SOFTWARE TOOL ARTICLE

REVISED 3D based on 2D: Calculating helix angles and stacking patterns using *forgi 2.0*, an RNA Python library centered on secondary structure elements. [version 2; peer review: 2 approved]

Bernhard C. Thiel [1], Irene K. Beckmann[1], Peter Kerpedjiev[2], Ivo L. Hofacker[1,3]

[1]Department of Theoretical Chemistry, Faculty of Chemistry, University of Vienna, Vienna, 1090, Austria
[2]Department of Biomedical Informatics, Harvard Medical School, Boston, Massachusetts, 02115, USA
[3]Research Group Bioinformatics and Computational Biology, Faculty of Computer Science, University of Vienna, Vienna, 1090, Austria

v2 **First published:** 14 Mar 2019, **8**:287 (https://doi.org/10.12688/f1000research.18458.1)
Latest published: 23 Apr 2019, **8**:287 (https://doi.org/10.12688/f1000research.18458.2)

Abstract
We present *forgi*, a Python library to analyze the tertiary structure of RNA secondary structure elements. Our representation of an RNA molecule is centered on secondary structure elements (stems, bulges and loops). By fitting a cylinder to the helix axis, these elements are carried over into a coarse-grained 3D structure representation. Integration with Biopython allows for handling of all-atom 3D information. *forgi* can deal with a variety of file formats including dotbracket strings, PDB and MMCIF files. We can handle modified residues, missing residues, cofold and multifold structures as well as nucleotide numbers starting at arbitrary positions. We apply this library to the study of stacking helices in junctions and pseudoknots and investigate how far stacking helices in solved experimental structures can divert from coaxial geometries.

Keywords
RNA, Python, RNA tertiary structure, RNA secondary structure, coaxial stacking, pseudo knots

This article is included in the International Society for Computational Biology Community Journal gateway.

This article is included in the Python Collection collection.

Open Peer Review

Referee Status: ✓ ✓

	Invited Referees 1	2
REVISED version 2 published 23 Apr 2019	✓ report	
version 1 published 14 Mar 2019	? report	✓ report

1 **Marta Szachniuk**, Poznan University of Technology, Poland

2 **Jerome Waldispühl**, McGill University, Canada
Roman Sarrazin-Gendron, McGill University, Canada

Any reports and responses or comments on the article can be found at the end of the article.

Corresponding author: Ivo L. Hofacker (ivo@tbi.univie.ac.at)

Author roles: Thiel BC: Data Curation, Investigation, Methodology, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Beckmann IK**: Data Curation, Investigation, Methodology, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Kerpedjiev P**: Methodology, Software, Writing – Review & Editing; **Hofacker IL**: Conceptualization, Funding Acquisition, Methodology, Supervision, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: This work was partly funded by the Austrian science fund (FWF) projects F 43 “RNA regulation of the transcriptome” and I 2874 “Prediction of RNA-RNA interactions” and Doctoral College W 1207 “RNA Biology”.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

How to cite this article: Thiel BC, Beckmann IK, Kerpedjiev P and Hofacker IL. **3D based on 2D: Calculating helix angles and stacking patterns using *forgi 2.0*, an RNA Python library centered on secondary structure elements. [version 2; peer review: 2 approved]** F1000Research 2019, **8**:287 (https://doi.org/10.12688/f1000research.18458.2)

First published: 14 Mar 2019, **8**:287 (https://doi.org/10.12688/f1000research.18458.1)

REVISED **Amendments from Version 1**

As requested by reviewer 1 we include the exact formulas and complete code used to compute angles between helices.

See referee reports

Introduction

In RNA 3D structure prediction, knowledge-based potentials are commonly used, especially for coarse-grained approaches that are suitable for larger RNA molecules[1]. The creation of such potentials requires knowledge extraction from solved RNA structures, usually taken from the Protein Data Bank (PDB)[2]. PDB files and their newer replacement, MMCIF files, contain atomic coordinates and additional information in the header fields, but do not contain any base-pairing annotations. To extract information about base pairs and their types[3], dedicated software like MC-Annotate[4], RNAView[5], FR3D[6], DSSR[7] or the RNApdbee web server[8] is required. Due to the hierarchical organization of the RNA energy landscape[9], it is often most convenient to treat secondary and tertiary structure of RNA molecules separately and predict tertiary structures given a secondary structure[10].

