Published online 4 March 2019

Nucleic Acids Research, 2019, Vol. 47, No. 7 ***3321–3332***
doi: 10.1093/nar/gkz102

Automated, customizable and efficient identification of 3D base pair modules with BayesPairing

Roman Sarrazin-Gendron[1], Vladimir Reinharz[2], Carlos G. Oliver[1], Nicolas Moitessier[3] and Jérôme Waldispühl[1,*]

[1]School of Computer Science, McGill University, Montreal, QC H3A 0E9, Canada, [2]Center for Soft and Living Matter, Institute for Basic Science, Ulsan 44919, Republic of Korea and [3]Department of Chemistry, McGill University, Montreal, QC H3A 0B8, Canada

Received October 03, 2018; Revised February 06, 2019; Editorial Decision February 07, 2019; Accepted February 28, 2019

ABSTRACT

RNA structures possess multiple levels of structural organization. A secondary structure, made of Watson–Crick helices connected by loops, forms a scaffold for the tertiary structure. The 3D structures adopted by these loops are therefore critical determinants shaping the global 3D architecture. Earlier studies showed that these local 3D structures can be described as conserved sets of ordered non-Watson–Crick base pairs called RNA structural modules. Unfortunately, the computational efficiency and scope of the current 3D module identification methods are too limited yet to benefit from all the knowledge accumulated in the module databases. We present `BayesPairing`, an automated, efficient and customizable tool for (i) building Bayesian networks representing RNA 3D modules and (ii) rapid identification of 3D modules in sequences. `BayesPairing` uses a flexible definition of RNA 3D modules that allows us to consider complex architectures such as multi-branched loops and features multiple algorithmic improvements. We benchmarked our methods using cross-validation techniques on 3409 RNA chains and show that `BayesPairing` achieves up to ∼70% identification accuracy on module positions and base pair interactions. `BayesPairing` can handle a broader range of motifs (versatility) and offers considerable running time improvements (efficiency), opening the door to a broad range of large-scale applications.

INTRODUCTION

RNA structures are hierarchically organized (1). Initially, the molecule folds by forming Watson–Crick and Wobble base pairs stacking onto each other to create stems. These stems are connected by loops, themselves stabilized by intricate networks of non-canonical base pair interactions and structural modules. Those structural modules can be defined as recurrent networks of base pairs and stacking interactions. Then, the specific 3D structures of these connecting loops help shaping the tertiary structure of the RNA molecule. Eventually, other interactions connecting distant secondary structure elements are also formed to stabilize the full complex.

The identification of local 3D modules is therefore an important step in the RNA 3D structure prediction pipeline. However, in contrast to the well-established secondary structure prediction methods (2,3), the technology to address this challenge is still in its infancy.

A first step toward accurate prediction of 3D modules is the identification of conserved 3D structures occurring in natural sequences. Several algorithms have been developed to retrieve these local 3D modules (4–7) and characterize their function (8), and databases such as the `RNA 3D Motif Atlas` (9), `RNA FRABASE` (10), `RNA Bricks` (11) and `CaRNAval` (12) have already started assembling this information. Yet, the scope of these systems varies and no widely accepted solution has yet emerged.

Based on this information, several groups developed computational tools to score and retrieve 3D modules in RNA sequences. To date, `RMDetect` is among the most promising approaches (13). The latter uses Bayesian networks to represent base pairing tendencies learned from sequence alignments of 3D modules, and uses this knowledge to identify candidate modules within annotated sequences. Another recent option is `JAR3D` (14), which has refined the methodology for scoring multiple 3D modules on new sequence variants. This technology has been used within the `metaRNAmodules` pipeline for scanning complete RNA databases and retrieving modules showing evidence of evolutionary conservation (15,16).

However, the aforementioned methods still have some limitations. In particular, despite the excellent performances reported by `RMDetect`, the approach suffers of high com-

*To whom correspondence should be addressed. Tel: +1 514 398 5018; Email: jeromew@cs.mcgill.ca

© The Author(s) 2019. Published by Oxford University Press on behalf of Nucleic Acids Research.
This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/4.0/), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited. For commercial re-use, please contact journals.permissions@oup.com

putational costs and a minimal structural diversity in the modules considered due to its base pair probabilities scanning method. Similarly, JAR3D has not been designed to maximize its scanning capabilities and is thus best used for scoring applications.

In this paper we present BayesPairing, an efficient and customizable tool for (i) building Bayesian networks representing RNA 3D modules and (ii) rapid identification of 3D modules enabling genome-wide applications. BayesPairing expands upon the RMDetect methodology and features multiple algorithmic improvements that result in considerable speed improvements, while maintaining similar or better accuracy. Moreover, BayesPairing uses a flexible definition of RNA 3D modules allowing us to consider complex modules such as multi-branched loops and flexible modules allowing us to capture 3D features not supported by previous methods. Our tool is therefore highly customizable, interfaceable with any module databases and is already available at http://bayespairing.cs.mcgill.ca for scanning for new modules.

We tested BayesPairing on two different 3D module datasets focusing on two distinct classes of modules, and evaluated our methods on 3409 chains of experimentally determined 3D structures. The first module dataset is composed of hairpins, internal loops and multi-branched loops retrieved with Rna3Dmotif (6). By contrast, the second dataset includes complex modules featuring variations in the base pair type and distance between nucleotides among different examples of the same module (9). Overall, we apply our techniques to hundreds of modules from two datasets. Our results show a base pair identification accuracy of ∼35–75%, with ∼30–60% of fully predicted modules (i.e. all canonical and non-canonical base pairs were correctly identified).

MATERIALS AND METHODS

Our approach is implemented in a Python package called BayesPairing and is freely available as a web-server and a downloadable git repository at http://bayespairing.cs.mcgill.ca. An installation guide can be found in the Supplementary Data, as well as on the web-server. The required input is a sequence to parse and a dataset of modules to identify (a basic, all-purpose dataset is provided). The addition of a new module requires a description of 3D interactions in the .desc format (6) and a .fasta file with the sequences, which can come directly from the graph examples, from prior knowledge, or from an Rfam alignment (17).

Modeling RNA structure as a graph

An RNA module can be conveniently represented as a directed graph with labeled edges, where each node is a base and edges represent pairing interactions from 5′ to 3′. The graph representation of an RNA structure contains 13 types of interactions. Backbone interactions, or phosphodiester bonds, determine the sequence organization in the 5′-3′ order of nucleotides. However, most backbone interactions are not included in the edge set when building the Bayesian network due to the generally weak dependence relationship among partners of a backbone interaction outside the sequence signal, which is preserved and encoded in the sequence search rather than in the Bayesian Network. In this article, we define a strand of nucleotides that is part of a module as a component. For example, a hairpin loop has a single component that we can scan the sequence for, whereas a three-way junction has three. The remaining interaction types are base pairs classified with respect to their geometry. Following the Leontis-Westhof classification (18), there are 12 classes of base pairs. They are defined following their orientation cis (c) (resp. trans (t)) and the name of the nucleotides' interacting faces. Three symbols are also used to describe the interactions. They are the Watson-Crick (W) • (resp. ◯), Hoogsteen (H) ■ (resp. □) or Sugar-Edge (S) ▶ (resp. ▷) faces. Thus, each base pair is annotated using a string from the set: $\{c,t\}\times\{W,S,H\}^2$ or by combining previous symbols. When it is the same edge interacting on both nucleotides, only one symbol is used. We show in Figure 1 how a sequence in a specific module configuration can be represented as a graph. This module contains five base pairs, two canonical GC, one canonical AU, all cis Watson–Crick/Watson–Crick, one GA trans Sugar/Hoogsteen and one CA cis Watson–Crick/Sugar. There are also 10 backbone interactions.

