INFERRING RNA 3D MOTIFS FROM SEQUENCE

James Elwood Roll

A Dissertation

Submitted to the Graduate College of Bowling Green
State University in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2019

Committee:

Craig Zirbel, Advisor

Paul Morris,
Graduate Faculty Representative

James Albert

Maria Rizzo

Junfeng Shang

ABSTRACT

Craig Zirbel, Advisor

An outstanding problem in molecular biology is the prediction of the 3D structure of RNA molecules based on the sequence of the RNA. An important step toward prediction of full RNA 3D structures from sequence is predicting the 3D structures of the non-helical regions, which are often referred to as loop regions. We have developed a methodology for modeling the sequence variability of known RNA 3D loop structures, using data from the RNA 3D Motif Atlas. Our models are stochastic context free grammars (SCFGs) that utilize Markov random fields (MRFs) where necessary. The models are parameterized based on the geometry of the pairwise interactions in the loop 3D structure as well as the sequences that have been observed making the structure in 3D, with the result that a reasonable model can be generated using only one sequence variant observed forming the 3D loop structure. Work has also been done to measure and compare how these sequence variability models overlap in sequence space.

We have developed a software package in which these models for the sequence variability of RNA 3D loop structures can be quickly and automatically generated. The software, called JAR3D, is available on Github for download, and a web server and a command line tool by the same name is publicly available. There are a variety of applications for the JAR3D package. It can be used to align loop sequences to a particular known 3D loop geometry, as well as accept or reject a loop sequence as a viable candidate to form a particular geometry. JAR3D can also be used to address a matching problem: given a novel loop sequence, which known 3D geometry, if any, is the sequence likely to form? This matching problem use case is not addressed by current tools for RNA 3D structure prediction, and is a new addition to the field.

To my father.

ACKNOWLEDGMENTS

I would first like to thank my wife, Nicole, who is a constant inspiration to me. I would also like to thank my mother, whose support and encouragment has always been invaluable to me. I thank Dr. Zirbel for his patience, and for all the time help he gave me. Finally, I would like to thank the members of my committee, Dr. Albert, Dr. Rizzo, Dr. Sheng, and Dr. Morris, for taking the time to read my dissertation and for their invaluable feedback.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

Table Page

CHAPTER 1 INTRODUCTION AND OVERVIEW

1.1 Overview of the modeling problem

This dissertation focuses on the study of RNA molecules and certain statistical problems that arise from the relationship between the nucleotide sequence of an RNA, typically represented by a string of A, C, G, and U letters, and the 3D structure of the molecule. Because all life on earth evolved from a common ancestor, homologous RNA molecules (molecules with a common ancestor, the same 3D structure, and that perform the same cellular functions) can be found in different organisms. However, mistakes made in the copying of genomic sequences have resulted in changes in the sequences of these molecules. This can be viewed as a rejection method for generating random variables, i.e., mutations generate new RNA sequences, and if they cannot form a 3D structure that allows the molecule to function, the organism will die and the new sequence is rejected.

Structured RNA molecules fold back on themselves to form helices and loops. RNA helices are primarily and almost exclusively composed of Watson-Crick basepairs, and so the statistical variability of their sequences is relatively easy to model. See Section 2.3 and Section 3.1 for more information on RNA helices and the prediction of them from sequence. The loops occur between and at the ends of helices. They come in a wide variety of geometries with various non-Watson-Crick basepairs and other interactions. These loops have been found to be recurrent, meaning that they are found in different RNA molecules with the same 3D geometry but often with different sequences. The sequences of a given loop can mutate from one generation to the next but need to maintain their 3D structure and are thus under selection pressure as explained previously. The particular focus of this dissertation is on modeling the loop sequences that can form a given loop 3D structure.

Our approach is to make probabilistic models, based on RNA 3D structures, that describe which sequences are likely to form the same 3D structure, which may occur in different molecules and

different organisms. These structures and the parameters for these models are based on previously studied RNA 3D structures. The source of our RNA 3D loop structures is the RNA 3D Motif Atlas (Petrov, Zirbel, and Leontis, 2013), which collects and clusters RNA 3D loop instances from RNA-containing 3D structures in the Protein Data Bank (Berman, Westbrook, Feng, Gilliland, Bhat, Weissig, Shindyalov, and Bourne, 2000).

1.2 Why model RNA loop sequence variability?

New RNA molecules are being discovered frequently (Kashi, Henderson, Bonetti, and Carninci, 2016). For many of these molecules, only sequence level data will be available, even though their function will likely depend on the 3D structure that they form. Because of this, prediction of an RNA molecule's 3D structure from its sequence is an important problem in the study of RNA. Predicting the geometry of loops is an important step on the way to predicting the full 3D geometry of an RNA molecule from its sequence. The process of predicting RNA 3D structure from sequence, and existing approaches to the problem, is explained in more detail in Chapter 3. Our models for RNA loop sequence variability allow us to address several problems in the study of RNA. Our work addresses three problems in particular.

The first is an alignment problem; that is, given an RNA loop sequence and a 3D loop structure, find the most likely way that the sequence forms the 3D structure; this is called an alignment of the sequence to the 3D structure. By aligning multiple sequences to the same 3D structure we can also create sequence to sequence alignments. Sequence to sequence alignments, also called multiple sequence alignments, have a long history in the study of biological molecules and have many uses, such as giving insights into genetic diseases. Current techniques for creating RNA multiple sequence alignments often perform poorly in loop regions, so a tool that can improve multiple sequence alignments in loop regions is desirable.

The second problem is a recognition or acceptance problem; in which we wish to recognize novel sequences that can form a particular known loop 3D structure. Several other tools that model RNA sequence variability are designed to address this problem. The typical use case here is to scan genomes for instances of an RNA loop structure or other small RNA substructure. JAR3D is not

designed to scan genomes, but it can be used to accept or reject a sequence or a set of sequences as candidates to form a particular loop geometry.

Finally, the third and most difficult problem that we address is a matching problem; given a sequence or a group of RNA loop sequences, match them to which, if any, known 3D structure or structures they can form. In our work, we only match to RNA loop structures in the RNA 3D Motif Atlas, but the RNA 3D Motif Atlas is constantly adding new structures, so the number of structures that we can match to will grow over time. The matching problem is not really addressed by other available tools. Other tools either are made to scan genomes, and only consider one geometry at a time, or to do de-novo prediction through fragment assembly and molecular dynamics, and what is gained in the ability to predict new geometries is lost in accuracy and run time. This means our models and methods offer something new and unique to the field of RNA 3D structure prediction.

1.3 Challenges to overcome in modeling RNA loop sequence variability

There are a number of issues that make modeling the sequence variability of RNA 3D loops difficult, many of which arise from the data we have to work with. There are many RNA containing 3D structures in the PDB, and the number available is constantly growing. Many loop geometries have only been observed once in 3D structures, with many more that have been observed less than 10 times. This is not enough to completely model the numerous possible sequence variants through traditional statistical methods. Thus, the first difficulty with modeling the sequence variability of RNA loops is that we have few 3D instances of these loops, and the instances that we do have do not show much of the full range of possible sequence variability.

One possible source of additional data for modeling sequence variability is sequence alignments: data structures in which sequences of the same RNA molecule from many organisms are mapped to (aligned to) a sequence that has been observed in 3D. There are some issues with using sequence alignments to parameterize our models, however. Firstly, sequence alignments are often unreliable, especially in loop regions. Secondly, even very reliable hand curated alignments are for a particular location in a particular RNA molecule. This will make models parameterized from alignments biased towards sequences that work for that particular instance of the loop, which may

have constraints particular to the context in which it occurs. We want to be able to detect new sequences that can form the same 3D geometry in novel settings, for example, in a newly-discovered RNA, so we have decided not to use sequence alignments to parameterize models. Note, however, that in the future, our models can be used to improve the multiple sequence alignments and remove sequences which are unlikely to form the 3D structure in question, and then one could use these improved alignments to build better models for sequence variability.

A problem that compounds the difficulty of the relatively small amount of data available is the sheer number of loop structures to be modeled. We used Motif Atlas release 1.13 for our research, which has 278 internal loop groups and 253 hairpin loop groups. These are large numbers, making it impossible to construct each model by hand, or even to validate each model by hand. Furthermore, the number of motif groups will only grow over time as new RNA 3D structures are solved. Because of this, it is not feasible to have a modeling system that requires the models to be parameterized by hand. An automated pipeline is needed if we want our collection of models to be as up to date and as useful as possible.

The number of 3D structures to be modeled limits our ability to make use of other popular methods for the prediction of molecular 3D structures, such as molecular dynamics and physics based approaches. Imagine taking an input sequence and simulating its ability to form a particular geometry. Each molecular dynamics or physics based calculation would likely take hours, or at least a number of minutes (Kruse, Havrila, and Sponer, 2014). Multiply this by the over 200 structures that would need to be tried, and it will easily take a number of hours or even days to get results on what 3D structures the sequence can form. We want results to be available in a matter of seconds, so these other methods are simply too slow and computationally intensive to be applied to such a broad problem in a reasonable time frame.

Many of the RNA loop structures that we want to model have complex networks of interactions and dependencies. RNA helices have comparatively simple stacked interactions that are much easier to model. To make computationally efficient models that take into account the complex networks of interactions that can be found in RNA loops we developed new techniques, which are

discussed in Chapter 4.

There is also a considerable issue with false positives for the matching problem. With over 250 motif groups for both hairpin and internal loops, a system with a low threshold to match a sequence to a given motif group can easily generate false positive matches when one sequence is scored against all motif groups. In addition, it is possible that a new RNA loop sequence has a 3D structure that has not been seen before, and has no matching group. Finally, it is possible that a single RNA loop sequence can form multiple 3D structures depending on context, so the sequence to 3D structure matching problem may also have multiple correct answers.

1.4 Nature of the models

The models we make for the sequence variability of RNA internal and hairpin loops are probabilistic models that are based on the sequence, geometry, and interactions found in the 3D structures in each motif group. The models are stochastic context-free grammars (SCFGs) that make use of Markov random fields (MRFs) when necessary. SCFGs work well for modeling nested interactions, which allows for most RNA loops to be modeled, due to the way RNA molecules fold back on themselves, which creates stacks of nested interactions. RNA and RNA structures are discussed in depth in Chapter 2. Both the generation and application of pure SCFG models are fast and efficient. A significant number of RNA loops have more complex networks of interactions, and we can model these loops by inserting MRFs into the SCFG models for these groups. Models using MRFs allow us to model many loops which we could not otherwise model.

The SCFG/MRF form of these models was first outlined in Michael Sarver's dissertation (Sarver, 2006). Sarver described the basic form of the SCFG/MRF models and discussed how such models could be used to create multiple sequence alignments based on the 3D structure of a complete RNA molecule. He also outlined a MLE method for parameterizing the models using multiple sequence alignments, assuming that very high quality alignments would exist.

Here, we envision a new use case for these models, modeling the sequence variability of individual RNA 3D loops. Because high-quality alignment data is not generally available for loop regions, we base the parameterization on the 3D geometry of the loop, the various RNA-RNA

interactions in the loop, and known data on the sequence variability associated with those interactions. The majority of my work centers around new methods for parameterizing these SCFG/MRF models for modeling sequence variability of RNA loop geometry. The underlying form of the models has also been updated relative to Michael Sarver's models in some cases, to better reflect our current understanding of RNA structures and the needs of modeling RNA 3D geometries.

The parameters for these models are assigned using a method which uses a strong prior based on the geometry of the motif and observations of all RNA 3D structures. Sequences from 3D structures alone cannot be the only basis for assigning probabilities due to the low number of 3D instances available for the majority of the motif groups. Using the geometry of the motif to assign a strong prior allows us to make models with only a single instance from 3D structures. We use geometry through a concept called isostericity, which is explained in Section 4.2. This "prior" parameterization is based on real and extensive studies of the geometries of RNA molecules in solved RNA 3D structures, and can make effective models even with only one observed 3D instance. After this geometry-based prior is calculated, it is then updated based on the specific sequences observed in 3D structures. The more observed 3D instances a group has, the more the final combined parameters are weighted towards observed data.

SCFGs have been used before in the study of RNA, specifically for generating sequence alignments based on the 2D structure of RNA molecules (Rivas and Eddy, 2001; Pedersen, Bejerano, Siepel, Rosenbloom, Lindblad-Toh, Lander, Kent, Miller, and Haussler, 2006). RNA 2D structures indicate which nucleotides are involved in helices. More information on RNA 2D structures can be found in Chapter 2. Infernal (Nawrocki, Kolbe, and Eddy, 2009) is a tool for modeling RNA 2D structures and searching genomic databases for sequences that likely encode homologous RNA structures. Infernal uses guide tree SCFGs that are very similar to those used in this work. SCFGs and Infernal are explained further in Section 3.4 and Section 3.5.

MRFs have not previously been used in the study of RNA, but Bayesian networks have been used to overcome the problem of complex interaction networks in a similar way by a tool called RMDetect (Cruz and Westhof, 2011). RMDetect trains Bayesian network models on heavily cu-

rated sequence alignments of certain RNA loops, with the intent to search for new instances of those loops in other RNA sequence alignments that are not tied to a full 3D structure, as well as improve RNA 2D structures and assist in the assembly of full RNA 3D models. A comparison of results from our method to RMDetect is given in Section 6.4.

1.5 Results

The main results of the dissertation include a new way of modeling sequence variability of RNA loops based on pairwise interactions between nucleotides seen in 3D structures, improvements in the SCFG/MRF formalism introduced by Sarver (Sarver, 2006), new criteria for the recognition problem which control the false positive rate, some first results on the matching problem described above, and new techniques for describing the overlap between the probabilistic models in sequence space. In addition, solutions to the alignment, recognition, and matching problems have been implemented in a software package which is available for download from Github and which is also available as a web server which has many helpful features for displaying the output.

We have developed a software suite that implements our method for modeling RNA loop sequence variability, **J**ava-based **A**lignment of **R**NA using **3D** structure, or JAR3D. One part of the system makes new JAR3D models through an automated process, based on the motif groups in a release of the RNA 3D Motif Atlas. These models take into account observed sequence variability and geometric constraints from basepairing and base-backbone interactions. The second part of the system scores sequences against one or more JAR3D models, optionally making an alignment to a JAR3D model.

Two papers have been published on the basis of this work. The first, for which I was one of six authors, outlines how the SCFG/MRF models are made, describes the software package JAR3D, and tells how JAR3D can be used for identifying new variants of known motifs (Zirbel, Roll, Sweeney, Petrov, Pirrung, and Leontis, 2015). This was the culmination of over 10 years of work in the BGSU RNA group. The second paper, for which I was the lead author, is about the JAR3D Web Server tool that I led the development of (Roll, Zirbel, Sweeney, Petrov, and Leontis, 2016), to make JAR3D more accessible to those studying RNA. The web server allows a user to input one

or more RNA loop sequences, see the best-matching motif groups, know whether the sequence is recognized by each motif group or not, and optionally view the alignment of the sequence(s) to the motif group.

I will now highlight results in which I played a significant role. First and foremost, I worked on the modeling of RNA sequence variability based on instances and interactions observed in the RNA 3D Motif Atlas. I improved how we convert the differences in basepair geometries into substitution probabilities, specifically the function that we use to invert isodiscrepancy scores. I observed that deletion probabilities for the nodes in our models where not high enough, which was creating too much overlap in the models, and made sure they were appropriately increased. I designed and implemented modeling procedures to make our parameters more data driven, when the data is available. This work shows up in how we set basepair substitution matrices when we have multiple instances of a loop, and how we parameterize variable length insertions. I recognized the need for hairpin nodes to implement Markov random fields, as well as the need for Markov random fields to be normalized so that probability scores can be compared across models. I also worked on using independence to make calculating the normalization constant for large Markov random fields feasible. These updates to our modeling methodology for RNA loop sequence variability are outlined in Chapter 4.

I also made major contributions to the false positive control we use for JAR3D. I figured out how to properly generate distractor sequences to use in our false positive control studies; these sequences mimic RNA loop sequences without necessarily fitting a particular 3D geometry. I also suggested combining information from alignment score deficit and edit distance, which led to our creation of the acceptance region and cutoff score. These contributions and methods are discussed in more detail Chapter 6.

I also did much of the work in converting code that was previously written for full RNA sequences to a form that works for RNA loops. This includes Java code for JAR3D that Meg Pirrung developed and Matlab code written primarily by Craig Zirbel and Michael Sarver. This code now builds models based on Motif Atlas releases automatically as part of a pipeline, so the models

are updated based on the latest RNA 3D structures. As of the writing of this dissertation, there are JAR3D models available for 20 releases of the Motif Atlas, with the latest, 3.2, taking advantage of a major update to the Motif Atlas. The latest release has nearly 300 internal loops and 300 hairpin loops. The source code for JAR3D is posted online through Github and is easily accessible, see https://github.com/BGSU-RNA/JAR3D. Many updates to the JAR3D code were needed to make this possible. I also added functionality to the Java code that parses sequences against JAR3D models to check internal loops with both strand orientations, functionality which is needed when parsing loops by themselves, but wasn't needed for previous uses of JAR3D.

JAR3D can be used through a command line tool and web server, both of which I took a primary role in developing. The web server has a number of user friendly features, such as the ability to parse a variety of input formats, select the release of the Motif Atlas to use, and view 3D models of instances of the motifs. A very significant feature is the ability to align input sequences to a motif group of interest, and to see exactly where the input sequences differ from the most similar 3D sequences. I also developed a command line Java jar file, which can be run locally and thus is more suitable for large scale analysis. The command line tool offers flexibility and fast, efficient processing of larger alignments and batch jobs.

I also developed tools to help us understand the JAR3D models from a probabilistic standpoint. This work is discussed in Chapter 7. JAR3D models can be viewed as probability distributions over the space of all possible sequences. We wanted to be able to compare JAR3D models based on these probability distributions to be able to identify those which are similar in sequence space but might not have similar basepairing patterns. JAR3D models assign probabilities to so many sequences that existing methods for comparing distributions were not feasible. I have developed a dissimilarity measure and a graphical comparison tool based on rank ordering of probabilities that allow us to do meaningful comparisons between our SCFG/MRF distributions on sequence space without the need to consider all sequences that the groups can produce.

In addition, preliminary studies have been done on accuracy of matching novel sequences to the correct motif group. These indicate that JAR3D can accurately match novel sequences to

the correct motif group, and they help to compare different approaches to the matching problem. These preliminary studies are discussed in Section 6.3. A more comprehensive review of JAR3D's performance on the matching problem is planned for a future paper.

Finally and most importantly, JAR3D is being used by scientists to help in their study of RNA. We frequently help graduate students and other researchers who want to use JAR3D for a variety of applications, and some groups we work closely with have already published results. Theis, Gorodkin, Zirbel, and Hofacker have used JAR3D to search genomes for motifs (Theis, Zirbel, Zu Siederdissen, Anthon, Hofacker, Nielsen, and Gorodkin, 2015). Akkuratov et al. used JAR3D in their study of plant orthologous intron (Akkuratov, Walters, Saha-Mandal, Khandekar, Crawford, Zirbel, Leisner, Prakash, Fedorova, and Fedorov, 2014). Additional work is under way with other research groups.

1.6 Overview of the dissertation

Chapter 1 gives an introduction to RNA and RNA 3D loops, the problems related to the study of RNA 3D loops that we set out to address, and the tools we developed to do so. It also provides this self-referential overview of the dissertation. Chapter 2 gives a more detailed introduction into RNA primary, secondary, and 3D structures, as well as an introduction to other important concepts such as RNA basepairs and isostericity. Chapter 3 gives a review of the literature on both RNA 3D structure prediction and the use of SCFGs to model RNA sequence variability, as well as some necessary background information for those topics. Chapter 4 discusses our methodology for modeling RNA sequence variability with SCFG/MRF models. Chapter 5 introduces the JAR3D software package we made that implements the SCFG/MRF methodology. Chapter 6 discusses how we handle false positive control in JAR3D, and also explains some validation studies that we have done. Chapter 7 discusses the idea of RNA sequence space and tools I have developed to study how JAR3D models cover and overlap one another in sequence space. Finally, Chapter 8 gives conclusions and looks at possible follow-up and future work.

CHAPTER 2 RNA PRIMARY, SECONDARY, AND 3D STRUCTURE

2.1 An introduction to RNA

RNA is an acronym for ribonucleic acid, a nucleic acid molecule that is involved in a variety of cellular functions. Many RNA molecules are important in forming proteins, such as messenger RNA or mRNA, which carries protein information. Other RNA molecules besides mRNA are often called non-coding RNA. Some of these are also important in the forming of proteins, such as ribosomal RNA or rRNA, which forms ribosomes which reads mRNA and forms proteins, and transport RNA or tRNA, which brings the amino acids that form proteins to the ribosome. In addition to these well known and studied RNAs there are a wide variety of other non-coding RNAs that have been discovered in recent years, and that continue to be discovered.

RNA molecules are long chained molecules that are made of of smaller, repeated parts called nucleotides. RNA nucleotides each have a common backbone and one of four bases, adenine, cytosine, guanine, and uracil, typically denoted by A, C, G, and U, respectively. It is similar to DNA, which is also made up of adenine, cytosine and guanine, but DNA has thymine instead of uracil. DNA is always produced to be double stranded, and both strands are made together so that each nucleotide is paired in a helix. On the other hand, RNA is copied from DNA and is single stranded and can fold back upon itself to form helices similar to those in DNA, as well as other structures. The structure or structures an RNA molecule can form is dependent on the sequence of A, C, G, and U bases that make up the molecule.

The International Union of Pure and Applied Chemistry, or IUPAC, has a set of codes are that are used to represent nucleotides, or sets of nucleotides. All IUPAC codes are a single capital letter. As discussed above, there are the basic A, C, G, and U codes which represent the four RNA bases. The other codes represent sets of bases. For example N is short for nucleotide and can be any of the four bases, and Y is short for Pyrimidine and can be C or U.

The three dimensional structures that many RNA molecules make when they fold back upon

themselves are integral to the functions that the RNAs perform in cells. Mistakes in the process through which DNA is passed on through reproduction can lead to mutations that change the sequence of RNA bases in a molecule for the offspring and subsequent generations. It is possible, however, for the same 3D structure to form from different sequences of RNA, and for the molecule to still perform its function even after mutations have occurred. However, some sequence changes will alter the structure of the RNA in a way that will not be function, and are not observed. Thus, we see sequence variability in a given RNA molecule from one organism to the next, but the possible variability is restrained.

2.2 RNA 3D structure determination

As previously discussed, the function of RNA molecules is often dependent on the 3D structure of the molecule after it folds back on itself. This means that the study of RNA molecules often involves determining the 3D structure of them experimentally. It is, however, difficult and expensive to experimentally determine the 3D structure of an RNA. The 3D structure for the 5S molecule that helps form E. coli's ribosome is show below in Figure 2.1. The structure has been color coded to help with interpretation of the model. The molecules that form the backbone are in grey, and those forming Watson-crick double helices are colored in orange, red, green, blue, and brown. The molecules in black are those in the loop regions.

Because RNA molecules are very, very small, determining their 3D structure experimentally is not a simple task. The 3D structure of 5S ribosomal RNA pictured above in Figure 2.1 was discovered through X-ray crystallography. The 2009 Nobel Prize in Chemistry was awarded to Thomas Steitz, Ada Yonath and Venkatraman Ramakrishnan for work done solving the structure of the ribosome with X-ray crystallography in the late 1990s and early 2000s (Samhita and Varshney, 2010). X-ray crystallography is a process in which many copies of a RNA molecule are formed into a crystal. The crystal is then bombarded with X-rays, which causes the X-rays to diffract in different directions. The pattern in which the X-rays diffract can then be captured and analyzed to "solve" for the 3D structure of the RNA molecule. Unfortunately, crystallizing RNA molecules is both difficult and expensive, and is not feasible for some RNA molecules.

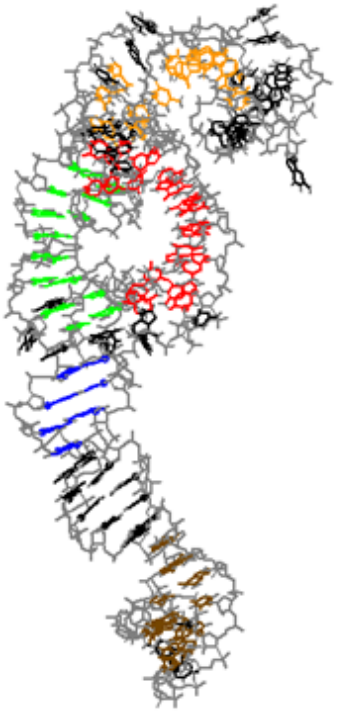


Figure 2.1 E. coli 5S rRNA 3D structure.

Another technique used to analyze the 3D structure of RNA molecules is nuclear magnetic resonance, or NMR, spectroscopy. NMR spectroscopy uses electromagnetic radiation to create a shift in the spin and energy level on the nuclei of the atoms in a molecule. These shifts can be measured and used to solve for a structure. NMR is fairly effective and affordable compared to X-ray crystallography, but unfortunately is only suitable for small RNAs of 100 nucleotides or less.

There are also newer methods being developed to solve the structures of RNAs and other molecules, such as cryo-EM (cryo electron microscopy), which involves analyzing molecules that have been frozen in a solution instead of crystallized (Bai, McMullan, and Scheres, 2015). This promises to reduce the cost and increase the rate of solving RNA 3D structures, but even then it is simply not feasible to for the structures for all of the vast number of RNAs that are being sequenced to be solved for. Because of this, a very important problem in the study of RNA is prediction of RNA 3D structure from sequence.

2.3 RNA secondary structure, Watson-Crick helices, and isostericity

When an RNA molecule folds back on itself to form a 3D structure, the primary substructures that will make up the resulting structured RNA is Watson-Crick helices. Watson-Crick helices are made up of consecutive Watson-Crick basepairs. These Watson-Crick basepairs, when seen in helices, are typically between an A and a U or a C and a G, and these are combinations are called canonical Watson-Crick basepairs. However, helices may also occasionally contain Watson-Crick pairs with G and a U. More on basepairs can be found in Section 2.6. In general, the order nucleotides appear in a basepair matters, so there are four canonical Watson-Crick base combinations, AU, CG, GC, and UA. Both the GU and the UG combinations also appear in helices, but less frequently and are again not considered canonical combinations. Watson-Crick basepairs are called cWW basepairs for short. More on these 3 letter abbreviations, and other types of RNA basepairs, is in Section 2.6.

The four canonical Watson-Crick base combinations can be used interchangeably in helices, meaning that if an AU basepair mutates into a CG combination, it would not change the overall structure of the helix or the larger RNA. This is because the four canonical are very geometrically similar. We refer to this similarity as isostericity, and it specifically measures the change the backbone of the RNA molecule would have to make to accommodate a change in the base combination being used to make a basepair. For changes between the four canonical Watson-Crick base combinations, the change in the backbone is miniscule, so we refer to these base combinations as isosteric. To change from a canonical Watson-Crick base combination, such as CG, to a UG pair requires the backbone to shift a more noticeable but still fairly small amount, so we refer to the CG and UG Watson-Crick basepairs as near-isosteric. Interestingly, UG and GU Watson-Crick basepairs differ from the canonical basepairs in different ways, so switch from a UG to a GU Watson-Crick base combination creates a substantial shift in the backbone of the RNA, so we would refer to UG and GU Watson-Crick base combinations as non-isosteric. It is through isostericity that we can understand the sequence variability seen in Watson-Crick helices. For example, if we observe a GC basepair, we would put high probability on seeing GC, CG, AU, and UA in the

same position in homologous molecules, some probability on GU and UG, and low probabilities on other possible base combinations.

From an RNA 3D structure, the nucleotides involved in creating Watson-Crick basepairs and Watson-Crick helices can be identified. This information can be used to to create a diagram which shows the Watson-Crick helices in the model and which nucleotides are forming Watson-Crick basepairs to form those helices. These diagrams are called RNA 2D diagrams, because they are drawn in a simple 2 dimensional manner and provide a level of information between an RNA sequence and its full 3D structure. A 2D structure diagram for the 5S region of E. coli's ribosome is shown below in Figure 2.2. There are five helices in the 5S chain, and they are shown in the diagram and labeled 1-5. The sequence for the 5S region of the E. coli ribosome is shown below in Figure 2.3. RNA sequences are typically represented by a string of the letters A, C, G, and U.

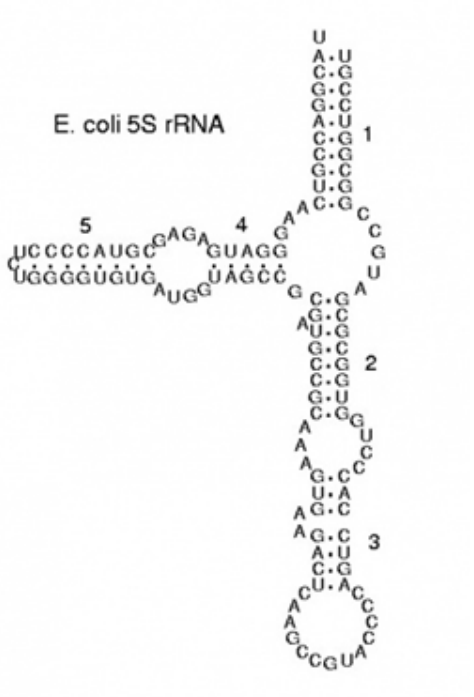


Figure 2.2 E. coli 5S rRNA secondary structure. Note the sequence runs counter clockwise, starting with UGCC in the upper right, and corresponds to the sequence in Fig seq example below. This is called 5' to 3' order, and is the usual way of representing RNA sequences. Obtained from RFAM (Griffiths-Jones et al., 2003).

In the 2D structure above in Figure 2.2, Watson-crick basepairs are indicated by a dot between

UGCCUGGCGGCCGUAGCGCGGUGGUCCCACCUGACCCCAU
GCCGAACUCAGAAGUGAAACGCCGUAGCGCCGAUGGUAGU
GUGGGGUCUCCCCAUGCGAGAGUAGGGAACUGCCAGGCAU

Figure 2.3 E. coli 5S ribosomal RNA sequence. Obtained from RFAM (Griffiths-Jones et al., 2003).

the bases. In other 2D structure diagrams, the basepairs might be indicated with a dash instead, making helices look like ladders. Bases not involved in a helix are typically drawn in circular loops in 2D diagrams. These RNA "loops" are the main focus of our research, and are discussed in more detail in Section 2.5.

Bases can bulge out of a helix without changing the overall structure of the helix. This occurs in helices 2 and 3 in Figure 2.2 above. In helix 2 a single A is bulging out, and in helix 3 two As are bulging out. Because these bulged bases do not create a change in the structure of the rest of the helix, we do not consider them as a feature that breaks a helix into two seperate helices, like the loop region between helix 2 and 3. These bulges can happen in many places in RNA molecules, and are another aspect of sequence variability that we understand and can model. Later, in Chapter 4, we will discuss how we build the possibility for similar bulged bases in out models for non-helical regions of RNA.

When working with RNAs, we may want to refer the sequence of just a substructure within the RNA, such as a helix. For example, we might want to talk about the sequence of helix 2 in Figure 2.2 above. The sequence starts with UGCC at the top and runs counter clockwise, so the sequence of helix 4 would be written as GGAUG ... UAUCC. In the BGSU RNA group, we use a * symbol to denote a break in a sequence when talking about RNA sub-structures, so we would write the sequence GGAUG*UAUCC. Note that we can see the canonical basepairing in the sequence. The basepairs go from the outermost bases to the innermost, with the outermost G matching with the outermost C, then the next G with the next C, and so on. The innermost basepair is a GU combination, which is not uncommonly seen in helices.

RNA secondary structures can also be expressed in what is called dot-bracket notation. This is a way of notating which nucleotides in a sequence are making Watson-Crick basepairs in an RNA

using just text. A line of text is written above the sequence, in a fixed width font, with opening and closing brackets or parentheses indicating nucleotides that are making a Watson-Crick basepair as part of a helix, and a dot symbol or period over bases which are not. As an example, the dot bracket notation for helices 4 and 5 of the E. coli 5S structure shown in Figure 2.2 is shown below in Figure 2.4. Note that the parentheses or brackets also indicate which other base a base is making the basepair with, so the first G is making a Watson-Crick basepair with the last C, as the parentheses above them are paired together.

