Modeling Complex RNA Tertiary Folds with Rosetta

Clarence Yu Cheng*, Fang-Chieh Chou*, Rhiju Das*,†,1
*Department of Biochemistry, Stanford University, Stanford, California, USA
†Department of Physics, Stanford University, Stanford, California, USA
1Corresponding author: e-mail address: rhiju@stanford.edu

Contents

Abstract

Reliable modeling of RNA tertiary structures is key to both understanding these structures' roles in complex biological machines and to eventually facilitating their design for molecular computing and robotics. In recent years, a concerted effort to improve computational prediction of RNA structure through the RNA-Puzzles blind prediction trials has accelerated advances in the field. Among other approaches, the versatile and expanding Rosetta molecular modeling software now permits modeling of RNAs in the 100–300 nucleotide size range at consistent subhelical (~1 nm) resolution. Our laboratory's current state-of-the-art methods for RNAs in this size range involve Fragment Assembly of RNA with Full-Atom Refinement (FARFAR), which optimizes RNA conformations in the context of a physically realistic energy function, as well as hybrid techniques that leverage experimental data to inform computational modeling. In this chapter, we give a practical guide to our current workflow for modeling RNA three-dimensional structures using FARFAR, including strategies for using data from multidimensional chemical mapping experiments to focus sampling and select accurate conformations.

Methods in Enzymology
ISSN 0076-6879
http://dx.doi.org/10.1016/bs.mie.2014.10.051

1. INTRODUCTION

Computational modeling of RNA structures is advancing rapidly, with recent developments improving prediction and design of both secondary and tertiary structures of RNA. Continuing improvements to secondary structure prediction algorithms (Tinoco et al., 1973), classification of RNA structural motifs (Petrov, Zirbel, & Leontis, 2013), molecular dynamics and quantum mechanical techniques (Ditzler, Otyepka, Sponer, & Walter, 2010), conformational sampling with energy scoring (Das, Karanicolas, & Baker, 2010), atomic-scale loop and motif modeling (Sripakdeevong, Kladwang, & Das, 2011), integration with conventional crystallographic (Chou, Sripakdeevong, Dibrov, Hermann, & Das, 2013) and NMR approaches (Sripakdeevong et al., 2014), and connections with recent single-molecule (Chou, Lipfert, & Das, 2014) and internet-scale videogame (Lee et al., 2014) technologies hold promise for eventually attaining confident 3D modeling and design of RNAs with high spatial resolution. An important driver of recent innovation has been the establishment of blind prediction trials, proposed during a community-wide collation of 3D RNA modeling methods in 2010 (Sripakdeevong, Beauchamp, & Das, 2012) and begun soon thereafter. The RNA-Puzzles trials (Cruz et al., 2012), modeled after the 20-year-old CASP trials in protein structure prediction, challenge participating groups to create accurate 3D models of RNAs from sequence alone; the submitted models are compared to unreleased crystallographic structures of the targets to assess the methods' predictive power. These trials provide a rigorous testing ground for current computational as well as hybrid experimental/computational structure prediction methods on RNA domains that are of strong biological interest.

This chapter describes methods from our laboratory of medium computational and experimental expense that achieve subhelix-resolution accuracy for 3D models of 100- to 300-nucleotide RNAs, a typical size range for many riboswitch and ribozyme domains and representative of RNA-Puzzles target sizes. Subhelical resolution, while not the ultimate achievable, has still been useful in guiding mutational experiments *in vitro* and *in vivo*, detecting partial structure in riboswitches without their ligands, and in revealing or illustrating evolutionary connections that are not obvious from sequence comparisons alone. The primary tools for this approach are constraints from chemical mapping experiments, which we discuss briefly here and will be described in more detail elsewhere, and computational modeling to integrate chemical mapping data into 3D portraits.

Our laboratory is developing several tools that seek to advance 3D macromolecule modeling at multiple length scales. For small RNA motifs, we leverage algorithms based on a "stepwise ansatz," which enable modeling of RNA loops and motifs with near-atomic accuracy (better than 2 Å RMSD), particularly if limited NMR or crystallographic data are available (Chou et al., 2013; Sripakdeevong et al., 2014, 2011). Unfortunately, the computational expense of those high-resolution tools is currently prohibitive for *de novo* modeling of large RNAs. Instead, our practical tools for large RNAs have largely been built on Fragment Assembly of RNA with Full-Atom Refinement (FARFAR) in the Rosetta framework, which was first introduced to model small motifs of RNAs in 2007 (Das & Baker, 2007) and was initially based on Rosetta protein structure prediction methods that we had helped in advance. Since that time, FARFAR has been progressively developed to allow for nucleotide-resolution building of not just individual RNA motifs but also more complex RNA folds involving dozens of helices. This chapter is intended to offer a practical guide to getting started with Rosetta using an up-to-date workflow from our laboratory, laid out in Fig. 1. We will illustrate this workflow below using the ligand-binding region of a tandem glycine-binding riboswitch from *F. nucleatum*, which forms a complex pseudosymmetric fold stabilized by A-minor interactions between two glycine-binding subdomains. A homolog of this domain was posed as an RNA-Puzzles challenge (Cruz et al., 2012), and crystallographic and biochemical work on this system by several RNA laboratories (Butler, Xiong, Wang, & Strobel, 2011; Cordero, Kladwang, VanLang, & Das, 2012; Erion & Strobel, 2011; Kladwang, VanLang, Cordero, & Das, 2011) have made this RNA a useful model system for calibrating and illustrating experimental and computational methodologies.

2. SETTING THE STAGE FOR 3D MODELING USING EXPERIMENTAL DATA

Several pieces of information can provide powerful constraints to help construct accurate 3D models of RNA. The most fundamental of these is the RNA's secondary structure. If phylogenetic inference of secondary structure is precluded by the lack of sequence homologs, difficulties in sequence alignment, or targeting of "alternative" states of the RNA (e.g., without ligands or in misfolded conformations), chemical mapping techniques provide useful guides to computational secondary structure prediction (Cordero, Kladwang, VanLang, & Das, 2014; Hajdin et al., 2013; Kladwang et al., 2011). In traditional "one-dimensional" (1D) chemical mapping experiments, solution-state

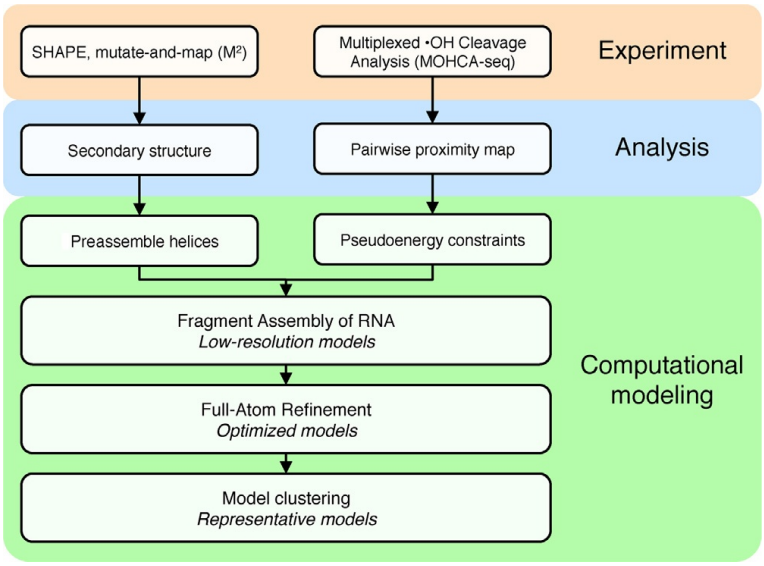

Figure 1 Workflow for modeling RNA structures in the Rosetta framework guided by experimental data. One-dimensional chemical mapping and mutate-and-map methods guide confident secondary structure prediction. To save computational expense during global modeling, secondary structure elements are separately preassembled. These ensembles of preassembled helices, along with experimental proximity mapping data from MOHCA-seq, are the inputs to global modeling by Fragment Assembly of RNA (FARNA), which generates low-resolution models. A fraction of the low-resolution models with the lowest Rosetta energy scores are then minimized using the Rosetta all-atom energy function (FARNA with Full-Atom Refinement, FARFAR) to resolve chain breaks and unreasonable local geometries that can arise from fragment insertion. Finally, the minimized models are clustered using an RMSD threshold to collect 0.5% of the total low-resolution models in the largest cluster; this step identifies representative conformations sampled by the algorithm.