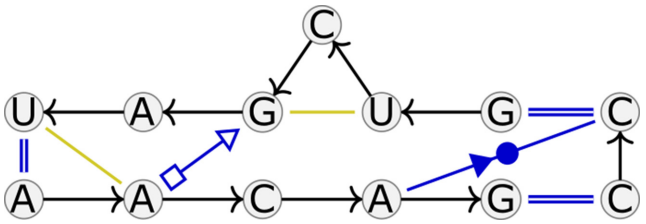

Figure 1. The structure of an RNA 3D module, specifically a three helix multi-branched loop from Rna3Dmotif. This module has three cWW (a double line), one tSH, one cSW (in Leontis-Westhof notation) and ten directed backbone interactions (a single line with an arrow). There are also two stacking interactions (a single line with no arrow).

It should be noted that such a graph encodes a recurrent network of interactions. Using a definition based on biological significance, an RNA 3D module can be defined as a set of recurrent networks of base pairs and stacking interactions that lead to the same local 3D structure. Recurrence here can be defined in two ways. In RMDetect, it is observed from the presence of a conserved sequence motif associated with local 3D structure in a multiple sequence alignment. In Rna3Dmotif and CaRNAval, recurrence is observed by finding the same set of interactions in crystal structures of distinct RNA molecules with graph isomorphism methods. As a consequence, there can be several distinct graphs based on FR3D (5) annotations that are representations of the same module, due to biological variability but also eventual imprecision of annotations. This is why our methods focus on detecting variations of the model graph as potential occurrences of the module.

Learning a Bayesian network from structural module data

Module graphs to Bayes Nets. The aim of this work is to model the sequence variability and dependence relationships for any given 3D module. More generally, we wish to

model the distribution $\mathbb{P}(x_1, x_2, ..., x_N \mid b)$, where x_i s are the N nucleotides in the module and b is a given module. To model the dependence relationships among the partners of a base pair, there is a natural transformation from the graph of a 3D module to its representation as a Bayesian network. In such network, a node represents a nucleotide, and an edge represents a base pair interaction (canonical, non-canonical, or stacking interaction). Importantly, Bayesian networks require directed edges and the absence of cycles. To maintain this condition, and because the backbone is oriented, base pairs are modeled as directed edges from the 5′ end to the 3′ end of the module.

The probability distribution of each node is learned from data and has a cardinality of 4. The $\mathbb{P}(node, parent)$ Bayesian table has a size of $4^{|\pi_n+1|}$, where $|\pi_n|$ is the number of parents (or in degree) of the node n. The Bayesian networks were implemented with `pgmpy`, a Python library for probabilistic models.

Learning node and edge parameters. By definition, each sequence belonging to a module shares the same, or a closely similar set of interactions, and therefore maps to the same Bayesian network. The node values (nucleotide probabilities) are learned from the sequences with respect to nodes they are conditioned on. For a node with no parent, the known sequences are used to generate the statistics, which is identical to a position-weight matrix. For a node with a parent, the statistic is $\mathbb{P}$(nucleotide | parent). In that case, for each node, for each combination of parent–nucleotide, the known sequences are parsed until the conditional probability distribution (CPD) is established. To correct for potentially missing data, artificial sequences are created and injected into the dataset. They are built such that a minimal probability of 0.1 is added for each possible combination, after which we renormalize the probabilities to maintain a sum of 1.

Sequence probability. The probability of a sequence over a module (i.e. an assignment of node values) is derived from the properties of the Bayesian network. Given a sequence, we first compute a probabilistic score as $\mathbb{P}$(sequence | module). It is defined as the product of the probability of each node taking the value of some position in the input sequence given its parents, obtained by inference. Second, we chose as background probabilities the equiprobability of any nucleotide at each position. We then take the log of this value, a score output observed to be between -40 and 50.

Let N be the number of nucleotides in the module, and m be the nucleotide of the scored sequence corresponding to the node n. π_n is the set of parents of node n, and Q_n the actual nucleotide associated with each parent in the sequence. The complete equation for the probability, from the joint distribution of the Bayes net, is defined as follows:

$$\mathbb{P}(\text{sequence} \mid \text{module}) = \prod_{n=1}^{N} \mathbb{P}(n = m \mid \pi_n = Q_n). \quad (1)$$

The probability score is then divided by the background probabilities $\left(\frac{1}{4}\right)^N$ to normalize by size.

Regular expressions as a sequence mining tool

The secondary structure of some RNA strand is more conserved than its sequence, so the base pair signal is a safer information to look for when parsing for modules. The state-of-the-art tool, `RMDetect`, scans the secondary structure landscape (the base pair probabilities predicted by `RNAfold` (2). Indeed, this is especially true for hairpins, which have an extremely well-defined secondary structure signature, a sequence of unpaired bases within a base pair. However, this comes at the price of facing serious roadblocks when attempting to predict modules that are not always consecutive in sequence, because the different components of the module can occur in any order and, due to the lack of specificity of the secondary structure signal, there will often be up to thousands of possible insertion sites to consider, even in a sequence of 200 nucleotides. This would make it practically impossible to predict k-way junctions, which is a downside of `RMDetect`.

To tackle this issue, we use a new scanning method based on regular expressions, a string searching method based on pattern matching. Regular expressions are efficient and allow our approach to focus on the sequence rather than the secondary structure. The extended alphabet allows us to distinguish between sites with similar secondary structures, improving accuracy for complex modules like multi-branched loops. If a sensitivity comparable to the state-of-the-art methods can be obtained with regular expressions, then the overall search will be significantly faster, and this method will represent an improvement.

In order to achieve such a high level of sensitivity, many different regular expressions are used to scan the sequence for the different forms the module can adopt. For instance, the components of an internal loop can be found in two different 5′-3′ orders, thus we use two distinct regular expressions to parse for both possibilities.

From Bayes net to a fuzzy regular expression. Since the number of potential sequences for a module is exponential in the length of the module, we approximate the distribution of Bayes net sequences with Gibbs sampling, a method that performs random walks on the sequence space. We generate 25 000–100 000 sequences from the probability distribution of the Bayesian network, $\mathbb{P}$(sequence | module). The sampling process can take several minutes for large modules but only has to be performed once to build a re-usable model.

The set of maximal probability sequences from this sample is used to filter the sequences to build the regular expression. All sequences with a probability below 5% of the highest value are discarded. To modify the sensitivity of the tool, this value can be modified through a user parameter.

For each position, the allowed nucleotides are determined as follows. For each base, we collect all sampled sequences having that nucleotide at the current position, and take the sum of their probabilities. We then build sequence statistics by selecting as possibilities all nucleotides that have a value of at least 25% of the maximal value for that position, and computing their weighted average. This parameter can be adjusted by the user based on prior knowledge, but a higher threshold becomes too permissive and a lower too restrictive (data not shown).

When the nucleotides of a module are not consecutive, distance between different strands must be established. Two strands are considered separate when the distance between them is more than 5 bases. For consecutive strands, most of the search is focused on a distance range of at most 150% of the maximum observed in the module alignment, although longer matches are considered up to the full length of the input sequence. We add this limit to the distance because our approach, like many other structure prediction tools, relies on a structure prediction on the input sequence and, given that `RNAfold` loses much of its accuracy beyond lengths of 200 nucleotides, `BayesPairing` could not establish with high confidence that the secondary structure context allows for the insertion of such module. Thus, a search for perfect or near-perfect sequence matches at long distances is executed, and such modules will be detected by our tool, but more subtle module site candidates cannot be called with high confidence when separated by more than 200 nucleotides. The minimum distance is set to 4, one nucleotide longer than the minimum length of a hairpin, the smallest structure that can separate two strands. For strands between which the distance varies at most by 1 through the training data, we assume that there is a biological meaning to this value, so the range is fixed to the distances observed in the data. In general, regular expressions are known to prefer long matches over variable distance. This bias must be compensated by the use of not one, but several regular expressions, allowing increasing distances between the components for each. The level of coverage of the distance between components, which will be correlated with sensitivity, can be determined by a user-defined parameter.

