Published online 14 April 2015 *Nucleic Acids Research, 2015, Vol. 43, No. 13* ***e87***
doi: 10.1093/nar/gkv300

cMonkey$_2$: Automated, systematic, integrated detection of co-regulated gene modules for any organism

David J. Reiss[1,*], Christopher L. Plaisier[1], Wei-Ju Wu[1] and Nitin S. Baliga[1,2,*]

[1]Institute for Systems Biology, 401 Terry Ave N, Seattle, WA 98109, USA and [2]Department of Microbiology, University of Washington, Seattle, WA 98103, USA

Received October 17, 2014; Revised March 05, 2015; Accepted March 26, 2015

ABSTRACT

The `cMonkey` integrated biclustering algorithm identifies conditionally co-regulated modules of genes (biclusters). `cMonkey` integrates various orthogonal pieces of information which support evidence of gene co-regulation, and optimizes biclusters to be supported simultaneously by one or more of these prior constraints. The algorithm served as the cornerstone for constructing the first global, predictive Environmental Gene Regulatory Influence Network (EGRIN) model for a free-living cell, and has now been applied to many more organisms. However, due to its computational inefficiencies, long run-time and complexity of various input data types, `cMonkey` was not readily usable by the wider community. To address these primary concerns, we have significantly updated the `cMonkey` algorithm and refactored its implementation, improving its usability and extendibility. These improvements provide a fully functioning and user-friendly platform for building co-regulated gene modules and the tools necessary for their exploration and interpretation. We show, via three separate analyses of data for *E. coli*, *M. tuberculosis* and *H. sapiens*, that the updated algorithm and inclusion of novel scoring functions for new data types (e.g. ChIP-seq and transcription factor over-expression [TFOE]) improve discovery of biologically informative co-regulated modules. The complete `cMonkey`$_2$ software package, including source code, is available at https://github.com/baliga-lab/cmonkey2.

INTRODUCTION

It is widely acknowledged that gene regulatory networks (GRNs) are inherently modular in nature and organized hierarchically (1–3). This modular structure results from the regulation of genes by distinct combinations of regulatory factors; transcripts regulated by the same (set of) factor(s) are presumed to express similar patterns of differential expression over different cellular and environmental conditions. Such modularity is evident in GRNs across organisms, from the simplest prokaryotes to complex metazoans. Therefore, identifying co-regulated gene modules can significantly reduce the complexity of the problem of inference of genome-wide GRNs from data, and they can be exploited to greatly improve the accuracy of the inferred regulatory network topology (4–7).

For this reason, the detection of co-regulated gene modules via integrated modeling of multiple supporting data types has been an active research topic for more than a decade. Since the seminal integrated ‘module networks’ publications of Segal *et al.* (5,8), SAMBA (2) and `cMonkey` (9), many groups have released tools with similar overarching goals of data integration for the discovery of conditionally co-regulated modules, using various underlying statistical models and optimization methods. For example, LeMoNe (4) infers co-regulated gene modules from expression data, including detection of conditionality of co-regulation. DISTILLER (10) extends the LeMoNe framework to integrate *known* regulation with gene expression data. Like `cMonkey`, COALESCE (11) and Allegro (12) integrate *de novo* detection of sequence motifs with co-expression clustering to identify co-regulated gene modules and the *cis*-regulatory sequence features putatively responsible for their co-regulation. We refer the reader to a recent review (13) of integrated methods for detection of biological modules, and note the caveats presented by (14) regarding the large search-space involved (particularly for complex metazoan systems). One primary issue with all aforementioned methods and tools is the difficulty of easily extending them to new data types or organisms. Many of these are implemented as complex command-line tools or graphical user-interfaces that can only be applied to a limited predefined set of model organisms (typically, a few metazoans

*To whom correspondence should be addressed. Tel: +1 206 732 1266; Fax: +1 206 732 1299; Email: dreiss@systemsbiology.org
Correspondence may also be addressed to Nitin S. Baliga. Tel: +1 206 732 1266; Fax: +1 206 732 1299; Email: nbaliga@systemsbiology.org

[human, mouse, fruit fly, etc.], and often *E. coli* and *S. cerevisiae*).

The cMonkey integrated biclustering algorithm (9) was designed to decipher, from genome-wide measurements, the conditional co-regulation of genes by integrating different types of information which support evidence for their coregulation and effectively constrain, or regularize, the complex search space mentioned above. In the end, cMonkey produces biclusters that are constrained by one or more of these streams of data (9), over subsets of experimental measurements. The three primary data sources that were originally integrated and optimized by cMonkey were (i) transcript co-expression (similarity of expression profiles) of clustered genes across subsets of measurements; (ii) *de novo* detection of common putative *cis*-acting gene regulatory motifs (which we will hereafter abbreviate as GREs, for gene regulatory elements) in the promoters of clustered genes (putative binding locations of the same transcriptional regulators); and (iii) significant connectivity between clustered genes in functional association or physical interaction networks (implying meaningful functional association, which is often correlated with co-regulation). We used cMonkey to construct a Environmental Gene Regulatory Influence Network (EGRIN) model (15) for *Halobacterium salinarum NRC-1*, and have more recently used it to construct EGRIN models for many more organisms covering all three branches in the tree of life ((7,16–19), and unpublished). Work on the cMonkey algorithm and its implementation has been ongoing during that time, and we are now releasing a completely updated and reengineered version of the cMonkey software tool, with optimized performance, improved documentation and ease-of-use for end-users, and enhanced modularity which will make it straightforward to extend by interested developers.

The primary algorithmic modification in the new implementation is that it uses a global optimization, rather than the local, individual cluster optimization utilized by the original procedure. Additional algorithm updates include changes to the individual scoring scheme for subnetwork clustering, as well as to the heuristic used to integrate the different scores. All of these changes, which serve to improve the procedure's runtime performance by roughly 3-fold, result in additional benefits which we will elucidate below. We have reimplemented the updated algorithm into a new framework, called $\texttt{cMonkey}_2$, which improves ease-of-use for the end-user; greatly simplifies automated integration of additional data types and scoring mechanisms; and enhances the resulting output to facilitate visualization and exploration of biclusters and their associated evidence (e.g. *de novo* predicted GREs) in the context of other databases and web services. These improvements make $\texttt{cMonkey}_2$ a fully functioning, unified platform for integrating many kinds of genome-wide data to build gene coregulatory modules, plus the necessary tools to explore and use them to inform biological insights.

MATERIALS AND METHODS

Hereon, we refer to the originally published version of cMonkey as $\texttt{cMonkey}_1$. For a detailed overview of the $\texttt{cMonkey}_1$ algorithm and its data integration model, we refer the reader to (9).

$\texttt{cMonkey}_2$ algorithm modifications

In the following we describe only the relevant and notable algorithm changes which have been made in the updated version, which we call $\texttt{cMonkey}_2$. These include (i) a switch from local bicluster optimization to a global optimization procedure; (ii) a switch from a probabilistic association network-based score to a network density-based scoring function; (iii) a modified, more efficient heuristic for combining the three model components into an integrated clustering score while enabling stochastic exploration of the search space. Although all three of these modifications appear to replace rigorous statistical models and distributions with heuristics, as we will show, the practical effect is a significant decrease in algorithm run-time with no detriment to performance (to the contrary, $\texttt{cMonkey}_2$ actually achieves improvement in performance).

$\texttt{cMonkey}_1$ is a local optimization procedure, in which biclusters are *seeded*, one at a time, and then optimized individually. As each additional bicluster is generated and optimized, any overlap (in the form of gene membership) between it and previously-optimized biclusters was reduced by constraining the number of biclusters (default expected value \$v\$ = 2) into which each gene may fall. The most significant modification we have made to the algorithm is that $\texttt{cMonkey}_2$ instead performs a global optimization, that is modeled on the simple, widely-used and effective *k*-means clustering algorithm (20). After beginning with a chosen distance metric and an initial partitioning of all genes into exactly *k* clusters (\$v\$ = 1 cluster per gene), the basic *k*-means algorithm iterates between two steps until convergence: (i) (re-)assign each gene to the cluster with the closest centroid and (ii) update the centroids of each modified cluster. The updated $\texttt{cMonkey}_2$ algorithm performs an analogous set of moves with four primary distinctions relative to *k*-means: (i) the distance of each gene to the centroid of each cluster is computed using a measure that combines condition-specific expression profile similarity, similarity of putative GREs detected in gene promoters, and connectedness in one or more gene association networks (and/or additional scoring measures added via the new modular plug-in framework; see below); (ii) each gene can be (re-)assigned to more than one cluster (default \$v\$ = 2); (iii) at each step, conditions (in addition to genes) are moved among biclusters to improve their cohesiveness; and (iv) at each step, genes and conditions are not always assigned to the most appropriate clusters. We now elaborate upon these four details.