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

Figure 2.4 Dot bracket notation for helices 4 and 5 from the 5S region of the E. coli ribosome. Parentheses indicate bases that are making Watson-Crick basepairs, and periods indicate those that are not.

2.4 RNA alignments

It is possible that when working with RNA sequences one will have access to multiple sequences, from different organisms, thought to share a common ancestor (we say they are homologous) and fold into the same structure. For many structure prediction applications, RNA sequences need to be aligned before they can be used. An alignment of RNA sequences is a data structure that somehow shows which nucleotides in the sequence are in the same place performing the same function in the 3D structure of an RNA. Making alignments is not a simple task, because evolution may cause insertions or deletions in the sequence of the RNA as different organisms evolve, resulting in sequences of different lengths.

The most common way of making alignments is by arranging RNA sequences into columns, with the columns corresponding to a specific location and function within the overall structure of the RNA. For example, the first and last columns of the alignment might be positions in an opening Watson-Crick basepair in a helix. Other alignments assign ranges of nucleotides to structural features instead of each individual base to a column. For example, nucleotides 1-5 and 96-100 might be assigned to helix 1 for sequence 1, and nucleotides 1-6 and 97-102 might be assigned to the same helix 1 in sequence 2. This allows for structural features to vary in size in a way that

is difficult to do well in column-based alignment, but these feature-based alignments are hard to make and require in depth knowledge of the underlying RNA structure.

2.5 RNA loops

The primary focus of this dissertation is the prediction of the 3D geometries of RNA internal and hairpin loops from sequence. This is an important step in the process of predicting RNA 3D structures from RNA sequences. Internal loops and hairpin loops occur between two RNA helices or at the end of helices, respectively. Many RNAs also contain junction loops, where three or more helices meet, although JAR3D does not yet work with junction loops. There is a three-way junction in Figure 2.2 above in Section 2.3.

It is useful when studying RNA internal loops to have a format to express the sequence of the internal loop by itself without the other nucleotides in the RNA. To do this we need to indicate where the "break" in the internal loop is, where the first strand ends and the second begins. We use the * character to indicate this break in the strands. We also include the last cWW base pair on the helices, the flanking basepairs, with the internal loops. This is because sometimes the flanking nucleotides make other interactions with nucleotides in the internal loops, creating dependencies between them and the interior nucleotides. So, if an internal loop has a sequence written CAAGU*ACCUG, we know that the first nucleotide, a C, is making a cWW basepair with the last nucleotide, a G. The fifth and sixth nucleotides, a U and an A, are also making a cWW basepair, and are separated by the * to indicate that they are on different strands. The sequence is listed in the usual 5' to 3' order used for RNA.

RNA loops are much more varied in their structure than helices. Sometimes loop structures may even be unique to the molecule and location that they occur in. However, many loop structures are modular, and occur both in different molecules and multiple times within the same molecule. Some of these recurrent loop structural motifs have been observed in many different RNA molecules and have been well studied.

One of the most studied internal loop structures is the sarcin-ricin internal loop motif. Sarcin-ricin internal loops are very stable and provide binding sites for proteins and other RNA molecules.

Sarcin-ricin internal loops have been observed in many different RNA molecules, including several in the ribosome, which have received the most study. A sarcin-ricin internal loop from the 28S ribosomal RNA of a rat is show below in Figure 2.5. It was taken from PDB structure 1Q96, and is listed in the RNA 3D Motif Atlas as loop IL_1Q96_001 in motif group IL_85647.3, which can be viewed at `http://rna.bgsu.edu/rna3dhub/motif/view/IL_85647.3`. The sequence of the loop is CUCAGUAU*AGAACCG, and the nucleotides are color coded. The first strand runs from the C in the lower left to the U in the upper right, and the second strand runs G in the lower right to the A in the upper left. The G in the middle of the first strand, seen bulging out of the left side, is the binding site. The G and the following U are also side by side, instead of stacked on top of each other on different levels.

Figure 2.5 Sarcin-ricin internal loop from rat 28S ribosomal RNA. This loop is a member of RNA 3D Motif Atlas group IL_85647.3.

Another well known internal loop structural motif is the kink-turn. As the name implies, a kink-turn internal loop creates a tight turn between two helices. There are a variety of different kink-turn structures, and many kink-turn structures are repeated in different molecules. A kink-turn from the large ribosomal subunit of Haloarcula marismortui is shown below in Figure 2.6. It was taken from PDB structure 1S72. In the RNA 3D Motif Atlas, it has the loop ID IL_1S72_048, and is a member of motif group IL_65553.8, which can be viewed at `http://rna.bgsu.edu/rna3dhub/motif/view/IL_65553.8`. Its sequence is CGAGAAC*GGGAG. The first flanking basepair is at the top of the figure, facing up, the second is in the lower right, facing to the right. The kink-turn creates an approximately 90 degree turn between the helices that flank it.

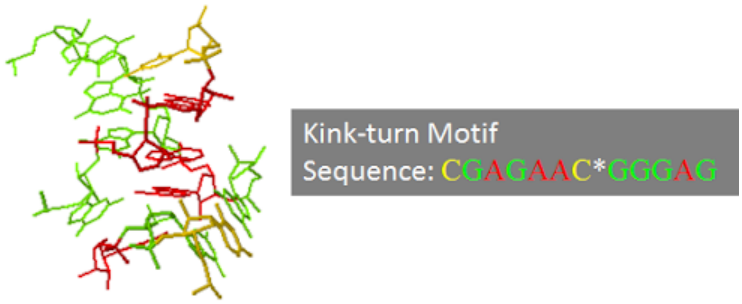


Figure 2.6 Kink-turn from the Haloarcula marismortui ribosome. It is a member of RNA 3D Motif Atlas group IL_65553.8.

A well known hairpin loop is the GNRA motif. It is named after the sequence of nucleotides after the last Watson-Crick basepair. The N stands for nucleotide, and can be any of the four bases, and the R stands for puRine, which means A or G. Typical sequences seen in a GNRA loop would then be GAGA, GGGA, or GAAA. Full sequence variability can be viewed on the motif group's webpage at http://rna.bgsu.edu/rna3dhub/motif/view/HL_67042.12#variants. GNRA hairpins are quite common, so common, in fact, that HL_67042.12 is the largest hairpin loop group in the RNA 3D Motif Atlas release 1.13. Like sarcin-ricin internal loops serve as binding sites for other molecules, as well as long range interactions within a molecule. A GNRA loop from the 60S ribosomal subunit of T. thermophila is shown below in Figure 2.7, taken from PDB structure 4A1B. In the RNA 3D Motif Atlas, it has the loop ID HL_4A1B_011, and is a member of motif group HL_67042.12, which can be viewed at `http://rna.bgsu.edu/rna3dhub/motif/view/HL_67042.12`. Its sequence is UGAAAA. Note we include the flanking UA cWW basepair at the beginning and end of the sequence. In this instance both the N and the R are As. The NRA part of the loop is clearly stacked over the flanking basepair, which is another characteristic of the motif.

These are just a small sampling of the known loop structures. The 3D structures of new RNAs are rapidly being solved, and with them come new RNA loop structures. Some of these will be the same as existing RNA loops, but many will be new structures.

Figure 2.7 GNRA hairpin loop motif from the 60S ribosomal subunit of T. thermophila. It is a member of the RNA 3D Motif Atlas motif group HL_67042.12.

2.6 RNA basepairs

While RNA double helical regions consist of stacked Watson-Crick basepairs, loop regions frequently contain non Watson-Crick basepairs. Watson-Crick basepair means that both nucleotides are bonding on their Watson-Crick edges. Using the Leontis-Westhof notation (Leontis and Westhof, 2001; Leontis, Stombaugh, and Westhof, 2002), nucleotides have three edges, Watson-Crick, Hoogsteen, and sugar. Basepairs are symmetric in the sense that a sugar-Hoogsteen basepair is the same as a Hoogsteen-sugar basepair, so there are 6 possible edge combinations. Figure 2.8 below shows an A and a U with their edges labeled.

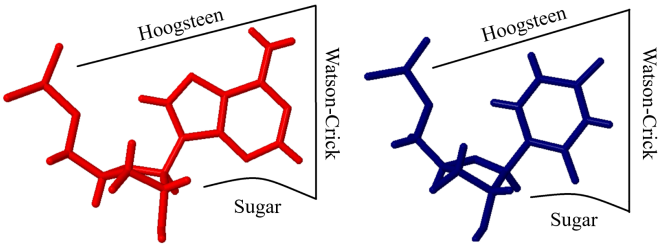


Figure 2.8 Adenine and uracil bases with Watson-Crick, Hoogsteen, and sugar edges labeled.

There are also two possible orientations for each edge combination, cis and trans. Cis indicates that both bases have the same orientation with respect to the RNA backbone, and trans indicates that one has been flipped, so the base-backbone connections will be on different sides. All edge combinations can be cis or trans, so there are 12 total basepair families. Examples of AU cis-Watson-Crick Watson-crick and trans-Watson-Crick Watson-Crick basepairs are shown below in

Figure 2.9.

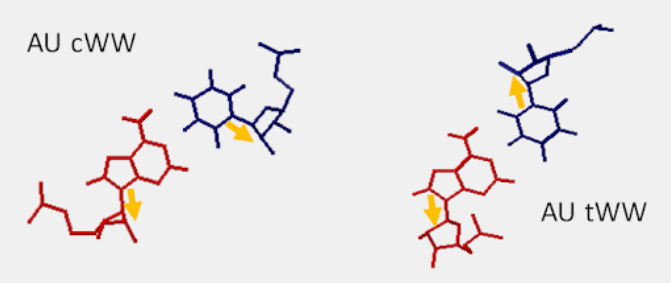


Figure 2.9 AU Watson-Crick basepairs in cis and trans. Yellow arrows highly the base-backbone connection. For the cis pair, they are on the same side, for the trans pair, they are on opposite sides.

For shorthand, basepairs are referred to with a three letter code, first a c or t for cis or trans, then W, S, or H for the edge of the first base, and finally another W, S, or H for the edge of the second base. So a cis-Watson-crick Watson-crick basepair is referred to as a cWW basepair, and a trans-Hoogsteen sugar basepair is a tHS basepair. The four bases give each basepair 16 possible base combinations, resulting in 192 possible choices of basepair family and base combination. The order of the bases does not matter, so there is no difference between, for example a AU tSH basepair and a UA tHS basepair. It is also worth noting that some base combinations are not possible because the charges on the edge of the bases would repel one another, for example a GG cWW basepair is not possible. Most base combinations are possible and have been observed in 3D structures. See the RNA Basepair Catalog for all observed base combinations across the 12 basepair families, `http://ndbserver.rutgers.edu/ndbmodule/services/BPCatalog/bpCatalog.html`.

2.7 RNA base-backbone interactions

Basepairing is not the only way that nucleotides in an RNA can interact with the rest of the RNA molecule. The base part of a nucleotide can also make hydrogen bonds with the backbone of another nucleotide in the RNA molecule. There are two types of base-backbone interactions, base-ribose or base-phosphate interactions. Our studies have shown that bases making a base-phosphate interactions are conserved at a very high rate ($> 90\%$), and those that are making base-ribose interactions are also very highly conserved, as discussed in the first JAR3D paper and shown in its

supplementary material (Zirbel et al., 2015). Note that only the base for which the base part of the molecule is interacting is constrained, the backbone is the same for all bases, so the base with the interacting phosphate is not constrained.

CHAPTER 3 REVIEW OF RELEVANT LITERATURE

This chapter will review the literature relevant to our methodology for RNA loop sequence modeling. The chapter will first introduce RNA 2D and 3D structure prediction, then review the relevant literature on RNA 3D structure prediction. There is then an introduction to stochastic context free grammars (SCFGs), then finally a review of the relevant literature on SCFGs used in the study of RNA.

3.1 Introduction to RNA 2D structure prediction

This section will give a brief introduction to RNA 2D structure prediction, as it is necessary background information for the review of RNA 3D structure prediction in Section 3.3.

There are a number of reasons that RNA 3D structure prediction often starts with 2D structure prediction. The fact that helices are the substructure that are repeated the most in structured RNAs, and the fact that they have a very reliable sequence pattern in the appearance of consecutive AU, CG, GC, and UA basepairs, make RNA 2D structures a good starting point for RNA structure prediction. Unfortunately, a single RNA sequence might fold many different ways that form reasonable looking 2D structures, so finding the right one is not a simple task.

RNA helices are very stable structures, and the physics and chemistry involved in the folding of RNA means that more stable arrangements are more likely to form. Because of this, a very reasonable and effective method of predicting RNA 2D structures from a sequence is to find the fold that will produce the most stable structure. This method is known as free energy minimization. Free energy minimization looks for the structure that minimizes a quantity called Gibb's free energy change, which measures the difference between the energy of the folded structure and the completely unfolded state of the molecule. A number of factors contribute to this. In general, a structure with more Watson-Crick basepairs will be more stable and have a lower Gibbs free energy change than one with less. However, longer helices are also more stable than several shorter helices, so a structure with a few long helices might have a better Gibbs free energy change

than a structure with more Watson-Crick basepairs in shorter helices. One of the oldest and best known tools for predicting RNA 2D structure using free energy minimization is MFold, described in (Zuker, 2003) and currently available at `http://unafold.rna.albany.edu`.

In addition, it has been shown that the order of basepairs in a helix, which affects how the basepairs stack on top of each other in the helix, also affects the stability of the structure. These stacking parameters were first described by the Turner group in (Turner and Mathews, 2009). Taking the stacking (sometimes referred to as nearest neighbor) effects into account greatly improved RNA 2D structure prediction. The Turner nearest neighbor parameters are used by MFold.

When more than one sequence is available, RNA 2D structure prediction becomes more reliable. When multiple sequences are available, the known sequence variability of helices can be fully leveraged. This is done by searching through columns of aligned RNA molecules for columns that show covariation between GC, AU, UA, and CG, an indication that the nucleotides corresponding to the columns are making a Watson-Crick basepair. Covariation means that a change in one column means a change in another column is likely, and the sort of covariation being looked for in RNA 2D structure prediction is that which preserves canonical Watson-Crick basepairs. This technique is known as comparative sequence analysis, and is still considered the most reliable way to predict RNA 2D structure, though it has actually been in use for quite a long time. It was first used to predict the structure of transfer, or tRNA, in 1965 (Holley, Apgar, Everett, Madison, Marquissee, Merrill, Penswick, and Zami, 1965), and those predictions were later confirmed when the 3D structure of tRNA was solved in 1974 (Robertus, Ladner, Finch, Rhodes, Brown, Clark, and Klug, 1974).

3.2 Introduction to RNA 3D structure prediction

Predicting RNA 3D structure directly from sequence is even more difficult, and perhaps impossible in some cases, than solving structures via x-ray diffraction, NMR, or electron microscopy. So far there are no reliable computational methods for predicting RNA 3D structure from sequence. Being able to predict 3D structure from sequence is highly desirable though, because it is comparatively inexpensive and easy to reliably determine the sequence of an RNA. It is therefore an

important problem in the study of RNA to be able to predict an RNA molecule's 3D structure from its sequence.

When a 2D structure has been predicted for an RNA sequence, attempts at predicting the full 3D structure can be made. Because the 3D structure of Watson-Crick double helices is well known and quite regular, the secondary structure of an RNA is nearly as good as a full 3D structure for helical regions. The gap between 2D structure prediction and 3D structure prediction is then, primarily, the prediction of the 3D structure of the loop regions. The structure of the loop regions determines the orientations of the helices and thus contributes to the final 3D structure of the RNA. One of the functions of JAR3D is to assist in the prediction of the 3D structure of these loop regions.

3.3 Review of literature on RNA 3D structure prediction

This section reviews different methods for prediction of the 3D structure of RNAs from sequence. Some of them start with 2D structures, others predict the 2D structures along the way using techniques talked about above in Section 3.1. Many rely on matching small stretches of sequences to small fragments of existing 3D structures, and using those 3D structural fragments to assemble new 3D structures.

One of the first attempts at breaking down RNA 3D structural elements and using this information to assist in tertiary structure prediction was Lemieux and Major's examination of cyclic motifs (Lemieux and Major, 2006). They divided the large ribosomal subunit (LSU) into 4 nucleotide cycles, then clustered those cycles based on basepair and stacking classifications. The cycles were essentially arranged in squares, so that each position can be dependent on two of the other positions. So, they can model stacked basepairs, and base triples, as long as there are no extra nucleotides between the bases making the triple.

They theorized that these small cyclic motifs comprised most of the LSU, and could likely be used in tertiary structure prediction for other RNA molecules. They do this through what can be described as a fragment assembly method, meaning that parts of the sequence are matched to sequences seen in 3D structure fragments, and then a new 3D structure is assembled from those existing 3D fragments. In this case, overlapping four nucleotide sets from sequences are matched

to cycles, and overlapping cycles are assembled into a new structure. The tools they built to do these predictions are called MC-Fold, which produces augmented secondary structures that include non Watson-Crick basepairs, based on the cyclic motifs, and MC-Sym, which takes the augmented secondary structures and produces 3D structures (Parisien and Major, 2008). They reported being able to reproduce 11 of 13 test structures. MC-Fold and MC-Sym work on full RNA structures instead of motifs specifically, and runtime increases exponentially with size, so the test structures and suggested use cases have short sequences, generally less than 50 nucleotides. MC-Fold and MC-Sym work on single sequences at a time. MC-Fold and MC-Sym are capable of making de novo predictions of loop regions, since the system assembles 3D structures from small fragment sets. For known motif geometries, though, we believe we can build more accurate sequence models by matching sequences to entire motif geometries.

Another tool that uses small fragments of RNA to predict 3D structures was published a year later in FARNA (Das and Baker, 2007). FARNA stands for Fragment Assembly of RNA, and was inspired by the Rosetta tool for protein structure prediction. FARNA works by assembling test structures from fragments of RNA observed in 3D structures, then running molecular dynamic simulations on those test structures. This is less computationally intensive and less time consuming than running molecular dynamics simulations from scratch, yet is still much more computationally intensive than MC-Sym and the JAR3D system we developed. Their tests on FARNA found about a 90% success rate in predicting Watson-Crick basepairs, which was in line with most secondary structure prediction tools at the time. They also found that they could identify about a third of non-Watson-Crick basepairs in the RNA structures they tested. This was a good first step towards prediction of structures in loop regions.

The FARNA methodology was built upon with a tool that specifically focused on RNA motifs, called FARFAR, or Fragment Assembly of RNA with Full-Atom Refinement (Das, Karanicolas, and Baker, 2010). FARNA and FARFAR both work by taking small fragments of RNA 3D structures matched to sequences, and assembling them into new structures. They work with smaller fragments than MC-Fold and MC-Sym, of 1-3 nucleotides each. Both FARNA and FARFAR then

refine the model using a low resolution energy function, then FARFAR further refines the model at an atomic level. FARNA tests found that FARFAR was able to reproduce 50% of a set of 32 6-20 nucleotide motifs, a definite improvement over FARNA, but still leaving room for improvement. FARNA and FARFAR work on one sequence at a time, and have the ability to predict completely new geometries, whereas many other tools only look for new instances of known geometries.

RNAWolf (Höner zu Siederdissen, Bernhart, Stadler, and Hofacker, 2011) is similar to MC-Fold in that it produces extended secondary structures, secondary structures that include non-Watson Crick basepairs. Unlike MC-Fold, however, RNAWolf allows for base triples of non adjacent nucleotides. RNAWolf is built on a 2D structure prediction tool, MC-Fold-DP, that uses dynamic programming. They compared the tool to state of the art thermodynamic folding algorithms, and found that it performed poorly in general, but did much better on 3D data specifically gathered from the Protein Data Bank. The authors themselves admit that RNAWolf does not reach their desired level of accuracy.

RNA-MoIP, or RNA Motif Integer Programming, also does RNA structure prediction by assembling models from fragments seen in 3D structures (Reinharz, Major, and Waldispühl, 2012). The program produces augmented secondary structures with non Watson-Crick basepairs similar to MC-Fold, and they feed these secondary structures to MC-Sym to produce 3D structures. RNA-MoIP is designed to work on larger structures than MC-Fold, and they tested it on a set of 9 structures ranging from 53-128 nucleotides. These came from a larger set of structures, with smaller structures and structures with pseudoknots removed, as well as removing two more that did not have homologous loops in the dataset. The first version of RNA-MoIP required exact sequence matches to the fragments from 3D structures, but it has since been extended to allow deletions and insertions, with the latter only allowed in junctions, and is also now available as a web server (Yao, Reinharz, Major, and Waldispühl, 2017).

RNAComposer (Popenda, Szachniuk, Antczak, Purzycka, Lukasiak, Bartol, Blazewicz, and Adamiak, 2012) is another fragment assembly tool for predicting RNA 3D structure. The program takes a sequence and secondary structure as input, and primarily builds a 3D structure by matching

elements of the secondary structure to 3D structures, like helices and loops, in the RNA FRABASE, a database of RNA 3D structure fragments (Popenda, Błażewicz, Szachniuk, and Adamiak, 2007). It also allows for generation of structural elements that are missing from their structural dictionary, using the CYANA structure calculation program. The structural elements from RNA FRABASE, or CYANA calculated structures when needed, are then combined into a full 3D structure for the molecule. Their approach allows them to tackle larger molecules than many other fragment assembly methods, and is accurate when structures are available in the RNA FRABASE. However, they found that although the CYANA calculated regions had several good properties, they were much less accurate than those from the RNA FRABASE.

RMDetect (Cruz and Westhof, 2011) is one of the first attempts at modeling and identifying RNA motifs without piecing together fragments from 3D structures. Instead of using 3D fragments, RMDetect trains Bayesian network models for entire motifs on hand curated sequence alignments. In Cruz and Westhof's paper, they made models for four RNA motifs, the G-bulge loop, C-loop, kink-turn, and tandem-GA loop. They found a false discovery rate of 23% in a control test of known sequences. They were also able to use the models to find 21 unreported instances of the motifs in a set of 1,444 alignments.

A pipeline was made to automatically produce RMDetect models on a much broader scale (Theis, Höner zu Siederdissen, Hofacker, and Gorodkin, 2013). This pipeline, metaRNAmodules, extracts RNA motif instances from the RNA 3D Motif Atlas at BGSU and maps them onto Rfam sequence alignments. They used 977 internal loop and 17 hairpin loop instances, and found their models to have "clear discriminatory power." Their models are based on a single alignment, whereas the original RMDetect models are consensus models based on sets of highly curated alignments. Because the new models were trained on smaller, lower quality data sets, they did not perform as well, but they do cover a much broader range of motifs.

RNAMotifScanX (Zhong and Zhang, 2015) is another tool for searching sequences for new instances of known motif families. The search tool is based on "a base-interaction graph alignment algorithm." Their paper shows query results for five motif families, the kink-turn, C-loop, sarcin-

ricin (referred to as G-bulge in RMDetect publications), reverse kink-turn, and E-loop. They found it performed well compared to other motif search algorithms, such as RMDetect. It is also quite fast and efficient.

RNA Puzzles is a competition in which competitors are challenged to predict the full 3D structure of RNAs from sequences. The RNA molecules for these competitions have been solved in crystal structures, but publication of the 3D structures is held until after the RNA Puzzles competition. As of February 2019, 21 RNA puzzles have been run.

Three articles have been published summarizing the results of RNA Puzzles competitions (Cruz, Blanchet, Boniecki, Bujnicki, Chen, Cao, Das, Ding, Dokholyan, Flores, et al., 2012; Miao, Adamiak, Blanchet, Boniecki, Bujnicki, Chen, Cheng, Chojnowski, Chou, Cordero, et al., 2015; Miao, Adamiak, Antczak, Batey, Becka, Biesiada, Boniecki, Bujnicki, Chen, Cheng, et al., 2017), and they serve as a good summary of the "state of the art" in RNA 3D structure prediction. In the most recent review, it was found that homology based predictions, that is predictions for which 3D structural data exists for a homologous molecule from another organism, is quite good. They stated "It is possible to model nearly all the structural details when a clear homologue can be identified." For molecules without homologues with known 3D structure, predictions for smaller molecules were quite promising and obtained by multiple independent groups. For larger molecules without homologues, results were not as good. Some groups predicted structures that were close to the actual structure, but ranked poorer models better than their best models.

Fragment assembly methods, such as MC-Fold, RNAWolf, and RNA Composer, are effective at homology modeling and assembling new structures when fragments have high sequence identity with fragments in their databases. Motif search tools, like RMDetect and RNAMotifScanX, are effective at finding new instances of well known motif geometries, when the sequences have been observed in alignments. Motifs for which high quality alignments are not available are therefore not handled as well. These search tools are also made for search for new instances of particular geometries, as opposed to predicting the geometry of a particular sequence or set of homologous sequences.

The review of the literature shows several areas that our method, and the JAR3D software suite, adds to the current state of motif prediction. Firstly, our JAR3D has a more complete approach to comparing to existing geometries than similar tools like RMDetect and RNAMotifScanX, as it creates models for all of the structures in the RNA Motif Atlas, instead of curated set of well studied motifs. Because it uses this approach, our method is the only one available that attempts to answer the question "Does this motif sequence form a geometry that has been observed before in 3D?" In addition, our method more completely models the network of interactions in the 3D motifs, and so will more faithfully match novel sequences to known geometries. In addition, our method can create alignments of sequences or sets of sequences to these structures, so that it makes specific predictions of the 3D position of each nucleotide in the sequence. Reliance on sequences observed in 3D data is a limitation of many currently existing 3D RNA prediction tools. Sequence data is a way to expand the base of available data for these tools, but it generally needs to be in high quality alignments to be useful. Alignments often do quite well in helical regions and suffer in loop regions. Our method's ability to align loop sequences can be used to improve alignments and could therefore be quite useful for future 3D structure prediction tools. Third, the JAR3D software runs much faster than many other tools, especially those that rely on molecular dynamics simulations, such as fragment assembly tools.

Overall, while progress has been made on predicting full RNA 3D structures from sequence, the problem is nowhere near being solved. Current tools are either slow, unreliable, or both. The JAR3D software is fast and makes some strides with reliability, when dealing with predicting already known 3D geometries, and also makes strides in dealing with the fact that one sequence might make multiple geometries. But because the overall 3D structure prediction is not solved, the methods listed above can be thought of as complementary instead of being in competition with each other. For example, a researcher might first use the JAR3D software to see if a loop sequence matches a structure that is already known from 3D structures, and then, if it isn't, use a fragment assembly method like FARFAR to attempt a de novo prediction of its 3D structure.

3.4 Introduction to SCFGs

Our approach to modeling RNA loop sequence variability uses Stochastic Context-Free Grammars, or SCFGs, to model RNA loop regions. When RNAs fold back on themselves to form 2D and 3D structures, they form networks of long range dependencies. These dependencies between the different letters in different positions of the RNA sequence are caused by the nucleotides creating basepairs. The need to maintain these basepairs creates a statistical dependence between the positions. These long range dependencies are often nested, which is well modeled by SCFGs. A more complete discussion of the dependencies seen in RNA sequence variation is in Section 4.1. Because of this, SCFGs have a long history of use in the study of RNAs, which is reviewed in Section 3.5 below. This section gives an introduction to context free grammars (CFGs) and SCFGs.

Context-free grammars (CFGs) can be made to model possible RNA secondary structures by describing a sequence of possible ways to re-write a text string consisting of "terminal" and "non-terminal" symbols. For example, to model cWW basepair helices, one can construct a simple CFG with one non-terminal symbol (B for basepair) and five non-terminal symbols (a, c, g, and u for the four bases and * for the strand break). Note that lower case letters are used for the four bases instead of the usual uppercase letters for easier differentiation from non-terminal symbols. The production rule (or re-write rule) for the non-terminal symbol **B** is shown below.

$$\mathbf{B} \rightarrow a\mathbf{B}u \mid u\mathbf{B}a \mid g\mathbf{B}c \mid c\mathbf{B}g \mid *$$

This means that the symbol B can be re-written in five different ways, four of which generate an au, ua, gc, or cg basepair, and the final one which produces an * symbol.

A generation history refers to a series of uses of the rewrite rule to produce a sequence. For example, the above grammar could produce a four basepair helix as follows:

$$\mathbf{B} \rightarrow c\mathbf{B}g \rightarrow ca\mathbf{B}ug \rightarrow cau\mathbf{B}aug \rightarrow cauc\mathbf{B}gaug \rightarrow cauc*gaug$$

In the resulting sequence, the first letter, c, makes a Watson-Crick basepair with the last letter, g; the second pairs with the second last, and so on.

This simple CFG can be made into a stochastic context free grammar by assigning probabilities to the possible productions. For example, we could give each of the four basepairs a 20% probability and also give the break symbol a 20% probability. This would cause the grammar to produce helices of geometric length, with an average length of 4. Grammars like this can be extended so they can model generic RNA sequences, including double helices, hairpin loops, junction loops, and internal loops, and we will refer to them as general SCFGs.

Say we want more have control over the length of the sequences produced, for example, to model a four basepair helix specifically. One way to achieve this is through the use of a special type of SCFG, called a guide tree SCFG. In traditional SCFGs, non-terminal symbols can repeat in a generation history many times, but in a guide tree SCFG, each non-terminal symbol is unique and occurs only once, and in a set order. A simple guide tree SCFG for a four basepair cWW helix can be made using rules similar to the one used in the CFG above. This guide tree SCFG is shown below:

$$\mathbf{B1} \rightarrow a\mathbf{B2}u \mid u\mathbf{B2}a \mid g\mathbf{B2}c \mid c\mathbf{B2}g \; (each\ with\ 25\%\ probability)$$

$$\mathbf{B2} \rightarrow a\mathbf{B3}u \mid u\mathbf{B3}a \mid g\mathbf{B3}c \mid c\mathbf{B3}g \; (each\ with\ 25\%\ probability)$$

$$\mathbf{B3} \rightarrow a\mathbf{B4}u \mid u\mathbf{B4}a \mid g\mathbf{B4}c \mid c\mathbf{B4}g \; (each\ with\ 25\%\ probability)$$

$$\mathbf{B4} \rightarrow a\mathbf{T}u \mid u\mathbf{T}a \mid g\mathbf{T}c \mid c\mathbf{T}g \; (each\ with\ 25\%\ probability)$$

$$\mathbf{T} \rightarrow * \; (with\ 100\%\ probability)$$

The above guide tree SCFG will always start with **B1**, then move to **B2**, then to **B3**, then to **B4**, then terminate. It also offers greater flexibility when it comes to assigning probabilities to the production rules. For example, CG and GC basepairs have stronger bonds than AU and UA basepairs, and we may find that it is more likely to see CG and GC at the beginning and end of the

helix. With a guide tree SCFG, one can assign higher probabilities to CG and GC in **B1** and **B4** while leaving the distribution even in **B2** and **B3**.

3.5 Review of literature on SCFGs for RNA

Stochastic Context-Free Grammars, or SCFGs, have a long history of use in the study of RNAs. This section will discuss the history of the use of SCFGs to study RNA, including the previous work from the BGSU RNA group that led to the creation of JAR3D.

In 1994, two groups independently published papers on using general SCFGs to model RNA secondary structure. Both Eddy and Durbin (Eddy and Durbin, 1994) and (Sakakibara, Brown, Hughey, Mian, Sjölander, Underwood, and Haussler, 1994) independently described methods for modeling the secondary structure of tRNAs. Sakakibara, whose paper came out later in the year, noted that the models were very similar, but they were trained differently. Eddy and Durbin's method assumed that both the structure of the grammar and its parameters need to be estimated. They defined an iterative process in which a grammar is constructed based on an initial alignment, then parameters are set using an expectation maximization algorithm. This model is used to improve the training alignment, which can then be used to re-estimate the grammar and its parameters. This process is iterated until it converges.

Sakakibara started with a set of four different grammars, so his work only needed to address the parameter estimate problem. He developed a method he called the "Tree Grammar Re-estimator". This algorithm works on folded sequences, so some basepairs must already be identified in the alignment. Both methods produced grammars that could perform similar tasks; discriminating tRNA sequences from non-tRNA sequences, and making alignments of tRNA sequences. It is worth noting that tRNAs were used because they were one of the only RNA molecules at the time for which many confirmed sequences were available, and that the base grammars were general CFGs for use with any structured RNAs. The parameters to make the CFGs SCFGs were trained with the goal that tRNA sequences would have high probabilities and other RNA sequences would have low probabilities.

