RNA 3D Structure Prediction by Loop Motif Assembly

Paul-Andre Henegar

School of Computer Science

McGill University, Montreal

March 2023, revised May 2023

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science in Computer Science

Table of Contents

List of Figures and Tables

Abstract

RNA is a macromolecule, found in all living beings, composed of repeating building blocks called nucleotides. It folds onto itself into complex structures, and can be studied at different levels of organization: sequence, secondary structure, and 3D structure. The 3D structure of RNA molecules is what defines their biological function, and hence obtaining it is important for studying them.

Many computational methods were developed to predict the 3D structure of RNA starting from secondary structure or sequence alone. One prediction paradigm is to decompose the secondary structure into components and assemble the molecule out of 3D fragments, extracted from known structures, that most closely match those components.

In this thesis, we developed a program, called rna_builder, that builds RNA molecules by assembling together fragments given to it. We then developed a pipeline for 3D structure prediction that assembles together helices and whole loop fragments chosen based on sequence similarity. Finally, we developed a similar pipeline but that selects loop fragments using motif predictions from BayesPairing2. We called the set of utilities we developed rna_bits.

We tested our methods by systematically predicting the 3D structures of RNA molecules from the Protein Data Bank and validating against their known structures.

Abrégé

L'ARN est une macromolécule présente dans tous les êtres vivants, composée de blocs répétitifs appelés nucléotides. Il se replie sur lui-même pour former des structures complexes et peut être étudié à différents niveaux d'organisation: séquence, structure secondaire et structure 3D. La structure 3D des molécules d'ARN est ce qui définit leur fonction biologique, et il est donc important de l'obtenir pour les étudier.

De nombreuses méthodes informatiques ont été conçues pour prédire la structure 3D de l'ARN à partir de structures secondaires ou de séquences uniquement. Un paradigme de prédiction consiste à décomposer la structure secondaire en composants et à assembler la molécule à partir de fragments 3D, extraits de structures connues, qui correspondent le mieux à ces composants.

Dans cette thèse, nous avons développé un programme, appelé rna_builder, qui construit des molécules d'ARN en assemblant des fragments fournis. Nous avons ensuite développé un pipeline de prédiction 3D qui assemble des hélices et des boucles entières choisis sur la base de la similarité des séquences. Enfin, nous avons développé un pipeline similaire mais qui sélectionne les fragments de boucle en utilisant les prédictions de motifs de BayesPairing2. Nous avons appelé l'ensemble des programmes que nous avons conçus rna_bits.

Nous avons testé nos méthodes en prédisant systématiquement les structures 3D de molécules d'ARN à partir de la Protein Data Bank et en les validant par rapport à leurs structures connues.

Acknowledgment

I want to thank my supervisor Prof. Jérôme Waldispühl for his guidance and help throughout my masters as well as for providing funding. I want to thank Prof. Vladimir Reinharz and Roman Sarrazin-Gendron for additional research directions and for answering my questions and providing useful pointers. I want to thank David Becerra for his mentorship and advice. Finally, I want to thank Étienne Reboul and Arnaud Chol for their support and discussions, and to everyone else I worked alongside in the lab for making my experience more social and fun.

Contributions

Under the guidance of Prof. Jérôme Waldispühl and Prof. Vladimir Reinharz, Paul-Andre Henegar developed and coded the rna_bits utilities, designed and implemented the methods, designed and generated the datasets used for validation, and analyzed the results. He wrote and created all the content within this thesis, with the exception of some images licensed under the Creative Commons, for which I gave proper attribution.

1. Introduction

RNA (ribonucleic acid) is an important class of macromolecules found in all living organisms. RNA molecules have many functions, including coding proteins, translating proteins, gene expression and regulation, and catalytic activity, which they do alone or by interacting with other RNAs and proteins. The function of RNA molecules depends on their 3D structure, and hence understanding that structure is essential to studying their biological function, which in turn is important for downstream uses such as understanding diseases and developing therapeutics.

Techniques for experimentally obtaining 3D structures, such as X-ray crystallography and NMR spectroscopy, are time-consuming, and especially challenging for RNA since it is a flexible molecule that can adopt a wide range of conformations. This makes computational methods for predicting RNA 3D structure that more important. Multitude of computational methods have been developed based on a variety of principles, including simulations of quantum mechanics, molecular dynamics, Monte-Carlo simulations, assembly of fragments extracted from known structures, comparative modeling, or interactive manipulation.

In this work, we propose a set of tools and a pipeline for predicting RNA 3D structure from its secondary structure by assembling RNA motif fragments. We specifically focus on simplicity, modularity, and integration with other tools. We test our pipeline by systematically reconstructing a wide range of molecules and comparing them to their known crystal structure.

2. Background

2.1 RNA Structure

RNA is a linear polymer composed of *nucleotides*. Each nucleotide consists of a nitrogenous base (also known as *nucleobase*, or simply *base*), a sugar (normally ribose in the case of RNA), and a phosphate group, connected covalently. The sugar of one nucleotide is connected to the phosphate group of the next with a covalent bond called the phosphodiester bond. The sugars and phosphate groups are said to form the *backbone* of the RNA.

There are 4 main bases that occur in RNA: adenine (A), cytosine (C), guanine (G), and uracil (U). Sometimes however they are chemically modified and become different. Each atom in the nucleotides has a standardized name.

RNA is usually single-stranded, meaning it does not come with long complementary strands like in the case of DNA. It does however fold onto itself or other RNA molecules in intricate ways, forming short double helical segments that define its structure. The structure of RNA is held together by stacking interaction between the aromatic rings of the bases, and by hydrogen bonds between the bases and between bases and the backbone.

Coplanar bases forming stable interactions consisting of hydrogen bonds are said to be paired or to form *base-pairs*. There is a whole nomenclature for the existing types of base-pairs, described by (N. B. Leontis & Westhof, 2001), determined by the

edge that interacts (Watson-Crick, Sugar, or Hoogstein) of each base, and the relative orientation (cis or trans) of the bases in the interaction.

The most common base-pair, and the one that forms the double helices, is the Watson-Crick base pair, which occurs between Watson-Crick edges with a cis orientation. When the nucleotides it pairs are A and U or G and C it's called a *canonical* base pair, which is the most common and stable base-pair. When it pairs G and U it is called a *wobble* base pair.

The GC, CG, AU, and UA canonical base-pairs are all isosteric, meaning they have the same "3D shape" and can substitute for each other without impacting the 3D structure (which is what allows the construction of regular helical structures). The GU wobble pair isn't exactly isosteric to them (and isn't self-isosteric either, meaning GU and UG are slightly different), so it slightly deforms and destabilizes the double helix .

Organization Levels

The Watson-Crick base-pairs and the helices that they form by stacking upon each other are the most stable parts of RNA and tend to form first (Brion & Westhof, 1997; Tinoco & Bustamante, 1999). Based on this insight, it is useful to think of RNA as organized into the following hierarchical levels:

- *Sequence*, also called *primary structure*, is the string of nucleotides in the order it was transcribed, and can be simply written as a string of letters, like "`CCACACCGUUCUAGGUGCUGG`". Note that RNA is directional; if you reverse the sequence it will no longer be the same molecule. The

beginning of the molecule is said to be the 5’ end and the end of the molecule the 3’ end (this refers to the C5’ and C3’ atoms of the nucleotide, which are just the names given to two of the carbon atoms in the structure.) Sometimes we consider RNA “molecules” that consist of multiple disconnected chains, in which case we indicate the separation in the displayed sequence by a character like “+” or “&”.

- *Secondary structure*, sometimes called *2D structure*, is the set of Watson-Crick base pairs. This level allows us to see how the molecule folds and organizes into helices. Secondary structure can be given as a list of nucleotide positions, represented as a graph drawn in 2D, or given in dot-bracket notation (defined below), like “((((((.....)))).)))”. Sometimes some additional base-pairs are considered to be part of the secondary structure.
- *3D structure*, also called *tertiary structure*, is the actual structure of the molecule as it appears in 3D space, including 3D coordinates for all the atoms. Coordinates of hydrogen atoms can be computed from the rest, so they are often not included. *Heavy atoms* refer to atoms that aren’t hydrogen.

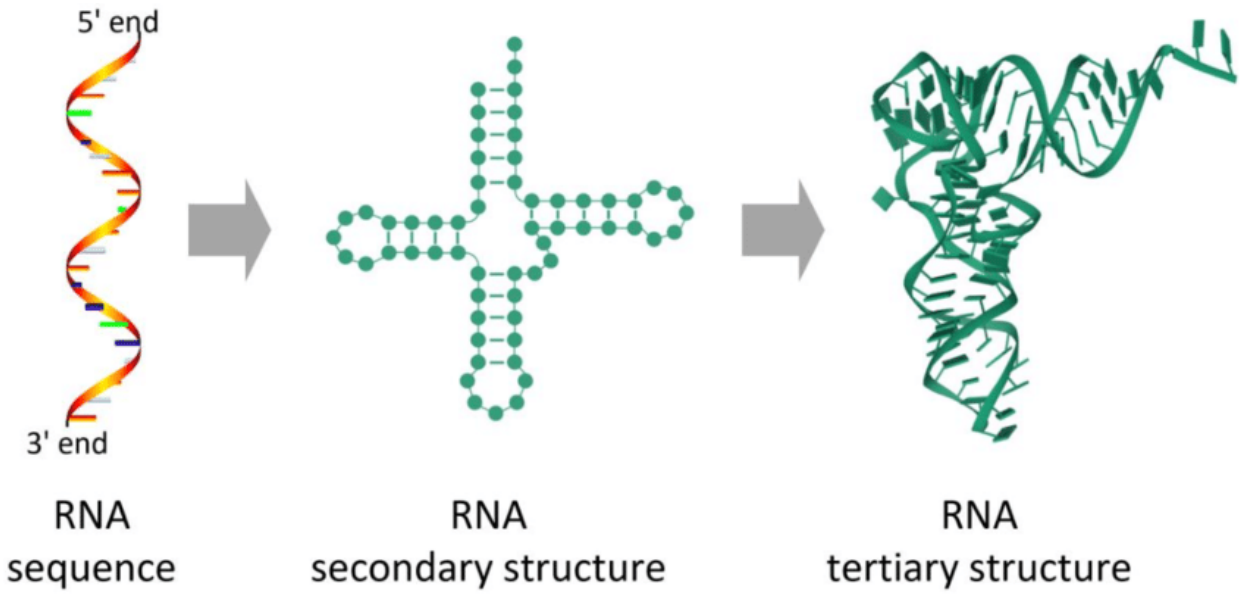


Figure 2.1: RNA structure organization levels.
Figure from (Q. Zhao et al., 2020), used under CC BY-NC-SA 4.0

Secondary Structure

The secondary structure of an RNA molecule can be conceptualized as a graph where the nodes are nucleotides and the edges are backbone connections and Watson-Crick base-pairs.

We might assume that a single nucleotide cannot be Watson-Crick paired to multiple nucleotides. (There are some rare cases where it happens, however in those cases the bases are no longer very coplanar, so it's questionable whether to consider them truly base-paired.) We also might exclude *lonely base-pairs* from the secondary structure, which are Watson-Crick base pairs whose nucleotides aren't directly followed by any other paired nucleotides.

It is useful to define the various components of RNA secondary structure.

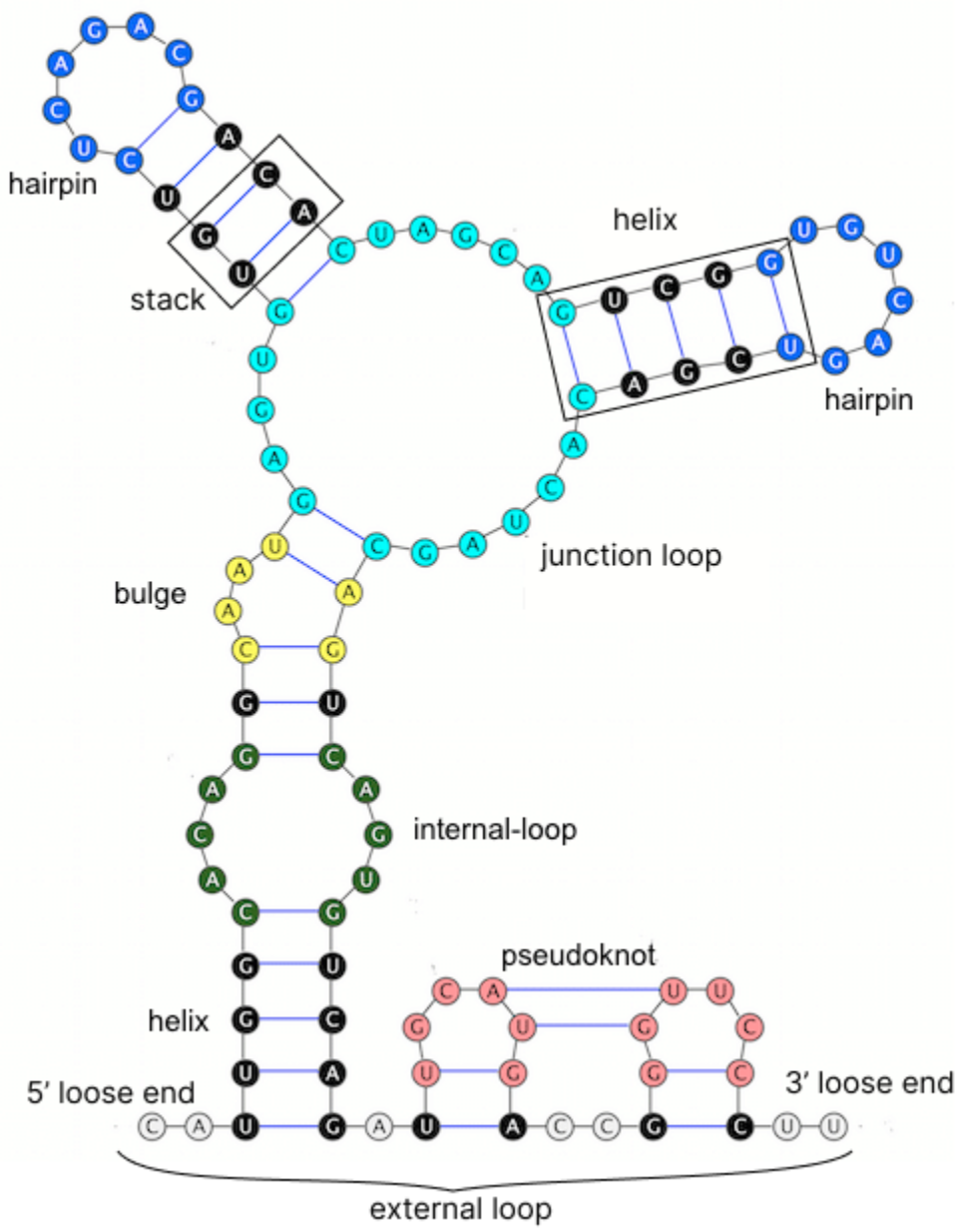


Figure 2.2: Secondary structure components.

Figure from (Mamuye et al., 2016), used under CC BY 3.0 with some labels modified and added.

Stacks are consecutive base-pairs connected on both sides by the backbone. A stack can be formally defined as nucleotides *a*,*b*,*c*,*d* such that

- *b* directly follows *a* in the sequence
- *d* directly follows *c* in the sequence
- *a* is base-paired with *d*
- *b* is base-paired with *c*

In 3D, stacked base-pairs form quite a stable structure. Unless otherwise specified, we usually talk about stacks composed of Watson-Crick base-pairs, but it is sometimes useful to consider stacks composed of other types of base-pairs.

Helices, also called *stems*, are regions composed of one or more consecutive stacks. Alternatively they can be seen as two complementary RNA strands paired together. In 3D, helices form the very characteristic and stable double helix structure.

Loops are cyclical regions in the secondary structure graph delimited by base-pairs.[1] Formally, a loop is a cycle in the secondary structure graph such that:

- No two nucleotides that aren't consecutive in the cycle form a base-pair in the secondary structure[2]
- It is not a stack

[1] Note that this definition is different from the one typically used in proteins, where loops refer to unstructured regions that aren't necessarily cyclical.

[2] They may however form base-pairs not considered in the secondary structure, such as non-canonical base-pairs and lonely base-pairs. In fact, such non-canonical base-pairs are what make loops interesting.

Loops are a generalization of multiple different structures, depending on how many base-pairs delimit them. *Hairpin loops* or simply *hairpins* are delimited by one base-pair. *Internal loops* are delimited by two base-pairs (*bulges* are a special case of internal loops where there's only two nucleotides on one of the sides). *Loops* delimited by more base-pairs are called *junction loops* or simply *junctions* or *multi-branched loops* or *multi-way loops*, or n-*way loops* where n refers to the number of delimiting base pairs.

We say that a structure has a *pseudoknot* if there's one or more base-pairs that "cross" each other. Formally, if we have nucleotide numbers a,b,c,d such that $a<b<c<d$ and a is paired with c and b is paired with d, we call that a pseudoknot.[3]

Note that loops are well-defined even for structures that have pseudoknots (see Figure 2.3). In such cases, certain nucleotides will be shared by multiple loops, and certain nucleotides of a loop will form base-pairs with nucleotides outside the loop. We call those nucleotides a *pseudoknotted region*.

If there's an unstructured region on either side of an RNA chain, we call those regions the 5' and 3' dangling ends, depending on what side it occurs on.

We call the collection of non-cyclical unstructured regions that might occur at the end of stems the *external loop* or *open loop*. It is not technically a loop according to our definition. The logic for that name is that it would become a loop if we were to connect the 3' and 5' ends, and that it might contain interesting structure in the form of non-canonical interactions.

[3] Some works make a distinction between pseudoknots and other structures such as kissing hairpins; we do not make such distinctions and will call all cases where base-pairs "cross" each other pseudoknots.

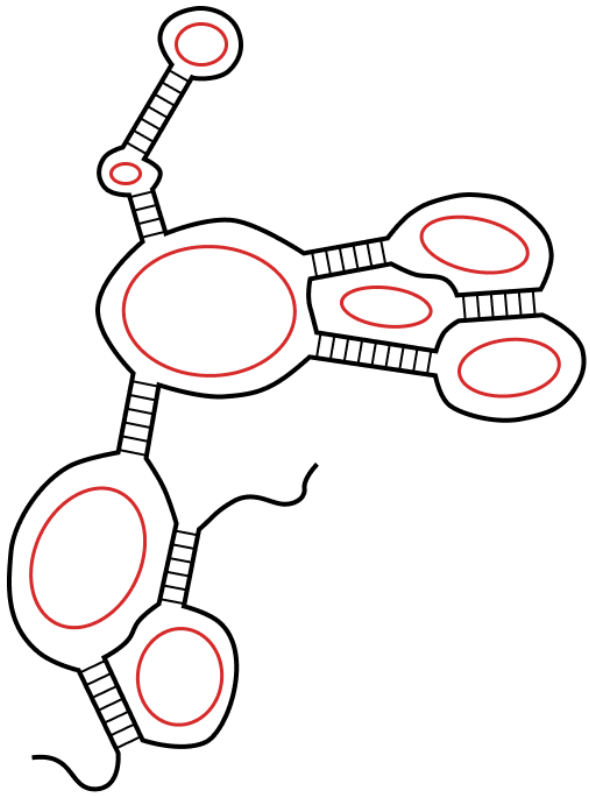


Figure 2.3: Example of loops within a secondary structure with pseudoknots.
The red ellipses represent all the distinct loops.

3D Motifs

The 3D structure is stabilized by interactions such as non-canonical base-pairs and base-stacking, as well as interactions with ions and hydrogen-bonds involving the backbone (Batey et al., 1999). Certain patterns consisting of multiple interactions recur across many structures (N. B. Leontis et al., 2006), and we call those patterns *recurrent motifs* or simply *motifs*. Motifs tend to be evolutionarily conserved, and conserve their non-canonical base-pairing interaction networks and 3D shape even as some nucleotides get changed or added or deleted (Lescoute et al., 2005).

Lemieux and Major decomposed RNA secondary structures with non-canonical base-pairs into recurrent cyclic motifs (Lemieux & Major, 2006). Other databases compile motifs for loops as found within canonical secondary structures, such as Rna3DMotif (Djelloul & Denise, 2008), and the RNA 3D Atlas (Petrov et al., 2013). CaRNAval (Reinharz et al., 2018) compiles general motifs within interaction graphs, and Vernal (Oliver et al., 2022) additionally allows to mine motifs within interaction graphs allowing for some fuzziness.

2.2 Representation

Dot-Bracket Notation

A concise way to represent *secondary* structure is *dot-bracket notation*. In dot-bracket notation, each character represents a nucleotide, and corresponding pairs of parentheses represent base pairs. For example, in "`((....))`", the first nucleotide is paired with the last one and the second is paired with the second-to-last. In cases where there's pseudoknots, some base-pairs will "cross-over" other, and we will have to use multiple different kinds of brackets for dot-bracket notation, for example: "`((((...[[[...))))....]]]`".

3D Molecule Representation

Different RNA 3D modeling methods use different representations for the 3D structure. Some methods, called *all-atom* (*AA*), or *atomistic*, or *full-atom*, represent all the atom coordinates (usually excluding hydrogen) within a molecule. Typical examples

are standard molecular dynamics software like AMBER (Case et al., 2005; Cornell et al., 1995) and CHARMM (Brooks et al., 1983). Other methods, called *coarse-grain* (*CG*), simplify the representation. Some replace each nucleotide by a smaller number of "*pseudo-atoms*" or "*beads"*, ranging from 1 (e.g. NAST (Jonikas et al., 2009)) to 6 or 7 (HiRE-RNA (Pasquali & Derreumaux, 2010)). Others simplify even further by representing the helices by edges of a graph in 3D, such as RNAJAG (Laing et al., 2013) and ERNWIN (Kerpedjiev et al., 2015).

Coordinate system

The natural representation of an RNA 3D structure is by the cartesian coordinates of its atoms. However, since the lengths of covalent bonds and the angles between them stay relatively fixed, some methods fix them and only vary the dihedrals, also known as torsion angles, of the covalent bonds. This allows representing a nucleotide using only 6 angles for the backbone and one for the base, greatly reducing the degrees of freedom (see Figure 2.4). Some coarse-grain models also use dihedral angles. Vfold (Cao & Chen, 2011) simplifies the search space even further by restricting pseudo-atom positions to a lattice in 3D, by allowing only a restricted set of dihedral angle values.

