binding from genome-wide ChIP data. *Nucleic Acids Res.*, **41**, 21–32.
14. Kulakovskiy,I.V., Boeva,V., Favorov,A.V. and Makeev,V.J. (2010) Deep and wide digging for binding motifs in ChIP-Seq data. *Bioinformatics*, **26**, 2622–2623.
15. Kulakovskiy,I., Levitsky,V., Oshchepkov,D., Bryzgalov,L., Vorontsov,I. and Makeev,V. (2013) From binding motifs in ChIP-Seq data to improved models of transcription factor binding sites. *J. Bioinform. Comput. Biol.*, **11**, 1340004.
16. Levitsky,V.G., Kulakovskiy,I.V., Ershov,N.I., Oshchepkov,D.Y., Makeev,V.J., Hodgman,T. and Merkulova,T.I. (2014) Application of experimentally verified transcription factor binding sites models for computational analysis of ChIP-Seq data. *BMC Genomics*, **15**, 80.
17. Machanick,P. and Bailey,T.L. (2011) MEME-ChIP: motif analysis of large DNA datasets. *Bioinformatics*, **27**, 1696–1697.
18. Pavesi,G., Mereghetti,P., Mauri,G. and Pesole,G. (2004) Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. *Nucleic Acids Res.*, **32**(Suppl. 2), W199–W203.
19. Zambelli,F., Pesole,G. and Pavesi,G. (2014) Using Weeder, Pscan, and PscanChIP for the discovery of enriched transcription factor binding site motifs in nucleotide sequences. *Curr. Protoc. Bioinformatics*, **47**, 2–11.
20. Narlikar,L. (2014) Multiple novel promoter-architectures revealed by decoding the hidden heterogeneity within the genome. *Nucleic Acids Res.*, **42**, 12388–12403.
21. Mitra,S. and Narlikar,L. (2016) No Promoter Left Behind (NPLB): learn de novo promoter architectures from genome-wide transcription start sites. *Bioinformatics*, **32**, 779–781.
22. Hubert,L. and Arabie,P. (1985) Comparing partitions. *J. Classification*, **2**, 193–218.
23. Thomas-Chollier,M., Herrmann,C., Defrance,M., Sand,O., Thieffry,D. and van Helden,J. (2012) RSAT peak-motifs: motif analysis in full-size ChIP-seq datasets. *Nucleic Acids Res.*, **40**, e31.
24. Landt,S.G., Marinov,G.K., Kundaje,A., Kheradpour,P., Pauli,F., Batzoglou,S., Bernstein,B.E., Bickel,P., Brown,J.B., Cayting,P. *et al.* (2012) ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. *Genome Res.*, **22**, 1813–1831.
25. Karolchik,D., Baertsch,R., Diekhans,M., Furey,T.S., Hinrichs,A., Lu,Y., Roskin,K.M., Schwartz,M., Sugnet,C.W., Thomas,D.J. *et al.* (2003) The UCSC genome browser database. *Nucleic Acids Res.*, **31**, 51–54.
26. Hubisz,M.J., Pollard,K.S. and Siepel,A. (2010) PHAST and RPHAST: phylogenetic analysis with space/time models. *Brief. Bioinformatics*, **12**, 41–51.
27. Sandelin,A., Alkema,W., Engström,P., Wasserman,W.W. and Lenhard,B. (2004) JASPAR: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Res.*, **32**(Suppl. 1), D91–D94.
28. Mathelier,A., Fornes,O., Arenillas,D.J., Chen,C.-y., Denay,G., Lee,J., Shi,W., Shyr,C., Tan,G., Worsley-Hunt,R. *et al.* (2016) JASPAR 2016: a major expansion and update of the open-access database of transcription factor binding profiles. *Nucleic Acids Res.*, **44**, D110–D115.
29. Kulakovskiy,I.V., Medvedeva,Y.A., Schaefer,U., Kasianov,A.S., Vorontsov,I.E., Bajic,V.B. and Makeev,V.J. (2012) HOCOMOCO: a comprehensive collection of human transcription factor binding sites models. *Nucleic Acids Res.*, **41**, D195–D202.
30. Wang,J., Zhuang,J., Iyer,S., Lin,X.-Y., Greven,M.C., Kim,B.-H., Moore,J., Pierce,B.G., Dong,X., Virgil,D. *et al.* (2013) Factorbook. org: a Wiki-based database for transcription factor-binding data generated by the ENCODE consortium. *Nucleic Acids Res.*, **41**, D171–D176.
31. Hunt,R.W. and Wasserman,W.W. (2014) Non-targeted transcription factors motifs are a systemic component of ChIP-seq datasets. *Genome Biol.*, **15**, 412.
32. Otto,S.J., McCorkle,S.R., Hover,J., Conaco,C., Han,J.-J., Impey,S., Yochum,G.S., Dunn,J.J., Goodman,R.H. and Mandel,G. (2007) A new binding motif for the transcriptional repressor REST uncovers large gene networks devoted to neuronal functions. *J. Neurosci.*, **27**, 6729–6739.
33. Grant,C.E., Bailey,T.L. and Noble,W.S. (2011) FIMO: scanning for occurrences of a given motif. *Bioinformatics*, **27**, 1017–1018.