Eddy's work was expanded on in QRNA (Rivas and Eddy, 2001) QRNA uses three probabilistic

models, two hidden Markov models (HMMs), called OTH and COD, and an SCFG, called RNA, that work together to model sequence variability of RNA sequences in alignments. The OTH model assumed that nucleotides in the alignment mutated independently, the COD model assumed that nucleotides coded homologous proteins, and the RNA model assumed the sequences were forming a conserved secondary structure. These models can then be used together to scan genomes or test specific alignments for genes that encode non-coding RNAs. The performance of the system was broken down based on the percentage of sequence identity in alignments of SRP RNAs and RNaseP RNAs. The system performed quite well for alignments with 50-90% sequence identity, with both high specificity and sensitivity. Specificity suffered for alignments with higher sequence identity, while sensitivity suffered for those with lower sequence identity.

In 1999, Knudsen and Hein described a method for predicting RNA secondary structures using SCFGs (Knudsen and Hein, 1999). They described a method that takes alignments of RNA molecules and produces predictions of a common secondary structure. The method uses a simple SCFG with three nonterminal symbols, one for "stems" (Watson-Crick helices), one for loops, and another that can produce either single nucleotides or new stems, and two terminal symbols, one for single bases and another for Watson-Crick basepairs. The probabilities for the production rules are estimated using the inside-outside algorithm. The method also uses evolutionary data, which requires a phylogenetic tree associated with the alignment, however, the authors describe a method for estimating a tree if one is not provided.

Knudsen and Hein found that their method compared favorably to other folding methods at the time, but did have several limitations. Their method relies on having aligned sequences, which is a strength when high quality alignments are available and is a weakness when they are not available. The method cannot predict pseudoknots, a limitation shared by many other secondary structure prediction tools. Finally, due to the simple nature of the SCFG they used, helices and loops were assumed to be of geometric length. In 2003, Knudsen and Hein improved their method and released a tool called pfold (Knudsen and Hein, 2003). The improvements were primarily to the efficiency of the algorithm, which allowed it to be applied to larger RNAs, and improvements that made the

method more robust to errors in input alignments. Knudsen and Hein's method was also built upon in 2006 in EvoFold (Pedersen et al., 2006), which searched the human genome for genes encoding structured RNAs using an alignment of the human genome to seven other vertebrate genomes.

The groundwork for our SCFG/MRF models was laid out in 2006 in Michael Sarver's dissertation (Sarver, 2006). Sarver's work with SCFGs was focused on using them for alignment of multiple RNA sequences using their common structure. Unlike other SCFG RNA tools , Sarver described grammars that were specific to a particular RNA 3D structure, as opposed to being general grammars for describing structured RNAs, or grammars that were based on 2D structures. Nonterminal symbols appear only once, in a set sequence. Such SCFGs are sometimes referred to as guide-tree SCFGs. Sarver's work also went beyond modeling secondary structure by describing ways to model the non-Watson-Crick interactions in the loop regions of structured RNA molecules.

Sarver describes a number of different types of "nodes", templates for types of nonterminal symbols that can appear. The basepair, initial, cluster, and hairpin nodes appear in our SCFG/MRF models. Junction nodes and alternative nodes are described in Sarver's dissertation, but not used in our SCFG/MRF models describing internal and hairpin loops. Sarver also described how to parameterize these nodes using a multiple sequence alignment which is also aligned to an RNA 3D structure, using maximum likelihood estimators. Because no method for parameterizing the grammars without an alignment was given, Sarver's work described a method for improving existing alignments using 3D structure, as opposed to aligning sequences from scratch. More comparisons with Sarver's work will be given in Chapter 4.

Infernal, short for "INFErence of RNA ALignment" is another tool that uses guide tree SCFGs to model RNA structures (Nawrocki et al., 2009). Infernal builds guide tree SCFGs based on a multiple sequence alignment, or a single sequence, and an attached consensus secondary structure that indicates which bases are involved in basepair interactions. The grammar is built based on the secondary structure, then parameterized using the sequence alignment. There are two main use cases for Infernal models; scanning genomes for sequences that form the same secondary structure, and producing alignments of novel sequences to a particular RNA secondary structure.

Although designed for use with secondary structures indicating Watson-Crick helices, Infernal can also model non Watson-Crick basepairs when provided with an appropriate alignment and secondary structure. It is, however, limited to the typical nested structures SCFGs are best suited for, and cannot model pseudoknots, base triples, or crossed basepairs.

CHAPTER 4 MODELING RNA LOOP SEQUENCE VARIABILITY WITH SCFG MODELS

This chapter will discuss how we use guide tree stochastic context free grammars (SCFGs), supplemented with Markov random fields (MRFs), to model the sequence variability in RNA loops. Here, we need to go beyond the modeling of Watson-Crick basepairs that was described in Section 2.3 when discussing modeling sequence variability based on 2D structures. We use observations and data from existing 3D structures to inform these new models.

4.1 Statistical dependence due to RNA basepairs

RNA molecules fold back on themselves when forming 3D structures. This folding means that bases that are far apart in the sequence of the chain might be making basepairs or interacting in other ways. Each basepair creates a statistical dependency between the bases involved in it, so that if one of them changes (due to a DNA copying error) the other is likely to change as well (over the next many generations in that genetic line of descent), to keep the basepair isosteric or near isosteric to the original, so the overall structure of the molecule doesn't change. These interactions in an RNA molecule create complicated patterns of long range statistical dependence in RNA sequences.

The network of basepair interactions made by RNA molecules when they fold back on themselves often result in networks of nested basepair interactions. These networks of nested dependencies are very efficiently modeled by stochastic context free grammars (SCFGs), one of the two main tools used in JAR3D models. SCFGs were introduced in Section 3.4. Linear arc diagrams are an excellent way of viewing the nested nature of basepair interactions in RNA structures. A linear arc diagram for the E.coli 5S ribosomal RNA from chain A of PDB structure 2QBG is shown below in Figure 4.1. This figure was generated by the tool R3D Align, and is the top half of an alignment to another 5S structure (Rahrig, Leontis, and Zirbel, 2010). The alignment can be seen at `http://rna.bgsu.edu/r3dalign/results/50dcfcf045337`. Each arc represents a basepair and thus a dependence between two nucleotides. The cyan and green arcs show non-

Watson-Crick basepairs, and most of these are nested as well. Some do break the nested structure, around position 40, and so cannot be modeled using the typical SCFG techniques discussed in Section 3.4. These basepairs are one of the reasons for augmenting SCFGs with Markov Random Fields, which we explain next.

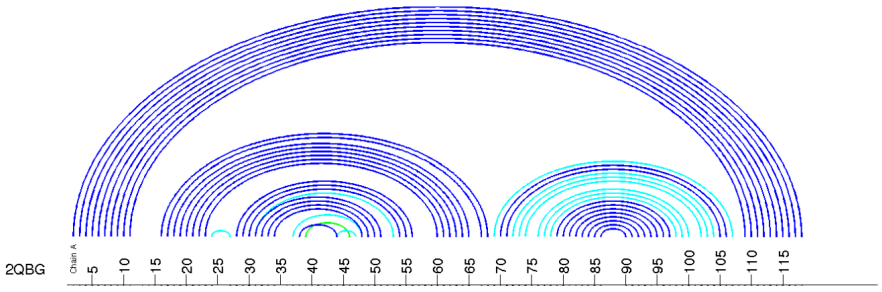


Figure 4.1 Linear arc diagram for 5S region of the E. coli ribosome. It was taken from PDB structure 2QBG.

Unfortunately, not all dependencies in RNA molecules create these easily modeled nested interactions. Within-strand basepairs, basepairs that occur between bases that are on the same strand within a loop instead of on opposing strands, break the typical nested interaction structure that is easily modeled by SCFGs. Because bases have three edges that can make interactions, they are not limited to making only single base pairs, but may also be making basepairs on two edges at once, creating a base triple.

The three dimensional graphic in Figure 2.5 above shows a base triple, which also has a within-strand interaction. The triple is between the G, the adjacent U on the same strand, and an A on the other strand. A two dimensional basepair diagram of the sarcin-ricin motif is shown below in Figure fig:base triple motif, and it shows the same triple. These two dimensional basepair diagrams are similar to secondary structures but show non-Watson-Crick basepairs as well. The symbols between the bases indicate the edges involved in the basepair as well as the bases' orientations. A circle indicates the Watson-Crick edge, a square the Hoogsteen edge, and a triangle the sugar edge. Open symbols indicate a trans pair and closed symbols indicate a cis pair.

A base triple does not always have a within-strand interaction, but it will always create an in-strand dependency, because the base triple creates a dependence between the two bases in the

triple not making a basepair, through transitivity. These within-strand dependencies create a more complex network of dependencies than is usually modeled in RNA with SCFGs alone. Note that it also possible, although much less common, for a base to be making interactions on all three of its edges, forming a network of four dependent interacting bases.

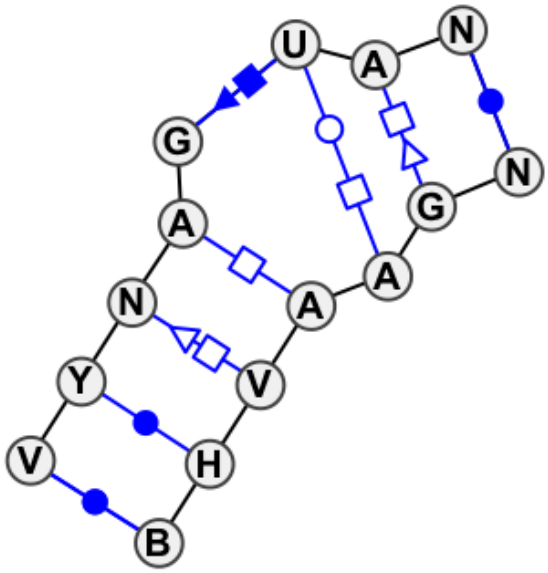


Figure 4.2 Basepair diagram for a sarcin-ricin internal loop. This is the diagram for loop IL_85647.3. The G and U in the left strand are involved in a cHS basepair, and the U is also making a tWH basepair with the A on the other strand, forming a base triple.

Finally, it is also possible for basepairs to "cross" each other in RNA loop regions, which also breaks the nested dependency structure. At least one of the crossing basepairs is usually involved in base triples, creating even more complex networks of difficult to model interactions. This can be seen wherever the arcs cross each other in Figure 4.1. Figure 4.3 below shows an extreme case of crossing bases, the c-loop. C-loops are modeled by motif group IL_73276.5, and they change the orientation of Watson-Crick helices by increasing their "twist". IL_73276.5 can be viewed at `http://rna.bgsu.edu/rna3dhub/motif/view/IL_73276.5`. Both flanking Watson-Crick basepairs are involved in base triples that cross the motif, resulting in a network

of dependencies that spans the entire motif.

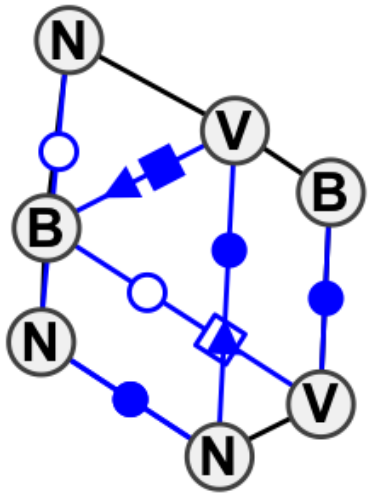


Figure 4.3 Basepair diagram for a c-loop internal loop. This is the diagram for IL_73276.5. C-loops have a complex network of dependencies between nucleotide positions. The Watson-Crick basepairs are indicated by an edge with a closed circle and no other symbol, one between circles labeled N and N, the other between circles labeled B and V.

4.2 Sequence variability in RNA basepairs and isodiscrepancy

As discussed in Chapter 1, errors in copying DNA lead to sequence changes in RNA, some of which do not disrupt the 3D structure of the molecule, and some of which are fatal and so are not observed. We wish to model the probability of observing a change in a given RNA basepair. There are 12 basepair families, each of which has up to 16 base combinations. This makes it prohibitive to estimate the probabilities of all possible base combinations in all basepair families being substituted by another base combination in a homologous molecule using data. A further complicating factor for the problem being addressed by JAR3D is that we are hoping to identify known motif structures in new molecules, not just in homologous molecules. A basepair may have additional constraints in one molecule that are not present in another. A data-only approach might lead one to believe a substitution is impossible in all instances of the motif, when it is instead only impossible in that particular instance of a motif, in the context in which it was found.

One might also think that you could use molecular dynamics and free energy calculations, similar to those used in 2D structure prediction, to compute what sequence changes are acceptable in a given motif. Unfortunately, most attempts to date have not worked very well. The calculations are hard to do, take a long time, and the results are so far have not been very reliable. See, for example, Jiri Sponer's research into the molecular dynamics of the sarcin-ricin motif (Kruse et al., 2014) under sequence changes.

What we need for the problem we are addressing is a quick, efficient way to calculate or estimate substitution probabilities for non-Watson-Crick basepairs. We can look at Watson-Crick basepairs for inspiration. Study of Watson-Crick helices has shown us that geometrically similar base combinations are more likely to substitute for each other, and that this assumption matches up with the physics involved with RNA folding. This has led us to an approach not based on data frequency or physics, but on geometry and isostericity. This helps us avoid the need for extensive datasets, which we lack, and complicated physics calculations, which our application often does not have time for.

As introduced in Section 2.3, if the geometries of two different base combinations in the same basepair family are very similar, we say that they are isosteric. Specifically, isostericity between basepairs means that the atoms that connect the bases to the backbone of the molecule are in very similar positions and are the same distance apart. When basepairs are isosteric, they can often be substituted for one another in a molecule without affecting the molecule's structure or function. In general, the more isosteric a base substitution is, the more likely it is to occur.

Isostericity gives us a way to estimate the probability that base substitutions will occur without needing to deal with detailed physics simulations and calculations. An isostericity based IsoDiscrepancy Index (IDI) score is used to give a numeric measure to the similarity between two basepairs (Stombaugh, Zirbel, Westhof, and Leontis, 2009). It is worth mentioning that although this method has worked quite well for us, there will always be ways to improve on estimation of these substitution probabilities. This method is a good starting point and is what we were capable of at the time.

4.3 Structure of SCFG/MRF models for modeling RNA loop sequence variability

This section will explain the SCFG/MRF structure of the models we use to model RNA loop sequence variability. This will include discussions of the basic SCFG nodes used to model nested interactions as well as the MRFs used to model more complex networks of interactions. The general structure of the SCFG/MRF models was outlined in Michael Sarver's dissertation (Sarver, 2006). Updates to the structure of the SCFG/MRF models have been noted.

It should be noted that how the models are parameterized is completely new. Sarver's work anticipated estimating the parameters for the models from sequence alignments, because his work anticipated making SCFG/MRF models for the sequence variability of specific RNA molecules, based on the 3D structure of those models and on available multiple sequence alignments of the molecule. Our SCFG/MRF models are for RNA loop structures, and we parameterize using isostericity and verified 3D sequences instead of sequence alignments. How we parameterize SCFG/MRF models for RNA loop sequence variability is discussed in detail later in this chapter. The software package and web servers that implement and use these models will be discussed in Chapter 5.

The production rules for the non-terminal symbols in our models are much more complicated than those in the simple guide tree SCFG above in Section 3.4. We refer to the non-terminal symbols as nodes. There are 5 different types of nodes in JAR3D, and we give each a single letter abbreviation for easy reference. There are basepair (B) nodes, fixed (F) nodes, cluster (C) nodes, initial (I) nodes, and hairpin (H) nodes. These nodes are described briefly in this section. Fixed nodes are new as compared to Sarver (Sarver, 2006) and hairpin nodes have been extended, but the other node types appeared in Sarver's dissertation.

A basepair (B) node usually produces one base on each strand. Basepair nodes have a very small probability of being deleted, and if they are deleted, they produce no nucleotides. If not deleted, the basepair node will always produce a nucleotide on each strand, and the nucleotides produced will typically be dependent. Our models assign non-zero probability to all 16 base combinations, even if they are extremely unlikely. Probabilities for basepairs are expressed by basepair

substitution matrices, which are 4x4 matrices that give probabilities for each of the base combinations. How these matrices are calculated is discussed in Section 4.5. A basepair node can also produce insertions on either strand after the two nucleotides making the basepair. This can be 0, 1, or even more nucleotides on either strand, generated independently of each other and of the nucleotides generated for the basepair itself.

A fixed (F) node simply generates a single base on one of the strands. Fixed nodes can also be deleted, with a very small probability, and if deleted they will not produce any bases. They give a distribution over the four possible bases of A, C, G, and U. Fixed nodes are used to model nucleotides that are not involved in basepair interactions, but are interacting with the nucleotides in the loop in some other way. This could be base-backbone interactions, or the nucleotide could be stacked on other bases in the loop. The need for fixed nodes was not anticipated in Michael Sarver's dissertation, so they are a new addition to the SCFG/MRF methodology. Fixed nodes are discussed in 4.9.

A cluster (C) node can be used to generate 2 or more dependent bases simultaneously. Cluster nodes can be deleted as well, and will produce no nucleotides if deleted. Because they often model more nucleotides than other nodes, the probability of cluster nodes being deleted is extremely small. Cluster nodes implement Markov Random Fields (MRF) to model these more complex networks of dependency. Cluster nodes may model multiple basepairs, each one using a 4x4 substitution matrix as described in Section 4.5. Cluster nodes may also be used to model bases that are stacked on other bases in the cluster node or making base-backbone interactions but are not involved in basepairing. These bases are modeled as fixed positions, and are parameterized similarly to fixed nodes, as described in Section 4.9. In practice these fixed positions are modeled as self interactions, and are parameterized with a diagonal 4x4 matrix. This is so that all interactions modeled by a cluster node are between two nucleotides, which simplifies their coding. Fixed positions in cluster nodes are new compared to Sarver (2006).

In addition to modeling interacting nucleotides, cluster nodes may also model variable length insertions, nucleotides in between the interacting nucleotides which are not making any stacking,

basepairing, or base-backbone interactions. Variable length insertions can happen between any of the base positions in the cluster node, but are only allowed for and modeled if an insertion has been observed in 3D sequences. Cluster nodes do not model insertions after the modeled bases. These insertions are modeled by the addition of an initial node, described below.

Initial (I) nodes model variable length insertions on both strands. They are called initial nodes because all our models start with an initial node, but they are used in other situations as well. Initial nodes do not have a deletion probability, because they have the ability to produce no bases built into them without being deleted. Ideally, motif sequences passed into our system will not have any nucleotides outside of the flanking cWW basepairs. However, sometimes users may have many loops pulled from sequence alignments, and depending how the alignments were made and curated, extra nucleotides may be passed in. Modeling the possibility of extra nucleotides outside of the flanking basepair allows our system to parse these sequences reasonably so that one bad sequence will not ruin the scoring for the whole batch. This is not the only time the possibility of extra nucleotides need to be modeled, however. Initial nodes are also used after cluster and fixed nodes, to model the possibility of insertions after the features modeled by those nodes. Basepair nodes have insertion probabilities built into the node itself, so they do not need to be followed by an initial node.

Finally, all of our loop motif model will end with a hairpin (H) node. A hairpin node only produces terminal symbols, meaning no nodes will follow it. In internal loops, the hairpin node has a 100% chance of generating a "*" symbol to represent the break between strands, and nothing else. This means that internal loop models cannot parse sequences without the strand break symbol. Hairpin nodes cannot be deleted.

In a hairpin motif model, the hairpin node models all nucleotides after the last basepair that follows the nested basepair structure. In Michael Sarver's work, hairpin nodes could just model these nucleotides as fixed positions or variable length insertions. I discovered that crossed basepairs and base triples could occur in hairpins past the nested basepair structures. Because of this, we converted hairpin nodes to MRF nodes, and they now operate nearly identically to cluster nodes

when they are in hairpin motif models, with the only difference being that the hairpin node is still not followed by another node.

4.4 A concrete example : the SCFG/MRF model for IL_95652.3

In this section we will look at an example of a specific SCFG/MRF model, the model for motif group IL_95652.3. The geometry of the instances in IL_95652.3 is known colloquially as a sarcin-ricin motif. The JAR3D model for IL_95652.3 contains examples of each of the node types discussed in Section 4.3. In this section, the probabilities and parameters for the model will simply be given, and in later sections we will discuss the procedure by which we set the parameters and how those procedures are based on observed data.

Motif group IL_95652.3 has 14 core nucleotides, eight on the left strand and six on the right. There are four instances of the motif from 3D structures, and they can be viewed at `http://rna.bgsu.edu/rna3dhub/motif/view/IL_95652.3`. The model is made up of five basepair nodes, a cluster node that models a base triple, and a single fixed node. In addition to these seven nodes that model specific nucleotides, three initial nodes are used to model possible variable-length insertions at the start of the model and after the cluster and fixed nodes, and the model is capped with a hairpin node that produces the strand break character *. A basepair diagram for IL_95652.3, generated by VARNA, with overlays showing how the SCFG nodes model the motif is shown below in Figure 4.4. Note that, from the model's perspective, the top strand is the left strand and the bottom strand is the right strand.

The first node seen in the model is an initial node, in node I_1. All of our loop models start with a node like this, to allow for nucleotides before the first flanking basepair. If produced, these nucleotides would be outside of the internal loop. They are accounted for only due to the possibility of there being an error in input. Initial nodes are comprised of four probability vectors. Two vectors are for the number of nucleotides inserted on each strand, generally with zero being the most likely result. The other two probability vectors are for the distribution of the nucleotides that are produced. The lengths of the insertions as well as the nucleotides produced, if any, are independent.

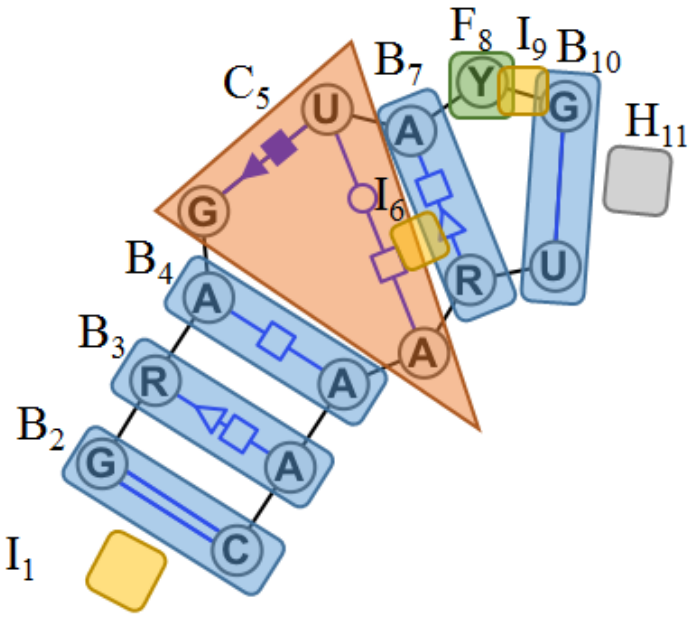


Figure 4.4 Basepair diagram for sarcin-ricin internal loop with SCFG node overlays. This image shows the consensus basepair diagram for IL_95652.3 with overlays indicating the nodes in the SCFG model for the motif group. Basepair nodes are blue, the fixed node is green, the cluster node is orange, initial nodes are yellow, and the capping hairpin is in grey. For this example, the nodes are numbered in the order they appear in the model for easy reference.

Node I_1 has a 0.01% chance of producing one nucleotide, on each strand independently, with the remaining 99.99% probability being for no nucleotides being produced on that side. If a nucleotide is produced, there is an equal probability of it being an A, C, G, or U. These probabilities are always the same because they are never changed by the training data; the training data will never have extra nucleotides outside of the flanking basepairs.

The next node in the model, B_2, is a basepair node used to model the flanking GC Watson-Crick basepair. Basepairs are modeled with a substitution matrix. A substitution matrix is a 4x4 probability matrix that gives probabilities for each of the 16 possible base combinations for the pair. The probabilities in the substitution matrix are set based on not only the frequency of basepairs observed in 3D structures, but also on the geometry of observed basepairs, so that base combinations with similar geometries are given higher probabilities, even if they have not been observed in 3D

instances of the motif. How these matrices are parameterized is discussed in detail in Section 4.5 and in Section 4.6.

The first basepair modeled, in node B_2, is an interesting case. Because it is a canonical cWW basepair, we may expect to see high probabilities for AU, UA, GC, and CG basepairs. This is because these four canonical cWW basepairs are very geometrically similar and interchange frequently. We would also expect to see these probabilities shifted slightly in favor of the GC basepair, as this is the specific basepair observed in the four 3D structures that comprise motif group IL_95652.3. However, one of the bases in the cWW basepair, the C, is also involved in a base-backbone interaction, with the following nucleotide in its chain. If this C is changed to another nucleotide, it will either significantly change the geometry of this base backbone interaction, or the interaction will not take place. For this reason, the model gives roughly 60% probability to the GC combination. Most of the rest of the probability is given to the other canonical cWW basepairs, with about 8% going to the most similar CG basepair, and 7% going to the nearly as similar AU and UA base combinations. The influence of base-backbone interactions on substitution probabilities is explained in Section 4.7.

Basepair nodes can also model insertions that happen after them, so initial nodes are not required after every basepair node. No insertions were observed after B_2, or any of the other basepairs in the model for motif group IL_95652.3, in any of the four 3D instances in the motif group. Because of this, all the insertion lengths and letter distributions seen in basepair nodes in the model are the same. The basepair nodes can produce 0, 1, or 2 nucleotides on each strand, with probabilities of 99.599004%, 0.0039996%, and 0.001%. If insertions do occur, the four bases all occur with equal probability, independent of the number of nucleotides inserted or the letter of any other bases inserted. These insertion probabilities are set based on a rough sense of how often insertions occur in RNA motifs and on the number of 3D instances which have zero insertions at these locations. Since every 3D motif has different physical characteristics, there is no other data to train on to determine the correct probability distribution over possible numbers of insertions.

Node B_3 models a trans-Sugar-Hoogsteen (tSH) basepair. Unlike the cWW basepair modeled

in node B_2, there is some variation in the sequences seen in 3D for this tSH basepair. Three of the instances from 3D structures are GA basepairs, but one is an AA basepair. GA and AA tSH basepairs are fairly geometrically similar to each other, as are a number of other base combinations in this basepair family, so this node is fairly diffuse across the 16 base combinations. The commonly observed GA combination is given a 16% probability, and the AA, CA, UA, CU and AC combinations each get 12 to 13% probability. The AU and GU combinations can also form tSH basepairs. Each is somewhat similar geometrically to one of the observed combinations but not the other, so they are given slightly lower 9 to 10% probability. The remaining base combinations have not been observed to form tSH basepairs in any RNA-containing 3D structures, or, in the case of GG, are very dissimilar to the observed combinations geometrically, and are given low probabilities. Node B_3 produces insertions after the basepair with the same probabilities as node B_2.

Node B_4 models a trans Hoogsteen-Hoogsteen (tHH) basepair. In all 3D instances in the motif group the base combination observed was AA. Even though several other base combinations are geometrically similar to an AA tHH basepair, node B_4's probability of producing an AA is 57%, because the A on the longer strand is also making a base-backbone interaction. The AC combination is geometrically similar to the AA tHH basepair, and preserves the A making a base-backbone interaction, so it has a 10% probability even though it was not observed in any 3D instance. The GC, CG, and UC combinations are geometrically similar, but since they change the base making a base backbone interaction, they are given 6% probabilities. The remaining combinations have very low probabilities. Details on how these basepair substitution probabilities are parameterized are discussed in Section 4.5 and in Section 4.6. Node B_4 also produces insertions after the basepair with the same probability as node B_2.

Cluster node C_5 models the characteristic sarcin-ricin base triple. Because base triples break the nested basepair structure that is well modeled by SCFGs, they are modeled by a node that implements a Markov Random Field (MRF), like the cluster node. A cluster node consists of fixed positions, which represent each core nucleotide modeled by the cluster node, and interactions between these positions.

Cluster node C_5 is fairly simple for a cluster node, and has three fixed positions and two interactions. There are two positions on the left strand and one position on the right strand. There is an interaction between the two positions on the left strand, and it is a cis Sugar-Hoogsteen (cSH) basepair. The other interaction is between the second position on the left and the position on the right and is a trans Watson Crick-Hoogsteen (tWH) basepair. In all 3D structures for this motif group, and indeed for the vast majority of sarcin-ricin motifs in general, the cSH base combination is GU and the tWH base combination is UA. This is because a change in any of the three nucleotides will greatly change geometry of at least one of the two basepairs, because the both involve the second position on the left strand (the U).

In the case of the GU cSH basepair, no other base combination is very similar geometrically, and the G is involved in a base-backbone interaction, so the node gives the GU combination nearly 50% probability. No other combination receives more than six percent probability. The UA tWH basepair is even more specific, and the node gives the UA combination 57% probability, with again no other combination getting more than six percent probability.

To obtain the actual probability that a cluster node produces a set of nucleotides, the probabilities for the involved interactions need to be multiplied together. Because positions can be shared by interactions, the sum over all possible base combinations will not add to 100%. This creates some problems, particularly when comparing models for different motif groups. To combat this, cluster nodes have normalization constants. This constant is the number that the probabilities need to be divided by such that the sum over all possible base combinations does sum to 1. The normalization constant for C_5 is 0.47, and exactly how this normalization constant is calculated is discussed in Section 4.10. We then find that the probability for the node to produce the classic GUA base triple is 0.491 * 0.573 / 0.472, which is about 59.6%. This might not seem like an overly high probability for something that is almost always conserved in nature, but we should remember that our models are intentionally diffuse. A probability of 59.6% over a base triple is actually quite high, for our models. MRF node normalization is covered in more depth in Section 4.10.

Cluster nodes can model insertions between the fixed positions that are modeled in the node.

This is only done where insertions are observed in the 3D structures, and since none were observed in the instances in this motif group, the cluster node does not allow for any insertions.

Cluster nodes cannot, however, produce insertions after the node like basepair nodes, so initial nodes are used to model possible insertions. No insertions were observed after node C_5, so the length and base probabilities for the initial node, I_6, that follows it are the same as the the insertion probabilities in the basepair nodes, like B_2.

Node B_7 models a trans Hoogsteen-Sugar (tHS) basepair. The sequences seen in 3D structures show three AG base combinations in this position and one AA base combination. Node B_7 is an interesting case, because the second base is making a base-backbone interaction, but we do see some variation in the sequences seen in 3D structures, seeing 3 G's and 1 A in that position. This means that when we parameterize the model, we will average three matrices which heavily favor having a G in that position with one that heavily favors having an A in that position. This results in the AG combination having a 31% probability, and the similar UG combination having a 22% probability, even though it has not been observed. The observed AA combination has a 14% probability, and the geometrically similar CA combination has an 11% probability. All other combinations have very low probability. Details on how these basepair substitution probabilities are parameterized are discussed in Section 4.5 and in Section 4.6. Node B_7 produces insertions after the basepair with the same probabilities as node B_2.

Node F_8 is a fixed node. Fixed nodes are used to produce a single base on one strand that is not basepairing with other nucleotides in the motif. Such nucleotides are modeled as a fixed node instead of as a variable length insertion when the base is either stacking on other bases or is making a base-backbone interaction. Node F_8 produces a nucleotide on the left strand between B_7 and B_{10}, a position which is included in the core of the model because it stacks with the bases in B_7 and B_{10} on the left strand. Two Cs and two Us were observed in 3D structures in this position. This results in about a 42% probability of the model producing a C or U, and an 8% chance of it producing an A or G. Fixed nodes do not produce insertions so possible insertions after F_8 are handled by an initial node, I_9. Fixed positions are discussed in more detail in Section 4.9.