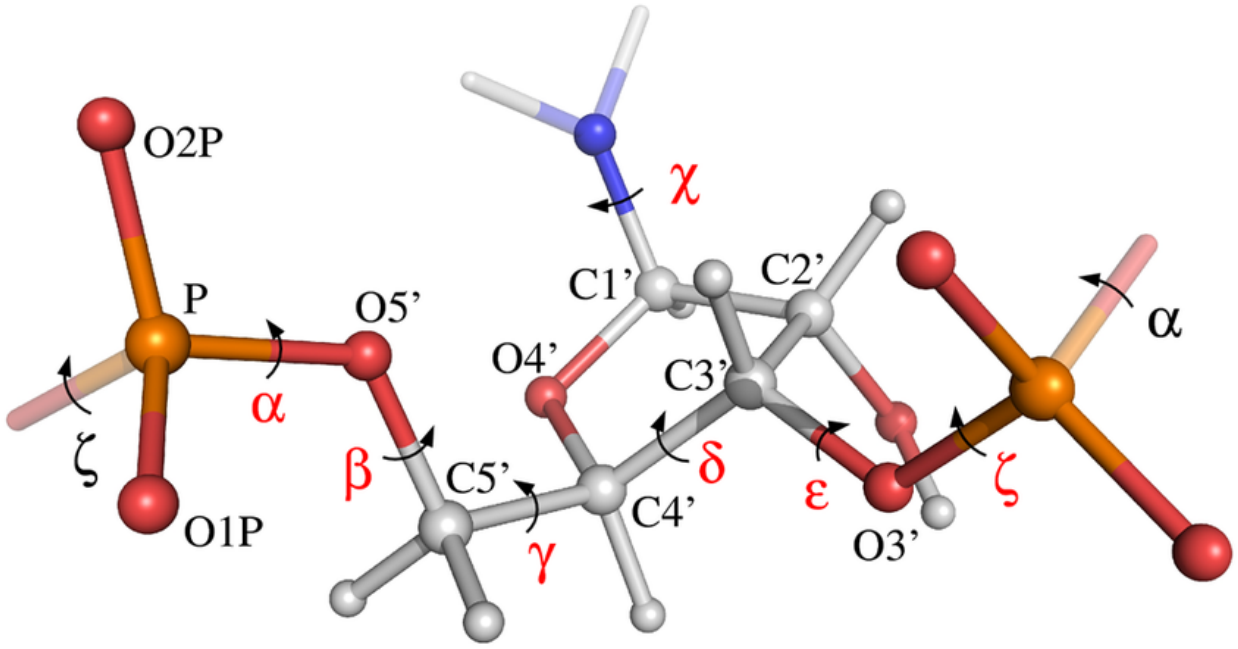


Figure 2.4: Dihedral angles of a nucleotide.
Figure from (Frellsen et al., 2009), used under CC BY 4.0

2.3 Prediction

The goal of 3D structure prediction is to take the sequence and secondary structure of an RNA molecule (or sometimes just the sequence), and obtain a 3D model of how that molecule would look once folded in nature. Lots of different methods were created for this task. According to (Dawson & Bujnicki, 2016), methods can be conceptualized as existing on a continuum between “Greek science”, where modeling happens from first principles of physics, and “Babylonian science”, where modeling is based on the knowledge of existing structures, as displayed in Figure 2.5.

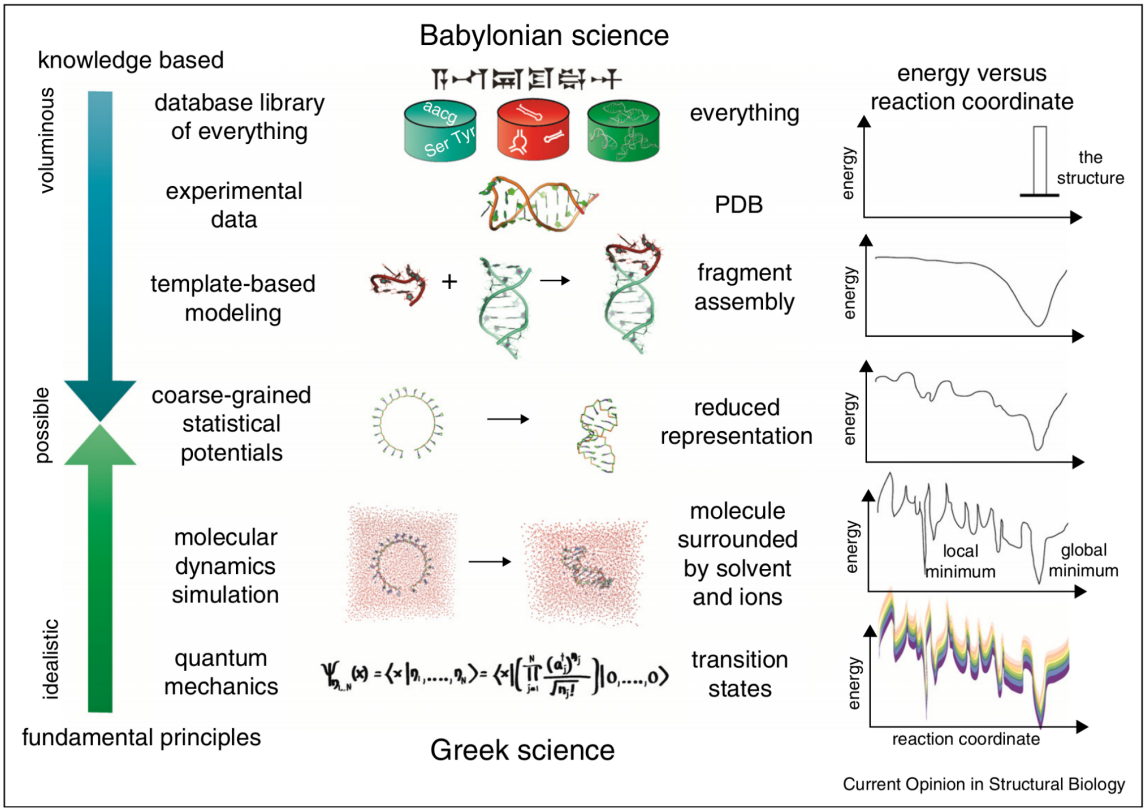


Figure 2.5: Overview of 3D prediction methods as described in (Dawson & Bujnicki, 2016). Used under CC BY 4.0

Assessment

In order to measure the performance of prediction methods, we want to measure how similar predicted structures are to their corresponding *native* structures (the experimentally obtained structure as found in the Protein Data Bank). Multiple different similarity measures are used to compare the 3D structure of macromolecules (Kufareva & Abagyan, 2012). The measures we used were:

- *Root Mean Square Deviation (RMSD)*, which is a measure of global similarity that measures how close corresponding atoms are when two structures are

optimally aligned. More precisely, the RMSD is the minimum or the following expression minimized over all possible 3D alignments of the two structures, where *N* is the number of atoms in the structure, and p_a and p_a' are the positions of atom *a* in the two structures.

$$RMSD = \sqrt{\frac{1}{N}\sum_{a}|\vec{p}_a - \vec{p}_a{}'|^2}$$

- *Interaction Network Fidelity (INF) (Parisien et al., 2009)*, which measures how similar the set of base stacking and base-pairing interactions is between the two structures. More precisely, it is defined as the Matthews correlation coefficient of the two sets:

$$\mathrm{INF} = \frac{|S_n \cap S_p|}{\sqrt{|S_n||S_p|}}$$

- *Distortion Index (DI) (Parisien et al., 2009)*, is a combination of RMSD with INF:

$$\mathrm{DI} = \mathrm{RMSD}/\mathrm{INF}$$

Scoring Function

A *scoring function* (also known as *potential* or *energy*) is a function used to guide the prediction process or to select the best models among multiple predicted candidates. A good scoring function is one that correlates with the quality of the prediction. (This can be assessed by looking at graphs that plot the scores of candidate solutions versus their RMSD compared to the native structure.)

Scoring functions can be created from an understanding of the underlying physics, from the statistical knowledge extracted from known 3D structures, or both. Secondary structure information might be used in the scoring function to enforce that certain nucleotides form base-pairs. Sometimes, additional data, such as data from structural probing experiments such as SHAPE, will be included to guide the 3D structure prediction.

One approach to creating energy function is by modeling the atoms or pseudo-atoms as beads interacting through springs and forces of attraction and repulsion. Sometimes explicit forces are added in between the atoms that we want to be close to each other, called *distance restraints*. The parameters of these forces can be obtained by studying the underlying quantum mechanics, or by attempting to fit parameters to experimental data.

A typical form of such an energy function is as follows, as taken from (Dawson et al., 2016):

$$E_{total} = \sum_{bonds} k_r(r - r_o)^2 + \sum_{angles} k_\theta(\theta - \theta_o)^2 + \sum_{dihedrals} \frac{V_n}{2}(1 + \cos(n\phi - \gamma))$$
$$+ \sum_{i<j} \left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{\varepsilon r_{ij}} \right) + \sum_{H-bonds(i<j)} \left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}} \right)$$

Figure 2.6: Typical physics-inspired energy function.
Figure from (Dawson et al., 2016), used under CC BY 4.0

The first three terms represent the energy contribution of the covalent bonds, the next represents the energy contribution of the forces between atoms and the last represents the energy contribution of hydrogen-bonds. This kind of potential function is used in all-atom molecular mechanics simulation and in some coarse-grain models.

Another approach, known as *statistical potentials*, is to get scoring function terms by estimating the probability of certain features based on their occurrence in known 3D structures. For example, SimRNA (Boniecki et al., 2016) models the probability distribution of neighboring nucleobase configurations using a 3D grid around each base, which was obtained by compiling occurrences from known 3D structures.

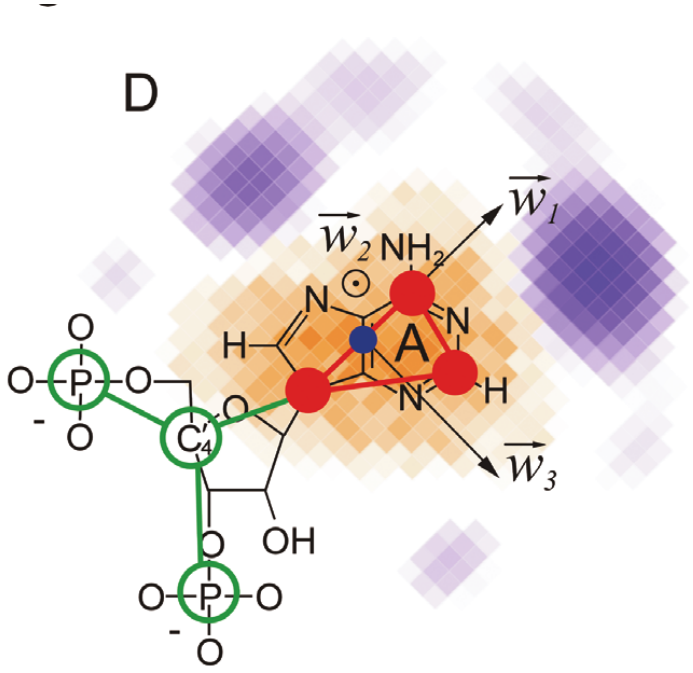


Figure 2.7: A cross-section of a 3D grid statistical potential used by SimRNA.
In orange is represented the volume of the atoms of the base, used to calculate repulsion. In blue are the densities where nearby nucleobases would appear in known 3D structures. Note that this statistical potential takes into account both canonical and non canonical base pairings. Image from (Boniecki et al., 2016) used under CC BY 4.0.

Finally, ARES (Townshend et al., 2021) is a scoring function that uses an equivariant convolutional neural network. It was trained to predict the RMSD of candidates generated by FARFAR2 for 18 small known RNA molecules, and generalizes as a scoring function for larger molecules generated by FARFAR2.

Note that scoring functions can appear at various levels and different scoring functions might be used for different steps of a method. For example, a different scoring function could be used for predicting the 2D structure if applicable, for scoring 3D Monte

Carlo steps, for doing an energy minimization on the final structures, and for scoring structure candidates to pick the best.

Sampling Procedures

Sampling and Optimization Algorithms

To explore the search space and find the best structures according to a scoring function, prediction software use various algorithms. Among those algorithms, there are some recurring techniques.

Monte Carlo is a general term used to refer to randomized algorithms that run for a fixed amount of time. In the world of RNA 3D structure prediction, it usually means applying small “moves” to a system in order to bring it closer to a desired state.

Hill climbing is a basic optimization algorithm that attempts small modifications and keeps those that improve the scoring. *Simulated annealing* is like *hill climbing* but sometimes stochastically allows modifications that worsen the score. As time goes on, the “temperature” is decreased, decreasing the odds of selecting modifications that significantly worsen the score. *Markov Chain Monte Carlo* is a sampling technique and is similar to simulated annealing in that small modifications are applied to a solution, but the aim isn’t to necessarily find an optimum but instead to get a sample from an underlying distribution. *Replica exchange* means that an algorithm simulates multiple copies of a system at different temperatures and occasionally exchanges them in order to break out of local optima.

Molecular mechanics is the modeling of molecules using classical mechanics, where atoms have masses and velocities and electrical charges, and interact with each other through springs and forces. *Molecular dynamics* is the use of molecular mechanics to simulate the movement of molecules. It usually implies that the energy function is differentiable, since force is the derivative of the energy potential with respect to coordinates. However, *discrete molecular dynamics* (Dokholyan et al., 1998) is an approach which uses a stepwise approximation of potential functions, and instead of continuous force being applied, the jumps in the potential are treated as “collisions” that change the bead velocities. *Energy minimization* is an application of molecular mechanics that follows the gradient of the potential function to find stable conformations, but unlike molecular mechanics doesn’t take velocity into account, and is not intended to model the movement of molecules.

Sampling Units

To better understand certain prediction methods, we also need to understand what are the smallest units that are being sampled, or equivalently, what are the moves applied to the molecule in Monte Carlo simulations.

Sampling units can be discrete or continuous. Discrete sampling units are usually “fragments” that have been extracted from experimentally known structures. These fragments can be small, ranging from conformation of single nucleotides, to conformations of small chains or stacks or nucleotides (for example chains of 3 nucleotides for FARFAR2 (Watkins et al., 2020)), to whole secondary structure

components such as loops and helices, to whole molecules in the case of comparative modeling such as ModeRNA (Rother et al., 2011).

Continuous sampling can be obtained for example by modeling the joint distribution of torsion angle, like in Barnacle (Frellsen et al., 2009). Inspired by robotics, KGSrna (Fonseca et al., 2016) uses null space perturbations to randomly deform RNA molecules while preserving hydrogen bonds.

Some methods start with a full representation, containing all nucleotides, and then apply modifications to bring it closer to the desired conformation, such as FARFAR2 (Watkins et al., 2020). Other methods build the representation step by step, potentially back-tracking, such as MC-Sym (Parisien & Major, 2008). Rosetta Stepwise Monte Carlo (Watkins et al., 2018) does something in between, keeping an incomplete representation of a molecule, and stochastically adding or removing nucleotides while being guided by Rosetta's energy function.

Secondary Structure Prediction

There's a multitude of methods that predict (sometimes referred to as *fold*) the secondary structure of RNA.

One common approach is free energy minimization. The classic model is to assign a free energy to each stack and loop, and try to minimize their sum. The free energies for stacks and some other structures have been determined experimentally (Mathews et al., 1999; Mathews & Turner, 2002a; Turner et al., 1988) . Multiple approaches use this principle together with dynamic programming to find the lowest free

energy structure or to sample multiple structures. These include Mfold (Zuker, 2003; Zuker & Stiegler, 1981), RNAfold (Hofacker, 2004), and RNAstructure (Reuter & Mathews, 2010). MC-Fold (Parisien & Major, 2008) expands the model by allowing stacks of base-pairs. The secondary structure prediction done by Vfold (Cao & Chen, 2011) estimates certain energy terms by enumerating loop conformations in 3D. Some methods simulate folding dynamics together with free energy minimization, such as Kinwalker (Geis et al., 2008) and Kinefold (Xayaphoummine et al., 2003).

Another approach to predict RNA secondary structure is comparative sequence analysis. It is based on the idea that secondary structure is more evolutionarily conserved than exact sequences. This approach compares and aligns multiple homologous sequences, and uses information on which nucleotides seem to vary together as hints that they are likely to be paired. The structure of tRNA was solved *manually* in this manner (Levitt, 1969; Madison et al., 1966). Some methods first align sequences and then fold them, such as RNAalifold (Bernhart et al., 2008). Others simultaneously align and fold, such as Dynalign (Mathews & Turner, 2002b), which also includes free energy information. The third paradigm is to fold first, then align, such as with RNAforester (Höchsmann et al., 2004).

Allowing pseudoknots complicates the problem. Prediction of general pseudoknots using energy minimization models is NP-complete (Lyngsø & Pedersen, 2000). However, multiple approaches solve the problem in multiple ways. PKnots (Rivas & Eddy, 1999) predicts a restricted class of pseudoknots using dynamic programming

and some approximated thermodynamic parameters. Kinefold allows the prediction of pseudoknots by stochastically simulating the opening and closing of helices.

3D Motif Prediction

A prediction problem somewhere between secondary structure prediction and full 3D structure prediction is the prediction of 3D motifs within a sequence or secondary structure. Presumably, accurately predicting 3D motifs would help to construct the full 3D structure.

RNA-MoIP (Reinharz et al., 2012) is a program that uses integer programming to select the best secondary structure for the insertion of motifs within its loops. It can remove some base-pairs in order to accommodate more motifs. The original version only inserted motifs with exact sequence matches.

BayesPairing2 (Sarrazin-Gendron et al., 2019) is a program that detects putative locations for loop motifs based on RNA sequence. It samples multiple secondary structures and uses Bayesian networks to model the covariation of nucleotides within motifs, hence predicting the likelihood of motifs occurring even when the sequence does not match exactly.

3D Structure Prediction

Here's the descriptions of a few 3D structure prediction software:

FARFAR2 (Watkins et al., 2020) (successor of FARNA (Das & Baker, 2007) and FARFAR (Cheng et al., 2015)) is a Monte Carlo simulation in torsion angles starting with

a fully extended chain. It stochastically selects 3-nucleotide segments and replaces their torsion angles by torsion angles from randomly selected fragments. Note that this changes the global configuration of the chain. This process is guided by a mix of physical and statistical potentials. It is then followed by a full-atom refinement step.

SimRNA (Boniecki et al., 2016) is a Monte Carlo simulation of a 5-bead coarse grain model with multiple different types of Monte Carlo moves that modify local nucleotide configurations, guided by physical and statistical potentials. Since the moves make modifications that are only local, this method can also be used to simulate the folding process.

iFoldRNA (Ding et al., 2008; Krokhotin et al., 2015) is a discrete molecular dynamics simulation of a 3-bead coarse-grain representation. NAST (Jonikas et al., 2009) is a molecular dynamics method based on an extremely coarse grain representation of a single bead per nucleotide.

Comparative modeling, such as ModeRNA (Rother et al., 2011), takes as input a whole 3D structure of a template RNA molecule and a sequence alignment, and threads the target sequence onto the template structure. Some programs like Assemble (Jossinet et al., 2010) and RNA2D3D (Martinez et al., 2008) allow users to interactively build models.

MC-Sym (Parisien & Major, 2008) is a structure prediction tool based on the assembly of *nucleotide cyclic motifs* (NCMs), which include stacks and small internal loops and hairpins. MC-Sym is intended to work together with MC-Fold, which is a secondary structure prediction program that includes some non-canonical base-pairs in

its prediction, and hence NCMs can have delimiting base-pairs that are non-canonical. To generate areas not fully covered by NCMs, such as large loops, MC-Sym stochastically assembles together chains of 2 to 4 nucleotides.

MC-Sym was also used together (Waldispühl & Reinharz, 2015) with RNA-MoIP (Reinharz et al., 2012) to detect and include larger loop motifs into the assembly process.

RNAComposer (Biesiada et al., 2016; Popenda et al., 2012) takes as input a secondary structure and decomposes it into helices, loops, loose ends, and strands. It then queries the closest matching structures in the RNA FRABASE database (Popenda et al., 2010). Unknown structures are generated using CYANA (Güntert et al., 1997). After assembly, it does two refinement steps, one in torsion angle space, and one in full atom coordinate space. It includes some pseudoknot information when querying 3D structures.

Vfold3D (Cao & Chen, 2011) decomposes the secondary structure into helices and loops and finds the loops in a database based on sequence similarity. In order to increase the number of candidate models, it tries unzipping the delimiting base-pairs of the loops.

VfoldLA (Xu & Chen, 2018) decomposes the secondary structure into helices and *strands*. A 3-way junction, for example, would be decomposed into its 3 constituent strands. This has the advantage that the decomposition works even on pseudoknotted structures. It then finds 3D structures for these strands from a database, based on sequence similarity and the ability for the loops to close.

Vfold3D and VfoldLA were combined (Xu & Chen, 2021), to select whole loops when possible, and otherwise use strands.

3dRNA (Wang et al., 2019; Y. Zhao et al., 2012) decomposes secondary structure into helices, loops, and some pseudoknots. One peculiarity of their way of decomposing structures is that their components overlap each other not at one, but at two base pairs. For components that aren't found in their database, conformations are generated either using a simulated annealing simulation based on bi-residue fragments, or using the distance geometry EMBED algorithm (Havel, 2002). The generated molecules are then optimized using a simulated annealing algorithm, possibly with distance restraints.

Multiple methods can be combined to get the best aspect of each, for example combining ModeRNA and SimRNA (Piatkowski et al., 2016) to use comparative modeling to predict the structure of a conserved molecule core, together with physics to predict the folding of the rest.

2.4 Rationale and Contribution

The original rationale was inspired by the method that combined RNA-MoIP with MC-Sym (Waldispühl & Reinharz, 2015). We wanted a tool that could replace MC-Sym in the pipeline for constructing the final structure from the motif predictions of RNA-MoIP and that would be easier to use and made in-house and be open-source.

Once I had a program that would assemble molecules from stacks and loop motifs, in order to have a vast selection of loop motifs, including junctions, to test with,

and to make the program comparable to RNAComposer, Vfold3D and 3DRNA, I generated a custom library of loops and a program to insert those loops into a secondary structure.

Finally, to add novelty to the method, and to connect with BayesPairing2, I created a script that would bridge between BayesPairing2 and the RNA assembling tool.

The main contribution of my work is that my method is conceptually simple, modular and easy to use. It is also a lightweight implementation compatible with the tools of the lab and that will permit us to create more use cases in the future.

3. Methods

rna_bits is a python library and a set of command line utilities that work together to generate RNA 3D structures using fragment assembly. The included command line utilities are:

- rna_builder: a tool for constructing 3D structures
- rna_insert_loops: a tool for selecting loop fragments from a custom database
- bp2_bridge: a tool for integrating BayePairing2 with rna_builder

3.1 Builder

The *rna_builder* utility builds a 3D model of an RNA molecule from the specification of its sequence, secondary structure, and motifs to be inserted.

Input and Output

Input

The input for the *rna_builder* is a ".rass" file (short for "Rna ASSembly"). It is a normal text file whose first two lines are respectively the sequence and secondary structure of the molecule we want to build (the *target* molecule), and the rest represent possible instructions.

Comments are represented by starting a line with "#". Blank lines after the first two lines are ignored. Currently the only instruction supported by the builder is the motif

insertion instruction, which starts by “motif:”, and instructions that aren’t understood are ignored.

The first line represents the sequence of the target molecule, and should be a string composed of the letters “A”, “U”, “C”, “G”, with “&”, “+”, or “ “ to indicate chain separation. No distinction is currently made between uppercase and lowercase letters.

The second line represents the secondary structure in dot-bracket notation. Multiple types of brackets can be used, as well as letters (where the uppercase letter represents the opening “bracket” and the lowercase the “closing” bracket). For example, “((([[[...)))]]]” and “(((AAA…)))aaa” represent the same secondary structure. The base pairs that are represented by matching brackets are intended to be strictly canonical or wobble pairs. (Other types of base pairs will need to be represented as motifs.) Chain separation is indicated using “&”, “+”, or “ “ and should be indicated in both the sequence and secondary structure.

Example:

```
CAGAGUGUAGCUUAACACAAAGCACCCAACUUACACUUAGGAGAUUUCAACUUAACUUGACCGCUCUGA
(((((((..(((........))).((.(((.......)).)).....((((......)))).))))))).
# this is a comment
motif:./my_junction.pdb:      7-10, 23-25, 41-47, 60-62

motif:out/loops/loop_models/NR_4.0_94750.1.pdb/206.pdb:  12-21
motif:/home/paul/NR_4.0_24610.2/20.pdb:     26-28,38,39,40
```

Motif Insertion

Motif insertions follow the format:

```
motif:[motif_filename]: [target_positions]
```

For example:

```
motif:./my_motifs/asdf.pdb: 5-8, 2, 3,10, 11-13
```

This indicates that the nucleotides corresponding to the target positions in the output molecule will be taken from the provided file. The motif file should be a PDB (“.pdb”) or an PDBx/mmCIF (“.cif”) file or a gzip-compressed version of them (“.pdb.gz” or “.cif.gz”).

