Prosperi *et al. BMC Bioinformatics* (2021) 22:445
https://doi.org/10.1186/s12859-021-04355-6

BMC Bioinformatics

SOFTWARE

Open Access

Fast and exact quantification of motif occurrences in biological sequences

Mattia Prosperi[1*], Simone Marini[1] and Christina Boucher[2]

*Correspondence: m.prosperi@ufl.edu
[1] Data Intelligence Systems Lab, Department of Epidemiology, College of Public Health and Health Professions and College of Medicine, University of Florida, Gainesville, FL, USA
Full list of author information is available at the end of the article

Abstract

Background: Identification of motifs and quantification of their occurrences are important for the study of genetic diseases, gene evolution, transcription sites, and other biological mechanisms. Exact formulae for estimating count distributions of motifs under Markovian assumptions have high computational complexity and are impractical to be used on large motif sets. Approximated formulae, e.g. based on compound Poisson, are faster, but reliable *p* value calculation remains challenging. Here, we introduce 'motif_prob', a fast implementation of an exact formula for motif count distribution through progressive approximation with arbitrary precision. Our implementation speeds up the exact calculation, usually impractical, making it feasible and posit to substitute currently employed heuristics.

Results: We implement motif_prob in both Perl and C++ languages, using an efficient error-bound iterative process for the exact formula, providing comparison with state-of-the-art tools (e.g. MoSDi) in terms of precision, run time benchmarks, along with a real-world use case on bacterial motif characterization. Our software is able to process a million of motifs (13–31 bases) over genome lengths of 5 million bases within the minute on a regular laptop, and the run times for both the Perl and C++ code are several orders of magnitude smaller (50–1000× faster) than MoSDi, even when using their fast compound Poisson approximation (60–120× faster). In the real-world use cases, we first show the consistency of motif_prob with MoSDi, and then how the p-value quantification is crucial for enrichment quantification when bacteria have different GC content, using motifs found in antimicrobial resistance genes. The software and the code sources are available under the MIT license at https://github.com/DataIntellSystLab/motif_prob.

Conclusions: The motif_prob software is a multi-platform and efficient open source solution for calculating exact frequency distributions of motifs. It can be integrated with motif discovery/characterization tools for quantifying enrichment and deviation from expected frequency ranges with exact *p* values, without loss in data processing efficiency.

Keywords: Bioinformatics, Motifs, Probability distribution, Markov model

Background

Motif discovery and characterization are important for the study of gene evolution, duplication, transcription sites, and protein identification [1], as well as of genetic diseases caused by unstable repeat expansion [2, 3].

Several tools have been developed for de novo motif discovery [4–6]—including discriminative regular expression motif elicitation (DREME), hypergeometric optimization of motif enrichment (HOMER), multiple expectation maximizations for motif elicitation (MEME), the memetic framework for motif discovery (MFMD), peak-motifs, prosampler, regulatory sequence analysis tools (RSAT), Trawler Web, and Weeder—either generic or specialized, e.g. for ChIP-seq data [7–15].

Assessing the statistical significance of motif enrichment is a fundamental and challenging step of motif discovery, and can severely hamper downstream analytics. Kiesel et al. [16] pointed out that p values *"Small enrichment factors can occur frequently in practice simply due to an imperfect background model that slightly underestimates the expected frequency of occurrence"*. In addition, p values are crucial not only in the discovery phases, but also in motif comparison and motif-motif similarity studies [17]. The classical definition of the motif enrichment problem (in terms of differences among motifs occurrences within background genome contents) has been proven to be NP-hard [18]. The p value calculation is not straightforward, and requires making assumptions on a background model of base frequencies and co-occurrence in order to derive a distribution of motif occurrences in reference genomes [19]. Several formulae—approximated and exact—and algorithms for estimating motif count distributions have been devised and implemented [20–28]. Exact formulae for estimating count distributions of motifs under Markovian assumptions have high computational complexity and are impractical to be used on large data sets. Approximated formulae, e.g. based on compound Poisson, are faster, but reliable p value calculation remains challenging [19, 25]. Thus, methods for p value estimation can be a bottleneck in large-scale projects. HOMER, Weeder and Peak-motifs do not report motif statistical significance, MEME uses an approximation approach (very conservative), later improved by DREME and the new simple, thorough, rapid, enriched motif elicitation (STREME) [10, 15], and MFMD uses information content score and complexity scores [29].