Node I_9 is another initial node that models insertions after the fixed node F_8. Because F_8 is modeling a fixed position on the left strand, the probability of the initial node producing nucleotides on the right strand is zero. Node B_7 is already modeling possible insertions in this position. On the left strand, the probabilities are the same as seen in node I_6. This is because in both positions we are accounting for the possibility of seeing new insertions when we have not seen any in our 3D data. Note that if I_9 could produce insertions on the right stand, it is possible that both B_7 and I_9 could produce insertions between the modeled positions in B_7 and B_{10} in the right strand, resulting in undesirable ambiguity. Having initial nodes that occur after fixed nodes only produce insertions on the same strand as the fixed node avoids this ambiguity.

The last basepair in the model, the closing flanking cWW basepair, is modeled by node B_{10}. All four sequences from 3D structures have a GU base combination in this position. GU and UG are near-isosteric to the canonical cWW basepairs (AU, CG, GC, and UA), and therefore sometimes appear in helices and as flanking cWW basepairs. Node B_{10} has a 38% chance of producing a GU combination, and a 30% chance of producing an AC base combination, which is an isosteric substitution. The canonical AU, CG, GC, and UA combinations have around a 4% chance of being produced, and UG about 1.5%, as it is geometrically quite different from GU. Details on how these basepair substitution probabilities are parameterized are discussed in Section 4.5 and in Section 4.6. The closing basepair node B_{10} can produce insertions, but like the insertions produced by node I_1, these nucleotides would be outside of the flanking basepairs, and therefore would not really be a part of the internal loop. Because of this, B_{10} produces insertions following the same probabilities as node I_1, not the probabilities used by the other basepair nodes, such as B_2.

All of our SCFG/MRF motif models end with a hairpin node. If the node is actually modelling a hairpin motif, the hairpin node implements an MRF like a cluster node, and models all interactions after the last nested basepair. In a internal loop, the closing hairpin node simply produces the * chain break character with 100% probability, which is exactly what the final node in the model, H_{11}, does.

4.5 Parameterization of basepair substitution probabilities

Dependencies between positions in an RNA sequence occur as a result of basepairs. Because of this, the most important thing to correctly model is the anticipated sequence for basepairs. The primary tool we use to anticipate sequence variability in basepairs is isostericity, which was introduced in Section 2.3 and Section 4.2. This section will discuss how we use isostericity to make basepair substitution matrices, which model the sequence variability for a basepair. There are four possibilities for each base in a basepair, so the basepairs are modeled with 4x4 matrices of probabilities. We refer to these 4x4 matrices as substitution matrices.

The basis for using isostericity was laid out by the BGSU RNA group in (Stombaugh et al., 2009). In this paper they introduced the isodiscrepancy index, or IDI for short, the first quantitative measure for basepair isostericity. Our basepair sequence variability modeling is based on this IDI score. IDI measures how much the backbone of the RNA would have to move, in units of Angstroms, to change to different nucleotides making the same basepair.

We use IDI to classify substitutions between basepairs into 3 groups. Two basepairs with an IDI less than 2 are said to be isosteric, meaning their geometries are very similar. For example, UA, AU, CG, and GC cWW basepairs are all isosteric. Two basepairs with IDI between 2 and 3.5 are called near isosteric. Near isosteric base substitutions are less common than isosteric substitutions, because they cause some change in geometry. Two basepairs with an IDI greater than 3.5 are said to be non-isosteric. Substitutions between non-isosteric base combinations are rare because of the large geometry change needed to accommodate them. IDI values for the different basepairs are available in the RNA Basepair Catalog, see `http://ndbserver.rutgers.edu/ndbmodule/services/BPCatalog/bpCatalog.html`.

In the 2009 paper, our group studied how often isosteric, near isosteric, and non-isosteric base substitutions are in bacterial alignments of E. coli and Thermus thermophilus. It was reported that non-isosteric substitutions appear in only 2% of cases. This is for both cWW and non-cWW basepairs. Near isosteric base substitutions are more common for cWW basepairs, occurring in 10% of cases, while they remain rare for non-cWW basepairs, again occurring in only 2% of

cases. These probabilities inform how we convert IDI scores into probabilities, as will be shown later in this section.

Figure 4.5 below shows three tWH basepairs, an AG, and AA, and a CC. The connections to the backbone have been highlighted to make the similarities and differences more clear. The AG and AA are very similar and will have a low IDI. They are both very dissimilar to the CC tWH basepair, with which they will have a high IDI. The original AG basepair is shown on the left, with the A colored red and G colored green. The Watson-Crick edge of the A is in contact with the Hoogsteen edge of the G and the backbone connections are on opposite sides of the contacting edges, which is why this is annotated as a tWH pair. An isosteric AA tWH basepair is shown in the middle, with nearly identical backbone connections, highlighted with thick yellow bars. On the right, a non-isosteric CC tWH basepair is shown on the right, with very different backbone connections.

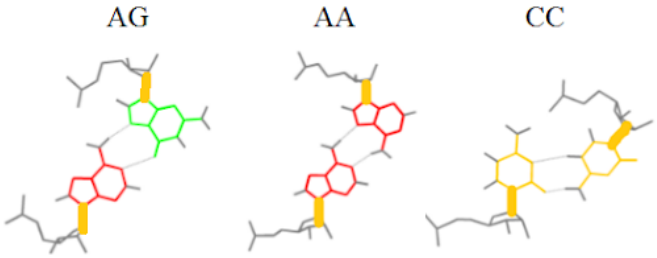


Figure 4.5 Examples of isosteric and non-isosteric substitutions for a AG tWH basepair.

To make a 4x4 substitution matrix for a particular basepair, the starting point is a 4x4 IDI matrix, based on the IDI values for each base combination compared to a particular basepair. For example, the IDI matrix for a AG tWH basepair is shown below in 4.1. AG has an IDI of zero with another AG basepair, because it is the same. AA and AC tWH basepairs are isosteric to AG tWH basepairs, have low values of IDI (1.28 and 1.94, respectively), and have a very similar 3D geometry. They are marked in blue. CG, GG, and UU are near isosteric and are in yellow. UA, UG, and CC are non-isosteric, with CC being particularly dissimilar, as noted above. These are colored orange and red. Several base combinations cannot make a tWH basepair, including AC, AU, CU, GA, GC, and UC. These are colored gray.

	A	C	G	U
A	1.28	X	0	X
C	1.94	5.23	2.26	X
G	X	X	2.67	2.01
U	3.67	X	3.83	2.56

Table 4.1 IDI matrix for a AG tWH basepair. Base combinations that cannot make a tWH basepair are marked with an X.

After the IDI matrix for a basepair is obtained, we convert the IDI values into probabilities. The IDI scores need to be inverted, because basepairs with low IDI are closer to the original basepair and should have higher probability.

A piecewise linear function is used to convert IDI scores. The function is shown in Figure 4.6 below. Isosteric substitutions get the highest scores, then there is a steep drop to near isosteric scores, then a gradual decrease to the minimum score of 0.01 at an IDI of 9. Base combinations that cannot make a particular basepair are given a score of 0.01, as well.

There are a number of reasons that this piecewise linear empirical method was decided upon. Initially, a smooth function, $1/(1+d^2)$, was used. I discovered that this function created substitution matrices that were too diffuse, giving too high probability to non-isosteric basepair substitutions. This resulted in models that were very non-specific, in the space of all possible sequences, a concept which is explored more in Chapter 7. Adjusting the same function to have a steeper drop-off resulted in probabilities that were too low for isosteric substitutions, resulting in models that would not predict many possible sequences that have not yet been seen in 3D. We decided that using a piecewise linear function allows the characteristics of the function to be set quite easily and to reflect known rates at which isosteric, near-isosteric, and non-isosteric substitutions occur, and there are no discernible disadvantages to using a non-smooth function for the inversion. That function is shown below in Figure 4.6.

The inverted scores for the AG tWH matrix are shown in Table 4.2 below. The scores fall between 0 and 1, but need to be normalized to add to 1.

As previously mentioned, base combinations that cannot make the basepair in an IDI matrix are given an inversion score of 0.01, the lowest possible score for a base combination that can

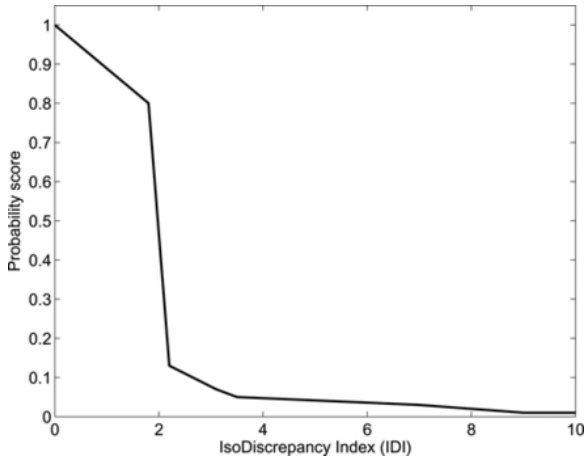


Figure 4.6 Inversion function to translate IDI scores into probabilities. The function is piecewise linear between these (IDI,Score) points: (0,1), (1.8,0.8), (2.2,0.13), (3.1,0.07), (3.5,0.05), (7,0.03), (9,0.01), (10.0,0.01).

	A	C	G	U
A	**0.86**	**0.01**	**1**	**0.01**
C	**0.78**	**0.04**	**0.13**	**0.01**
G	**0.01**	**0.01**	**0.1**	**0.45**
U	**0.05**	**0.01**	**0.05**	**0.11**

Table 4.2 Matrix of inverted isostericity scores for an AG tWH basepair. The base combinations that cannot make the tWH basepair have been given the minimum inversion score of 0.01.

make the basepair. One might think it would be more sensible to assign a score of 0, or some very small number, but doing so actually carries several disadvantages. We want not just to score single sequences against models, but also multiple sequences extracted from multiple sequence alignments. It does this by taking the mean of the log probability for the most likely parse for each sequence. Sequence alignments can contain errors, both in sequences and the alignment itself. It is also possible that not all sequences in an alignment are making the same structure in the position where a particular loop is. If an overly low score was given for non-isosteric substitutions or substitutions that cannot make the pair, a single bad sequence could ruin the overall score for an alignment of loop sequences that otherwise fit the group very well. A score of negative infinity (the logarithm of 0) is quite possible, and would be given to any sequence the model cannot produce, so one erroneous sequence could quite easily ruin the matching for a group of loop sequences.

These SCFG/MRF models are also used to align sequences to motif groups. If the probability

for a combination of bases was too low, the model might find an alternative parse for the sequence that would not make much sense in an alignment. It would be strange and not useful if an alignment suggested that a basepair was deleted from an internal loop and then a nucleotide was inserted on either strand. This is how the models would align a sequence to a model if the probability in a basepair substitution matrix was overly low, however.

4.6 Using multiple instances of a loop structure

We have discussed how the geometry of a single basepair is converted into a substitution matrix using IDI scores, but we can make also make models based on multiple instances of the same motif geometry. In this section we will cover how we combine information from multiple instances of a motif when making a model.

When there are multiple instances of a basepair, a substitution matrix is calculated for each instance following the steps outlined in Section 4.5. These matrices are also adjusted for base-backbone interactions, as described later in Section 4.7. It is possible that not all instances of the motif will have the same base-backbone interactions, so this step might cause instances with the same nucleotides to have different substitution matrices. Finally, all the matrices are averaged together.

Although the basepair probabilities in our models are based primarily on the geometry of base-pairs through isostericity, we also use more traditional statistical methods to adjust these probabilities based on the existing data. The more existing examples from 3D structures that we have for a particular motif group, the more that probabilities should be shifted towards the distribution seen in 3D sequences. A weighted average is used to combine the substitution matrix obtained from isostericity and adjusted for base-backbone interactions with one based on the frequencies of sequences observed from 3D structures. The factor used to weight the instances matrix is (N/(N+100)), where N is the number of sequences observed from 3D structures. So, if the motif group is a singleton with one instance from 3D, 1% weight will be given to the frequency matrix and 99% weight will be given to the isostericity matrix. If there are 100 instances from 3D structure in the motif group, half weight will be given to both the frequency matrix and the IDI matrix. The largest motif group

in release 1.13 is motif group IL_97217.11, which has 248 members, so it will be heavily weighted towards the frequency matrix.

4.7 Parameterization of base-backbone interactions

Base-backbone interactions occur between atoms in a base and atoms in the RNA backbone that connects the bases together in a chain. Studies have shown that bases involved in base-backbone interactions are strongly conserved (Stombaugh et al., 2009; Zirbel et al., 2015), because any change in the base that is interacting with the backbone tends to create a large change in the geometry of the interaction. This is because base-backbone interactions tend to use very specific parts of the base involved in the interaction. There are two types of base-backbone interactions, base-phosphate and base-ribose. Most positions making base-phosphate interactions are over 90% conserved. Because of this, any bases making the base part of a base-backbone interactions have all probability matrices and vectors associated with them shift strongly towards conserving the observed base. We do not, however, push the probability of conservation all the way to the observed 90%. This is partly because of our desire for our models to be diffuse, but also because many of the bases in our study are making basepair interactions as well, which also contribute to conservation. The atoms on the backbone don't vary with the base connected to the backbone, so base-backbone interactions can be thought of as single base interactions.

To model the effect of base-backbone interactions, the Dirichlet priors are adjusted by a constant factor, weakening their effect, before being added to count vectors. Bases involved in base-phosphate interactions are more highly conserved then those involved in base-ribose. Accordingly, the factor used for base-phosphate interactions, 7, is slightly larger than the one used for base-ribose interactions, 5. These factors are used to weaken the prior distribution used to smooth out the probability matrices and vectors, as well as strengthen columns in substitution matrices. Without further studies on base-backbone interactions, we do not have the data to estimate more specific parameters.

For example, say that a motif group has fixed position. This motif group has only one instances in 3D structures, and the position is a G. If the base was not making a base-backbone interaction,

the base count vector would be [0, 0, 1, 0] and the base prior vector would be [0.5, 0.5, 0.5, 0.5], resulting in a combined vector of [0.5, 0.5, 1.5, 0.5] and a normalized vector of about[0.222, 0.222, 0.333, 0.222]. If the position makes a base ribose interaction weakens the prior by a factor of five to [0.1, 0.1, 0.1, 0.1], resulting in a combined vector of [0.1, 0.1, 1.1, 0.1] and a normalized vector of about [0.07, 0.07, 0.79, 0.07], nearly 80% conserved. If the position is making a base phosphate interaction, the normalized vector would be [0.055, 0.055, 0.835, 0.055], reflecting the higher conservation rates for bases making base-phosphate interactions. In both cases the observed base is shifted to be more likely to be conserved, but not the the extent that we observed in the data.

4.8 Parameterization of insertions

Variable length insertions occur in RNA loops when a base or multiple bases "bulge" out from a loop. These bases are not making any sort of interaction (basepair, base-backbone, or stacking) with any of the other bases in the loop.

Four nodes can produce variable length insertions. Basepair nodes model a variable length insertion on both strands. Initial nodes can also produce variable length insertions after both strands, and sometimes are used to produce insertions on only one strand. Cluster and hairpin nodes, can produce variable length insertions that have been observed in sequences from 3D structures between positions modeled by the MRF.

Variable length insertions are parameterized differently depending on whether or not they have been observed in 3D structures at a particular position. If no insertions have been observed, then we estimate the probability insertions might occur at that position based on percentage of loop positions that have insertions in the RNA 3D Motif Atlas. If any of the 3D structures has variable length insertions at the position, then we know that insertions are biologically allowed there, and we switch to parameterization that leans heavily on the sequences that have been observed in 3D structures.

Regardless of the node that is modeling them, all variable length insertions are parameterized in the same manner. If no insertions are observed in sequences from 3D structures, a fixed distribution depending on the number of sequences in the motif group is used. All of these distributions have

at least a 99% chance of having no insertions, with a slight probability of 1 insertion and an even smaller chance of 2 insertions. If there are more sequences from 3D structures the probability goes down, to a final distribution for motif groups with four or more 3D instances. If insertions do occur, they will have an equal chance of being A, C, G, or U.

When at least one sequence with an insertion is observed in a 3D structure, the probabilities for insertions are based on the sequences from 3D structures. A vector of counts for sequence lengths of insertions is created, and then slightly smoothed out so that observed insertion length also slightly increases the probability of seeing slightly longer or shorter sequences. The smoothing effect is lessened the more data is available (i.e., the more sequences from 3D structures are in the motif group). Specifically, each instance will add 1 to the count for its length, 1/(20*L) for counts one more or one less, and 1/(400*L) for the count two more than the observed length, where L is the number of instances from 3D structures. The vector is then normalized into probabilities. The probabilities for inserted bases are also updated based on sequences from 3D structure. Counts for the four bases are tallied, then a prior of 0.5 is added to each count, then the vector is normalized.

4.9 Fixed bases

Not all conserved bases in motif groups are involved in an interaction with another base. These bases are either involved in base backbone interactions, or are stacked with another base. In either case, the base is modeled as a single base interaction. This could be either a single base interaction in a cluster or hairpin node, or a fixed node if the base is not inside an MRF. The concept of fixed bases is something we have come to understand since Michael Sarver's dissertation, and are a new addition to our methodology. Either way, a vector with the frequencies for the different bases is combined with a prior weight vector and then normalized for probabilities. The prior vector adds 0.5 to each of the frequency counts, unless the fixed base is involved in a base-backbone interaction. If the base is making a base backbone interaction, the influence of the prior is substantially reduced, to reflect the high conservation rate seen for bases making base-backbone interactions, see Section 4.7 for details on how this is done.

4.10 Markov random field node normalization

For Initial, Basepair, and Fixed nodes, the sequences that can be generated are built up from simple operations such as generating a basepair or generating an insertion length and then a letter A, C, G, or U for the inserted bases. These are done independently, and one can directly calculate the probability of each generation history and these probabilities add up to one. This is not the case for Cluster and Hairpin nodes that utilize Markov Random Fields (MRFs). This section will discuss how such nodes are normalized. Normalization of these nodes is important in comparing the results between different models.

The nodes that make use of MRFs (cluster and hairpin nodes) are not automatically normalized. Each basepair in an MRF node has its own substitution probability matrix. So, for example, he probabilities for a base triple will be determined by two different substitution probability matrices, one for each basepair. When calculating the probability score for a sequence making a triple, the probability scores for each basepair are multiplied together. However, because one of the bases is included in both basepairs, the sum of probability scores over all possible sequences for the triple will be less than one.

This is an issue when one wishes to compare a sequence's probabilities against different models, because models using cluster nodes will have inherently lower scores, and the larger the cluster node the lower the scores will be. Normalization can easily remedy this problem, however. If, for each cluster and hairpin node in a model the probabilities over all possible sequences are summed, a normalization constant for the node is obtained. If the scores for sequences against this model are divided by this constant, the sum of probabilities over all possible generation histories will be one, creating a level playing field.

Consider, for example, a simple cluster node for modeling a base triple. The triple from IL_95652.3, discussed in Section 4.4, is a good example. The node will contain two basepairs, one between nucleotides 1 and 2, and another between nucleotides 2 and 3. Substitution matrices for these basepairs are shown below in 4.3 and 4.4.

There are 64 different possible sequences for this triple, because the 2nd base in the first base-

GU cSH	A	C	G	U
A	0.054	0.056	0.003	0.003
C	0.052	0.061	0.001	0.061
G	0.037	0.02	0.041	0.491
U	0.005	0.054	0.001	0.061

Table 4.3 Table of substitution probabilities for a GU cSH basepair. This table shows the substitution probabilities for the basepair that forms the first half of the base triple in IL_95652.3.

UA tWH	A	C	G	U
A	0.025	0.005	0.026	0.005
C	0.053	0.044	0.035	0.005
G	0.005	0.005	0.063	0.025
U	0.573	0.005	0.066	0.057

Table 4.4 Table of substitution probabilities for a UA tWH basepair. This table shows the substitution probabilities for the basepair that forms the second half of the base triple in IL_95652.3.

pair must be the same as the 1st base in the second basepair. This means the sum over all 64 different combinations will be less than 1. 4.5 shows the probability for each possible sequence, as well as the sum of the probabilities for each sequence.

The sum of the probabilities over all generation histories is the normalization constant. Dividing the original probabilities by this constant, a set of proper probabilities that sum to 1 is obtained. Thus, for example, the probability of the GUA base triple becomes 0.281052 / 0.471704 = 0.595823.

For some MRF nodes, however, simply iterating over all sequences in the cluster node is not required to calculate the normalization constant. In some cases, the cluster node can be broken down into smaller parts to simplify the process. Some cluster nodes can produce so many generation histories that they cannot realistically be enumerated. This method makes it possible for all cluster nodes used by JAR3D thus far to be normalized. Motif group IL_16415.2 provides a good example; its basepair diagram is shown below in Figure 4.7. The 8 positions in the cluster node result in 4^8; over 65,000; possible sequences.

It is, however, not necessary in this case to even iterate over all of the sequences that the cluster node can generate to find the normalization constant for it. Instead, each independent group of

Seq.	**Prob.**	**Seq.**	**Prob.**	**Seq.**	**Prob.**	**Seq.**	**Prob.**
AAA	0.00138	CAA	0.00133	GAA	0.00094	UAA	0.00012
AAC	0.00029	CAC	0.00028	GAC	0.0002	UAC	2.5E-05
AAG	0.00142	CAG	0.00137	GAG	0.00097	UAG	0.00012
AAU	0.00029	CAU	0.00028	GAU	0.0002	UAU	2.5E-05
ACA	0.00296	CCA	0.00322	GCA	0.00107	UCA	0.00288
ACC	0.00247	CCC	0.0027	GCC	0.0009	UCC	0.0024
ACG	0.00198	CCG	0.00215	GCG	0.00072	UCG	0.00192
ACU	0.0003	CCU	0.00033	GCU	0.00011	UCU	0.00029
AGA	1.6E-05	CGA	3E-06	GGA	0.00022	UGA	3E-06
AGC	1.6E-05	CGC	3E-06	GGC	0.00022	UGC	3E-06
AGG	0.00019	CGG	0.00004	GGG	0.00259	UGG	0.00004
AGU	7.6E-05	CGU	1.6E-05	GGU	0.00103	UGU	1.6E-05
AUA	0.00144	CUA	0.03507	***GUA***	***0.28105***	UUA	0.0349
AUC	1.3E-05	CUC	0.00033	GUC	0.00262	UUC	0.00033
AUG	0.00017	CUG	0.00404	GUG	0.03234	UUG	0.00402
AUU	0.00014	CUU	0.00351	GUU	0.02816	UUU	0.0035
						Total	0.4717

Table 4.5 Table of product of probabilities for a basetriple. This table shows the products of probabilities for all possible basetriples using the probabilities from the substitution matrices in Table 4.3 and Table 4.4. Note that the highest entry, GUA, is in bold and is italic.

basepair interactions can be iterated over separately, and the resulting constants can be multiplied to give the overall normalization constant. The cluster node in IL_16415.2 contains eight positions. One of the positions is not involved in any basepairing, and can safely be ignored. The remaining seven create two disjoint sets of base triples. The normalization constants for these two sets can be calculated separately. Positions 2, 5, and 7 are involved in a base triple, the simplest situation which requires normalization, and only requires 64 sequences have their probabilities calculated. Positions 1, 3, 4, and 8 are in another set, connected by two base triples between positions 1, 3, and 4 and 1, 4, and 8. This four position set will require the calculation of probabilities for only 256 sequences. These means the normalization constant for the node can be found by calculating probabilities for only 320 sequences, not 65,000.

The ability to break down cluster node normalization to independent position sets allows us to now properly normalize all cluster and hairpin nodes, when it was previously impossible for us to do so for a number of larger nodes.

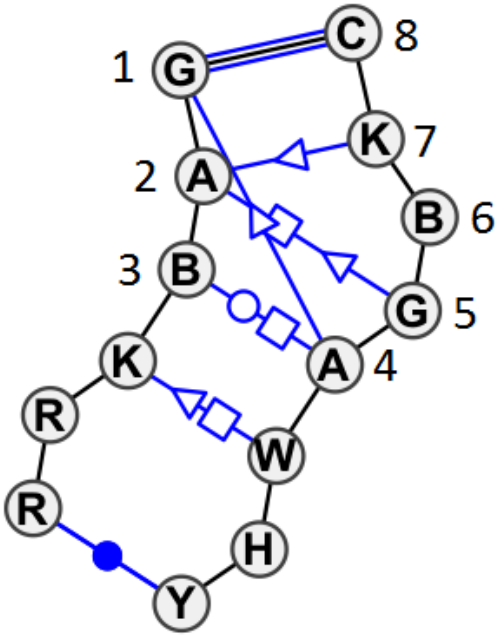


Figure 4.7 Basepair diagram for IL_16415.2. IL_16415.2 is an internal loop which has an SCFG/MRF model that contains a large cluster node. The positions in the cluster node have been numbered. Due to basepair crossing, IL_16415.2 contains a large cluster node covering 8 nucleotides.

CHAPTER 5 JAR3D SOFTWARE AND WEB SERVERS

This chapter will introduce and discuss the software package we developed to implement our SCFG/MRF models for RNA loop sequence variability. The software is called JAR3D, which stands for **J**ava **A**lignment of **R**NA in **3D**. The source code for JAR3D is available on GitHub, and both a command line tool and a Web Server have been made available to researchers.

The JAR3D software package consists of two main parts. The first is a set of Matlab programs that analyze RNA 3D structures to produce SCFG/MRF models for RNA loop sequence variability, based on the RNA loop motif groups in the RNA 3D Motif Atlas. The second part implemented and Java and is used to compare novel sequences to JAR3D models.

Section 5.1 introduces the JAR3D software suite. Section 5.2 describes how the JAR3D java code compares novel sequences to JAR3D SCFG/MRF models. Section 5.3 discusses using JAR3D to align sequences to a loop geometry. Finally, Section 5.4 discusses using JAR3D to match novel loop sequences with unknown geometry to possible geometries from 3D structures using the JAR3D SCFG/MRF models.

5.1 Introduction to JAR3D

JAR3D, short for Java-based Alignment of RNA using 3D structure, is a collection of programs that make and score probabilistic models of the sequences which form RNA 3D structures. These models utilize Stochastic Context Free Grammars (SCFGs) and Markov Random Fields (MRFs) and are referred to as SCFG/MRF models. A JAR3D SCFG/MRF model for an RNA 3D structure can be used both to assess the likelihood that an RNA sequence forms a 3D structure and to align a sequence to a 3D structure.

Alignment of RNA sequences to an RNA 3D structure via a JAR3D model essentially means assigning nucleotides in the sequence to positions in the 3D structure. There are two primary reasons one might be interested in aligning RNA sequences to 3D structures. The first is to align sequences to each other, and the second is to infer 3D structure of RNA motifs. The JAR3D model

acts as the link between the sequence and the 3D structure; the model is based on the 3D structure and sequences can then be aligned to the model. RNA structures can have insertion of nucleotides, deletion of nucleotides, and changes in structural elements, so this is not always a straightforward problem. The JAR3D Java code can use either the CYK (Cocke–Younger–Kasami) (Younger, 1967) algorithm to produce a most likely alignment between the structural elements of an RNA 3D structure and a given RNA sequence. It can also use the inside-outside algorithm (Baker, 1979) to calculate the total probability that a model will produce a given sequence.

JAR3D models can be used to align novel sequences to 3D structures when a homologous 3D structure has been solved. For example, someone might be interested in studying the ribosome of a bacterium for which no ribosomal 3D structure has been solved. However, many 3D structures have been solved for the ribosome of E. coli. A JAR3D model could be used to align the sequence of the novel bacterial ribosome to the E. coli ribosomal structure, which should be very similar to the ribosome of the novel bacterium. Areas which align poorly might be of particular interest for future study.

Another use for JAR3D is to match sequences of RNA internal loops (IL) and hairpin loops (HL) to possible 3D structures. RNA has a wide range of cellular functions, and the 3D structure of an RNA is key to the performance of that function. Matching sequences of RNA internal loops and hairpin loops to known 3D motifs is an important step on the path to predicting the 3D structure formed by a full-length RNA sequence.

JAR3D is primarily implemented through two programming languages. The JAR3D models are produced by a set of Matlab programs. The models are used to parse and align sequences to models by a set of Java programs. The code is freely available and easily accessed at `https://github.com/BGSU-RNA/JAR3D`.

JAR3D models for RNA motifs are more diffuse, by design, than the sequence variation typically observed in the few 3D instances of the motif that have been observed experimentally. This is so that they can catch potential new instances of a motif. Because of this, the probabilities for even the most likely sequences for a JAR3D model to generate are usually quite small. Because

the probabilities are quite low, we take the natural logarithm of the probability of the model generating the sequence. Because we are interested in sequence alignments, we usually do not use the total probability of a model generating a sequence, but rather the probability for the most likely generation history. We call the logarithm of the maximum probability generation history result an Alignment Score, because it is a result of aligning the sequence to the model, and we use the word score to avoid implying a relationship to the probabilities found in nature.

The work that would eventually become JAR3D started as Matlab code, written by Michael Sarver as part of his 2006 dissertation work (Sarver, 2006). Unfortunately, the parsing was too slow, even using the CYK algorithm, so that part of the system was ported to Java, and JAR3D was born. Model generation does not suffer from the same need for efficiency and speed that parsing does, since models are generated infrequently, so that code was left in Matlab. The initial work of translating JAR3D's implementation of the CYK algorithm into Java was done by Meg Pirrung. I picked up were Meg left off, fixing bugs, making parts of the the JAR3D code more efficient, adding many new features, and, most importantly, modifying it for modeling 3D loop sequence variability.

One of the major changes I had to make to JAR3D was adding parsing for both strand orders for internal loops. Because internal loops are two stranded, a sequence could possibly form the geometry of a JAR3D model in two different ways, and both ways need to be checked to find the best parse for a sequence. To better understand this need, an example will be shown using motif group IL_85647.3, a motif group for the sarcin-ricin motif. Sarcin-ricin motifs have a characteristic AGUA sequence within their longest strands. This sequence appears in every instance of IL_85647.3 seen in 3D. The sequence CUC**AGUA**U*AGAACCG is seen twice in 3D instances of IL_85647.3. However, if the other strand order, AGAACCG*CUC**AGUA**U, was given as input to JAR3D, it could score very poorly, because the signature sub-sequence is on the wrong strand. This is why we need to check both strand orders for internal loop sequences.

I added code to JAR3D to reverse strand order for internal loop sequences given as input and score both orientations. After both orientations have been parsed, the orientation with the highest

maximum probability generation history is selected as the orientation to use when comparing the sequence to the model, and is recorded for output.

Anothing major addition I made to the JAR3D code was the addition of a command line interface. JAR3D functions initially just produced Java objects or text files, and had to be called from Matlab or within a Java Development Environment such as Eclipse. I produced two Java jar files, one for use with the JAR3D web server which is discussed more below, and one designed to be called independently from the command line.

The command line version of JAR3D, the main function of which can be viewed at `https://github.com/BGSU-RNA/JAR3D/blob/master/src/main/Java/edu/bgsu/rna/jar3d/cli/Main.Java`, takes fasta-formatted internal or hairpin loops, as a group or a list, and produces simple but highly informative csv formatted output. Fasta format is a text-based format for representing nucleotide sequences. In fasta files, each nucleotide sequence is prefaced with a descriptor line which indicates where the sequence came from. For example, the sarcin-ricin sequence from the strand order example above could be expressed in fasta format as shown below. Multiple sequences can be expressed in a fasta file, each preceded by its own descriptor line, as shown below.

```
>Sarcin-Ricin Sequence 1
CUCAGUAU*AGAACCG
```

The command line version also needs to be given a text file that lists the JAR3D models that the sequences in the fasta file will be parsed against. This text file could list models from an entire Motif Atlas release, a single model, or something in between. Two csv files are created as output, one that has “sequence level” results for each sequence in the input file and one that has “loop level” results, which combine results for all the sequences in the sequence file against all models in the model list file. The format and contents of this output is explained in detail in Appendix F of (Zirbel et al., 2015).

The JAR3D command line tool is distributed as a JAR file. A .jar file is an archive of Java files that make Java code easy to distribute, and they can be run on almost any platform using the

Java Virtual Machine. The JAR3D command line JAR file is fast and powerful, and is useful for experienced users who want to process large amounts of data. It does, however, requires the user to have command line expertise, and for the user to download JAR3D model files, or make them themselves.