Nucleotide positions in the output molecule start with 1. When inserting motifs in a multi-chain molecule, the positions should match the positions in the input sequence string. For example, if the sequence we want to build is “AAG&CUU” and we want to insert a motif that spans the G and C, the positions of the motif will be “3,5”.

File paths prefixed with a “./” will be interpreted relatively to the path of the input file. File paths with no prefix will be interpreted relatively to the rna_bits internal data directory. File paths prefixed with “/” represent absolute paths in the user’s file system.

Input and Output Files

Multiple specification files can be provided to rna_builder, in which case it will build them all. To build all the .rass files in a directory of a Unix environment, we can simply execute `rna_builder *.rass`. The output of rna_builder will be one PDB file for

each input file, and the output file names will be the same as the input file names with ".pdb" appended (ending up with a ".rass.pdb" file extension).

Assembly Principle

rna_builder is based on the principle of fragment assembly. For us the fragments are the motifs provided to rna_builder, as well as built-in stacks of canonical and wobble base pairs that are used to generate helices for which no motifs were given.

Specifically, rna_builder figures out the relative orientation between fragments it needs to assemble by 3D aligning the nucleotides they have in common. As such, it works best for assembling fragments that have overlaps with a relatively stable structure, such as base pairs. We used Biopython's Superimposer module to align the fragments.

For the overlapping nucleotides between 2 fragments, it will have to choose which to place into the final structure, and it will do so by prioritizing the nucleotides of the fragment that was defined higher-up in the input file, and prioritizing motifs over built-in stacks.

For almost all the use cases we worked with, the provided motifs were loops that included their delimiting base pairs, meaning that the overlapping regions were always canonical (or wobble) base pairs.

To make sure that fragments can be properly aligned, rna_builder will rename the atoms in the fragments to follow a single convention (see "Normalization" under section 3.2: Loop Library)

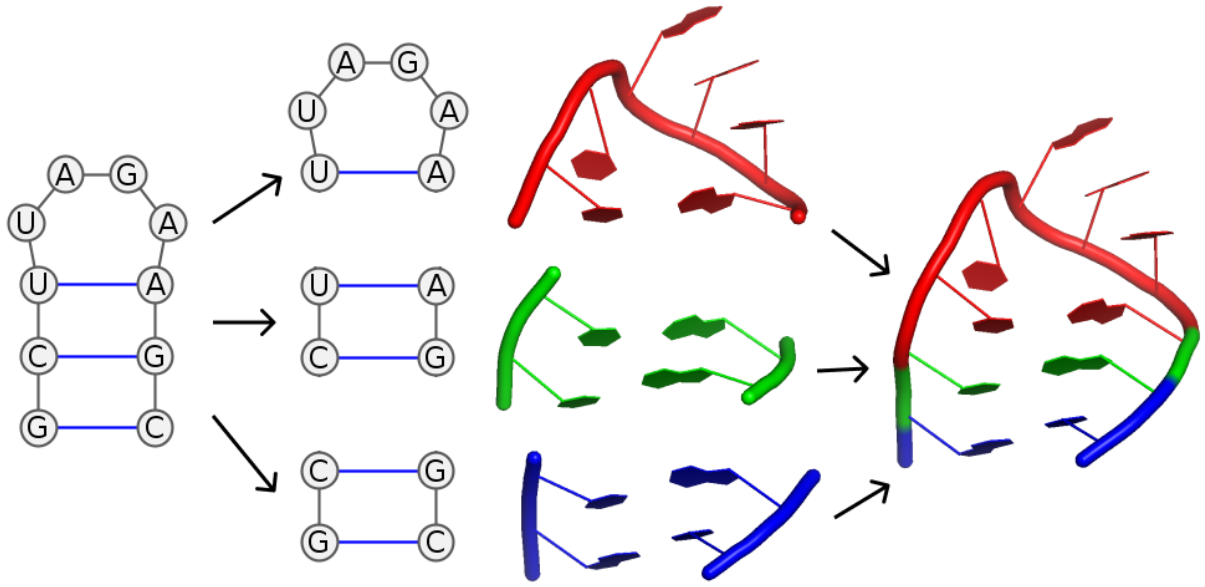


Figure 3.1: Secondary structure decomposition and fragment assembly.

Base Substitution

Motif detection programs such as BayesPairing2 are based on the idea that the 3D structure of motifs is more evolutionarily stable than their exact nucleotide sequence, and that different instances of the same motif can have very different sequences. As such, it's likely that we want to insert a certain motif, but not have a known instance that corresponds to the exact target sequence we want to build. For cases like these, we want to substitute (also known as *mutate*) the nucleotides of the motif to match the target. By default, rna_builder will do this automatically if it detects that the motif and target sequences do not match.

In theory, the structure of the backbone is conserved between instances of a motif, and we only need to change the bases. Base substitution is done by

superimposing (i.e. aligning in 3D) an exemplar base to the base in the motif (using Biopython's Superimposer), deleting the old base, and using the transformation matrix to insert a different base with the right translation and rotation.

Handling of Uncovered Nucleotides and Pseudoknots

If not all nucleotides are covered by fragments, rna_builder will nevertheless build a structure and use various strategies to orient fragments and place nucleotides. In such cases, it is understood that the output molecule is not realistic and is meant to be a starting point for further steps such as molecular dynamics or Monte Carlo.

In all cases, rna_builder will proceed by first assembling the fragments that can be assembled together into connected components, which it then treats as rigid. It then uses various procedures to orient and merge those components, continuing merging until the whole molecule constitutes a single rigid component. Throughout this process, the molecule can be conceptualized as a graph of rigid components, where nodes are the nucleotides and components that have been merged together yet, and where edges are backbone connections linking those components.

Cycles

A cycle in the graph of components may occur in various situations, notably when a loop is not covered by a motif, when a motif has some nucleotides missing that need to be filled in, and when there is a pseudoknot in the structure.

In all cases when there is a cycle, rna_builder will place the rigid connected components and nucleotides of that cycle in a circle in 3D, making the distance between

the C3’ atoms 5.78 Å (which is the distance used by NAST (Jonikas et al., 2009)). It will orient the rigid components such that their centroid points away from the center of the circle to attempt to reduce clashes.

This procedure might happen multiple times, until there are no more cycles in the graph. In some cases, the order in which the cycles are built will affect the final model. If there’s more than one cycle, rna_builder will build the smallest cycle first.

Dangling Ends

For dangling 3’ or 5’ ends, rna_builder will curl them into a helix, as if they were part of an A-form double helix for which one of the strands was deleted.

Exterior Loops

Once there are no more cycles in the graph of rigid connected components, there might still be non-closed paths remaining in the graph, usually corresponding to external loops. rna_builder will do a similar procedure as for the cycles, but placing the components of the paths onto arcs instead of circles.

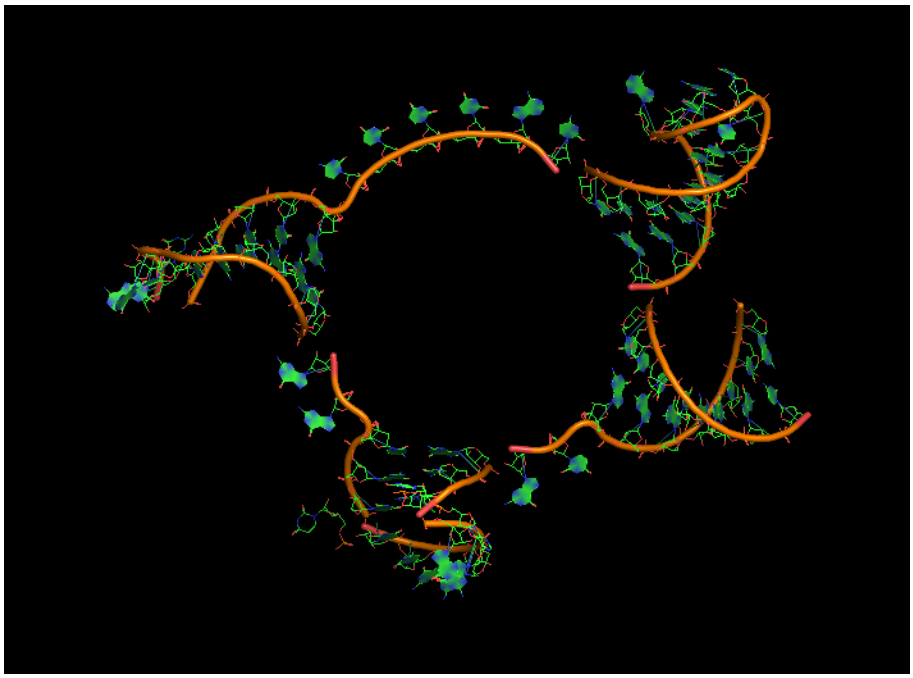


Figure 3.2: Example of a circular structure constructed for a loop with no inserted motif.

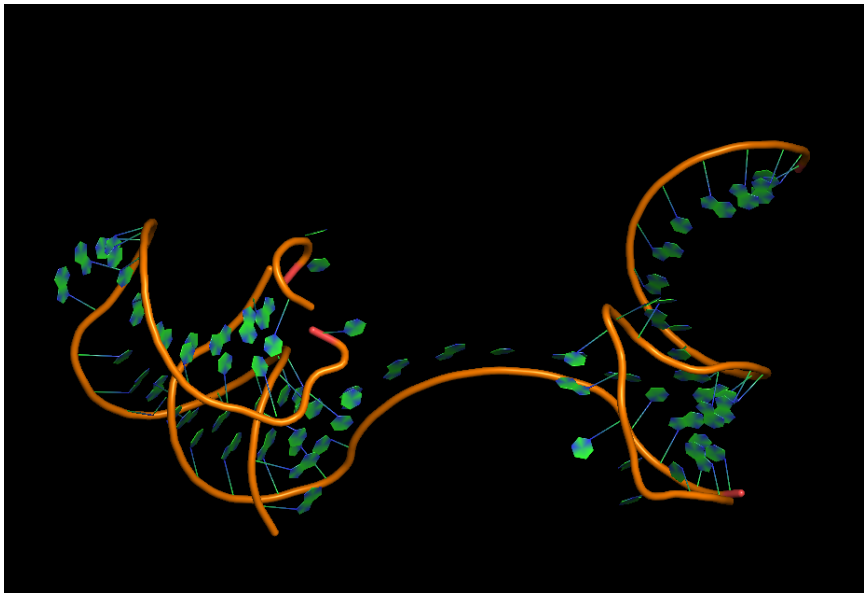


Figure 3.3: Example of an external loop placed on an arc and a dangling end curled into a helix.

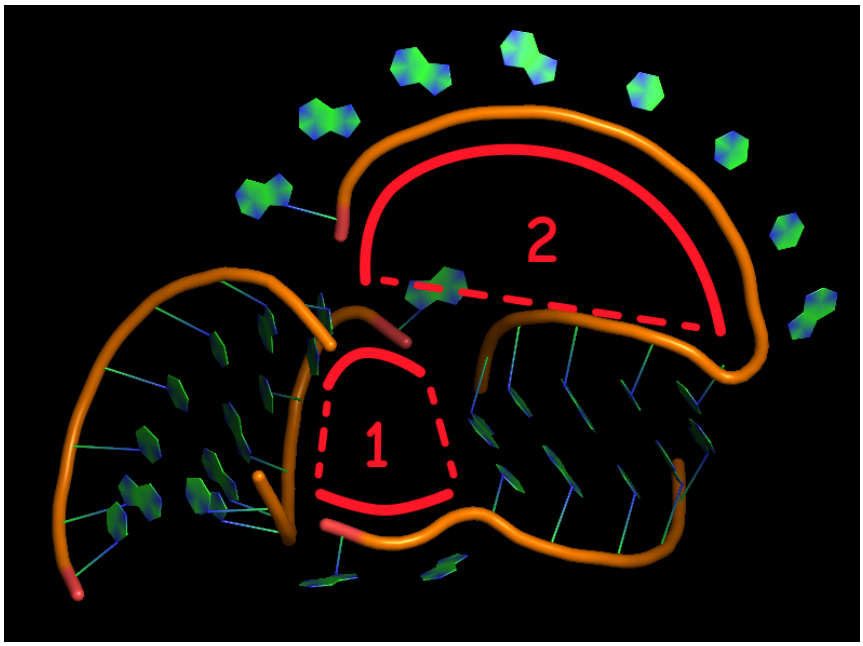


Figure 3.4: Example of nucleotide placements for a pseudoknotted structure, with cycle construction order indicated.

3.2 Loop Library

To build a full 3D structure using rna_builder, we need to provide it with 3D models of the loop motifs to insert. For this purpose, we created a library of 3D loop models extracted from known RNA structures in the Protein Data Bank.

We used BGSU's Nonreduntant Datasets (N. Leontis & Zirbel, 2012) to select the structures from which to extract loops, and MC-Annotate (Gendron et al., 2001) to annotate their base-pairing interactions.

Pipeline

Download the representative structures from the Non-Redundant List

This requires downloading the correct PDB codes and extracting the indicated chains. We used the Nonredundant Dataset 4.0A list, version 3.267.

Normalize them and save them in PDB file format

Normalizing means doing the following things:

- Renaming atoms to follow a single convention, for example converting "O5*" to "O5'".
- Removing nucleotide modification by deleting the atoms that aren't part of the unmodified nucleotide and renaming the residue to the unmodified name.
- Turning nucleotides represented by hetero residues into homo residues.

We do atom renaming and removal of nucleotide modification by using rename lists based on the ones from https://github.com/RNA-Puzzles/RNA_assessment, that can be found in the “data” directory

Additionally, in order to use MC-Annotate, we need to save our structures in PDB format, and in order to fit into the constraints of the format we renamed the chains to single characters and reduced the amount of atoms by deleting waters and some hetero residues.

Run MC-Annotate on the PDB files

We simply run MC-Annotate on each PDB file and save the output.

Extract a secondary structure from the output of MC-Annotate.

Process the output of the above step to create a secondary structure. We annotate a pairing as part of the secondary structure if:

- it is between the right nucleotide types (A-U, G-C, G-U),
- it is annotated by MC-Annotate as “Ww/Ww”, “Ws/Ww” or “Ww/Ws”
- it is annotated as “cis”
- and, if it is *not* annotated as “adjacent_5p”.

Since PDB chains don’t necessarily correspond to actual chains in the 3D structure, and since MC-Annotate does not annotate backbone connectivity, we annotate backbone connectivity ourselves by iterating through the nucleotides in each

PDB chain and checking whether the distance between the O3’ and P atoms of consecutive nucleotides is <= 2.0Å.

We output the secondary structure as a list of chains and a list of nucleotide pairs. This way we avoid encoding the structure in dot-bracket notation and having to decide which pairs correspond to the “main” secondary structure and which are pseudoknots (which is a research question in itself).

Annotate loops with the secondary structure

First, if a nucleotide has more than one canonical pairing in the secondary structure, we only keep one of them, arbitrarily. Then, we remove lonely base-pairs.

In order to efficiently annotate the loops, we turn the resulting secondary structure graph into a directed graph composed of helices and strands, where the direction is 5’ -> 3’. Then, to find loops, we start at each helix and find all the ways to traverse the directed graph and return to that helix, while passing any other helix at most once. In the resulting loop, each helix will either become a delimiting base-pair, or a pseudo-knotted region. Note that loops other than hairpin loops will be counted more than once (for example, a 3-way junction will be counted 3 times with different “rotations”, depending on the helix from which we started traversing). We chose to keep it that way, and it made the rest of our pipeline simpler, since we did not have to write code to rotate the loops.

Save the loops as PDB files and save metadata in a format amenable for querying

We extract the annotated loops and save them as PDBs. We also generate a set of metadata files which allow us to rapidly find all the loops of a given 2D shape. For example, information on all 3-way junctions composed of: a strand of 8 nucleotides, followed by a strand of 4 nucleotides, followed by a strand of 5 nucleotides, will be found in 8_4_5.json. Areas in the loops that form helices with nucleotides outside the loop, which we call *pseudo-knotted areas*, are annotated in the secondary structure using non-round brackets, for example “(....]]()..()...)”. The type of brackets used for the pseudo-knotted areas do not matter, however different characters are used to represent different pseudo-knotted areas.

Statistics of the Number of Loops

	Number of models	Number of models counting rotations a single time
Hairpin loops	6109	6109
Internal loops and bulges	15394	7697
3-way junctions	4443	1481
4-way junctions	3368	842
5-way junctions	1250	250
6-way junctions	534	89
7-way junctions	658	94

Table 3.1: Number of loop models of each type within our library.

Because some RNA molecules, like ribosomes, are huge and have lots of interactions, we end up finding some crazy “loops” such as 13-way junctions with 17 pseudo-knotted regions, that nevertheless satisfy our definition of loops.

3.3 Loop Insertion

rna_insert_loops is a utility for selecting loops from the library described in the previous section based on a given sequence and secondary structure, and outputting .rass files that can then be passed to rna_builder.

Input and Output

To select loops, rna_insert_loops needs the sequence and secondary structure of the target RNA molecule, provided either via a .rass file, or through command line parameters.

The output will be a sampling of rass files with different motifs inserted. By default, for each loop, it will sample among the top 10 motifs closest to the target, based on secondary structure and sequence similarity. By default it will generate 10 output files. Both of these numbers can be changed via command line arguments “--top” and “--num_outputs” respectively. In particular, if you want to only generate the top structure according to its scoring, you can run it with “--top 1 --num_outputs 1”

Example inputs:

```
rna_insert_loops input.rass
rna_insert_loops --seq “CCACACCGUUCUAGGUGCUGG” --ss  “(((((((.....)))).)))”
```

Example output file “il_out/1.rass”:

```
CCACACCGUUCUAGGUGCUGG
(((((((......)))).)))

# Excluding None

# top 5, selected uniformly at random from the top 10 matches
# wanted: ACGCU ((().)  selected: ACGCU ((().)  score: 0.0
motif:/home/paul/Masters/RNA/data/out/loops/loop_models/NR_4.0_58991.1.pdb/1.pdb:  3,4,17,18,19

# top 1, selected uniformly at random from the top 10 matches
# wanted: CGUUCUAG (......)  selected: CGUUCUAG (......)  score: 0.0
motif:/home/paul/Masters/RNA/data/out/loops/loop_models/NR_4.0_58991.1.pdb/2.pdb:
7,8,9,10,11,12,13,14
```

In order to facilitate evaluating the construction of a structure without using any fragments that originally came from that structure, rna_insert_loops has an “--exclude_native” command line flag that, combined with a “native:” command given in the .rass file, allows to exclude all fragments that come from that structure. For example, “input.rass” could look like:

```
CCACACCGUUCUAGGUGCUGG
(((((((......)))).)))
native: NR_4.0_89230.1
```

Principle

rna_insert_loops finds all the loops in the main secondary structure (the one indicated by round brackets), and then samples loop models for each one. (Alternatively, instead of focusing on the round brackets, we could have detected loops using the same segmentation tool we used for generating the loop library, which might give better support for pseudoknots. This would be an improvement for future work.)

For each loop, it looks up all models of that shape (for example, all 8_4_5 loops) and for each one calculates a score based on sequence match, purine/pyrimidine

match, and pseudo-knotted area match. Models will then be sampled from the top models according to that score.

Example output of running rna_insert_loops on a structure with a pseudoknot

```
GGCGCAGUGGGCUAGCGCCACUCAAAAGGCCCAU
((((((..[[[[[[.))))).........]]]]]].

# Excluding None

# top 7, selected uniformly at random from the top 10 matches
# wanted: CAGUGGGCUAG (..[[[[[[.)  selected: CGACCGUCUGG (..]]]]]..)  score: -6.25
motif:/home/paul/Masters/RNA/data/out/loops/loop_models/NR_4.0_72776.1.pdb/4.pdb:  5,6,7,8,9,10,11,12,13,14,15
```

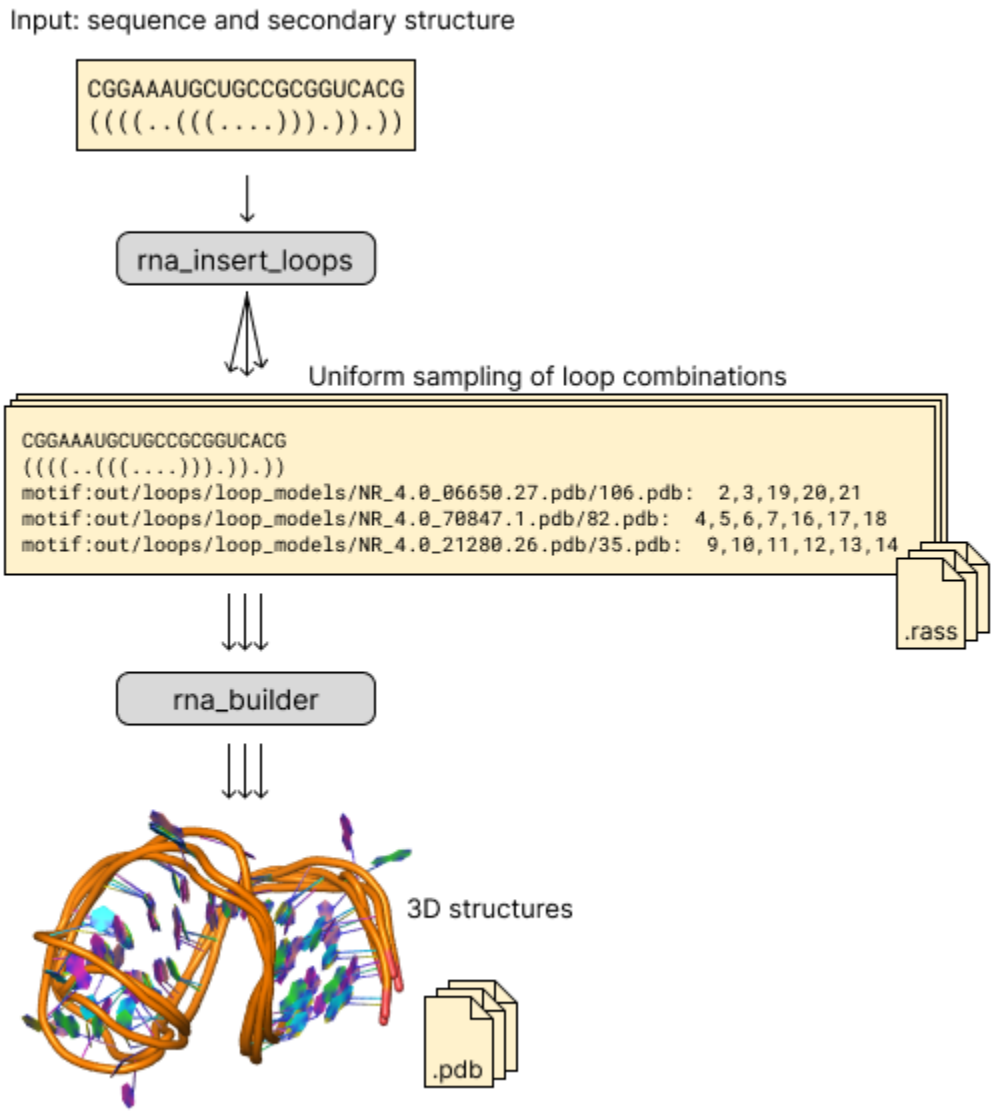


Figure 3.4: Full 3D structure prediction pipeline based on rna_insert_loops.