In $\texttt{cMonkey}_2$, as with standard *k*-means (as well as the original $\texttt{cMonkey}_1$), *k* must be chosen *a priori*; by default, $\texttt{cMonkey}_2$ sets *k* such that each bicluster will contain ∼20 genes on average (thus, given that each gene is assigned to \$v\$ = 2 biclusters by default, *k* is set to $N_g \times$ \$v\$/20, where N_g is the number of transcripts measured across all experiments). $\texttt{cMonkey}_2$ begins each iteration with a set of bicluster memberships m_i for each element (gene or condition) *i*, where by default $|m_i|$ = \$v\$ = 2 for genes (as described previously), and $|m_i| = N_c/2$ for conditions (N_c is the number of conditions, or measurements, in the expression data set; note that for standard k-means clustering, $|m_i| = 1$ for genes and $|m_i| = N_c$ for conditions). cMonkey$_2$ then computes log-likelihood score matrices R_{ij}, S_{ij} and T_{ij}, for membership of each element i in each bicluster j based upon, respectively, co-expression with the current gene members (**R**), similarity of GREs in gene promoters (**S**), and connectivity of genes in networks (**T**). For the network scores (**T**), the original procedure computed a p-value for enrichment of network edges among genes in each bicluster using the cumulative hypergeometric distribution. This computation was inefficient, and moreover could not account for weighted edges in the input networks, so we replaced it in cMonkey$_2$ with a more standard weighted network clustering coefficient (20) evaluated only over the genes within each bicluster.

Following computation of the individual component scores, cMonkey$_2$ computes a score matrix M_{ij} that contains the integrated score (a weighted sum of log-likelihoods, as in cMonkey$_1$) supporting the inclusion of gene i in bicluster j. At this stage cMonkey$_1$ would then train an 'iteratively-reweighted constrained logistic regression' on each bicluster's $M_{\cdot j}$ to obtain a posterior probability distribution p_{ij}, to classify potential bicluster members i based upon these scores. This procedure proved to be a significant bottleneck on algorithm performance. In cMonkey$_2$ we instead compute a kernel-based cumulative density distribution from these scores, to estimate the relative probability p_{ij} that each element i belongs in each cluster j. The width of the density distribution kernel is set dynamically to be larger for smaller (fewer gene) biclusters, so as to increase the tendency to add genes to small biclusters, rather than remove them. Whereas cMonkey$_1$ would then sample elements i from p_{ij} to stochastically add or remove elements from each bicluster j, in the new implementation cMonkey$_2$ we instead add a small amount of normally-distributed random 'noise' to the scores M_{ij} in order to achieve a similar type of stochasticity (which helps prevent the algorithm from falling into local minima; this noise decreases during the run to zero at the final iteration). The result of this noise is that at the beginning of a cMonkey$_2$ run, biclusters are rather poorly defined (co-expression, for example, is poor), but during the course of a full set of 2000 iterations, as this noise is decreased, the biclusters settle into a much more significant set of minima (Supplementary Figure S1).

At the end of each iteration, cMonkey$_2$ chooses a random subset of genes or conditions i, and moves i into bicluster j if, for any biclusters j' which it is already a member, $p_{ij} > p_{ij'}, \forall j'$, and out of the corresponding worse bicluster j' for which $p_{ij} > p_{ij'}$. Thus, as with the k-means clustering algorithm, cMonkey$_2$ performs a global optimization of all biclusters by moving elements among biclusters to improve each element's membership scores, rather than by optimizing each bicluster one-at-a-time (as cMonkey$_1$ did). Note, however, that we have introduced an added degree of stochasticity to the optimization procedure, both from the selection of a random subset of genes and conditions to be moved at each iteration, and from the randomization described above. This type of metaheuristic does not exist in the standard k-means clustering algorithm.

The cMonkey$_2$ tool: implementation details

We have completely re-implemented the cMonkey$_2$ software tool, transforming it into a data integration platform that enables non-technical researchers to easily analyze their gene expression data in the context of additional evidence, while allowing developers to extend and tailor the base functionality with minimal development effort. cMonkey$_2$ is available as a command-line-driven Python application. All aspects of the implementation center on modularity and extensibility, enabling developers to easily incorporate their novel data types and/or scoring methodologies into the procedure. For data input and integration, the tool now downloads, automatically from external databases, relevant information for nearly any microbe, including genome data and gene annotations (currently using NCBI parsed annotations from RSAT (21); Microbes Online (22)); gene functional associations (STRING (23)), expression data (Gene Expression Omnibus (GEO) (24)) and others (e.g. DoE KBase).

We implemented cMonkey$_2$ in object-oriented Python, in order to more effectively modularize and streamline the codebase. This switch alleviated all of the major speed and memory bottlenecks which were causing difficulty with the original implementation in GNU R.

For a typical cMonkey$_2$ run, the user provides a file containing a set of mRNA expression log-ratios in standard tab-delimited text file format. The user will also typically provide a three-letter KEGG organism code (e.g. 'eco' for *E. coli*), which is used to identify and automatically fetch additional information (genome sequence, annotations, operon predictions, promoter sequences, functional associations) from various scientific databases (21–23). For sequenced prokaryotes and unicellular eukaryotes (such as *S. cerevisiae*), cMonkey$_2$ provides solutions to common tasks such as name mapping and abstraction of organism-specific aspects including genomic information. Upon initialization, the tool downloads (if necessary) and caches the relevant public data locally. For example, predicted operons (25) are downloaded from Microbes Online (22), full genome sequences and gene annotations are fetched from NCBI via RSAT (21), and these data are used to parse out predicted promoter sequences for each annotated transcript (Figure 1). In cases where the organism of interest is not yet included in these databases (e.g. it is newly sequenced), custom versions of these files may be supplied in standard formats (e.g. tab-delimited, GFF, FASTA, etc.) to enable motif searching and network clustering. If these additional data are not available, biclustering on the expression data *only* is still possible, and will be performed by default.

cMonkey$_2$ scoring functions. At the core of a cMonkey$_2$ computation lies a cMonkey run. A run consists of a set of input data and configuration parameters and a set of scoring algorithms that are activated at certain iterations of the optimization procedure (i.e. following a schedule), and combined using user-specified weights. All parameters have defaults which have been configured to work well with all test cases, data sets and organisms tested (at this point, over twenty). The user can override certain configuration parameters or methods to customize the run. The configuration

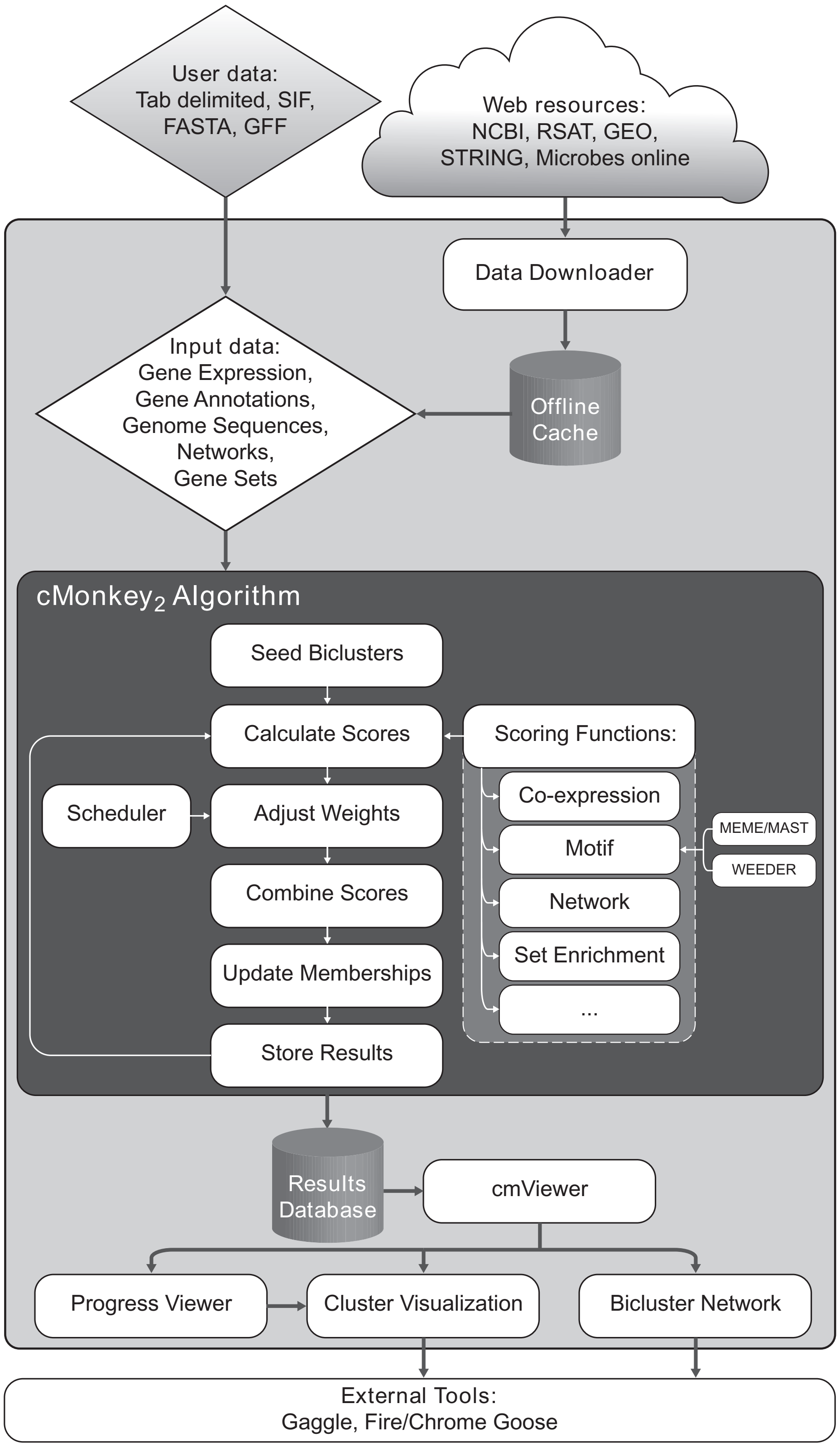

Figure 1. The `cMonkey`$_2$ data integration and analysis pipeline. Only expression data (*User data*) is required as input by the user. All other input data, if not provided by the user, is fetched automatically from online services (*Web resources*). All of these data are passed into the `cMonkey`$_2$ algorithm (dark gray area) which seeds, and then iteratively refines biclusters by integrating the output of various modularized scoring functions, following a weighting schedule (*Scheduler*) which may be customized by the user. Resulting clusters are stored in a database which may be queried by the *cmViewer* tool to view progress and visualize results.

parameters may be set via command-line or a hierarchy of user-defined and default configuration files.