Because we are presenting a framework to assign a probability score to candidates, it is important to adapt the regular expression to gather as many promising candidates as possible to maximize the chances of finding at least one candidate. The input sequence is scanned multiple times by a regular expression, while increasing the number of allowed substitutions, from zero to a third of the size of the module (the $s \leq x$ term in the regular expression). Moreover, the first half of the sequence is also scanned separately to correct for the bias of the regular expressions, and find more distinct candidate for scoring, which can improve the accuracy at a low computation cost. More fractions of the sequence can be scored based on a user-defined parameter depending on how much sensitivity is desired.

Finally, to allow for the representation of components in any order (as, for example, the two interacting components of an internal loop can appear in either order), the components are swapped, and regular expressions are used to mine the sequence for any order. This process requires the use of $a \cdot k$ regular expressions for a k-way junction, where a is the number of expression variants per regular expression, and k is the number of branches. Indeed, only the 5′-3′ strand orders preserving the correct helix order need to be considered.

This method is consistently faster than the alternative for non-hairpin modules (see 'Results' section).

The regular expression is implemented with the python module `regex`, an extension of the standard `re` module that allows substitutions.

Here is an example of such regular expression:

((C[A|G])([ACGU]{3, 72})(A[C|G])

([ACGU]{3, 84})([C|G][A|G][A|G]))\{s <= x\}

In this example there are three strands. The first always starts with C, followed by A or G. The term ([ACGU]{4, z}) defines a stretch of any nucleotide (a gap) of length 3 (the minimum) to a maximum of z. In this case, the first stretch is at most 72 nucleotides (150% of the maximum observed in examples). The second strand starts with an A followed by C or G. Then there is another gap of at most 84 positions. Finally the last strand has three nucleotides, the first is a C or a G, the last two are each an A or a G. The outside parentheses represent regex groups, which are used subsequently to extract those nucleotides from the one continuous sequence returned by the regex library. Finally, the s <= x term is a feature of the python `regex` module that specifies a fuzzy matching, with at most x substitutions. In this expression, s signifies substitution and x is the parameter that determines how many are allowed. In our implementation, x starts at zero and is progressively increased to allow for candidates that are distinct from a perfect match. The maximal value of x depends on the size of the Bayes net and the level of flexibility required by the user.

Scanning input sequences for candidate module sites. The software we present, `BayesPairing`, can take as input any valid RNA sequence. Those above 300 nucleotides are pre-processed (see 'Long sequences' section). This cutoff value was chosen as twice what has been proven to be a realistic length for the local folding of RNAs (19). The accuracy on long sequences can be greatly improved if the secondary structure is known as it can be provided to our pipeline.

Evaluating the probability of the presence of the module at a candidate site

Probability score. All candidates obtained with the regular expression parsing are evaluated with the $\mathbb{P}$(sequence | module) score previously presented, which is sometimes sufficient, but does not account for secondary structure compatibility. Indeed, there are situations in which a set of positions have the perfect subsequence to fold into a specific module, but doing so would require a very suboptimal secondary structure. To account for that common occurrence, the probability score is corrected by a secondary structure compatibility evaluation.

Modeling module insertion as secondary structure constraints. To include secondary structure information, we used an approach very similar to the base pair probability computation presented in `RMDetect` (13). We leverage Cruz and Westhof's assumption that computing a secondary structure ensemble with `RNAfold` is a sufficient approximation of the influence of the module on the secondary structure. Note that we benchmarked `BayesPairing` in the situation where the secondary structure of the input sequence was not known, but a secondary structure provided by the user can be used for more accurate folding and faster execution.

The 3D modules described in the context of this project can have secondary and tertiary structure base pairs. In terms of secondary structure, this implies that all nodes of every module provide information about being part of a canonical base pair, or the location of an absence of canonical base pair. This situation can easily be modeled by RNAfold's (version 2.3, the latest) hard constraints. The candidate sequence c is folded first with RNAsubopt without constraints, and then folded again with constraints (" (","(") " for base paired positions, "x" for unpaired positions) at the positions of the candidate module placement. We use the method presented by RMDetect (13) to assign an energy score to the quotient of the unconstrained and constrained ensemble energies. This energy score is computed with constants $T = 274.5L$ and $k = 1.98717 \times 10^{-3}$ kcalmol^{-1}, from the Vienna package source code and RMDetect. It is defined as follows :

$$BPP_c = \frac{e^{\frac{-\text{Ens. FE}_c}{kT}}}{e^{\frac{-\text{Ens. FE}_{\text{all}}}{kT}}}$$

or, in a simplified version :

$$BPP_c = e^{\frac{FE_{\text{all}} - FE_c}{kT}}$$

It leads us to the final score formula:

$$S = \log(4^N \mathbb{P}(\text{sequence}_c \mid \text{module})) - \frac{1}{w} \cdot \log(\text{BPP}_c)$$

where we typically assign a value between 1 and 2 to w, depending on how much weight is put on secondary structure compatibility. At values lower than 1, the secondary structure information tends to vastly outweigh the sequence signal, and at values higher than 2, the difference between two insertion options often becomes negligible.

Final score and output. As described in the equation for S, we compute the log of the structural context score described above and add it to the $\mathbb{P}$(sequence | module) score. If the secondary structure allows for the insertion of the 3D module at the candidate positions, the constrained and unconstrained folds will be similar and the score will not be significantly changed, whereas unfavorable ensembles will penalize the probabilistic scores proportionally to how unfavored they are. After this correction, the candidate positions are returned to the user with their subsequence, position and probability score. The number of candidate positions in this output is defined by the user-defined parameter n (default value : 4).

Long sequences

Because of the small alphabet, regular expressions do not perform well on sequences longer than 300 bases. Moreover, our methods require multiple RNAfold calls for each sequence, which strongly decreases efficiency and/or accuracy depending on the technique used. This folding difficulty means that this length limitation is not unique to regular expression-based methods, and state of the art software like RMDetect also relies on window scanning for longer sequences. Longer sequences can still be parsed by the software we present, but the input sequence is cut in windows (of size selected by the user). Module insertion site candidates are returned in terms of their position in the initial (long) sequence.

RESULTS

Datasets

While many datasets of RNA 3D modules exist, we chose to evaluate BayesPairing two that contain complex modules that cannot be predicted with previous methods.

The first is Rna3Dmotif (6), which contains arbitrary secondary structure elements (SSEs), such as hairpins and interior loops but also multi-loops, obtained from subgraph recurrences in the Protein Data Bank (PDB) database. Rna3Dmotif features modules obtained from a strict graph match, thus hairpin modules are represented as full hairpins, and the base pairs are always identical between occurrences. All modules with occurrences in more than 20 distinct PDBs annotated by FR3D were included in the dataset, for a total of 98.

The second is RNA 3D Motif Atlas (version 3.2), an exhaustive database of hairpin and internal loops generated from the current representative set of FR3D, with a non-redundant clustering technique based on maximum cliques (9). RNA 3D Motif Atlas features more realistic, flexible modules, which can have different sizes, can include only the interacting bases of a hairpin, and can have different base pairs between examples of the same module. All modules with at least two examples, and for which at least two occurrences had the same number of edges (for a sensible consensus graph representation), were included, for a total of 134.

The k-way junctions of Rna3Dmotif and the flexible module definition of RNA 3D Motif Atlas are complementary tools to validate BayesPairing, and compare it to the state of the art.