3.4 BayesPairing2-based Pipeline

BayesPairing2 is a program for finding putative motif insertion sites in an RNA sequence. In order to build a structure containing the motifs proposed by BayesPairing2, we provide the bp2_bridge utility, that takes the output of BayesPairing2, downloads and saves the indicated motifs as PDB files, and generates .rass files to be used as input by rna_builder.

bp2_bridge Input and Output

In order to run bp2_bridge, it needs the output.json file generated by BayesPairing2 and the path to the motifs dataset that BayesPairing2 used (make sure that it matches the dataset BayesPairing2 was called with!). bp2_bridge also needs the secondary structure(s) for which to generate the 3D models, provided via the --secondary_structures command line parameter (unless the --chefs_choice flag is used to generate the “chef’s choice” secondary structure as provided by BayesPairing2.)

By default, bp2_output will save the generated .rass files in a “bp2b_out” folder and will place the saves motifs in “bp2b_out/motifs/”

Example calls:

```
bp2_bridge  --dataset ~/rnabayespairing2/bayespairing/models/ALL.json --bp2_result
output.json --chefs_choice

bp2_bridge  --dataset ~/rnabayespairing2/bayespairing/models/ALL.json --bp2_result
output.json --ss "..(((((((...))))))).((((...((((((....))))))...))))..."
"..(((((((...)))))))..(((...((((((....))))))...)))...."
```

Example output: bp2b_out/out_0_0.rass :

```
ACGGUGUAUAUCGUGCGCUUUGAUGUAGCGGUAGUGAUGCUGUGUGAUCGUGA
..(((((((...))))))).((((...((((((....))))))...)))).
motif:./motifs/with_gap_contents/43/HL_4W90_003.pdb: 33,34,35,36,37,38
motif:./motifs/with_gap_contents/285/HL_3SN2_001.pdb: 9,10,11,12,13
motif:./motifs/with_gap_contents/539/IL_3K0J_003.pdb: 24,25,26,27,28,43,44,45,46,47
```

Principle

For each secondary structure, bp2_bridge considers all the loops, and for each loop considers all the motifs that BayesPairing2 inserted at that location, and samples from them. Note that a motif might have multiple instances, and in such cases bp2_bridge will sample among all the valid instances of all the motifs for that location. Not all instances will necessarily be valid because some might not have a number of nucleotides that matches the structure we are trying to generate.

Selecting Motifs of the Right 2D Shape

The BayesPairing2 datasets that we worked with are based on the RNA 3D Motif Atlas (Petrov et al., 2013). A single RNA 3D Motif Atlas motif family might have motif instances with different numbers of nucleotides in each strand. However, for the sake of generating 3D structures, we want to select motif instances that have a number of nucleotides exactly matching the structure we are trying to generate. The Atlas indicates the “core” nucleotides of the motif, and additional nucleotides might be found in between them. For each motif instance, bp2_bridge downloads the PDB structures from which it came to see where there’s nucleotides in between to decide if that instance is valid.

When outputting PDB files of the motifs, bp2_bridge will output all the nucleotides, not just the core ones.

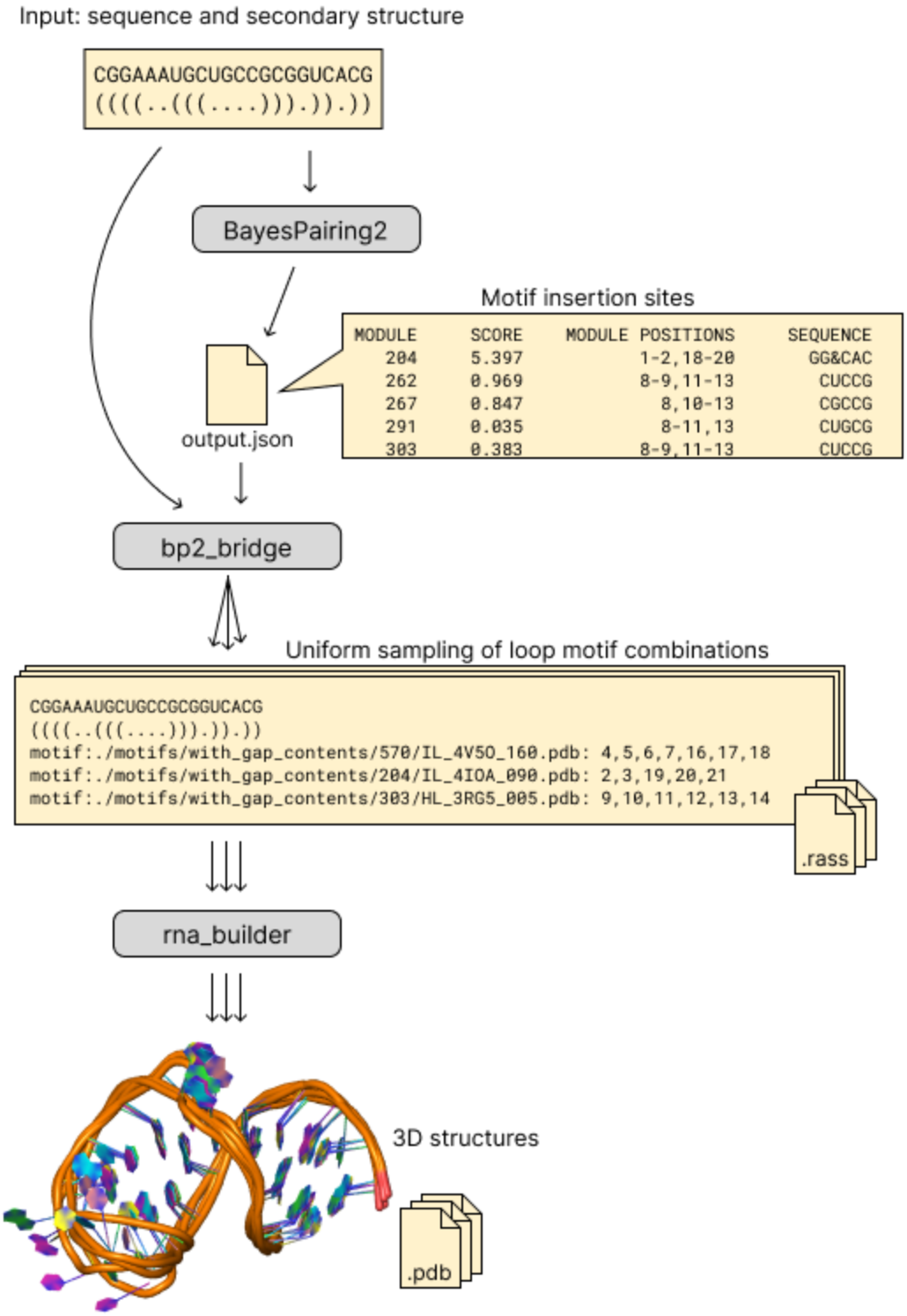


Figure 3.5: Full 3D structure prediction pipeline based on BayesPairing2.

4. Results

4.1 Generating Data for Experiment

In order to evaluate our tools, we automatically generated some datasets based on known 3D structures for the purpose of trying to reconstruct them.

Molecule Selection

We took the molecules from BGSU's Nonreduntant Dataset (N. Leontis & Zirbel, 2012) and filtered them to only keep those that:

- Have a single chain.
- Have some base-pairs.
- Have a single stem at the base of the molecule. In other words, the "external loop" of the molecule is attached to a single stem.
- Do not have both a 5' and a 3' dangling end.
- Do not have pseudoknots
- Do not have insertion codes (This is because the tool we used to evaluate our molecules did not understand insertion codes. Alternatively, we could have kept those molecules and renumbered their residues.)

Since the datasets BayesPairing2 uses only contain hairpin loops and internal loops (no junctions) we also generated a dataset with the additional constraint of no junctions.

Additionally, we removed NR_4.0_83043.1 and NR_4.0_80025.2 because the base-pairs in both were not well formed and MC-Annotate did not detect most of them, giving us a secondary structure with very few base-pairs and what appears to be a giant hairpin loop, even though the 3D model itself has a lot of visible structure and helices.

Analysis of Dataset

Out of the molecules we selected, 125 have no junctions, 122 have a single junction, one structure has 2 junctions and one has 3. The latter two are NR_4.0_47162.1 (3PDR|1|X) and NR_4.0_33922.2 (4R4V|1|A) respectively. Among the single-junction molecules, 51 have a 3-way junction, 67 have a 4-way junction, and 4 have a 5-way junction. Both multi-junction molecules only use 3-way junctions. Calculating the number of junctions was done after removing lonely base pairs.

Although the nonredundant dataset does not contain exact duplicates, it does contain a lot of homologous structures, as can be seen by the clusters in Figure 4.1. For example, the large cluster of molecules with a 4-way junction are all tRNA (Figure 4.2), and the cluster of molecules with 3-way junctions are all instances of the same preserved structure in the ribosome (Figure 4.3). All the molecules with 5-way junctions are also variants of tRNA (Figure 4.4).

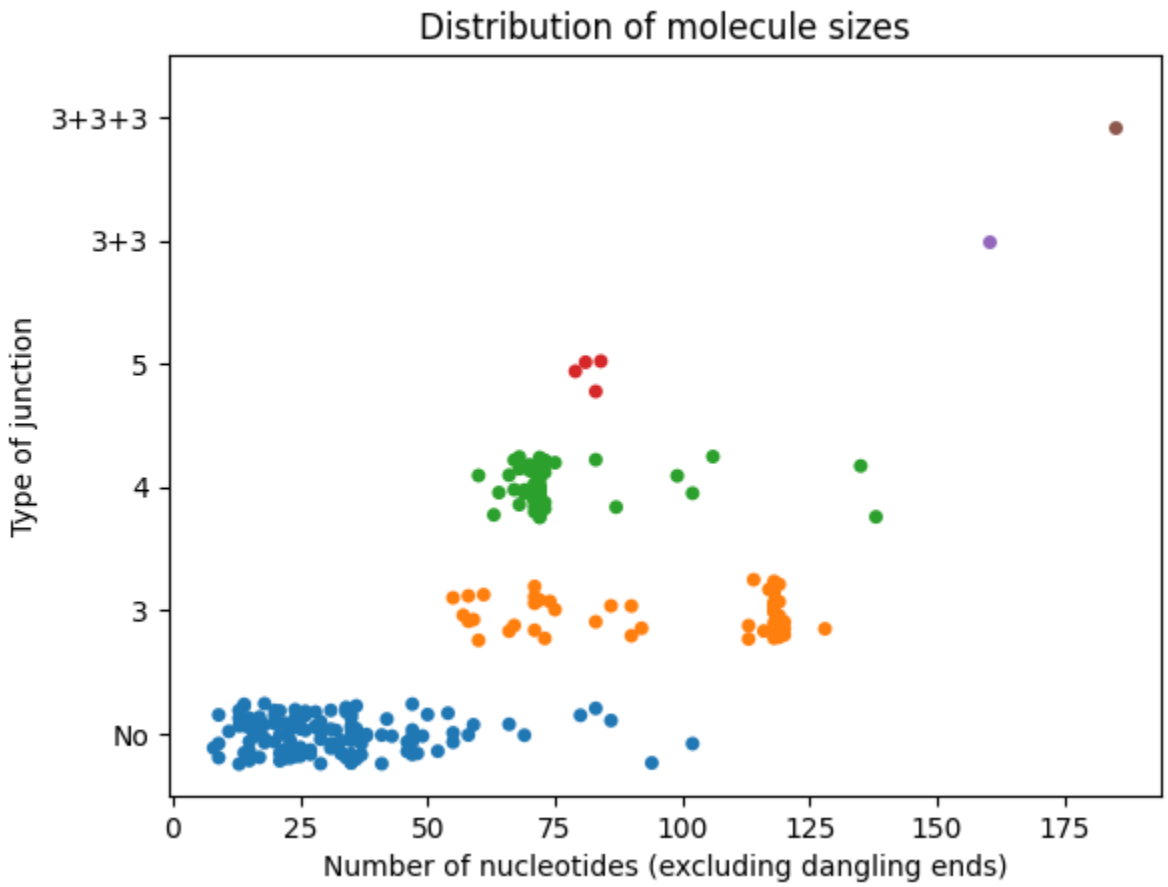


Figure 4.1: Strip plot showing the distribution of molecule sizes in the test dataset by type of junction.

The y-axis categorizes each molecule based on the type of multi-way junction(s) it contains. "No" means it has no multi-way junctions. "3", "4", "5" mean the molecule has a single 3-, 4-, or 5-way junction. "3+3" and "3+3+3" mean that they have two and three 3-way junctions respectively.

The y-axis jitter within each category is random to make it easier for the eye to notice clusters.

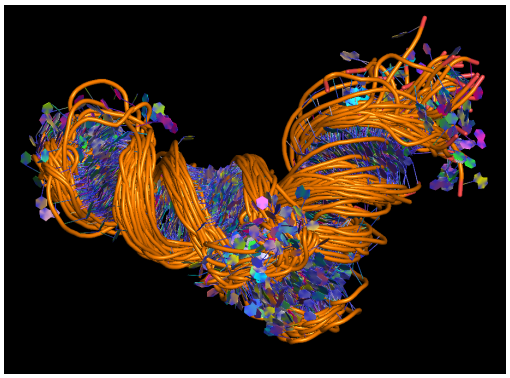


Figure 4.2: Cluster of tRNA within the test dataset.

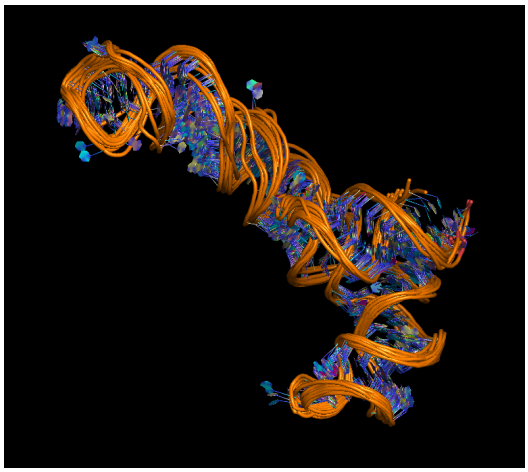


Figure 4.3: Cluster of molecules within the test dataset with a 3-way junction. One instance is NR_4.0_07539.1 (8EUG|1|9)

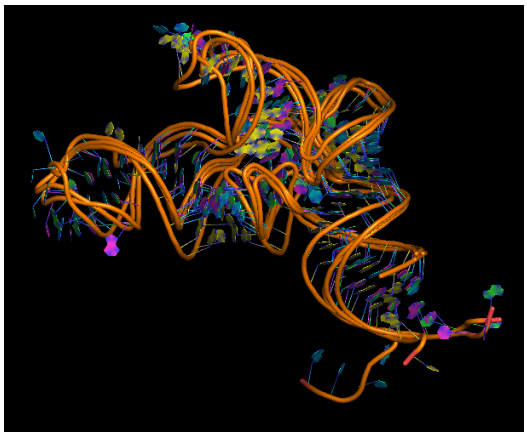


Figure 4.4: The four molecules within the test dataset with a 5 way junction. They are similar to standard tRNA but have an extra fifth helix.

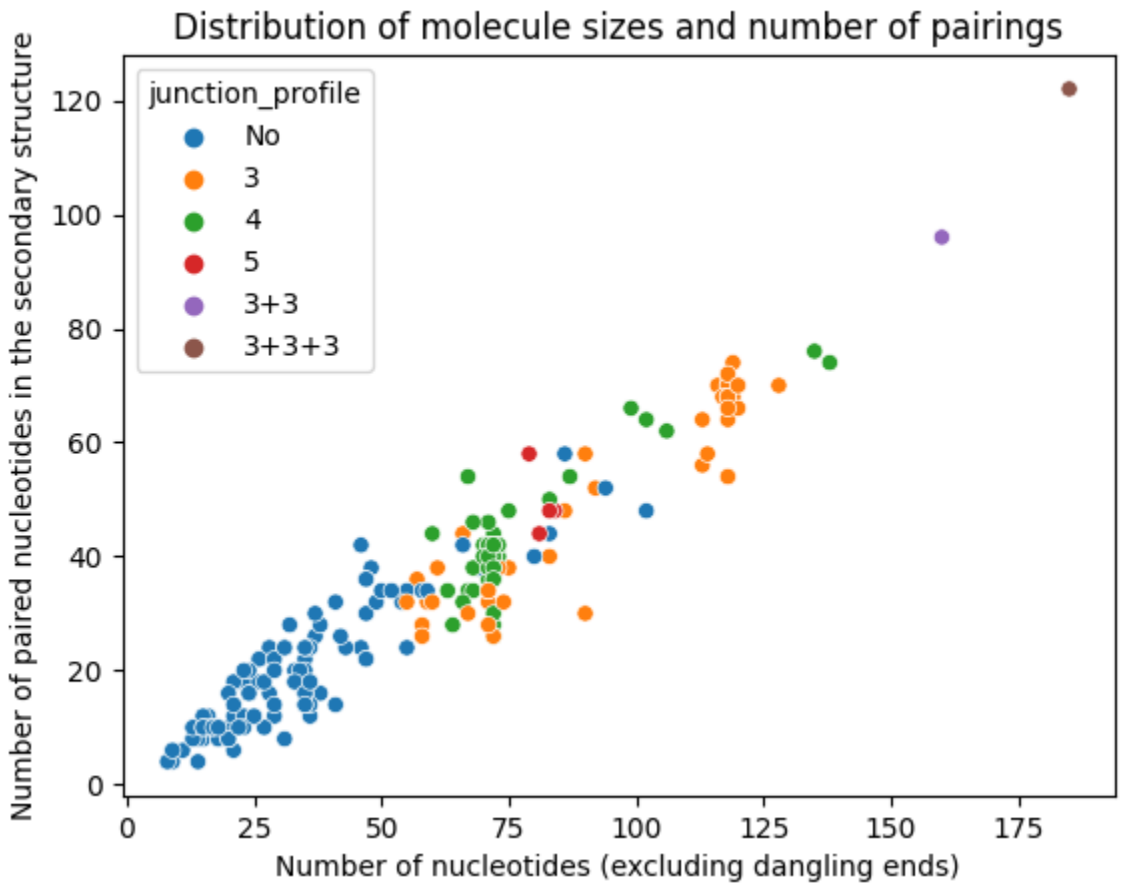


Figure 4.5: Distribution of molecule sizes and number of pairings within the test dataset. The colors indicate what category each molecule is in based on the multi-way junctions it contains.

4.2 Experiment Description

Methods Tested

We tested different methods for inserting motifs into the selected molecules based on their native sequences and secondary structures. For each molecule we had the following methods:

- **from_native**: To evaluate the performance of rna_builder alone, and to have a baseline for the other methods, we inserted motifs that were native to that molecule.
- **loops_sample**: We used rna_insert_loops to sample 50 motif combinations
- **loops_top**: We used rna_insert_loops to generate a single top motif combination according to its scoring function
- **bp2_all** and **bp2_reliable**: We ran BayesPairing2 with secondary structure as input with the ALL and RELIABLE datasets and then used bp2_bridge to sample 50 motif combinations for each. We only applied these methods to molecules with no junctions.

We then used rna_builder to create 3D structures based on all these methods. Note that rna_insert_loops did not insert fragments that are native to the structure being generated.

4.3 Results

Loop Insertion Success

The first thing we evaluate is whether the methods successfully found models for all loops within each input molecule. Since BayesPairing2 does not have a dataset that contains junctions, we only tested BayesPairing2 on molecules with no junctions.

Junction profile:	No	3	4	5	3+3	3+3+3
rna_insert_loops	124/125	43/51	57/67	4/4	1/1	1/1
BayesParing2 RELIABLE	47/125	-	-	-	-	-
BayesPairing2 ALL	99/125	-	-	-	-	-

Table 4.1: Number of molecules that had all loops inserted for each method.

rna_insert_loops inserts more loops than BayesPairin2 for a few reasons:

- BayesPairing2 only uses loops that have been deemed motifs by the RNA Motif Atlas and for which, on top of that, there were enough known homologous sequences to train the bayesian networks. rna_insert_loops, on the other hand, uses *all* the loops that were found in the nonredundant dataset.
- BayesPairing2 only inserts motifs that exceed a certain score threshold, in order to be precise. On the other hand, rna_insert_loops simply chooses the best loops among what it has.

The one loop where rna_insert_loops failed to find a non-native motif for the no-junction dataset was a “(.....().)” loop in NR_4.0_26576.1 (4LCK|1|F) at positions 37-47,69-71. This was the only occurrence of a loop with that secondary structure in the

whole nonredundant dataset. When observing the native structure (Figure 4.6), we can see that that loop interacts with another hairpin loop and maybe for that reason it would not typically appear by itself. BayesPairing2 with the ALL dataset found that motif, but the only instance of the motif group is from 4LCK, the native occurrence.

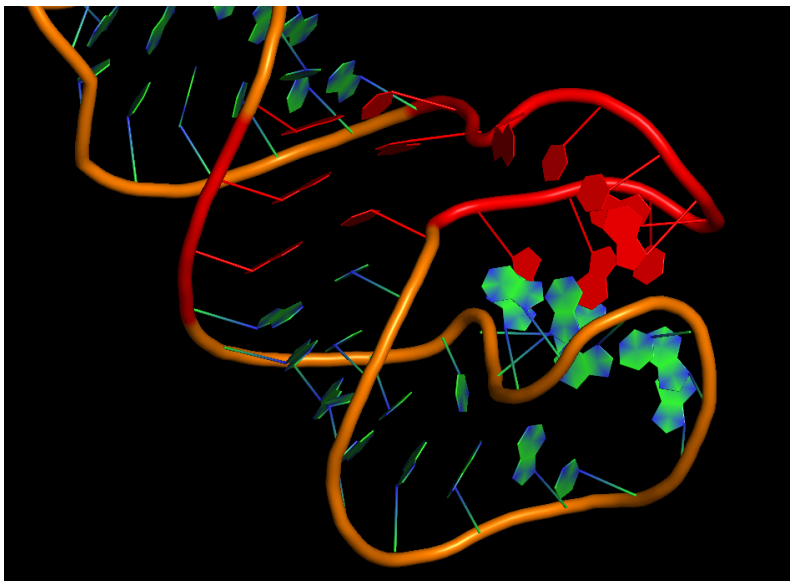


Figure 4.6: The loop at position 37-47,69-71 of 4LCK|1|F (NR_4.0_26576.1), in red. This was the only instance of a (.....().) loop in the whole nonredundant dataset.

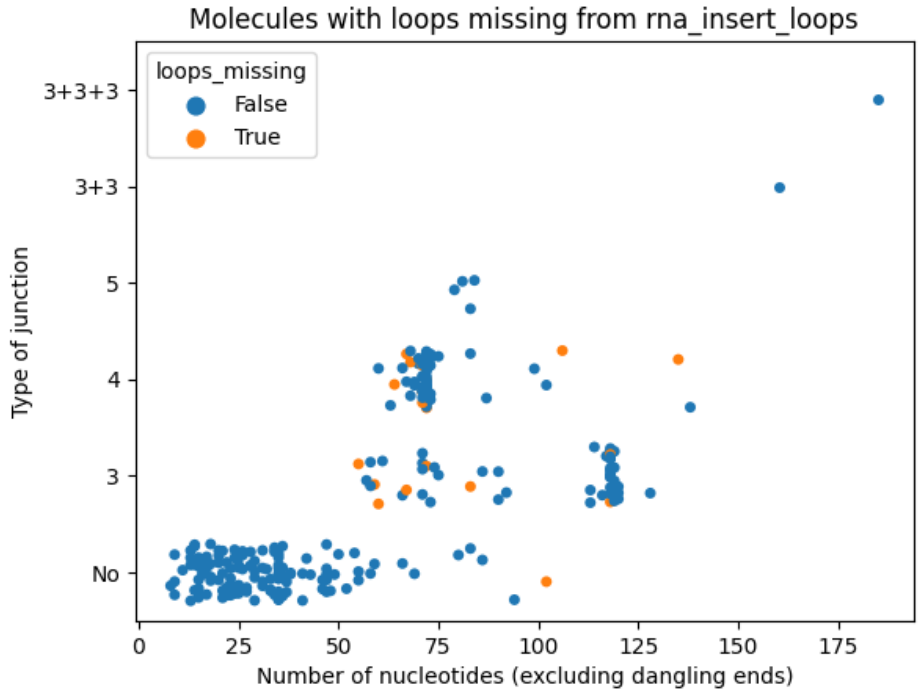


Figure 4.7: Visualization on a strip plot of molecules with missing loops from rna_insert_loops, by size and junction type.