At the heart of the cMonkey run is a scheduler that executes specific scoring functions at user-defined iterations and with a user-defined scoring weight. Each scoring function computes the corresponding $k \times |i|$ scores for associating a given gene/condition i with each of the k biclusters. An important detail is that these scores are not required to be comprised of standard distance measures which may be difficult or expensive to calculate. They can alternatively simply consist of a (relative) measure of prior expectation that gene/condition i belong in cluster k given the data and given the other genes/conditions in the cluster. An example of such an *ad hoc* scoring mechanism is delineated in detail in our description of the new set-enrichment scoring function (see below).

After computation of all scoring matrices, cMonkey$_2$ integrates the scores, as described in more detail below, via a combiner function, which runs a list of scoring functions in sequence and combines their results according to specified weights. Bicluster memberships (rows and columns) are then updated based upon these combined scores. A cMonkey$_2$ scoring function has a standard interface, so users can implement and run an arbitrary number of differently-weighted scoring functions based upon user preferences. The user has the choice to override the parameters of the scoring functions, via command-line options and/or configuration files. Default scoring functions for the three standard cMonkey data types—co-expression, MEME-detected conserved promoter GREs and network clustering—are provided by default. Below, we describe implementations and use-cases for two additional, newly implemented scoring functions.

Implementation and integration of novel row scoring functions

To demonstrate the utility and ease of integrating additional streams of evidence for gene co-regulation via implementation of new scoring mechanism (as described above), we implemented two additional row (gene) scoring functions which were not part of the original cMonkey$_1$ algorithm. The first of these integrates a 'set-enrichment' scoring function, which enables the user to influence bicluster optimization to enrich for user-defined sets of genes (e.g. similar gene functional annotations; known promoter binding mapped via ChIP or RNase hypersensitivity; or known GREs). The second scoring function integrates an additional motif detection algorithm (Weeder (26)) to add motif detection via enriched k-mers, and to search for motifs in regions other than gene promoters (here, we use 3' UTRs)—a functionality complementary to that already provided by MEME. Below, we describe the motivation and implementation of these two scoring functions. Later, in the Results section, we present an analysis of their influence on cMonkey$_2$ biclustering results on data sets for three different organisms.

Set-enrichment row scoring function. The set enrichment scoring function was developed to easily incorporate (and enrich biclusters for) predefined gene sets. Given a file which lists annotations or groupings of genes into (possibly overlapping) sets with unique identifiers, the set enrichment scoring function computes, at each iteration, the significance of overlap between each bicluster's member genes and the genes annotated for each set using the Fisher's exact test. For each bicluster, the set with the most significant overlap (smallest p-value) is chosen for training, and a set of row scores is generated that increases the probability of retaining cluster genes or adding new genes that are the enriched set. The gene scores are computed by a simple heuristic in which we multiply the $\log_{10}$ of the aforementioned p-value by 1.0 for genes which are in the bicluster *and* are members of the enriched set; by 0.5 for genes which are in the set but *are not* in the bicluster; and by 0.0 for all other genes. To increase stability of the set enrichment approach we added a low-pass filter which sets the p-value to a minimum of the Bonferroni cutoff p-value given the number of sets tested. These final gene (row) scores are then normalized and combined with the other scoring functions to train cMonkey$_2$ biclusters.

Weeder motif detection row scoring function for detection of enriched k*-mers.* While MEME (27) has been shown (28,29) to be among the most sensitive and robust sequence motif detection algorithms available, the fact that it models GREs as position-specific scoring matrices (PSSMs) and uses expectation maximization to detect the most overrepresented signatures in a set of input sequences means that it is more sensitive to detecting certain types of motifs. The Weeder algorithm (26) searches for overrepresented degenerate k-mers (rather than PSSMs), and has been shown (28) to be quite sensitive to detecting other types of regulatory motif signatures, including miRNA binding sites in mammalian genomes (30). Thus, in order to detect putative signatures for miRNAs in human genomes that might be associated with disease, we developed a cMonkey$_2$ scoring function that integrates Weeder motif detection and optimization into the algorithm. This custom row scoring function replaces MEME with Weeder to search for enriched motifs in gene promoters (using default parameters, in 'medium' search mode on both strands), and returns gene-specific scores assessing the significance of each promoter's match to the detected motifs. Specifically, the function records the $n \leq 4$ highest-scoring motifs, ranked by score and provided as PSSMs by Weeder. As with the default MEME scoring function, it then provides those PSSMs as input to MAST (31) to compute the significance (sequence p-values) of the match of each gene's promoter to the detected motif(s). These sequence p-values are then used by the remainder of the cMonkey$_2$ pipeline identically to the default MEME calculation.

cMonkey$_2$ output, monitoring and visualization and exploration

Provenance is a key concern in the new implementation, and all information necessary to completely reproduce a given run is stored in the output, including code version, input parameters and configuration files and random number seeds. cMonkey$_2$ provides a web interface to the output cMonkey$_2$ results database, served via an embedded web server, that enables a user to observe the progress of the run, via histograms of scores and optimization progress of mean statistics at each iteration (Supplementary Figure S4(A)). The design of this monitoring implementation as an embedded web server allows cMonkey$_2$ analyses to be run on a remote server (e.g. Amazon's EC2 platform) and monitored using a local web browser. In the future, we intend to extend this interface to enable initialization and control over cMonkey$_2$ runs, as well as remote storage, visualization and exploration of computation results.

The interface also enables searching, selection and viewing of individual biclusters during a run, and upon completion, a Cytoscape (32) network visualization for exploring relationships between genes, detected GREs, and their membership in biclusters (Supplementary Figure S4(B)). All of these resources are populated with FireGoose (33)/ChromeGoose XML microformats, which enable easy integration and exploration using other external tools and databases via the Gaggle (34), such as MeV (35) for expression analysis, STAMP (36) and RegPrecise/RegPredict (37,38) for motif comparisons, DAVID (39) for function analysis, KEGG (40) for pathway analysis, among many others. Additionally, we have implemented an interface which enables recorded exploration of biclusters via an interactive IPython notebook.

Methods for evaluation of cMonkey$_2$ module detection

E. coli*: Comparisons to other published module detection algorithms.* In order to assess the ramifications of the algorithm changes which we made to cMonkey$_2$, we evaluated its performance relative to both cMonkey$_1$, to other popular clustering methods—*k*-means (41) and WGCNA (42), and to published data integration/module detection algorithms COALESCE (11), DISTILLER (10) and LeMoNe (4). We note that this list is by no means comprehensive, but is rather meant to provide a representative sampling of the various data integration and module detection algorithms available. For all algorithms, we used an *E. coli* gene expression compendium, containing 868 measurements of mRNA expression for 4203 genes, compiled and normalized by (10). To quantify algorithm performance, we used intrinsic (tightness of cluster co-expression; motif significance) and extrinsic (recapitulation of known biology) measures. For the intrinsic measures, we used the *mean square residue* (43) and MEME (27) motif *E*-value to quantify bicluster and detected GRE quality, respectively. For the extrinsic quality assessment, we used the RegulonDB (44) database as the gold standard for comparing with known *E. coli* regulatory network modularity and known regulation. We ran all algorithms listed, other than DISTILLER, for which we used the author-provided clusterings (based upon the same *E. coli* data set). For all algorithm runs, we attempted to generate clusters with a similar average number of genes to that of the cMonkey runs (either, for example, by adjusting *k* for *k*-means, or trying different size *k*-mers for COALESCE).

Comparisons of intrinsic measures of cluster quality. For the intrinsic measures of cluster and motif detection, we used cluster *mean squared residue* (43)) to quantify cluster cohesiveness across included conditions. For motif detection, we quantified the likelihood of detecting clusters with genes that contain a statistically significant putative GRE (MEME (27) *E*-value $\leq$ 1).

Detection of known* E. coli *regulons. The primary goal of cMonkey is to reconstruct, from expression data, a comprehensive set of co-regulated gene modules. We have chosen to define a co-regulated gene module as a set of genes which are regulated by the same combination of transcription factors (TFs). We assessed cMonkey and the other algorithms in their capability to recapitulate experimentally annotated *E. coli* regulons in RegulonDB (44) using precision and recall. For precision, we computed the fraction of computed biclusters that had significant gene membership overlap (more than two genes, computed via cumulative hypergeometric *p*-value, controlled for false discovery rate FDR $\leq$ 0.01) with at least one of the 257 such combinatorial regulons in RegulonDB. For recall, we computed the fraction of all 257 combinatorial regulons which were rediscovered by the algorithm.

A standard measure of algorithm performance relative to an incomplete gold standard is an area under the precision-recall (AUPR) curve. However given our evaluations which use a single clustering for each algorithm, we use the geometric mean of precision and recall, or *G*-measure (45). Thus, for a given number of true positives (TP), false positives (FP) and false negatives (FN),

$$P = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}};\ R = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}};\ G = \sqrt{R \times P}. \qquad (1)$$

For completeness, we also report the F_1 score, which is the harmonic mean of precision and recall.