For knowledge extraction from RNA-containing PDB files, it is highly desirable to have a software library at hand, which understands the semantics of RNA secondary structures and makes tasks like iterating over loops of a certain type straightforward. Ideally, such a library should be written in an easily accessible scripting language, be well documented and tested and should be available under an open-source license.

Libraries other than the `forgi` library presented here only partly fill these needs. The Vienna RNA package[11] can be used to predict secondary structures from sequence, including advanced features like G-Quadruplex prediction and incorporation of SHAPE data. It provides Python and Perl bindings, but deals exclusively with secondary structure. Biopython[12,13] is useful for dealing with RNA sequence data and can be used to load RNA 3D structures, but has no dedicated support for RNA secondary structure. PyCogent[14], a library for genomic biology, has extensive support for nucleic acid sequences, but contains only a lightweight class for RNA secondary structures and code for pseudoknot removal[15]. A stand-alone version of the latter was included into the `forgi` library under the terms of the GNU General Public License 3.0. More specialized libraries include `modeRNA`[16] (homology modeling, Python) and the FR3D suite[6] (RNA motif search, Matlab). Here, we present the `forgi` library[17], which is centered on RNA secondary structure elements (such as stems, bulges and loops) and makes them usable for 3D structure analysis. `forgi` aims at providing a high level API for many common operations, but can be easily extended with new functionality. The flexibility of providing an open source library in a scripting language is a clear benefit over other programs that are only distributed as binaries. While not restricted to work with PDB or MMCIF files, `forgi` shines especially where 3D information is analyzed in the context of its secondary structure environment.

Methods

Implementation

The `forgi` library[17] is strongly object-oriented, but takes advantage of module-level Python functions where appropriate. The core object representing the secondary structure is the `BulgeGraph` object, which holds a `Sequence` instance for the primary sequence. To include secondary structure based 3D coordinates, the `BulgeGraph`'s subclass, `CoarseGrainRNA` is available. For all-atom 3D analysis, the `forgi` library has a built-in integration with Biopython.

We will briefly describe the three main data structures that hold the sequence, secondary structure and the tertiary structure representation of an RNA.

Primary structure: The `Sequence` class. Each `BulgeGraph` holds a `Sequence` object. Since `forgi` supports loading of data from PDB files (see below), this `Sequence` object has to account for many special cases arising from experimental considerations, which will be detailed in the following paragraphs. There are two numbering schemes commonly used for sequences: 1-based indexing and indexing based on an external reference. In particular, many sequences in structural experiments use "1" to dedicate the first residue of a biological macromolecule, whereas the first residue actually used in the experiment can be upstream of the functional RNA (leading to negative indices) or after the start of the biological unit (leading to an index above 1). The latter is especially common if fragments of larger RNA molecules like the ribosome are studied. Finally, experimenters might decide to insert nucleotides in the middle of a molecule. In order not to affect the numbering of subsequent residues, these inserted residues get the same number as the previous one, followed by a letter (called insertion code).

To handle both kinds of indexing, the `Sequence` class distinguishes indices by type. Integer indices always refer to 1-based indexing, while tuples compatible to the indices used in Biopython's PDB module are interpreted as the second kind of indices.

For many applications, it is necessary to restrict the RNA alphabet to 4 letters (i.e. 4 residue types), "G", "C", "A" and "U". However, in the cell many RNA molecules are post-transcriptionally modified at certain positions. Many modifications, including the methylation of OH or NH_2 groups and A to I editing, have been implicated with a variety of biological functions[18]. During parsing of PDB files, we automatically convert such modified residues to the unmodified parent, but in addition store the modification as an annotation in the `Sequence` object. The corresponding unmodified parents for 3-letter codes of common modified residues were obtained from PDBeChem[19] (http://www.ebi.ac.uk/pdbe-srv/pdbechem/) and the `forgi` library has the ability to query this database on the fly if it encounters a new 3-letter code.