A software that provides a comprehensive occurrence and probability estimation is the bioinformatics toolkit for Motif Statistics and Discovery (MoSDi) by Marschall [30], written in Java, featuring models based on the approximated compound Poisson and nth level Markov order, as well as (quasi-)exact combinatorial formulae to reduce computational complexity (https://bitbucket.org/tobiasmarschall/mosdi). Another tool is motifcounter [31], an R-Bioconductor library implementing existing methods [27, 32], as well as an improvement on the compound Poisson model. One limitation of these programs is that calculation of occurrence distribution—even using the fast compound Poisson—becomes impractical with longer motifs (10+) and longer reference genomes (millions of bases), besides large motif datasets.

Prosperi et al. [28] provided an exact formula for counting the distribution of strings that do not overlap with themselves (i.e. *non-clumpable*), coupled with a mathematical demonstration of its validity, under both Bernoullian and Markovian assumptions. The calculation of the formula was exponential in the genome length by the length of the motif, but the authors demonstrated that it could be calculated efficiently within an arbitrary tolerance level.

This software article describes "motif_prob", a count distribution tool suitable for long motifs and long reference genomes, implementing the exact method by Prosperi et al. [28] with the efficient error-bound algorithm. In addition to the relevance of this software piece for large-scale processing, another motivation for our work is that the majority of probability distribution or p value calculators, even the most recent ones, use heuristics. To our knowledge, the formula by Prosperi et al. is still among the most efficient for *exact* calculation. The proposed motif_prob implementation thus makes exact quantification suitable with large scale projects, and posits to substitute currently employed heuristics. We compare motif_prob with other tools in terms of run time and precision, showing that its exact algorithm is several orders of magnitude faster even than the approximated methods, and finally we describe use cases for long motifs in bacteria.

Methods

Theoretical formulation

The exact formula by Prosperi et al. [28], for the calculation of the frequency distribution j of a string of length m within a text of length n ($m<n$) over alphabet k, under the Markovian model, is

$$P\left(j,m,n\right)=P(S)^{j}\sum_{z=1}^{|C_{n,m,j}|}\prod_{y=1}^{j+1}P\left(S_{0,d_{yz}}\right), \tag{1}$$

where $P(S)=P(a_1)\cdot P(a_2\mid a_1)\cdot \ldots \cdot P(a_{m-1}\mid a_m)$, $P(S_{0,n})=P(S_{0,n-1})-P(S)\cdot P(S_{0,n-m})$, $S_{0,n}=S_{0,n-1}\cdot k-P(S)\cdot k^m\cdot S_{0,n-m}$, $d_1 \ldots d_{j+1}$ are the lengths of the $j+1$ segments where the j strings divide the text of length n in exact configurations with $d_1+\cdots+d_{j+1}=n-mj$, and

$$\left|C_{n,m,j}\right|=\binom{n+j(1-m)}{n-mj}. \tag{2}$$

Formula (1) has a complexity of O(n^j), which becomes quickly intractable. However, by defining $R=P(S_{0,n+1})/P(S_{0,n})$ as a constant, Prosperi et al. show that for any positive (arbitrarily small) number ε, there is an index η_ε such that for every $\eta>\eta_\varepsilon$ then

$$P\left(S_{0,n+x}\right)\sim P\left(S_{0,n}\right)\cdot R^{x}. \tag{3}$$

By using this approximation, the summation of the original formula can be reduced to a single step, and calculations can be stopped when the ratio $P(S_{0,n})/P(S_{0,n-1})$ reaches a desired level of tolerance ε. Specifically, after plugging the iterative approximation (3) in (1), we obtain the final formula

$$P\left(j,m,n\right)\sim P(S)^{j}\cdot R^{n-mj-n_{\varepsilon}(j+1)}\cdot P\left(S_{(0,n_{\varepsilon})}\right)^{j+1}\cdot\binom{n+j(1-m)}{n-mj}. \tag{4}$$

We note that $P(j,\ m,\ n)$ is the same irrespective of the position of the nucleotides in a query string, e.g. AACCC and CCCAA have the same probability. This property permits to extrapolate a probability for clumpable strings by permutation, e.g. ACCA into CCAA, although the value is not guaranteed given possible overlap. Another way is to replace the first or the last character with another one that has the same frequency. All details on the derivation of the exact formula and the proof for its progressive approximation, along with comparison against other state-of-art algorithms, can be found in the original work by Prosperi et al. [28].

