Published online 7 August 2015 *Nucleic Acids Research, 2016, Vol. 44, No. 1* ***e2***
doi: 10.1093/nar/gkv807

DynaMIT: the dynamic motif integration toolkit

Erik Dassi* **and Alessandro Quattrone**

Laboratory of Translational Genomics, Centre for Integrative Biology, University of Trento, Trento, Italy

Received March 13, 2015; Revised July 02, 2015; Accepted July 28, 2015

ABSTRACT

De-novo motif search is a frequently applied bioinformatics procedure to identify and prioritize recurrent elements in sequences sets for biological investigation, such as the ones derived from high-throughput differential expression experiments. Several algorithms have been developed to perform motif search, employing widely different approaches and often giving divergent results. In order to maximize the power of these investigations and ultimately be able to draft solid biological hypotheses, there is the need for applying multiple tools on the same sequences and merge the obtained results. However, motif reporting formats and statistical evaluation methods currently make such an integration task difficult to perform and mostly restricted to specific scenarios. We thus introduce here the Dynamic Motif Integration Toolkit (DynaMIT), an extremely flexible platform allowing to identify motifs employing multiple algorithms, integrate them by means of a user-selected strategy and visualize results in several ways; furthermore, the platform is user-extendible in all its aspects. DynaMIT is freely available at http://cibioltg.bitbucket.org.

INTRODUCTION

De-novo motif search is a bioinformatics procedure aimed at the identification of biologically relevant recurrent patterns in a set of related sequences. Being originally employed for understanding transcription factor sequence-specific binding (1), it has become of widespread use for many applications, the most common being the analysis of differentially expressed genes derived from high-throughput analysis of biological samples.

A considerable number of motif search tools have been developed in the last 20 years (2–4), tackling this problem from different perspectives and computational approaches. These tools can roughly be grouped in sequence, secondary structure and discriminative tools. The very popular MEME (5), for instance, attempts at finding sequence motifs by means of an extended expectation maximization algorithm, while RNAforester (6) aims at discovering RNA secondary structure motifs by a tree alignment model. Discriminative tools such as SeAMotE (7) instead try to identify motifs by comparing different sets of sequences derived from high-throughput experiments.

While available algorithms offer a vast spectrum of possible choices, it is undeniably difficult to decide what the best match for a given problem is. Furthermore, if using more than one algorithm, it is especially complex to understand how results could be compared to yield a robust biological hypothesis. Different motif reporting formats, scoring systems and statistical evaluation approaches indeed turn the task of integrating such results into a truly daunting one.

Only a few works (8–13) have tried to tackle this issue, although to a limited extent and addressing only specific aspects. In particular, Fan et al. (8) have focused on integrating three tools, two of which performing sequence-based and one devoted to secondary structure-based motif search; on a similar line, Melina II (9) compares the results of five preselected motif search tools on the specific task of finding elements in promoters, forcing a selection among them for the final results. Ma et al. (10) introduce a toolkit performing motif comparison and clustering, but considering only sequence elements and limited to their own motif search algorithm. On the same line, RSAT (11) provides utilities for comparing motifs, but again in the context of the search algorithms they provide and limited to non-coding sequences. Stegmaier and colleagues (12) have approached the general issue of clustering motifs to identify common binding specificities. Eventually, MotifLab (13) is quite flexible and allows the user to integrate external data in addition to executing several tools: it is however not extendible and limited to the specific task of finding elements in promoters, and thus not generally applicable.

To comprehensively address this issue we need a tool which can be used with many search algorithms and general enough to address as many scenarios as possible: such a platform should be easily extendible to accommodate newly developed and custom search tools without being modified in its structure. Furthermore, it should also allow the same degree of flexibility in results integration and output methodologies. This flexibility, coupled to the possibility of customizing every aspect of its execution (from tools selection to output appearance) would considerably advance our motif integration capabilities.

We thus introduce here the Dynamic Motif Integration Toolkit (DynaMIT), a flexible platform designed to imple-

*To whom correspondence should be addressed. Tel: +390461283665; Fax:+390461283937; Email: erik.dassi@unitn.it ment this vision: it provides the means to execute multiple motif search tools, integrate their output and display the obtained results in many different ways. It is customizable and extendible in all its aspects, allowing for a truly personalized and fine-tuned usage experience. DynaMIT is implemented as an open-source Python package, and can be used stand-alone or easily embedded in complex bioinformatic pipelines; it is freely available at http://cibioltg.bitbucket.org.

MATERIALS AND METHODS

DynaMIT implementation

The toolkit was implemented in Python, exploiting the BioPython (14), scikit-learn (15), scipy (16), numpy (17), matplotlib (17) and weblogolib (18) libraries, which provided advanced functionalities for sequence handling, machine learning and data plotting tasks. Care was taken to avoid any OS-specific code, so DynaMIT can be run on Linux, Windows and Mac machines; only a few motif search tools may not be usable, depending on the availability of an OS-specific version of these. The toolkit architecture is illustrated by the class diagram in Supplementary Figure S1.

Installing and running DynaMIT

The package can be downloaded from the DynaMIT website (http://cibioltg.bitbucket.org) or installed directly through the Python Package Index (https://pypi.python.org), using the pip package (by typing "*pip install dynamit*" at the shell prompt). While both installation methods will take care of installing the required Python libraries, motif search tools must be installed separately (URLs and references are provided on the website).

The DynaMIT virtual machine

We also provide an Ubuntu virtual machine, obtainable through the DynaMIT website. It runs on the free VirtualBox software (Oracle) and includes pre-installed libraries and motif search tools: it thus allows running DynaMIT out-of-the-box, requiring the user to provide only input sequences and a configuration file.

Motif searchers

Motif searchers can be divided in two categories, namely searchers running an existing tool and searchers implementing custom logics. The former include *CMfinder*, *GibbsMotifSampler*, *GLAM2*, *GraphProt*, *HOMER*, *MDscan*, *MEME*, *MEMERIS*, *RNAforester*, *RNAhybrid*, *RNAprofile* and *Weeder* (5,6,19–27): tools were downloaded from the respective website, and Python methods written to execute each tool and output the resulting motifs in DynaMIT format. The latter include the *KnownSites*, *Matrix*, *PreviousResultsLoader* and *RegionsIntersection* searchers, which were implemented from scratch in Python, exploiting the BioPython (14) package functionalities.

Integration strategies

These components were developed from scratch, exploiting the scikit-learn (15) and BioPython (14) packages functionalities. Clustering is performed by the strategies through an affinity propagation algorithm (15). The following strategies are provided:

- *AlignmentStrategy*: computes the pairwise alignment of each motif pair and uses the resulting scores to execute clustering; eventually, a multiple alignment is performed on the motif consensuses composing each cluster.
- *BiclusteringStrategy*: computes a vector for each motif, listing positions occupied on input sequences as 1 and non-occupied positions as 0. A spectral biclustering algorithm (15) is then applied to these vectors.
- *JaccardStrategy*: computes the Jaccard similarity for each motif pair as $|intersection|/|union|$ of positions occupied by the motifs in the pair; this measure is then used to perform motif clustering.
- *CoOccurrenceStrategy*: scores motif pairs by the Jaccard similarity of the mutual presence of both motifs on each sequence: pairs in which instances of the first motif are often found on the same sequence as instances of the second motif will get an high score and vice versa; this measure is then used to perform motifs clustering.
- *MIStrategy*: computes motif vectors as described for the *BiclusteringStrategy*: the similarity between each motif pair is computed as the normalized mutual information of these vectors (considering only occupied positions); this measure is then used to perform motif clustering.
- *ProximityStrategy*: scores motif pairs by the fraction of instances pairs (made by one instance of the first and one of the second motif) on each sequence that are within n nucleotides: motif pairs in which instances of the first motif are often near to instances of the second will get an high score and vice versa; this measure is then used to perform motif clustering.
- *PCAStrategy*: computes vectors as described for the *BiclusteringStrategy*. A kernel PCA algorithm (15) is applied to these vectors to obtain a 2-component motif representation, onto which clustering is performed.

Modes of results visualization

These components were developed from scratch, exploiting the matplotlib (17) and weblogolib (18) packages functionalities. Three of them (*ClusterEvaluation*, *Table* and *SequenceView*) are compatible with the output of any integration strategy and provide basic cluster evaluation, tabular and graphical printing, respectively. The other visualization modes (*BidimensionalSpace*, *Heatmap*, *PositionalDensity* and *WebLogo*) require additional information to be run and their compatibility with the selected integration strategy must be checked prior to running the toolkit.

The scores computed by the *ClusterEvaluationPrinter* are obtained as follows: the instances score is the average score of motif instances for all motifs composing the cluster; the consensuses coherence score is defined as the average pairwise alignment score for all motif pairs in the cluster, while the silhouette (15) score ranges in [-100,100] and describes how cohesive and separated a cluster is. The size score is computed as 10 × number of motifs (composing the cluster), while the presence of multiple motif types (sequence, structure, etc.) is also taken into account as 10 × number of motif types composing the cluster. The overall cluster score is then obtained as the sum of all these scores.