Detection of known **E. coli** *transcription factor binding sites.* cMonkey has the distinction among the algorithms tested (along with COALESCE) that it infers, *de novo*, conserved putative GREs in the promoters of genes in each cluster using MEME (27) (by default), and optimizes the clusters to improve those GREs. To evaluate this aspect of cMonkey, we assessed the performance of all algorithms in identifying groups of genes containing significant combinations of *bona fide* GREs in their promoters. For each cluster generated by each algorithm, we applied MEME for motif detection *post facto*, with the same set of parameters utilized by cMonkey, to the promoters of each cluster's member genes. Using FIMO (46), we then scanned the GREs detected by MEME across the entire *E. coli* genome to identify significant (FDR $\leq$ 0.05) motif instances. We then compared the locations of these motif instances with 2283 experimentally-determined binding locations for 101 transcription factors (TFs) with at least three binding sites in the RegulonDB *BindingSiteSet* table. If positions of a GRE aligned to the positions of a TF significantly more often than expected at random (FDR $\leq$ 0.01), then we classified that GRE as a match to the TF.

As previously, we assessed each algorithm's precision (fraction of all clusters with a match to a RegulonDB TF) and recall (fraction of all 101 RegulonDB TFs independently detected), and combined these into a single *G* measure.

***Evaluations of new set-enrichment scoring module for* E. coli.** To evaluate the efficacy of the set-enrichment module (see Methods section for details), we parsed the `RegulonDB` regulons into gene sets and integrated these into `cMonkey`$_2$ using the set enrichment row scoring function. Thus, we added an additional constraint to `cMonkey`$_2$ which allowed the algorithm to optimize biclusters that (simultaneously with the other aforementioned constraints—coexpression, GRE detection, and network connectivity) should be more consistent with the annotated *E. coli* regulons.

Mycobacterium tuberculosis*: Integration of ChIP-seq and TF overexpression via set-enrichment.* The newly-introduced `cMonkey`$_2$ set-enrichment scoring function was developed to improve the capability of `cMonkey`$_2$ to construct co-regulated gene modules which are simultaneously enriched for known gene sets. By enriching clusters for gene sets which are expected to include additional evidence for co-regulation (e.g. regulons from ChIP-chip/seq or known regulons; functional annotation such as Gene Ontology; or co-regulation via some other pre-computed evidence type). To further test the capability of the `cMonkey`$_2$ set-enrichment scoring function to improve detection of experimentally validated regulons, we investigated its influence on modules detected for *Mycobacterium tuberculosis*, using a large gene expression compendium and new global ChIP-seq and transcription factor overexpression (TFOE) measurements.

M. tuberculosis *data.* We used a compendium of 2,325 publicly available *Mycobacterium tuberculosis* transcriptome measurements collated from TBDB (http://tbdb.org), as described in (18). For our set-enrichment assessment, we integrated genome-wide binding measurements for 154 *M. tuberculosis* TFs, assayed via ChIP-seq (47), and transcriptome measurements following induction (over-expression; hereafter, TFOE) of 206 TFs (48). As described in (18), from the ChIP-seq measurements we distilled 7,248 significant TF-gene interactions through significant binding in regions spanning -150 to +70 nucleotides around transcriptional start sites for 142 TFs. Similarly, for the TFOE measurements, we used RNA measurements from the same cultures in which the TFs were induced for ChIP-Seq to obtain transcriptomes resulting from overexpression of 205 TFs. From these measurements, we identified 3,785 mRNAs with significant expression change (p-value ≤ 0.01).

***Running* `cMonkey`$_2$ *on* M. tuberculosis.** We ran `cMonkey`$_2$ on the *M.* tuberculosis data in eight different combinations: (i) without the ChIP-seq or TFOE gene sets (default); (ii) with only the ChIP-seq gene sets via the set-enrichment scoring function; (iii) with only the TFOE gene sets via the set-enrichment scoring function; and (iv) both ChIP-seq and TFOE gene sets, weighted equally via the standard `cMonkey`$_2$ weighting mechanism. We additionally ran `cMonkey`$_2$ with motif detection/integration turned off for all four combinations. For all runs, we used $k = 600$, as in (18), but excluded the (default) inclusion of EMBL STRING functional co-association networks, in order to eliminate the possibility of redundancy between test- and training data. Given the non-deterministic optimization of `cMonkey`$_2$, we ran each combination of parameterizations ten times, enabling us to report estimates of the mean and standard errors to facilitate comparisons. In total, this investigation comprised 80 separate `cMonkey`$_2$ runs.

Recovery of gene sets significantly enriched in* `cMonkey`$_2$ *modules. For each run, we used a Benjamini-Hochberg-corrected p-value (p-value ≤ 0.01) to identify the total number of biclusters (out of 600) which were significantly enriched for any of the 142 ChIP-seq gene sets or any of the 205 TFOE gene sets.

Human Lung Squamous Cell Carcinoma (LUSC). To test the capability of the new `cMonkey`$_2$ set-enrichment and Weeder scoring functions to improve detection of validated regulons in mammalian systems, we investigated their performance on The Cancer Genome Atlas (TCGA) lung squamous cell carcinoma data set. In particular, we assessed the recovery of miRNA regulators (using 3' UTR sequences as input for training to the Weeder scoring function, and a pre-computed database of miRNA to target gene predictions as training input for the set-enrichment function).

Human LUSC data. We downloaded RNA-seq gene level counts for 20 351 genes across 475 lung squamous cell carcinoma (LUSC) tumors (49) from the October 17, 2014 run of the Broad TCGA GDAC Firehose (doi:10.7908/C1CJ8CFD). Using DESeq2 (50) we normalized the RNA-seq gene level counts using the variance-stabilizing transformation, computed the coefficient of variation for each gene, and selected the top 2000 genes with largest coefficients of variation as input for `cMonkey`$_2$. Each `cMonkey`$_2$ run detected 133 biclusters. Significant co-expression for each bicluster was ensured by filtering out all biclusters where the variance explained by the first principal component was less than the variance explained by the first principle component for randomly sampled gene sets of same size for more than 5% of the random samples. This filtering led to an average of 112 ± 8 biclusters per `cMonkey`$_2$ run on the LUSC tumors.

Running* `cMonkey`$_2$ *on LUSC tumors. We used GeneMANIA (51,52) as the gene-gene interaction network training input for human `cMonkey`$_2$ runs. A gene synonym thesaurus was created from the Ensembl BioMart database to covert between different gene identifiers. Weeder was used to discover motifs in the 3' UTR sequences that were extracted from the UCSC genome browser FTP site using the same methods we have described previously (30,53). The TargetScan database of predicted human miRNA target genes release 6.2 (54) was used for set-enrichment training. We ran `cMonkey`$_2$ on the LUSC normalized RNA-seq gene expression using three different training approaches: (i) no cis-regulatory training inputs (i.e. no *de novo* motif detection); (ii) training on *de novo*-detected 3' UTR Weeder motifs; and (iii) training only on set-enrichment using TargetScan miRNA target gene predictions. As `cMonkey`$_2$ is non-deterministic in nature we ran each of the three training approaches six times to provide estimates of the mean and standard errors to facilitate comparisons.

Human LUSC: comparing number of significant weeder motifs. The significance of Weeder motifs was determined by calculating empirical *p*-values for each bicluster Weeder motif score using pre-computed Weeder motifs scores from 1000 randomly sampled gene sets. As bicluster sizes vary we precomputed gene sets sizes from 5 to 65 genes on the interval of 5 genes and selected the closest gene set size for empirical *p*-value calculation. A Weeder motif score was considered to be significant if it had an empirical *p*-value less than or equal to 0.05. A post-hoc Weeder motif discovery and empirical *p*-value calculation was conducted to determine the number of significant motifs detected for $\texttt{cMonkey}_2$ runs that did not train on Weeder motifs. A Student's *t*-test was used to compare the number of significant Weeder motifs between the three approaches.

Recovery of miRNAs implicated in LUSC. Each bicluster was tested for enrichment of miRNA target genes from the TargetScan database of predicted human miRNA target genes as described previously (53). For each run we determined the overlap of the miRNAs enriched in biclusters with the manually curated miR2Disease database which implicates 110 different miRNAs in the etiology of lung cancer and/or non-small cell lung cancer. A Student's *t*-test was used to compare the number of LUSC miRNAs rediscovered between the three approaches.

RESULTS

Below, we summarize results of our three separate analyses described in the Methods section. Overall, $\texttt{cMonkey}_2$ proves to be a worthy successor to $\texttt{cMonkey}_1$, and, based upon our assessments of both cluster quality and recapitulation of known modularity and mechanisms in prokaryotic gene regulatory networks, is an excellent tool for this purpose. We also would like to note that these comparisons may be supplemented by our previously-published evaluations of $\texttt{cMonkey}_1$, which included extensive performance comparisons with many other biclustering methods, and an analysis of performance on randomized and shuffled data sets.

Evaluation and comparison of module detection for *E. coli*

In the following sections, we evaluate the performance of $\texttt{cMonkey}_2$ in recapitulating known regulation as annotated in the RegulonDB database (see Methods section for details). Results of all comparisons are summarized in Table 1 and plotted in Supplementary Figure S2.