Besides the command line tool, JAR3D is also available through a web server (Roll et al., 2016). The JAR3D web server provides a convenient, user friendly way for users to access JAR3D. The JAR3D web server can be found at `http://rna.bgsu.edu/jar3d`. I worked extensively on the JAR3D webserver, particularly the Java jar file that it uses. Anton Petrov helped with the user interface and Blake Sweeney developed an input/output system for the both the jar file used by the command line tool and the webserver, and also helped with the user interface of the web server. A major component of the JAR3D webserver which I developed is a routine to show the alignment of input sequence(s) to a specified JAR3D model.

The input page for the JAR3D web server allows for a variety of input formats. All inputs can either be fasta formatted or sequence only, with each sequence on a new line. Sequence only inputs are given generic fasta headers by the PHP code that processes the input, such as "Sequence 1, Sequence 2, ...". Both hairpin loops and internal loops can be input as single loops, either as single sequences or as multiple sequences from homologous molecules. Full molecule sequences can also be input, provided that they are aligned and a dot-bracket secondary structure indicating the Watson-Crick basepairs is also provided; in this case, the web server separates out the internal and hairpin loops and submits their sequences to JAR3D, showing the results for all loops on the results page. Appendix A shows an example query on the JAR3D web server input page.

The loop level output page generated after submitting a query to the JAR3D web server shows the top scoring groups, and provides a link to view all results if desired. There are also links available that will produce alignments of input sequences to a specific group. Using JAR3D for alignments is discussed more in Section 5.3. The loop level output page generated by the input shown in Appendix A is shown in Appendix B. An alignment of the input sequences to a specific JAR3D model is shown in Appendix C.

5.2 Parsing sequences against JAR3D models

In Chapter 4, we discussed how we build JAR3D models for the sequence variability for an RNA motif. In this chapter, we will discuss how JAR3D models can be used. Most use cases for JAR3D models involve taking an RNA loop sequence and seeing how likely it is that a JAR3D model would generate that sequence, and how it would generate the sequence. This process is called parsing the sequence against the model.

There are actually two relevant probabilities one might want to calculate when parsing a sequence against an SCFG model. This is because it is possible for SCFGs to produce the same sequence in different ways. Each method of producing a particular sequence is called a generation history. The easiest to understand is total probability, which is the sum of the probabilities of all possible generation histories for the sequence. Or, more simply put, the probability that the particular sequence is produced if the model is asked to produce a random sequence. The second probability which parsing a sequence against a model can produce is the maximum generation history probability, the probability of the most likely generation history for the sequence. We call this the maximum probability for short. The latter is less computationally complex to find, and for our models is often very close to total the probability, because typically there is only one straightforward way for a JAR3D model to generate a given sequence, and a mixture of deletions and insertions are needed in every other generation history.

Because even our fairly small JAR3D SCFG/MRF models can produce billions of sequences, it is not feasible to save these probabilities for every sequence that every JAR3D model can produce. This means that it is necessary that these probabilities can be calculated quickly and efficiently upon request. This is the primary reason that JAR3D models are restrictive SCFGs, because there is an algorithm to quickly and efficiently parse sequences against stochastic context free grammars. The Cocke–Younger–Kasami algorithm, commonly abbreviated as the CYK algorithm, is a very efficient parsing algorithm for context free grammars (Younger, 1967). JAR3D employs the CYK algorithm to produce maximum generation history probability. A second algorithm, the Inside-Outside algorithm, is also implemented to calculate total probability if needed, but using CYK to

produce maximum generation history probabilities is JAR3D's default.

It is important to note that, although parsing a sequence against a JAR3D model produces a probability, that probability is not meant to estimate or reflect the probability of finding that sequence making a particular RNA 3D motif in nature. We simply do not have the available data to estimate such probabilities. For the applications we have for JAR3D models, though, such probabilities are not needed. We simply need to be able to compare probabilities of the same sequence being generated by different models, or to compare probabilities of multiple sequences being generated by the same model. Because of this, it is important that JAR3D models are parameterized in a very consistent manner.

5.3 Using JAR3D for alignments

One of the most basic uses for JAR3D models is to align a sequence or set of sequences to a 3D geometry using a JAR3D model for the sequences consistent with the observed geometry. One of the advantages of a guide-tree SCFG approach is that the CYK algorithm, which finds the highest probability parse for a sequence by an SCFG model, also easily produces a traceback for that parse. The traceback for the parse of a JAR3D model creates a direct mapping between the sequence and the nodes in the SCFG, which have a natural mapping to the 3D structure that the model is based on. Thus, every nucleotide in the sequence will be mapped to either a position in the 3D structure or to an insertion location between two positions.

The JAR3D web server can also produce alignments of input sequences to specific motif groups when requested from the loop level output page. These alignments show exactly what each nucleotide in the input sequence is expected to do in the matched structure, using the maximum probability generation history. An alignment of the input sequences to a specific JAR3D model is shown in Appendix C.

5.4 Using JAR3D to match loop sequences to 3D structures

Another use for JAR3D is to match new loop sequences to possible motif 3D structures. For this problem, we have a loop sequence or a set of homologous loop sequences, and we want to know

which, if any, known 3D structures the sequences are able to form. This is a difficult problem for a number of reasons. Firstly, there are hundreds of known 3D structures that the sequences could form. This makes matching a sequence to the correct group more difficult, and makes false positives likely. Secondly, a sequence could form a novel 3D structure, one that has not been seen before in 3D structures, so it might match nothing. Control of false positive matches is addressed in Chapter 6. Third and finally, some motif sequences might be able to form more than one 3D structure. These are sometimes called polymorphic motifs, meaning that the structure of the loop will change depending on the situation that molecule is in. Many instances of this have been observed across the RNA 3D Motif Atlas.

CHAPTER 6 JAR3D FALSE POSITIVE CONTROL AND VALIDATION STUDIES

In this chapter we will talk about how we control for false positives in JAR3D and describe some validation studies we have done to test JAR3D's effectiveness. Because the RNA 3D Motif Atlas contains over 270 internal loop geometries and over 230 hairpin loop geometries to compare novel sequences to, it is very important to control for false positives. Section 6.1 outlines how we generate random RNA loop sequences to use as a distractors in controlling for false positives, and Section 6.2 talks about the methods we developed to control for false positives using those random sequences. Section 6.3 looks at a study of JAR3D's performance on the recognition and matching tasks for two alignments of sarcin-ricin loops. Finally, Section 6.4 compares the ability of JAR3D models to recognize novel loop sequences to a similar tool, RMDetect (Cruz and Westhof, 2011).

6.1 Randomly generating sequences for false positive control

Every possible RNA sequence can be scored against a given JAR3D model. How do we tell if the sequence scores well enough to possibly form the associated 3D structure? This is especially relevant when we score a sequence against hundreds of JAR3D models; we want to control for the possibility of false positive matches. To develop a system to control for false positives when using JAR3D, we need a population of distractor sequences that look like RNA loop sequences, but should not be accepted by the given model. We considered using the loop sequences from other 3D structures as the population of distractor sequences, but sequences vary so much in size that many would be trivial to reject. This problem also precludes using alignment data from other 3D structures for false positive control. So, for data that should not be accepted, we decided it would be best to generate random loop sequences of the appropriate length.

We generated a set of random loop sequences for each loop length observed in the RNA 3D Motif Atlas. We also included observed lengths with an insertion on either or both strands, since models allow for insertions beyond what is observed. For internal loops, since both strand orderings are considered when comparing a sequence to the JAR3D models, we only look on strand

lengths with the shorter strand on the left. This resulted in 93 different pairs of lengths for internal loops, ranging from (2,3) to (13,17).

In addition to sequences being of the correct length, we also took some statistics from the loop sequences in the RNA 3D Motif Atlas to insure a similar distribution of nucleotides. Because we include the flanking cWW basepairs in loops, the flanking basepair positions in random sequences (and their probabilities) are always CG (0.2761), GC (0.3596), AU (0.1121), UA (0.1251), GU (0.0639), UG (0.0632), which reflect the frequencies with which those flanking pairs are seen in loop sequences seen in 3D structures.

Positions in the interior of the sequences are generated using a Markov chain trained on the sequences from 3D data. This means that distribution of a nucleotide in the interior is dependent on the previous nucleotide, which echos what is seen in helices based on the Turner energy parameters. The distribution for the first nucleotide in the interior of the sequence is A (0.2653), C (0.1527), G (0.3227), U (0.2593). The transition matrix that is used afterwards is shown below in Table 6.1. The end result is sequences that look like loop sequences in general, but have no restrictions on sequence variability from basepair or base-backbone interactions, so they have no reason that they should match any particular 3D motif.

	A	**C**	**G**	**U**
A	0.459	0.124	0.236	0.181
C	0.432	0.191	0.225	0.152
G	0.508	0.120	0.175	0.197
U	0.434	0.139	0.243	0.184

Table 6.1 Transition probabilities for random sequences for JAR3D's false positive control.

6.2 False positive control, alignment score deficit, and cutoff score

Because there are so many possible motif geometries, good false positive control is essential for both the recognition and the matching problems. We considered a number of different options before deciding on a method to accept or reject a given sequence for a given model. We use what we call an acceptance region and a measure called the cutoff score, which will be discussed in this section. The acceptance region is based on two numbers; alignment score deficit (calculated from

a JAR3D model) and interior edit distance from the given sequence to sequences of 3D instances. As will be illustrated in Figures 6.1 and 6.2 below, both numbers are informative about the match of a sequence to a motif group.

An alignment score is simply the max log probability score for a sequence, which is the log of the maximum probability parse for the sequence. Alignment scores are negative numbers, and their size depends on how many nodes there are in the JAR3D model. The alignment score deficit is the difference between the best alignment score of sequences from 3D instances of the motif and the alignment score of the given sequence. Alignment score deficit is a positive number, usually between 0 and 20. It roughly tallies the effect of sequence changes compared to sequences seen in 3D structures; lower numbers indicate a better match. Using the alignment score deficit instead of the straight alignment score makes it easier to compare alignment scores between different motif groups.

Edit distance is the minimum Levenshtein edit distance from a given sequence to the sequences observed in 3D structures for the motif group. Interior edit distance is the edit distance, excluding the flanking basepairs. We ignore the flanking basepairs because for many loop geometries, the flanking cWW basepairs can change to other canonical cWW basepairs without changing the geometry of the loop.

After extensive manual examination of the scores of 3D sequences, sequences from multiple sequence alignments, and randomly-generated distractor sequences, we set a maximum acceptable interior edit distance of 5 and a maximum acceptable alignment score deficit of 20. Sequences beyond these limits are simply too different from the observed 3D sequences to make any claim that they can form the 3D structure. The acceptance region is further constrained by the inequality $Deficit + 3 * EditDistance <= k$, where the constant k is specific to each motif group. For each JAR3D model, we set k so that 4% of the sequences from the set of randomly generated sequences, with interior edit distances between 1 and 5 and alignment score deficit less than 20, fall within the cutoff region. Also, the minimum value of k is set to 9.5 and the maximum value is set to 25 to avoid overly small or large cutoff regions; manual examination determined that these were

necessary for JAR3D models for very small and very large motifs, respectively. The coefficient 3 that is multiplied by edit distance was originally estimated separately for each motif group using linear regression (on deficit and edit distance as predictors of being a known sequence or random sequence as the response variable), giving numbers sometimes below and sometimes above 3. However, as it was not clear what features of the motif called for higher or lower values of this coefficient, and as the parameter selection needs to run autonomously without human supervision, it was safer to simply use 3 as the consensus value. This can be refined with additional study. In summary, a sequence falls within the cutoff region if its interior edit distance is less than or equal to 5, its alignment deficit is less than 20, and if $Deficit + 3 * EditDistance <= k$.

We also calculate a cutoff score to quantify where in the acceptance region the sequence falls. A sequence with 0 alignment score deficit and 0 interior edit distance will have a cutoff score of 100. Cutoff score falls linearly to the line that defines the acceptance region, so if a sequence's $Deficit + 3 * EditDistance = k$ exactly, the sequence's cutoff score will be 0. A negative cutoff score indicates that the sequence falls outside the acceptance region and is rejected by the group. Importantly, the interpretation of the cutoff score is the same for all motif groups, making it easier for users of the JAR3D website to interpret the results.

Figures 6.1 and 6.2 below show the acceptance regions for IL_95652.3 and IL_73276.5. The grey areas are the acceptance regions, and the line in the middle of it denotes a cutoff score of 50. The coordinates for three sets of sequences are also shown on the graphs. Blue Xs indicate sequences from 3D structures, black dots indicate sequences extracted from an associated multiple sequence alignment, and red dots indicate randomly generated distractor sequences. Ideally, most of the black dots will fall inside of the cutoff region. However, alignments are not perfect, and due to errors in the alignments, it is likely that many of the sequences in the alignment extract should not be accepted by the models. A full analysis of JAR3D's performance on sequences extracted from multiple sequences alignments for IL_95652.3 is available in Section 6.3, and analysis of more cutoff graphs is available in Zirbel et al. (2015).

For any one particular motif group, the acceptance region is set to accept 4% of the randomly

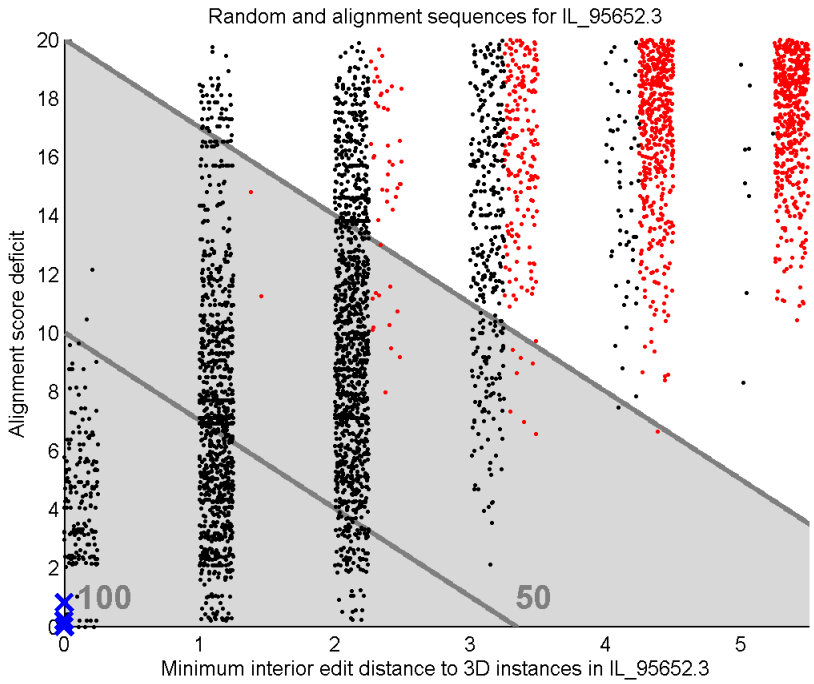


Figure 6.1 The acceptance region for IL_95652.3. Blue Xs are sequences from 3D structures, black dots are sequences from an alignment extract, and red dots are randomly generated sequences. Note that the black dots have been shifted to the right by a uniformly distributed random variable between 0 and 0.25 so that they are all visible; red dots have been shift to the right by numbers from 0.25 to 0.5.

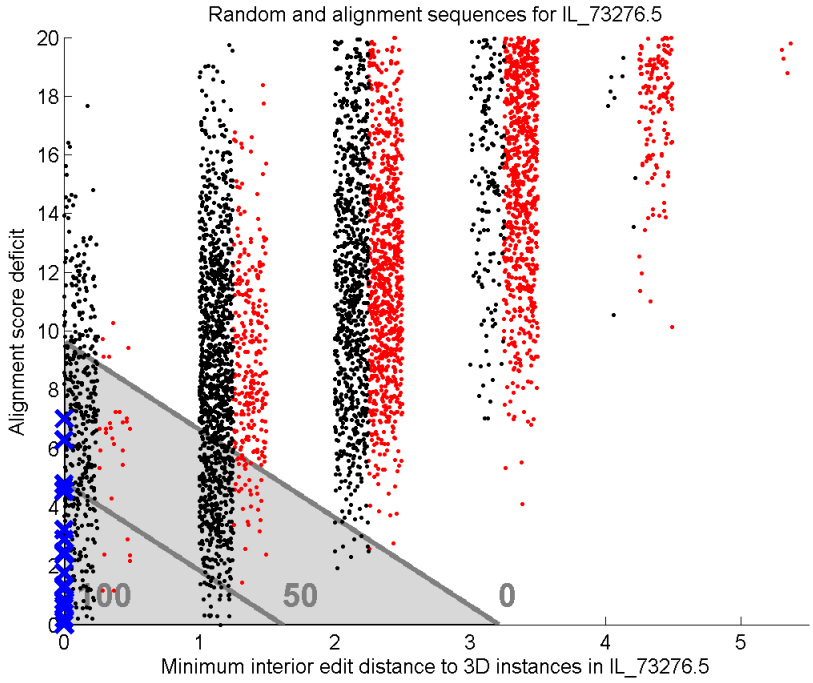


Figure 6.2 The acceptance region for IL_73276.5. Blue Xs are sequences from 3D structures, black dots are sequences from an alignment extract, and red dots are randomly generated sequences. As with the other acceptance region graph, the black dots have been shifted to the right by a uniformly distributed random variable between 0 and 0.25 so that they are all visible; red dots have been shift to the right by numbers from 0.25 to 0.5.

generated distractor sequences which have edit distance up to 5 and alignment score deficit up to 20. When a novel sequence is presented to JAR3D, it is compared to over 250 models for different loop geometries, and it could be accepted by any of them. We found that 22.6% of the sequences in the set of random internal loop sequences were accepted by at least 1 group. To put this number into a bit more context, 4.8% of randomly generated sequences have interior edit distance 0 to one or more groups, and so should be accepted by those groups, and 7.4% of randomly generated sequences have a cutoff score over 50 with respect to at least one motif group. This means that the 22.6% number should not be interpreted as a false positive rate, although it is giving us similar information. It is possible, and for smaller sequence lengths even quite likely, to randomly generate sequences that would form one, or even more, of the 277 internal loop geometries. Strand ordering also complicates things, as both strand orderings need to be considered for internal loops. For example, there are 22 5x5 geometries modeled in IL_1.13, so all 5x5 sequences have 44 chances to match a group. Because an insertion is not an automatic rejection, the 5x6 sequences can match to all 44 of these 5x5 groups as well as any of the 6x5 and 5x6 motif groups. In general, there are simply no true negatives, only sequences that we suspect should be negative. We simply do not have enough information to definitively claim that a sequence cannot form a given geometry.

6.3 Alignment extract study

To examine JAR3D's performance on the recognition and matching problems, we have done a small scale study using alignment extracts of sarcin-ricin motif loops. We use motif group IL_95652.3; recall that the SCFG/MRF model for this group was described in detail in Section 4.4. Two instances in the motif group can be aligned to multiple sequence alignments. The first alignment extract is from a Silva (Quast, Pruesse, Yilmaz, Gerken, Schweer, Yarza, Peplies, and Glöckner, 2012) alignment of eukaryal small ribosomal subunit sequences, which contains loop IL_4BPP_052 in motif group IL_95652.3 in Motif Atlas release 1.13. The second alignment extract is from a GreenGenes (DeSantis, Hugenholtz, Larsen, Rojas, Brodie, Keller, Huber, Dalevi, Hu, and Andersen, 2006) alignment, of bacterial small ribosomal subunit sequences which contains loop IL_1FJG_034 in IL_95652.3.

From these alignments we have removed all sequences with interior edit distance to instances in IL_95652.3 of zero. Edit distance of zero means that the sequence is an exact match to a sequence seen in 3D, and interior means we ignore the flanking Watson-Crick basepairs. Leaving these sequences in would essentially be using training data as testing data, so they are removed. Thus, the sequences we consider here are truly novel sequences for this motif group. For the rest of this section, "the Silva alignment" or "the GreenGenes alignment" refers to the alignments after these interior edit distance zero sequences have been removed. This results in 31,298 rows in the Silva alignment, and 65,530 rows in the GreenGenes alignment. Many sequences appear in the alignment more than once, because the same sequence occurs in more than one organism. For example, the sequence GAAGUACG*UGAAAC occurs on 23,773 rows of the Silva alignment, and so we say it has multiplicity 23,773. If we look at only unique sequences, the Silva alignment has 829 unique sequences, and the GreenGenes alignment has 2,249 unique sequences.

We will first look at a recognition problem, using the cutoff region described in Section 6.2. The question we are trying to answer here is: which of the novel sequences in the two alignments does the model for IL_95652.3 recognize as possible sequence variants? If a sequence has a positive cutoff score, it falls in the cutoff region and is recognized by the model. Therefore, we might refer to a sequence that meets the cutoff as "recognized" or "accepted" by the model for IL_95652.3.

Table 6.2 shows some example results for the recognition problem from the Silva alignment. The first three rows show the sequences with the highest multiplicity for the alignment. All of them have interior edit distance of one, and all the changes are expected to cause little change in the 3D geometry of the motif, so they are easily within the cutoff region and recognized by JAR3D.

The next two rows, 4 and 5, show the two sequences with highest multiplicity that don't make the cutoff. The sequence with the highest multiplicity that doesn't make the cutoff is sequence 4, GAAGUACG*UGAAC. It is interior edit distance of two away from the closest sequence in 3D. This isn't an overly large interior edit distance, but one of the edits is a deletion on the right strand. The other edit, a substitution, is actually so poor for the basepair the nucleotide that changed is making, that JAR3D aligns the sequence to the group by deleting that nucleotide and making a low

probability insertion. Because both edits would make large changes to the geometry of the loop, the sequence is not recognized by the model. The typical GUA sarcin-ricin base triple is maintained, so it is possible that this is a properly aligned, biologically viable sequence, that is just geometrically different from the sarcin-ricin geometry modeled in IL_95652.3. It also could be a small issue with the alignment, as adding another C to the end of the right strand makes the resulting sequence match IL_95265.3 fairly well, with an interior edit distance of 2. The sequence with the extra C, GAAGUACG*UGAACC appears in both the Silva alignment and the GreenGenes alignment, but only once in both.

The next sequence, sequence 5, has the second highest multiplicity that isn't recognized by the model is AAGUACG*UGAAAC. It has an interior edit distance of 1, a deletion on the left strand. Normally, this would not be enough of a change to make the JAR3D model miss recognition of the sequence, since the deletion does not affect the GUA triple. However, the outside flanking basepair is also changed to AC. This doesn't affect the interior edit distance, but an AC cWW basepair is non-isosteric with the canonical cWW basepairs seen in Watson-Crick helices. This sequence is almost certainly making a sarcin-ricin geometry, and small issues with the alignment caused an alignment problem with the outside flanking basepair. In fact, the sequence **G**AAGUACG*UGAAAC, with a G added the front of the left strand, matches IL_95652.3 quite well; and in fact is the sequence with the highest multiplicity in 6.2.

Finally, we look at the last two sequences, 6 and 7, that only appear in the alignment extract once. These sequences don't make the cutoff, and it is fairly obvious they should not. They are included to highlight some of the complications that can arise when working with alignment data. These sequences are UGUGUGUG*AGACGGAC and GUAUU*UGA, with interior edit distance of 5 and 6, respectively. These sequences are quite different from the sequences we expect to see for the geometry in IL_95652.3, both in size and in the nucleotides seen in the sequence. In addition, none of the flanking cWW basepairs for either sequence are canonical. This strongly suggests problems with the alignment for these sequences, or possibly a situation where this RNA molecule in these organisms has a completely different structure than what was observed in the 3D

structures that motif group IL_95652.3 was based on. The existence and the frequency of sequences like this explain why we chose to parameterize JAR3D models based on 3D instances instead of on sequence variability observed in multiple sequence alignments. But now we can use the JAR3D models to automatically recognize and exclude such sequences, and then use the remaining vetted sequences to improve the parameterization of the models.

#	Sequence	Multiplicity	Recognized	Cutoff Score	Interior Edit Distance
1	GAAGUACG*UGAAAC	23773	Yes	83.99	1
2	GUAGUAUG*UGAAAC	1270	Yes	83.77	1
3	GAAGUACG*UAAAAC	672	Yes	79.91	1
4	GAAGUACG*UGAAC	83	No	-8.52	2
5	AAGUACG*UGAAAC	77	No	-1.37	1
6	UGUGUGUG*AGACGGAC	1	No	-103.04	5
7	GUAUU*UGA	1	No	-147.01	6

Table 6.2 Table showing select results for the sequence recognition problem. This table show select sequences and the results of the analysis for the recognition problem using some sequences selected from the Silva extract corresponding to loop IL_4BPP_052 in motif group IL_95652.3.

Table 6.3 below summarizes JAR3D's performance on the recognition problem. If we look at all rows in the alignment, the JAR3D model for IL_95652.3 recognizes 98% of the sequences in each alignment. Looking at just unique sequences, the JAR3D model recognizes roughly 65% of the sequences in both alignments. It is important to note that looking at unique sequences gives an outsized weight to sequences which only appear in the alignment once and are probably not properly aligned, like those in the last two rows of 6.2. Because sequence alignments are unreliable, the most important takeaway from this study is that JAR3D performs very well on the highest multiplicity sequences, which are those that we have the most evidence for those being legitimate instances of the loop geometry in question.

Now we will look at JAR3D's performance on the matching problem, using the same alignment extracts. For each unique sequence in each alignment extract, we score the sequence against 270 non-trivial internal loop motif groups (we exclude motif groups whose JAR3D model only includes the two flanking Watson-Crick basepairs, leaving 270 motif groups). For each motif group, we

Alignment	All Sequences	Unique Sequences
Silva	98.00%	64.17%
GreenGenes	98.05%	65.72%

Table 6.3 Table summarizing recognition problem performance on alignment extracts. This table summarizes the results of JAR3D's performance on the recognition problem using Silva and GreenGenes alignment extracts corresponding to instances in motif group IL_95652.3.

calculate the Maximum Log Probability Score (MLPS), which is the log of the probability of the most probable parse of the sequence. Since these are internal loop sequences, we present the two strands to JAR3D in two different orders and take the higher MLPS. Then we find the highest MLPS across all 270 motif groups. The correct match occurs when the model for IL_95652.3 has the highest MLPS. This is a matching problem with n=1, and there are 270 possible motif geometries that the sequence could be matched to, so this is a difficult problem.

In Table 6.4 below a sample of representative sequences from the GreenGenes alignment are listed, and the results for those sequences on the matching problem. Sequences 1-3 in Table GGmatching are the highest multiplicity sequences in the alignment. They are all interior edit distance 1 away from a sequence observed in 3D structures, and they all match to IL_95652.3 using MLPS.

Sequences 4-5, GAGUACG*UAAAAC and GAGUACG*UGAAAC are both one deletion on the left strand away from sequences seen in 3D structures. They are the first two sequences by multiplicity that do not match to IL_95652.3 using MLPS. Deletions are given low probability by JAR3D models, so the penalty for the deletions is enough to give several other JAR3D models higher MLPS scores. It is worth noting that both of the sequences do make the cutoff region for IL_95652.3, even though it is not the top match for them. It is an open scientific problem whether these sequences fold into the same geometry as is seen in motif group IL_95652.3, and so it is not known if these sequences really should match that group or not. In the statistics, however, they will reduce the apparent success rate of JAR3D.

Sequence 6, GUA*UGAAAU, appears in the alignment only once, and is interior edit distance

#	Sequence	Multiplicity	Matched to IL_95652.3	Interior Edit Distance
1	GAAGUACG*UGAAAC	14205	Yes	1
2	GGACUACG*UAAAAC	8230	Yes	1
3	GAAGUACG*UAAAAC	7245	Yes	1
4	GAGUACG*UAAAAC	607	No	1
5	GAGUACG*UGAAAC	581	No	1
6	GUA*UGAAAU	1	No	5
7	GUAGUAGG*UGAUUU	1	No	4

Table 6.4 Table summarizing select results for the sequence matching problem. This table summarizes the results for the matching problem using sequences selected from the GreenGenes extract.

5 from the most similar sequence observed in 3D structures. The flanking cWW basepairs for this loop (GU and AU) are sensible, but the edits are all deletions on the left strand. This means there is no physical way for this sequence to form the sarcin-ricin loop geometry. There is probably an error in the alignment of the left strand for this sequence, as the right part does look like it is part of a sarcin-ricin motif. Here, we would not want this sequence to match motif group IL_95652.3, but since it is in the test set, it is counted as an error by JAR3D.

Finally, sequence 7, GUAGUAGG*UGAUUU, is an interesting case. It appears in the alignment only once and is interior edit distance 4 away from the most similar sequence observed in 3D structures. It is interesting in that does not make the cutoff for IL_95652.3, so it is not recognized by the model, but it does match best to IL_95652.3 using MLPS. Even though the sequence is not a particularly good match to IL_95652.3, it is an even worse match for the other groups. This highlights the fact that it is often better to look at all the measures together, instead of solely focusing on MLPS or the cutoff region.

Table 6.5 below summarizes JAR3D's performance on the matching problem, using the results of the alignment extracts. Given the difficulty of the n=1 matching problem with so many alternative models, JAR3D performed quite well. About 95% of rows in the alignment matched IL_95652.3 as the top model by MLPS. When just looking at unique sequences, around 60%

matched correctly. Keep in mind that it is not known what percentage of the sequences really should match IL_95652.3 best; we simply don't have data on that. Again, the most important takeaway is that JAR3D matches the sequences with high multiplicity.

Alignment	**Multiplicity**	**Unique Sequences**
Silva	95.79%	58.75%
GreenGenes	94.58%	62.78%

Table 6.5 Table summarizing recognition problem performance on alignment extracts. This table summarizes the results of JAR3D's performance on the matching problem using Silva and GreenGenes alignment extracts.

It can often be difficult to match a sequence to the correct geometry, particularly for smaller sequences, where a single sequence can easily be a good match for multiple geometries. However, a researcher studying an RNA molecule will often know the sequence of the molecule from multiple organisms, and the sequence variability across those multiple organisms may only be consistent with one geometry. If we presented 5 distinct novel sequences to JAR3D and calculated the average MLPS of the five sequences against each model, we would expect that the percentage of the time that JAR3D would match to IL_95652.3 would be substantially higher than the percentages seen here.

6.4 Comparison of JAR3D acceptance regions to RMDetect

An important finding of the JAR3D paper was that, even with the false positive controls described above, JAR3D accepts more appropriate sequences than a similar motif recognition tool, RMDetect.

RMDetect is a tool designed to scan genomes and multiple sequence alignments for possible instances of loop geometries. When RMDetect was released, there were models for four internal loop geometries, the C-loop, sarcin-ricin, kink-turn, and tandem GA loops. Here we will look at a comparison to the performance of JAR3D and RMDetect on sarcin-ricin loops, specifically a set of loop sequences that correspond to IL_2QBG_011, which is in motif group IL_85647.3. There are 320 distinct sequences taken from the Silva bacterial LSU alignment.

6.6 below summarizes the results of the comparison. The table also shows the five most repeated sequences in the alignment in each of four categories: sequences that both RMDetect and JAR3D recognize, sequences that only RMDetect recognizes, sequences that only JAR3D recognizes, and sequences that neither recognize. RMDetect was run with its default parameters, and recognition for JAR3D means a sequence falls within the cutoff region as described in Section 6.2.

There are 121 sequences recognized by both JAR3D and RMDetect, and this includes many of the high multiplicity sequences in the alignment. The higher the multiplicity of a sequence in the alignment, the more likely it is to be correctly aligned and to fold into the correct geometry. There are 136 sequences recognized by neither JAR3D or RMDetect as well. These have lower multiplicity, and many may be included due to errors in the alignment, or fold into another geometry that should not be matched, but is biologically viable.

There are 7 sequences accepted by RMDetect that are not accepted by JAR3D. These sequences are all singletons in the alignment, and often have large edit distances that include deletions that would likely cause them to fold into a different geometry, so it is not at all clear that they should be recognized as valid sarcin-ricin sequences.

There are 56 sequences that are accepted by JAR3D but not by RMDetect. This is a large difference, and it could be attributed to JAR3D simply having a more lax acceptance criterion. However, many of the sequences recognized by JAR3D but not recognized by RMDetect have high multiplicity and low edit distance, indicating that there is a high probability that they do fold into the correct geometry and should be accepted. The JAR3D cutoff score for many of these sequences is also quite high, meaning that even if we were to tighten the standard we use for false positive control the sequences would still be accepted.