RNAs are exposed to chemical modifiers which form adducts to the backbone or nucleobases depending on backbone flexibility or base-pairing status (Fig. 2A). These modifications are traditionally detected by reverse transcription, which stops at the modified location, followed by gel or capillary electrophoresis or, more recently, deep sequencing to identify the sequence position of each modification. The reactivity of each nucleotide position to the chemical modifier can be quantified using several publically available software suites, with HiTRACE (Kim, Cordero, Das, & Yoon, 2013; Yoon et al., 2011) (https://github.com/hitrace/hitrace) and MAPseeker (Seetin et al., 2014) (https://github.com/DasLab/map_seeker) particularly optimized for high-throughput analysis of capillary electrophoresis and deep-sequencing data, respectively. Secondary structure prediction servers such as RNAstructure (Reuter & Mathews, 2010) (http://rna.urmc.rochester.edu/

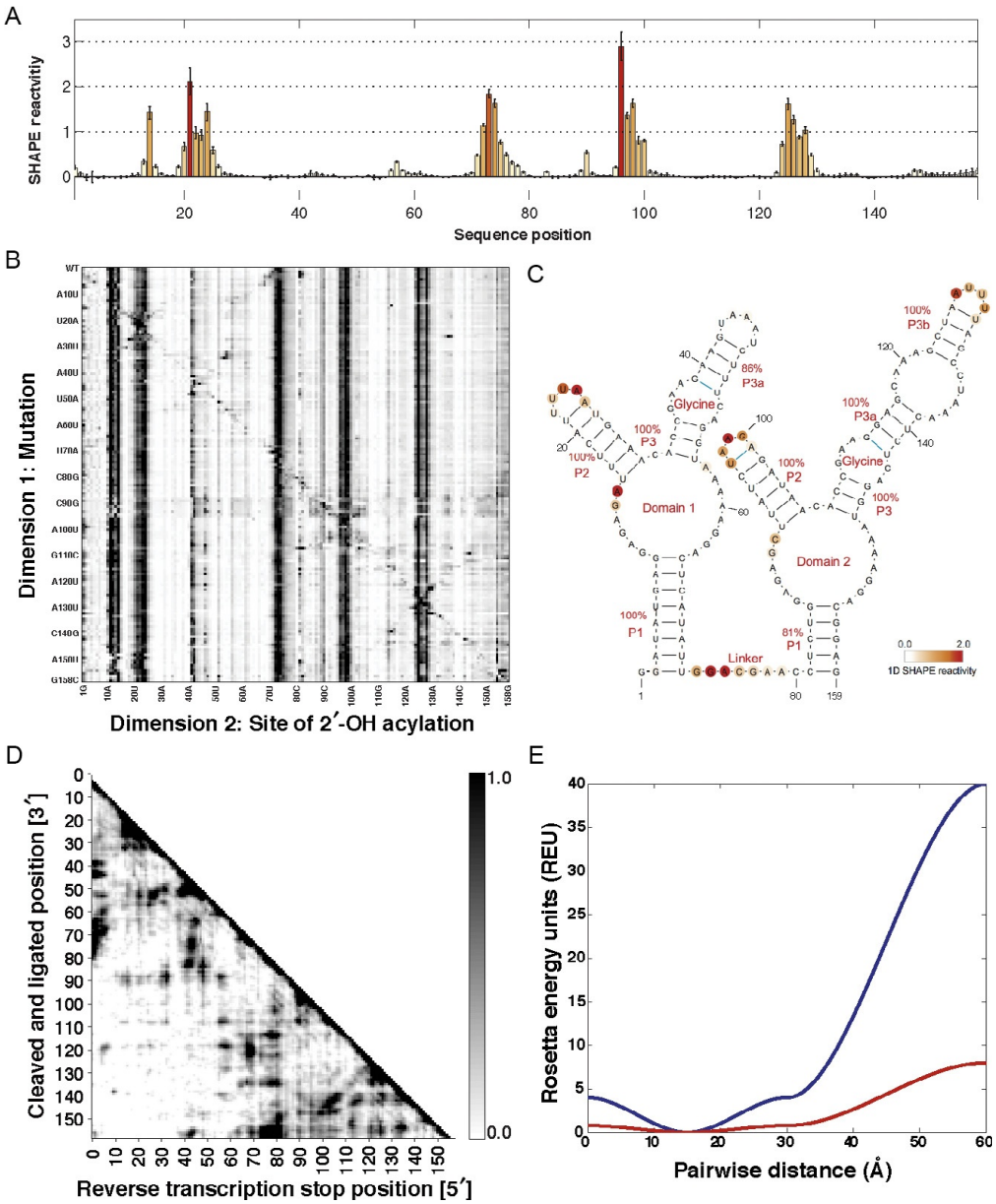

Figure 2 Rapidly acquired chemical mapping data for modeling a complex RNA fold. (A) One-dimensional SHAPE chemical mapping data for the *F. nucleatum* glycine riboswitch double ligand-binding domain in the presence of 10 m*M* glycine. Reactivities are normalized to reference hairpins (not shown) (Kladwang et al., 2014). Data are available at the RNA Mapping Database (RMDB, http://rmdb.stanford.edu) under accession code GLYCFN_1M7_0005. (B) Mutate-and-map (M^2) chemical mapping data for the glycine riboswitch in the presence of 10 m*M* glycine. Data are available at the RMDB under accession code GLYCFN_SHP_0002. (C) M^2-derived secondary structure model of the glycine riboswitch in the presence of 10 m*M* glycine, from Kladwang et al. (2011). Blue lines indicate Watson–Crick base pairs predicted in the model but not present in the crystallographic secondary structure. Red percentage values for each helix indicate confidence estimates from bootstrapping two-dimensional SHAPE chemical

(Continued)

RNAstructureWeb) and the RNA mapping database structure server (Cordero, Lucks, & Das, 2012) (http://rmdb.stanford.edu/structureserver) can accept reactivities from chemical mapping experiments, providing additional scoring terms to guide the predictions. Nonparametric bootstrapping (Kladwang et al., 2011) can provide confidence estimates for these models.

While 1D chemical mapping experiments can provide reactivity values for every nucleotide in an RNA, the data do not directly reveal which nucleotides are base paired with which other nucleotides in the sequence, which generally limits the accuracy of the resulting models. Higher confidence secondary structures can be derived from multidimensional expansions of conventional chemical mapping. For example, the "mutate-and-map" (M^2) approach (Cordero et al., 2014) (Fig. 2B) involves systematic mutagenesis of every residue in the RNA; the suite of mutated RNAs are chemically mapped in parallel. The mutations disrupt individual Watson–Crick and noncanonical base pairs, causing the base-pairing partners of the mutated residues to increase in reactivity to the chemical modifier. Thus, M^2 can identify the base-pairing interactions throughout RNAs, which provide powerful restraints for secondary structure prediction and, in some cases, can reveal base interaction-mediated tertiary contacts (Kladwang, Chou, & Das, 2012). For the glycine riboswitch domain, M^2 was able to automatically and blindly predict the secondary structure of the domain, recovering all helices correctly and with confidence, as assessed by bootstrapping. In all cases tested to date, including blind RNA-Puzzles test cases, M^2 models achieve such accuracy; all residual errors involve helix edge base pairs (Fig. 2C). High-throughput mutation-rescue experiments read out by chemical mapping now offer the prospect of testing secondary structures at base pair resolution, and we

Figure 2—Cont'd mapping data. Nucleotides are colored according to SHAPE reactivity. (D) MOHCA-seq proximity map of the glycine riboswitch in the presence of 10 m*M* glycine, from Cheng et al. (2014). The *y*-axis represents positions that were cleaved by hydroxyl radicals, while the *x*-axis represents the locations of the radical sources from which the radicals originated. Pairwise positions are colored according to two-point correlation calculated by MAPseeker analysis (Seetin, Kladwang, Bida, & Das, 2014). Data are available at the RMDB under accession code GLYCFN_MCA_0000. (E) Pseudoenergy potential applied during modeling in Rosetta to constrain pairs of residues indicated to be in proximity by MOHCA-seq experimental data. Residue pairs showing strong MOHCA-seq signal are constrained with the blue potential and those with weaker signal are constrained with the red potential (1/5 of the blue potential).

recommend compensatory rescue tests for problems that require particularly high confidence (Tian, Cordero, Kladwang, & Das, 2014).

Another form of information that can be critical for selecting an RNA's correct 3D fold involves pairwise proximities, which reflect the topology of the tertiary structure. An experimental pipeline, Multiplexed hydroxyl radical (·OH) Cleavage Analysis by paired-end sequencing (MOHCA-seq), has been developed that can collect such pairwise proximity information, independent of traditional 3D structure determination techniques such as X-ray crystallography, cryo-EM, and NMR. In MOHCA-seq, sources of hydroxyl radicals are randomly incorporated into the RNA backbone during transcription (Cheng et al., 2014; Das et al., 2008). Activation of the sources produces localized hydroxyl radicals that diffuse outward, causing strand breaks at positions that are far away in sequence from the radical source but are brought into proximity by the 3D fold. In order to identify the locations of cleavage events and the radical sources that caused them, a DNA tail is ligated to the 3′-end of the fragmented RNAs, and reverse transcription primed on this tail stops at the radical source location. Sequencing of these complementary DNA fragments and analysis using the MAPseeker software (Seetin et al., 2014) produces pairwise proximity maps of the RNA's tertiary structure (Fig. 2D). MOHCA-seq data can be incorporated into 3D modeling via pseudoenergy terms (Cheng et al., 2014; Das et al., 2008) (Fig. 2E), as is described in further detail below.

3. MAKING MODELS OF RNA TERTIARY FOLDS

Our overall modeling pipeline still requires some manual setup of steps and has not been fully automated, mainly because it is under rapid development but also because particular steps depend on the computer cluster on which the code is tested or executed (see later). Nevertheless, it is currently fully functional without expert inspection. The following is a procedure optimized to make use of constraints from chemical mapping experiments.