Intrinsic measures of cluster quality. When compared to the algorithms tested (see Methods section, we found that $\texttt{cMonkey}_2$ identified clusters with, on average, tighter co-expression (cluster *mean squared residue* (43)), and, other than $\texttt{cMonkey}_1$, with a greater likelihood of containing a statistically significant GRE (MEME (27) *E*-value ≤ 1) (Table 1 and Supplementary Figure S2). Typically, there is, somewhat paradoxically, a tradeoff between obtaining tight co-expression and detecting significant GREs. Thus it is noteworthy that $\texttt{cMonkey}_2$ obtained tighter clusters, while still detecting more clusters with more statistically significant GREs. While $\texttt{cMonkey}_1$ clusters were more likely to contain a significant motif (96%), this is primarily because it is both (a) training more heavily on GREs than on expression data, which explains the less coherent (higher residual) $\texttt{cMonkey}_1$ biclusters; and (b) redundantly detecting the same significant GREs in multiple redundant clusters (i.e. achieving greater precision at the expense of lower recall). This also explains the reason that even $\texttt{cMonkey}_2$ (no motif), for example, achieved greater recall (and hence greater *G* score) than $\texttt{cMonkey}_1$. The modified algorithm of $\texttt{cMonkey}_2$, which only allows each gene to be assigned to no more than two biclusters, is far more stringent than the probabilistic constraint in $\texttt{cMonkey}_1$. A similar effect explains the greater precision of some other algorithms (e.g. WGCNA) than cMonkey—the discovery of relatively fewer (e.g. only 25 by WGCNA) and significantly larger ($\sim 8 \times$ larger, for WGCNA) modules, enables it to focus on only the most significant (and thus most easily characterized) modules.

Detection of known E. coli *regulons.* Surprisingly, $\texttt{cMonkey}_2$ detected combinatorial regulons with substantially greater precision (51% vs. 42%) *and* recall (79% vs. 50%) than $\texttt{cMonkey}_1$. In fact, $\texttt{cMonkey}_2$ achieved greater recall than all algorithms tested, and greater precision than all except COALESCE (54%). $\texttt{cMonkey}_2$ achieved the greatest *G* score (see Methods section) for combinatorial regulon detection vs. RegulonDB (Table 1 and Supplementary Figure S2; using the F_1 score instead (Methods section) produces the same outcome). If we instead compare performance in recovering standard regulons (as opposed to combinatorial regulons; see Methods section), $\texttt{cMonkey}_2$ again achieved the greatest *G* (and F_1). It is noteworthy that $\texttt{cMonkey}_2$ surpassed even DISTILLER on these combined measures, even though DISTILLER uses known regulon data as part of its training set, which gives it greater precision.

Detection of known E. coli *GREs.* $\texttt{cMonkey}_2$ again outperformed $\texttt{cMonkey}_1$ in both precision and recall for detection of validated GRE sites in RegulonDB (Table 1 and Figure S2; see Methods section). It also achieved greater recall than all other assessed methods, although with relatively lower precision than COALESCE. However, $\texttt{cMonkey}_2$ surpassed all methods in the combined precision-recall *G* (and F_1) measure. While DISTILLER achieved greater performance than the original $\texttt{cMonkey}_1$ in these measures on our *E. coli* GRE detection benchmarks, primarily due to its greater precision, our analysis reveals that the algorithm modifications in $\texttt{cMonkey}_2$ have enabled it to outperform all methods.

Evaluation of cMonkey$_2$ integration of motifs and networks for E. coli. In order to evaluate whether the data integration scheme of $\texttt{cMonkey}_2$ performed as expected, we included results for runs of $\texttt{cMonkey}_2$ in which motifs and/or networks were not included as part of the training data. Not surprisingly, using the full complement of data performs significantly better than excluding motif information. However, we found that excluding only the network data (here, STRING (23) functional association links) did not significantly handicap the algorithm; although exclud-

Table 1. Summary of intrinsic and extrinsic measures of module construction performance on *E.* coli data set, for several algorithms, as described in **Methods** and also shown in Figure S2

							GRE			Regulon			Combi-Regulon[7]		
Method	k[1]	$<E \leq 1>$[2]	$<N_g>$[3]	$<\mathrm{resid}>$[4]	$\left\langle \frac{N_{cl}}{\mathrm{Gene}} \right\rangle$[5]	Gene Cvg.[6]	Recall	Prec.	G	Recall	Prec.	G	Recall	Prec.	G
cMonkey$_2$	400	0.69	21.0	0.61	2.0	1.00	0.61	0.29	0.39	0.89	0.52	0.65	0.79	0.51	0.62
no motif	400	0.13	21.0	0.60	2.0	1.00	0.43	0.21	0.28	0.80	0.46	0.58	0.72	0.46	0.56
no net	400	0.75	21.0	0.61	2.0	1.00	0.63	0.31	0.42	0.82	0.47	0.60	0.73	0.49	0.59
no mot, net	400	0.11	21.0	0.60	2.0	1.00	0.43	0.21	0.28	0.60	0.30	0.40	0.49	0.28	0.35
set-enrich	400	0.71	21.0	0.61	2.0	1.00	0.60	0.28	0.38	0.94	0.55	0.69	0.79	0.59	0.68
cMonkey$_1$	400	0.96	25.7	0.68	2.6	0.94	0.52	0.24	0.33	0.64	0.42	0.51	0.50	0.42	0.46
DISTILLER	150	0.09	19.1	0.64	1.2	0.57	0.38	0.43	0.40	0.41	0.55	0.47	0.27	0.31	0.28
LeMoNe[8]	64	0.34	60.0	0.70	1.0	0.92	0.17	0.34	0.23	0.26	0.34	0.29	0.23	0.38	0.28
LeMoNe[8]	373	0.27	60.5	0.63	6.9	0.78	0.42	0.23	0.30	0.66	0.53	0.59	0.62	0.54	0.58
COALESCE[9]	175	0.38	99.5	0.64	5.1	0.81	0.24	0.25	0.24	0.60	0.58	0.59	0.50	0.61	0.55
WGCNA	25	0.52	168.0	0.73	1.0	1.00	0.11	0.48	0.18	0.09	0.28	0.13	0.11	0.36	0.17
***k*-means**	213	0.12	19.7	0.66	1.0	1.00	0.44	0.32	0.37	0.51	0.32	0.39	0.43	0.32	0.37

Notes: (1) Total number of clusters/modules detected. (2) Fraction of clusters containing a detected motif with MEME *E*-value ≤ 1. (3) Mean number of genes over all clusters. (4) Mean cluster mean squared residue. (5) Mean number of clusters per gene. (6) Gene coverage: fraction of all genes included in at least one cluster. (7) 'Combi-Regulon' is short for combinatorial regulon (see **Methods**). (8) Default LeMoNe parameters except minimum cluster size of 3. (9) COALESCE *k*-mer length of 7 (default) or 8.

ing both networks *and* motifs performed significantly worse than only excluding either data type separately.

Enrichment for E. coli *known regulons via new set-enrichment row scoring module.* We will now present the results of integrating the two novel cMonkey$_2$ scoring modules (described in Methods section) in more detail and evaluate their utility in improving the method's performance in recapitulating RegulonDB regulons and known GREs.

Table 1 and Figure S2 shows that this integration effectively improved the cMonkey$_2$ recapitulation of RegulonDB regulons and combinatorial regulons (particularly, the precision of regulon detection), while not significantly impeding its ability to meet the other default constraints of tight conditional co-expression and significant GRE detection. Clearly, the degree to which this module can improve these measures (and by result decrease other measures) depends upon adjustment of its weighting schedule. While we acknowledge the circularity of this assessment, it proves that the set enrichment row scoring module has the intended effect and could be effectively used to integrate ChIP, functional annotations, or related data types into the cMonkey$_2$ module detection process.

Evaluation of co-regulated modules detected for *Mycobacterium tuberculosis*

All assessments of cMonkey$_2$ module predictions for *M. tuberculosis* are summarized in Table 2 and Supplementary Figure S3.

Set-enrichment scoring function significantly increases recovery of set-enriched modules. Recently we reported the construction of a global gene regulatory network for *Mycobacterium tuberculosis* (Mtb) by applying cMonkey to 2,325 publicly available Mtb transcriptome profiles (18). We previously validated this predicted network using two separate global data sets: (i) genome-wide binding locations for 143 TFs measured via ChIP-seq (47), and (ii) global transcriptional consequences of overexpressing 206 TFs (TFOE) (48). We hypothesized that training on these experimentally determined TF-regulated genes from ChIP-seq or TFOE would improve cMonkey$_2$ gene regulatory module inference. We tested this hypothesis by applying cMonkey$_2$ analysis to the same transcriptome profiles for Mtb as used previously (18), and varying the training inputs (MEME *de novo cis*-regulatory motif detection; ChIP-seq set-enrichment; and TFOE set-enrichment).

Importantly, we first found that integration of MEME motifs in cMonkey$_2$ optimization significantly increased the number of modules that were enriched for both ChIP-seq (350 vs. 187 significantly enriched modules, Student's *t*-test *p*-value $= 5.0 \times 10^{-12}$, Table 2) and TFOE (346 vs. 249 significantly enriched modules, *p*-value $= 6.2 \times 10^{-10}$, Table 2) TF targets, in comparison runs in which motifs were excluded from training (i.e. co-expression alone). This result demonstrates clearly that the cMonkey$_2$ integration of sequence information from MEME *de novo* motif detection significantly improves discovery of biclusters that are enriched for Mtb TFs.