Accepted by both: 121	**Multiplicity**	**Interior Edit Distance**	**Cutoff Score**
CUAAGUAC*GGAACUG	7415	0	98.27
CUAAGUAG*UGAACUG	1951	0	86.36
CUCAGUAC*GGAAGUG	1488	1	60.47
CUUAGUAG*CGAACUG	1106	1	82.87
CUCAGUAC*GGAACUG	614	0	100
RMDetect accepts, JAR3D does not: 7			
ACAAGUAC*UGACCGA	1	3	-4.09
CUAAGUA*AGAACUG	1	1	-0.58
CUAAGUAC*CUG	1	4	-296.71
CUAAGUAC*GGAAACGUG	1	2	-4.73
CUAAGUAC*GGAGUG	1	2	-26.55
JAR3D accepts, RMDetect does not: 56			
CUAAGUAC*AGAACUG	1129	0	-0.58
CUUAGUAC*AGAACUG	617	1	-21.47
CUUAGUAA*CGAACUG	138	1	-72.49
CUUAGUAG*CGAAUUG	78	1	-126.65
CUAAGUAA*AGAACUG	47	0	-10.43
Accepted by neither: 136			
CUAAGUAC*GAACUG	35	1	-0.58
CUUUUUCG*CAAAGUG	9	4	-21.47
GGAAAAAC*UGGAUUG	8	5	-72.49
UUAAUCGU*AGCCCG	7	6	-126.65
CCAAAUAG*CAAACCG	6	4	-10.43

Table 6.6 Table summarizing comparison of JAR3D to RMDetect. This table summarizes the results of a comparison between JAR3D and RMDetect on 320 unique sequences corresponding the the sarcin-ricin loop in helix 29 taken from the Silva bacterial LSU alignment.

CHAPTER 7 DISTRIBUTION OF JAR3D MODELS OVER RNA SEQUENCE SPACE

7.1 Introduction

From a probabilistic standpoint, JAR3D models are categorical distributions over the space of possible internal loop or hairpin loop sequences, which can be of varying length. The space of possible sequences is quite large, even though it is discrete. For example, consider a 5x5 internal loop model. Most of the probability for the model will be concentrated on sequences with five base on each strand, which is a quite manageable $4^{10} = 1,048,576$ sequences. However, JAR3D models also allow for variable length insertions. If this 5x5 model is made of five consecutive basepair nodes, it will allow for variable length insertions before and after the flanking Watson-Crick basepairs, and between each set of basepairs in the motif. The insertion points before and after the fanking basepairs can add one base each, and the interior insertion points can add two bases each. This means that a model for a 5x5 motif can produce 15x15 sequences. Because insertions can be any base, all $4^{30} = 1.15 * 10^{18}$ 15x15 sequences can be produced by the model, as can all shorter sequences!

It is desirable to be able to both understand a single model's distribution over sequence space, and to compare a pair of models and their distributions over sequence space. For example, when looking at a single model, one might want to know if the model is diffuse and spreads probability over many sequences, or concentrates probability on fewer sequences. Computing informational entropy is one way of addressing this problem. When comparing two models, one might want to know if the distributions assign similar probabilities to the same sequences. However, the space of possible sequences is so large that it is difficult to apply traditional methods for visualizing the amount of spread and for comparing distributions. Because JAR3D models allow for all 4-base sequences of an appropriate length for the model and also allow for insertions and deletions, this is true even for the smallest JAR3D models.

Traditional measures for comparing discrete distributions, such as relative entropy, would re-

quire the enumeration of all sequences assigned probability by the model. Because JAR3D models assign probability to such a large number of sequences, these methods are not feasible. However, JAR3D models have very long and thin tails, and assign comparatively large probabilities to much fewer sequences. Because of this, methods for analyzing and comparing models that only need information on high probability sequences would be quite useful.

As part of this research, a number of tools have been developed to assist in the study of these distributions, and are discussed in this chapter. They include an algorithm for finding the sequences assigned the highest probability for a model, a cumulative probability style graph useful for understanding how diffuse a model is as well as for understanding the overlap between two models, and a measure of similarity between models that can be calculated without considering all sequences assigned non-zero probability by the models.

7.2 Top k algorithm

When working with SCFG/MRF models for RNA 3D motif sequence variability, it is desirable to be able to list the best-scoring sequences for a group in order, from highest to lowest probability. Under some circumstances it might be desirable to list the top k sequences, in others we might want all sequences with a probability above a particular threshold, or to list sequences in order until a certain amount of the probability mass of the model has been accounted for. A simple algorithm has been developed that will list the best-scoring sequences for the models under any of these circumstances.

The SCFG/MRF models consist of a sequence of nodes which generate subsequences independently. As a prerequisite to the algorithm, a list of possible sequence outputs, with probabilities, for each node of the guide tree model is required. For most models and nodes it is simple write a program to list all of them. However, some motifs have cluster nodes which account for so many different nucleotide positions that making an exhaustive list is not reasonable, and for these nodes it is sufficient to list the most probable outputs. How many sequences need to be calculated depends on how many most probable sequences are needed overall, several thousand should be more than sufficient in most cases. Note that as a consequence of the sheer number of sequences that

some cluster and hairpin nodes can produce, for some nodes the input list of sequences will have probabilities that add up to considerably less than 1.

The list of these possible outputs need to be sorted by their probability. The algorithm does not require that the scores be probabilities, or that the overall score for the combined output be the sum or product of the node scores. It does, however require that the output scoring function is increasing in all of its variables. For example, the algorithm could work with the logarithms of probabilities and sum them. The algorithm keeps track of 2 lists of possible outputs: the list of most probable sequences, a list of candidate sequences, and additionally, a dictionary of possible future candidates sequences. In this discussion "output" will refer to the output of a particular node and "sequence" will refer to a full set of node outputs.

The algorithm needs three main data structures: a "best list" of the top scoring sequences in order, a "candidate list" of possible additions to the best list, and a dictionary of outputs that have met some but not all of the requirements to be on the candidate list. We will write $a, b, c, \ldots$ for the 1st, 2nd, 3rd, $\ldots$ nodes. Then a_1 is the most likely output to be generated by Node a, b_1 is the most likely to be generated by Node b, etc. The algorithm discusses the parents and children of sequences, which are defined as follows. For a given sequence $a_{n1}b_{n2}c_{n3}...$ its parents are $a_{n1-1}b_{n2}c_{n3}, a_{n1}b_{n2-1}c_{n3}..., a_{n1}b_{n2}c_{n3-1}...$, etc. The children for the sequence are $an1 + 1b_{n2}c_{n3}..., a_{n1}b_{n2+1}c_{n3}..., a_{n1}b_{n2}c_{n3+1}...$, etc. Each sequence will have exactly m parents and m children, except the most probable sequence, which has no parents, and the most probable sequence's children, which only have one parent.

The algorithm hinges on fact that for a sequence to be added to the best list, all of its parents must be on the best list already. This is because each node's output is independent, and the input is given in order of best scoring to worst scoring for each node. The candidate list is all sequences for which have all of their parents in the best list, and the dictionary tracks sequences for which some but not all parents are in the best list.

The best list can be initialized with the most likely sequence,$a_1b_1c_1...$, since the input should be listed from best score to worse score in all node values. The candidate list can then be initialized

with each of the children of the best sequence, $a_2b_1c_1, a_1b_2c_2, a_1b_1c_2$, etc. At this point the list of future potential candidates is empty. The list of potential future candidates contains an array of booleans which indicate which parents of the potential candidate have already been moved to the best sequences list. If all of these booleans are true, then the potential candidate is upgraded to a true candidate and can be removed from the list of potential future candidates.

After the initialization has finished the algorithm proceeds with three steps:

1. Select the candidate output with the best score from the candidate list, add it to the list of best scoring sequences, and remove it from the candidate list.

2. Update the dictionary entries for the new "best" sequence's children, or add them to the dictionary if it is not already present.

3. Move any outputs for which all dictionary entries are true from the dictionary to the candidate list.

These steps need simply be repeated until the desired termination criterion is met. If the candidate list is kept sorted and updated by properly placed insertions, the first step will be to "pop" the top entry from the list. If memory is limited, the memory burden of the potential candidate dictionary can be transferred to a computational burden: the parents of each child that would have been added to the dictionary can be searched for in the best list, and if they are all present the child is added to the candidate list.

If a dictionary is used for potential candidates, the algorithm is quite computationally efficient. If there are m nodes, then each step involves n updates to the dictionary and checks to see if dictionary outputs can be moved to the candidate list, and at most n insertions into the candidate list, though it will be quite rare to have n updates to the candidate list. It is only necessary to compute the score for a sequence when it is added to the candidate list.

The other performance concern is the amount of memory used by the algorithm. Both the candidate list and potential candidate dictionary grow at less than linear rates. Both can increase by at most n in each step. There is no outlier case where the candidate list can consistently grow

at this rate, however, the requirements to be added to the list are too stringent for it to happen even two steps in a row. For n sequences to be added to the candidate list even once would require many previous steps in which very few sequences were added to the candidate list. The dictionary can grow at a rate of n per step if changes consistently occur in only one variable. If this were to occur, the candidate list would not be growing at all, only one new output would be added to replace the one moved to the best list.

7.1 below shows the top 50 sequences from the JAR3D model for IL_03282.1, as well as their probabilities, found through this algorithm. The most probable sequence, GAGGU*AAGUC, has a probability of 0.75%. Note that probability here refers to maximum generation history probability, not total probability.

An implementation of this algorithm in Python can be found in Appendix D.

Number	Sequence	Probability	Number	Sequence	Probability
1	GAGGU*AAGUC	0.0075	26	UAGGC*GAGUA	0.00553
2	GAGAU*AGGUC	0.00701	27	UAGAA*UGGUA	0.00553
3	CAGGU*AAGUG	0.00696	28	UAGGG*CAGUA	0.0055
4	GAGGA*UAGUC	0.00691	29	AAGAC*GGGUU	0.0052
5	CAGAU*AGGUG	0.0065	30	AAGAG*CGGUU	0.00517
6	GAGGC*GAGUC	0.00646	31	UAGAC*GGGUA	0.00517
7	AAGGU*AAGUU	0.00646	32	UAGAG*CGGUA	0.00514
8	GAGAA*UGGUC	0.00646	33	GAUGU*AAAUC	0.00099
9	GAGGG*CAGUC	0.00642	34	GAGAU*AAGUC	0.00093
10	UAGGU*AAGUA	0.00642	35	GAGGU*GAGUC	0.00093
11	CAGGA*UAGUG	0.0064	36	GAUAU*AGAUC	0.00092
12	GAGAC*GGGUC	0.00604	37	GAGGU*AAGUU	0.00092
13	AAGAU*AGGUU	0.00604	38	CAUGU*AAAUG	0.00091
14	GAGAG*CGGUC	0.006	39	GGGGU*AAGUC	0.00091
15	UAGAU*AGGUA	0.006	40	GAUGA*UAAUC	0.00091
16	CAGGC*GAGUG	0.00599	41	GAGCU*AUGUC	0.0009
17	CAGAA*UGGUG	0.00599	42	GAGAU*GGGUC	0.00087
18	CAGGG*CAGUG	0.00595	43	CAGAU*AAGUG	0.00087
19	AAGGA*UAGUU	0.00595	44	GGGGU*AAGGC	0.00086
20	UAGGA*UAGUA	0.00591	45	CAGGU*GAGUG	0.00086
21	CAGAC*GGGUG	0.0056	46	GAGAU*AGGUU	0.00086
22	CAGAG*CGGUG	0.00557	47	GAGAA*UAGUC	0.00086
23	AAGGC*GAGUU	0.00556	48	CAUAU*AGAUG	0.00086
24	AAGAA*UGGUU	0.00556	49	GGGAU*AGGUC	0.00085
25	AAGGG*CAGUU	0.00553	50	GAUGC*GAAUC	0.00085

Table 7.1 Table showing the top 50 sequences from the JAR3D model for IL_03282.1. The probabilities for the sequences are also shown.

7.3 Entropy calculations

One way to get a basic understanding of how diffuse or compact a probability distribution is would be to calculate the informational, or Shannon, entropy. First described by Claude Shannon in (Shannon, 1948), informational entropy measures the average rate at which information is produced by a stochastic process. It is defined by the equation below:

$$S = -\sum_{i} P_i ln(P_i)$$

So, the more rare an event is, the more information it carries when it occurs. Therefore, the higher the informational entropy of a distribution, the more diffuse the distribution is.

Calculating informational entropy does require iterating over all possible values for a distribution, but informational entropy is also additive between independent events. Because the nodes in a JAR3D model are independent, we can simply calculate the informational entropy of each node in a model, and then add them together to get the informational entropy of the entire model. It is this property that makes calculating informational entropy for JAR3D models possible, as even fairly small JAR3D models can produce too many sequences to iterate over in a reasonable amount of time.

7.4 Rank order cumulative probability graphs

JAR3D models are diffuse by design, and assign probabilities to billions to trillions or more sequences. Above, we listed the top-scoring 50 sequences from motif group IL_03282.1, in decreasing order of probability. Listing all of the top 100,000 sequences in a table is not feasible, and would be very difficult to absorb. But in fact in many groups the 100,000th sequence still has appreciable probability. In fact, the 100,000th most likely sequence in the JAR3D model for IL_03282.1, GGUGA*GAAUA, has a probability of $2.7 * 10^{-7}$. While this is quite small, it is not entirely negligible.

Because the sequences are discrete and hard to break down into specific dimensions, visualization of the way in which JAR3D models distribute their probability over the space of all possible

sequences is difficult using traditional lists of possible outcomes or probability density graphs. The distributions created by JAR3D models are essentially very large categorical distributions, which would traditionally be graphed with bar graphs. A bar graph with 50 bars, each with a sequence label, would be difficult to absorb. One with 100,000 or more rows would be nearly meaningless. Instead, I have developed a new kind of graph to help make sense of the distribution of JAR3D models over sequence space.

A rank-order cumulative probability (ROCP) graph is one way to visualize JAR3D models, and such graphs will also give us a way to compare models pairwise. The basic idea is to sort the possible sequences in decreasing order of probability and then graph the cumulative probability versus sequence number. We will introduce the idea with an example below.

Graphs like these could be also be used for other categorical distributions. Sorting by decreasing probability is often recommended for categorical data when making other graphs for categorical data, such as bar graphs representing probabilities.

7.5 Single group rank order cumulative probability graphs

It is useful to look at simple examples with a small number of possible sequences to understand how the graphs before moving on to JAR3D models which cover, as discussed in Section 7.1, can assign probability to quintillions of sequences. We begin with a simple example, where we can easily list all possible outcomes. Basepairs allow for 16 possible sequences, and below in Figure 7.1 are graphs for substitution probability distributions for GC and UU cWW basepairs, taken from JAR3D substitution matrices.

The graph shows the number of sequences in the cumulative sum on the x axis and cumulative probability on the y axis, with sequences ordered from most to least likely. So, for example, the height at x=3 would be the sum of the first three most probable sequences. The graphs are stepwise approximations of concave down curves.

The graphs are useful in revealing how the distributions distribute probability. For example, in the graphs above we see that GC cWW basepairs assign similar probabilities to four different base combinations, resulting in a fairly diffuse distribution, at least compared to a UU cWW basepair,

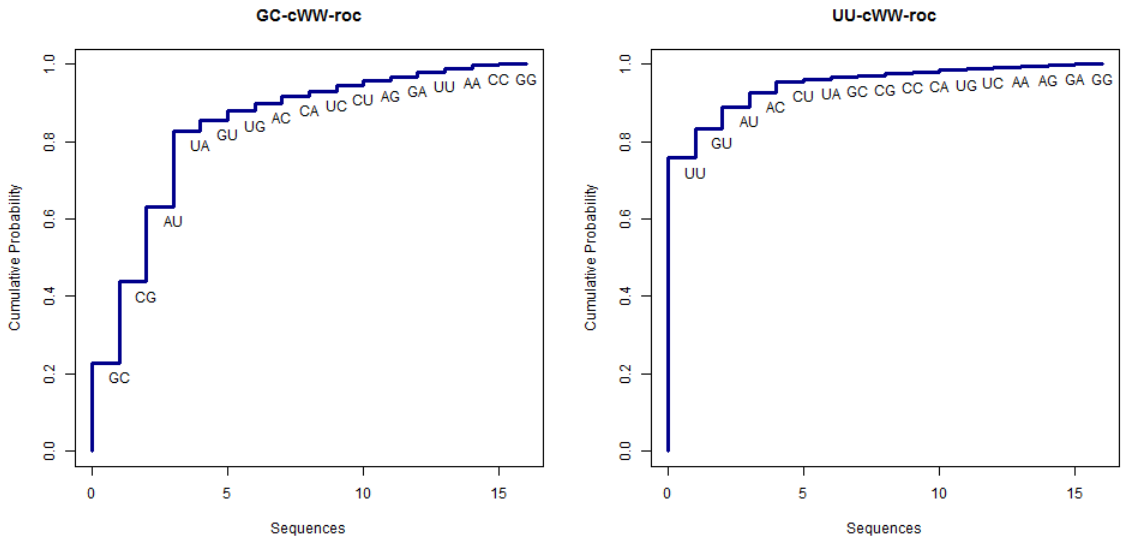


Figure 7.1 ROCP graphs for GC and UU cWW basepairs. The base combination that is added to the cumulation is shown at each step. The same 16 possible base combinations are summed over in each group, but in different orders based on the rank ordering of the distributions.

which assigns over 70% of its probability to its native UU combination. Because the graphs show the sum of probabilities over all possible sequences, they terminate at probability one.

In practice, at least for JAR3D models, it is useful to make ROCP graphs to cover a certain percent of all probability for a model or for the first N sequences, since JAR3D models have very long tails and it is not feasible to calculate probabilities for all sequences a model can generate.

7.6 Examples of ROCP graphs for 10-nucleotide motif groups

Some examples for internal loops with a total of 10 nucleotides are shown below. We also calculate the entropy of each model. By themselves, these graphs give information about how diffuse a model is, giving a graphical interpretation to information summarized by the informational entropy of the model. Three different motif groups are examined, each modeling internal loops that have 10 core nucleotides.

First is the graph for IL_03282.1, pictured below in Figure 7.2. It has an informational entropy of 13.12. This is fairly low for motif groups of its size, as it concentrates its probability on relatively few sequences, because all nucleotides in the group are involved in a basepair. Basepairs shift probabilities towards particular base combinations, making the model less diffuse.

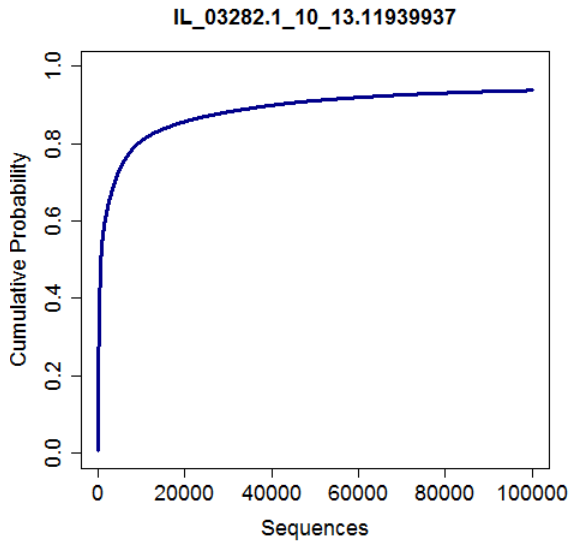


Figure 7.2 ROCP graph for IL_03282.1. IL_03282.1 is a motif group model for a 10 nucleotide motif with an informational entropy of 13.12.

Below, we look at IL_21077.1, show below in Figure 7.3. It has an informational entropy of 18.87. This group contains no basepairs beyond the flanking cWW basepairs, so it is much more diffuse than the previous models. The piecewise linear appearance of the graph reveals that it assigns the same probability to many sequences.

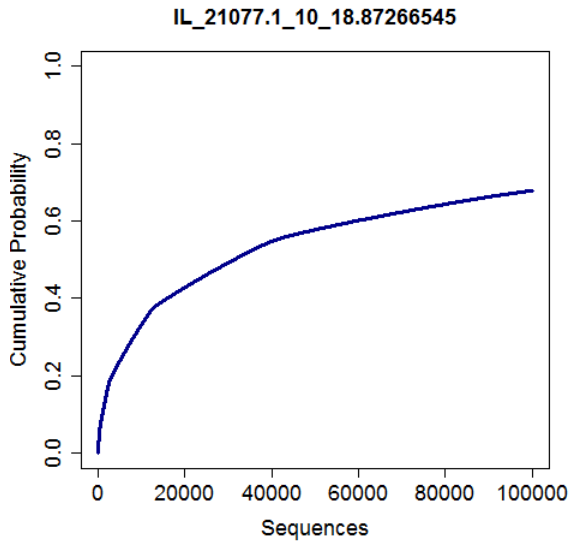


Figure 7.3 ROCP graph for IL_21077.1. IL_21077.1 is a model for a motif group with 10 nucleotides and an informational entropy of 18.87.

7.7 Paired rank order cumulative probability graphs

Rank order cumulative probability (ROCP) graphs can also be used to compare two different models to see if they are assigning probability to sequences in the same part of sequence space. Say that one wishes to compare model A to model B. One can score the N most probable sequences for model B against model A, and compare the cumulative probability curves generated by these sequences against the curves generated by the N most probable sequences for the original group. We call these "Paired Rank Order Cumulative Probability", or PROCP, graphs.

Again, it is useful to look at a simple example where all possible sequences can be explored before discussing JAR3D models. An example comparing GC cWW basepairs to UU cWW basepairs is shown below in Figure 7.4.

The blue curve generated by the N most probable sequence for group A scored against group A represents the steepest possible curve that can can be generated for group A. The red curve is generated by the N most probable sequences for group B scored against group A. So the closer the red (bottom) curve generated by scoring the N most probable sequences for model B against group A is to the blue curve, the closer model A and model B are in sequence space. If the models assign

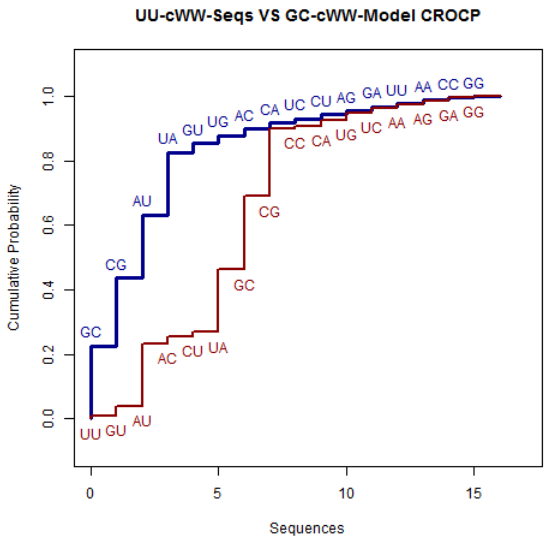


Figure 7.4 PROCP graph comparing UU cWW rank ordering against GC cWW distribution. The blue curve is the same as it was in the single distribution ROCP graphs, for a GC cWW basepair. The red curve sums the probabilities for GC cWW basepair as well, but in the rank order for UU cWW basepairs.

most of their probability in different parts of sequence space, then the B sequences versus A model curve will stay very close to zero until N is quite large. Because both models assign probability to a finite number of sequences, eventually both curves approach 1.

Some examples comparing 5x5 internal loop models are shown below. Since A sequences versus B model produces different graphs from B sequences versus A model, both are shown.

Graphs comparing the models for IL_05723.1 and IL_24982.5 are shown in Figure 7.5. Both motif groups have a variety of sequences from 3D, but they share a sequence from 3D in common. Also, they share a number of basepairs in common. Because of this, their curves are fairly close together. Because their geometries are different, there are some sequences only given high probability by one group or the other, so the curves do not overlap completely.

Graphs comparing IL_05723.1 and IL_23639.1 are shown in Figure 7.6. Unlike IL_05723.1 and IL_24982.5, these two groups are assigning probability in relatively different parts of sequence space. Even though these two motif groups are very different for motif groups originating from 3D instances of the same sequence length, the red lines do climb at a steady rate and reach about 20%

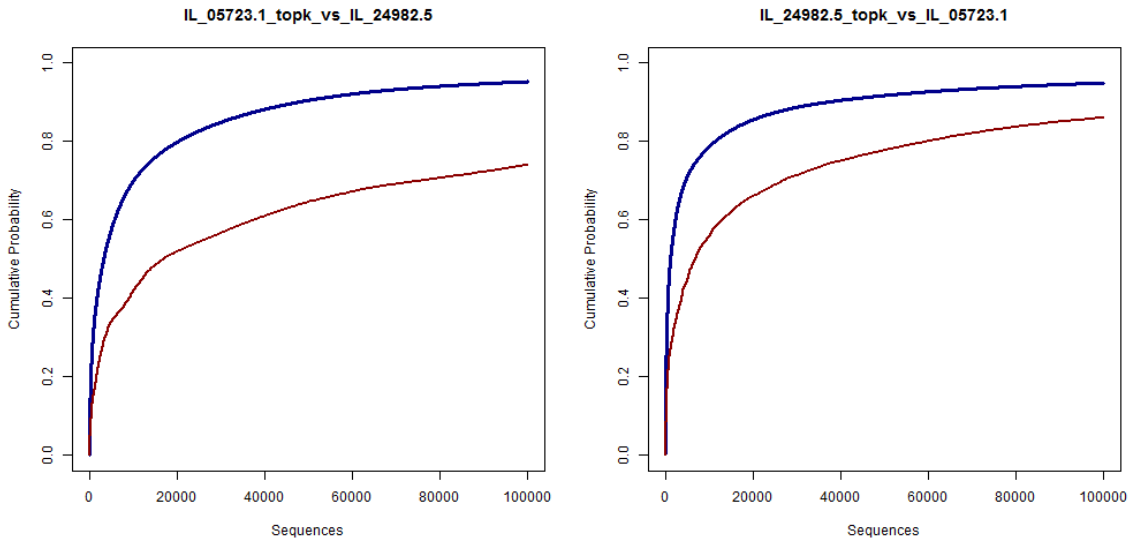


Figure 7.5 PROCP graphs comparing IL_24982.5 and IL_05723.1. The left graph shows IL_24982.5 sequences versus IL_24982.5 model (blue) and IL_05723.1 sequences versus IL_24982.5 model (red). The right graph shows IL_05723.1 sequences versus IL_05723.1 model (blue) and IL_24982.5 sequences versus IL_05723.1 model (red).

probability after 100,000 sequences for both graphs. This is because both groups will like the same nucleotides in the four positions that account for the flanking basepairs, AU, UA, CG, GC, and to a lesser extent GU and UG. This means that there are only $4^6 = 4096$ possible interior sequences with 36 relevant flanking sequences possible for a 5x5 motif. So, even a list of random sequences (with appropriate flanking basepairs) will accumulate a reasonable amount of probability after 100,000 sequences.

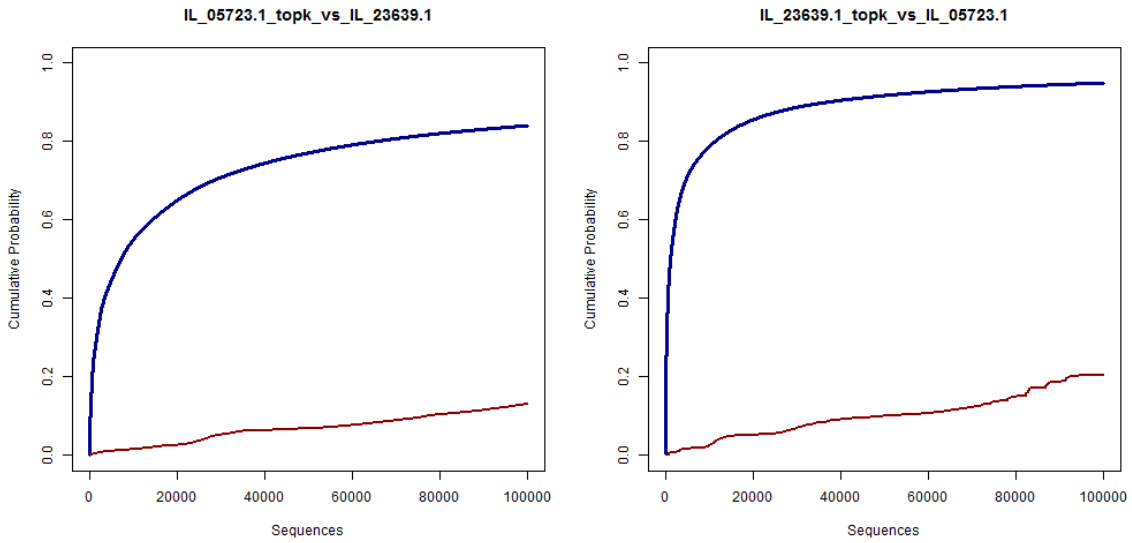


Figure 7.6 PROCP graphs comparing IL_23639.1 and IL_05723.1. The left graph shows IL_23639.1 sequences versus IL_23639.1 model (blue) and IL_05723.1 sequences versus IL_23639.1 model (red). The right graph shows IL_05723.1 sequences versus IL_05723.1 model (blue) and IL_23639.1 sequences versus IL_05723.1 model (red).

7.8 Using PROCP graphs to analyze sarcin-ricin groups

Using the comparison cumulative probability graphs on a group of related motifs gives some insights into how JAR3D works in practice. Here we will look at PROCP graphs for the 12 sarcin-ricin motif groups in IL_1.13. There are 12 sarcin-ricin motif groups in IL_1.13, ranging in in number of nucleotides from 13 to 17 nucleotides.

There are three 13 nucleotide motif groups in IL_1.13, IL_49493.4, IL_97191.1, and IL_31754.1. IL_49493.4 has 13 instances, while the other two are singletons. They share a tWH basepair and the characteristic sarcin base triple, but differ somewhat in the final two non-flanking basepairs. Because all three groups are the same size and have similar basepairing, they assign high probabilities to many of the same sequences. The PROCP graphs for the 13 nucleotide sarcin groups are shown below in Figure 7.7.