Unlike RMDetect, BayesPairing does not directly require a Rfam alignment to learn models. However, information from a Rfam alignment can be used as an input when available and improves the quality of the model when provided.

Validation

Test sets from PDB. Both Rna3Dmotif (see 'Datasets' section) and RNA 3D Motif Atlas learn their modules from PDB graphs, and each example is taken from a PDB structure. The test set we used to evaluate the software consisted of the sequences from those PDB structures as reported in the PDB file. We performed leave-one-out cross-validation on PDB sequences by excluding the corresponding chain from the data, learning and sampling the Bayes net for each input, and testing BayesPairing on the excluded sequence.

Evaluating BayesPairing's accuracy with two metrics. To assess the performance of BayesPairing, we extracted the RNA sequences from the PDB structures that were used to learn each 3D module, and used each of those sequences as an input for the software with leave-one-out

cross-validation. The first metric used to measure the accuracy of a module identification on such input sequence is the proportion of correctly predicted base pairs. A base pair is considered correctly predicted when BayesPairing predicts some interaction, in the Leontis–Westhof nomenclature, and the same interaction is found at the same position in the PDB structure. However, the predicted base pairs does not tell the whole story. Indeed, for many modules including a component that is distant in sequence, it is possible to predict correctly the majority of the positions in the module without predicting any correct base pair. A score of 0 is then quite misleading because although imperfect, this partial information can definitely be leveraged, namely with better secondary structure information about accessible partners. For this reason, we use the proportion of correctly predicted nucleotides as an alternate accuracy metric. The accuracy reported for each sequence is obtained from the highest scored candidate returned by BayesPairing. To put those scores in the context of the filtering tool, we also reported best of five candidates results in Table 1. For that experiment, BayesPairing outputs five candidate insertion sites, and the highest accuracy one is used to compute the accuracy score, in order to demonstrate how much the identification improves when considering more than one candidate. The mean of the accuracy is then computed over the PDB sequences of length under 300, or on a window of 105 to 300 nucleotides known to include this module for longer sequences. A total of 3409 sequences from FR3D-annotated structures composes the test set.

Comparing Rna3Dmotif results to negative sequences

The first key characteristic of a useful probabilistic identification software is that its prediction score is meaningfully correlated with the presence of a module in the input sequence. A fast way to assess whether that is the case is to accumulate sequences of 98 modules, and compile the score returned by BayesPairing for the top prediction, for each sequence. Then, this distribution can be compared to its corresponding background distribution by repeating this method, but after shuffling the sequences while preserving their dinucleotide distribution.

As shown in Figure 2, the score distribution is completely different between negative and positive sequences, which tends to indicate that BayesPairing could meaningfully be used as a filtering tool. We also performed validation on the BayesPairing predictions on shuffled sequences, and confirmed that no module achieved an average accuracy above 0.06, indicating that a strong accuracy store is unlikely to be obtained by chance.

Validating on 98 Rna3Dmotif modules

Rna3Dmotif contains modules that come from strictly equivalent graphs, which means that they are usually near-continuous in sequence (or built from larger strands), and are more conserved than the modules generated by RNA 3D Motif Atlas. It also does not handle redundancy as well as the RNA 3D Motif Atlas, which means that some of the modules are artifacts of similar PDB structures. Nevertheless, this allows us to study the upper limit of the performance of BayesPairing. We observe that a majority of

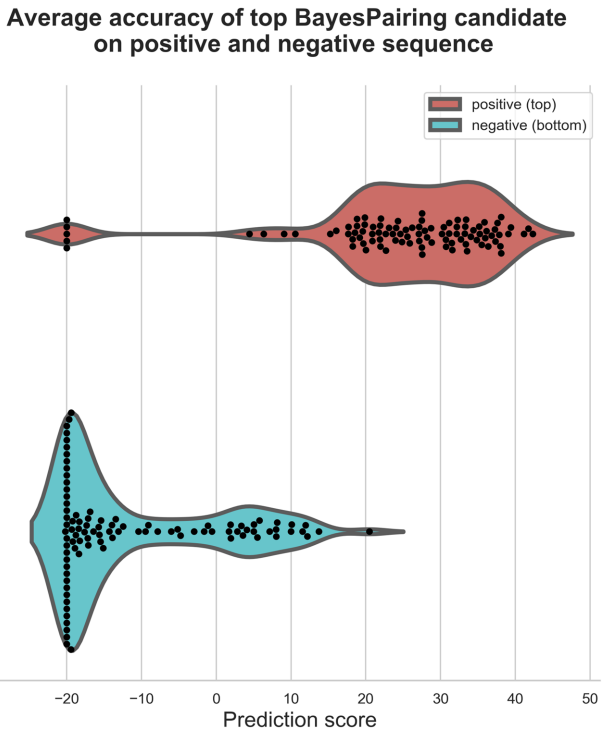

Figure 2. Comparison of BayesPairing module probability score between positive and negative sequences.

modules are identified perfectly, with 63 of the 98 modules achieving the maximum score on the base pair prediction metric.

The position metric results are similar, albeit slightly worse. This discrepancy can be mainly attributed to two reasons. First, some of the occurrences come from PDBs that have an off by one error in their sequence (a nucleotide is absent), which is not caught by FR3D, so the base pair identifications that we validate that way are correct, but the positions that we obtain from the PDB structure itself have a one off error. This is an error that we were able to reproduce on modules of this dataset with RMDetect, a software with a significantly different workflow.

Second, for modules with a distant partner strand, a different partner strand can be predicted by the software which happens to have the correct base pairs. In that case, the base pair identification will be perfect, but the position identification will be slightly off. It should be noted that since those alternate long range partners have the same sequence and base pair signatures, they could potentially constitute viable module insertion sites, and detecting them is not strictly a false positive in the context of a tool that will eventually be used for *de novo* discovery.

Validating on 134 BGSU RNA 3D Motif Atlas modules

Comparatively to the very well-defined and sometimes redundant modules of Rna3Dmotif, the RNA 3D Motif Atlas modules are much harder to predict. Indeed, even though those two module datasets are generated from the same source data, they are curated for different purposes and have different biases. RNA 3D Motif Atlas mod-

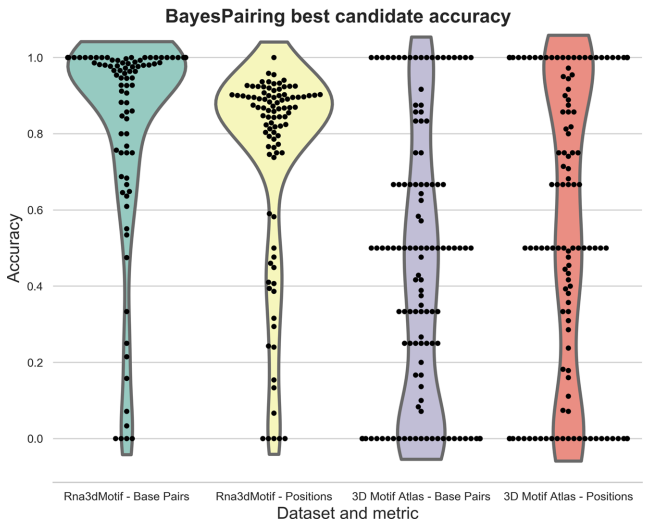

Figure 3. Mean leave-one-out cross-validation accuracy of base pair and position predictions on our `Rna3Dmotif` and `RNA 3D Motif Atlas` datasets, considering the accuracy of a single module location output by `BayesPairing`

ules typically have many fewer occurrences as they are organized based on conserved resulting 3D structure rather than graph structure similarity of the occurrences, resulting in differences between occurrences that, with a small set of examples, can make prediction difficult. Due to 64 of the modules having only two examples, the leave-one-out cross-validation method is particularly punishing for the design of a regular expression because the two examples are often too different in terms of sequence identity, size and base pairs for one to be sufficient to predict the other. As opposed to the previous section that could give us a glimpse at a performance ceiling, this is an opportunity to draw a performance floor. Despite the difficulty of the task, `BayesPairing` is able to find a significant base pair signal for a majority of the modules.