Using the set-enrichment scoring function to train on ChIP-seq or TFOE (while excluding motif detection), also significantly increased the number of enriched modules beyond what was discovered by co-expression training alone (ChIP-seq *p*-value $= 2.3 \times 10^{-9}$, TFOE *p*-value $= 2.5 \times 10^{-13}$, Table 2), or to the runs trained on MEME motifs (ChIP-seq *p*-value $= 1.3 \times 10^{-7}$; TFOE *p*-value $= 5.8 \times 10^{-7}$, Table 2). Notably, this improvement was achieved with relatively little decrease in bicluster co-expression (i.e. little increase in residual, Table 2), suggesting that, by integrating this alternate form of prior *cis*-regulatory information, cMonkey$_2$ is effectively exploiting the complex, multifactorial biclustering search space to result in modules with similar co-expression 'quality,' but which are significantly more enriched with the desired sets.

Due to the indirect nature of TFOE responses (vs. inherently direct interactions measured via ChIP), the condition sensitivity of the experiments, and the noise inherent to ChIP-seq measurements, there is a small amount of overlapping cis-regulatory information between the ChIP-seq and TFOE data sets (18), which leads to little to no increase in the number of TFOE set-enriched modules detected when trained on ChIP-seq sets, and vice versa. The complementary nature of these data sets and the lack of a gold-standard set of cis-regulatory predictions meant that we did not have an independent validation that could be used to assess the

Table 2. Summary of cMonkey$_2$ module construction on *M.* tuberculosis expression data set, including varying combinations of additional prior data (ChIP-seq and TFOE; see **Methods**) via the set-enrichment scoring function

	Expression		ChIP-Seq[(a)]		TFOE[(a)]		Both[(a)]	
	Motif	No Motif	Motif	No Motif	Motif	No Motif	Motif	No Motif
mean residual	0.53 ± 0.01	0.49 ± 0.01	0.53 ± 0.01	0.49 ± 0.01	0.53 ± 0.01	0.50 ± 0.01	0.53 ± 0.01	0.50 ± 0.01
mean motif log$_{10}$ p-val.	-9.14 ± 0.07	–	-9.21 ± 0.10	–	-9.19 ± 0.08	–	-9.31 ± 0.12	–
clusters w. motif $E \leq 1$	564.8 ± 7.6	–	565.3 ± 7.2	–	563.0 ± 2.9	–	571.5 ± 3.4	–
ChIP-seq								
clusters signif.[(1)]	349.5 ± 5.8	187.3 ± 7.7	397.5 ± 8.5	418.1 ± 30.4	349.9 ± 11.2	182.8 ± 9.9	395.3 ± 8.9	419.2 ± 11.9
TFs signif.[(2)]	142.0 ± 0.0	133.4 ± 3.5	142.0 ± 0.0	141.5 ± 1.0	142.0 ± 0.0	129.9 ± 4.8	142.0 ± 0.0	142.0 ± 0.0
unique clusters signif.[(1)]	95.7 ± 3.2	87.1 ± 3.9	97.6 ± 4.3	116.2 ± 2.5	93.0 ± 3.8	88.1 ± 4.0	98.3 ± 3.9	118.4 ± 4.3
unique TFs signif.[(2)]	135.6 ± 1.8	134.7 ± 2.9	137.0 ± 1.3	136.5 ± 1.5	136.3 ± 1.6	136.2 ± 1.2	136.5 ± 1.7	135.8 ± 3.3
TFOE								
clusters signif.[(1)]	346.3 ± 7.8	249.1 ± 9.7	347.8 ± 12.5	255.2 ± 10.3	402.2 ± 11.3	485.5 ± 15.8	413.1 ± 14.9	491.1 ± 7.2
TFs signif.[(3)]	198.8 ± 1.8	179.0 ± 3.9	200.0 ± 1.9	181.4 ± 5.6	201.6 ± 1.7	199.5 ± 1.8	202.8 ± 1.0	199.8 ± 5.4
unique clusters signif.[(1)]	138.9 ± 3.8	136.2 ± 5.3	135.6 ± 2.4	137.2 ± 5.7	144.4 ± 3.5	174.9 ± 4.6	146.3 ± 8.2	171.9 ± 4.2
unique TFs signif.[(3)]	191.7 ± 2.7	189.2 ± 3.4	191.6 ± 2.6	188.4 ± 3.2	189.6 ± 2.9	190.8 ± 2.3	191.0 ± 5.0	190.7 ± 2.5

Shown are statistics regarding recapitulation of TF target gene sets in the ChIP-seq and TFOE measurements, for cMonkey$_2$ runs on data with inclusion of varying prior information. The values in the table rows labeled 'clusters signif.' (number of significant clusters) correspond to those in the bar chart of Figure S3. The rows labeled 'Unique clusters/TFs signif.' denote clusters/TFs with a single unique match to a TF/cluster, respectively, and respectively represent precision and recall.
Notes: [(a)] All runs included expression data as well. [(1)] Out of a total of 600 clusters predicted. [(2)] Out of 142 TFs tested via ChIP-seq. [(3)] Out of 205 TFs tested via overexpression (TFOE).

strength of these predictions. We address this lack of external validation in the following section on lung squamous cell carcinoma where an external validation set is available.

Evaluation of module detection for human Lung Squamous Cell Carcinoma (LUSC)

Increased recovery of LUSC-implicated miRNAs by training on Weeder motifs. Previously, we developed methods to discover miRNA mediated regulation from gene co-expression clustering by discovering 3' UTR motifs using Weeder *post-facto* on the 3' UTR sequences of genes in the clusters (53). The integration of Weeder into cMonkey$_2$ allows us to train biclusters based simultaneously on co-expression and Weeder 3' UTR motifs thereby increasing the potential for discovering meaningful miRNA co-regulation. We tested this hypothesis by applying cMonkey$_2$ to The Cancer Genome Atlas (TCGA) lung squamous cell carcinoma (LUSC) patient tumors to discover miRNA mediated co-regulation. We observed significant increase in the number of significant 3' UTR motifs discovered in when cMonkey$_2$ is trained on Weeder motifs when compared with runs not trained on any cis-regulatory inputs (p-value $=1.5 \times 10^{-3}$; Table 3). Training on Weeder 3' UTR motifs also led to a significant 2.5 fold increase in recovery of miRNAs implicated in lung cancer as compared to runs not trained on any cis-regulatory inputs (p-value $= 6.0 \times 10^{-5}$; Table 3). We have demonstrated that integration of training on Weeder 3' UTR motifs into cMonkey$_2$ has improved the discovery of 3' UTR motifs, which in turn leads to an impressive increase in the discovery of disease implicated miRNAs.

Increased recovery of LUSC-implicated miRNAs by set-enrichment training. A faster alternative for training cMonkey$_2$ runs to discover miRNA mediated regulation in mammalian species with large genomes is set-enrichment with databases of pre-computed miRNA target gene predictions such as TargetScan (54). cMonkey$_2$ was run on the TCGA LUSC patient tumors (see above) and trained, using set-enrichment, on TargetScan miRNA target gene predictions. This analysis led to a significant 2.3-fold increase in recovery of miRNAs implicated in lung cancer as compared to runs not trained on any cis-regulatory inputs (p-value $= 1.8 \times 10^{-6}$; Table 3). Importantly, there was not a significant difference in the number of miRNAs recovered between Weeder 3' UTR motifs or TargetScan miRNA target gene training approaches (p-value $= 0.24$); however omitting the *de novo* motif detection resulted in a $\sim 3 \times$ improvement in cMonkey$_2$ run time in the TargetScan set-enrichment runs (4.7 ± 0.08) hours, versus the Weeder 3' UTR motif runs 14.3 ± 0.2 hours; p-value $= 9.1 \times 10^{-5}$). These results demonstrate that if regulatory factor to target gene databases exist that set-enrichment approaches can be used in place of de novo motif detection and also lead to significant performance improvements.

DISCUSSION

We have described cMonkey$_2$, an updated and improved framework for detecting co-regulated modules of genes via automated data integration and optimization. We have described our recent modifications to the algorithm, which served to improve both its runtime performance, as well as its ability to discover optimized and experimentally validated gene regulatory modules. Based upon our tests on *E. coli*, cMonkey$_2$ proves to be a strong performer in this crowded arena of regulatory network module detection and data integration.

We have completely overhauled the cMonkey$_2$ implementation, focusing on ease-of-use for the end user (with automatic downloading and integration of many different data sources for any sequenced and annotated microbe), and on simplicity for the developer in integrating new data types and scoring schemes into the procedure. We demonstrated the utility of two new scoring mechanisms with use-cases for three different organisms—*E.* coli, *M.* tuberculosis and *H.* sapiens, and showed that a simple integration of a new set-enrichment scoring procedure, as well as a new motif detection algorithm (Weeder) improved upon the existing capability of cMonkey$_2$ to detect valid co-regulated gene modules and cis-regulatory motifs. These tests moreover demonstrated the importance of motif integration as part of cMonkey, revealing that this constraint can significantly improve module detection performance when additional

Table 3. Summary of `cMonkey`$_2$ module construction performance on *H.* sapiens lung squamous cell carcinoma data set, for several different training schemes, including novel Weeder-based 3' UTR motif detection, and pre-computed miRNA target gene predictions via novel set-enrichment scoring function. See **Methods** for details.