Implementing a custom component

DynaMIT offers the possibility of implementing new components for each of the three phases composing its workflow. To allow such a feature, an abstract Python class is exposed for the *MotifSearcher*, *IntegrationStrategy* and *ResultsPrinter* components. The user willing to add a new component to DynaMIT must thus create a Python class implementing the appropriate abstract class: in particular, classes for all components must provide a *setConfiguration* method allowing DynaMIT to load the component configuration (e.g. tool path, score thresholds, etc.); a *MotifSearcher* class must also provide a *runSearch* method to actually execute the search; an *IntegrationStrategy* class must also provide a *doIntegration* method to execute the integration procedure; a *ResultsPrinter* class must also provide a *printResults* method to perform the actual printing.

Furthermore, search results must be provided by *MotifSearcher* classes in a predefined format, requiring some compulsory fields (one line per motif instance, including the motif consensus, its type, the sequence name and the position on the sequence) and permitting the inclusion of additional, searcher-specific ones. The *IntegrationStrategy* output must be a list of items (i.e. motifs clustering, motifs pair scores, etc.): while a few are required (e.g. the list of motifs to be considered by visualization modes), the content of this list might vary widely and it determines the compatibility of an *IntegrationStrategy* with a *ResultsPrinter* and vice versa. Each results visualization mode may indeed require different information: several modes, exploiting the most basic data or able to cope with the absence of more specific ones, will be executable regardless of the chosen integration strategy.

This rather limited amount of rules allows DynaMIT to run any complying component without knowing anything about its implementation details, just as if it was a pre-implemented one.

Additional annotation

A set of annotations, based on the content of the AURA 2 database (28) are available at http://aura.science.unitn.it: these include BED files for human and mouse experimentally derived RBP and cis-element sites on UTRs, formatted for use with the *KnownSites* and *RegionsIntersection* searchers. RBP binding motif matrices obtained from CISBP-RNA (29) and RBPDB (30) are also included, along with matrices for some cis-elements types, computed with BioPython (14) on matching sequences in AURA 2: these can be used with the *Matrix* searcher.

Performance assessment

DNA sequence data sets were retrieved from (31): these include motif sites at known positions, surrounded by upstream and downstream sequence obtained either from the genomic sequence context of these sites (real sequences) or through a Markov model algorithm. DynaMIT (with either all or only two motif searchers, namely *MEME* and *GLAM2*, and the XXX integration strategy), *MEME*, *GLAM2* and *Weeder* were run on these data sets. The RNA sequence data sets were composed by including three randomly generated sequences sets produced with MotifGen (https://galaxy.cbio.mskcc.org/) by implanting the motif of case example 1 or the two 9-nucleotides AU-rich element pattern (32). Furthermore, seven sets of UTRs containing various cis-elements (BRD-BOX, Histone stem loop, IRE, IRES, PAS, SECIS, TOP) at known positions were also included, for a total of 10 data sets. DynaMIT (with two different integration strategies, namely JaccardStrategy and ProximityStrategy with maximum distance of 3 nucleotides, and using all searchers), *CMfinder*, *MEMERIS*, *RNAforester* and *Weeder* were run on these data sets. Sensitivity, specificity and Matthews correlation coefficient were computed for each tool as the weighted average of each data set and plotted with R (33).

Case examples

Sequences for the first case example (hyperconserved portions of the 3'UTRs of 23 RBPs) were obtained from (34); the configuration was set to run the *Weeder* and *RNAforester* searchers, coupled to the *Alignment* integration strategy and to *ClusterEvaluation*, *PositionalDensity*, *SequenceView, WebLogo* and *Table* visualization modes. The complete set of input and output files for this case example is available in the Supplementary File 1.

Sequences for the second example are the 3'UTRs of the 50 top down-regulated genes following TTP overexpression, obtained from (35); the configuration was set to run the *MEME* searcher, two *RegionsIntersection* searchers (one with TTP PAR-CLIP-derived binding sites (35) and one with HuR PAR-CLIP-derived binding sites (36)), seven *Matrix* searchers for ARE binding proteins (HNRNPA1, HNRNPA3, HNRNPC, HNRNPL, MEX3D, NCL and TIA1 PWM matrices, obtained from CISBP-RNA (29)) and one for CELF3, known not to bind these elements. The *Proximity* integration strategy was applied (with 5 nucleotides as proximity distance threshold) and *ClusterEvaluation*, *Heatmap*, *SequenceView, WebLogo* and *Table* visualization modes were used. The complete set of input and output files for this case example is available in the Supplementary File 2. Sequences for the third example are the best 5000 peaks identified by ChIP-seq of the CTCF factor in HEK293 cells (GEO id: GSM749668): the configuration was set to run the *Homer, MDscan and Weeder* searchers. The *JaccardStrategy* was applied and *ClusterEvaluation*, *WebLogo* and *Table* visualization modes were used. The complete set of input and output files for this case example is available in the Supplementary File 3.

RESULTS

The *dynamic motif integration toolkit*

DynaMIT is a platform allowing to perform motif search through multiple tools at once, integration of their output and results visualization in a totally customizable way: users can choose which components should be run at each step and how they should behave. All these steps can be extended by adding new components: this makes DynaMIT virtually able to run any motif search tool, exploit any integration paradigm or visualization mode one can think of. We will now proceed by describing the various phases of its workflow and its individual components in details.

DynaMIT workflow

The toolkit is organized in three phases, illustrated by Figure 1: first of all, motif search is performed by means of the selected tools through the *MotifSearcher* components. As this is the most time-consuming step of the workflow, DynaMIT can run these components in parallel using multiple processors. The results of this step are then forwarded to the second module, performing motif integration through the chosen *IntegrationStrategy*; eventually, integration results are provided as input to the last phase, which visualizes the results in the various selected ways by means of the *ResultsPrinter* components.

Selection of components to be run at each step of this workflow is made through a simple tab-separated configuration file. A graphical application, the *ConfigurationGenerator*, is provided to assist in generating this file.

This toolkit is thus extremely dynamic, as every step of its execution can be tuned to fit the user needs and preferences. Furthermore, while we provide many pre-implemented components (16 motif searchers, 7 integration strategies and 7 results visualization modes), an additional strength of DynaMIT lies in its total extendibility: indeed, the three components expose a set of rules to be followed (described in the *Methods* section), allowing any newly implemented component respecting these rules to be used with DynaMIT right away (not requiring any change to its internal structure). Users can thus add new motif searchers, come up with a new integration principle or results visualization approach, and plug these easily into the toolkit.

The search phase and the MotifSearcher: *the platform workhorse*

The first phase of the workflow consists in performing the actual motif search by means of one or more *MotifSearcher* components: any number of such can be specified in the configuration file, to reach the maximum flexibility. It must be stressed, however, that the integration phase will proceed only if at least two motifs are identified at this step.

We provide 16 pre-implemented motif searchers, 12 of which execute external tools and 4 offering additional search capabilities. External tools were chosen so to encompass a wide range of applications, thus including DNA and RNA sequence, RNA secondary structure and binding specificity motif search tools. These searchers are listed in Table 1 along with the tool they implement and the motif type they retrieve. Annotations for the *MatrixSearcher*, *KnownSitesSearcher and RegionsIntersectionSearcher*, focusing on post-transcriptional regulation of gene expression, are provided at the AURA 2 (28) database website, as described in the *Methods* section.

The integrate phase and the IntegrationStrategy: *making sense of the multitude*

The second phase of the workflow aims at integrating the identified motifs, by defining groups of related motifs according to a particular criterion: this is the central step of DynaMIT execution, as the comparison between motifs obtained by the different tools takes place there. The component mediating this phase is called *IntegrationStrategy*, and only a single one of these will be executed per DynaMIT run; the *PreviousResultsLoaderSearcher* provides the means of re-using the motif search results when the selected integration strategy changes, thus avoiding the most time-consuming step.

We provide seven pre-implemented strategies, listed in Table 2 and based on the general idea of clustering the motifs identified in the previous phase. In particular, aside from the *BiclusteringStrategy* (performing a sequence and motif biclustering directly), all other components compute a score for every motif pair which is then used to drive motif clustering and produce the results. One strategy, the *AlignmentStrategy*, uses the motif consensus to compute its score (*consensus-based* strategy), while the others exploit the instances of each motif on the input sequences to do so (*instance-based* strategies).

To help reducing potential false positive calls, a motif cluster "*polishing*" step is available in DynaMIT pipeline, right after the integration phase. This procedure, which is optional, removes motif instances which do not overlap by at least *n* nucleotides (with *n* being user-specified) with at least another instance of the same cluster. Doing so reduces the number of isolated motif instances, which are more likely to be false positive calls, and consequently highlights motifs which share an high fraction of occupied sequence positions with other motifs (and are thus likely to be stronger).

The print phase and the ResultsPrinter: *displaying is understanding*

The last phase of DynaMIT workflow consists in visualizing the results produced by both motif search and integration steps: this task can be approached in many different ways, from a simple textual output of motif clusters to complex graphical motifs representations. The component performing this step is called *ResultsPrinter* and selected ones are indicated in the configuration file. If none is specified, two default modes (*TablePrinter* and *ClusterEvaluationPrinter*) will be run anyway to provide a basic summary of the results.

We provide seven pre-implemented visualization modes, developed from scratch and generating widely different outputs. All of them, except for the *TablePrinter*, result in publication-quality pictures at high resolution in the PDF format (an example of which is shown in Figure 2). In particular these are:

- *TablePrinter:* represents the identified motifs/clusters along with their instances in a tabular format. As it provides basic and necessary information, it is one of the two default visualization modes.