In Figure 3, we observe that 33 of the 134 modules receive a base pair accuracy of zero. Out of those 33 modules, 20 only have two examples. Most of those examples have a difference of size or distance between the two occurrences, which means that `BayesPairing` is searching through the sequence for a module of the wrong size, and explains its failure as a direct consequence of the validation method. Figure 5 shows `BayesPairing` performs much better when given both examples to build its model. Out of the other 13, the software is able to perceive a position signal for all but one; only 1 of the 138 modules is thus truly unaccounted for in terms of signal.

Overall, some modules are very difficult to predict because they have few occurrences and include at least one component that is small, distant in sequence and has low sequence specificity. However, some partial credit should be given for finding one or several of the components that form the module. Indeed, Figure 3 presents RNA 3D Motif Atlas position identification results that are closer to the distribution shown for the other dataset, even for those modules with low base pair signal.

Finally, unlike in the case of `Rna3Dmotif`, the full module is rarely predicted perfectly. The software identifies the location of most of the nodes of the module, occasionally missing out on the base pairs with partners that are distant in sequence, but both datasets show a very clear cross-validation signal. The overall scores presented in Table 1 demonstrate a significant signal, indicating that `BayesPairing` could at least successfully be used as a filter tool for module candidates, as long as it does not perform worse than state-of-the-art tools.

Comparison to `RMDetect`

We need to demonstrate that the use of regular expressions in sequence scanning does not cause a loss in sensitivity compared to other methods. There is currently only one other software that performs quantitative sequence search for user-submitted modules: `RMDetect`, from which some of our methods are inspired. In order to compare the sensitivity of the two software, because `RMDetect` does not yet allow the addition of more than a single model at a time, we sampled 47 modules from five categories in the two datasets. For each dataset, we aimed to select 10 modules for which our software had strong accuracy, 10 for which the accuracy was average and 10 that it could not predict. When fewer than 10 were available in a reasonable range for such categorization, all available modules were used. When more than 10 modules could fit the conditions, 10 were selected randomly. From the `RNA 3D Motif Atlas` dataset, 10 modules among those who received an accuracy score over 0.9, 10 between 0.4 and 0.6, and 10 under 0.2. From the `Rna3Dmotif` dataset, 10 modules among those who obtained 100% accuracy, and the 7 worst, as only 7 modules received an accuracy score under 50%. Due to the paucity of low-accuracy modules in the latter dataset, the low-accuracy category was skipped as including modules with accuracy above 50% in such a category would be misleading.

We also wanted to measure the gain in speed associated with this approach. Because speed mostly depends on the number of components and the length of the sequence, we compared the average speed on sequences of increasing lengths containing two hairpins, one three-way junction and one four-way junction. Since the only category of module for which search speed can be disputed between the two softwares is the internal loop, we compiled the average for sequences from four internal loop modules.

Every test mentioned in this paper was run on Intel(R) Xeon(R) CPU E5-2667 0 @ 2.90GHz, Ubuntu 16.0.4 with 23 cores, with a total physical memory of 792 gigabytes.

Running time. `BayesPairing`'s running time, like most sequence mining tools, is dependent on parameters relating to sensitivity. The two main ones are m, an integer between 2 and 10, which represents the number of module insertion candidates to be structurally tested with `RNAsubopt`, and m, the number of extra substitutions allowed in the regular expression compared to the baseline number of $size/3$ (default 0).

The asymptotically slowest step in the pipeline is the `RNAfold` secondary structure validation step, which takes

Table 1. Mean leave-one-out cross-validation accuracy for both metrics and datasets

	Rna3Dmotif		3D motif atlas	
Candidates	Position	Base pair	Position	Base pair
5	0.794	0.879	0.655	0.472
1	0.660	0.769	0.441	0.379

The number of candidates refers to how many insertion site suggestions are outputted by BayesPairing. The highest accuracy suggestion is considered for this test.

up to 0.3 s for a sequence of length 200, for each candidate. This folding step is essential for a method that scans the sequence itself rather than the secondary structure, as outputting a candidate without any kind of secondary structure information would be reckless. Beyond the folding, the search time increases linearly based on the number of strands of the module.

This is a marked difference with RMDetect, which must scan base pair probabilities for every possible combination of sequence components. In a sequence of length 200, when scanning for a hairpin of 10 consecutive bases, there are only 181 possibilities. However, a four-way junction, with four components of size 10, would have at least 10^8 layouts to test. Hence, the search time increases exponentially with the length of the sequence, reaching times above 15 min per module per sequence to find a k-way junction in a sequence of length 200.

For this comparison, we define the search time as the time elapsed between the start of the sequence scanning and the output of the potential module locations.

The time comparison in Figure 4 demonstrates that the constant cost of folding a certain number of sequences for each test represents a significant time loss for hairpin mining, as a large majority of the search time is spent folding. However, both methods are very fast for that task, and the absolute loss in performance is marginal. The speed of the regular expression search method starts compensating for the folding cost at internal loop search, for which BayesPairing is significantly faster, and reaches its full potential when predicting k-way junctions, which BayesPairing is the only software to predict in close to linear time.

Sensitivity. For the sensitivity comparison, both softwares were set to a comparable level of high sensitivity, predicting between 0 and 100 candidates for RMDetect, and between 0 and 15 for BayesPairing, with no floor on the overall score, nor the base pair probability score. The only requirement was that the considered candidate had to be within 10 points of the best candidate (both softwares have a comparable score scale). Both softwares were tested on up to five sequences from the test set based on how many were available, for each of the 47 modules in the comparison. A software requiring more than 15 min to predict a single module on a 200 nucleotides sequence could not be used in practice. RMDetect not having a timer and being able to reach up to 140 000 s on a single query required the addition of a 15 min timer, after which the software was judged unable to predict the module. The average for each software, for each module, for each category is presented in Figure 5.

Out of those 47 modules, BayesPairing outperforms RMDetect on 28, and is outperformed on 2, as shown by the overall score distribution. It is to be noted that both software produce similar results on modules that are well-suited for identification. The extra sensitivity of BayesPairing allows it to catch insertion candidates that would not be a perfect secondary structure insertion match. It is also able to model modules that are not continuous in sequence, allowing it to outperform RMDetect on very flexible RNA 3D Motif Atlas modules and on k-way junctions. An interesting side result is that BayesPairing appears to perform better on low score modules than on average ones. This is an artifact of the validation method; most of the modules that received a score of zero were severely punished by leave one out cross-validation, and were predicted much more accurately when the tested sequence was part of the model in real-life use.

Overall, it appears the sensitivity cost that could be associated with the regular expression method is mostly nonexistent, as BayesPairing is definitely not less sensitive than the state of the art on the two datasets presented.