Not all the sarcin-ricin groups are so similar, however. Not all sarcin-ricin groups have the same number of nucleotides on each strand, and some that do are still dissimilar. Take, for example, IL_95653.3 and IL_94973.1, both 14 nucleotide sarcin-ricin motif groups, shown below, along with

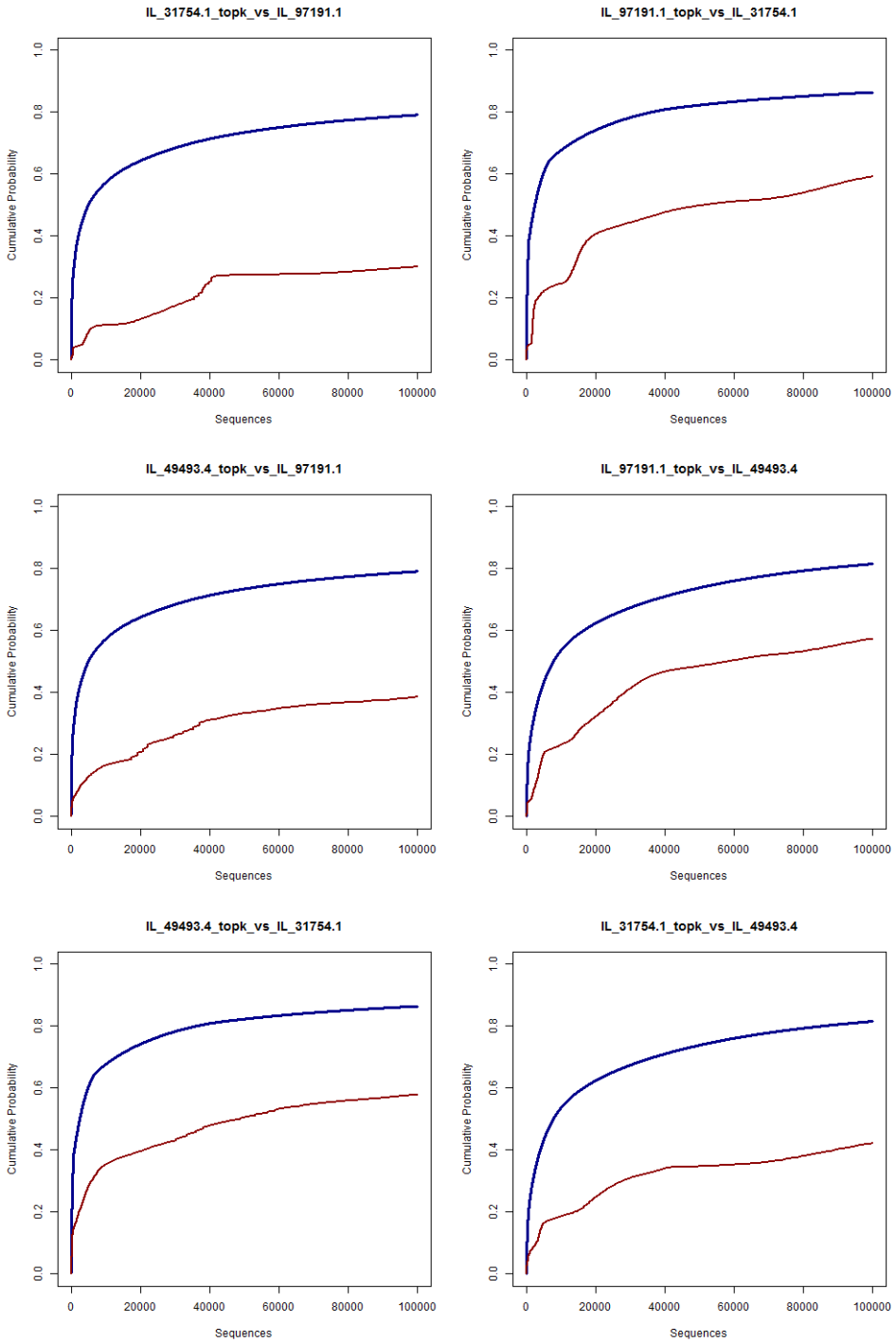


Figure 7.7 PROCP graphs for 13 nucleotide sarcin-ricin motifs. The 13 nucleotide sarcin-ricin motifs include IL_31754.1, IL_49493.4, and 97191.1. They all show a fairly high amount of overlap in sequence space, because their basepairing patterns are very similar.

comparisons to IL_71685.1, which also has 14 nucleotides, shown below in Figure 7.8, along with PROCP graphs for the other 14 nucleotide sarcin-ricin motifs.

IL_71685.1 has an insertion on the shorter strand compared to the 13 nucleotide motif groups, while IL_95653.3 and IL_94973.1 are both very similar to 13 nucleotide sarcin-ricin motifs with an insertion on the longer strand. However, the insertion point is different in each group. This causes the groups to concentrate in relatively different parts of sequence space, even though their 3D structures are very similar. To fit one 14-nucleotide sequence to a different model, you need to not have a conserved insertion, and you need to add an unusual insertion.

It is possible that the 13 nucleotide sarcin-ricin motifs groups appear to be similar primarily because the part of sequence space occupied by 13 nucleotide sequences is comparatively small compared to the part of sequence space occupied by longer sequences. This would imply that the 14 nucleotide sarcin-ricin motif groups would only appear to be more dissimilar because there are simply more 14 nucleotide sequences which need to be assigned probability. However, if we look at the comparison graphs for the largest sarcin-ricin groups, which have 17 nucleotides, we see that this is not the case.

The comparison graph for the 17 nucleotide sarcin-ricin groups, IL_54954.1 and IL_17682.1, are shown below in Figure 7.9. We can see that even though they assign probability to the much larger part of sequence space than the 14 nucleotide groups, they show a high amount of overlap. The red line holds fairly close to half the height of the blue line in both graphs. There are some other pieces of information that this graph does a good job of illustrating. It shows that as the motif groups get larger, it takes many more sequences to cover a reasonable amount of their probability. IL_17682.1 covers about 50% of its total probability in the first 500,000 sequences, while IL_54954.1 covers less than 30%. The 13 nucleotide sarcin-ricin groups cover around 80% of their probability in 500,000 sequences, and the 14 nucleotide groups cover about 70%. For groups much larger than these sarcin-ricin groups, millions of sequences or more would need to be scored to assess the similarity between the groups.

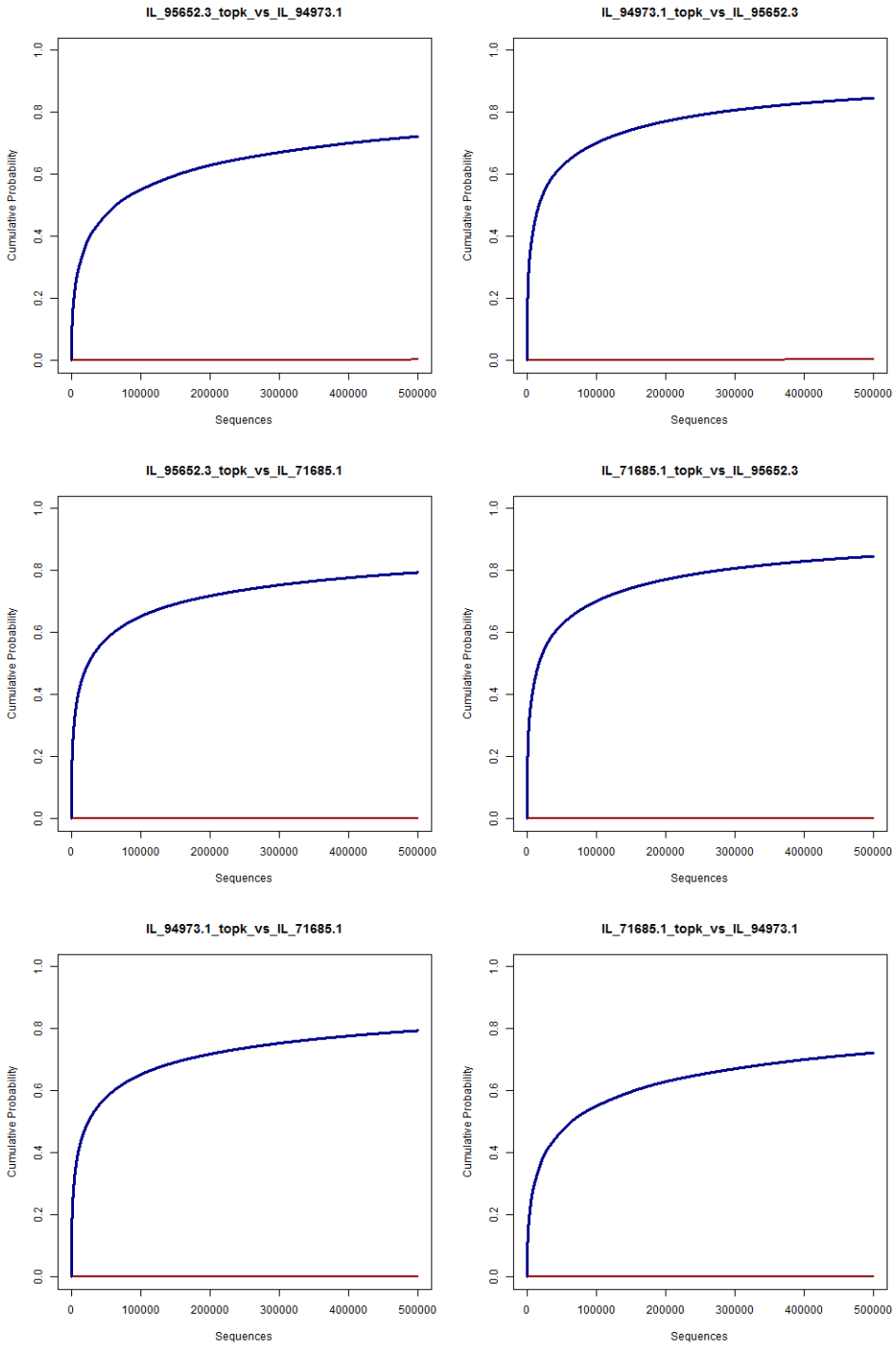


Figure 7.8 PROCP graphs for the 14 nucleotide sarcin-ricin motifs. The 14 nucleotide sarcin-ricin motifs include IL_71685.1, IL_95653.3 and IL_94973.1. They show a very low amount of overlap in sequence-space, especially compared to the 13 nucleotide sarcin-ricin motifs.

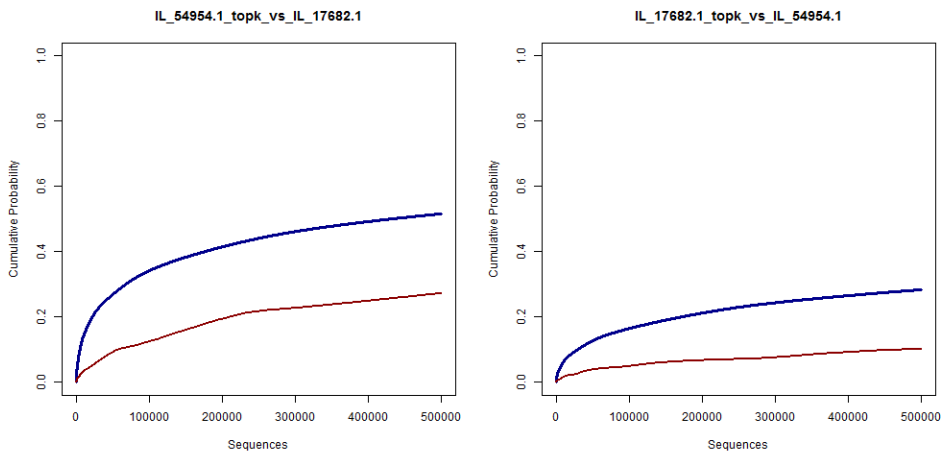


Figure 7.9 PROCP graphs for the 17 nucleotide sarcin-ricin motifs. The 17 nucleotide sarcin-ricin motifs include IL_54954.1 and IL_17682.1. They show a high amount of overlap in sequence-space, although a comparatively small amount of their probability is covered in the first 500,000 sequences compared to the smaller sarcin-ricin groups.

As mentioned before, IL_17682.1 covers about 50% of its probability in its first 500,000 sequences while IL_54954.1 covers only about 30%. This is quite a large difference given that the motif groups are the same size and have very similar 3D structures, and also overlap a fair amount in sequence space. This tells us that whatever is different between IL_17682.1 and IL_54954.1 makes the later more diffuse than the former. Further investigation reveals that IL_17682 does not have the GUA base triple that most sarcin-ricin motifs do. The bulged base gets probability spread out over A, C, G, U. The sarcin-ricin graphs also highlight an inherent asymmetry in the JAR3D models. JAR3D models give more probability to sequences that are longer than the sequences from 3D used to make the model than they give to shorter sequences. This is by design, because extra nucleotides can be inserted into an internal loop and bulge out to the side, allowing the sequence to still fold into the same structure. However, if crucial nucleotides are deleted, such as a basepair, the resulting structure will likely be different. Since sarcin-ricin motifs come in different sizes that have very similar overall structures over most of the motifs, they can be used to illustrate this disparity, as shown below in Figure 7.10.

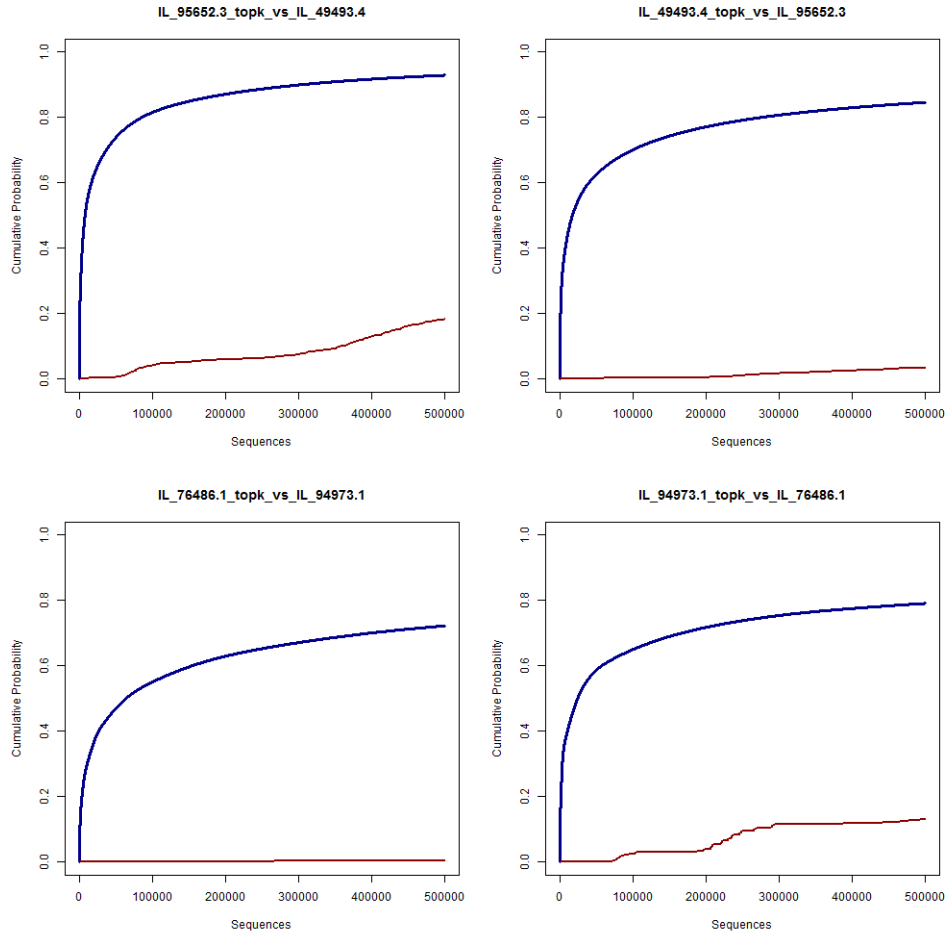


Figure 7.10 PROCP graphs for sarcin-ricin motifs of different sizes. The graphs show the 14 nucleotide IL_95652.3 and the 13 nucleotide IL_49493.4 (top), and 15 nucleotide IL_94973.1 and 14 nucleotide IL_76486.1 (bottom). In both case, the sequences from the larger group show some overlap with the smaller group, but the short sequences show little to no overlap with the larger group.

7.9 Distance measure based on rank-ordered cumulative probability

The two-group comparison graphs give a good indication of how much two JAR3D models overlap in sequence space, but it is often desirable to compare many groups at once, in which case a single number giving a measure of the distance between the JAR3D models would be prefered to having many graphs to sort through. However, traditional measures for measuring the distance between categorical distributions, such as relative entropy, require enumerating over all possible values (in the case of JAR3D models, sequences). Unfortunately, JAR3D models assign probability to far too many sequences for it to be feasible to enumerate over all of them when comparing even two groups. Because of this, I developed a method that can give a measure of the distance between the JAR3D models without enumerating over all possible values.

The distance measure can be used for any two categorical distributions that assign probability to the same countable set of m outcomes. We will call these distributions A and B. Let $\{a_1, a_2, ...a_m\}$ be the outcomes ordered from most to least probable against distribution A and let $\{b_1, b_2, ...b_m\}$ be the outcomes ordered from most to least probable against distribution B. The distance measure between the groups, $rocp(A, B)$, can then be expressed as $rocp(A, B) = max\_nA(\{a\_1...a_n\}) - A(\{b_1...b_n\})$. Because $\{a_1...a_m\}$ is ordered from most to least probable against distribution a, for any n between 0 and m, $max_nA(\{a_1...a_n\}) - A(\{b_1...b_n\})$ will be positive. Because $A(\{a_1...a_m\}) = 1$, $rocp(A, B)$ will be less than or equal to 1. It should be noted that the distance measure is not symmetric, that is rocp(A,B) does not necessarily equal $rocp(B, A)$.

This distance measure has a number of desirable properties. It will always fall between 0 and 1 (inclusive), as outlined above, and is fairly easy to interpret. A distance of 0 indicates that the distributions have identical rank ordering of the possible outcomes. In this special case, $rocp(A, B)$ will always equal $rocp(B, A)$. Unfortunately, a distance of zero does not imply that the distributions themselves are identical, as they can vary in anyway that does not change the rank ordering of outcomes. If both $rocp(A, B)$ and $rocp(B, A)$ are equal to 1, then the two distributions are completely disjoint. It is possible that $rocp(A, B) = 1$ but $rocp(B, A) < 1$, in which case A assigns 0 probability to b_1 to $b_n a$, where na is the number of outcomes A assigns a positive

probability, but B assigns some probability to at least 1 outcome in a_1 to a_{nb}. Finally, in many cases, the distance can be found without enumerating over all possible values.

Because $A(\{a_1...a_n\})$ and $A(\{b_1...b_n\})$ are increasing with respect to n and are bound above by 1, the maximum value that $A(\{a_1...a_n\}) - A(\{b_1...b_n\})$ from some n1 to m is $1 - A(\{b_1...b_n\})$. If the maximum value of $A(\{a_1...a_n\}) - A(\{b_1...b_n\})$ over 1 to n1 is greater than or equal to the maximum possible from n to m, then we can be sure that the maximum over all possible n has been found, and there is no need to do further calculations.

7.10 Relationship between distance measure and comparison graphs

The distance measure easily maps to the two-group comparison graphs. The distance measure is the maximum distance that occurs between the comparison red line and the maximum blue line over the graph. Because the graph of group A sequences scored against group B and group B sequences scored against group A are different graphs, the distance measure is not symmetric.

Because the graphs are not calculated to their end, they may not actually cover the location of the actual distance however. However, calculations can be done based on the state of the graphs at the last calculated point to determine if the maximum has been conclusively found. If the maximum hasn't been conclusively found, bounds can be calculated for the maximum between the remainder of the curves, as well as an extrapolated estimation of the maximum distance between the curves.

The maximum distance that could be observed after the end of a calculated graph is one minus the current last value of the red comparison curve. This could be achieved if the blue curve attains its maximum value of one before the red curve increases again. If the maximum distance observed in the enumerated data is greater than this value, then the maximum has already been observed conclusively. Even if this is not the case, it is possible that the currently observed maximum is the actual maximum. This is more likely if the currently observed maximum is in the interior of the graph, and less likely if the currently observed maximum is the last point in the graph.

In the case that the maximum is not conclusively observed in the enumerated graph data, an estimate can be made for the maximum distance after the enumerated data. The estimate used is to assume that the ratio between the height of the red and the blue curves is maintained until the

blue curve reaches one. This can simply be found by dividing the last value for the red curve by the last value for the blue curve, and subtracting the result from one. This value will always fall between the maximum value for the distance after the graph of one minus the current value and the minimum value for the distance after the graph, which is the distance between the curves at the last point in the graph. If this is greater than the largest distance observed in the graph, it is used as an estimate of the actual distance. If the maximum distance observed in the graph is larger, it is used.

7.11 Examples of distance measure used on IL_1.13 groups

To illuminate how the difference measure is calculated, several examples of the distance measure to compare motif groups from IL_1.13 will be shown in this section.

Show below in Figure 7.11 are the two PROCP graphs for IL_03282.1 and IL_53323.1, both directions, with the distance measure shown. IL_03282.1 and IL_53323.1 are both 5x5 motif groups. Even though the red comparison line reaches nearly 50% by the end of the graph for both graphs, the groups assign probability to fairly different parts of sequences space, which is evident when looking at the distance measure for the graphs. For the graph scoring IL_03282.1 sequences against the IL_53323.1 motif group, a maximum distance of 0.675 is observed after accumulating the probability of the first 9851 sequences. For scoring IL_53323.1 sequences against the IL_03282.1 motif group, a maximum distance of 0.703 is observed after accumulating the probability of the first 7054 sequences.

IL_032282.1 and IL_53323.1 are both fairly specific for JAR3D motif groups, assigning around 80% of their probability in their first 10,000 sequences. This is the reason the distance measures between the groups are relatively high despite the fairly high value for the red comparison line by the end of the graph. In this case the maximum distance has been observed with certainty in both graphs. The final value for the red line in the graph scoring IL_03282.1 sequences against the IL_53323.1 motif group is 0.532, meaning the maximum that the distance measure could be after the graph is is 0.468. The final value for the red line in the graph scoring IL_53323.1 sequences against the IL_03282.1 motif group is 0.558, meaning the maximum that the distance measure

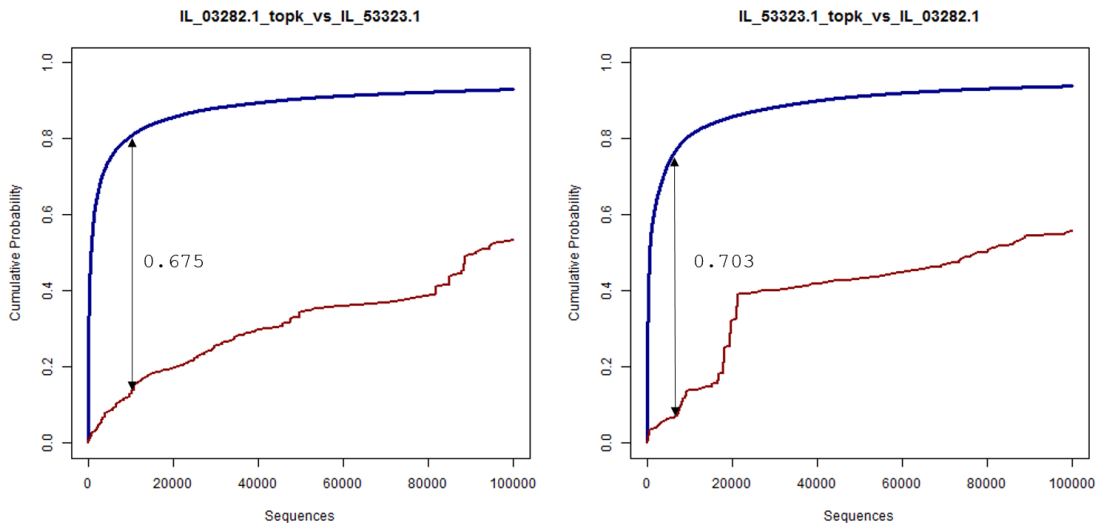


Figure 7.11 PROCP graphs for IL_032282.1 and IL_53323.1 with distance measures shown. The motif groups are both 5x5, and are relatively different for groups of that size.

could be after the graph is is 0.442. In both cases, the maximum possible distance between the lines after the portion shown in the graphs is less than the maximum distance between the lines observed in the graphs, so the observed maximum is the overall maximum.

In some cases, it can be fairly obvious that the true value for the distance measure has not yet been observed. As an example, there are the 17 nucleotide sarcin-ricin motifs, IL_54954.1 and IL_17682.1. The PROCP graphs for IL_54954.1 and IL_17682.1, with indicators for the location of the maximum observed distance added, and are shown below in Figure 7.12. Because the motifs they model are so large, even after accumulating probability from the most probable 500,000 sequence only about 40% of the probability IL_17682.1 is covered, and about half of that for IL_54954.1. The groups are fairly similar and the red comparison line remains at about 50% of the blue line. The observed maximum distance for both graphs occurs at 500,000 sequences. Because at the end of the graphs it seems the blue line is still increasing at a faster rate than the red line, it is quite likely that the maximum distance between the curves has not yet been observed.

In other cases, it is not clear if the maximum distance between the lines has or has not been observed. The 5x5 motif groups IL_46306.1 and IL_52958.1 serve of an example of this case, and are shown below inFigure 7.13. They are fairly disperse compared to the other 5x5 motif groups,

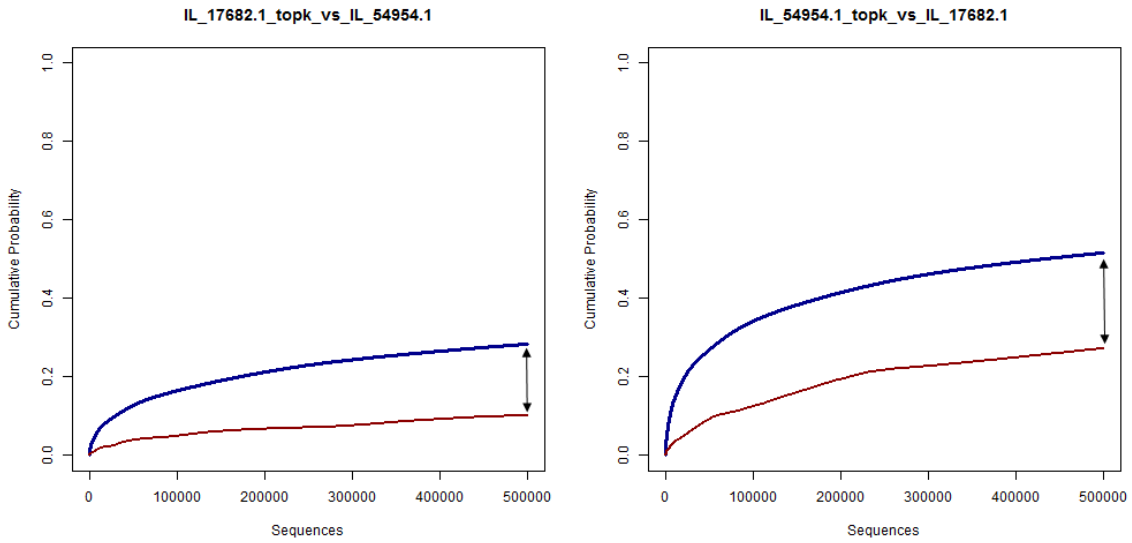


Figure 7.12 PROCP graphs for IL_17682.1 and IL_54954.1 with observed maximum distance. Both maximum distances occur at 500,000 sequences, and more data likely needs to be calculated to find the actual ROCP dissimilarity measure.

and cover about 70% of their probability in their 100,000 most probable sequences. They are also fairly similar, with the red comparison line stay fairly close to the blue line. For the graph scoring IL_46306.1 sequences against the IL_52958.1 motif group, a maximum distance of 0.374 is observed after accumulating the probability of the first 28379 sequences. For scoring IL_52958.1 sequences against the IL_46306.1 motif group, a maximum distance of 0.381 is observed after accumulating the probability of the first 14682 sequences. Both the maximums happen well before the end of the calculated data, but it is possible that the actual maximum occurs after the enumerated calculated data, because the distance from the red line to one for both graphs is greater than the observed maximum to this point.

For this particular instance, however, it is likely that the observed maximum distances are the actual maximum distances. Given that we known that the underlying distributions are 5x5 JAR3D motif groups, it is unlikely that the red line will remain relatively flat long enough for distance to pass the observed maximum distance, since both distributions concentrate most of their probability on sequences of the same length. In general, however, it would be necessary to process more sequence to be sure the correct maximum distance has been found.

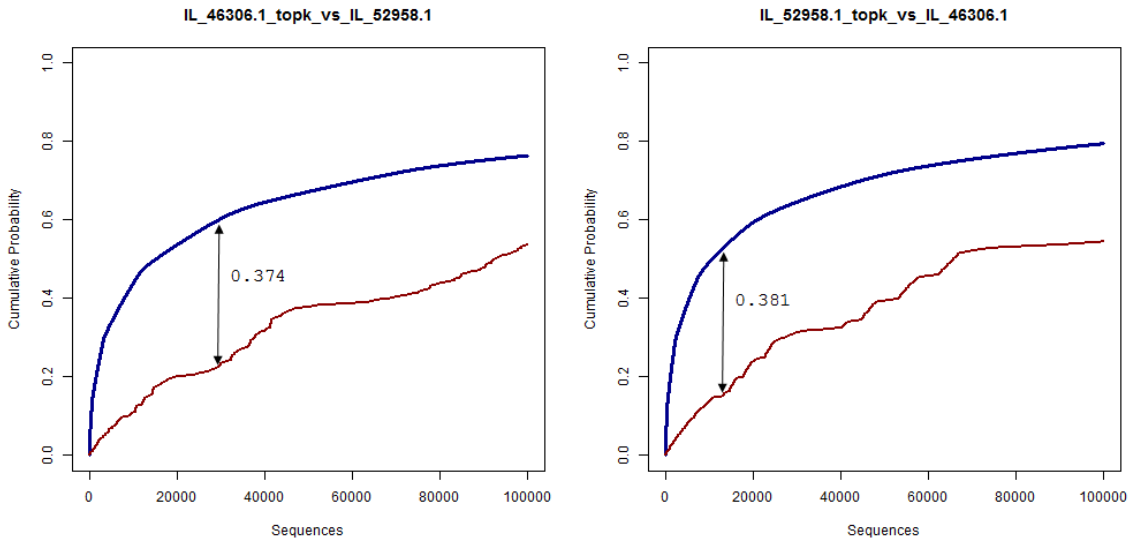


Figure 7.13 PROCP graphs for IL_46306.1 and IL_52958.1 with observed maximum distance. Both maximum distances occur before 500,000 sequences, but it is not clear they are the maximum when they are observed.

7.12 Approximation of the ROCP distance

Calculating the ROCP distance based on the 100,000 or 5000,000 most likely sequences will find the exact distance for some comparisons, but not all. If the exact distance has not been found, one of two assumptions are made to estimate the distance based on available information. If the current maximum distance is in the interior of the sequence list, that is, if it occurs somewhere other than at the last sequences added to the sum, it is assumed that the current maximum is the global maximum. While this will not always be the case, if the sums have grown closer, it is likely they will continue to do so, so the current maximum is likely to remain the maximum.

If the current maximum occurs at the last sequences added to the cumulative sums, then the maximum distance is still growing. In this case, the ROCP difference measure is estimated with the ratio between the in-group ordered sum and the comparison-group sum. It is assumed that this ratio is the same when the in-group sum reaches 1, and the difference at that point is the maximum. This is very unlikely to actually occur, but it gives a reasonable and conservative estimate of the ROCP difference.

7.13 Heatmaps of ROCP for sets of motif groups

Heatmaps are a useful way of displaying pairwise distances between multiple objects in a group. In this section we will look at heatmaps using the ROCP difference measure based on 3 sets of motif groups, 5x5 motifs, sarcin-ricin motifs, and the entire IL_1.13 motif atlas release. The heatmaps will show the ROCP distances between SCFG/MRF models for sequence variability.

Most heatmaps are symmetric because they use distance metrics which are symmetric. The ROCP difference is not symmetric however, so the resulting heatmaps will be asymmetric as well. The asymmetric heatmaps can reveal more information about the motif groups, but can also be harder to read, especially when many groups are being compared at once. Symmetric heatmaps could be made by taking the average, minimum, or maximum of rocp(A,B) and rocp(B,A).

7.14 Heatmap for IL_1.13 5x5 motifs

We will start by looking at a heatmap using the ROCP dissimilarity measure for the 5x5 internal loops from IL_1.13. These are 12 motif groups with exactly five nucleotides on each strand for all 3D instances of the motifs. A heatmap for these 12 motifs is shown below in 7.14. Note that because the ROCP difference measure isn't symmetric, the heatmap is asymmetric.

One of the most obvious things that is shown by this heatmap is that the 5x5 motifs have a fair amount of overlap, as there are no white squares and very few light yellow squares. It also apparent that there are three pairs of motif groups that are very similar to each other; IL_81398.1 and IL_52958.1, IL_24982.5 and IL_05723.1, and IL_69536.1 and IL_58454.1. The first pair, IL_81398.1 and IL_52958.1 are also part of a larger cluster which includes IL_23639.1 and IL_46306.1. The last motif group in the ordering generated for the heatmap, IL_53323.1 in the lower right, also seems to be comparatively dissimilar to the other 5x5 motif groups, as well.