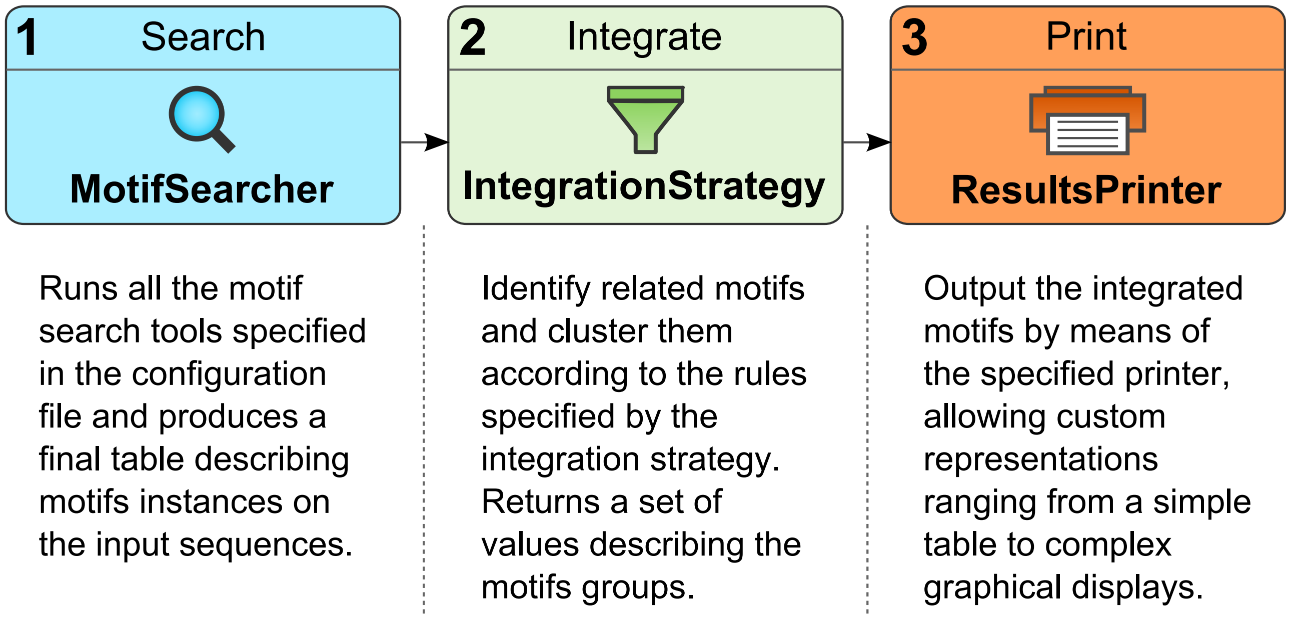

1 Search
MotifSearcher
Runs all the motif search tools specified in the configuration file and produces a final table describing motifs instances on the input sequences.
2 Integrate
IntegrationStrategy
Identify related motifs and cluster them according to the rules specified by the integration strategy. Returns a set of values describing the motifs groups.
3 Print
ResultsPrinter
Output the integrated motifs by means of the specified printer, allowing custom representations ranging from a simple table to complex graphical displays.

Figure 1. DynaMIT workflow. The figure describes the three steps composing the toolkit workflow. First of all, motif search is performed by means of all tools specified in the configuration, and all results collected together; then, obtained motifs are thus integrated according to the user-selected integration strategy. Eventually, the various specified results visualization modes are executed to provide tabular and graphical displays of the obtained data.

Table 1. Pre-implemented motif searcher components

Searcher name	Type	Implementation	Motif type
CMfinderSearcher	tool	CMfinder (19)	RNA sequence motifs
GibbsSearcher	tool	Gibbs Motif Sampler (20)	DNA/RNA sequence motifs
GLAM2Searcher	tool	GLAM2 (5)	gapped DNA/RNA sequence motifs
GraphProtSearcher	tool	GraphProt (21)	RBP binding preferences
HOMERSearcher	tool	HOMER (27)	Transcription factor binding preferences
KnownSitesSearcher	custom	loads a user-specified set of known sites in input sequences	regions of the input sequences
MatrixSearcher	custom	identifies matches for a user-specified PFM/PWM/PSSM matrix in input sequences	DNA/RNA sequence motifs
MDscanSearcher	tool	MDscan (26)	Transcription factor binding preferences
MEMESearcher	tool	MEME (5)	DNA/RNA sequence motifs
MEMERISSearcher	tool	MEMERIS (22)	RNA sequence motifs
PreviousResultsLoaderSearcher	custom	loads motif search results from a previous DynaMIT run	any type
RegionsIntersectionSearcher	custom	positionally intersects a set of user-specified regions with the input sequences	regions of the input sequences
RNAforesterSearcher	tool	RNAforester (6)	RNA secondary structure motifs
RNAhybridSearcher	tool	RNAhybrid (23)	regions of hybridization between miRNAs and input sequences
RNAprofileSearcher	tool	RNAprofile (24)	RNA secondary structure motifs
WeederSearcher	tool	Weeder (25)	DNA/RNA sequence motifs

The table lists the 16 pre-implemented *MotifSearcher* components, along with their type (either running an external tool or executing a custom logic), implementation details and the kind of motif identified by these searchers.

- *SequenceViewPrinter* (Figure 2A): displays motifs/clusters instances at their position on input sequences.
- *PositionalDensityPrinter* (Figure 2B): plots the positional distribution of motifs/clusters on the sequences, so to highlight their potential positional binding preferences.
- *BidimensionalSpacePrinter* (Figure 2C): plots motifs on a two-dimensional space, using reduced representation coordinates such as the ones derived by PCA or similar approaches.
- *ClusterEvaluationPrinter* (Figure 2D): is the other default mode and plots several scores to help in cluster prioritization: computed scores consider the cluster size, the number of different motif types it includes, the average score of its instances, the coherence of its composing motif consensuses and its silhouette value.

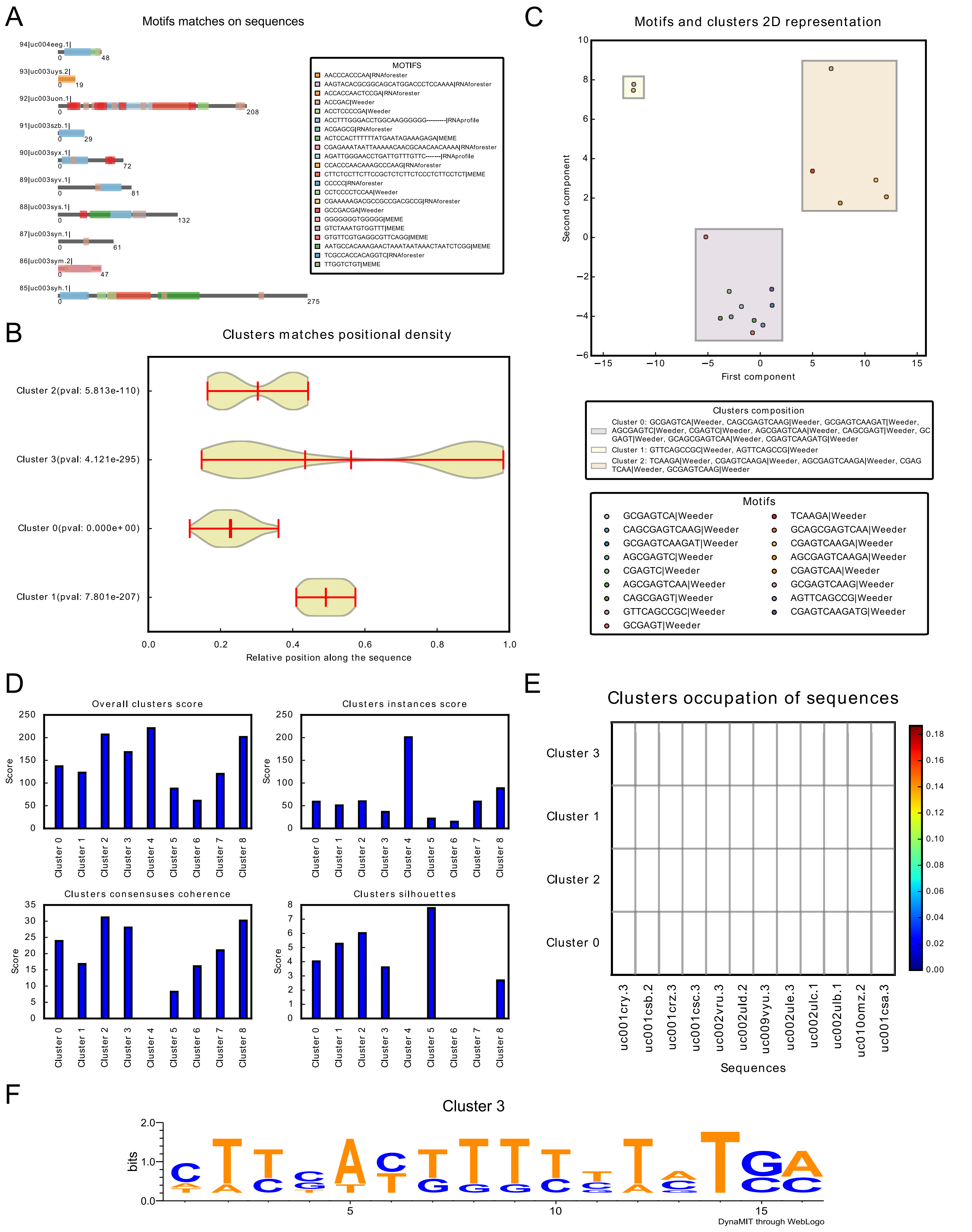


Figure 2. An example output of DynaMIT result visualization modes. The figure displays an example output from DynaMIT pre-implemented *ResultsPrinter* components. In the interest of space, an example of cluster composition legend is displayed only in part C. **(A)***SequenceViewPrinter* output,