3.1. Installing software and accessing computation resources

The principal framework for RNA computational modeling using our workflow is Rosetta, a collaboratively developed software suite for structure prediction and engineering of a wide range of macromolecules (https://www.rosettacommons.org/) (Leaver-Fay et al., 2011). Documentation for Rosetta can be found online (https://www.rosettacommons.org/docs/

latest/) and the modular design of the software has been described in detail (Leaver-Fay et al., 2011). Noncommercial users can install Rosetta by requesting a free license from RosettaCommons Web site, and then downloading and installing the software from the same site. Users can select which build of Rosetta to compile; we recommend that Mac users compile the `build_mac_graphics` version, which provides real-time visualization of conformational sampling and Linux users to compile the `build_release` version. General installation instructions are provided in `Rosetta/main/source/cmake/README` (see also: https://www.rosettacommons.org/docs/latest/Build-Documentation.html). Rosetta is consistently updated with weekly build releases, and the command lines referenced later in the text and given in the Appendix have been tested using a recent weekly build (weekly_releases/2014_35_57232). Beyond the core Rosetta installation, we are also developing an additional set of tools for RNA modeling, which are required for the workflow described in this chapter. The RNA tools collection is located in `Rosetta/tools/rna_tools/bin`, and documentation for setting up RNA tools is available on RosettaCommons (https://www.rosettacommons.org/docs/latest/RNA-tools.html).

The PyMOL open-source molecular visualization tool is helpful for inspecting and evaluating structural models (http://www.PyMOL.org/) (Schrodinger, 2010). Free educational subscriptions to PyMOL are available at the Web site; there is a fee for other users. Our laboratory's tools for easy visualization of RNA models in PyMOL are freely available on GitHub (https://github.com/DasLab/PyMOL_daslab). These scripts include commands to render RNAs with various levels of molecular detail, as well as to superimpose models and to color models by chemical mapping reactivities.

Most of the modeling protocols in Rosetta cannot be completed on single laptops but can be easily run on UNIX computer clusters. Sufficient computing power can be obtained from some freely available resources. For example, the Extreme Science and Engineering Discovery Environment (XSEDE, https://www.xsede.org/home) provides free startup allocations for high-performance computation. At the time of writing, 20,000 CPU hours can be acquired by research laboratories within a short time of submitting an allocation request, and this amount is more than enough to carry out several calculations. We typically carry out trial runs on local Macintosh machines and then transfer files to XSEDE or other resources for parts of the calculation that require large-scale runs.

We note that modeling of submotifs (up to 30 nucleotides) of a large RNA can also be carried out freely through the Rosetta Online Server that Includes Everyone (ROSIE, http://rosie.rosettacommons.org) (Lyskov et al., 2013), and, if desired, these submodels can be integrated into larger models (see Section 3.6). Runs on ROSIE may be useful to groups who wish to explore these tools before compiling and executing Rosetta RNA modeling on their own resources or on XSEDE.

3.2. Preassembling helices

An important principle in efficient macromolecular modeling is to not expend computation on regions of already known structure. For RNA, most helices form canonical A-form conformations. Therefore, to reduce computational expense, we preassemble the helices from high-confidence secondary structures that were predicted using chemical mapping (e.g., M^2) data.

First, we make a directory in which modeling of the target RNA will be performed. In this directory, we create a FASTA-formatted file with the name and sequence of the target RNA and a file with the secondary structure of the RNA in dot–parenthesis notation. Pseudoknots may be expressed in square brackets instead of parentheses. For example, FASTA files, secondary structure files, and UNIX command lines can be found in the Appendix and will be referenced in the text. Examples of initial FASTA and secondary structure files are given as files [F1] and [F2] in the Appendix, respectively.

To generate files containing the command lines for *de novo* RNA helix modeling in Rosetta, we run the `helix_preassemble_setup.py` script with the secondary structure and FASTA files as inputs (Appendix, command line [1]). The `helix_preassemble_setup.py` script will generate parameter and FASTA files for each helix detected in the input secondary structure, as well as a .RUN file that contains the command line for `rna_denovo`, the program that performs *de novo* RNA modeling in Rosetta. The files will be named according to order of helices in the secondary structure (e.g., `helix0.params`, `helix0.fasta`, `helix0.RUN`, `helix1.params`). The content of a `helix0.RUN` file should resemble command line [2] in the Appendix. This .RUN file can be run on a local machine in 10–20 min using `source helix0.RUN` (Appendix, command line [3]) and generates 100 FARFAR models for each helical region. The resulting models are output in compressed format (called "silent files" in Rosetta, for historical reasons) with names like `helix0.out`, etc. These files will be used as inputs for global modeling of the entire RNA. The helix models can be visualized, if desired,

using the `extract_lowscore_decoys.py` script (see also below). The preassembled helices are generally nearly identical except for small variations near the ends (Fig. 3). Sampling the helices in the target RNA from these models instead of from the database of RNA fragments used for global sampling allows a greater portion of the computational effort to be spent on nonhelical regions.

3.3. Defining the global fold using fragment assembly of RNA

With experimental constraints and preassembled helices in hand, the global fold of the target RNA can be tackled. At this stage, we create a set of low-resolution models using Fragment Assembly of RNA (FARNA) (Das & Baker, 2007). In FARNA, models are assembled using small RNA fragments sampled from a crystallographic database using a Monte Carlo algorithm. This heuristic allows the models to take on RNA-like conformations

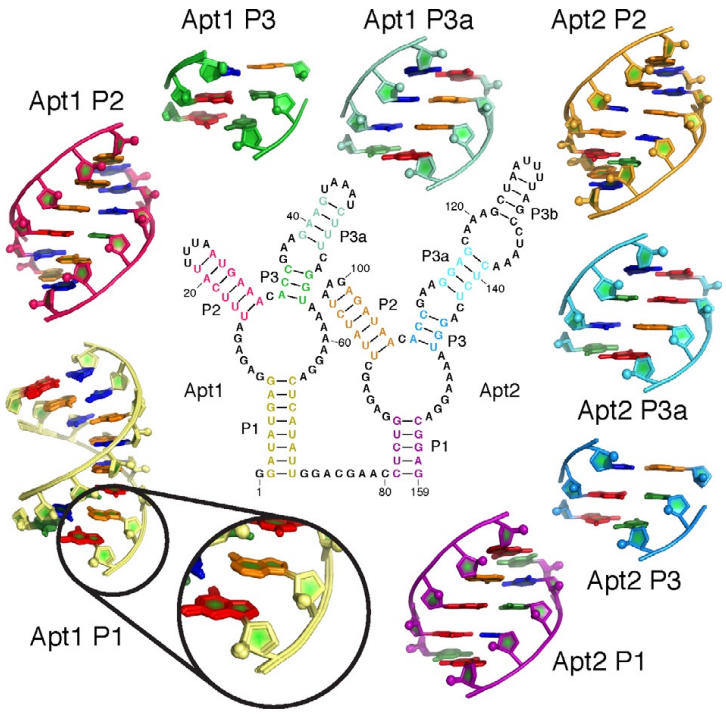

Figure 3 Preassembled helices for *F. nucleatum* double glycine riboswitch ligand-binding domain. The secondary structure is shown at center with the residues used for helix preassembly highlighted in color. Ensembles of 10 models of each helix generated by the helix preassembly protocol in Rosetta are shown at the periphery, labeled with the aptamer and helix number (e.g., Apt1 P1 for the P1 helix of aptamer 1). The magnified view of the Apt1 P1 helix highlights the slight differences in conformation between the preassembled helix models.

because the fragments are drawn from RNAs of known structure. This low-resolution modeling step does not include any refinement at the atomic level, because the all-atom energy landscape is too "rugged"; that is, it contains many energy minima that can trap the nascent model from exploring alternative conformations, and strategies for searching this landscape (Sripakdeevong et al., 2011) are currently too computationally expensive for RNA domains above 10–20 nucleotides.

For the following steps, if a comparison to a crystallographic or other reference model is desired, inputting the reference during the modeling runs will allow root mean square deviation (RMSD) values to be reported in the output silent files. To properly calculate RMSDs, reference models must have the same sequence as the construct being modeled. The `make_rna_rosetta_ready.py` command reformats PDB files with the correct sequence to be used as reference models (Appendix, command line [4]). For the glycine riboswitch example described in this chapter, the crystallographic structure includes a protein-binding loop that is not present in the construct used for experiments and modeling. To prepare the crystallographic structure for use as a reference model, we replace the protein-binding loop with a UUUA tetraloop to match the target sequence (Appendix, command lines [5] through [14]). These commands can also be used for more extensive remodeling of models and are described in detail in Section 3.6. We note that including a reference model is not required for the modeling workflow but can allow for easy visualization of modeling results through energy versus RMSD plots, such as those shown in Fig. 4.