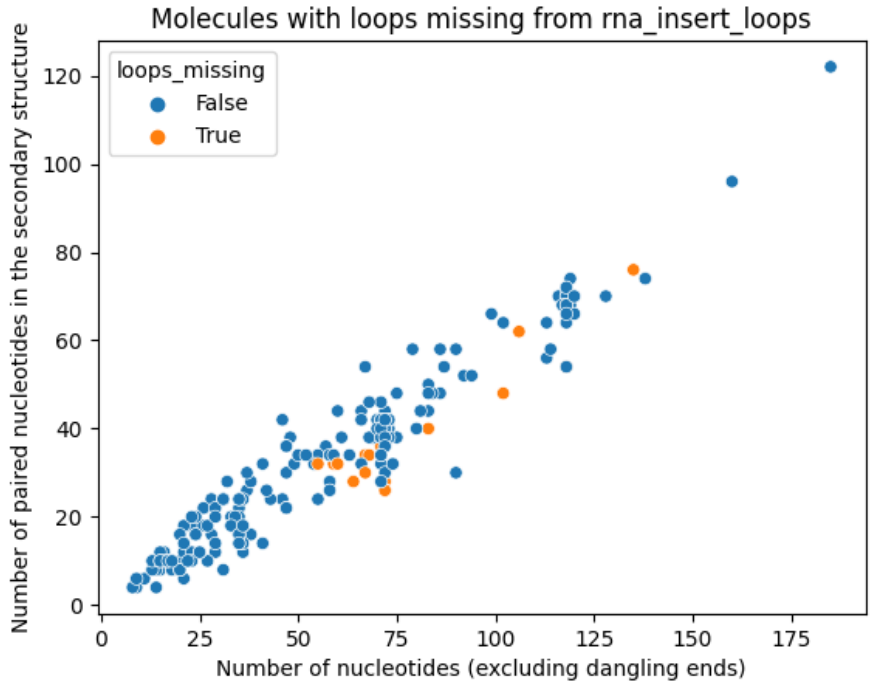


Figure 4.8: Visualization of molecules with missing loops from rna_insert_loops, by size and number of pairings

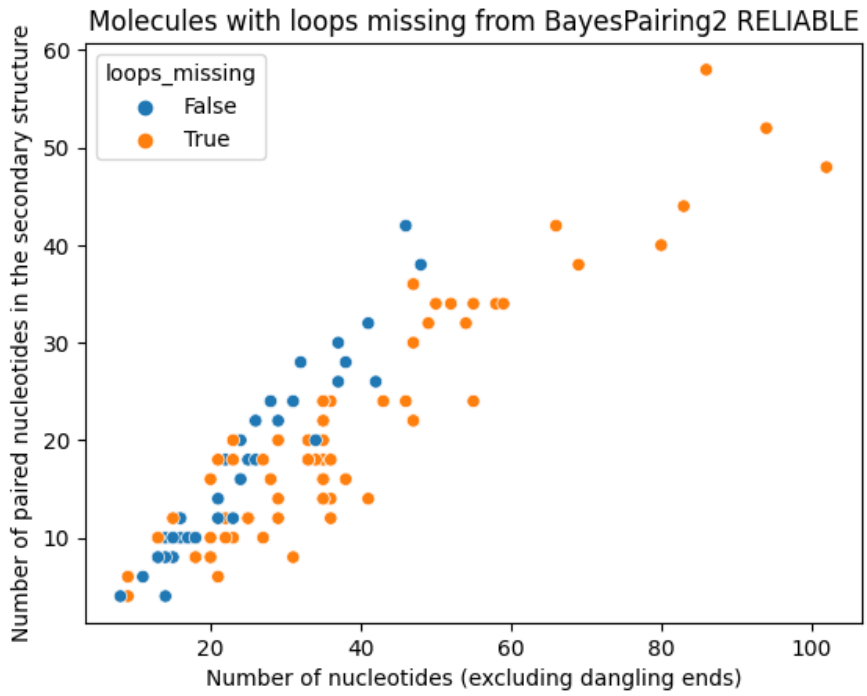


Figure 4.9: Visualization of molecules with missing loops from BayesPairing RELIABLE, by size and number of pairings

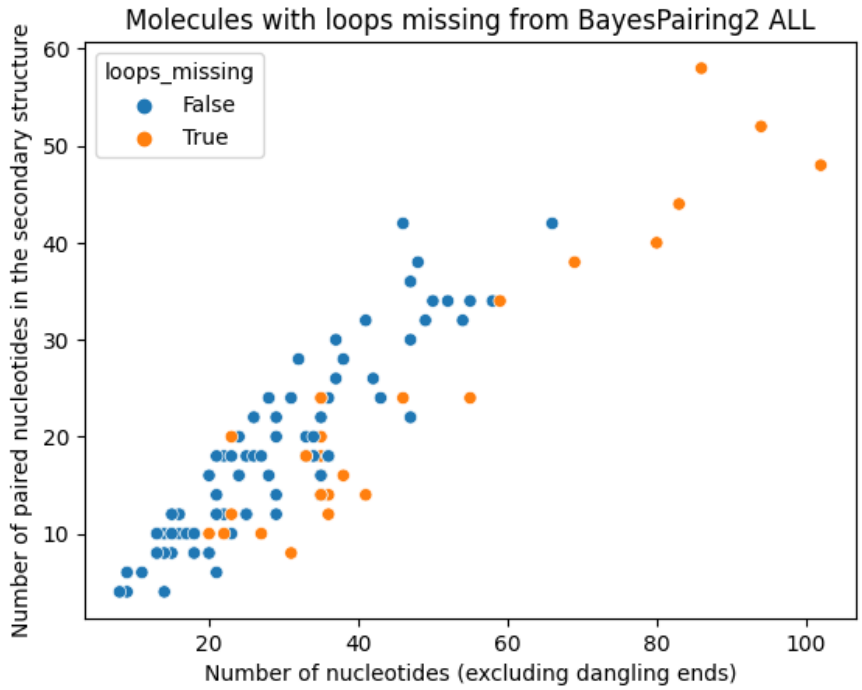


Figure 4.10: Visualization of molecules with missing loops from BayesPairing ALL, by size and number of pairings

3D Structural Similarity

In order to assess the performance of the rna_builder as well as rna_builder together with the rest of the pipeline, we calculated the RMSD, INF and DI of the constructed 3D structures compared to the native ones. We used the RNA_assessment package (Hajdin et al., 2010) provided by RNA-Puzzles toolkit to do those calculations. Since our methods aren't meant to predict the behavior of dangling ends, we did not include them in the calculations.

Performance of Reconstructing From Native

To assess the performance of rna_builder by itself, we reconstructed the structures from their native motif fragments.

RMSD scales linearly with the number of nucleotides, as expected. The outlier at the top is NR_4.0_37229.2 (1MFQ|1|A). It has 3 long helices connected to its central junction and a misalignment of fragments placed one of the helices at a wrong angle, and since it is long, the angle made it deviate a lot from the native structure (Figure 4.11).

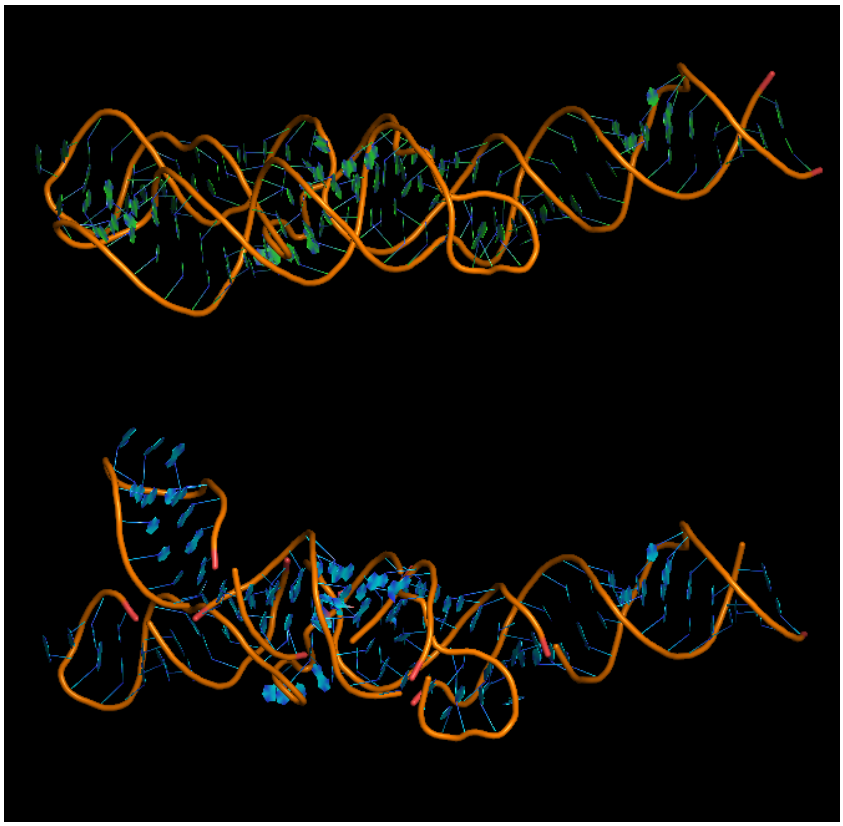


Figure 4.11: 3D structure of NR_4.0_37229.2 (1MFQ|1|A), native and reconstructed from native fragments

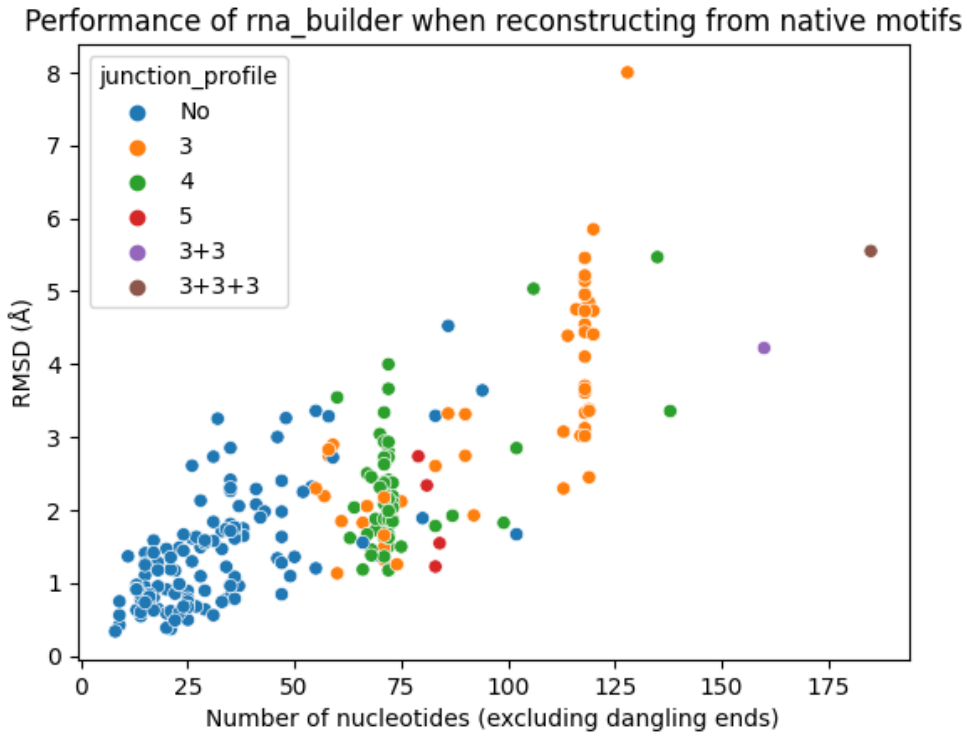


Figure 4.12: RMSD of reconstructing molecules from native fragments, by size and junction type.

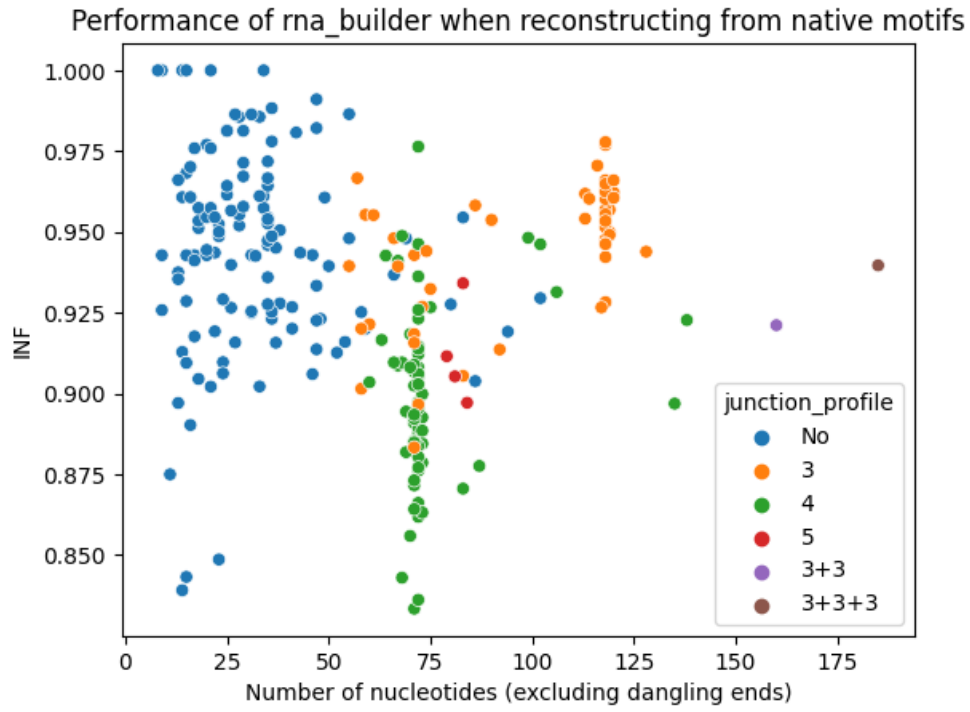


Figure 4.13: INF of reconstructing molecules from native fragments, by size and junction type.

Performance of Full Pipeline

The biggest issue when constructing a molecule is if it does not have a motif provided for one of its loops, and in that case the RMSD will be high. BayesPairing2 is more restricted in its motif selection than rna_insert_loops and BayesPairing2's RELIABLE dataset has fewer motifs than ALL. With this in mind, the relative distribution of the RMSD for the different methods as shown in Figures 4.14 and 4.15 makes sense.

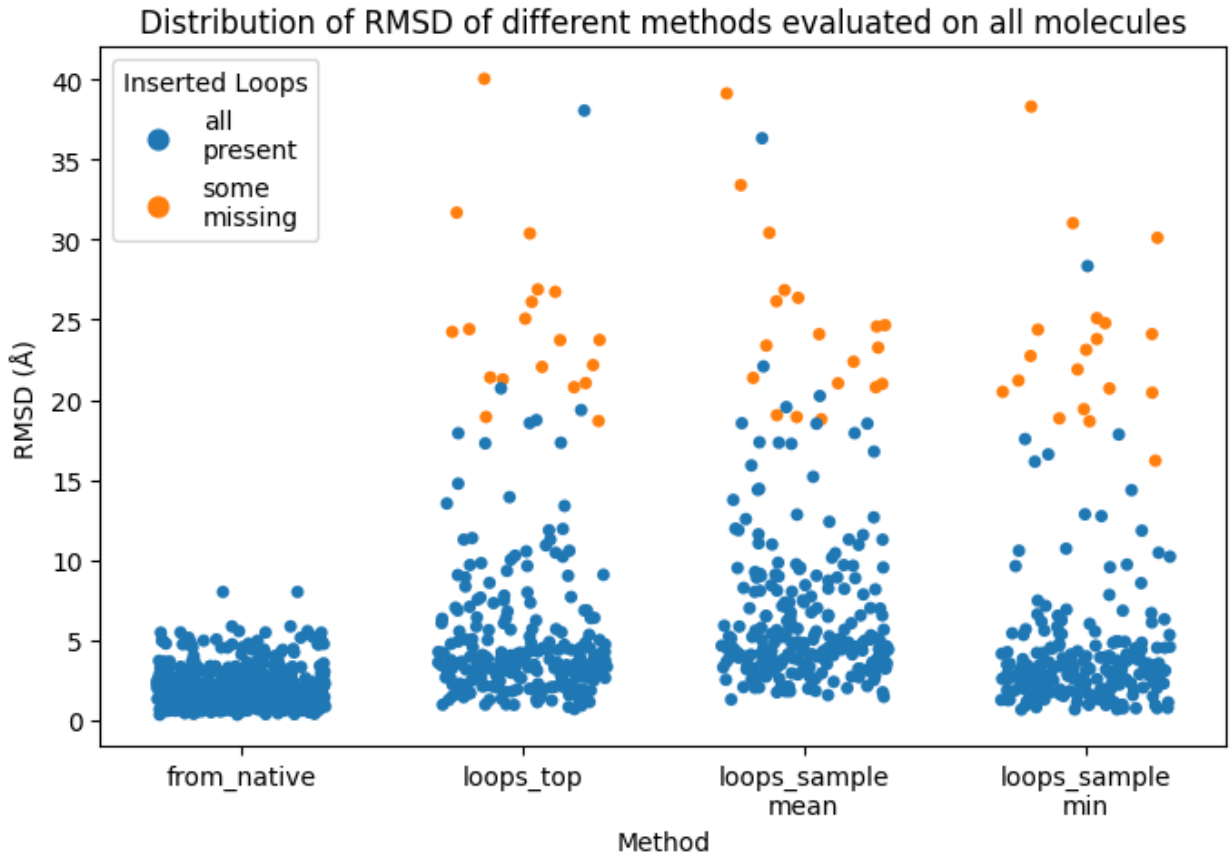


Figure 4.14: Strip plot of distribution of RMSD of different methods evaluated on all molecules. ***from_native*** *assembles molecules from their native fragments.* ***loops_top*** *uses a single combination of fragments predicted by rna_insert_loops.* ***loops_sample*** *uses 50 combinations of fragments sampled by rna_insert_loops for each molecule, and we report the mean and minimum RMSD of the structures built from them.*

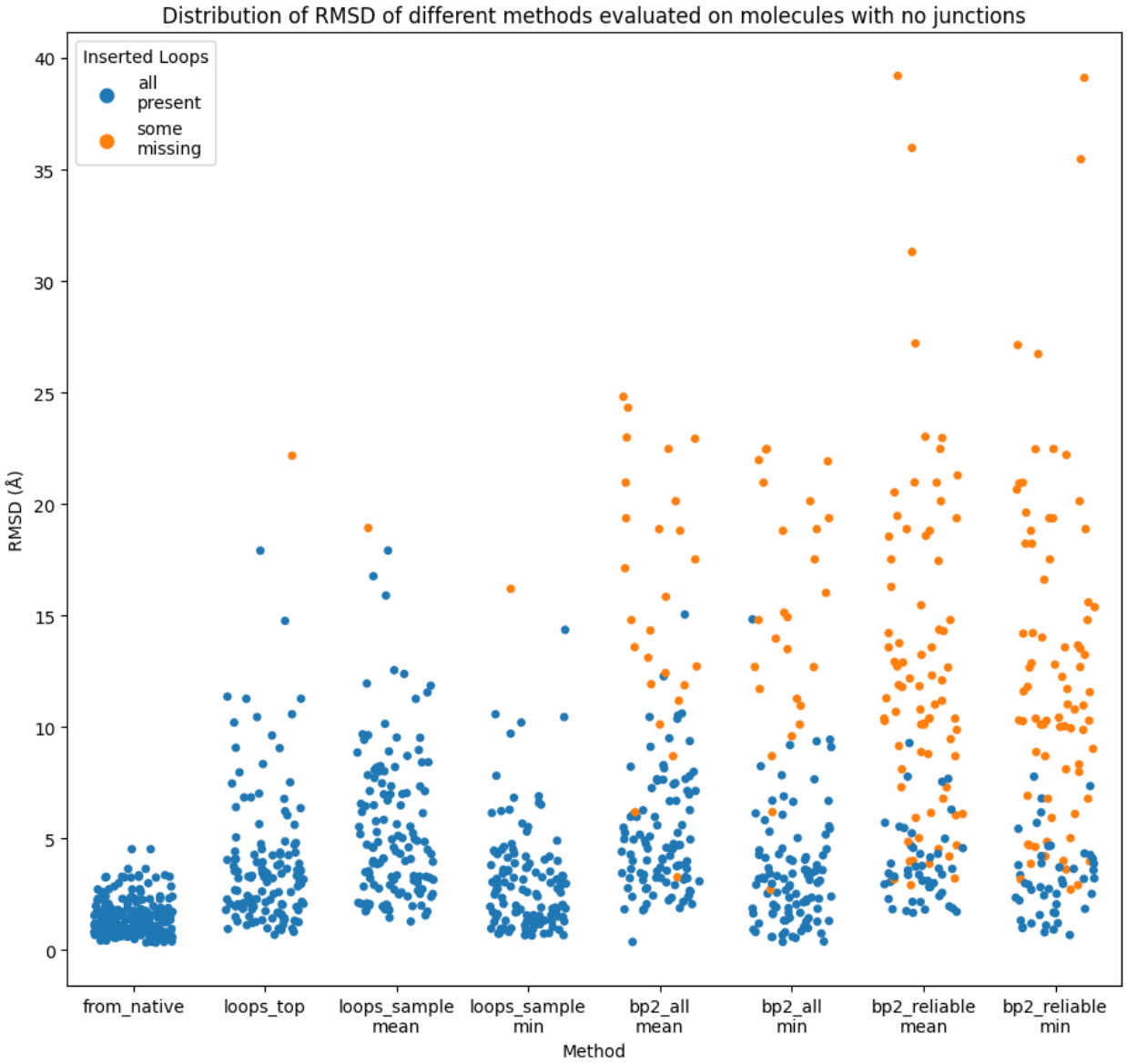


Figure 4.15: Strip plot of distribution of RMSD of different methods evaluated on molecules with no junctions.