Identification of known modules

In the previous section, we have presented cross-validation results for BayesPairing, as well as a comparison to RMDetect in a context of identification. Since this tool is meant to be used in *de novo* methods, we performed additional tests in a context of prediction on untouched sequences, or sequences that were in no way involved in learning the model. Important RNA 3D modules are found in different biological contexts, but remain quite similar in structure. As a consequence, our software should be able to learn a module model from a Rfam alignment of some family containing the module, and then find signal for this module on a distinct family that is also known to contain that RNA module. To illustrate the potential application of BayesPairing to module search for 3D structure improvement, we used two well-known modules for this experiment, *kink-turn 1* (Rfam motif family 10) and the *Sarcin-Ricin loop* (Rfam motif family 18). All families for which Rfam listed a PDB structure containing the module (as of December 2018) were used to build a model, which was then tested on other families, from which no structure nor sequence came into the learning of the model. For Sarcin-Ricin loops, those two families are RF02540 and RF02541, homologs of the ribosomal large subunit, respectively in archaea and bacteria. For the kink-turn, those are RF02540 and RF00162, the latter regrouping SAM riboswitch sequences. Because family RF02541 has many sequence examples and is known to contain kink-turns, the models learnt on the two other families were also tested on that one, although no models were learned on it due to the lack

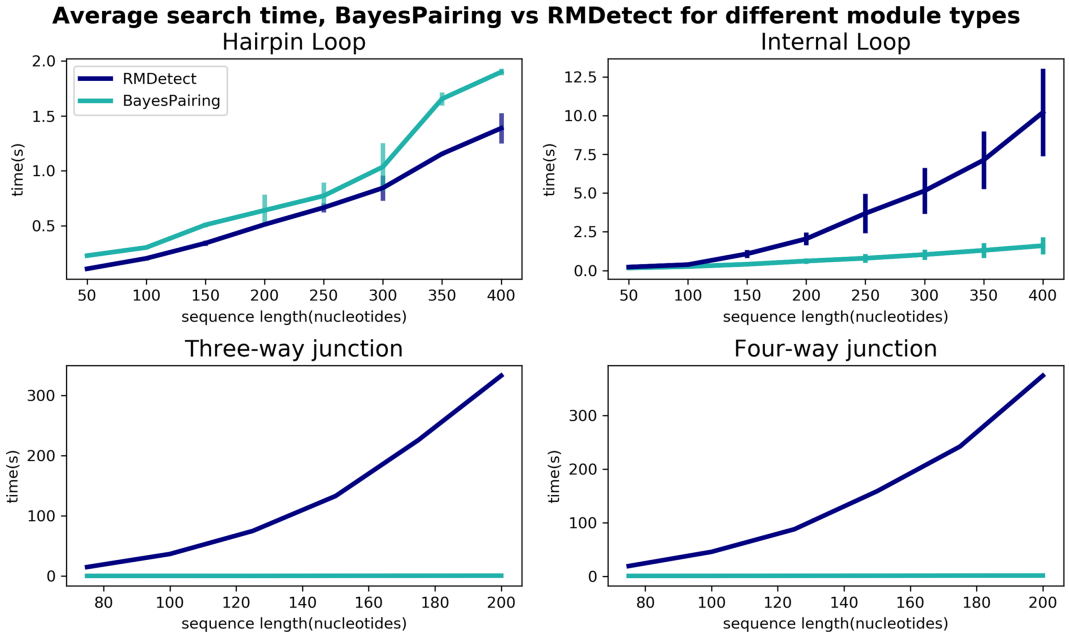

Figure 4. Search time comparison between `RMDetect` and `BayesPairing` for four different categories of modules. Search time includes the time between the start and the sequence scanning, and the output of the insertion sites.

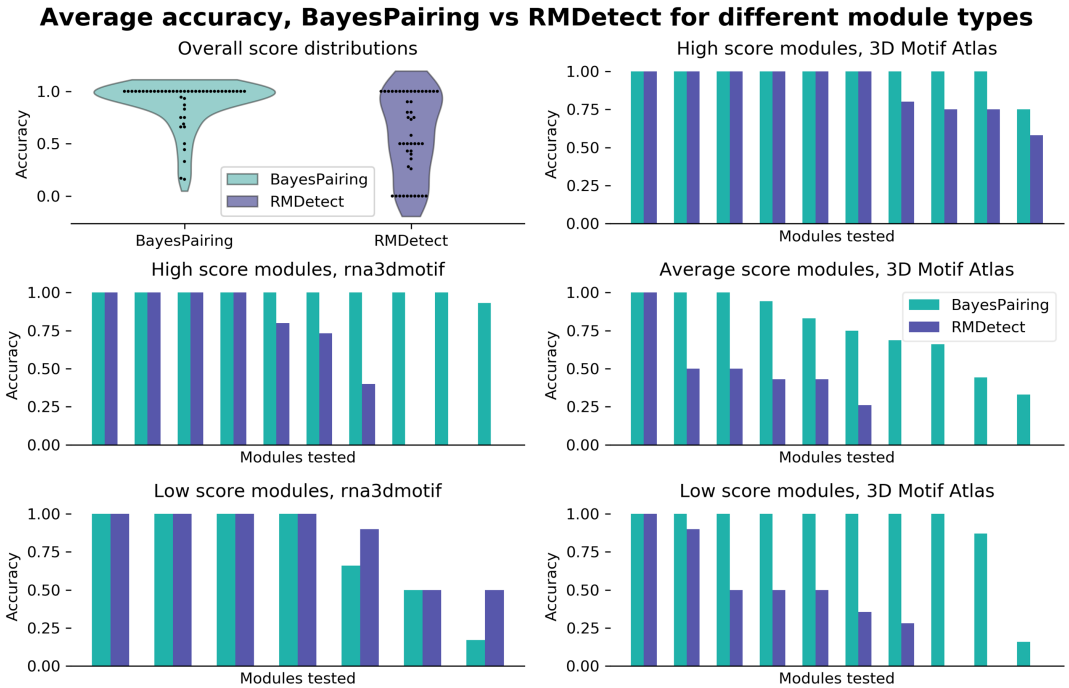

Figure 5. Sensitivity comparison between `RMDetect` and `BayesPairing` for five categories of modules, representative of the variations within the two datasets.

of a listed kink-turn structure of that family. The threshold used for calling a positive was set at a score of 15.7 associated with a false discovery rate of 0.10. Results were also provided for a much higher threshold of 21.6, associated with a FDR of 0.01. Identification results on sequences of the family on which the model was learned were also included, with leave-one-out cross-validation. Despite noisy data (there are slight differences in the sequence signature of same modules between family, so learning a module over alignments of multiple families is strongly recommended), `BayesPairing` can find a signal on new sequences for all families, for both modules, with prediction accuracies ranging from 0.35 to 0.91. Those results are detailed in Tables 2,3 for the Sarcin-Ricin loop, and Tables 4,5 for the kink-turn. Interestingly, the kink-turn models learned respectively from the SAM riboswitch and the archaea ribo-

Table 2. Sarcin-Ricin loop predictions on two Rfam families of large ribosomal subunit, with call threshold 15.7 (FDR = 0.10)

Trained on	RF02540	RF02541
RF02540	0.932	0.912
RF02541	0.875	0.892

Table 3. Sarcin-Ricin loop predictions on two Rfam families of large ribosomal subunit, with call threshold 21.6 (FDR = 0.01)

Trained on	RF02540	RF02541
RF02540	0.920	0.902
RF02541	0.807	0.882

Table 4. Kink-turn loop predictions on the SAM riboswitch and two Rfam families of large ribosomal subunit, with call threshold 15.7 (FDR = 0.10)

Trained on	RF00162	RF02540	RF02541
RF00162	0.966	0.647	0.833
RF02540	0.503	0.960	0.843

The module was not trained on RF02541 due to the lack of a structure.

Table 5. Kink-turn loop predictions on the SAM riboswitch and two Rfam families of large ribosomal subunit, with call threshold 21.6 (FDR = 0.01)

Trained on	RF00162	RF02540	RF02541
RF00162	0.921	0.353	0.602
RF02540	0.467	0.922	0.739

The module was not trained on RF02541 due to the lack of a structure.

some perform better on the bacterial ribosomal large subunit than on each other, which tends to indicate there is enough variation in the module signal between these families for those results to be significant. The full details of this investigation, as well as a more in-depth description of our validation methods, can be found in the Supplementary Data.