As with the helix assembly runs above, a series of text files will record the command lines used for setup and modeling. To set up a FARNA run, we create a file called `README_SETUP`, which calls a script called `rna_denovo_setup.py` to generate the command line for low-resolution modeling. Command line [15] in the Appendix shows an example `README_SETUP` file. Special tags can be used to specify advanced options for the modeling run, including specific noncanonical base pairs (Appendix), segments of the RNA that are thought to form a tertiary contact, or soft constraints from MOHCA-seq experiments. For example, to incorporate the MOHCA-seq data into computational modeling in Rosetta, a smooth pseudoenergy potential is applied between pairs of nucleotides showing strong MOHCA-seq signal, which indicates that they are proximal in the 3D fold. Two separate pseudoenergy potential functions are used, one for strong and one for weak MOHCA-seq hits (Fig. 2E); these potentials differ

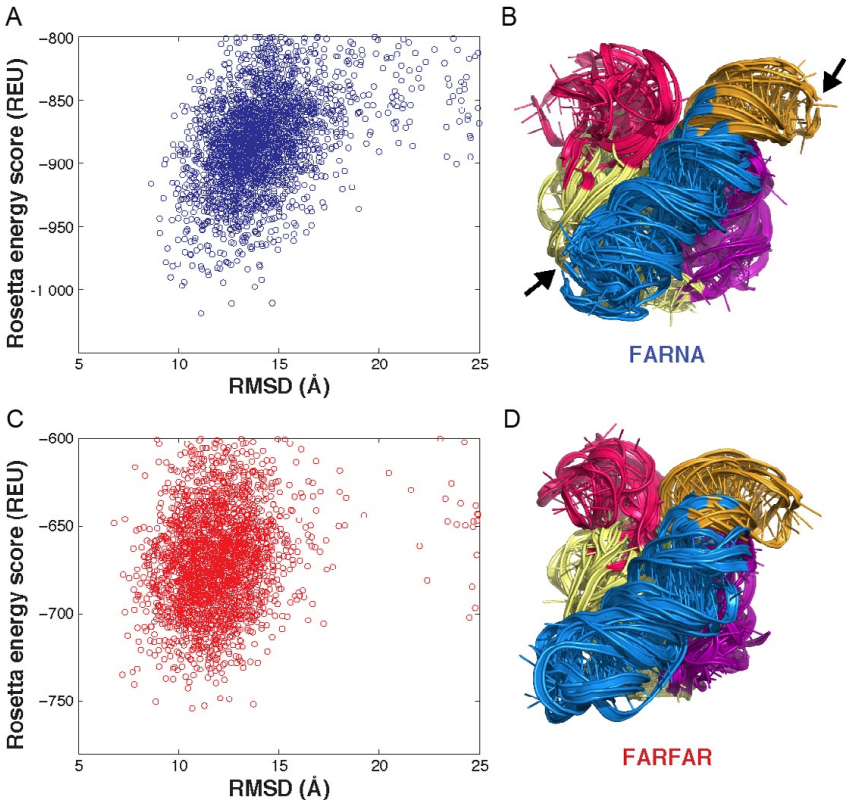

Figure 4 Low-resolution modeling and full-atom refinement using FARNA and FARFAR. (A) Rosetta energy score versus RMSD plot after low-resolution modeling using FARNA. (B) Overlaid 10 lowest-energy models after low-resolution modeling using FARNA. Chain breaks are visible in many models (arrows), and residues commonly adopt unrealistic geometries. (C) Rosetta energy score versus RMSD plot after minimization using the FARFAR algorithm. (D) Overlaid 10 lowest-energy models after minimization using the FARFAR algorithm. The models do not show any chain breaks, and poor residue geometries are greatly reduced.

only in the amplitude of the energy penalty applied for residues that are too close or too far apart. These potentials are specified in text-formatted files in Rosetta's "constraint file" format (example in Appendix, file [F3]) and can be input to `rna_denovo_setup.py`. The command `source README_SETUP` (Appendix, command line [16]) generates a file containing a command line for `rna_denovo` with the tags given in `README_SETUP`, called `README_FARFAR` (Appendix, command line [17]), as well as parameter and FASTA files.

It is a good idea to test the run locally before submitting it as a job to a cluster, in case the run is stopped by an error. To test the run, we use `source README_FARFAR` (Appendix, command line [18]) to begin a single job on a local computer and wait until sampling begins successfully (command line output similar to "`Picked Fragment Library for sequence u and sec. struct`

H ... found 2308 potential fragments") before canceling the run. Then perform modeling on a computer cluster by first using the rosetta_submit.py script to generate submission files (Appendix, command line [19]) and then using source on the submission file appropriate for the cluster's queuing system (e.g., Condor, LSF, PBS, etc.). For FARNA runs, it is best to generate around 10,000–15,000 low-resolution models, from which a subset will later be minimized. The models generated by rna_denovo are by default placed in a folder named out, which is created in the modeling folder. The out folder contains individual folders for each run with a silent .out file in each that describes all of the models from that run. To collect all of the models into a single silent file, we use the easy_cat.py script (Appendix, command line [20]). This creates a single concatenated .out file with the name tag initially provided in README_SETUP.

If a reference (native) model was input during FARNA modeling, the RMSDs of the FARNA models to the reference can be compared to their Rosetta energy scores, which are all recorded in the concatenated silent file, to assess the quality of the low-resolution models. An example energy versus RMSD plot is shown in Fig. 4A. Additionally, it may be helpful to visualize the low-resolution models with the lowest—that is, most favorable—Rosetta energy scores. To do this, we extract the lowest-scoring models from the concatenated .out file using extract_lowscore_models.py (Appendix, command line [21]). These PDB-formatted models can then be loaded in PyMOL for comparison (Fig. 4B). Note that the FARNA models may contain discontinuities in the RNA backbone, which are visible in PyMOL. These chainbreaks occur because crystallographic fragments that are sampled and built into the model first may prevent a continuous backbone from being built in other regions of the RNA. Chainbreaks are not a cause for concern, however, because the following all-atom minimization step typically resolves them.

3.4. Producing and selecting models with reasonable stereochemistry using refinement

As mentioned earlier, the low-resolution models generated by FARNA may contain chainbreaks and unrealistic atomic-level geometries due to the method of sampling rigid fragments of crystallographic RNA structures. To achieve more realistic models of the RNAs, we use the rna_minimize program in Rosetta to refine the lowest-energy 1/6 of the low-resolution models (e.g., if 12,000 FARNA models were generated, minimize 2000 of them). This FARNA with Full-Atom Refinement (FARFAR) strategy

optimizes the low-resolution models based on the Rosetta full-atom energy function, which accounts for physical and chemical features such as van der Waals forces, hydrogen bonding, desolvation penalties for polar groups, and RNA backbone torsion angles (Das et al., 2010; Sripakdeevong et al., 2011).

To set up refinement of the FARNA models, we create a `MINIMIZE` file similar to command line [22] in the Appendix. Running `source MINIMIZE` (Appendix, command line [23]) calls the `parallel_min_setup.py` script to generate the command lines for refinement in an output script specified in MINIMIZE (by default, `min_cmdline`). Each line in `min_cmdline` is one minimization command, and the number of lines in `min_cmdline` is the number of processors specified in MINIMIZE. As for FARNA runs, it is best to test the minimization before submitting the jobs to the cluster; here, we copy the first line from the `min_cmdline` file starting with `rna_minimize` and run it locally (Appendix, command line [24]), waiting for the output "`protocols.rna.RNA_Minimizer: Minimizing...round= 1`" before canceling the run. After confirming that the run proceeds without errors, we create submission files by running `rosetta_submit.py` on `min_cmdline` (Appendix, command line [25]), then using `source` to submit the jobs. For refinement runs, the jobs will automatically terminate after all of the specified models are minimized, which usually takes a few hours with 100 processors on a cluster. The silent files for minimized models outputted by `rna_minimize` are collected in individual folders in a folder called `min_out`, similar to the output of `rna_denovo`. Again, we use `easy_cat.py` to collect all of the minimized models into a single silent file with the tag given in `MINIMIZE` (Appendix, command line [26]).

Refinement using FARFAR improves low-resolution models by relaxing them into more realistic conformations. This generally results in better RMSDs to input reference models, as seen by energy versus RMSD plots (Fig. 4C), and more realistic models, which can be visualized using PyMOL in the same way as earlier (Fig. 4D). More base pairs are correctly formed, chainbreaks that were present in FARNA models are typically fixed, and constraints from chemical mapping and MOHCA-seq tend to be better satisfied in minimized models.

3.5. Clustering to generate final set of models

The set of refined FARFAR models often contains subsets of models that adopt similar folds to within helical resolution, especially if modeling was performed in the context of chemical mapping and MOHCA-seq data.

To select a representative set of 3D models that is likely to reflect the native fold of the RNA, we collect the largest and lowest-energy subsets of models that fall within a certain RMSD threshold of each other as described later. Such clustering suggests that the fold adopted by those models is both energetically favorable and comparatively likely to be sampled (Shortle, Simons, & Baker, 1998), and the RMSD threshold value (see later) provides an estimate of modeling precision.