Table 2. Pre-implemented integration strategy components

Strategy name	Type	Implementation
AlignmentStrategy	consensus-based	performs pairwise motif alignment, yielding an alignment score
BiclusteringStrategy	biclustering	applies a spectral biclustering algorithm on motifs and sequences
CoOccurrenceStrategy	instance-based	computes a co-occurrence score for each motif pair (i.e. concomitant presence of both motifs instances on the same sequences)
JaccardStrategy	instance-based	computes a Jaccard similarity score on the set of sequence positions occupied by instances of the motifs in each pair
MIStrategy	instance-based	computes a mutual information score on occupied sequence positions for each motif pair
PCAStrategy	instance-based	executes a Principal Component Analysis (PCA) on the motifs, yielding a two-component reduced representation
ProximityStrategy	instance-based	computes a score based on the fraction of instances of each motif pair being proximal (i.e. within a certain nucleotide distance)

The table lists the seven pre-implemented *IntegrationStrategy* components, along with their type (either consensus-based, instance-based or biclustering) and related implementation details.

- *HeatmapPrinter* (Figure 2E): plots an heatmap of either presence/absence or fraction of occupied sequence of each motif/cluster on each sequence.
- *WebLogoPrinter* (Figure 2F): draws weblogos based on either motif instances or cluster information to help in consensus identification.

Applying DynaMIT and selecting the right components

Given the flexibility granted by DynaMIT structure and wealth of pre-implemented components, the naturally arising question is on the applications for which it might be useful. First of all, plain sequence motif searches can obviously be accommodated, both for DNA and RNA sequence sets. In that case, employing multiple tools and integrating their results serve the purpose of obtaining robust motifs, concordantly identified by different approaches. When analyzing RNA sequences, however, the most compelling application of DynaMIT consists in integrating both sequence and secondary structure motifs, allowing to precisely pinpoint recurring patterns potentially bound by RBPs or non-coding RNAs: indeed, apparently weak sequence motifs may result as strong candidates when considering also the secondary structure. Another interesting application is in the study of potential trans-factor cooperative/competitive patterns: through two dedicated strategies, one may investigate whether, for instance, two RBPs or transcription factors have often overlapping or proximal sites on input sequences, thus outlining potentially complex regulatory phenomena. A last application may consist in the identification of binding preferences for RBPs and transcription factors (in sequences set such as the ones derived from CLIP or ChIP-seq): integrating multiple predictions may indeed ease this task, which is often made difficult by the inherent noise present in large sequence data sets.

Obviously, given the extreme flexibility of the platform, one may come up with many additional applications for which DynaMIT could be suitable: we just described here the ones which we presently see as most interesting.

Furthermore, one may also ask what is the best combination of searchers, integration strategies and visualization modes for the specific task at hand. While there is no unique answer, and many combinations may produce good results in specific cases, we can try to provide some guidance toward this choice.

Concerning motif searchers, one can group them according to their most appropriate application. Searches in DNA sequences can be performed by the *GibbsMotifSampler*, *GLAM2*, *MEME* and *Weeder* searchers. While *GLAM2* is devoted to finding gapped motifs, the others are generally applicable and can handle small to medium-sized data sets. Concerning sets of RNA sequences (e.g. UTRs), DynaMIT offers both sequence and secondary structure-based motif search tools: these are the *Weeder*, *MEME*, *MEMERIS*. *CMfinder*, *RNAforester*, *RNAhybrid* and *RNAprofile* searchers. When dealing with RNA sequences, to perform a comprehensive motif search evaluating both sequence and structure-related aspects, we suggest to always include at least one sequence and one secondary structure motif searcher (e.g. *Weeder* + *RNAforester*).

Eventually, trans-factor binding specificities can be studied with DynaMIT as well: a tool dedicated to identifying such preferences for RBPs, *GraphProt*, is offered, along with

←

displaying motif instances on input sequences. Each motif is indicated by a different color, and a partial transparence allows identifying overlapping motif instances locations. **(B)***PositionalDensityPrinter* output, plotting the density of each cluster instances on input sequences positions, expressed in relative terms from start (0) to end of the sequences (1.0). The *P*-value indicates whether the deviation of each cluster positional density from a uniform distribution is significant. **(C)***BidimensionalSpacePrinter* output, plotting motifs on a 2D-space and highlighting clusters through superimposed boxes. **(D)***ClusterEvaluationPrinter* output, displaying several scores helping in cluster assessment and prioritization. **(E)***HeatmapPrinter* output, displaying sequence occupation patterns for each motif cluster on each sequence. **(F)***WebLogoPrinter* output, displaying a logo for the motifs in a cluster.

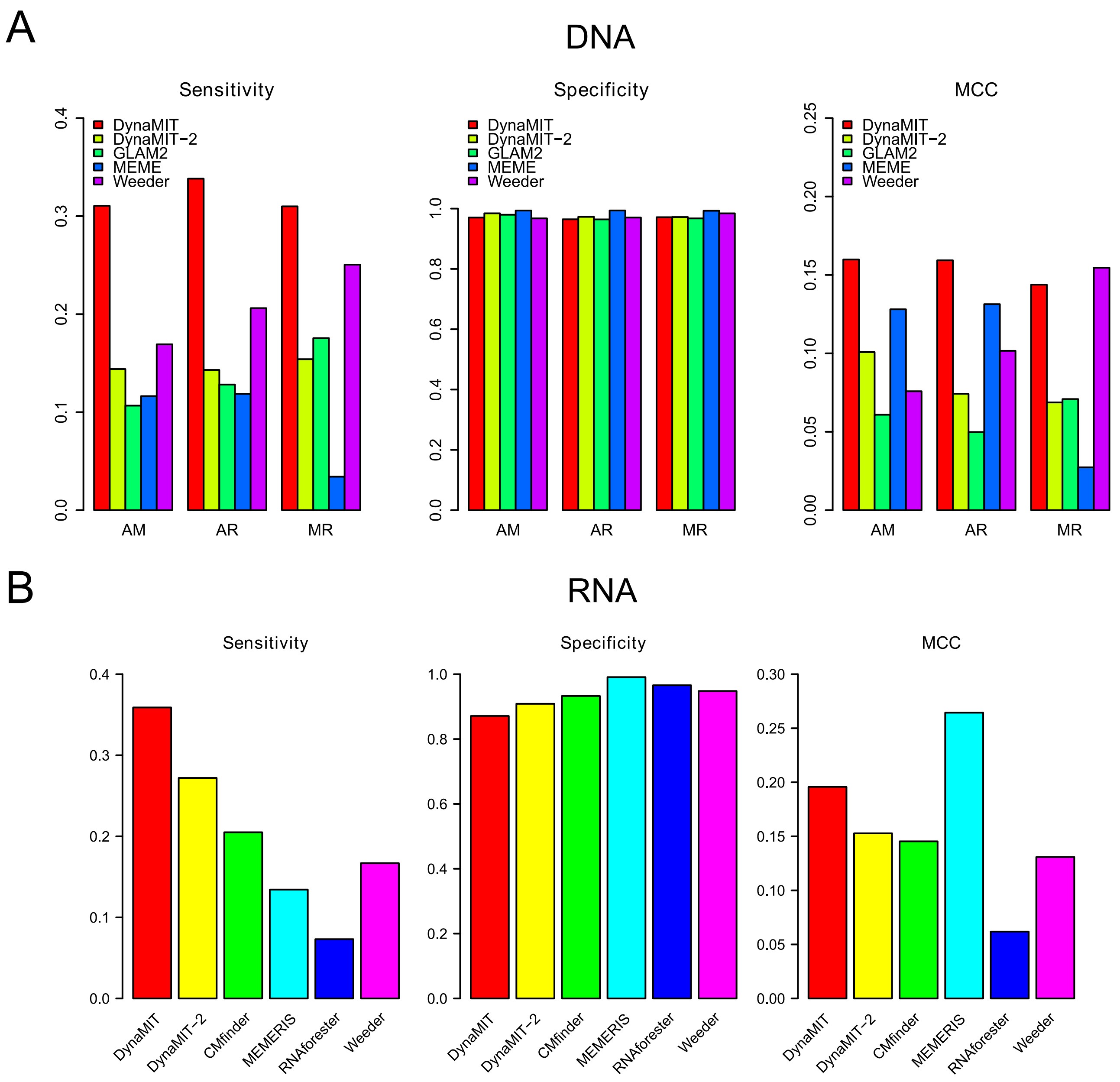


Figure 3. DynaMIT improves over the sensitivity of single tools. The figure displays the results of DynaMIT performance assessment on DNA and RNA data sets. **(A)** displays average sensitivity (left), specificity (center) and Matthews correlation coefficient on each of the three DNA data sets (AM = markov model generated sequences, AR = real sequences and MR = real sequences with lower signal to noise profiles). *DynaMIT-2* employs only two of the motif search tool used by *DynaMIT* (*MEME* and *GLAM2*) and the same integration strategy. **(B)** displays average sensitivity (left), specificity (center) and Matthews correlation coefficient on the RNA data set for each tool. *DynaMIT-2* employs the same motif search tool than *DynaMIT*, but a different integration strategy.