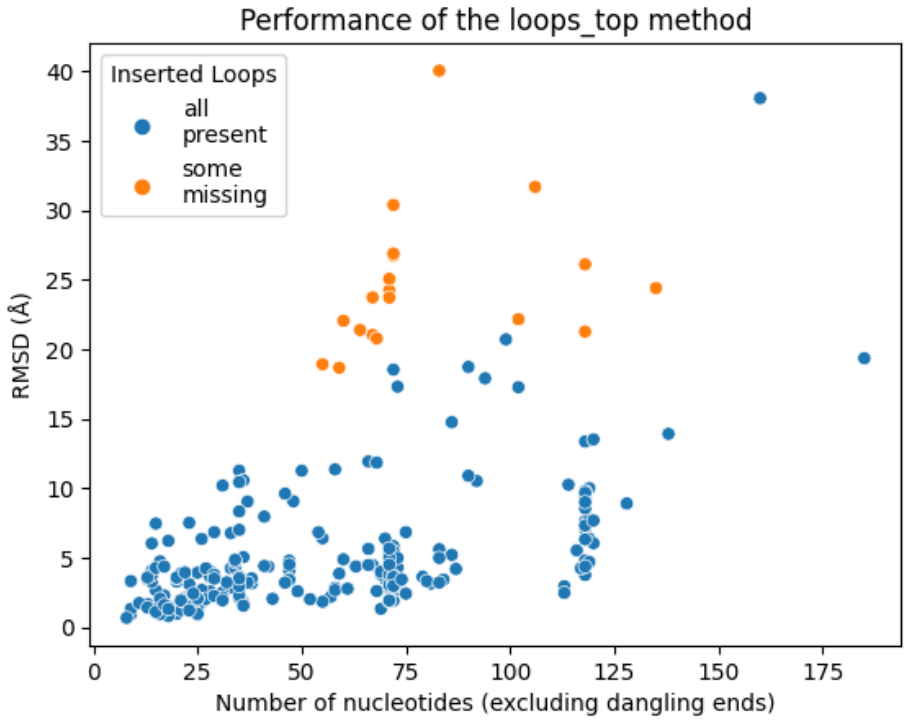


Figure 4.16: RMSD of molecules predicted by loops_top, by size and completeness of loop prediction

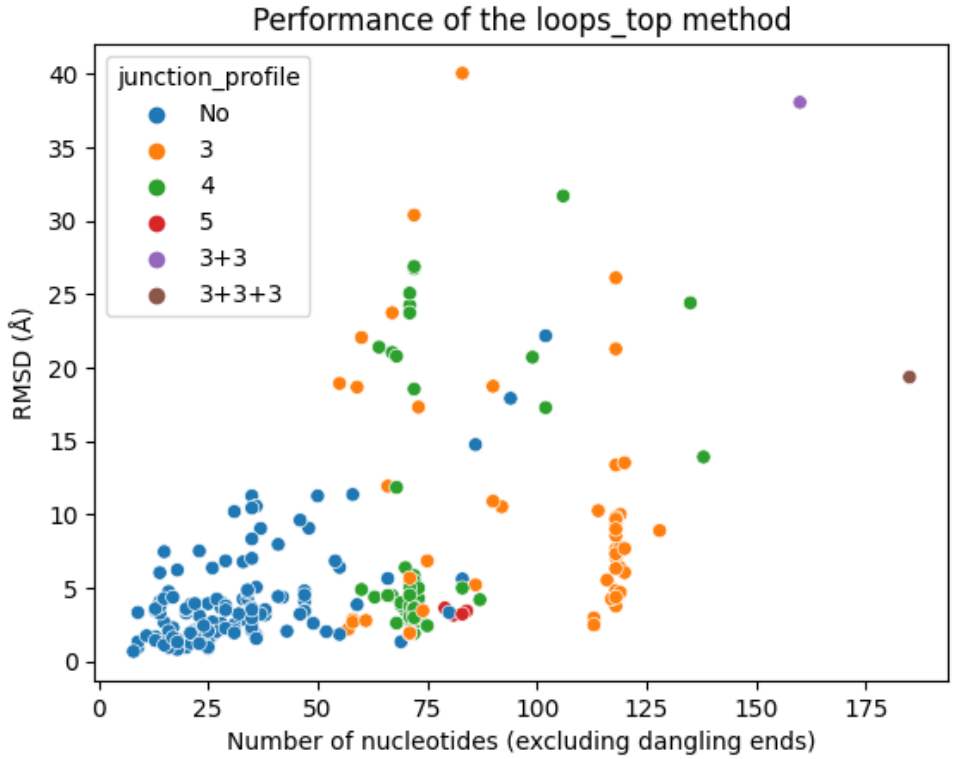


Figure 4.17: RMSD of molecules predicted by loops_top, by size and junction type

		RMSD mean	RMSD var	INF mean	INF var	DI mean	DI var
size	method						
<20	bp2_all	4.087165	3.654663	0.757032	0.012050	5.760914	11.09619
	bp2_reliable	4.040271	3.076777	0.783993	0.008637	5.386865	7.738186
	from_native	0.901799	0.098478	0.937412	0.001738	0.971263	0.129796
	loops_sample	3.728319	4.853984	0.789616	0.012160	5.155899	14.30526
	loops_top	2.640916	3.194002	0.828372	0.012219	3.546629	9.297225
≥20	bp2_all	8.358215	38.05370	0.756310	0.014413	12.57053	138.2587
	bp2_reliable	11.33803	56.26082	0.724101	0.015665	17.62036	200.3582
	from_native	1.576228	0.696043	0.947279	0.000726	1.673877	0.818522
	loops_sample	6.189393	18.47378	0.780234	0.013553	8.778932	52.80642
	loops_top	4.533936	13.44045	0.748364	0.148314	5.666179	38.99729

Table 4.2: Mean and variance for RMSD, INF, and DI over molecules with no junctions, split by size in nucleotides

		RMSD mean	RMSD var	INF mean	INF var	DI mean	DI var
size	method						
<20	from_native	0.901799	0.098478	0.937412	0.001738	0.971263	0.129796
	loops_sample	3.728319	4.853984	0.789616	0.012160	5.155899	14.30526
	loops_top	2.640916	3.194002	0.828372	0.012219	3.546629	9.297225
≥20	from_native	2.234849	1.514805	0.931138	0.001139	2.405420	1.710767
	loops_sample	8.291640	51.18314	0.781163	0.009311	11.41093	119.5261
	loops_top	7.032602	50.29862	0.743305	0.124629	8.996629	120.2419

Table 4.3: Mean and variance for RMSD, INF, and DI over all molecules, split by size in nucleotides

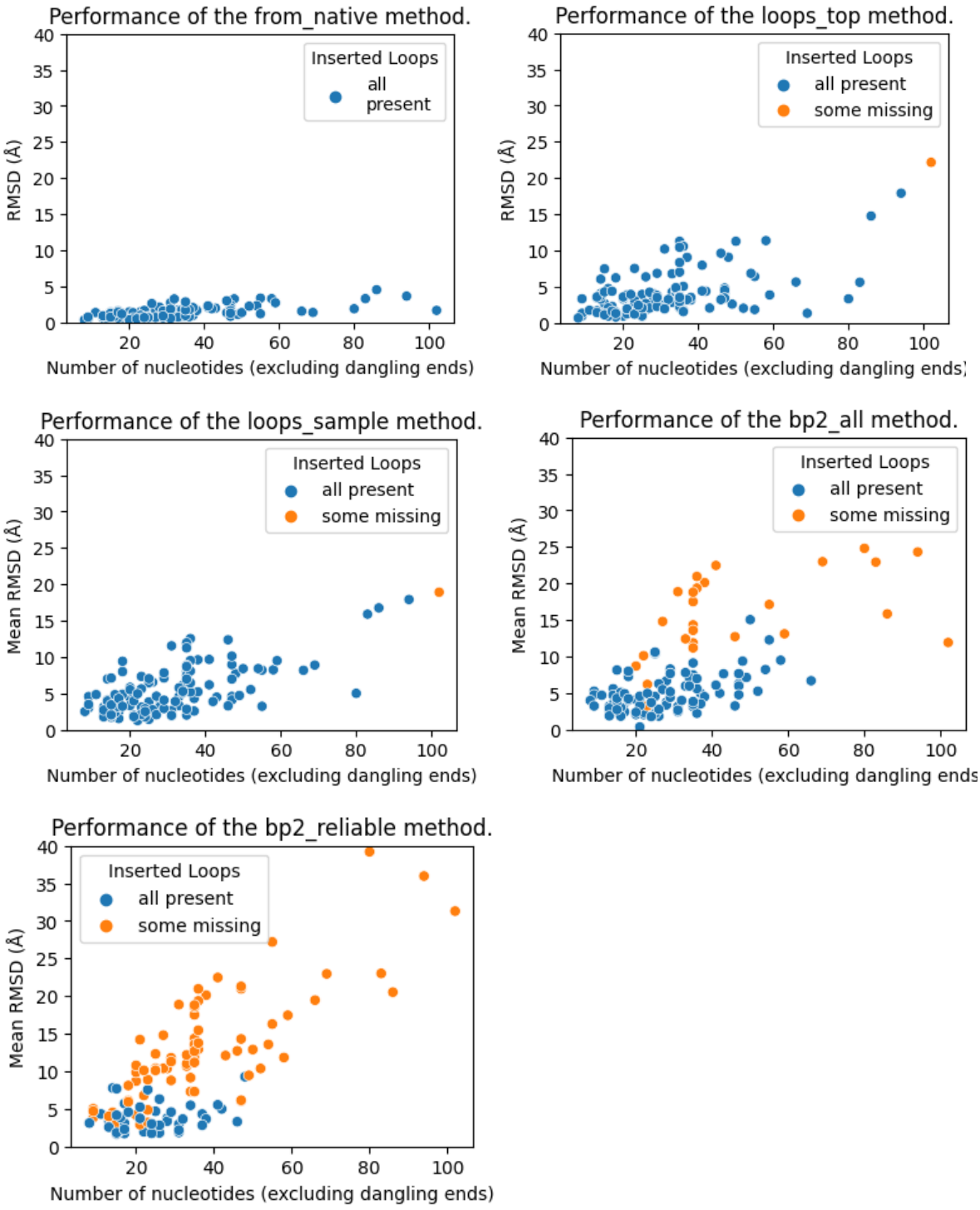


Figure 4.18: RMSD of the different methods on the molecules with no junctions, by size and completeness of loop prediction

Analysis of a few molecules

Lots of junctions: the Varkud satellite ribozyme, NR_4.0_33922.2 (4R4V|1|A)

We look at the results of predicting the structure of the Varkud satellite (VS) ribozyme (NR_4.0_33922.2 (4R4V|1|A)). The structure contains three 3-way junctions and is the biggest molecule that we predicted, with 185 nucleotides.

```
      GCGCUGUGUCGCAAUCUGCGAAGGGCGUCGUCGGCCCAAGCGGUAGUAAGCAGGGAACU
CACCUCCAAUGAAACACAUUGUCGUAGCAGUUGACUACUGUUAUGUGAUUGGUAGAGGCUAAGU
GACGGUAUUGGCGUAAGCCAAUACCGCAGCACAGCACAAGCCCGCUUGCGAGAUUACAGCGC

      ((((((((((.....((((...((((........))))..))))).((((((.((((..((
((((..(((((.....))))))..((((((((.......))))))).....))).)))((((..((
(.((((((((((((....))))))))))...)))))).....))))))))))..))).)))))))
```

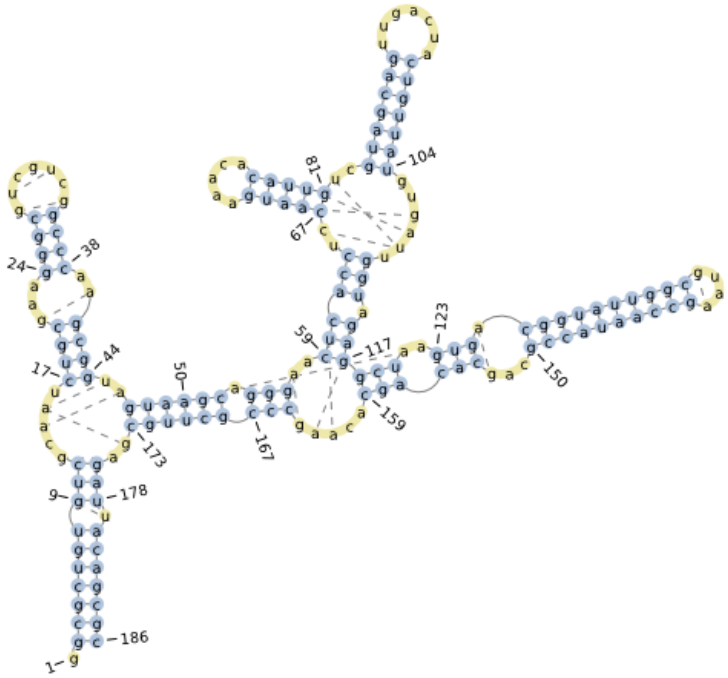
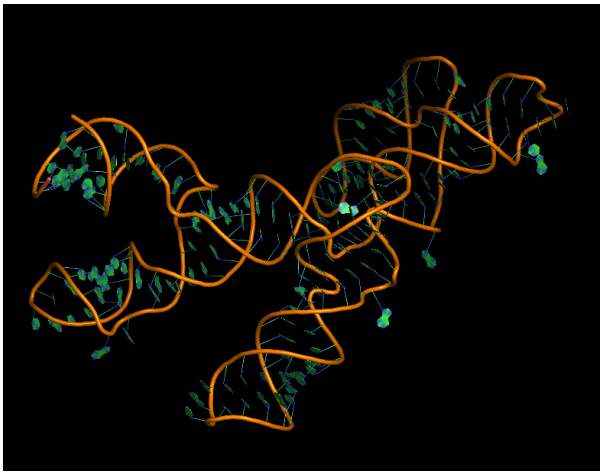
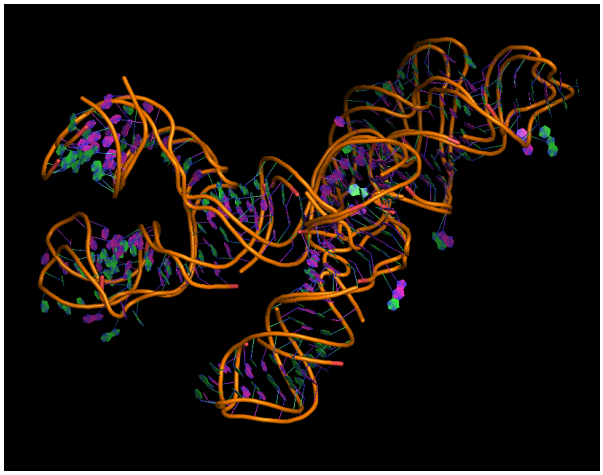


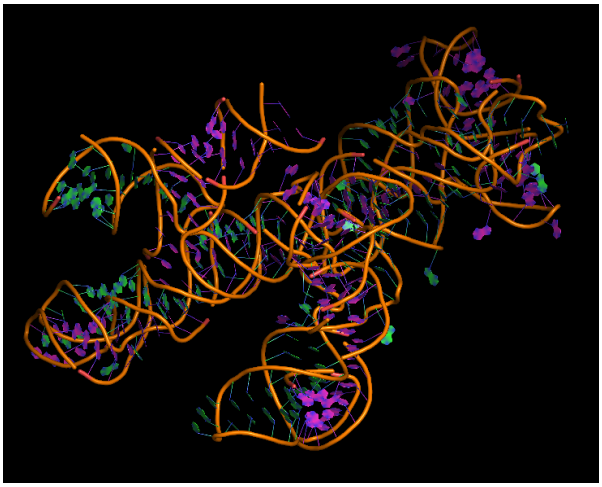
Figure 4.19: Secondary structure of NR_4.0_33922.2 (4R4V|1|A)



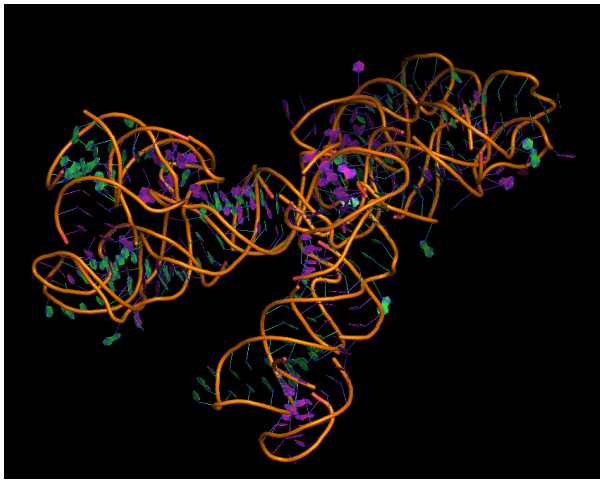
Native



Native and from_native superimposed



Native and loops_top superimposed



Native and loops_sample with the lowest RMSD superimposed

Figure 4.20: Native and predicted 3D structures of NR_4.0_33922.2 (4R4V|1|A)

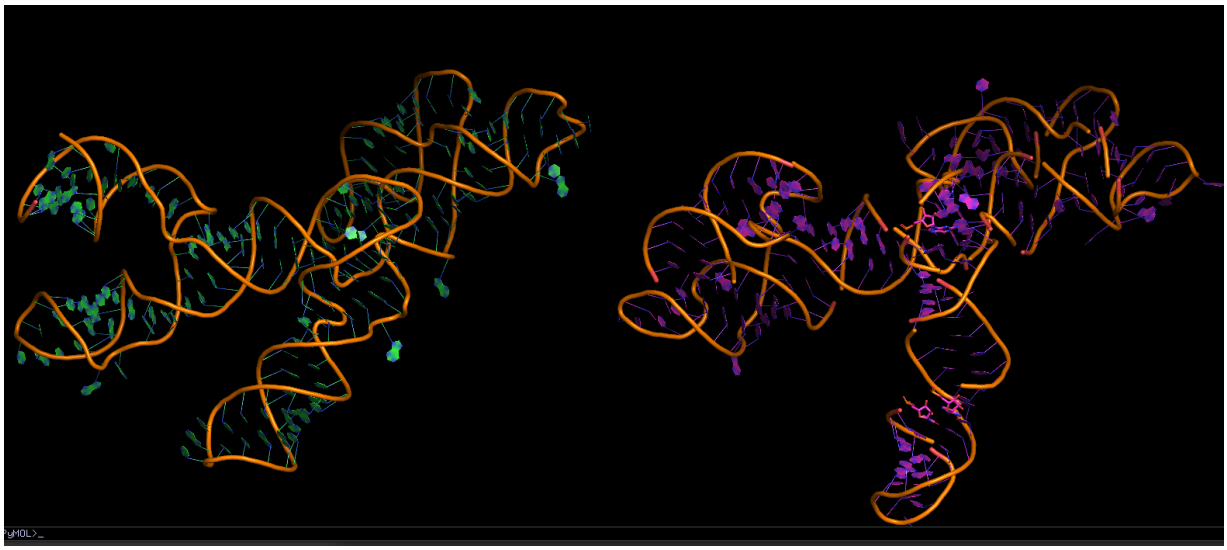


Figure 4.21: Native and loops_sample predicted 3D structure with the lowest RMSD for NR_4.0_33922.2 (4R4V|1|A), side by side.

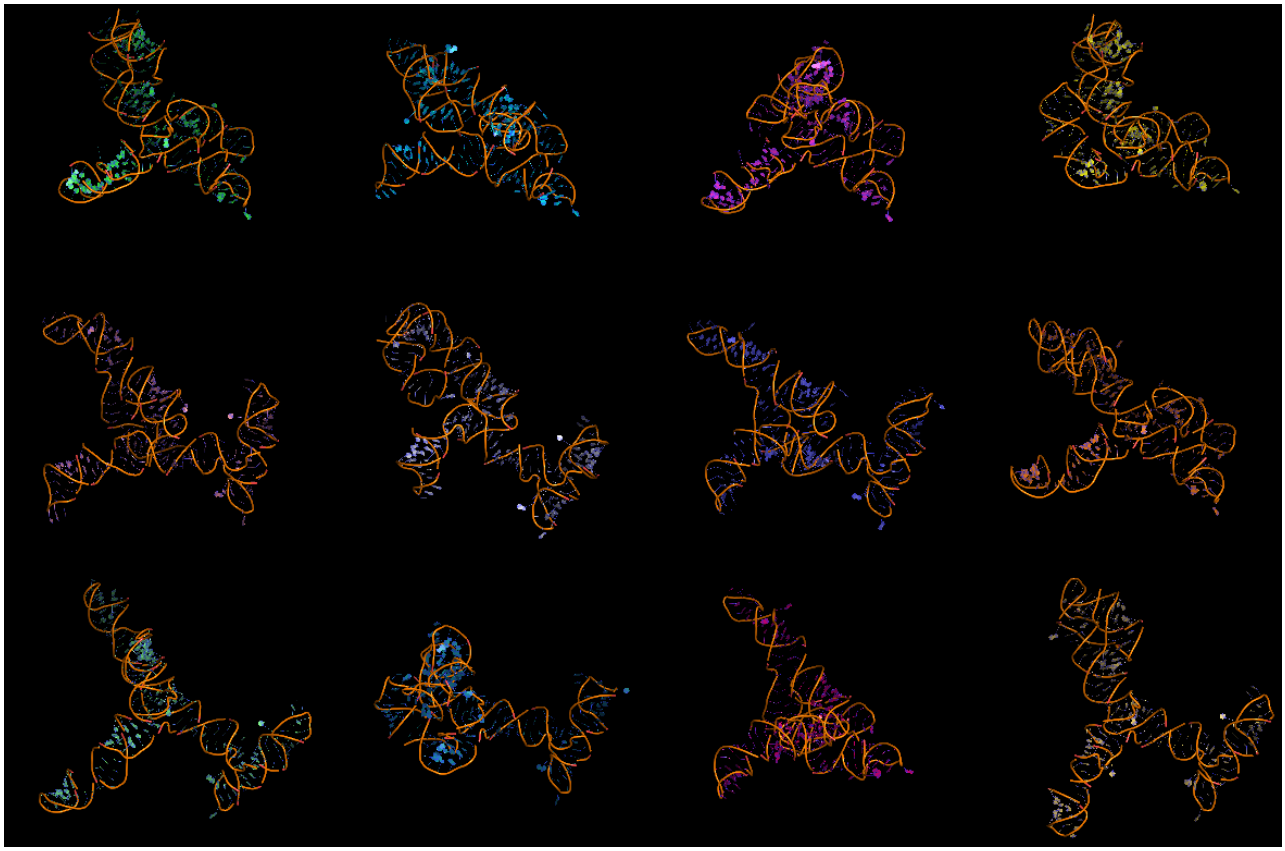


Figure 4.22: Variety of 3D structures sampled by loops_sample for NR_4.0_33922.2 (4R4V|1|A)

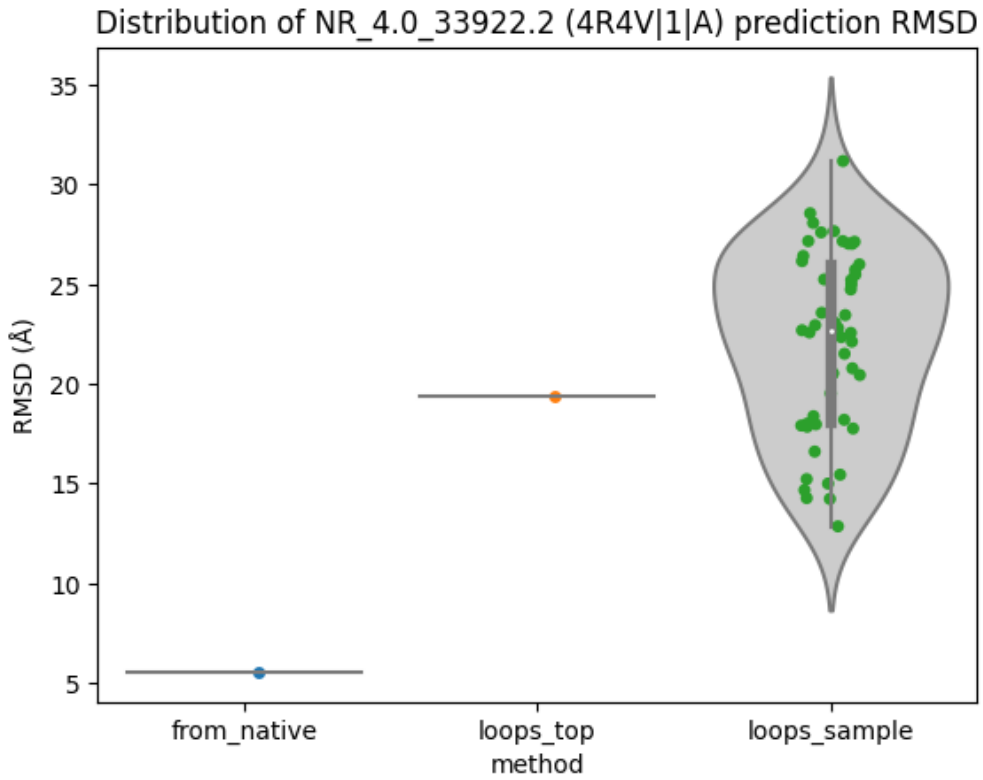


Figure 4.23: Distribution of NR_4.0_33922.2 (4R4V|1|A) prediction RMSD given by different methods

It's interesting that there is enough variety of 3-way junctions in our library that we generated molecules with very different global conformations, as seen in Figure 4.22. The 3-way junctions are:

- "`CGCAAUCGUAGCGAG (.....()..()..)`" which had 5 matches of that shape after excluding the native instance
- "`GAACGGCACAAGC (..()()....)`" which had 2 matches of that shape after excluding the native instance

- “CUCCGUCGUGUGAUUG (..()..()......)” which had 2 matches of that shape after excluding the native instance

Molecule With G-Quadruplex: NR_4.0_02963.1 (5DEA|1|C)

This is a relatively simple molecule if not for the G-quadruplex, that appears simply as a large hairpin loop on the canonical secondary structure:

```
GCUGCGGUGUGGAAGGAGUGGCUGGGUUGCGCAGC
(((((.((.....................)))))))
```

For this molecule, the RMSD really depended on whether the method found a G-quadruplex. loops_top and bp2_all found it, bp2_reliable did not find any motif for the hairpin, and loops_sample sampled all sorts of different motifs including the G-quadruplex. However, looking closer, the motifs that were found were extracted from a practically identical molecule, 5DE8, or from a symmetric chain of 5DEA, or it is a bit cheating. It’s clear that BayesPairing2 is more sensitive than rna_insert_loops since it did not include any motifs that did not exactly match the sequence.