Implementation

Two different implementations are produced: one in Perl and another in C++. Both programs take the same input and parameters, namely: (1) a query string or multiple strings to be analyzed; (2) the length of the reference genome; and (3) the nucleotide frequencies of the genome. In alternative to the genome length and nucleotide frequencies, a FASTA file containing the genome string can be passed as input to the program. The output file reports—for each motif—the count distribution and other summary information including a flag for clumped strings, string probability, and statistics on the precision and tolerance levels.

Since the computational complexity of the formula is exponential, motif occurrences are calculated at increasing counts until the occurrence probability becomes lower than given a tolerance level ε, or the upper limit of counts j is reached. We also control estimates at each iteration in order to avoid issues with floating point operations when frequency/length ratios diverge, and to handle relatively ill-posed configurations. Given the motif m and genome g lengths, one can set a tolerance level ε such that $P(0, m, n) > (1-\varepsilon)$, and in general each case where $(1-P(S))^{(\mathrm{m}-\mathrm{m}+1)} > (1-\varepsilon)$. This is equal to $(n-m+1)\cdot\log(1-P(S)) > \log(1-\varepsilon)$, which implies $n > m - 1 + \log(1-\varepsilon)/\log(1-P(S))$. In the source code, we have set ε to 10^{-7} and j to 500. Further, we implement the calculation of the expected number of strings and the motif's (stationary) occurrence probability at any text position, according to Robin et al. [33].

Figure 1 provides a flowchart of the data processing pipeline, showing the required input specifications, the method's internal parameters, and the output fields.

The source code, documentation, sample datasets, and executable files are available under the MIT license at https://github.com/DataIntellSystLab/motif_prob.

Results

An example of the occurrence distribution for motif query sequences of length 6, calculated on a randomly generated genome of 20,000 bases, varying the nucleotide frequencies, is illustrated in Fig. 2. The difference between the equiprobable base and the more general case is evident and demonstrates how the background distribution affects the p value calculation (see real-world use case after the benchmarks).

Table 1 shows run time benchmarks on different motif length and motif set size configurations, executed on a laptop machine with Intel(R) Core(TM) i9-10885H CPU @ 2.4 GHz, 32 GB RAM. Both the Perl and the C++ programs exhibit run times several orders of magnitude smaller than MoSDi, even when the latter is executed with the fast compound Poisson approximation. We set a maximum processing time of 30 min for datasets up to 400,000 motifs, and MoSDi can process them only with smaller values of k and the approximated model, while the exact model is not feasible for most of datasets.

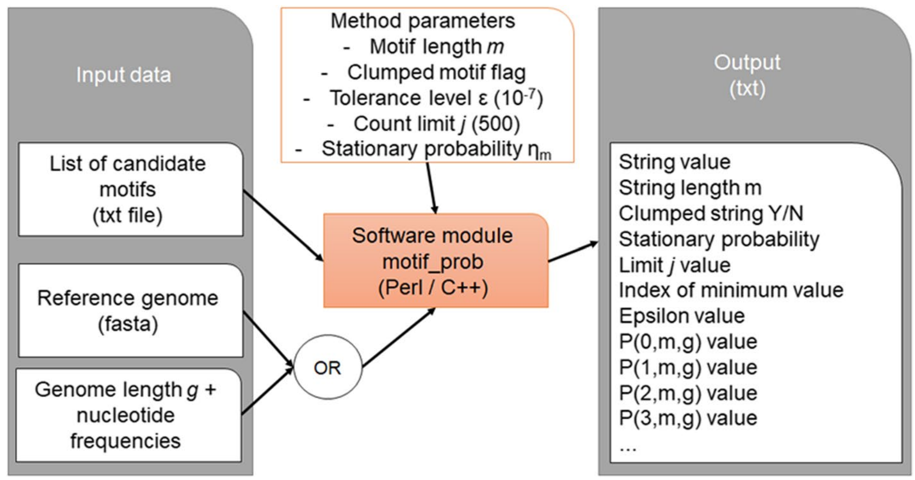

Fig. 1 Flowchart of the data processing pipeline for motif_prob, with input/output specifications and program parameters