Training Method	Number of Significant Motifs	Comparison *p*-value[1]	miR2Disease miRNAs	Comparison *p*-value[a]
Expression only	9.3 ± 2.1	-	3.5 ± 0.84	-
Weeder motif training	16 ± 3.1	1.5×10^{-3}	8.8 ± 1.5	6.0×10^{-5}
Set-enrichment training	9.5 ± 2.3	9.0×10^{-2}	8.0 ± 0.63	1.8×10^{-6}

Notes: [1] All *p*-value comparisons are relative to 'Expression only.'

data sets (other than expression data) are not available—as would be the case, for instance, for newly-sequenced and culturable microbes with no developed genetics capability.

Because various motif detection algorithms use different statistical models and heuristics for learning them, they have different, often complementary capabilities for detecting different types of real signatures (55). For this reason, a number of researchers (56,57) have taken to integrating the predictions of several different motif detection algorithms and have demonstrated resulting increased sensitivity and/or precision on prokaryotic genomes (56,58). Our long-term goal is to integrate a number of motif detection algorithms into an ensemble learning procedure for learning co-regulated modules with `cMonkey`$_2$. This will include searching, in addition to annotated gene promoters (the current default), their 5' and 3' UTRs for potential miRNA and post-transcriptional regulatory motifs. As we have shown, `cMonkey`$_2$ now provides a straightforward framework for this type of integration, and we have identified excellent candidates (29) including BoBro (59), Spacer (60) or dyad-analysis (61), and BioProspector (62), which model DNA motifs in different ways, as additional targets for integration.

In the future, in addition to integration of multiple motif detection algorithms, we intend to use this new framework to add additional constraints via phylogenetic genomic conservation (e.g. (63) and related), as well as other new data types including genomic location constraints provided by ChIP-seq (64), ATAC-seq (65) or DNase-seq (66) measurements, which will provide significant additional constraints on the bicluster (and in particular, motif) optimization. Moreover, the framework provides the opportunity to investigate other measures of gene expression pattern similarity (e.g. mutual information) to identify other potential patterns of co-regulation.

Our desire is to see `cMonkey`$_2$ become a focal point for a community of users and developers, with additional data types and scoring function modules being contributed by the community. To this end, development of the framework is openly hosted on Github (http://github.com/baliga-lab/cmonkey2), with extensive documentation, wikis, and discussion groups. We will moreover provide a framework for automatically testing modifications and improvements contributed by the community via benchmarks similar to the `RegulonDB` ones presented here.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENTS

We would like to thank Aaron Brooks, Justin Ashworth, Sam Danziger, Frank Schmitz and Serdar Turkarslan for their input into the design and implementation of `cMonkey`$_2$. We would like to acknowledge Eliza Peterson for filtering and providing the Mtb ChIP-seq and TFOE data in machine-readable format.

FUNDING

U.S. National Science Foundation [ABI NSF-1262637, EAGER-MSB-1237267, DB-1262637, MCB-1330912]; National Institute of Allergy & Infectious Diseases of the National Institutes of Health [U 19 AI10676]; National Institute of General Medical Sciences of the National Institutes of Health [P50GM076547]; ENIGMA, supported by the Office of Science, Office of Biological & Environmental Research of the US Department of Energy Contract No. DE-AC02-05CH11231); University of Luxembourg–ISB Partnership. Funding for open access charge: U.S. National Science Foundation [ABI NSF-1262637].

Conflict of interest statement. None declared.

REFERENCES

1. Hartwell,L.H., Hopfield,J.J., Leibler,S. and Murray,A.W. (1999) From molecular to modular cell biology. *Nature*, **402**, 47–52.
2. Tanay,A., Sharan,R., Kupiec,M. and Shamir,R. (2004) Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. *Proc. Natl. Acad. Sci. U.S.A.*, **101**, 2981–2986.
3. Ihmels,J., Friedlander,G., Bergmann,S., Sarig,O., Ziv,Y. and Barkai,N. (2002) Revealing modular organization in the yeast transcriptional network. *Nat. Genet.*, **31**, 370–377.
4. Michoel,T., De Smet,R., Joshi,A., Van de Peer,Y. and Marchal,K. (2009) Comparative analysis of module-based versus direct methods for reverse-engineering transcriptional regulatory networks. *BMC Syst. Biol.*, **3**, 49.
5. Segal,E., Shapira,M., Regev,A., Pe'er,D., Botstein,D., Koller,D. and Friedman,N. (2003) Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nat. Genet.*, **34**, 166–176.
6. De Smet,R. and Marchal,K. (2010) Advantages and limitations of current network inference methods. *Nat. Rev. Microbiol.*, **8**, 717–729.
7. Brooks,A.N., Reiss,D.J., Allard,A., Wu,W.J., Salvanha,D.M., Plaisier,C.L., Chandrasekaran,S., Pan,M., Kaur,A. and Baliga,N.S. (2014) A system-level model for the microbial regulatory genome. *Mol. Syst. Biol.*, **10**, 740.
8. Segal,E., Yelensky,R. and Koller,D. (2003) Genome-wide discovery of transcriptional modules from DNA sequence and gene expression. *Bioinformatics*, **19**, 273–282.
9. Reiss,D.J., Baliga,N.S. and Bonneau,R. (2006) Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. *BMC Bioinformatics*, **7**, 280.