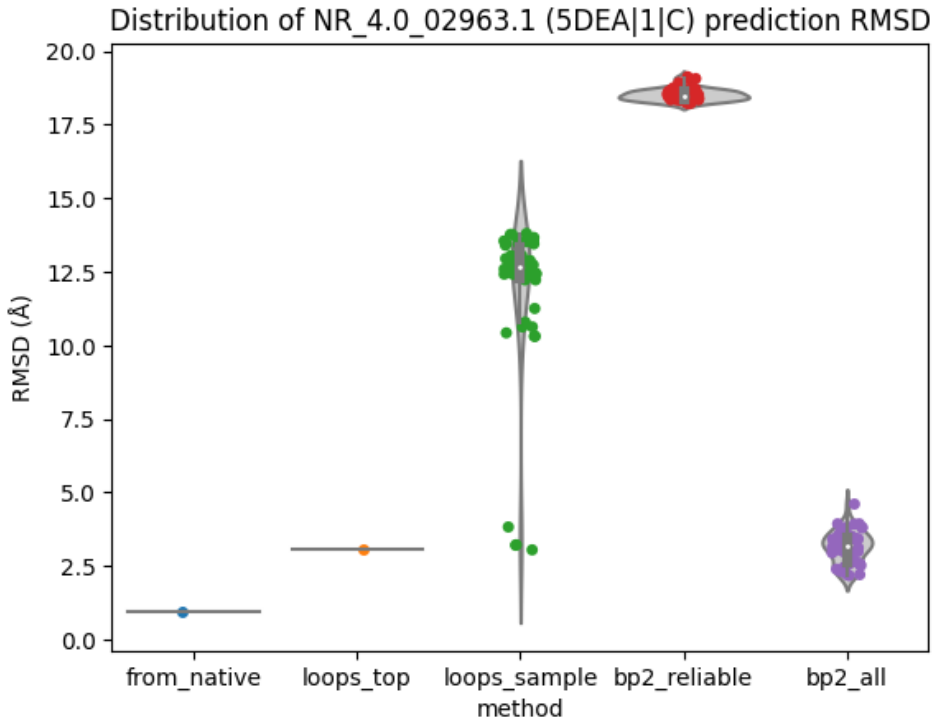


Figure 4.24: Distribution of NR_4.0_02963.1 (5DEA|1|C) prediction RMSD given by different methods

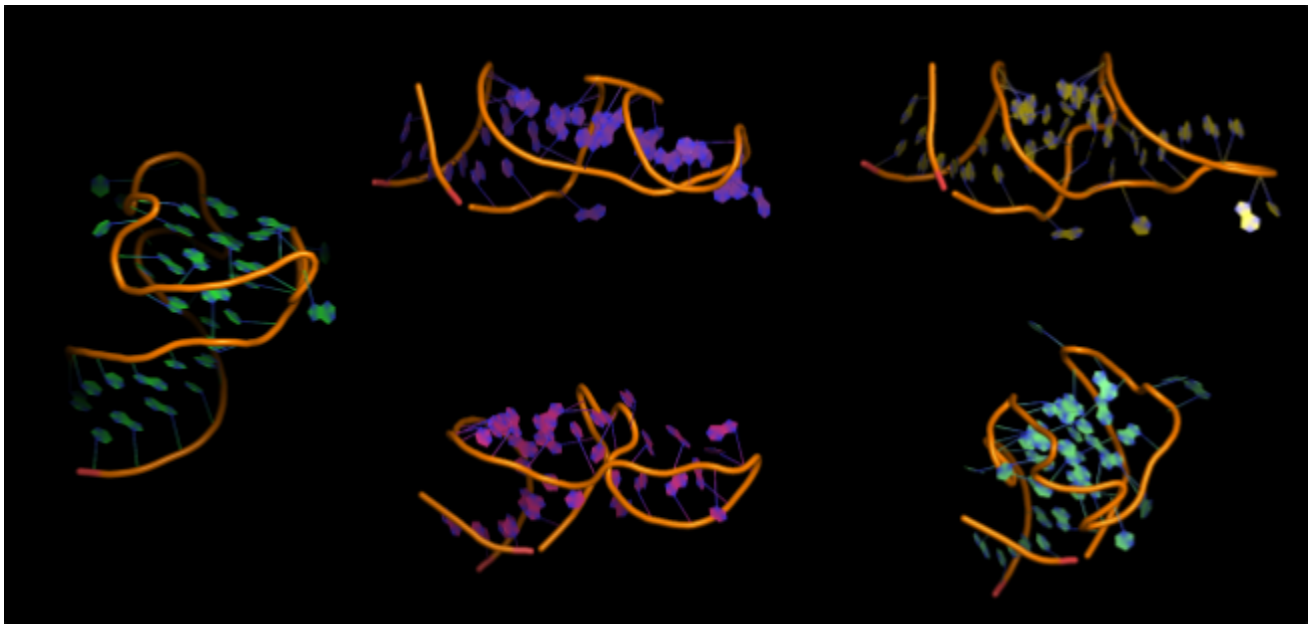


Figure 4.25: Native structure of NR_4.0_02963.1 (5DEA|1|C) and examples of outputs from the loops_sample method

Theophylline Aptamer: NR_4.0_56838.1 (8D29|1|F)

We looked at the prediction results for NR_4.0_56838.1 (8D29|1|F), the theophylline aptamer, as it was one of the molecules for which bp2_reliable found all loops. It has one hairpin loop, one internal loop, and one bulge:

```
GGCGAUACCAGCGAAACACGCCCUUGGCAGCGUC
((((...((((((.....))))...))))...))))
```

There was more variety in bp2_all than there was in loops_sample (Figure 4.26). This is because the loops_sample took only the 10 top loop models for each loop according to its scoring function, whereas bp2_all took all the motifs that passed the threshold for BayesPairing2's scoring function, resulting in 38 motif instance from 12 different motif groups for the hairpin loop, 36 instances from 11 groups for the inner loop, and 4 instances out of 3 groups for the bulge. The numbers of motif instances found by bp2_reliable were respectively: 22 from 3 groups, 59 from 4 groups, and 1 from 1 group.

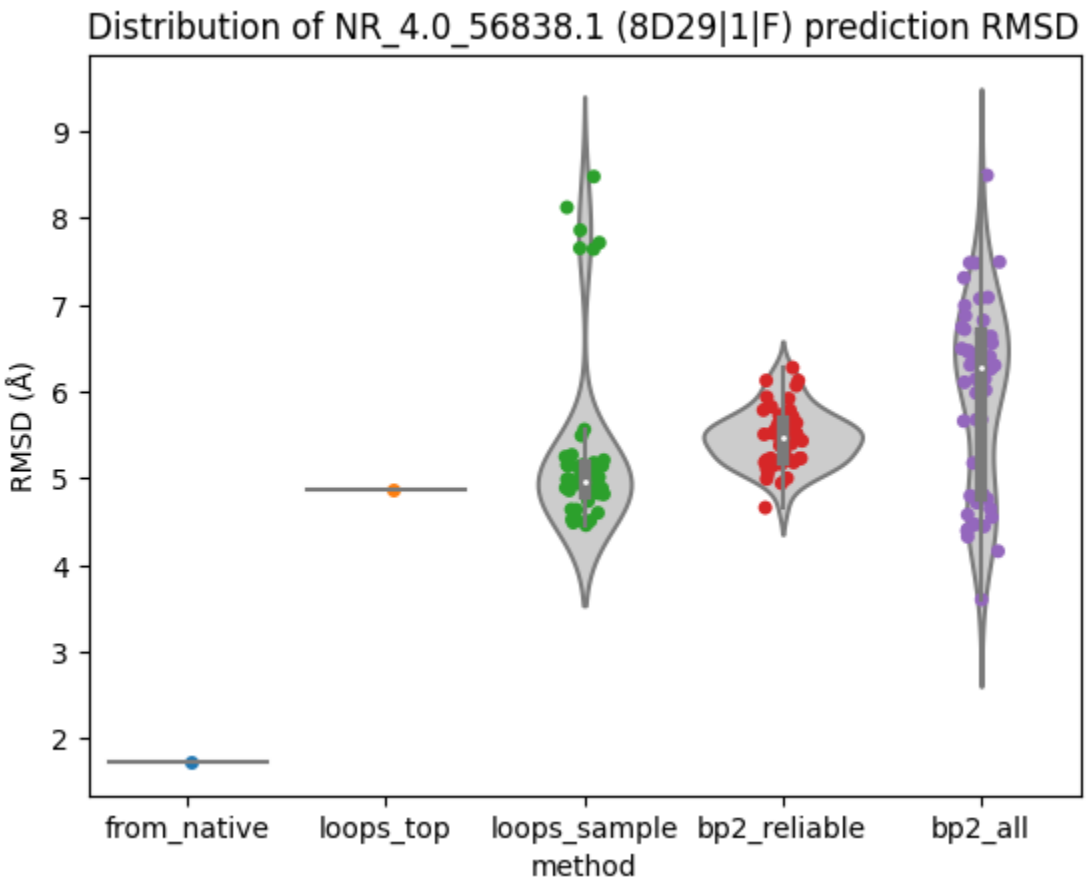


Figure 4.26: Distribution of NR_4.0_56838.1 (8D29|1|F) prediction RMSD given by different methods

To see the effect of the number of selected models on variety, we run rna_insert_loops with different amounts of structures to sample from for each loop: 10 (as in the original loops_sample dataset), 20, 38, and 50; these methods will be called ls_10, ls_20, ls_38, and ls_50 respectively. We note that the RMSD distribution starts to resemble the distribution produced by bp2_all, with a lower minimum RMSD, and more variance (Figure 4.27).

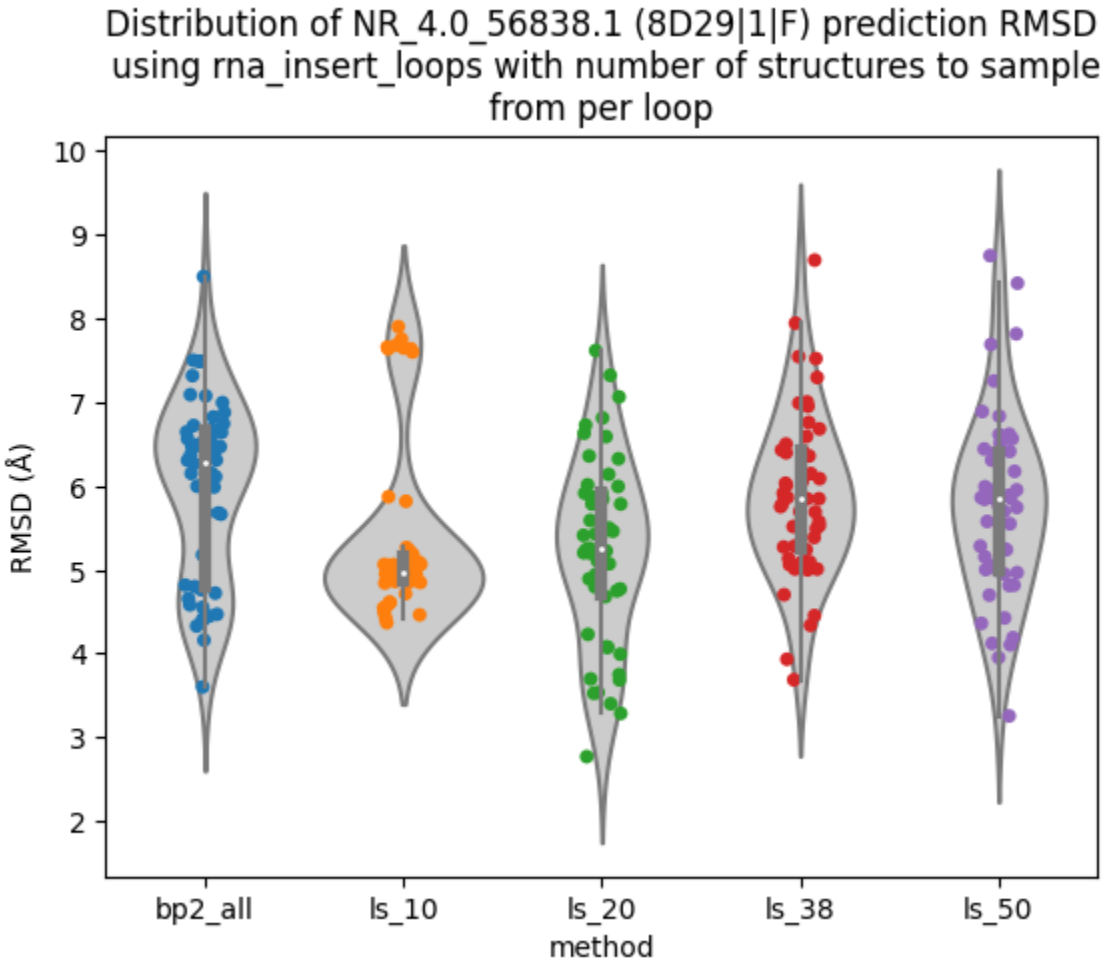
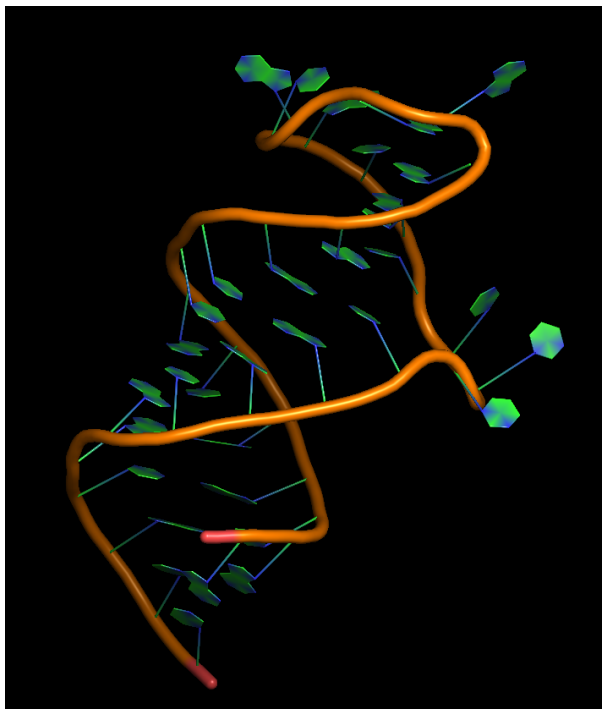


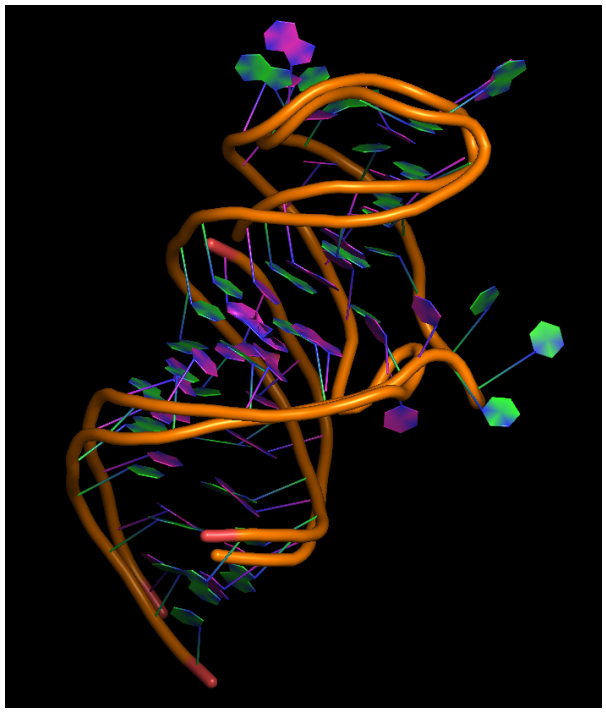
Figure 4.27: Distribution of NR_4.0_56838.1 (8D29|1|F) prediction RMSD given by BayesPairing2 ALL and by rna_insert_loop with different numbers of structure to sample per loop.

Figure 4.28: 3D structure predictions for NR_4.0_56838.1 (8D29|1|F)



Native

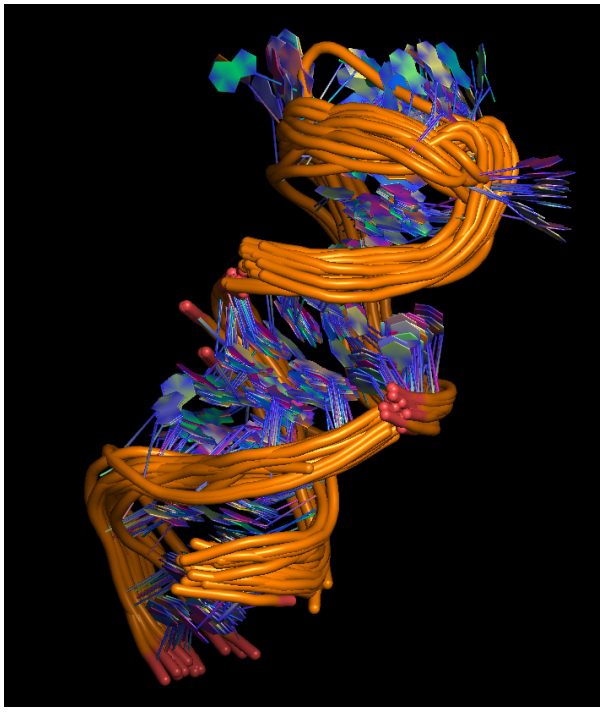
loops_top compared to native, RMSD=4.87



loops_sample compared to native, RMSD=4.46

loops_sample variety

bp2_reliable best compared to native, RMSD=4.66

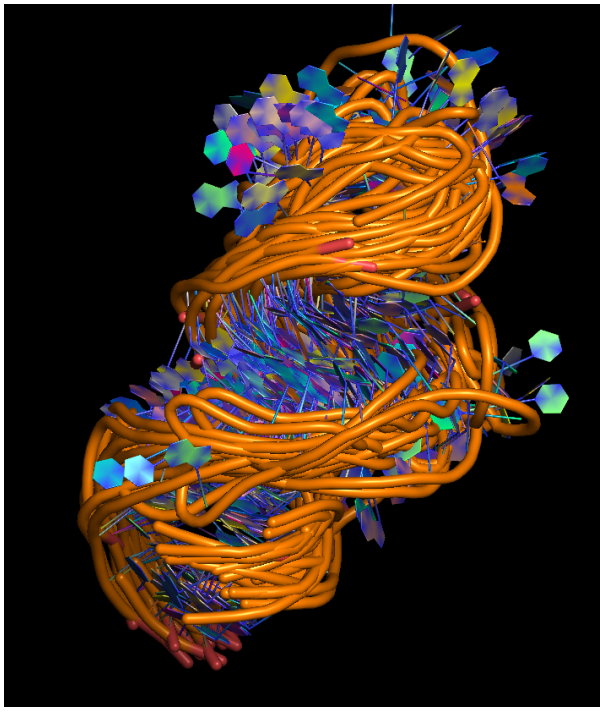


bp2_reliable variety

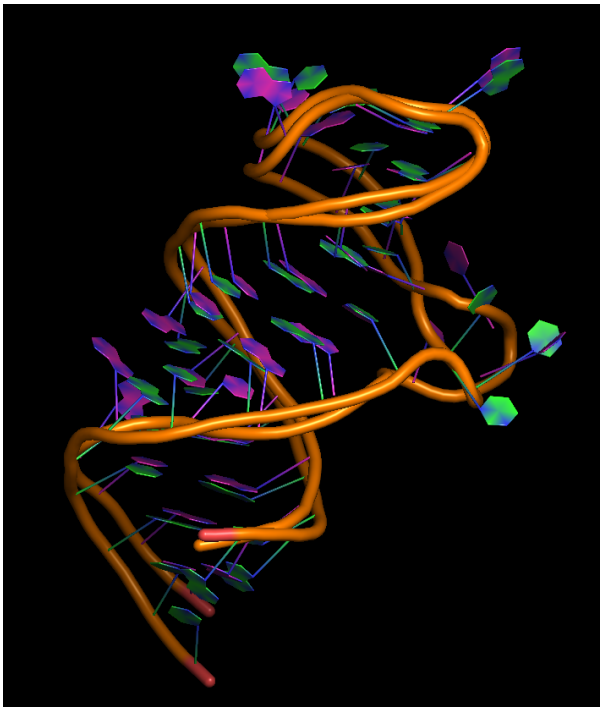
bp2_all best compared to native, RMSD=3.60

bp2_all variety

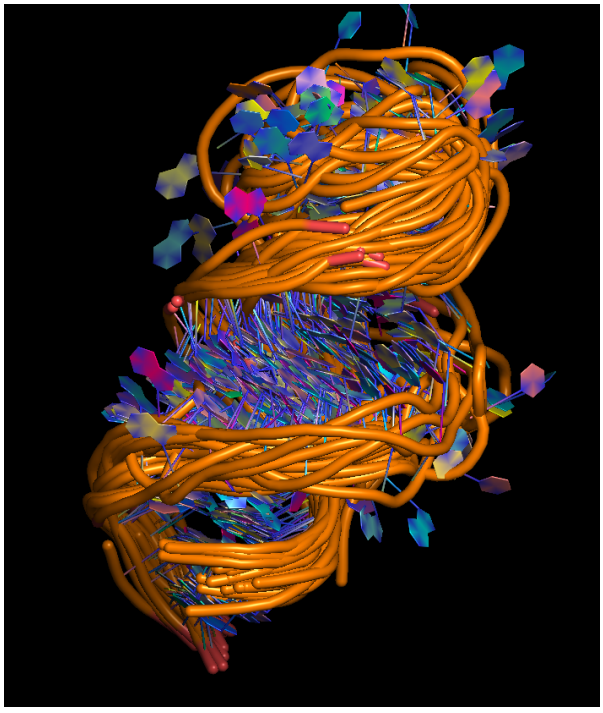
ls_20 best compared to native, RMSD=2.77



ls_20 variety



ls_50 best compared to native, RMSD=3.26



ls_50 variety

5. Discussion

5.1 Main Contributions and Strengths

We created rna_builder, a tool for building RNA 3D structures by assembling 3D models of motifs and stacks. This program works as a standalone command line utility and builds molecules from specification of their sequence and secondary structure, and motif PDB files with insertion positions.