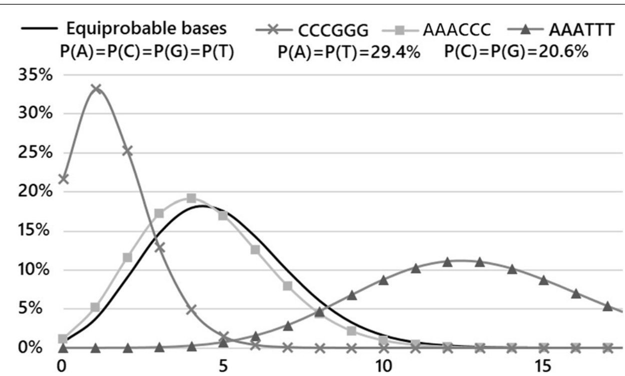

Fig. 2 Application output for motif sequences of length 6 over a genome of length 20,000, and different nucleotide frequencies

The C++ implementation is the fastest, and the expected run time increase due to higher motif lengths is well compensated by the implementation setup.

In terms of precision, we compare the exact probability values yielded by our program with both the compound Poisson and the exact estimates of MoSDi (allowing it to switch automatically to standard/doubling algorithms to improve run time). As previously described, usually the largest errors appears near the probability mass points [28]. For all motif lengths combinations of 4 bases, over a 10,000 bases reference genome, on average the peak probability values of MoSDi and motif_prob exact differ by two orders of magnitude, e.g. if the peak probability is in the range of 10^{-2} then the observed absolute difference is 10^{-4}. The difference with the compound Poisson approximation is larger, on average double than the exact, but the relative ratio it is still one-two orders of magnitude smaller than the actual values. The difference becomes smaller as the sequence lengths increase.

We further test the concordance among MoSDi and motif_prob using a real motif dataset, the library of DNA-binding site matrices for *Escherichia coli* (https://arep.med.

Table 1 Run time (mm:ss) of the Perl and C++ programs compared to MoSDi (exact and approximated using compound Poisson) for calculating the occurrence distribution for *s* motif query sequences of length *m* (13–31) over a reference genome of 5 million bases

No. of motifs *s*	Motif length *m*	C++	Perl	MoSDi exact (10/500)	MoSDi approx (10/500)
10	13	00:00	00:00	00:47/17:50	00:00/00:00
20,000	13	00:01	00:07	xx:xx/xx:xx	01:16/02:24
50,000	13	00:03	00:18	xx:xx/xx:xx	03:14/06:24
200,000	13	00:15	01:12	xx:xx/xx:xx	13:34/27:40
400,000	13	00:35	02:20	xx:xx/xx:xx	28:05/xx:xx
1,000,000	13	00:85	05:45	xx:xx/xx:xx	xx:xx/xx:xx
10	31	00:00	00:00	01:47/xx:xx	00:00/00:01
20,000	31	00:02	00:07	xx:xx/xx:xx	15:34/16:41
50,000	31	00:04	00:17	xx:xx/xx:xx	xx:xx/xx:xx
200,000	31	00:16	01:09	xx:xx/xx:xx	xx:xx/xx:xx
400,000	31	00:29	02:26	xx:xx/xx:xx	xx:xx/xx:xx
1,000,000	31	00:32	05:18	xx:xx/xx:xx	xx:xx/xx:xx