The heatmap is asymmetric because it is based on the asymmetric ROCP difference measure, but on the whole it is fairly close to symmetric. The most asymmetry occurs in the row and column associated with IL_70237.3. The column for IL_70237.3 is the lightest in the heatmap, while the row is darker yellows, with oranges indicating similarity with the IL_81398.1 to IL_46306.1 clus-

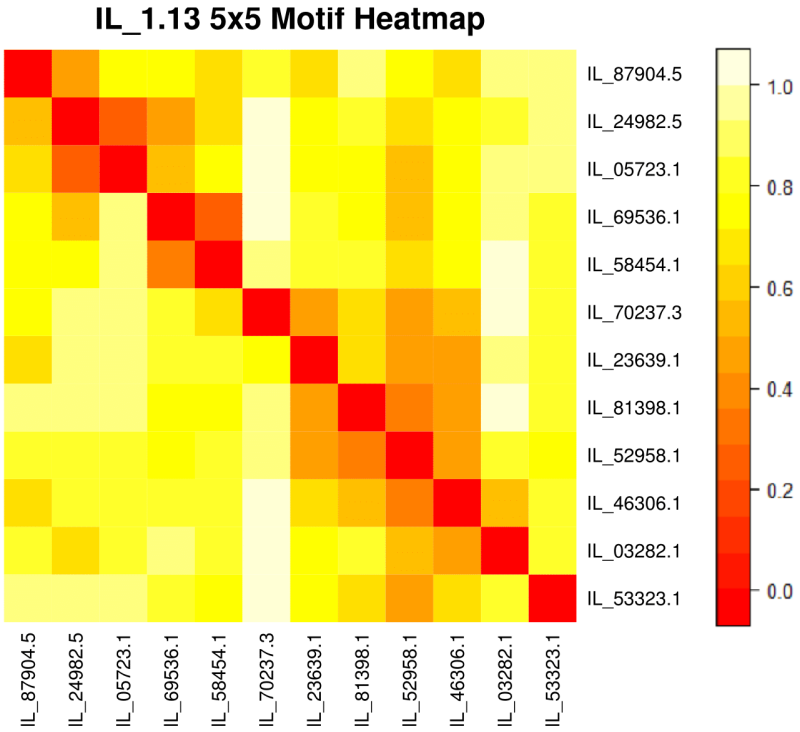


Figure 7.14 Heatmap using mutual ROCP dissimilarity for 5x5 internal loops. This heatmap uses the mutual ROCP dissimilarity for the 12 motif groups in IL_1.13 that have exactly 5 nucleotides on both strands for all 3D instances of the motif. It is asymmetric since it uses the asymmetric ROCP difference measure.

ter. This means that while the sequences ordered by IL_70237.3 score well against those models and alright against the rest, IL_70237.3 scores the sequences ordered by the other groups poorly. Investigation of IL_70237.3 reveals why this is the case. IL_70237.3 has only 2 basepairs beyond the flanking basepairs, but both of them are forming base triples with the flanking basepairs. These base triples make IL_70237.3 comparatively specific compared to other 5x5 motif groups, and the dependencies that single out the sequences which IL_70237.3 scores well are not present in the other 5x5 motif groups.

7.15 Heatmap for IL_1.13 sarcin-ricin motifs

We will next look at a heatmap for the sarcin-ricin motifs in IL_1.13. This is another group of 12 motif groups from IL_1.13, but with a different unifying feature. Sarcin-ricin motifs are defined by their basepairing pattern, particularly a defining basetriple. Sarcin-ricin motifs are discussed in more detail in Section 4.4. Because the basepairs that occur outside of the defining basepairing pattern can vary in number and type, there are a variety of sarcin-ricin motif groups of varying sizes, from 13 to 17 nucleotides. The heatmap for the IL_1.13 sarcin-ricin motifs is shown below in Figure 7.15.

There are a large number of ROCP distances near one, indicating little overlap in sequence space, especially in comparison to the the 5x5 IL_1.13 motifs. Much of these differences can be attributed to the differences in sequence length in the sarcin-ricin motifs. A difference of more than one nucleotide in the sequence length of the motif groups will result in very little overlap between the groups. This also explains some of the asymmetry in the heatmap, as the motif group models are much more likely to add a nucleotide as an insertion than they are to delete a nucleotide or basepair. This means when comparing a group to a slightly larger group, they smaller group is more likely to give higher probability to the sequences from the larger group than vice versa.

There are three fairly clear clusters in the heatmap. The strongest is actually in the similarity between the largest sarcin-ricin motifs, the 17 nucleotide groups, IL_54954.1 and IL_17682.1, which are very similar to each other and dissimilar to all the other sarcin-ricin motifs. IL_85647.3 and IL_76486.1, the 15 nucleotide sarcin-ricin groups also show similarities with each other but

are fairly dissimilar to the other sarcin-ricin motif groups. There is some asymmetric similarity between the 15 nucleotide sarcin-ricin motifs and the 14 nucleotide group IL_94973.1.

Three groups stand out as very strongly dissimilar to all other sarcin-ricin motif groups, IL_95652.3, IL_86981.1, and IL_71685.1. As was discussed in the section on ROCP comparison graphs, the 14 nucleotide sarcin-ricins, including IL_95652.3 and IL_71685.1, are very dissimilar to each other despite having the same number of nucleotides because of where the additional nucleotides occur in comparison to the basic sarcin-ricin geometry. IL_86981.1 has a bifurcated basepair, a basepair with with a water molecule in the middle of it. This makes it very dissimilar to the other sarcin-ricin motifs, even the other 15 nucleotide groups.

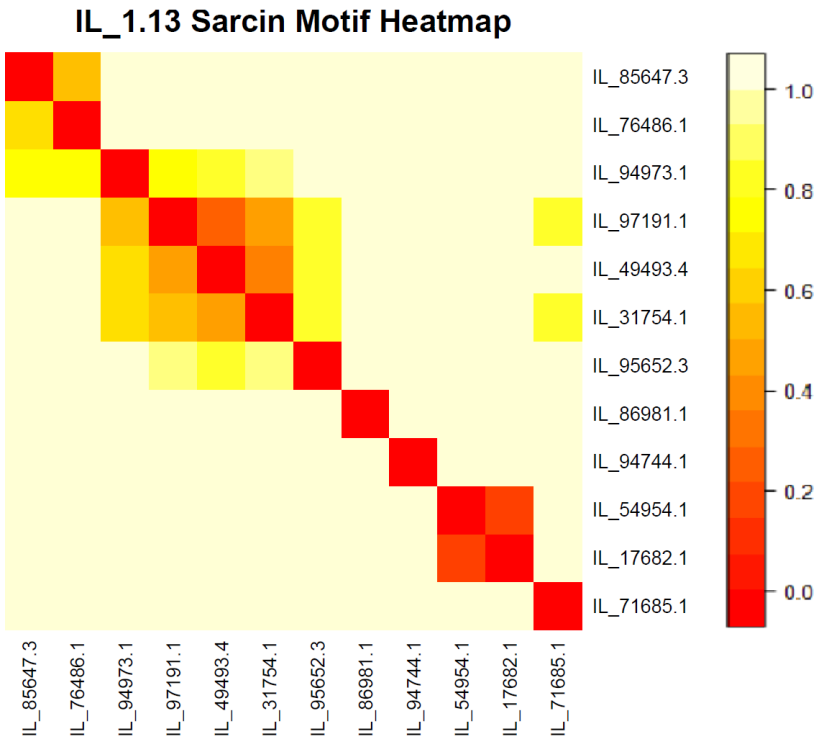


Figure 7.15 Heatmap using mutual ROCP dissimilarity for sarcin-ricin internal loops. This heatmap uses the ROCP dissimilarity measure for the 12 motif groups in IL_1.13 that are classified as sarcin-ricin motifs based on their basepairing pattern. It is asymmetric since it uses the asymmetric ROCP difference measure.

CHAPTER 8 FUTURE WORK

8.1 JAR3D for matching sequences to motif groups

One of the primary goals for JAR3D is that people studying RNA could use JAR3D to match sequences of RNA loops of unknown structure to motif groups. Many of our tools have been designed with this task in mind, but we have not yet published work showing how effective JAR3D is at this task. We have been waiting on two things before pursuing this publication.

The first is improvements to the RNA 3D Motif Atlas. The recent release of Motif Atlas version 3.2 is what we have been waiting for, and it provides several improvements. In particular, there are several new structures, as well as new structure quality indicator controls, that ensure that only x-ray structures that meet quality standards are included. These quality standards now work at the loop level, whereas in previous releases of the RNA 3D Motif Atlas quality indicators where only used at the structure level.

The other thing we need is high quality alignments that have been attached to 3D structures to use as testing data for the matching problem. Obtaining testing data has been difficult for a number of reasons. Because our training data is so sparse already, we cannot create a quality testing set by splitting our training data into two sets, as it will severely reduce the quality of our training data and produce a test data set that is still fairly small. Using sequence alignments for testing data can also be problematic. Firstly, we cannot be sure that the alignment is completely accurate, and many traditional alignment techniques can be especially unreliable near loop regions. JAR3D relies on loop sequences and their flanking Watson-Crick basepairs being accurately extracted from the overall RNA sequence. This issue can be overcome by using higher quality, often hand curated, alignments, but finding such alignment sets covering a sufficient number of different loop structures has proven difficult.

The other issue with sequence alignments is that we cannot be sure that the 3D structure that is attached to the alignment is conserved by all the sequences in the alignment and the loop level.

It is not unheard of for homologous RNAs to have different loop structures that perform the same function. This problem can be dampened by using high quality alignments between sequences for organisms that are fairly closely related from an evolutionary standpoint, as this will reduce the chances of different structures having evolved.

Despite the limitations, sequence alignments are still the best possible source for testing data for the JAR3D motif group matching application. Now that the new version of the Motif Atlas, and the accompanying JAR3D models, are available, the next priority for JAR3D will be finding alignments to use as data and seeking publication for an article on using JAR3D to match novell loop sequences to possible known 3D structures.

8.2 Improvements to JAR3D models

There are a number of potential improvements that could be made to the way that JAR3D models are made and parameterized. This section will discuss some of these possible improvements.

The first way JAR3D might be improved involves how IDI data is mapped to probability. The current function used to do this, pictured in Section 4.5 in Figure 4.6, was chosen carefully based on previous studies of basepair substitutions, but it is still an ad hoc function. We believe that a data driven approach might produce a better function that will produce better substitution matrices and more accurate models.

The data needed to make such a function will be difficult to obtain, however. One would need very accurate data, most likely from alignments, on exactly which nucleotides are involved in each of the different basepair families. Most non-Watson-Crick basepairs occur in loop regions, where many alignments are unreliable. One possible solution is use current JAR3D models to improve alignments, then use the improved alignments to improve the IDI to probability functions, and possibly repeat in an iterative fashion.

Such data could also be used to investigate if it is appropriate to use the same IDI to probability function for all basepairs. It is possible that different functions should be used for different basepair families, or for basepairs involved in base triples.

The way variable length insertions are handled in JAR3D might also be improved through data

driven research. We known that while insertions in loop regions are rarer than those in helices, they do occur and need to be accounted for. Because our training data is sparse and variable length insertions in loop regions are fairly rare, we cannot simply only model variable length insertions where they are observed.

Our current solution models variable length insertions in all locations where they have not been observed in the same way. Is this appropriate? Are there locations in loop regions where variable length insertions are more or less likely to occur, such as near base triples, near base-backbone interactions, or between specific families of basepairs? Answers to these questions could be used to better parameterize models, and hopefully make them less diffuse by reducing insertion probabilities in areas where they are even more unlikely to occur.

8.3 Other applications for JAR3D

Another possible use case for JAR3D is to return to JAR3D's originally intended purpose, aligning entire sequences to models for entire 3D structures. Sequence-based alignments that are not aware of 3D structure won't do as well in aligning loop regions in particular, and JAR3D could be used to produce better alignments. The improvements made to the JAR3D code for use with motifs will help with this application, as well. I did some work on doing this with the 5s region of the ribosome when I started working with the BGSU RNA group.

Even though this is the original use case for JAR3D, updates will be needed to make a useable tool. The code that makes JAR3D models for entire 3D structures can benefit from the lessons we learned working on loop motifs, as well as new features such as modeling variable length Watson-Crick helices. A web server could also be developed to put the tool in the hands of RNA researchers more easily.

BIBLIOGRAPHY

Akkuratov, E. E., L. Walters, A. Saha-Mandal, S. Khandekar, E. Crawford, C. L. Zirbel, S. Leisner, A. Prakash, L. Fedorova, and A. Fedorov (2014, Sep). Bioinformatics analysis of plant orthologous introns: identification of an intronic tRNA-like sequence. *Gene 548*(1), 81–90.

Bai, X. C., G. McMullan, and S. H. Scheres (2015, Jan). How cryo-EM is revolutionizing structural biology. *Trends Biochem. Sci. 40*(1), 49–57.

Baker, J. K. (1979). Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America 65*(S1), S132–S132.

Berman, H. M., J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne (2000). The protein data bank. *Nucleic acids research 28*(1), 235–242.

Cruz, J. A., M.-F. Blanchet, M. Boniecki, J. M. Bujnicki, S.-J. Chen, S. Cao, R. Das, F. Ding, N. V. Dokholyan, S. C. Flores, et al. (2012). Rna-puzzles: a casp-like evaluation of rna three-dimensional structure prediction. *Rna 18*(4), 610–625.

Cruz, J. A. and E. Westhof (2011). Sequence-based identification of 3d structural modules in rna with rmdetect. *Nature methods 8*(6), 513.

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

Das, R., J. Karanicolas, and D. Baker (2010). Atomic accuracy in predicting and designing non-canonical rna structure. *Nature methods 7*(4), 291.

DeSantis, T. Z., P. Hugenholtz, N. Larsen, M. Rojas, E. L. Brodie, K. Keller, T. Huber, D. Dalevi, P. Hu, and G. L. Andersen (2006). Greengenes, a chimera-checked 16s rrna gene database and workbench compatible with arb. *Appl. Environ. Microbiol. 72*(7), 5069–5072.

Eddy, S. R. and R. Durbin (1994). Rna sequence analysis using covariance models. *Nucleic acids research 22*(11), 2079–2088.

Griffiths-Jones, S., A. Bateman, M. Marshall, A. Khanna, and S. R. Eddy (2003). Rfam: an rna family database. *Nucleic acids research 31*(1), 439–441.

Holley, R. W., J. Apgar, G. A. Everett, J. T. Madison, M. Marquissee, S. H. Merrill, J. R. Penswick, and A. Zami (1965, Mar). Structure of a ribonucleic acid. *Science 147*(3664), 1462–1465.

Höner zu Siederdissen, C., S. H. Bernhart, P. F. Stadler, and I. L. Hofacker (2011). A folding algorithm for extended rna secondary structures. *Bioinformatics 27*(13), i129–i136.

Kashi, K., L. Henderson, A. Bonetti, and P. Carninci (2016, Jan). Discovery and functional analysis of lncRNAs: Methodologies to investigate an uncharacterized transcriptome. *Biochim. Biophys. Acta 1859*(1), 3–15.

Knudsen, B. and J. Hein (1999). Rna secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics (Oxford, England) 15*(6), 446–454.

Knudsen, B. and J. Hein (2003). Pfold: Rna secondary structure prediction using stochastic context-free grammars. *Nucleic acids research 31*(13), 3423–3428.

Kruse, H., M. Havrila, and J. Sponer (2014, Jun). QM Computations on Complete Nucleic Acids Building Blocks: Analysis of the Sarcin-Ricin RNA Motif Using DFT-D3, HF-3c, PM6-D3H, and MM Approaches. *J Chem Theory Comput 10*(6), 2615–2629.

Lemieux, S. and F. Major (2006). Automated extraction and classification of rna tertiary structure cyclic motifs. *Nucleic acids research 34*(8), 2340–2346.

Leontis, N. B., J. Stombaugh, and E. Westhof (2002). The non-watson–crick base pairs and their associated isostericity matrices. *Nucleic acids research 30*(16), 3497–3531.

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

Miao, Z., R. W. Adamiak, M. Antczak, R. T. Batey, A. J. Becka, M. Biesiada, M. J. Boniecki, J. M. Bujnicki, S.-J. Chen, C. Y. Cheng, et al. (2017). Rna-puzzles round iii: 3d rna structure prediction of five riboswitches and one ribozyme. *Rna 23*(5), 655–672.

Miao, Z., R. W. Adamiak, M.-F. Blanchet, M. Boniecki, J. M. Bujnicki, S.-J. Chen, C. Cheng, G. Chojnowski, F.-C. Chou, P. Cordero, et al. (2015). Rna-puzzles round ii: assessment of rna structure prediction programs applied to three large rna structures. *Rna 21*(6), 1066–1084.

Nawrocki, E. P., D. L. Kolbe, and S. R. Eddy (2009). Infernal 1.0: inference of rna alignments. *Bioinformatics 25*(10), 1335–1337.

Parisien, M. and F. Major (2008). The mc-fold and mc-sym pipeline infers rna structure from sequence data. *Nature 452*(7183), 51.

Pedersen, J. S., G. Bejerano, A. Siepel, K. Rosenbloom, K. Lindblad-Toh, E. S. Lander, J. Kent, W. Miller, and D. Haussler (2006). Identification and classification of conserved rna secondary structures in the human genome. *PLoS computational biology 2*(4), e33.

Petrov, A. I., C. L. Zirbel, and N. B. Leontis (2013, Oct). Automated classification of RNA 3D motifs and the RNA 3D Motif Atlas. *RNA 19*(10), 1327–1340.

Popenda, M., M. Błażewicz, M. Szachniuk, and R. W. Adamiak (2007). Rna frabase version 1.0: an engine with a database to search for the three-dimensional fragments within rna structures. *Nucleic acids research 36*(suppl_1), D386–D391.

Popenda, M., M. Szachniuk, M. Antczak, K. J. Purzycka, P. Lukasiak, N. Bartol, J. Blazewicz, and R. W. Adamiak (2012). Automated 3d structure composition for large rnas. *Nucleic acids research 40*(14), e112–e112.

Quast, C., E. Pruesse, P. Yilmaz, J. Gerken, T. Schweer, P. Yarza, J. Peplies, and F. O. Glöckner (2012). The silva ribosomal rna gene database project: improved data processing and web-based tools. *Nucleic acids research 41*(D1), D590–D596.

Rahrig, R. R., N. B. Leontis, and C. L. Zirbel (2010). R3d align: global pairwise alignment of rna 3d structures using local superpositions. *Bioinformatics 26*(21), 2689–2697.

Reinharz, V., F. Major, and J. Waldispühl (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.

Rivas, E. and S. R. Eddy (2001). Noncoding rna gene detection using comparative sequence analysis. *BMC bioinformatics 2*(1), 8.

Robertus, J. D., J. E. Ladner, J. T. Finch, D. Rhodes, R. S. Brown, B. F. Clark, and A. Klug (1974, Aug). Structure of yeast phenylalanine tRNA at 3 A resolution. *Nature 250*(467), 546–551.

Roll, J., C. L. Zirbel, B. Sweeney, A. I. Petrov, and N. Leontis (2016, 07). JAR3D Webserver: Scoring and aligning RNA loop sequences to known 3D motifs. *Nucleic Acids Res. 44*(W1), W320–327.

Sakakibara, Y., M. Brown, R. Hughey, I. S. Mian, K. Sjölander, R. C. Underwood, and D. Haussler (1994). Stochastic context-free grammers for trna modeling. *Nucleic acids research 22*(23), 5112–5120.

Samhita, L. and U. Varshney (2010). The ribosome and the 2009 nobel prize in chemistry. *Resonance 15*(6), 526–537.

Sarver, M. (2006). *Structure-based multiple RNA sequence alignment and finding RNA motifs*. Ph. D. thesis, Bowling Green State University.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal 27*(3), 379–423.

Stombaugh, J., C. L. Zirbel, E. Westhof, and N. B. Leontis (2009). Frequency and isostericity of rna base pairs. *Nucleic acids research 37*(7), 2294–2312.

Theis, C., C. Höner zu Siederdissen, I. L. Hofacker, and J. Gorodkin (2013). Automated identification of rna 3d modules with discriminative power in rna structural alignments. *Nucleic acids research 41*(22), 9999–10009.

Theis, C., C. L. Zirbel, C. H. Zu Siederdissen, C. Anthon, I. L. Hofacker, H. Nielsen, and J. Gorodkin (2015). RNA 3D Modules in Genome-Wide Predictions of RNA 2D Structure. *PLoS ONE 10*(10), e0139900.

Turner, D. H. and D. H. Mathews (2009). Nndb: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic acids research 38*(suppl_1), D280–D282.

Yao, J., V. Reinharz, F. Major, and J. Waldispühl (2017). Rna-moip: prediction of rna secondary structure and local 3d motifs from sequence data. *Nucleic acids research 45*(W1), W440–W444.

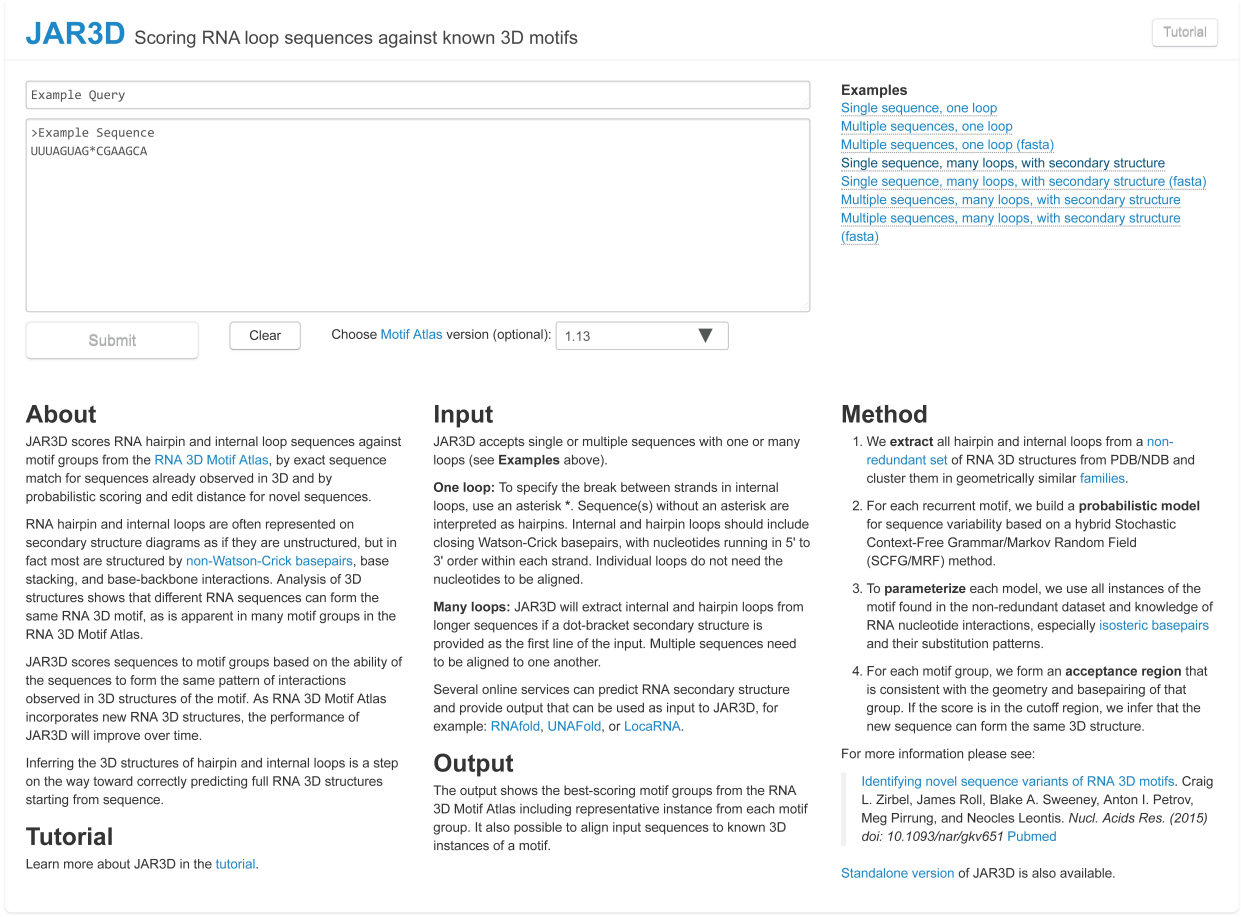
Younger, D. H. (1967). Recognition and parsing of context-free languages in time n3. *Information and control 10*(2), 189–208.

Zhong, C. and S. Zhang (2015). Rnamotifscanx: a graph alignment approach for rna structural motif identification. *RNA 21*(3), 333–346.

Zirbel, C. L., J. Roll, B. A. Sweeney, A. I. Petrov, M. Pirrung, and N. B. Leontis (2015, Sep). Identifying novel sequence variants of RNA 3D motifs. *Nucleic Acids Res. 43*(15), 7504–7520.

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

APPENDIX A JAR3D WEB SERVER INPUT PAGE EXAMPLE



APPENDIX B JAR3D WEB SERVER LOOP LEVEL OUTPUT PAGE EXAMPLE

JAR3D Query b84636a9-9c1f-4e08-addc-f13ac72e9284 completed

Run on Motif Atlas Version 1.13

Input Summary

```
UUUAGUAG*CGAAGCA
```

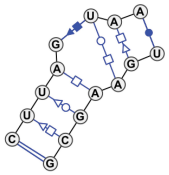
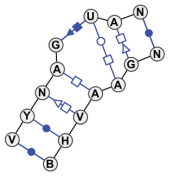
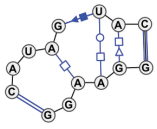
Refine query

Loop 1

Column Numbers: 1-14

```
UUUAGUAG*CGAAGCA
```

Click on headings to reorder table - View All Results for this Loop

#	Matching motif	Acceptance Rate	Mean Cutoff Score	Full Edit Distance	Interior Edit Distance	2D Diagram
1	**Motif IL_76486.1** Basepair signature: cWW-tSH-tSW-tHH-cSH-tWH-tHS-cWW Align sequences	100.00	98.58	4	0	
2	**Motif IL_85647.3** Basepair signature: cWW-cWW-tSH-tHH-cSH-tWH-tHS-cWW Align sequences	100.00	62.16	3	1	
3	**Motif IL_94973.1** Basepair signature: cWW-L-R-L-tHH-cSH-tWH-tHS-cWW Align sequences	100.00	33.47	6	2	

APPENDIX C JAR3D WEB SERVER ALIGNMENT PAGE EXAMPLE

JAR3D Query d74628d5-c66b-4125-84ef-046b84923ae3

Alignment to IL_85647.3

Run on Motif Atlas Version 1.13

Input sequences are initially listed on the top, with a grey background. Sequences from 3D structures are initially listed on the bottom, with a white background.
Click on a column header to sort by that column. Holding shift and clicking on multiple column headers will sort by each column in order.
Meets Cutoff is True if the sequence meets the group's cutoff critera and False if it does not.
Cutoff Score ranges from 0 to 100 for sequences that meet the cutoff, with 100 being the best possible score. Sequences outside the cutoff region have a negative score.
Interior edit distance is the minimum Levenshtein edit distance from the sequence to a sequence from 3D, not counting flanking basepairs. Full edit distance includes the flanking basepairs.
The alignment distance columns show the differences from the named sequence to the sequence in each row, summing the edit distances in each column of the alignment.

Column Position Insertion	1 I	2 1	3 I	4 2	5 I	6 3	7 I	8 4	9 I	10 5	11 I	12 6	13 I	14 7	15 I	16 8	17 I	18	19 I	20 9	21 I	22 10	23 I	24 11	25 I	26 12	27 I	28 13	29 I	30 14	31 I	32 15	33 I	Meets Cutoff	Cutoff Score	Interior Edit Distance	Full Edit Distance	Alignment Distance to >Example Sequence
>Example Sequence		U		U		U		A		G		U		A		G		*		C		G		A		A		G		C		A		True	62.163918	1	3	0
IL_1JBR_001		C		U		C		A		G		U		A		C				G		G		A		A		C		C		G						6
IL_1Q96_001		C		U		C		A		G		U		A		U				A		G		A		A		C		C		G						6
IL_1Q96_003		C		U		C		A		G		U		A		U				A		G		A		A		C		C		G						6
IL_1S72_006		C		U		C		A		G		U		A		U				A		G		A		A		C		U		G						7
IL_2QBG_011		C		U		A		A		G		U		A		C				G		G		A		A		C		U		G						7
IL_2QBG_048		A		U		A		A		G		U		A		A				U		G		A		A		A		U		U						7
IL_3U5H_122		C		U		U		A		G		U		A		C				G		G		A		A		C		A		G						6
IL_3U5H_148		A	C	C		G		A		G		U		A		G				C		G		A		A		A		C		U						6
IL_3V2F_006		C		U		C		A		G		U		A		C				G		G		A		A		C		U		G						7
IL_3V2F_046		A		U		G		A		G		U		A		A				U		G		A		A		A		U		U						7
IL_4A1B_003		C		C		C		A		G		U		A		A				U		G		A		A		C		A		G						8
IL_4A1B_118		G		U		U		A		G		U		A		C				G		G		A		A		C		A		C						6
IL_4A1C_008		G		U		U		A		G		U		A		C				G		G		A		A		G		U		C						5
IL_4IOA_006		C		U		C		A		G		U		A		C				G		G		A		A		C		U		G						7
IL_4IOA_007		G		U		U		A		G		U		A		G				C		G		A		A		C		C		C						3
IL_4IOA_047		A		U		G		A		G		U		A		A				U		G		A		A		A		U		U						7

APPENDIX D PYTHON IMPLEMENTATION OF TOP K ALGORITHM

```
import sys
import itertools
import math
from operator import itemgetter

def main(infile, outfile, memfile, limit):
    fo = open(outfile, "w")
    fm = open(memfile, "w")

    probs, chars = read_data(infile)

    limit = int(limit)

    n = len(probs)

    top = []
    top_set = set()
    candidates = []
    soon = dict()

    first = [0]*n
    top.append((first,prob_product(probs,first)))
    top_set.add(get_sequence(first,chars))

    for i in range(n):
        can = [0]*n
```

```
    if len(probs[i]) > 1:
        can[i] = 1
        candidates.append((can,prob_product(probs,can)))
i = 1

fm.write('K,Cans,Soon'+'\n')

while i < limit:
    if len(candidates) == 0:
        break
    next_best = max(candidates,key=itemgetter(1))
    candidates.remove(next_best)
    seq = get_sequence(next_best[0],chars)
    if seq not in top_set:
        top_set.add(seq)
        top.append(next_best)
        i = i + 1
    (candidates,soon) = update(candidates,soon,probs,next_best,n)
    if(len(top)%10000 == 0):
        print ("Found " + str(len(top)) + "  CProb = " +
            str(next_best[1]) + "  Cans = " + str(len(candidates))
            + "  Soon = " + str(len(soon)))
for ind,dup in enumerate(top):
    fo.write('>'+str(ind+1)+'_'+str(dup[0]).replace(',',';')+
            '_'+str(dup[1])+'\n')
    fo.write(get_sequence(dup[0],chars)+'\n')
fo.close()
fm.close()
```

```
def prob_product(probs,inds):
    out = 1
    for ind,i in enumerate(inds):
        node = probs[ind]
        prob = node[i]
        out = out*prob
    return out

def update(candidates,soon,probs,next_best,n):
    inds = next_best[0]
    for i in range(len(inds)):
        soon_inds = list(inds)
        soon_inds[i] += 1
        if soon_inds[i] >= len(probs[i]):
            continue
        key = str(soon_inds)
        prob = prob_product(probs,soon_inds)
        if key in soon:
            soon[key] = soon[key] + 1
        else:
            soon[key] = 1 + soon_inds.count(0)
        if soon[key] == n:
                candidates.append((soon_inds,prob))
                del soon[key]
    return (candidates,soon)

def get_sequence(nums,chars):
    size = len(nums)
    seq = chars[0][nums[0]]
```

```
    for i in range(1,size):
        j = nums[i]
        seq = seq.replace('&',chars[i][j])
    seq = seq.replace('&','')
    return seq

def read_data(filename):
    with open(filename, 'r') as raw:
        lines = raw.readlines()
        probs = []
        chars = []
        for line in lines:
            parts = line.split('\t')
            node = int(parts[0])
            prob = float(parts[3].strip())
            if len(probs) >= node:
                probs[node-1].append(prob)
                chars[node-1].append(parts[1] + '&' + parts[2])
            else:
                probs.append([prob])
                chars.append([parts[1] + '&' + parts[2]])
        return (probs,chars)

if __name__== "__main__":
    infile = sys.argv[1]
    outfile = sys.argv[2]
    memfile = sys.argv[3]
    limit = sys.argv[4]
    main(infile,outfile,memfile,limit)
```