First, we use the script `silent_file_sort_and_select.py` to sort the models in the silent file output by FARFAR and select the desired number of lowest-energy models, normally equal to 0.5% of the total unrefined (FARNA) models (Appendix, command line [27]). This script generates a new silent file containing only the selected lowest-energy models, usually 50–75 if 10,000–15,000 models were built by FARNA. Then, we perform clustering locally using the `cluster` application in Rosetta, which uses an RMSD threshold input by the user to sort the models in the silent file into groups that fall within the threshold (Appendix, command line [28]). Each clustering run normally takes less than a minute. The output of running `cluster` is a silent file containing the clustered models, as well as a screen output that reports how many clusters were generated and how many models were sorted into each cluster. Our standard practice is to choose an RMSD threshold that results in 1/6 of the clustered models being sorted into the largest cluster, by adjusting the input RMSD threshold over multiple clustering runs. Finally, we isolate the models in the top cluster, which is referred to as cluster0 in the output of the `cluster` application (Appendix, command line [29]). This can be done using a text editor by copying the clustered silent file, selecting the lines of the silent file comprising the cluster0 models (labeled in the silent file with `c.0.*`, where `*` is the number of the model in the cluster), and deleting the remainder. Then, we use `extract_lowscore_decoys.py` to collect these final models as PDB-formatted files (Appendix, command line [30]).

The RMSD threshold used in clustering represents an estimate of the "precision" of the final subset of FARFAR models. Because the precision captures the variation between the models, it also sets a lower bound on the accuracy of the modeling, although individual models within the cluster may have RMSDs to crystallographic models that are lower. When both chemical mapping and MOHCA-seq data are included in our pipeline, we find that the top cluster typically reflects the native fold of the target RNA, as compared to a previously or subsequently released crystal structure, to 7–15 Å RMSD (Cheng et al., 2014) (Fig. 5).

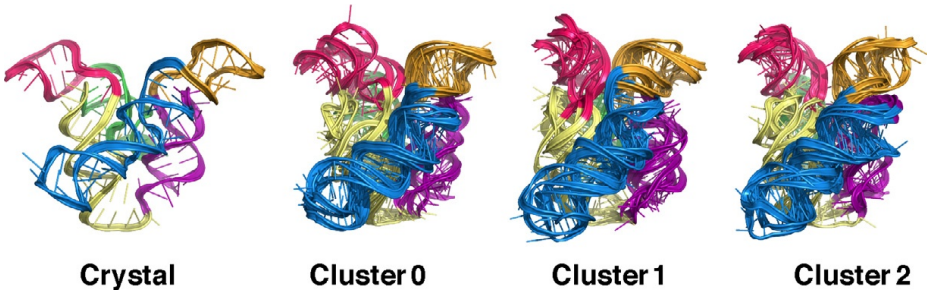

Figure 5 Clustering of minimized models to select representative models. Comparison of models generated by the experimental/computational pipeline. The crystal structure (PDB ID 3P49) is shown at left. At right, four representative models are overlaid for each of the top three model clusters. The cluster center model of cluster0 has a 7.9 Å RMSD to the crystal structure.

3.6. Advanced strategies: Building subpieces into existing models

In some cases, it may be beneficial to improve predictions of RNA structures by remodeling sections of the structure or adding additional regions to the structure. As an example, the tandem glycine-binding riboswitch, which binds two molecules of glycine using two sequentially arranged glycine aptamers, is thought to act as a cooperative sensor of glycine. However, recent studies showed that inclusion of a leader sequence abolishes cooperativity of the riboswitch, at least for the isolated ligand-binding domain. Sequence–structure alignment indicated that it likely forms a kink-turn motif (Kladwang et al., 2012; Rahrig, Petrov, Leontis, & Zirbel, 2013; Sarver, Zirbel, Stombaugh, Mokdad, & Leontis, 2008). The leader sequence was not included in prior models or crystallographic structures of the RNA (Butler et al., 2011), but modeling in Rosetta was able to automatically model the structure formed by the leader sequence when incorporated into the crystal structure (Kladwang et al., 2012) and gave support for a kink-turn conformation. Here, we will discuss how to perform this type of addition and remodeling in Rosetta.

In order to remodel a region of an RNA for which a piece is already available, e.g., in a crystallographic template, it may first be necessary to excise the desired piece from the template. This excision can be accomplished using the `pdbslice.py` command, which creates a new PDB file that contains a user-specified subset of the residues in the input PDB file (Appendix, command line [31]). In the example of the glycine riboswitch,

the first nucleotide must be excised, as well as the residues comprising the linker between the two aptamers of the ligand-binding domain, which base pair with the leader sequence. The sliced model will be used as an input to FARFAR modeling so that only the nucleotides that are not present in the model will be sampled. Here, because a 5′-leader sequence must also be added to the RNA, we must also revise the FASTA and secondary structure files and renumber the input PDB, so that the sequence numbers and identities are fully consistent. The revised FASTA and secondary structure files are given as files [F4] and [F5] in the Appendix. To renumber the input PDB, we use the `renumber_pdb_in_place.py` script, providing it with the PDB to be renumbered and the desired final sequence position ranges (Appendix, command line [32]). Then, we create a new `README_SETUP` file that reads the revised FASTA and secondary structure files and includes a flag to input the sliced and renumbered input model (Appendix, command line [33]). Finally, we run the modeling as before. If only a small region of the RNA is being remodeled, fewer processors or less computational time may be necessary to reach convergence, so adjust these parameters accordingly.

In cases where sequence analysis or other prediction algorithms suggest the presence of an RNA motif or fold of known structure, one strategy to save computational time is to use an instance of the known structure as a template for modeling the sequence of interest—this method is called "threading." Threaded fragments of structures, such as kink-turn motifs or loops, can in turn be used as input PDBs for global modeling or remodeling of RNAs and can help to focus sampling on regions of entirely unknown structure. See command line [34] in the Appendix; further documentation is also available at RosettaCommons (https://www.rosettacommons.org/docs/latest/rna-thread.html).

4. EVALUATION

The pipeline we have described in the preceding text achieves *de novo* models of RNAs with subhelical (~10 Å) resolution, based on benchmark and blind prediction studies. Independent validation or falsification of models at this resolution can be challenging, because available chemical mapping and MOHCA-seq constraints are usually included in the modeling. We recommend two strategies to test the final models. First, check

whether tertiary features of the RNA can be reconstituted without some of the available constraints; e.g., if mutate-and-map experiments identify tertiary contacts in the RNA, then exclude MOHCA-seq proximity constraints from the modeling and check for agreement of the final models with MOHCA-seq data. Recovery of proximities indicated by MOHCA-seq independent of modeling with those constraints lends support to those tertiary features. Second, one can perform mutational analysis to verify new tertiary contacts suggested by the modeling by using chemical mapping or MOHCA-seq experiments to assess the effects of mutations predicted to disrupt those new contacts or mutations that may rescue the structure through formation of compensatory base pairs (Tian et al., 2014; Xue et al., 2014).

5. CONCLUSION

Three-dimensional modeling of RNAs has improved greatly in recent years, aided by advances in both experimental methods and computational strategies for predicting secondary and tertiary structures. In this chapter, we have described a general workflow for modeling RNA 3D folds using the Rosetta framework for macromolecular modeling, guided by data from solution-state chemical mapping experiments. These experiments, particularly the two-dimensional M^2 and MOHCA-seq measurements, provide constraints for modeling by defining an RNA's secondary structure elements and identifying tertiary proximities within its fold. This experimental/ computational pipeline has allowed us to recover the tertiary folds of RNA-Puzzles challenges and continues to reveal avenues for exploring biological questions through *in vitro* and *in vivo* experiments.

The ultimate goal of prediction and design of RNA structures at consistent atomic accuracy has not yet been achieved, but continuing developments in computational and hybrid methods hold promise for making strides toward this goal. In particular, interfacing current methods for recovering RNA folds at medium resolution with new strategies for modeling small RNA motifs at near-atomic-accuracy; incorporating insights about local RNA tertiary conformations from NMR constraints or chemical mapping reagents into global modeling; and improving methods for classifying, sampling, and constructing RNA motifs are likely to have strong impacts in RNA structure modeling in the coming years.

ACKNOWLEDGMENTS

The writing of this chapter was supported by National Institutes of Health (5T32 GM007276 to C. Y. C.; R01 GM102519 to R. D.), the Burroughs-Wellcome Foundation (CASI to R. D.), and Stanford Bio-X and HHMI international fellowships (F. C. C.). Computational resources were provided by the Stanford BioX^3 computing cluster. We thank Caleb Geniesse and RosettaCommons for testing and helpful comments.

APPENDIX. EXAMPLE COMMAND LINES AND FILES FOR RNA MODELING IN ROSETTA

Command lines, input files, and example output files can be found in the Rosetta/demos/public/mohca_seq folder, which is included in the released Rosetta software package.

Documentation for setting up Rosetta and RNA tools:
https://www.rosettacommons.org/docs/latest/Build-Documentation.html
https://www.rosettacommons.org/docs/latest/RNA-tools.html

[F1] Example FASTA file:

```
>3P49_RNA.pdb
ggauaugaggagagauuucauuuuaaugaaacaccgaagaaguaaaucuuucagguaa
aaaggacucauauuggacgaaccucuggagagcuuaucuaagagauaacaccgaagga
gcaaagcuaauuuuagccuaaacucucagguaaaaggacggag
```

The RNA sequence must be lowercase.