Runs lasting over 30:00 were stopped

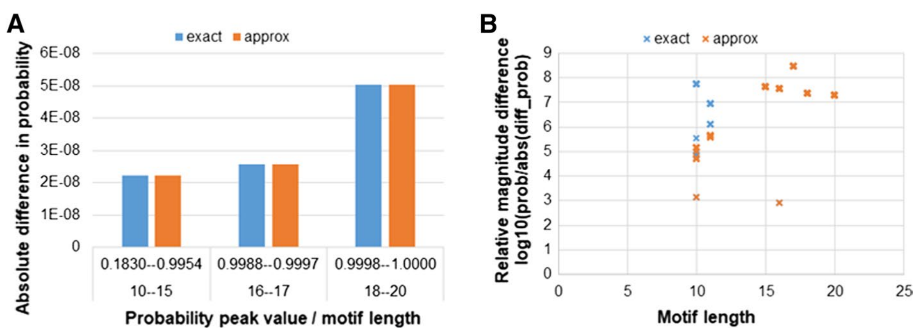

Fig. 3 Comparison between motif_prob and MoSDi (exact and approximated with compound Poisson) in terms of concordance of probability estimates. Panel **A** shows the absolute difference in peak probability values, stratified by motif length and probability mass value, while panel **B** shows the relative magnitude difference by motif length

harvard.edu/ecoli_matrices/), which contains 802 motifs from 67 housekeeping genes for a median motif length of 26 (interquartile range, IQR 20–29). We consider motifs length within 20 bases to be able to estimate non-near-zero probabilities on the genome length of *Escherichia coli*. The final set includes 230 motifs with a median length 16 (IQR 15–18). The median (IQR) difference between MoSDi and motif_prob exact overall is $2.6 \cdot 10^{-8}$ ($2.2 \cdot 10^{-8}$–$5.0 \cdot 10^{-8}$), while for all probabilities where the center of mass is not zero (median 0.18), it is $3.8 \cdot 10^{-8}$ ($3.3 \cdot 10^{-9}$–$2.6 \cdot 10^{-7}$). Once again, the differences with the approximated estimation are larger but of the same level of magnitude. Figure 3 illustrates the absolute difference in probability between motif_prob and MoSDi (exact/compound Poisson) as well as the relative magnitude difference, expressed as the $\log 10(\mathrm{Prob}_{motif\_prob}/\mathrm{abs}(\mathrm{Prob}_{motif\_prob} - \mathrm{Prob}_{MoSDi}))$, which well highlights how the difference between the two exact methods (and the compound Poisson too, although larger) is negligible with respect to the actual probability estimates.

As a final use case, we investigate the distribution of frequencies of antimicrobial resistance gene signatures found in bacteria under different GC content. Drug resistance mechanisms in bacteria involve acquisition of genes, often via mobile genetic elements, and in some cases changes within core housekeeping genes. A number of algorithms use k-mers, i.e. motifs of fixed k length, to classify antimicrobial resistance [34], as they can be handled efficiently through ad hoc data structures suitable to process high-throughput data. But assessing the importance of a k-mer with respect to their frequency in drug resistance genes is not straightforward; one issue is that bacteria and genes can have very different GC content [35]. When the GC content varies, the probability distributions of motif occurrence can change over a broad range (given also the underlying, individual A, C, G, and T content), and thus the p values of over- or under-representation. To show how the quantification can have large variance, we analyze k-mers from antimicrobial resistance genes collected in the MEGARes 2.0 database [36]. MEGARes contains 7868 genes, with an average gene length of 1030.29 nucleotide bases, 57 different antibiotic resistance classes, and 220 distinct resistance mechanisms.

From MEGARes, we select all the 3911 genes conferring resistance to beta-lactamase; we then identify all 13-mers, for a total of 453,308 motifs (50% GC content). In Table 2, we show how the count probability distribution of the 13-mers in MEGARes' beta-lactamase genes changes among bacterial species present in the human microbiome of respiratory tract [37], where we select uniformly 18 species on the basis of their GC content. The median probability of finding the aforementioned 13-mers at least once varies between 93 and 99%, and even species with a similar GC content can show different medians and interquartile ranges, such as *Stomatobaculum longum* (55% GC content, median $p=97\%$) and *Kluyvera intermedia* (52% GC content, median $p=93\%$).