HOMER and *MDscan*, which are instead tailored to motifs identification in DNA ChIP-seq data sets.

On a general note, we suggest to include external data in the analysis whenever possible, employing the functionalities provided by the *Matrix*, *KnownSites and RegionsIntersection* searchers. This may represent a considerable added value; using knowledge about already determined binding sites or employing a trans-factor binding motif matrix to look for matches in the sequences of interest may allow to uncover patterns which could otherwise go unnoticed.

Integration strategies are the core part of the DynaMIT workflow: three of them, the *Jaccard*, *MutualInformation* and *PCA* are devoted to the general task of robust motif identification. These use only the amount of motif overlap on sequences, and thus make no specific assumptions on the motifs biology. The *Biclustering* strategy provides biclusters of related motifs and sequences, thus making it the strategy of choice to uncover motifs groups regulating exclusive subsets of the input sequences. Two strategies are then dedicated to identifying cooperative/competitive patterns between trans-factors: these are the *CoOccurrence* and *Prox-*

imity strategies. They employ the concomitant presence of motifs on the same sequence (*CoOccurrence*) or the vicinity of their sites (*Proximity*) and should be used when such a regulatory phenomenon is suspected or sought after. Eventually, the *Alignment* strategy integrates the consensus of the identified motifs: this can be used to group motifs with slightly divergent sequence preference: one should keep in mind, however, that instances and their positions are not considered by this strategy.

Visualization modes eventually can illustrate various aspects of the results: three of them, the *ClusterEvaluation*, *BidimensionalSpace* and *Table*, should be used to prioritize clusters and study their composition. The other modes provide details on specific aspects: the *PositionalDensity* and *SequenceView* components should be employed to study the position of motifs on the sequences, while the *Heatmap* mode report on motif presence/occupation of sequences. Eventually, the *WebLogo* mode can be used to determine the consensus for the various identified motif clusters.

A graphical recapitulation of these considerations, helping in the choice of the most appropriate components, is provided by a decision chart in Supplementary Figure S2. Furthermore, the *ConfigurationGenerator* application provides default configurations for generic DNA, RNA, ChIP and CLIP motif search cases; eventually, a set of more than 20 sample configuration files is also available for download on DynaMIT website.

DynaMIT performance assessment

In order to understand DynaMIT performance profile, in particular with respect to employing a single tool, we set out to benchmark its performances on two "golden standard" data sets, one for DNA and the other for RNA sequences. The first one is an established benchmark derived from the literature (31), and is composed of DNA sequences (details can be found in the *Methods* section) containing transcription factor binding sites at known positions. The second data set was generated by us and is composed by seven sequence groups made of UTRs containing seven different cis-elements (e.g. iron-responsive elements) and by three groups made of randomly generated sequences containing other three elements at known positions. We thus run both DynaMIT (configured to employ a subset of available tools) and the individual tools on these two benchmark data sets, retaining only the best motif cluster (for DynaMIT) or the best motif (for individual tools) from the identified ones; positions of these motif instances were then compared with the known true motif positions, and sensitivity, specificity and Matthews correlation coefficient (MCC) were computed as weighted averages of all data sets (separately for DNA and for RNA data sets).

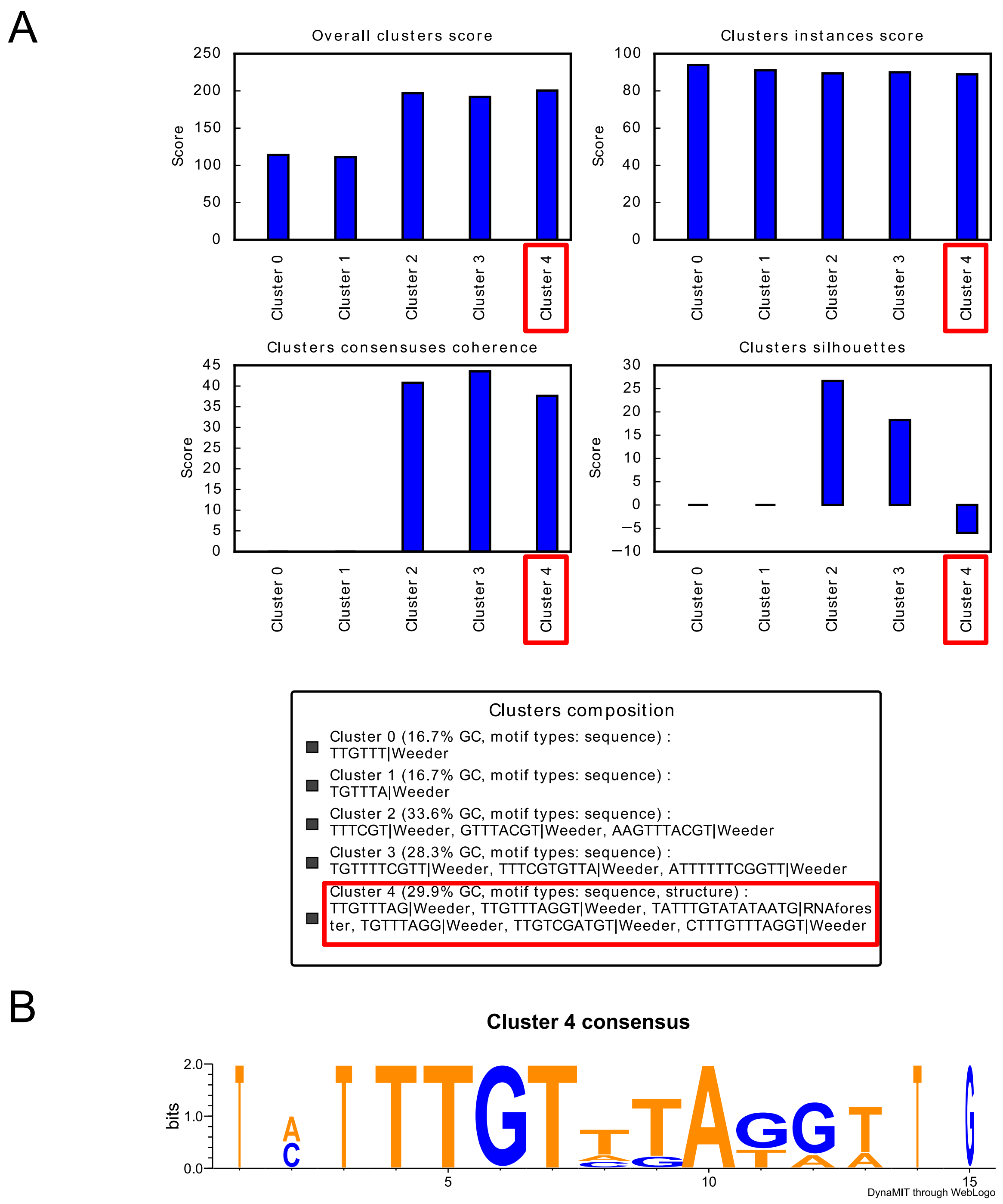
Figure 3 displays the results of this analysis for both DNA sequence data sets (**3A**) and for RNA ones (**3B**). As can be readily observed, the most striking improvement obtained when using DynaMIT lies in a considerably increased sensitivity; this increase is bigger when integrating a bigger number of tools (evident when comparing DynaMIT with "DynaMIT-2" results, where the latter was run with one tool less than the former). Furthermore, this increased sensitivity is accompanied by only a modest reduction in specificity, far smaller than the aforementioned increase. The MCC, which can be considered a comprehensive indicator of tool performance, eventually puts DynaMIT as the best or the second-best performer in both DNA and RNA data sets.

On a global perspective this suggests that DynaMIT allows for an increased motif detection power while keeping false positive calls under control, thus realizing an overall better performance than individual tools. Obviously, we used specific configurations of DynaMIT to run these benchmarks: however, many more are possible and could lead both to better or worse results with respect to the ones presented. Therefore, we suggest trying different configurations when employing DynaMIT, as a way of both tailoring the toolkit to the task at hand and optimizing its performances.

An example application: identifying the RNA motif bound by HuR

To showcase the capabilities of this toolkit, we will now describe three practical examples of its usage, focused on post-transcriptional and transcriptional regulation of gene expression.