[F2] Example secondary structure file:

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

[1] Generate command lines for helix preassembly:

```
helix_preassemble_setup.py -secstruct [secondary structure file] -fasta [FASTA file]
```

[2] Example command line for helix preassembly:

```
rna_denovo -nstruct 100 -params_file helix0.params -fasta helix0.fasta -out:file:silent helix0.out -include_neighbor_base_stacks -minimize_rna true -rna::corrected_geo -score:rna_torsion_potential RNA11_based_new -chemical::enlarge_H_lj -score:weights stepwise/rna/rna_helix -cycles 1000 -output_res_num 2-9 65-72
```

[3] Run command lines for helix preassembly (local):

```
source CMDLINES
```

[4] Prepare native/reference structure for Rosetta, if available:

```
make_rna_rosetta_ready.py 3P49.pdb
```

Outputs reformatted native model as "3p49_RNA.pdb," to be input to README_SETUP. In the glycine riboswitch example presented here, the 3P49 crystal structure includes a protein-binding loop that is not part of the construct used for *de novo* modeling. Command lines [5] through [14] show how to replace the extraneous residues with a tetraloop matching the experimentally probed construct using a short FARFAR modeling run.

[5] Cut out a segment of a model:

```
pdbslice.py[3p49_RNA.pdb]-subset[1-2136-169][slice_]
```

The first input is the model from which you want to excise regions of interest. The second input is the range of nucleotides that you want to keep in your model. The third input is the prefix that will be added to the beginning of the input model's filename. Here, the protein-binding loop is excised by specifying the range of residues given in the command line.

[6] Renumber a PDB:

```
renumber_pdb_in_place.py [slice_3p49_RNA.pdb] [1-21
26-159]
```

The first input is the PDB file to be renumbered and the second input is the desired final ranges of sequence positions. Gaps may be intentionally left in the final sequence range to allow for remodeling in the middle of the RNA. Here, a UUUA tetraloop will be built in place of the excised protein-binding loop.

[7] Example README_SETUP for *de novo* remodeling with a sliced input PDB:

```
rna_denovo_setup.py -fasta fasta -secstruct_file sec-
struct \
-tag native \
-working_res 1-159 \
-s slice_3p49_RNA.pdb \
-cycles 20000 \
-ignore_zero_occupancy false \
```

Options:

-fasta [fasta]	Input FASTA file
-secstruct_file [secstruct]	Input secondary structure file
-tag	Name for output files
-working_res	Specify range of residues to model
-s slice_3p49_RNA.pdb	See below
-ignore_zero_occupancy false	

The "-s" flag allows users to input a list of PDB files to use in the modeling; the residues that are part of the input PDB files will not be moved relative to each other, though if multiple PDB files are input, the orientations of the residues in the separate files may change. In this example, the full-atom refinement algorithm will be applied in the same run as fragment assembly.

[8] Generate command line for FARFAR modeling:

```
source README_SETUP
```

[9] Example README_FARFAR:

```
rna_denovo -nstruct 500 -params_file native.params -fasta
native.fasta -out:file:silent native.out -include_neighbor_
base_stacks -minimize_rna true -s slice_3p49_RNA.pdb -input_res
1-21 26-159 -cycles 20000 -ignore_zero_occupancy false -output_
res_num 1-159
```

[10] Test command line for FARFAR modeling:

```
source README_FARFAR
```

This command runs a single local job on your computer. Wait until sampling begins successfully (command line output similar to "`Picked Fragment Library for sequence u and sec. struct H ... found 2308 potential fragments`"), then cancel the run and submit the job to the cluster.

[11] Submit jobs to the cluster:

```
rosetta_submit.py README_FARFAR out [16] [1]
```

The first number states how many processors to use for the run, while the second number states the maximum time each

job will be allowed to run (walltime, in hours). Note that certain supercomputers only allow requests specific multiples of processors (e.g., the Stampede cluster requires a multiple of 16). Start the run with the appropriate command listed by the output above (e.g., source qsubMPI for the Stampede cluster).

[12] Concatenate all models from the `out` folder:

```
easy_cat.py out
```

Also outputs the number of models in the final silent file to the screen.

[13] Extract lowest-energy models to .pdb files for viewing in PyMOL:

```
extract_lowscore_decoys.py native.out [1]
```

Input the number of lowest-scoring models to extract from the silent file. Here, extract the single lowest-scoring model to use as the native model input for comparison to the *de novo* models.

[14] Rename lowest-score model for use as reference model:

```
mv native.out.1.pdb 3p49_native_RNA.pdb
```

[F3] Example pseudoenergy constraint file:

```
[atompairs]
O2′ 2 C4′ 38 FADE 0 30 15 −4.00 4.00
O2′ 2 C4′ 38 FADE −99 60 30 −36.00 36.00
O2′ 1 C4′ 44 FADE 0 30 15 −4.00 4.00
O2′ 1 C4′ 44 FADE −99 60 30 −36.00 36.00
O2′ 5 C4′ 60 FADE 0 30 15 −4.00 4.00
O2′ 5 C4′ 60 FADE −99 60 30 −36.00 36.00
O2′ 2 C4′ 64 FADE 0 30 15 −4.00 4.00
O2′ 2 C4′ 64 FADE −99 60 30 −36.00 36.00
O2′ 25 C4′ 54 FADE 0 30 15 −4.00 4.00
O2′ 25 C4′ 54 FADE −99 60 30 −36.00 36.00
O2′ 45 C4′ 64 FADE 0 30 15 −4.00 4.00
O2′ 45 C4′ 64 FADE −99 60 30 −36.00 36.00
O2′ 45 C4′ 75 FADE 0 30 15 −4.00 4.00
O2′ 45 C4′ 75 FADE −99 60 30 −36.00 36.00
O2′ 32 C4′ 88 FADE 0 30 15 −4.00 4.00
O2′ 32 C4′ 88 FADE -99 60 30 −36.00 36.00
O2′ 42 C4′ 84 FADE 0 30 15 −4.00 4.00
O2′ 42 C4′ 84 FADE −99 60 30 −36.00 36.00
```

```
O2' 48 C4' 84 FADE 0 30 15 -4.00 4.00
O2' 48 C4' 84 FADE -99 60 30 -36.00 36.00
O2' 55 C4' 88 FADE 0 30 15 -4.00 4.00
O2' 55 C4' 88 FADE -99 60 30 -36.00 36.00
O2' 55 C4' 108 FADE 0 30 15 -4.00 4.00
O2' 55 C4' 108 FADE -99 60 30 -36.00 36.00
O2' 58 C4' 118 FADE 0 30 15 -4.00 4.00
O2' 58 C4' 118 FADE -99 60 30 -36.00 36.00
O2' 67 C4' 119 FADE 0 30 15 -4.00 4.00
O2' 67 C4' 119 FADE -99 60 30 -36.00 36.00
O2' 67 C4' 121 FADE 0 30 15 -4.00 4.00
O2' 67 C4' 121 FADE -99 60 30 -36.00 36.00
O2' 78 C4' 113 FADE 0 30 15 -4.00 4.00
O2' 78 C4' 113 FADE -99 60 30 -36.00 36.00
O2' 78 C4' 135 FADE 0 30 15 -4.00 4.00
O2' 78 C4' 135 FADE -99 60 30 -36.00 36.00
O2' 42 C4' 157 FADE 0 30 15 -4.00 4.00
O2' 42 C4' 157 FADE -99 60 30 -36.00 36.00
O2' 74 C4' 156 FADE 0 30 15 -4.00 4.00
O2' 74 C4' 156 FADE -99 60 30 -36.00 36.00
O2' 100 C4' 148 FADE 0 30 15 -4.00 4.00
O2' 100 C4' 148 FADE -99 60 30 -36.00 36.00
O2' 100 C4' 145 FADE 0 30 15 -4.00 4.00
O2' 100 C4' 145 FADE -99 60 30 -36.00 36.00
O2' 113 C4' 153 FADE 0 30 15 -4.00 4.00
O2' 113 C4' 153 FADE -99 60 30 -36.00 36.00
O2' 135 C4' 154 FADE 0 30 15 -4.00 4.00
O2' 135 C4' 154 FADE -99 60 30 -36.00 36.00
O2' 5 C4' 119 FADE 0 30 15 -4.00 4.00
O2' 5 C4' 119 FADE -99 60 30 -36.00 36.00
O2' 25 C4' 88 FADE 0 30 15 -0.80 0.80
O2' 25 C4' 88 FADE -99 60 30 -7.20 7.20
O2' 37 C4' 62 FADE 0 30 15 -0.80 0.80
O2' 37 C4' 62 FADE -99 60 30 -7.20 7.20
O2' 79 C4' 103 FADE 0 30 15 -0.80 0.80
O2' 79 C4' 103 FADE -99 60 30 -7.20 7.20
O2' 15 C4' 88 FADE 0 30 15 -0.80 0.80
O2' 15 C4' 88 FADE -99 60 30 -7.20 7.20
O2' 32 C4' 108 FADE 0 30 15 -0.80 0.80
```

```
O2′ 32 C4′ 108 FADE −99 60 30 −7.20 7.20
O2′ 9 C4′ 138 FADE 0 30 15 −0.80 0.80
O2′ 9 C4′ 138 FADE −99 60 30 −7.20 7.20
O2′ 25 C4′ 118 FADE 0 30 15 −0.80 0.80
O2′ 25 C4′ 118 FADE −99 60 30 −7.20 7.20
```