10. Lemmens,K., De Bie,T., Dhollander,T., De Keersmaecker,S.C., Thijs,I.M., Schoofs,G., De Weerdt,A., De Moor,B., Vanderleyden,J., Collado-Vides,J. *et al.* (2009) DISTILLER: a data integration framework to reveal condition dependency of complex regulons in Escherichia coli. *Genome Biol.*, **10**, R27.
11. Huttenhower,C., Mutungu,K.T., Indik,N., Yang,W., Schroeder,M., Forman,J.J., Troyanskaya,O.G. and Coller,H.A. (2009) Detailing regulatory networks through large scale data integration. *Bioinformatics*, **25**, 3267–3274.
12. Halperin,Y., Linhart,C., Ulitsky,I. and Shamir,R. (2009) Allegro: analyzing expression and sequence in concert to discover regulatory programs. *Nucleic Acids Res.*, **37**, 1566–1579.
13. Mitra,K., Carvunis,A.R., Ramesh,S.K. and Ideker,T. (2013) Integrative approaches for finding modular structure in biological networks. *Nat. Rev. Genet.*, **14**, 719–732.
14. Ament,S.A., Blatti,C.A., Alaux,C., Wheeler,M.M., Toth,A.L., Le Conte,Y., Hunt,G.J., Guzman-Novoa,E., Degrandi-Hoffman,G., Uribe-Rubio,J.L. *et al.* (2012) New meta-analysis tools reveal common transcriptional regulatory basis for multiple determinants of behavior. *Proc. Natl. Acad. Sci. U.S.A.*, **109**, E1801–E1810.
15. Bonneau,R., Facciotti,M.T., Reiss,D.J., Schmid,A.K., Pan,M., Kaur,A., Thorsson,V., Shannon,P., Johnson,M.H., Bare,J.C. *et al.* (2007) A predictive model for transcriptional control of physiology in a free living cell. *Cell*, **131**, 1354–1365.
16. Yoon,S.H., Turkarslan,S., Reiss,D.J., Pan,M., Burn,J.A., Costa,K.C., Lie,T.J., Slagel,J., Moritz,R.L., Hackett,M. *et al.* (2013) A systems level predictive model for global gene regulation of methanogenesis in a hydrogenotrophic methanogen. *Genome Res.*, **23**, 1839–1851.
17. Danziger,S.A., Ratushny,A.V., Smith,J.J., Saleem,R.A., Wan,Y., Arens,C.E., Armstrong,A.M., Sitko,K., Chen,W.M., Chiang,J.H. *et al.* (2014) Molecular mechanisms of system responses to novel stimuli are predictable from public data. *Nucleic Acids Res.*, **42**, 1442–1460.
18. Peterson,E.J., Reiss,D.J., Turkarslan,S., Minch,K.J., Rustad,T., Plaisier,C.L., Longabaugh,W.J., Sherman,D.R. and Baliga,N.S. (2014) A high-resolution network model for global gene regulation in Mycobacterium tuberculosis. *Nucleic Acids Res.*, **42**, 11291–11303.
19. Turkarslan,S., Wurtmann,E.J., Wu,W.-J., Jiang,N., Bare,J.C., Foley,K., Reiss,D.J., Novichkov,P. and Baliga,N.S. (2014) Network portal: a database for storage, analysis and visualization of biological networks. *Nucleic Acids Res*, **42**, D184–D190.
20. Watts,D.J. and Strogatz,S.H. (1998) Collective dynamics of ‘small-world’ networks. *Nature*, **393**, 440–442.
21. 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.
22. Alm,E.J., Huang,K.H., Price,M.N., Koche,R.P., Keller,K., Dubchak,I.L. and Arkin,A.P. (2005) The MicrobesOnline Web site for comparative genomics. *Genome Res.*, **15**, 1015–1022.
23. Szklarczyk,D., Franceschini,A., Kuhn,M., Simonovic,M., Roth,A., Minguez,P., Doerks,T., Stark,M., Muller,J., Bork,P. *et al.* (2011) The STRING database in 2011: functional interaction networks of proteins, globally integrated and scored. *Nucleic Acids Res.*, **39**, D561–D568.
24. Barrett,T., Troup,D.B., Wilhite,S.E., Ledoux,P., Rudnev,D., Evangelista,C., Kim,I.F., Soboleva,A., Tomashevsky,M. and Edgar,R. (2007) NCBI GEO: mining tens of millions of expression profiles–database and tools update. *Nucleic Acids Res.*, **35**, D760–D765.
25. Price,M.P., Arkin,A.P. and Alm,E.J. (2006) OpWise: Operons aid the identification of differentially expressed genes in bacterial microarray experiments. *BMC Bioinformatics*, **7**, 19.
26. Pavesi,G. and Pesole,G. (2006) Using Weeder for the discovery of conserved transcription factor binding sites. *Curr. Protoc. Bioinformatics*, Chapter 2, Unit 2.11.
27. 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.
28. Tompa,M., Li,N., Bailey,T., Church,G., De Moor,B., Eskin,E., Favorov,A., Frith,M., Fu,Y., Kent,W. *et al.* (2005) Assessing computational tools for the discovery of transcription factor binding sites. *Nat. Biotechnol.*, **23**, 137–144.
29. Das,M.K. and Dai,H.K. (2007) A survey of DNA motif finding algorithms. *BMC Bioinformatics*, **8**(Suppl. 7), S21.
30. Plaisier,C.L., Bare,J.C. and Baliga,N.S. (2011) miRvestigator: web application to identify miRNAs responsible for co-regulated gene expression patterns discovered through transcriptome profiling. *Nucleic Acids Res.*, **39**, W125–W131.
31. Bailey,T.L. and Gribskov,M. (1998) Methods and statistics for combining motif match scores. *J. Comput. Biol.*, **5**, 211–221.
32. Shannon,P., Markiel,A., Ozier,O., Baliga,N.S., Wang,J.T., Ramage,D., Amin,N., Schwikowski,B. and Ideker,T. (2003) Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.*, **13**, 2498–2504.
33. Bare,J.C., Shannon,P.T., Schmid,A.K. and Baliga,N.S. (2007) The Firegoose: two-way integration of diverse data from different bioinformatics web resources with desktop applications. *BMC Bioinformatics*, **8**, 456.
34. Shannon,P.T., Reiss,D.J., Bonneau,R. and Baliga,N.S. (2006) The Gaggle: an open-source software system for integrating bioinformatics software and data sources. *BMC Bioinformatics*, **7**, 176.
35. Howe,E.A., Sinha,R., Schlauch,D. and Quackenbush,J. (2011) RNA-Seq analysis in MeV. *Bioinformatics*, **27**, 3209–3210.
36. Mahony,S. and Benos,P.V. (2007) STAMP: a web tool for exploring DNA-binding motif similarities. *Nucleic Acids Res.*, **35**, W253–W258.
37. Novichkov,P.S., Laikova,O.N., Novichkova,E.S., Gelfand,M.S., Arkin,A.P., Dubchak,I. and Rodionov,D.A. (2010) RegPrecise: a database of curated genomic inferences of transcriptional regulatory interactions in prokaryotes. *Nucleic Acids Res.*, **38**, D111–D118.
38. Novichkov,P.S., Rodionov,D.A., Stavrovskaya,E.D., Novichkova,E.S., Kazakov,A.E., Gelfand,M.S., Arkin,A.P., Mironov,A.A. and Dubchak,I. (2010) RegPredict: an integrated system for regulon inference in prokaryotes by comparative genomics approach. *Nucleic Acids Res.*, **38**, W299–W307.
39. Sherman,B.T., Huang,D.W., Tan,Q., Guo,Y., Bour,S., Liu,D., Stephens,R., Baseler,M.W., Lane,H.C. and Lempicki,R.A. (2007) DAVID Knowledgebase: a gene-centered database integrating heterogeneous gene annotation resources to facilitate high-throughput gene functional analysis. *BMC Bioinformatics*, **8**, 426.
40. Kanehisa,M. and Goto,S. (2000) KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, **28**, 27–30.
41. Hartigan,J.A. and Wong,M.A. (1979) Algorithm AS 136: A K-Means Clustering Algorithm. *J. R. Stat. Soc. C*, **28**, 100–108.
42. Langfelder,P. and Horvath,S. (2008) WGCNA: an R package for weighted correlation network analysis. *BMC Bioinformatics*, **9**, 559.
43. Cheng,Y. and Church,G.M. (2000) Biclustering of expression data. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **8**, 93–103.
44. Gama-Castro,S., Salgado,H., Peralta-Gil,M., Santos-Zavaleta,A., Muiz-Rascado,L., Solano-Lira,H., Jimenez-Jacinto,V., Weiss,V., Garca-Sotelo,J.S., Lpez-Fuentes,A. *et al.* (2011) RegulonDB version 7.0: transcriptional regulation of Escherichia coli K-12 integrated within genetic sensory response units (Gensor Units). *Nucleic Acids Res.*, **39**, D98–D105.
45. Powers,D.M. (2011) Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *J. Mach. Learn. Technol.*, **2**, 37–63.
46. Grant,C.E., Bailey,T.L. and Noble,W.S. (2011) FIMO: scanning for occurrences of a given motif. *Bioinformatics*, **27**, 1017–1018.
47. Minch,K.J., Rustad,T.R., Peterson,E.J., Winkler,J., Reiss,D.J., Ma,S., Hickey,M., Brabant,W., Morrison,B., Turkarslan,S. *et al.* (2015) The DNA-binding network of Mycobacterium tuberculosis. *Nat. Commun.*, **6**, 5829.
48. Rustad,T.R., Minch,K.J., Ma,S., Winkler,J.K., Hobbs,S., Hickey,M., Brabant,W., Turkarslan,S., Price,N.D., Baliga,N.S. *et al.* (2014) Mapping and manipulating the Mycobacterium tuberculosis transcriptome using a transcription factor overexpression-derived regulatory network. *Genome Biol.*, **15**, 502.
49. Hammerman,P.S., Lawrence,M.S., Voet,D., Jing,R., Cibulskis,K., Sivachenko,A., Stojanov,P., McKenna,A., Lander,E.S., Gabriel,S. *et al.* (2012) Comprehensive genomic characterization of squamous cell lung cancers. *Nature*, **489**, 519–525.
50. Love,M.I., Huber,W. and Anders,S. (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol.*, **15**, 550.

51. Warde-Farley,D., Donaldson,S.L., Comes,O., Zuberi,K., Badrawi,R., Chao,P., Franz,M., Grouios,C., Kazi,F., Lopes,C.T. *et al.* (2010) The GeneMANIA prediction server: biological network integration for gene prioritization and predicting gene function. *Nucleic Acids Res.*, **38**, W214–W220.
52. Mostafavi,S., Ray,D., Warde-Farley,D., Grouios,C. and Morris,Q. (2008) GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. *Genome Biol.*, **9**(Suppl. 1), S4.
53. Plaisier,C.L., Pan,M. and Baliga,N.S. (2012) A miRNA-regulatory network explains how dysregulated miRNAs perturb oncogenic processes across diverse cancers. *Genome Res.*, **22**, 2302–2314.
54. Friedman,R.C., Farh,K.K., Burge,C.B. and Bartel,D.P. (2009) Most mammalian mRNAs are conserved targets of microRNAs. *Genome Res.*, **19**, 92–105.
55. Wijaya,E. (2008) *Integrative methods for discovering generic cis-regulatory motifs.* Ph.D. Thesis. *School of Computing*, National University of Singapore.
56. Wijaya,E., Yiu,S.M., Son,N.T., Kanagasabai,R. and Sung,W.K. (2008) MotifVoter: a novel ensemble method for fine-grained integration of generic motif finders. *Bioinformatics*, **24**, 2288–2295.
57. Romer,K.A., Kayombya,G.R. and Fraenkel,E. (2007) WebMOTIFS: automated discovery, filtering and scoring of DNA sequence motifs using multiple programs and Bayesian approaches. *Nucleic Acids Res.*, **35**, W217–W220.
58. Hu,J., Yang,Y.D. and Kihara,D. (2006) EMD: an ensemble algorithm for discovering regulatory motifs in DNA sequences. *BMC Bioinformatics*, **7**, 342.
59. Li,G., Liu,B., Ma,Q. and Xu,Y. (2011) A new framework for identifying cis-regulatory motifs in prokaryotes. *Nucleic Acids Res.*, **39**, e42.
60. Chakravarty,A., Carlson,J.M., Khetani,R.S., DeZiel,C.E. and Gross,R.H. (2007) SPACER: identification of cis-regulatory elements with non-contiguous critical residues. *Bioinformatics*, **23**, 1029–1031.
61. Defrance,M., Janky,R., Sand,O. and van Helden,J. (2008) Using RSAT oligo-analysis and dyad-analysis tools to discover regulatory signals in nucleic sequences. *Nat. Protoc.*, **3**, 1589–1603.
62. 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.
63. Thompson,W., Conlan,S., McCue,L.A. and Lawrence,C.E. (2007) Using the Gibbs Motif Sampler for phylogenetic footprinting. *Methods Mol. Biol.*, **395**, 403–424.
64. 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.
65. Buenrostro,J.D., Giresi,P.G., Zaba,L.C., Chang,H.Y. and Greenleaf,W.J. (2013) Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. *Nat. Methods*, **10**, 1213–1218.
66. Crawford,G.E., Holt,I.E., Whittle,J., Webb,B.D., Tai,D., Davis,S., Margulies,E.H., Chen,Y., Bernat,J.A., Ginsburg,D. *et al.* (2006) Genome-wide mapping of DNase hypersensitive sites using massively parallel signature sequencing (MPSS). *Genome Res.*, **16**, 123–131.