DISCUSSION

Main contributions

The central contribution of this work is the identification of the location of k-way junctions from an input RNA sequence. The overall significant gains in speed allow for genome-wide scanning of sequences for a wide variety of modules. BayesPairing provides the automation required to take full advantage of this opportunity. The level of customization allowed is also an improvement on RMDetect, as users can upload any dataset from a single to hundreds of modules. BayesPairing is more user-friendly than its alternatives because it can be run from input to output with a single command.

Applications

The most natural application for BayesPairing is to search new sequences for known modules in order to drive more accurate 3D structure prediction (see the 'Results' section). However, the possibility of designing custom modules can be utilized for searching sequences for any local 3D structure signal. A common example of such a search would be to mine genomes for riboswitches, for instance signal related to the thiamine pyrophosphate (TPP) riboswitch. In such a pipeline, the user would identify recurrent local interactions in structures of this riboswitch, encode them as 3D modules with BayesPairing and then use those models to search new sequences for similar local structures. Given that a probabilistic score threshold for calling a module with high sensitivity would be associated with a false discovery rate of 0.10 to 0.15 (see Figure 2 and Supplementary Data), identifying a single module would likely not be sufficient to draw conclusions. However, BayesPairing can combine the search for multiple modules associated with the TPP riboswitch and, for instance, combining secondary structure-based search methods with the identification of two modules in an input sequence could warrant further investigation. A current method for parsing genomes for modules right now is with infernal's cmbuild (20). While cmbuild is a very strong discovery tool, it focuses on secondary structure and misses out on module information. In that context, combining it with a module prediction tool like BayesPairing is ideal for riboswitch mining.

The TPP riboswitch contains two recognizable modules that occur in all crystal structures of the complex: a very conserved small hairpin of five nucleotides closed by a trans Watson–Crick–Hoogsteen interaction, and a three-way junction with two non-canonical base pairs and a specific set of stacking interactions. Learning two configurations of the three-way junction from PDB structures and the TPP riboswitch Rfam family, BayesPairing achieves a postdiction accuracy of 81/109 with a probabilistic score floor of 15.7, associated with a false discovery rate of 0.10. The small hairpin, which is very conserved in sequence, is detected correctly in 108 of the 109 sequences. Pushing the threshold to 21.6, associated with a false discovery rate of 0.01, still yields a call for both modules on 43 of the 109 sequences, which indicates that BayesPairing is able to capture a significant signal.

From here, we provide an example pipeline for search genomes for a TPP riboswitch local structure signal. We used cmbuild and cmsearch (20) to mine 51 mammal genomes for a model based on the secondary structure of the Rfam family. The 43 hits found were then searched with BayesPairing for the two modules. One candidate was found to have a call for both modules, and those two called modules could be inserted into the Rfam family consensus structure predicted by cmbuild. Figure 6 shows the two modules inserted in the VARNA (21) representation of that secondary structure. Running BLAST on this sequence to find equivalents in other mammal genome yielded no significant result. Further testing is required, but this example workflow takes advantage of several levers to parse large sequences for potential signal while efficiently filtering out false positives, and could improve current methods for riboswitch discovery, as well as other approaches associated with local 3D structure. The details of this pipeline are available in Supplementary Data.

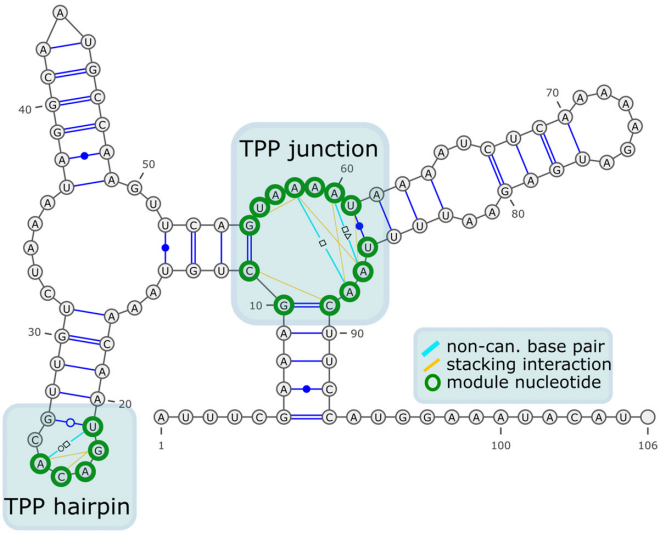

Figure 6. BayesPairing prediction for two modules identified in TPP-riboswitch crystal structures on a short sequence of the Megabat genome (figure drawn with VARNA (21)).

Limitations

BayesPairing in its current form is a tool that is best used in combination with other methods, as it focuses on informing the user about the affinity of some input sequence for some set of known modules. Thus, it needs to be combined with a module discovery method to build a dataset. Because it does not explicitly consider a single secondary structure in its prediction (unless such structure is given as input), it is best combined with tools like infernal for genome parsing, or RNA-MoIP for structure building. Indeed, basing predictions on ensemble free energy is a powerful and an efficient way of estimating structural affinity, but it is not allowing the software to output a secondary structure with modules inserted like RNA-MoIP does. In particular, situations where distinct modules are predicted with high confidence, but incompatible structure-wise because they cannot appear jointly in the same secondary structure are not yet handled by BayesPairing. This problem will typically be handled by the software tool used jointly with BayesPairing, either upstream by feeding it a secondary structure (or simply the *MFE* structure) or downstream by solving the problem of finding the optimal set of compatible modules. For example, in the case of RNA-MoIP, this is done by an integer programming framework.

A data-centric approach is only as good as its dataset; a high-redundancy dataset that is not representative of the reality of a module will not allow BayesPairing to perform well when predicting that module on new sequences. The size of the PDB dataset for RNA structure is also a strict limitation, as we can only predict modules based on examples we have already observed. Fortunately, as new structures are crystallized and new module sampling methods are developed, the performance of our software will keep improving since it can take full advantage of future data.

The regular expression methods also come with limitations, despite not performing worse in sensitivity than prior work. The 3D signal does not naturally map well to a 1D sequence, and some artificial adjustments, described in the methods, are required to obtain a good sensitivity. However, this sensitivity, while overall satisfying, varies between modules.

As we discover more structural modules that contain interactions between helices, junction modules with many disconnected strands, and new RNAs featuring occurrences of known modules with components very distant in sequence, alternative methods will have to be implemented, because the small alphabet, combined with the very long and very variable distance between components, will make those modules impossible to predict accurately. It should be mentioned that those modules also cannot be predicted accurately by secondary-structure based methods, but this is a bridge that cannot be gapped only by using regular expressions. Different levels of representation will need to be combined to achieve an accurate prediction of long-range interactions in RNA structures.

Another consideration is that overall, BayesPairing and RMDetect are not designed to be optimal scoring tools. A tool such as JAR3D that was fully designed for scoring remains a superior scoring tool. However, scanning-oriented and scoring-oriented methods can be complementary. For instance, JAR3D could be used downstream of BayesPairing to optimize scoring on the candidates highlighted by our software.

Future directions

BayesPairing is released as a downloadable software used through command line, like RMDetect, but also as a web-server. Sequence-wide prediction of RNA 3D structure can be greatly improved by an accurate identification of 3D modules, since the secondary structures that are most conserved in nature are generally the ones that contain 3D modules. This characteristic of structures has already been taken advantage of by RNA-MoIP (22), with the limitation that only exact sequence matches were allowed. Thus, our software could be used in combination with 3D structure building software (23–27) to improve their predictions. Another potential use could be as a tool for sequence classification. There are currently many tools to classify sequences by sequence or structure motifs, but few of them take tertiary structure information into account, even though that signal appears likely to exist in RNAs with specific functions.