We then recreated a pipeline for 3D structure prediction based on that tool following the paradigm of assembling structures from helices and known loop 3D models. For that we compiled a library of loops and created a tool to insert them into a secondary structure and prepare the input for rna_builder.

We then created a pipeline for 3D structure prediction centered around BayesPairing2. Its novelty lies in the use of the advanced 3D motif prediction algorithm for selecting which loop fragments to use for construction. It is different and potentially better than previous similar 3D prediction methods, because it does not just score loop fragments based on sequence similarity, but takes into account the covariation of the non-canonical base-pairs within loops.

One general strength of our method is its modularity and the simplicity of the format used for communication between the different utilities. It is simple to manually prepare inputs or to write programs that interact with the tools of our method. This may

allow other use cases and methods, perhaps based on completely different principles, to easily integrate with our tools.

5.2 Limitations and Possible Improvements

In general, prediction methods based on the assembly of large components are limited by the fragments they have access to, and our method isn't different, and are best at predicting structures for which there are known homologues. If there isn't a loop motif in our database that is similar to the one contained in the native structure, it is impossible for our method to predict an accurate structure. This is especially a problem for junction loops, for which there is relatively little data, and whose shape will impact the global shape of the molecule.

It is hence important to find ways to increase the amount of candidates for loop insertion. One approach would be opening and closing the base-pairs around loops. Another would be to discover ways to allow deletion and insertions of nucleotides from known motifs, or the merging of multiple motifs into one.

Another limitation is the treatment of situations when nucleotides aren't fully covered by fragments. Our current strategy of placing components on a circle in 3D is very unrealistic and ends up automatically giving a bad RMSD.

Even when all nucleotides were covered by fragments, some model's RMSD was up to 30Å away from the native. Some, but not all, had strong steric clashes, and we could detect that and filter them out. More generally, we could score the models using

an existing RNA scoring function, such as one available in the Rosetta suite (Watkins et al., 2020).

The BayesPairing2 centered pipeline is currently limited to structures that only contain hairpins and inner loops. This could be improved by combining the two pipelines and having rna_insert_loops pick up for regions where BayesPairing2 did not find a motif.

The quality of the 3D assembly can probably be improved, to bring the total RMSD of the from_native dataset closer to zero, and to reduce the amount of chainbreaks as seen in the Pymol visualizations. We could focus on assuring the quality of the helix fragments we use. We also have an hypothesis that changes in sugar-puckering at the loop edges cause changes in nucleotide conformation, causing bad alignments.

Finally, we have only tested our methods on structures with a very restricted secondary structure, and hence a lot of RNA molecules found in the wild were not included.

To summarize the rules of thumb: for this pipeline to work best, the molecule should be single-chained, with no pseudoknots, with limited dangling ends, and it should either have no multi-way junctions, or it should be homologous to a molecule whose structure is known.

5.3 Applications and Future Directions

Our pipeline can be made more complete and closer to other prediction methods by adding a minimization or Monte-Carlo simulation step at the end and a scoring function to rank the created molecules. A further extension of our method could involve secondary structure prediction within the pipeline, and having the feedback from the motif insertion and 3D construction help select the best secondary structure. Another addition to the pipeline could be an interactive interface that allows users to select which motifs they wish to insert.

More work should be done to ensure our method works with structures with less constraints on their secondary structures, such as structures with external loops (possibly by including external loops in the loop insertion step), multiple chains, or pseudoknots.

A further direction to explore would be to use the final global scores to learn which fragments seem to contribute most to construct the best molecules, and then repeat the building step focusing on those fragments, and continue doing so in an iterative cycle.

Another direction would be to integrate our tools with more methods. For example, in the same way we have a script that bridges between BayesPairing2 and rna_builder, we can create a script that bridges between RNA-MoIP and rna_builder, so its predictions could be automatically assembled and evaluated in 3D. Note that since our method integrates with BayesPairing2, it allows us to benefit from future

improvements to the program, such as additions of datasets including junctions, or the integration of RNA-MoIP directly into its “chef’s choice” feature.

Another thing we could evaluate is using rna_builder for use cases other than the prediction of 3D structure from sequence, such as assembling 3D models of modular synthetic RNA such as aptazymes.

Finally, we wish to test this pipeline or a future iteration of it on a blind 3D structure prediction evaluation such as RNA-Puzzles (Cruz et al., 2012) in order to evaluate its true performance.

Supplementary material

Protein Data Bank File Format

We worked with two file formats to represent the 3D structure of RNA: the original Protein Data Bank (PDB) file format, and the newer PDBx/mmCIF file format. We used Biopython to process both formats and, for our use cases, the two formats are semantically equivalent, with the distinction that PDB is older and has restrictions on the size of molecules it can represent, but is supported by more tools.

The organization of the format follows a hierarchical structure of: structure, model, chain, residue, atom.

A single file corresponds to a single *structure*. A structure contains a list of *models*. The indexing of the list might start either at 0 or at 1 depending on the implementation of the tool used to read the file. For example, Biopython has models starting at 0, whereas the indexing in BGSU unit identifiers starts at 1.

Each model will contain one or more *chains*, each having a name. In the original PDB format, the chain names are restricted to one character. Chains conceptually refer to chains of macromolecules such as proteins and RNA, although they also contain molecules that aren't technically part of the chains, such as waters and metal ions. Note however, as shown in the next section, that the correspondence between PDB chains and "real" chains isn't always perfect.

Each chain is composed of *residues*. A residue is a general term for the repeating building blocks that form macromolecules, like amino acids or nucleotides. Other molecules such as ions, water, and small organic molecules, are indicated by special residues within the PDB file format called *hetero residues*. Residues within a chain are indexed by a sequence number and sometimes a 1-character insertion code. Residues will have a residue name, which will be A, U, G, or C in the case of nucleotides.

Each residue is composed of *atoms*. Each atom will have a unique and standardized name (like C3', C4', P, etc.. in the case of nucleotides). Atoms that are part of hetero residues will be marked using the HETATM command in PDB files (instead of ATOM command).

Special Considerations When Processing PDB Files:

Here's a list of considerations to pay attention to when processing PDB files:

Sequence numbers can be negative.

Insertion codes. Other than sequence numbers, some residues have insertion codes in the form of letters. For example, a chain might have residues 20, 20A, 20B, 21.

It's possible that a chain contains a proper residue and a hetero residue with exactly the same sequence number and insertion code.

Some residues have "point-mutations", meaning that there are two instances of a residue with the exact same number and insertion code, and the only thing that

separates them is a different residue name. (I haven't ever noticed this in RNAs however.) This is probably why the BGSU unit ids include the residue name as part of the identifier even though the sequence number and insertion codes are usually enough to identify a residue.

Alternate locations for atoms. Sometimes an atom will have a few different "alternate locations". These will be indexed by a one-character alternate location indicator.

Disrupted chains. A chain might have jumps in the residue sequence numbers. Moreover, even when two residues have consecutive sequence numbers, they might not actually be connected in 3D. A chain in the PDB file isn't necessarily continuous in 3D space.

And the converse: a gap in the sequence numbers does not necessarily indicate a gap in the 3D chain. For example, residues 22 and 24 might actually be connected by a backbone.

There are at least two different standards for naming atoms. For example, the same atom within a nucleotide is called C5* in one standard and C5' in the other. This may cause problems when trying to align structures. The best thing would be to normalize the atom names before processing, and this can be done using a residue and atom rename list as done in https://github.com/RNA-Puzzles/RNA_assessment

It's possible that modified residues that are connected to the main chain nevertheless appear as hetero residues (made up from HETATM atoms).

It’s possible that residues from different chains appear “interleaved”. Biopython’s PDB package does not properly support that, and you might see warnings about it. It will end up just putting the residues in the correct chain, but without taking into account the order of residues within the file.

Bibliography

Batey, R. T., Rambo, R. P., & Doudna, J. A. (1999). Tertiary Motifs in RNA Structure and Folding. *Angewandte Chemie* , *38*(16), 2326–2343.

Bernhart, S. H., Hofacker, I. L., Will, S., Gruber, A. R., & Stadler, P. F. (2008). RNAalifold: improved consensus structure prediction for RNA alignments. *BMC Bioinformatics*, *9*, 474.

Biesiada, M., Purzycka, K. J., Szachniuk, M., Blazewicz, J., & Adamiak, R. W. (2016). Automated RNA 3D Structure Prediction with RNAComposer. *Methods in Molecular Biology* , *1490*, 199–215.

Boniecki, M. J., Lach, G., Dawson, W. K., Tomala, K., Lukasz, P., Soltysinski, T., Rother, K. M., & Bujnicki, J. M. (2016). SimRNA: a coarse-grained method for RNA folding simulations and 3D structure prediction. *Nucleic Acids Research*, *44*(7), e63.

Brion, P., & Westhof, E. (1997). Hierarchy and dynamics of RNA folding. *Annual Review of Biophysics and Biomolecular Structure*, *26*, 113–137.

Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S., & Karplus, M. (1983). CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *Journal of Computational Chemistry*, *4*(2), 187–217.

Cao, S., & Chen, S.-J. (2011). Physics-based de novo prediction of RNA 3D structures. *The Journal of Physical Chemistry. B*, *115*(14), 4216–4226.

Case, D. A., Cheatham, T. E., 3rd, Darden, T., Gohlke, H., Luo, R., Merz, K. M., Jr, Onufriev, A., Simmerling, C., Wang, B., & Woods, R. J. (2005). The Amber biomolecular simulation programs. *Journal of Computational Chemistry*, *26*(16), 1668–1688.

Cheng, C. Y., Chou, F.-C., & Das, R. (2015). Modeling complex RNA tertiary folds with Rosetta. *Methods in Enzymology*, *553*, 35–64.

Cornell, W. D., Cieplak, P., Bayly, C. I., Gould, I. R., Merz, K. M., Ferguson, D. M., Spellmeyer,

D. C., Fox, T., Caldwell, J. W., & Kollman, P. A. (1995). A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. *Journal of the American Chemical Society*, *117*(19), 5179–5197.

Cruz, J. A., Blanchet, M.-F., Boniecki, M., Bujnicki, J. M., Chen, S.-J., Cao, S., Das, R., Ding, F., Dokholyan, N. V., Flores, S. C., Huang, L., Lavender, C. A., Lisi, V., Major, F., Mikolajczak, K., Patel, D. J., Philips, A., Puton, T., Santalucia, J., … Westhof, E. (2012). RNA-Puzzles: a CASP-like evaluation of RNA three-dimensional structure prediction. *RNA* , *18*(4), 610–625.

Das, R., & Baker, D. (2007). Automated *de novo* prediction of native-like RNA tertiary structures. *Proceedings of the National Academy of Sciences of the United States of America*, *104*(37), 14664–14669.

Dawson, W. K., & Bujnicki, J. M. (2016). Computational modeling of RNA 3D structures and interactions. *Current Opinion in Structural Biology*, *37*, 22–28.

Dawson, W. K., Maciejczyk, M., Jankowska, E. J., & Bujnicki, J. M. (2016). Coarse-grained modeling of RNA 3D structure. *Methods* , *103*, 138–156.

Ding, F., Sharma, S., Chalasani, P., Demidov, V. V., Broude, N. E., & Dokholyan, N. V. (2008). Ab initio RNA folding by discrete molecular dynamics: from structure prediction to folding mechanisms. *RNA* , *14*(6), 1164–1173.

Djelloul, M., & Denise, A. (2008). Automated motif extraction and classification in RNA tertiary structures. *RNA* , *14*(12), 2489–2497.

Dokholyan, N. V., Buldyrev, S. V., Stanley, H. E., & Shakhnovich, E. I. (1998). Discrete molecular dynamics studies of the folding of a protein-like model. *Folding and Design*, *3*(6), 577–587.

Fonseca, R., van den Bedem, H., & Bernauer, J. (2016). Probing RNA Native Conformational Ensembles with Structural Constraints. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, *23*(5), 362–371.

Frellsen, J., Moltke, I., Thiim, M., Mardia, K. V., Ferkinghoff-Borg, J., & Hamelryck, T. (2009). A

probabilistic model of RNA conformational space. *PLoS Computational Biology*, *5*(6), e1000406.

Geis, M., Flamm, C., Wolfinger, M. T., Tanzer, A., Hofacker, I. L., Middendorf, M., Mandl, C., Stadler, P. F., & Thurner, C. (2008). Folding kinetics of large RNAs. *Journal of Molecular Biology*, *379*(1), 160–173.

Gendron, P., Lemieux, S., & Major, F. (2001). Quantitative analysis of nucleic acid three-dimensional structures. *Journal of Molecular Biology*, *308*(5), 919–936.

Güntert, P., Mumenthaler, C., & Wüthrich, K. (1997). Torsion angle dynamics for NMR structure calculation with the new program DYANA. *Journal of Molecular Biology*, *273*(1), 283–298.

Hajdin, C. E., Ding, F., Dokholyan, N. V., & Weeks, K. M. (2010). On the significance of an RNA tertiary structure prediction. *RNA* , *16*(7), 1340–1349.

Havel, T. F. (2002). Distance geometry: Theory, algorithms, and chemical applications. In *Encyclopedia of Computational Chemistry*. John Wiley & Sons, Ltd. https://doi.org/10.1002/0470845015.cda018

Höchsmann, M., Voss, B., & Giegerich, R. (2004). Pure multiple RNA secondary structure alignments: a progressive profile approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics / IEEE, ACM*, *1*(1), 53–62.

Hofacker, I. L. (2004). RNA secondary structure analysis using the Vienna RNA package. *Current Protocols in Bioinformatics / Editoral Board, Andreas D. Baxevanis ... [et Al.]*, *Chapter 12*, Unit 12.2.

Jonikas, M. A., Radmer, R. J., Laederach, A., Das, R., Pearlman, S., Herschlag, D., & Altman, R. B. (2009). Coarse-grained modeling of large RNA molecules with knowledge-based potentials and structural filters. *RNA* , *15*(2), 189–199.

Jossinet, F., Ludwig, T. E., & Westhof, E. (2010). Assemble: an interactive graphical tool to analyze and build RNA architectures at the 2D and 3D levels. *Bioinformatics* , *26*(16),

2057–2059.

Kerpedjiev, P., Höner Zu Siederdissen, C., & Hofacker, I. L. (2015). Predicting RNA 3D structure using a coarse-grain helix-centered model. *RNA* , *21*(6), 1110–1121.

Krokhotin, A., Houlihan, K., & Dokholyan, N. V. (2015). iFoldRNA v2: folding RNA with constraints. *Bioinformatics* , *31*(17), 2891–2893.

Kufareva, I., & Abagyan, R. (2012). Methods of protein structure comparison. *Methods in Molecular Biology* , *857*, 231–257.

Laing, C., Jung, S., Kim, N., Elmetwaly, S., Zahran, M., & Schlick, T. (2013). Predicting helical topologies in RNA junctions as tree graphs. *PloS One*, *8*(8), e71947.

Lemieux, S., & Major, F. (2006). Automated extraction and classification of RNA tertiary structure cyclic motifs. *Nucleic Acids Research*, *34*(8), 2340–2346.

Leontis, N. B., Lescoute, A., & Westhof, E. (2006). The building blocks and motifs of RNA architecture. *Current Opinion in Structural Biology*, *16*(3), 279–287.

Leontis, N. B., & Westhof, E. (2001). Geometric nomenclature and classification of RNA base pairs. *RNA* , *7*(4), 499–512.

Leontis, N., & Zirbel, C. L. (2012). Nonredundant 3D Structure Datasets for RNA Knowledge Extraction and Benchmarking. In *RNA 3D Structure Analysis and Prediction* (pp. 281–298). unknown.

Lescoute, A., Leontis, N. B., Massire, C., & Westhof, E. (2005). Recurrent structural RNA motifs, Isostericity Matrices and sequence alignments. *Nucleic Acids Research*, *33*(8), 2395–2409.

Levitt, M. (1969). Detailed molecular model for transfer ribonucleic acid. *Nature*, *224*(5221), 759–763.

Lyngsø, R. B., & Pedersen, C. N. (2000). RNA pseudoknot prediction in energy-based models. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, *7*(3-4), 409–427.

Madison, J. T., Everett, G. A., & Kung, H. (1966). Nucleotide sequence of a yeast tyrosine transfer RNA. *Science*, *153*(3735), 531–534.

Mamuye, A. L., Merelli, E., & Tesei, L. (2016). *A Graph Grammar for Modelling RNA Folding*. *231*(Proc. GaM 2016), 31–41.

Martinez, H. M., Maizel, J. V., Jr, & Shapiro, B. A. (2008). RNA2D3D: a program for generating, viewing, and comparing 3-dimensional models of RNA. *Journal of Biomolecular Structure & Dynamics*, *25*(6), 669–683.

Mathews, D. H., Sabina, J., Zuker, M., & Turner, D. H. (1999). Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of Molecular Biology*, *288*(5), 911–940.

Mathews, D. H., & Turner, D. H. (2002a). Experimentally derived nearest-neighbor parameters for the stability of RNA three- and four-way multibranch loops. *Biochemistry*, *41*(3), 869–880.

Mathews, D. H., & Turner, D. H. (2002b). Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *Journal of Molecular Biology*, *317*(2), 191–203.

Oliver, C., Mallet, V., Philippopoulos, P., Hamilton, W. L., & Waldispühl, J. (2022). Vernal: a tool for mining fuzzy network motifs in RNA. *Bioinformatics* , *38*(4), 970–976.

Parisien, M., Cruz, J. A., Westhof, E., & Major, F. (2009). New metrics for comparing and assessing discrepancies between RNA 3D structures and models. *RNA* , *15*(10), 1875–1885.

Parisien, M., & Major, F. (2008). The MC-Fold and MC-Sym pipeline infers RNA structure from sequence data. *Nature*, *452*(7183), 51–55.

Pasquali, S., & Derreumaux, P. (2010). HiRE-RNA: a high resolution coarse-grained energy model for RNA. *The Journal of Physical Chemistry. B*, *114*(37), 11957–11966.

Petrov, A. I., Zirbel, C. L., & Leontis, N. B. (2013). Automated classification of RNA 3D motifs

and the RNA 3D Motif Atlas. *RNA* , *19*(10), 1327–1340.

Piatkowski, P., Kasprzak, J. M., Kumar, D., Magnus, M., Chojnowski, G., & Bujnicki, J. M. (2016). RNA 3D Structure Modeling by Combination of Template-Based Method ModeRNA, Template-Free Folding with SimRNA, and Refinement with QRNAS. *Methods in Molecular Biology* , *1490*, 217–235.

Popenda, M., Szachniuk, M., Antczak, M., Purzycka, K. J., Lukasiak, P., Bartol, N., Blazewicz, J., & Adamiak, R. W. (2012). Automated 3D structure composition for large RNAs. *Nucleic Acids Research*, *40*(14), e112.

Popenda, M., Szachniuk, M., Blazewicz, M., Wasik, S., Burke, E. K., Blazewicz, J., & 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.

Reinharz, V., Major, F., & 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*(12), i207–i214.

Reinharz, V., Soulé, A., Westhof, E., Waldispühl, J., & Denise, A. (2018). Mining for recurrent long-range interactions in RNA structures reveals embedded hierarchies in network families. *Nucleic Acids Research*, *46*(8), 3841–3851.

Reuter, J. S., & Mathews, D. H. (2010). RNAstructure: software for RNA secondary structure prediction and analysis. *BMC Bioinformatics*, *11*, 129.

Rivas, E., & Eddy, S. R. (1999). A dynamic programming algorithm for RNA structure prediction including pseudoknots. *Journal of Molecular Biology*, *285*(5), 2053–2068.

Rother, M., Rother, K., Puton, T., & Bujnicki, J. M. (2011). ModeRNA: a tool for comparative modeling of RNA 3D structure. *Nucleic Acids Research*, *39*(10), 4007–4022.

Sarrazin-Gendron, R., Yao, H.-T., Reinharz, V., Oliver, C. G., Ponty, Y., & Waldispühl, J. (2019).

Stochastic Sampling of Structural Contexts Improves the Scalability and Accuracy of RNA 3D Modules Identification. In *bioRxiv* (p. 834762). https://doi.org/10.1101/834762

Tinoco, I., Jr, & Bustamante, C. (1999). How RNA folds. *Journal of Molecular Biology*, *293*(2), 271–281.

Townshend, R. J. L., Eismann, S., Watkins, A. M., Rangan, R., Karelina, M., Das, R., & Dror, R. O. (2021). Geometric deep learning of RNA structure. *Science*, *373*(6558), 1047–1051.

Turner, D. H., Sugimoto, N., & Freier, S. M. (1988). RNA structure prediction. *Annual Review of Biophysics and Biophysical Chemistry*, *17*, 167–192.

Waldispühl, J., & Reinharz, V. (2015). Modeling and predicting RNA three-dimensional structures. *Methods in Molecular Biology* , *1269*, 101–121.

Wang, J., Wang, J., Huang, Y., & Xiao, Y. (2019). 3dRNA v2.0: An Updated Web Server for RNA 3D Structure Prediction. *International Journal of Molecular Sciences*, *20*(17). https://doi.org/10.3390/ijms20174116

Watkins, A. M., Geniesse, C., Kladwang, W., Zakrevsky, P., Jaeger, L., & Das, R. (2018). Blind prediction of noncanonical RNA structure at atomic accuracy. *Science Advances*, *4*(5). https://doi.org/10.1126/sciadv.aar5316

Watkins, A. M., Rangan, R., & Das, R. (2020). FARFAR2: Improved De Novo Rosetta Prediction of Complex Global RNA Folds. *Structure* , *28*(8), 963–976.e6.

Xayaphoummine, A., Bucher, T., Thalmann, F., & Isambert, H. (2003). Prediction and statistics of pseudoknots in RNA structures using exactly clustered stochastic simulations. *Proceedings of the National Academy of Sciences*, *100*(26), 15310–15315.

Xu, X., & Chen, S.-J. (2018). Hierarchical Assembly of RNA Three-Dimensional Structures Based on Loop Templates. *The Journal of Physical Chemistry. B*, *122*(21), 5327–5335.

Xu, X., & Chen, S.-J. (2021). Predicting RNA Scaffolds with a Hybrid Method of Vfold3D and VfoldLA. *Methods in Molecular Biology* , *2323*, 1–11.

Zhao, Q., Zhao, Z., Fan, X., Yuan, Z., & Yao, Y. (2020). *Review of Machine-Learning Methods for RNA Secondary Structure Prediction*. http://dx.doi.org/

Zhao, Y., Huang, Y., Gong, Z., Wang, Y., Man, J., & Xiao, Y. (2012). Automated and fast building of three-dimensional RNA structures. *Scientific Reports*, *2*, 734.

Zuker, M. (2003). Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research*, *31*(13), 3406–3415.

Zuker, M., & Stiegler, P. (1981). Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, *9*(1), 133–148.