[15] Example README_SETUP:

```
   rna_denovo_setup.py -fasta fasta -secstruct_file sec-
struct \
   -fixed_stems \
   -no_minimize \
   -tag glycine_riboswitch \
   -working_res 1-159 \
   -native 3p49_native_RNA.pdb \
   -cst_file constraints \
   -staged_constraints \
   -cycles 20000 \
   -ignore_zero_occupancy false \
   -silent helix0.out helix1.out helix2.out helix3.out
helix4.out helix5.out helix6.out helix7.out \
   -input_silent_res 2-9 65-72 16-21 26-31 33-35 54-56
39-42 48-51 81-85 155-159 92-97 101-106 108-110 145-147
114-117 139-142 \
```

Options:

-fasta [fasta]	Input FASTA file
-secstruct_file [secstruct]	Input secondary structure file
-fixed_stems	Specify whether helices should be fixed
-no_minimize	Specify not to perform full-atom refinement; minimization will be performed in the next stage of modeling
-tag	Name for output files
-working_res	Specify range of residues to model
-native [native.pdb]	Input reference or native model; used for benchmarking cases and will return rms calculations for all models (see command line [5])
-cst_file [constraints]	Input file with pseudoenergy constraints
-staged_constraints	Apply constraints

-ignore_zero_occupancy false	
-silent [helix0.out helix1.out . . .]	Input silent files with preassembled helices
-input_silent_res [2–9 65–72 16–21 26–31 . . .]	Specify position ranges of helices in silent files

[16] Generate command line for FARFAR modeling:

```
source README_SETUP
```

[17] Example README_FARFAR:

```
rna_denovo -nstruct 500 -params_file glycine_riboswitch.
params -fasta glycine_riboswitch.fasta -out:file:silent
glycine_riboswitch.out -include_neighbor_base_stacks
-minimize_rna false -native glycine_riboswitch_3p49_
native_RNA.pdb -in:file:silent helix0.out helix1.out
helix2.out helix3.out helix4.out helix5.out helix6.out
helix7.out -input_res 2-9 65-72 16-21 26-31 33-35 54-56
39-42 48-51 81-85 155-159 92-97 101-106 108-110 145-147
114-117 139-142 -cst_file glycine_riboswitch_
constraints-staged_constraints -cycles 20000 -ignore_
zero_occupancy false -output_res_num 1-159
```

[18] Test command line for FARFAR modeling:

```
source README_FARFAR
```

[19] Submit jobs to the cluster:

```
rosetta_submit.py README_FARFAR out [96] [16]
```

[20] Concatenate all models from the `out` folder:

```
easy_cat.py out
```

[21] Extract lowest-energy models to .pdb files for viewing in PyMOL:

```
extract_lowscore_decoys.py glycine_riboswitch.out
[15]
```

[22] Example MINIMIZE:

```
parallel_min_setup.py -silent glycine_riboswitch.out
-tag glycine_riboswitch_min -proc [96] -nstruct [2000] -
out_folder min_out -out_script min_cmdline "-native
glycine_riboswitch_3p49_native_RNA.pdb -cst_fa_file
```

```
glycine_riboswitch_constraints -params_file glycine_
riboswitch.params-ignore_zero_occupancy false -skip_
coord_constraints"
```

The first number states how many processors to use for the run, while the second number is 1/6 the total number of previously generated FARNA models. If you are running on a supercomputer that only allows specific multiples of processors, use an appropriate number for the first input.

[23] Generate command lines for full-atom refinement:

```
source MINIMIZE
```

[24] Example command line from min_cmdline to run as test:

```
rna_minimize -native glycine_riboswitch_3p49_native_RNA.pdb
-cst_fa_fileglycine_riboswitch_constraint-params_fileglycine_
riboswitch.params -ignore_zero_occupancy false -skip_coord_
constraints -in:file:silentmin_out/0/0.silent -out:file:silent
min_out/0/glycine_riboswitch_min.out
```

[25] Submit jobs to the cluster:

```
rosetta_submit.py min_cmdline min_out [1] [16]
```

The first number states how many processors to use for each line in min_cmdline. Here, enter 1 for the first input so that the total number of processors used will be equal to the number of processors entered with the "-proc" flag in command line [12], above. The second number states the maximum time each job will be allowed to run (walltime). Start the run with the appropriate command listed by the output above (e.g., source qsubMPI for the Stampede cluster).

[26] Concatenate all models from the `min_out` folder:

```
easy_cat.py min_out
```

[27] Sort models by Rosetta energy and select a subset for clustering:

```
silent_file_sort_and_select.py [glycine_riboswitch_
min.out]-select [1-60]-o [glycine_riboswitch_min_sort.
out]
```

The range of models under the `-select` tag includes 0.5% of the total number of FARNA models generated previously. Outputs a new silent file containing selected number of lowest-energy models.

[28] Cluster models:

```
cluster -in:file:silent glycine_riboswitch_min_sort.
out -in:file:fullatom -out:file:silent_struct_type
binary -export_only_low false -out:file:silent clus-
ter.out -cluster:radius [radius]
```

Select a radius so that 1/6 of the models in the input sorted silent file are in the largest cluster (cluster0) of models.

[29] Copy clustered .out file to a new file to isolate cluster0:

```
cp cluster.out cluster0.out
```

[30] Extract lowest-energy models to .pdb files for viewing in PyMOL:

```
extract_lowscore_decoys.py cluster0.out [15]-no_
replace_names
```

Input the number of models in cluster0. The -no_replace_ names tag preserves the filenames of the cluster members to reflect their order in the cluster, rather than renaming them in order of Rosetta energy score.

[31] Cut out a segment of a model:

```
pdbslice.py [3p49_native_RNA.pdb] -subset [2-72
81-159] [slice_kinkturn_]
```

Here, the 3P49 crystal structure includes an additional G at position 0, which must be excised to allow the leader sequence to be added to the 5′-end, and the internal linker that forms the kink-turn motif with the leader sequence is also excised to allow remodeling.

[32] Renumber a PDB:

```
renumber_pdb_in_place.py [slice_kinkturn_3P49_native_
RNA.pdb] [10-80 89-167]
```

Here, the PDB is renumbered to allow the leader sequence to be added at the 5′-end.

[F4] Example revised FASTA file:

```
>3P49_RNA_kinkturn.pdb
ucggaugaagauaugaggagagauuucauuuuaaugaaacaccgaagaaguaaaucuu
ucagguaaaaaggacucauauuggacgaaccucuggagagcuuaucuaagagauaaca
ccgaaggagcaaagcuaauuuuagccuaaacucucagguaaaaggacggag
```

[F5] Example revised secondary structure file:

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

[33] Example README_SETUP for *de novo* remodeling with a sliced input PDB:

```
rna_denovo_setup.py -fasta fasta2 -secstruct_file sec-
struct2 \
    -fixed_stems \
    -tag glycine_rbsw_kinkturn \
    -working_res 1-167 \
    -s slice_kinkturn_3P49_native_RNA.pdb \
    -cycles 20000 \
    -ignore_zero_occupancy false \
```

[34] Thread an RNA sequence into a template structure:

```
rna_thread -in:file:fasta [fasta] -in:file:s [template
PDB] -o [output PDB]
```

The first input is a FASTA file containing two RNA sequences: (1) the sequence of interest, onto which the structure of the template sequence will be threaded and (2) the template sequence. The template sequence should be truncated to the regions into which the sequence of interest will be threaded; use hyphens ("-") to align the template sequence with the target sequence in the FASTA file. The second input, the template structure in PDB format, should be similarly truncated, using `pdbslice.py` if necessary. If the template PDB is not correctly formatted for Rosetta modeling, use `make_rna_rosetta_ready.py` to reformat it. The last input is the name of the output PDB.

Further documentation for RNA threading in Rosetta can be found at the RosettaCommons (https://www.rosettacommons.org/docs/latest/rna-thread.html).

REFERENCES

Butler, E. B., Xiong, Y., Wang, J., & Strobel, S. A. (2011). Structural basis of cooperative ligand binding by the glycine riboswitch. *Chemistry & Biology*, *18*(3), 293–298. http://dx.doi.org/10.1016/j.chembiol.2011.01.013.

Cheng, C., Chou, F.-C., Kladwang, W., Tian, S., Cordero, P., & Das, R. (2014). MOHCA-seq: RNA 3D models from single multiplexed proximity-mapping experiments. *bioRxiv*. http://dx.doi.org/10.1101/004556.

Chou, F. C., Lipfert, J., & Das, R. (2014). Blind predictions of DNA and RNA tweezers experiments with force and torque. *PLoS Computational Biology*, *10*(8), e1003756. http://dx.doi.org/10.1371/journal.pcbi.1003756.