Table 2 Median (interquartile range, IQR) probability of finding at least once 13-mer motifs (top-frequent among beta-lactamase resistance genes) in the MEGARes database over different bacterial species characterized by heterogeneous GC content

Species	Genome length	GC content	Median (IQR) probability
Nocardioides Salarius	4,429,322	0.73	0.98 (0.94–1)
Enhydrobacter aerosaccus	6,767,089	0.65	0.93 (0.87–0.98)
Paraburkholderia ginsengisol	6,541,884	0.64	0.93 (0.87–0.97)
Neisseria shayeganii	2,419,744	0.58	0.97 (0.95–0.98)
Stomatobaculum longum	2,308,581	0.55	0.97 (0.96–0.98)
Kluyvera intermedia	4,938,529	0.52	0.93 (0.92–0.94)
Buttiauxella noackiae	4,766,673	0.49	0.93 (0.93–0.93)
Megasphaera micronuciformis	1,765,374	0.45	0.97 (0.97–0.98)
Oribacterium sinus	2,727,518	0.43	0.96 (0.95–0.98)
Prevotella jejuni	3,913,006	0.42	0.95 (0.92–0.97)
Prevotella melaninogenica	3,168,282	0.4	0.96 (0.94–0.98)
Streptococcus pseudopneumoniae	2,195,458	0.4	0.97 (0.95–0.99)
Veillonella rogosae	2,187,106	0.39	0.97 (0.95–0.99)
Lachnoanaerobaculum orale	2,799,073	0.38	0.97 (0.94–0.99)
Catonella morbi	3,477,404	0.38	0.96 (0.93–0.99)
Staphylococcus argenteus	2,753,898	0.32	0.98 (0.95–0.99)
Leptotrichia wadei	2,337,418	0.29	0.98 (0.96–1)
Fusobacterium nucleatum	2,455,060	0.26	0.99 (0.97–1)

This variability is due to: the individual nucleotide content, which can differ even when the GC content is the same, and it directly affects the distribution (see also Fig. 2); the genome length; and the nucleotide content of the query motifs.

Conclusion

The motif_prob software is a multi-platform, open source, efficient solution for calculating exact frequency distributions of (long) motif occurrences in reference genomes using high-throughput data. We showed how our code estimates are consistent with other, slower, exact calculations, and how the run times of our code (both Perl and C++) are competitive even with the non-exact compound Poisson approximation. Specifically, motif_prob is 50–1000× faster than MoSDi exact and 60–120× faster than MoSDi compound Poisson.

The current implementation is limited to non-clumpable strings, although it extrapolates a probability for clumpable strings by permutation. As future development of our work we foresee to develop an exact formula for clumpable strings and to extend the approach to generalize over motifs that can include nucleotide changes, insertions or deletions.

In conclusion, our tool can be effectively used in conjunction with motif discovery suites that process high-throughput data, allowing them to compute exact count distributions and associated p values without loss of run time performance, instead of relying on to approximations.

Availability and requirements

Project name: motif_prob
Project home page: https://github.com/DataIntellSystLab/motif_prob
Operating system(s): Multi-platform (UNIX/Linux/Mac, Windows)
Programming language: Perl, C++
Other requirements: None
License: MIT
Any restrictions to use by non-academics: Permissible under the terms of the MIT license.

Abbreviations

DREME: Discriminative regular expression motif elicitation; HOMER: Hypergeometric optimization of motif enrichment; MEME: Multiple expectation maximizations for motif elicitation; MFMD: Memetic framework for motif discovery; RSAT: Regulatory sequence analysis tools; STREME: Simple, thorough, rapid, enriched motif elicitation; MoSDi: Motif statistics and discovery; mm:ss: Minutes:seconds; IQR: Interquartile range.

Acknowledgements

We thank Luciano Prosperi, *MSc*, and Roberto Di Castro, *MSc*, for aiding with the implementation and code maintenance.

Authors' contributions

MP conceived idea, wrote paper and code; SM wrote code and set up web repository; CB reviewed algorithm, wrote the paper. All authors have read and approved the final manuscript.

Funding

This work has been supported by the National Institutes of Health (NIH)—National Institute of Allergy and Infectious Diseases (NIAID) Grants No. R01AI141810 and R01AI145552, and by the National Science Foundation (NSF) Grant No. 2013998. The funding body did not have roles in the design of the study and collection, analysis, interpretation of data, and in writing the manuscript.