The first deals with the identification of a motif with coupled sequence/structure specificity, identified in a set of hyperconserved regions in the 3'UTRs of 23 human RNA-binding proteins, as described in our previous work (34). In this work, we manually combined the motifs obtained from one sequence and one secondary structure motif search tools, after having observed a marked similarity of their sequence consensus: this combined motif turned out to be a binding site for the RNA-binding protein HuR (ELAVL1). This appears to be a suitable case for applying DynaMIT, and understanding whether this observation is retrievable by executing it. We thus ran DynaMIT with the same two tools (the configuration is described in the *Methods* section, and all input and output files are available in the Supplementary File 1), the *AlignmentStrategy* and multiple visualization modes. Results from the *ClusterEvaluationPrinter* and the *WebLogoPrinter* are displayed in Figure 4: as highlighted there, DynaMIT identifies a sequence-structure motif cluster (Cluster 4) which indeed contains the motifs we had manually detected in the original publication. Furthermore (Figure 4A), this cluster scores at the top in terms of overall score and has pretty high instances and consensus coherence scores (only 5.4% and 13.5% lower than the top scorer). The silhouette score is the lowest (although Cluster 0 and 1 have one motif and thus their silhouette value is 0 by default) suggesting a higher heterogeneity of the cluster, probably due to the presence of motifs of widely different length. Eventually, the consensus WebLogo for that cluster (Figure 4B) shows its resemblance to the AU-rich motifs typically bound by HuR, which has been confirmed as a functionally relevant trans-factor in the previous study. Given the coupled sequence/structure specificity of this motif (as originally determined by pull-down assays), it seems evident that employing only one of these search tools would have resulted in a much lower detection power: Weeder, in particular, outputs several motifs which are scored comparably to the "true" one, thus confusing results interpretation



Clusters composition

■ Cluster 0 (16.7% GC, motif types: sequence) : TTGTTT|Weeder

■ Cluster 1 (16.7% GC, motif types: sequence) : TGTTTA|Weeder

■ Cluster 2 (33.6% GC, motif types: sequence) : TTTCGT|Weeder, GTTTACGT|Weeder, AAGTTTACGT|Weeder

■ Cluster 3 (28.3% GC, motif types: sequence) : TGTTTTCGTT|Weeder, TTTCGTGTTA|Weeder, ATTTTTTCGGTT|Weeder

■ Cluster 4 (29.9% GC, motif types: sequence, structure) : TTGTTTAG|Weeder, TTGTTTAGGT|Weeder, TATTTGTATATAATG|RNAforester, TGTTTAGG|Weeder, TTGTCGATGT|Weeder, CTTTGTTTAGGT|Weeder

Figure 4. DynaMIT identifies the expected sequence-structure motif cluster. The figure displays the results of running DynaMIT on the first case example. **(A)** displays *ClusterEvaluationPrinter* results, with the desired cluster (cluster 4) highlighted in red in the various plots. This cluster globally scores as the best one (highest overall score). **(B)** displays the WebLogo produced by the *WebLogoPrinter* for the consensus of cluster 4, highlighting this is an AU-rich motif as typically bound by HuR (as indeed experimentally confirmed in the paper from which data for this case example were obtained).

and potentially hiding this motif. Globally, we can conclude that DynaMIT provides an increased detection power for motifs with coupled sequence/structure specificity. Furthermore, it is able to reproduce a biologically relevant observation, scoring it accordingly.

An example application: ARE-BPs competitive/cooperative patterns

The second example we present deals with a class of RBPs binding to a widespread 3'UTR cis-element, the AU-rich element (ARE). A recent work (35) analyzed the antagonistic binding pattern of two such ARE-binding proteins (ARE-BPs), namely TTP (ZFP36) and HuR. We thus se-

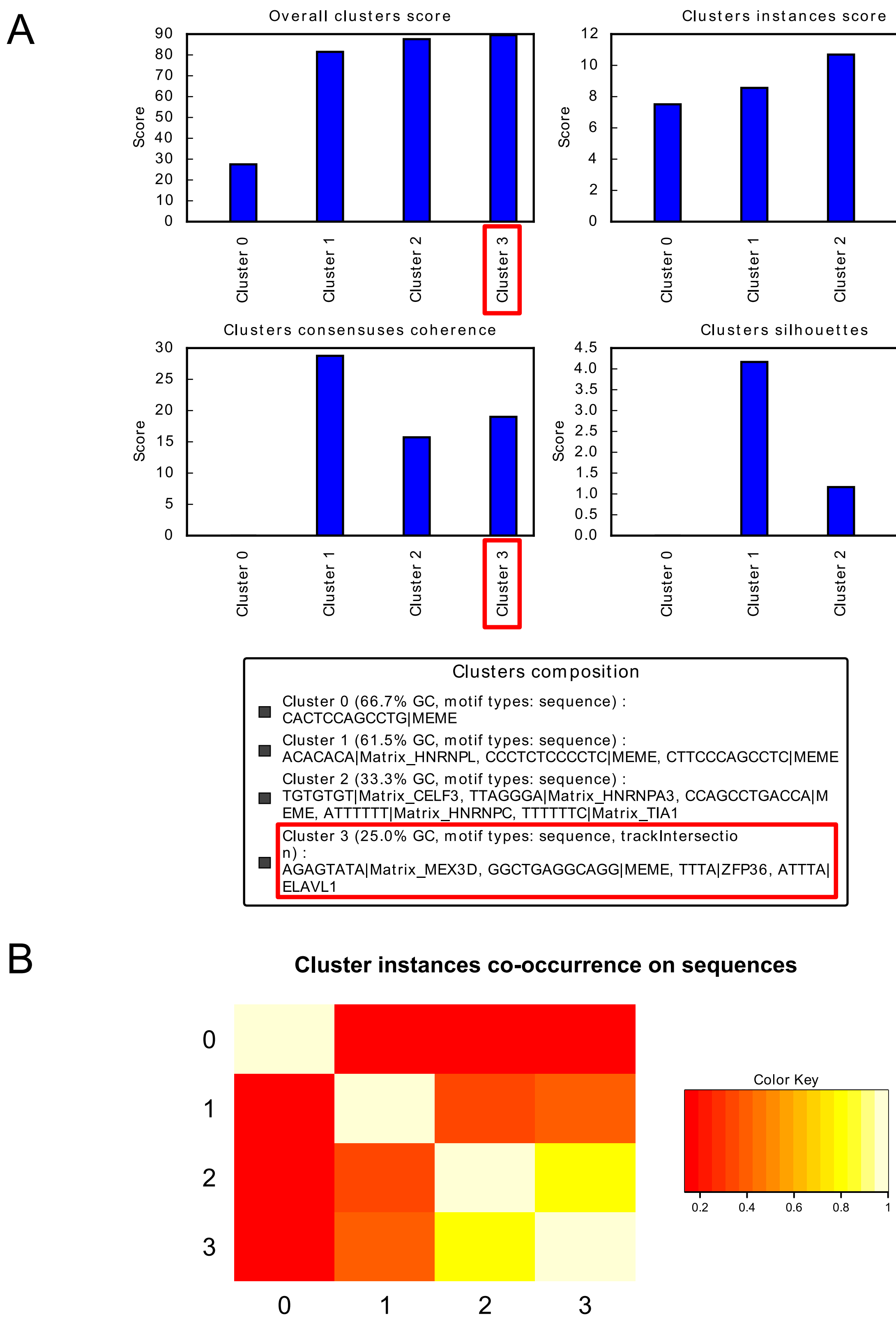


Figure 5. Multiple ARE-BPs clusters are identified by DynaMIT as potential co-regulators of TTP targets. The figure displays the results of running DynaMIT on the second case example. **(A)** displays *ClusterEvaluationPrinter* results, with the expected TTP-HuR cluster (cluster 4) highlighted in red and having the highest overall score. An interesting second cluster of potential ARE-BPs sites (Cluster 2) also emerges from the plot as the second best scorer. **(B)** heatmap displaying the fraction of sequences potentially co-regulated by each pair of clusters (thus containing co-occurring instances of both clusters). In particular, Cluster 2 and 3 reach 77.4%, with the other pairs having a much lower co-occurrence value.

lected the 50 genes which responded the most to TTP overexpression and extracted their 3'UTR exons sequences to run DynaMIT. Our aim was to understand whether this observation could be reproduced and, more in general, if additional ARE-BPs may participate to the regulation of these mRNAs. We thus configured DynaMIT to search for TTP and HuR binding sites (through the *RegionsIntersection* searcher and their respective PAR-CLIP data sets), for potential sites of other seven ARE-BPs (HNRNPA1, HNRNPA3, HNRNPC, HNRNPL, MEX3D, NCL and TIA1, through their PWM matrices and the *Matrix* searcher), for sites of one RBP not known to bind such elements (CELF3) and for generic sequence motifs with *MEME*. Given the potential overlap between binding sites of these ARE-BPs, we reasoned that the *Proximity* strategy, clustering together motifs whose instances are often within a certain distance from one another (at most 5 nucleotides in this case), would be the best choice. Eventually, several visualization modes were selected (the configuration is described in the *Methods* section, and all input and output files are available in the Supplementary File 2). Results displayed in Figure 5A suggest that, aside from the top-scoring cluster containing TTP-HuR binding sites (Cluster 3, containing also only three potential MEX3D sites), other ARE-BPs are consistently observed to potentially regulate these mRNAs. These form the second highest scoring cluster (Cluster 2) of proximal sites, composed by CELF3, HNRNPA3, HNRNPC and TIA1; furthermore, the overlap of sequences potentially co-regulated by multiple clusters (Figure 5B) is particularly high for Cluster 2 and Cluster 3, reaching 77.4%. This overlap is instead much lower for the other two clusters (Cluster 0 and 1), suggesting these may represent different regulatory specificities. Globally, this suggests that competition/cooperation with HNRNPA3, HNRNPC and TIA1 may also participate to the regulation of TTP and HuR targets, thus advocating for the adoption of a broader analysis perspective. This is further testified by CELF3, whose potential sites appear to be proximal to other ARE-BPs sites, although this RBP is not known to bind such elements: given such proximity, also this RBP may thus influence the outcome of the regulatory pattern. This prediction has been possible only by integrating multiple search tools (empowered by external data), which provides an increased ability to comprehensively consider the potential regulatory mechanisms at play.