Chou, F. C., Sripakdeevong, P., Dibrov, S. M., Hermann, T., & Das, R. (2013). Correcting pervasive errors in RNA crystallography through enumerative structure prediction. *Nature Methods*, *10*(1), 74–76. http://dx.doi.org/10.1038/nmeth.2262.

Cordero, P., Kladwang, W., VanLang, C. C., & Das, R. (2012). Quantitative dimethyl sulfate mapping for automated RNA secondary structure inference. *Biochemistry*, *51*(36), 7037–7039. http://dx.doi.org/10.1021/bi3008802.

Cordero, P., Kladwang, W., VanLang, C. C., & Das, R. (2014). The mutate-and-map protocol for inferring base pairs in structured RNA. *Methods in Molecular Biology*, *1086*, 53–77. http://dx.doi.org/10.1007/978-1-62703-667-2_4.

Cordero, P., Lucks, J. B., & Das, R. (2012). An RNA mapping database for curating RNA structure mapping experiments. *Bioinformatics*, *28*(22), 3006–3008. http://dx.doi.org/10.1093/bioinformatics/bts554.

Cruz, J. A., Blanchet, M. F., Boniecki, M., Bujnicki, J. M., Chen, S. J., Cao, S., et al. (2012). RNA-Puzzles: A CASP-like evaluation of RNA three-dimensional structure prediction. *RNA*, *18*(4), 610–625. http://dx.doi.org/10.1261/rna.031054.111.

Das, R., & Baker, D. (2007). Automated de novo prediction of native-like RNA tertiary structures. *Proceedings of the National Academy of Sciences of the United States of America*, *104*(37), 14664–14669. http://dx.doi.org/10.1073/pnas.0703836104.

Das, R., Karanicolas, J., & Baker, D. (2010). Atomic accuracy in predicting and designing noncanonical RNA structure. *Nature Methods*, 7(4), 291–294. http://dx.doi.org/10.1038/nmeth.1433.

Das, R., Kudaravalli, M., Jonikas, M., Laederach, A., Fong, R., Schwans, J. P., et al. (2008). Structural inference of native and partially folded RNA by high-throughput contact mapping. *Proceedings of the National Academy of Sciences of the United States of America*, *105*(11), 4144–4149. http://dx.doi.org/10.1073/pnas.0709032105.

Ditzler, M. A., Otyepka, M., Sponer, J., & Walter, N. G. (2010). Molecular dynamics and quantum mechanics of RNA: Conformational and chemical change we can believe in. *Accounts of Chemical Research*, *43*(1), 40–47. http://dx.doi.org/10.1021/ar900093g.

Erion, T. V., & Strobel, S. A. (2011). Identification of a tertiary interaction important for cooperative ligand binding by the glycine riboswitch. *RNA*, *17*(1), 74–84. http://dx.doi.org/10.1261/rna.2271511.

Hajdin, C. E., Bellaousov, S., Huggins, W., Leonard, C. W., Mathews, D. H., & Weeks, K. M. (2013). Accurate SHAPE-directed RNA secondary structure modeling, including pseudoknots. *Proceedings of the National Academy of Sciences of the United States of America*, *110*(14), 5498–5503. http://dx.doi.org/10.1073/pnas.1219988110.

Kim, H., Cordero, P., Das, R., & Yoon, S. (2013). HiTRACE-Web: An online tool for robust analysis of high-throughput capillary electrophoresis. *Nucleic Acids Research*, *41*(Web Server issue), W492–W498. http://dx.doi.org/10.1093/nar/gkt501.

Kladwang, W., Chou, F. C., & Das, R. (2012). Automated RNA structure prediction uncovers a kink-turn linker in double glycine riboswitches. *Journal of the American Chemical Society*, *134*(3), 1404–1407. http://dx.doi.org/10.1021/ja2093508.

Kladwang, W., Mann, T. H., Becka, A., Tian, S., Kim, H., Yoon, S., et al. (2014). Standardization of RNA chemical mapping experiments. *Biochemistry*, *53*(19), 3063–3065. http://dx.doi.org/10.1021/bi5003426.

Kladwang, W., VanLang, C. C., Cordero, P., & Das, R. (2011). A two-dimensional mutate-and-map strategy for non-coding RNA structure. *Nature Chemistry*, *3*(12), 954–962. http://dx.doi.org/10.1038/nchem.1176.

Leaver-Fay, A., Tyka, M., Lewis, S. M., Lange, O. F., Thompson, J., Jacak, R., et al. (2011). ROSETTA3: An object-oriented software suite for the simulation and design of

macromolecules. *Methods in Enzymology*, *487*, 545–574. http://dx.doi.org/10.1016/B978-0-12-381270-4.00019-6.

Lee, J., Kladwang, W., Lee, M., Cantu, D., Azizyan, M., Kim, H., et al. (2014). RNA design rules from a massive open laboratory. *Proceedings of the National Academy of Sciences of the United States of America*, *111*(6), 2122–2127. http://dx.doi.org/10.1073/pnas.1313039111.

Lyskov, S., Chou, F. C., Conchuir, S. O., Der, B. S., Drew, K., Kuroda, D., et al. (2013). Serverification of molecular modeling applications: The Rosetta online server that includes everyone (ROSIE). *PLoS One*, *8*(5), e63906. http://dx.doi.org/10.1371/journal.pone.0063906.

Petrov, A. I., Zirbel, C. L., & Leontis, N. B. (2013). Automated classification of RNA 3D motifs and the RNA 3D Motif Atlas. *RNA*, *19*(10), 1327–1340. http://dx.doi.org/10.1261/rna.039438.113.

Rahrig, R. R., Petrov, A. I., Leontis, N. B., & Zirbel, C. L. (2013). R3D align web server for global nucleotide to nucleotide alignments of RNA 3D structures. *Nucleic Acids Research*, *41*(Web Server issue), W15–W21. http://dx.doi.org/10.1093/nar/gkt417.

Reuter, J. S., & Mathews, D. H. (2010). RNAstructure: Software for RNA secondary structure prediction and analysis. *BMC Bioinformatics*, *11*, 129. http://dx.doi.org/10.1186/1471-2105-11-129.

Sarver, M., Zirbel, C. L., Stombaugh, J., Mokdad, A., & Leontis, N. B. (2008). FR3D: Finding local and composite recurrent structural motifs in RNA 3D structures. *Journal of Mathematical Biology*, *56*(1–2), 215–252. http://dx.doi.org/10.1007/s00285-007-0110-x.

Schrodinger, LLC (2010). *The PyMOL Molecular Graphics System, version 1.3r1.*

Seetin, M. G., Kladwang, W., Bida, J. P., & Das, R. (2014). Massively parallel RNA chemical mapping with a reduced bias MAP-seq protocol. *Methods in Molecular Biology*, *1086*, 95–117. http://dx.doi.org/10.1007/978-1-62703-667-2_6.

Shortle, D., Simons, K. T., & Baker, D. (1998). Clustering of low-energy conformations near the native structures of small proteins. *Proceedings of the National Academy of Sciences of the United States of America*, *95*(19), 11158–11162.

Sripakdeevong, P., Beauchamp, K., & Das, R. (2012). Why can't we predict RNA structure at atomic resolution? In N. B. Leontis & E. Westhof (Eds.), *RNA 3D structure analysis and prediction*.Heidelberg, New York: Springer, 400 p.

Sripakdeevong, P., Cevec, M., Chang, A. T., Erat, M. C., Ziegeler, M., Zhao, Q., et al. (2014). Structure determination of noncanonical RNA motifs guided by (1)H NMR chemical shifts. *Nature Methods*, *11*(4), 413–416. http://dx.doi.org/10.1038/nmeth.2876.

Sripakdeevong, P., Kladwang, W., & Das, R. (2011). An enumerative stepwise ansatz enables atomic-accuracy RNA loop modeling. *Proceedings of the National Academy of Sciences of the United States of America*, *108*(51), 20573–20578. http://dx.doi.org/10.1073/pnas.1106516108.

Tian, S., Cordero, P., Kladwang, W., & Das, R. (2014). High-throughput mutate-map-rescue evaluates SHAPE-directed RNA structure and uncovers excited states. *RNA*, *20*(11), 1815–1826. http://dx.doi.org/10.1261/rna.044321.114.

Tinoco, I., Jr., Borer, P. N., Dengler, B., Levin, M. D., Uhlenbeck, O. C., Crothers, D. M., et al. (1973). Improved estimation of secondary structure in ribonucleic acids. *Nature: New Biology*, *246*(150), 40–41.

Xue, S., Tian, S., Fujii, K., Kladwang, W., Das, R., & Barna, M. (2014). RNA regulons in Hox 5' UTRs confer ribosome specificity to gene regulation. *Nature*. http://dx.doi.org/10.1038/nature14010.

Yoon, S., Kim, J., Hum, J., Kim, H., Park, S., Kladwang, W., et al. (2011). HiTRACE: High-throughput robust analysis for capillary electrophoresis. *Bioinformatics*, *27*(13), 1798–1805. http://dx.doi.org/10.1093/bioinformatics/btr277.