Availability of data and materials
The datasets generated and/or analysed during the current study are available in the motif_prob GitHub repository (https://github.com/DataIntellSystLab/motif_prob), the MEGARes database (https://megares.meglab.org/) and the website for DNA-binding site matrices for Escherichia coli (https://arep.med.harvard.edu/ecoli_matrices/).

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
Dr. Christina Boucher is Associate Editor of BMC Bioinformatics. All other authors declare that they have no competing interests.

Author details
[1]Data Intelligence Systems Lab, Department of Epidemiology, College of Public Health and Health Professions and College of Medicine, University of Florida, Gainesville, FL, USA. [2]Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, USA.

Received: 4 April 2021 Accepted: 6 September 2021
Published online: 18 September 2021

References

1. Luu P-L, Schöler HR, Araúzo-Bravo MJ. Disclosing the crosstalk among DNA methylation, transcription factors, and histone marks in human pluripotent cells through discovery of DNA methylation motifs. Genome Res. 2013;23(12):2013–29.
2. Gatchel JR, Zoghbi HY. Diseases of unstable repeat expansion: mechanisms and common principles. Nat Rev Genet. 2005;6:743–55.
3. Luu PL, Schöler HR, Araúzo-Bravo MJ. Disclosing the crosstalk among DNA methylation, transcription factors, and histone marks in human pluripotent cells through discovery of DNA methylation motifs. Genome Res. 2013;23:2013–29.
4. Tompa M, Li N, Bailey TL, Church GM, De Moor B, Eskin E, et al. Assessing computational tools for the discovery of transcription factor binding sites. Nat Biotechnol. 2005;23(1):137–44.
5. Lee NK, Li X, Wang D. A comprehensive survey on genetic algorithms for DNA motif prediction. Inf Sci. 2018;1(466):25–43.
6. Hashim FA, Mabrouk MS, Al-Atabany W. Review of different sequence motif finding algorithms. Avicenna J Med Biotechnol. 2019;11(2):130–48.
7. Pavesi G, Mereghetti P, Mauri G, Pesole G. Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. Nucleic Acids Res. 2004;32(Web Server issue):W199–203.
8. Ettwiller L, Paten B, Ramialison M, Birney E, Wittbrodt J. Trawler: de novo regulatory motif discovery pipeline for chromatin immunoprecipitation. Nat Methods. 2007;4(7):563–5.
9. Bailey TL, Boden M, Buske FA, Frith M, Grant CE, Clementi L, et al. MEME SUITE: tools for motif discovery and searching. Nucleic Acids Res. 2009;37(Web Server issue):W202–8.
10. Bailey TL. DREME: motif discovery in transcription factor ChIP-seq data. Bioinformatics. 2011;27(12):1653–9.
11. Thomas-Chollier M, Herrmann C, Defrance M, Sand O, Thieffry D, van Helden J. RSAT peak-motifs: motif analysis in full-size ChIP-seq datasets. Nucleic Acids Res. 2012;40(4):e31–e31.
12. Dang LT, Tondl M, Chiu MHH, Revote J, Paten B, Tano V, et al. TrawlerWeb: an online de novo motif discovery tool for next-generation sequencing datasets. BMC Genomics. 2018;19(1):238.
13. Caldonazzo Garbelini JM, Kashiwabara AY, Sanches DS. Sequence motif finder using memetic algorithm. BMC Bioinform. 2018;19(1):4.
14. Li Y, Ni P, Zhang S, Li G, Su Z. ProSampler: an ultrafast and accurate motif finder in large ChIP-seq datasets for combinatory motif discovery. Berger B, editor. Bioinformatics. 2019;35(22):4632–9.
15. Bailey TL. STREME: accurate and versatile sequence motif discovery. bioRxiv. 2020;2020.11.23.394619.
16. Kiesel A, Roth C, Ge W, Wess M, Meier M, Söding J. The BaMM web server for de-novo motif discovery and regulatory sequence analysis. Nucleic Acids Res. 2018;46(W1):W215–20.
17. Gupta S, Stamatoyannopoulos JA, Bailey TL, Noble WS. Quantifying similarity between motifs. Genome Biol. 2007;8(2):R24.
18. Finding similar regions in many strings|Proceedings of the thirty-first annual ACM symposium on Theory of Computing [Internet]. [cited 2021 May 28]. https://doi.org/10.1145/301250.301376.
19. Zhang J, Jiang B, Li M, Tromp J, Zhang X, Zhang MQ. Computing exact *p* values for DNA motifs. Bioinformatics. 2007;23(5):531–7.
20. Gentleman JF, Mullin RC. The distribution of the frequency of occurrence of nucleotide subsequences, based on their overlap capability. Biometrics. 1989;45(1):35–52.
21. Régnier M. A unified approach to word occurrence probabilities. Discrete Appl Math. 2000;104(1):259–80.
22. Nicodème P, Salvy B, Flajolet P. Motif statistics. Theor Comput Sci. 2002;287(2):593–617.