An example application: identifying CTCF binding specificities.

The last example we present attempts at performing motifs identification in a high-throughput setting, employing thousands of sequences derived from a ChIP-seq experiment (37) of the CTCF transcriptional regulator, performed in the human HEK293T cell line.

CTCF has multiple zinc-finger domains that can be employed in different combinations to bind different DNA target sequences (38); it can work both as a transcriptional activator and as a repressor, in a context-dependent manner (38). We thus reasoned that identifying the motifs bound by this protein could be harder than for other transcription factors and that applying DynaMIT could make addressing this task easier. We configured DynaMIT to run the *Homer*, *MDscan, MEME* and *Weeder* searchers, use the *Jaccard* integration strategy (employing the overlap of different motifs instances to drive motif clustering) and output the results with the *Table*, *ClusterEvaluation* and *WebLogo* printers (configuration, input and output files are available in the Supplementary File 3).

Results displayed in Figure 6 highlight that the three most frequent binding motif (37) are the three top-scoring clusters identified by DynaMIT (*Cluster 2*, *1* and *0* as shown by Figure 6A). In particular, one of these motifs (*Cluster 1*) was identified by three of the four employed tools, thus supporting its strength and reliability. On the other end, two motifs (*Cluster 2* and *0*) were retrieved only by a single tool (respectively *MDscan* and *Weeder*) and would have most likely been missed if working in a non-integrative fashion. Furthermore, we can observe the cluster instance consensuses to be slightly less defined than the known ones (as shown by Figure 6B–D), probably because of noise due to including as much as 5000 sequences in the motif search phase (which was indeed chosen to test DynaMIT behavior with noisy input data). Globally, however, the results suggest that DynaMIT is able to overcome this type of noise and produce meaningful clusters even when the analyzed factor has complex and composite binding specificity, as is the case for CTCF.

DISCUSSION

We introduced here DynaMIT, a flexible and extensible toolkit aimed at motif identification and integration. With its dynamic structure, it allows new components to be plugged-in and used seamlessly along the many pre-implemented ones provided. We thus think DynaMIT is a very powerful workbench for studying motifs of multiple derivations and types in combination, at an unprecedented level of integration. We included a wide number of pre-implemented motif searchers, integration strategies and visualization components in the toolkit: this enables users to run DynaMIT right away without having to develop new components, although one of its strengths lies in this possibility.

One fundamental open issue in this type of analysis lies in integrated motifs scoring and prioritization: we attempted at providing a solution by implementing the *ClusterEvaluationPrinter*, which takes into account several aspects of the clusters and their composing instances. While evaluating parameters which are intrinsically related to the clusters (i.e. coherence, size, etc.) is relatively straightforward, the same cannot be said for motif instances: search algorithms indeed often employ widely different scoring systems, not always directly comparable. Devising a uniform scoring system, potentially by re-scoring motif instances independently, could be of considerable help in obtaining an unbiased evaluation of the clusters.

While the overall structure of the toolkit will remain unchanged in the future (due to its generality and flexibility), we will continue to improve and extend its components: first of all, our efforts will focus on providing additional motif searchers (especially newly developed ones). In that respect, a rising area of research is the one of

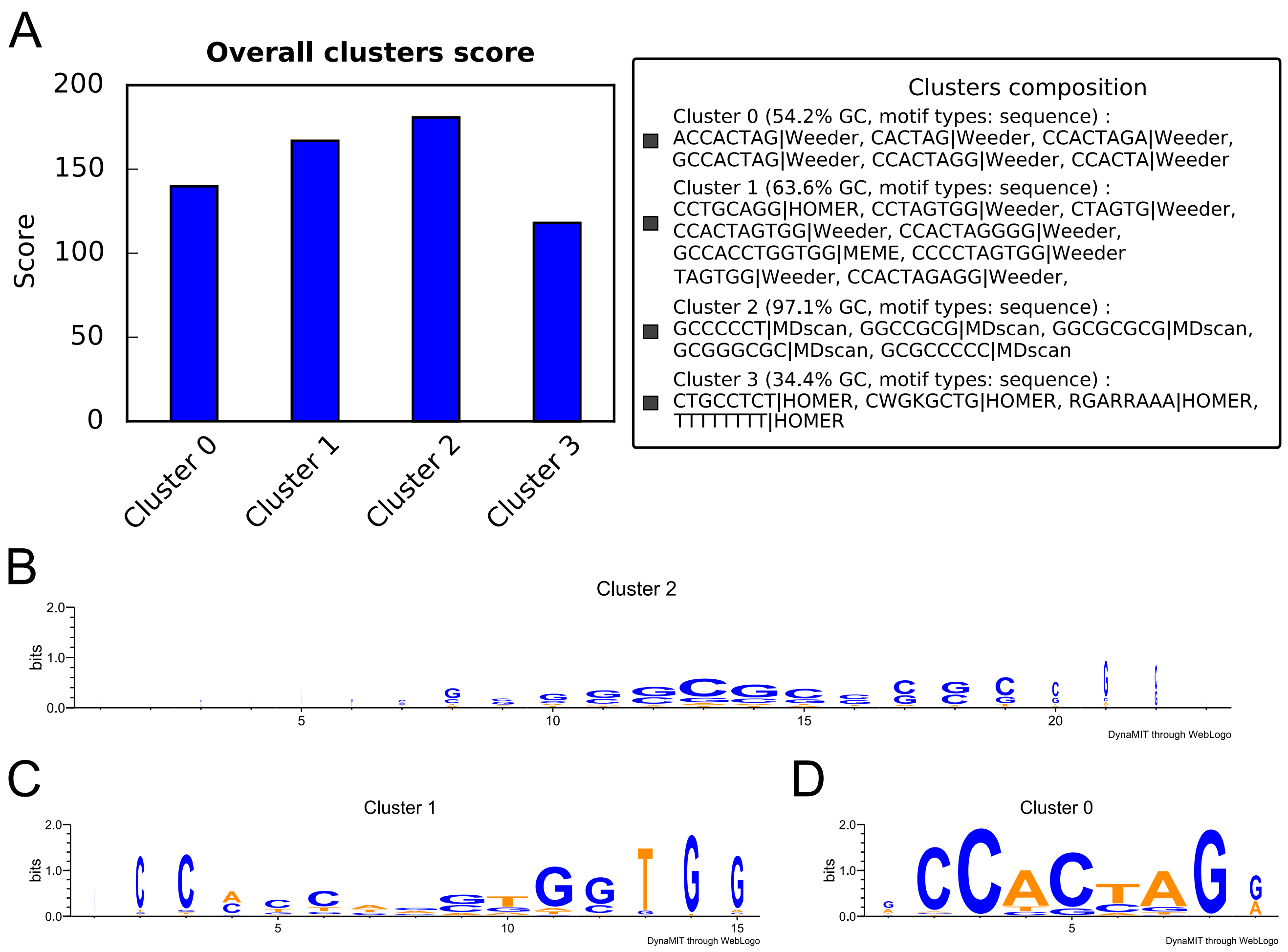


Figure 6. CTCF binding preferences can be identified by DynaMIT. The figure displays the results of running DynaMIT on the third case example. **(A)** displays *ClusterEvaluationPrinter* results, with the three top-scoring clusters representing the most frequent CTCF binding specificities. **(B)** displays cluster instances consensus for Cluster 2 as a WebLogo. **(C)** displays cluster instances consensus for Cluster 1 as a WebLogo. **(D)** displays cluster instances consensus for Cluster 0 as a WebLogo.

discriminative/ranking-based tools such as SeAMotE (7) and TEISER (39); while these algorithms require additional data with respect to what is currently requested by DynaMIT, this aspect can be addressed through the current configuration system, thus making it possible to include such tools. Concerning integration strategies, developing different integration paradigms (aside from clustering-based ones) could provide further perspectives to observe this kind of data. One potentially interesting approach consists in trying to learn integrated motifs by considering only the instances sequence, thus ignoring their grouping into motifs made by individual tools. Also new result visualization modes will of course be implemented, with particular focus on efficient ways to display the positional relationships between instances of the identified motif clusters: this could potentially exploit the drawing approach of tools such as Circos (40). Furthermore, a visualization mode highlighting motif cluster instances and their overlap on the RNA secondary structures could also be particularly useful.

Eventually, we are confident that the bioinformatics community will take on the challenge of implementing additional components for DynaMIT, further broadening its usefulness and applicability.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENT

We would like to thank Nicola Segata for critically reading the manuscript and providing constructive advices toward its improvement.

FUNDING

Laboratory of Translational Genomics funds, University of Trento. Fund for open access charge: Laboratory of Translational Genomics funds, obtained from the University of Trento, will provide the funding.
Conflict of interest statement. None declared.