CONCLUSION

We presented a novel software for the identification of RNA 3D modules from sequence data only. It aims to improve the accuracy of 3D prediction tools and facilitate functional annotation of genomes.

We reported results indicating an important signal on modules of various sizes and categories, from different datasets. Inspired by RMDetect, this software proposes key innovations that allow significant advancements in terms of running time, accuracy, range of structures predicted and customization.

Finally, the growth of experimental RNA structures, and thus module databases, promises significant improvements of the performance of these methods in terms of base pair

identification accuracy, but also in the diversity of the architecture of predicted modules, a diversity that BayesPairing was designed to take full advantage of. These advances will have the potential to fill the gap between current secondary and tertiary structure prediction tools.

BayesPairing has been released both as a stand-alone python package and a web-server at http://bayespairing.cs.mcgill.ca.

DATA AVAILABILITY

BayesPairing is freely available as a standalone python library and as a web-server at http://bayespairing.cs.mcgill.ca.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENTS

The authors would like to thank Éric Westhof and Hua-Ting Yao for their helpful comments and suggestions, as well as David Lougheed for his technical contributions to the software tool.

FUNDING

Natural Sciences and Engineering Research Council of Canada [RGPIN-2015-03786, RGPAS 477873-15] to J.W.; Genome Canada [BCB 2015] and Canadian Institutes of Health Research [BOP-149429] to J.W. and N.M.; Fonds de recherche Nature et technologies du Québec through graduate scholarships to R.S.G. and C.G.O., and a postdoctoral fellowship to V.R.; Institute for Basic Science Grant IBS-R020 to V.R.; Funding for open access charge: NSERC RGPAS [477873-1].

Conflict of interest statement. None declared.

REFERENCES

1. Tinoco,I. Jr and Bustamante,C. (1999) How RNA folds. *J. Mol. Biol.*, **293**, 271–281.
2. Lorenz,R., Bernhart,S.H., Höner Zu Siederdissen,C., Tafer,H., Flamm,C., Stadler,P.F. and Hofacker,I.L. (2011) ViennaRNA Package 2.0. *Algorithms Mol. Biol.*, **6**, 26.
3. Mathews,D.H. (2006) RNA secondary structure analysis using RNAstructure. *Curr. Protoc. Bioinformatics*, doi:10.1002/0471250953.bi1206s13.
4. Lemieux,S. and Major,F. (2006) Automated extraction and classification of RNA tertiary structure cyclic motifs. *Nucleic Acids Res.*, **34**, 2340–2346.
5. Sarver,M., Zirbel,C.L., Stombaugh,J., Mokdad,A. and Leontis,N.B. (2008) FR3D: finding local and composite recurrent structural motifs in RNA 3D structures. *J. Math. Biol.*, **56**, 215–252.
6. Djelloul,M. and Denise,A. (2008) Automated motif extraction and classification in RNA tertiary structures. *RNA*, **14**, 2489–2497.
7. Zhong,C., Tang,H. and Zhang,S. (2010) RNAMotifScan: automatic identification of RNA structural motifs using secondary structural alignment. *Nucleic Acids Res.*, **38**, e176.
8. Gardner,P.P. and Eldai,H. (2015) Annotating RNA motifs in sequences and alignments. *Nucleic Acids Res.*, **43**, 691–698.
9. Petrov,A.I., Zirbel,C.L. and Leontis,N.B. (2013) Automated classification of RNA 3D motifs and the RNA 3D Motif Atlas. *RNA*, **19**, 1327–1340.
10. Popenda,M., Szachniuk,M., Blazewicz,M., Wasik,S., Burke,E.K., Blazewicz,J. and Adamiak,R.W. (2010) RNA FRABASE 2.0: an advanced web-accessible database with the capacity to search the three-dimensional fragments within RNA structures. *BMC Bioinformatics*, **11**, 231.
11. Chojnowski,G., Walen,T. and Bujnicki,J.M. (2014) RNA Bricks–a database of RNA 3D motifs and their interactions. *Nucleic Acids Res.*, **42**, D123–D131.
12. Reinharz,V., Soulé,A., Westhof,E., Waldispühl,J. and Denise,A. (2018) Mining for recurrent long-range interactions in RNA structures reveals embedded hierarchies in network families. *Nucleic Acids Res.*, **46** , 3841–3851.
13. Cruz,J.A. and Westhof,E. (2011) Sequence-based identification of 3D structural modules in RNA with RMDetect. *Nat. Methods*, **8**, 513–521.
14. Zirbel,C.L., Roll,J., Sweeney,B.A., Petrov,A.I., Pirrung,M. and Leontis,N.B. (2015) Identifying novel sequence variants of RNA 3D motifs. *Nucleic Acids Res.*, **43**, 7504–7520.
15. Theis,C., Höner Zu Siederdissen,C., Hofacker,I.L. and Gorodkin,J. (2013) Automated identification of RNA 3D modules with discriminative power in RNA structural alignments. *Nucleic Acids Res.*, **41**, 9999–10009.
16. Theis,C., Zirbel,C.L., Zu Siederdissen,C.H., Anthon,C., Hofacker,I.L., Nielsen,H. and Gorodkin,J. (2015) RNA 3D modules in Genome-Wide predictions of RNA 2D structure. *PLoS One*, **10**, e0139900.
17. Kalvari,I., Argasinska,J., Quinones-Olvera,N., Nawrocki,E.P., Rivas,E., Eddy,S.R., Bateman,A., Finn,R.D. and Petrov,A.I. (2017) Rfam 13.0: shifting to a genome-centric resource for non-coding RNA families. *Nucleic Acids Res.*, **46**, D335–D342.
18. Leontis,N.B. and Westhof,E. (2001) Geometric nomenclature and classification of RNA base pairs. *RNA*, **7**, 499–512.
19. Hofacker,I.L., Priwitzer,B. and Stadler,P.F. (2004) Prediction of locally stable RNA secondary structures for genome-wide surveys. *Bioinformatics*, **20**, 186–190.
20. Nawrocki,E.P. and Eddy,S.R. (2013) Infernal 1.1: 100-fold faster RNA homology searches. *Bioinformatics*, **29**, 2933–2935.
21. Darty,K., Denise,A. and Ponty,Y. (2009) VARNA: Interactive drawing and editing of the RNA secondary structure. *Bioinformatics*, **25**, 1974–1975.
22. Reinharz,V., Major,F. and Waldispühl,J. (2012) Towards 3D structure prediction of large RNA molecules: an integer programming framework to insert local 3D motifs in RNA secondary structure. *Bioinformatics*, **28**, i207–i214.
23. Boniecki,M.J., Lach,G., Dawson,W.K., Tomala,K., Lukasz,P., Soltysinski,T., Rother,K.M. and Bujnicki,J.M. (2015) SimRNA: a coarse-grained method for RNA folding simulations and 3D structure prediction. *Nucleic Acids Res.*, **44**, e63.
24. Parisien,M. and Major,F. (2008) The MC-Fold and MC-Sym pipeline infers RNA structure from sequence data. *Nature*, **452**, 51–55.
25. Jossinet,F., Ludwig,T.E. and Westhof,E. (2010) Assemble: an interactive graphical tool to analyze and build RNA architectures at the 2D and 3D levels. *Bioinformatics*, **26**, 2057–2059.
26. Biesiada,M., Purzycka,K.J., Szachniuk,M., Blazewicz,J. and Adamiak,R.W. (2016) Automated RNA 3D structure prediction with RNAcomposer. In: *RNA Structure Determination*. Humana Press, NY, pp. 199–215.
27. Popenda,M., Szachniuk,M., Antczak,M., Purzycka,K.J., Lukasiak,P., Bartol,N., Blazewicz,J. and Adamiak,R.W. (2012) Automated 3D structure composition for large RNAs. *Nucleic Acids Res.*, **40**, e112.