23. Robin S, Daudin J-J, Richard H, Sagot M-F, Schbath S. Occurrence probability of structured motifs in random sequences. J Comput Biol J Comput Mol Cell Biol. 2002;9(6):761–73.
24. Rivals E, Rahmann S. Combinatorics of periods in strings. J Comb Theory Ser A. 2003;104(1):95–113.
25. Bejerano G, Friedman N, Tishby N. Efficient exact p-value computation for small sample, sparse, and surprising categorical data. J Comput Biol J Comput Mol Cell Biol. 2004;11(5):867–86.
26. Lladser ME, Betterton MD, Knight R. Multiple pattern matching: a Markov chain approach. J Math Biol. 2008;56(1):51–92.
27. Marschall T, Rahmann S. Efficient exact motif discovery. Bioinformatics. 2009;25(12):i356–64.
28. Prosperi MCF, Prosperi L, Gray RR, Salemi M. On counting the frequency distribution of string motifs in molecular sequences. Int J Biomath. 2012;5:1250055.
29. Fogel GB, Weekes DG, Varga G, Dow ER, Harlow HB, Onyia JE, et al. Discovery of sequence motifs related to coexpression of genes using evolutionary computation. Nucleic Acids Res. 2004;32(13):3826–35.
30. Marschall T, Rahmann S. Speeding up exact motif discovery by bounding the expected clump size. In: Moulton V, Singh M, editors. Algorithms in bioinformatics. Lecture notes in computer science. Berlin: Springer; 2010. p. 337–49.
31. Kopp W. motifcounter: R package for analysing TFBSs in DNA sequences [Internet]. Bioconductor version: Release (3.12); 2021 [cited 2021 Mar 17]. https://bioconductor.org/packages/motifcounter/.
32. Pape UJ, Rahmann S, Sun F, Vingron M. Compound poisson approximation of the number of occurrences of a position frequency matrix (PFM) on both strands. J Comput Biol J Comput Mol Cell Biol. 2008;15(6):547–64.
33. DNA, Words and Models: Statistics of Exceptional Words by S. Robin, F. Rodolphe, S. Schbath | 9780521847292 | Hardcover | Barnes & Noble® [Internet]. [cited 2021 Mar 17]. https://www.barnesandnoble.com/w/dna-words-and-models-s-robin/1110953123.
34. Clausen PTLC, Zankari E, Aarestrup FM, Lund O. Benchmarking of methods for identification of antimicrobial resistance genes in bacterial whole genome data. J Antimicrob Chemother. 2016;71:2484–8.
35. Hildebrand F, Meyer A, Eyre-Walker A. Evidence of selection upon genomic GC-content in bacteria. PLoS Genet. 2010;6:e1001107.
36. Doster E, Lakin SM, Dean CJ, Wolfe C, Young JG, Boucher C, et al. MEGARes 2.0: a database for classification of antimicrobial drug, biocide and metal resistance determinants in metagenomic sequence data. Nucleic Acids Res. 2020;48:D561–9.
37. Ibironke O, McGuinness LR, Lu S-E, Wang Y, Hussain S, Weisel CP, et al. Species-level evaluation of the human respiratory microbiome. GigaScience. 2020;9:giaa038. https://doi.org/10.1093/gigascience/giaa038.

Publisher's Note