REFERENCES

1. Stormo,G.D. (2000) DNA binding sites: representation and discovery. *Bioinformatics*, **16**, 16–23.
2. Tompa,M., Li,N., Bailey,T.L., Church,G.M., De Moor,B., Eskin,E., Favorov,A.V., Frith,M.C., Fu,Y., Kent,W.J. *et al.* (2005) Assessing computational tools for the discovery of transcription factor binding sites. *Nat. Biotechnol.*, **23**, 137–144.
3. Das,M.K. and Dai,H.K. (2007) A survey of DNA motif finding algorithms. *BMC Bioinformatics*, **8**(Suppl. 7), S21.
4. Weirauch,M.T., Cote,A., Norel,R., Annala,M., Zhao,Y., Riley,T.R., Saez-Rodriguez,J., Cokelaer,T., Vedenko,A., Talukder,S. *et al.* (2013) Evaluation of methods for modeling transcription factor sequence specificity. *Nat. Biotechnol.*, **31**, 126–134.
5. Bailey,T.L., Boden,M., Buske,F.A., Frith,M., Grant,C.E., Clementi,L., Ren,J., Li,W.W. and Noble,W.S. (2009) MEME SUITE: tools for motif discovery and searching. *Nucleic Acids Res.*, **37**, W202–W208.
6. Hochsmann,M., Toller,T., Giegerich,R. and Kurtz,S. (2003) Local similarity in RNA secondary structures. *Proc. IEEE Comput. Soc. Bioinform. Conf.*, **2**, 159–168.
7. Agostini,F., Cirillo,D., Ponti,R.D. and Tartaglia,G.G. (2014) SeAMotE: a method for high-throughput motif discovery in nucleic acid sequences. *BMC Genomics*, **15**, 925–933.
8. Fan,D., Bitterman,P.B. and Larsson,O. (2009) Regulatory element identification in subsets of transcripts: comparison and integration of current computational methods. *RNA*, **15**, 1469–1482.
9. Okumura,T., Makiguchi,H., Makita,Y., Yamashita,R. and Nakai,K. (2007) Melina II: a web tool for comparisons among several predictive algorithms to find potential motifs from promoter regions. *Nucleic Acids Res.*, **35**, W227–W231.
10. Ma,Q., Liu,B., Zhou,C., Yin,Y., Li,G. and Xu,Y. (2013) An integrated toolkit for accurate prediction and analysis of cis-regulatory motifs at a genome scale. *Bioinformatics.*, **29**, 2261–2268.
11. Thomas-Chollier,M., Defrance,M., Medina-Rivera,A., Sand,O., Herrmann,C., Thieffry,D. and van Helden,J. (2011) RSAT 2011: regulatory sequence analysis tools. *Nucleic Acids Res.*, **39**, W86–W91.
12. Stegmaier,P., Kel,A., Wingender,E. and Borlak,J. (2013) A discriminative approach for unsupervised clustering of DNA sequence motifs. *PLoS Comput. Biol.*, **9**, e1002958.
13. Klepper,K. and Drablos,F. (2013) MotifLab: a tools and data integration workbench for motif discovery and regulatory sequence analysis. *BMC Bioinformatics*, **14**, 9–23.
14. Cock,P.J., Antao,T., Chang,J.T., Chapman,B.A., Cox,C.J., Dalke,A., Friedberg,I., Hamelryck,T., Kauff,F., Wilczynski,B. *et al.* (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics.*, **25**, 1422–1423.
15. Pedregosa,F., Varoquaux,G., Gramfort,A., Michel,V., Thirion,B., Grisel,O., Blondel,M., Prettenhofer,P. *et al.* (2011) Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, **12**, 2825–2830.
16. Jone,E., Oliphant,T. and Peterson,P. (2001) SciPy: Open Source Scientific Tools for Python. http://www.scipy.org.
17. Hunter,J.D. (2007) Matplotlib: A 2D Graphics Environment. *Comput. Sci. Eng.*, **9**, 90–95.
18. Crooks,G.E., Hon,G., Chandonia,J.M. and Brenner,S.E. (2004) WebLogo: a sequence logo generator. *Genome Res.*, **14**, 1188–1190.
19. Yao,Z., Weinberg,Z. and Ruzzo,W.L. (2006) CMfinder–a covariance model based RNA motif finding algorithm. *Bioinformatics.*, **22**, 445–452.
20. Thompson,W.A., Newberg,L.A., Conlan,S., McCue,L.A. and Lawrence,C.E. (2007) The Gibbs Centroid Sampler. *Nucleic Acids Res.*, **35**, W232–W237.
21. Maticzka,D., Lange,S.J., Costa,F. and Backofen,R. (2014) GraphProt: modeling binding preferences of RNA-binding proteins. *Genome Biol.*, **15**, R17.
22. Hiller,M., Pudimat,R., Busch,A. and Backofen,R. (2006) Using RNA secondary structures to guide sequence motif finding towards single-stranded regions. *Nucleic Acids Res.*, **34**, e117.
23. Kruger,J. and Rehmsmeier,M. (2006) RNAhybrid: microRNA target prediction easy, fast and flexible. *Nucleic Acids Res.*, **34**, W451–W454.
24. Zambelli,F. and Pavesi,G. (2015) De Novo Secondary Structure Motif Discovery Using RNAProfile. *Methods Mol. Biol.*, **1269**, 49–62.
25. 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**, W199–W203.
26. Liu,X.S., Brutlag,D.L. and Liu,J.S. (2002) An algorithm for finding protein-DNA binding sites with applications to chromatin-immunoprecipitation microarray experiments. *Nat. Biotechnol.*, **20**, 835–839.
27. Heinz,S., Benner,C., Spann,N., Bertolino,E., Lin,Y.C., Laslo,P., Cheng,J.X., Murre,C., Singh,H. and Glass,C.K. (2010) Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and B cell identities. *Mol. Cell*, **38**, 576–589.
28. Dassi,E., Re,A., Leo,S., Tebaldi,T., Pasini,L., Peroni,D. and Quattrone,A. (2014) AURA 2: Empowering discovery of post-transcriptional networks. *Translation*, **2**, e27738.
29. Ray,D., Kazan,H., Cook,K.B., Weirauch,M.T., Najafabadi,H.S., Li,X., Gueroussov,S., Albu,M., Zheng,H., Yang,A. *et al.* (2013) A compendium of RNA-binding motifs for decoding gene regulation. *Nature*, **499**, 172–177.
30. Cook,K.B., Kazan,H., Zuberi,K., Morris,Q. and Hughes,T.R. (2011) RBPDB: a database of RNA-binding specificities. *Nucleic Acids Res.*, **39**, D301–D308.
31. Sandve,G.K., Abul,O., Walseng,V. and Drablos,F. (2007) Improved benchmarks for computational motif discovery. *BMC Bioinformatics*, **8**, 193–206.
32. Gruber,A.R., Fallmann,J., Kratochvill,F., Kovarik,P. and Hofacker,I.L. (2011) AREsite: a database for the comprehensive investigation of AU-rich elements. *Nucleic Acids Res.*, **39**, D66–D69.
33. R Core Team. (2014) *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, http://www.R-project.org.
34. Dassi,E., Zuccotti,P., Leo,S., Provenzani,A., Assfalg,M., D'Onofrio,M., Riva,P. and Quattrone,A. (2013) Hyper conserved elements in vertebrate mRNA 3'-UTRs reveal a translational network of RNA-binding proteins controlled by HuR. *Nucleic Acids Res.* **41**, 3201–3216.
35. Mukherjee,N., Jacobs,N.C., Hafner,M., Kennington,E.A., Nusbaum,J.D., Tuschl,T., Blackshear,P.J. and Ohler,U. (2014) Global target mRNA specification and regulation by the RNA-binding protein ZFP36. *Genome Biol.*, **15**, R12.
36. Mukherjee,N., Corcoran,D.L., Nusbaum,J.D., Reid,D.W., Georgiev,S., Hafner,M., Ascano,M. Jr, Tuschl,T., Ohler,U. and Keene,J.D. (2011) Integrative regulatory mapping indicates that the RNA-binding protein HuR couples pre-mRNA processing and mRNA stability. *Molecular cell*, **43**, 327–339.
37. Wang,J., Zhuang,J., Iyer,S., Lin,X., Whitfield,T.W., Greven,M.C., Pierce,B.G., Dong,X., Kundaje,A., Cheng,Y. *et al.* (2012) Sequence features and chromatin structure around the genomic regions bound by 119 human transcription factors. *Genome Res.*, **22**, 1798–1812.
38. Ong,C.T. and Corces,V.G. (2014) CTCF: an architectural protein bridging genome topology and function. *Nat. Rev. Genet.*, **15**, 234–246.
39. Goodarzi,H., Najafabadi,H.S., Oikonomou,P., Greco,T.M., Fish,L., Salavati,R., Cristea,I.M. and Tavazoie,S. (2012) Systematic discovery of structural elements governing stability of mammalian messenger RNAs. *Nature*, **485**, 264–268.
40. Krzywinski,M., Schein,J., Birol,I., Connors,J., Gascoyne,R., Horsman,D., Jones,S.J. and Marra,M.A. (2009) Circos: an information aesthetic for comparative genomics. *Genome Res.*, **19**, 1639–1645.