Finally, many experimental 3D structures do not contain coordinates for all residues present in the experiment. The `forgi Sequence` class can store two version of the sequence, with and without missing residues, and the file parser can extract this information from PDB and MMCIF files.

The secondary structure of an RNA is internally represented as a graph, the Bulge Graph, where secondary structure elements (stems, single-stranded regions, interior loops and hairpin loops) form the nodes. Whenever these elements are adjacent along the backbone, they are connected by an edge in this graph. During the Bulge Graph creation, each node gets a unique name such as "s0" for the first stem or "h0" for the first hairpin. The concept of the Bulge Graph, illustrated in Figure 1, has been described previously in more detail[20] and is related to the independently developed RAG (RNA as Graph) approach[21].

`forgi` supports element-based transformations of the Bulge Graph, such as condensing the secondary structure to an representation similar to RNAshapes[22], which we use to classify pseudoknots, for example (see below).

The `BulgeGraph` object allows for easy identification, selection and classification of structural domains, such as multi loops, helices consisting of multiple stems and bulges (termed "rods" in `forgi`), and pseudoknots.

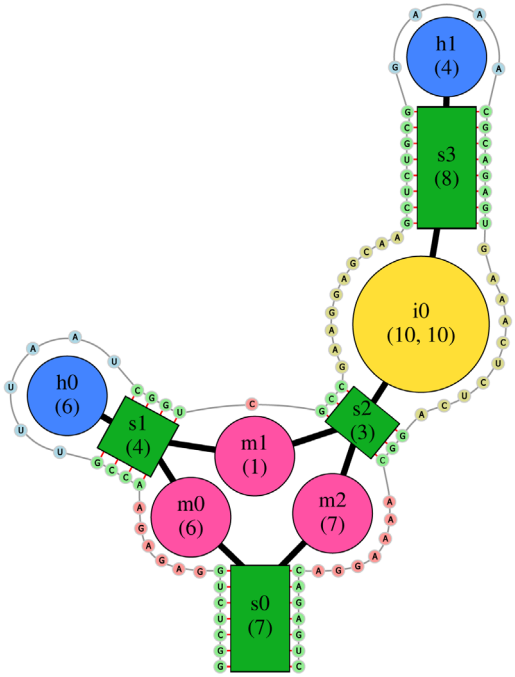

Figure 1. Illustration of the Bulge Graph representation underlying the `forgi` library. Secondary structure elements (stems, bulges and single-stranded regions) are nodes connected by edges. The sequence is shown around the Bulge Graph.

The `CoarseGrainRNA` class holds 3D structures. 3D structures are loaded from PDB or MMCIF files using Biopython's PDB module[13]. In order to assign a secondary structure to the RNA, `forgi` can call either MC-Annotate[4] (Linux only, no MMCIF-support, binary available at http://major.iric.ca/MajorLabEn/MC-Tools.html) or DSSR[7] (binary available after free registration at http://forum.x3dna.org). As an alternative, `forgi` has a built-in heuristic for the detection of canonical base pairs and GU wobble pairs. This heuristic is based on distances along the hydrogen bonds and the coplanarity of the bases, and is not intended to compete with the power of more specialized tools since it will fail in some edge cases, e.g. involving modified residues or residues with missing atoms. This heuristic is a useful fallback, if the above mentioned programs are not available.

During loading of the RNA, a helix axis is assigned to stems as described previously[20]. For each stem, we store start and end coordinates of the helix axis as well as two twist vectors that point towards the minor groove at the beginning and the end of the stem. Similarly, start and end coordinates for bulges, loops and single-stranded regions are stored and can be accessed using the element's name.

`forgi` 2.0 now fully supports co- and multi-fold structures and can load multiple chains that are connected by base pairs into a single `CoarseGrainRNA` object, while chains not connected by any base pair are loaded into separate objects.

Using Biopython's KD-Tree implementation (`Bio.PDB.NeighborSearch`), a list of residues within 6 angstrom from a non-RNA C or N atom is obtained. These residues are considered protein-/ligand interacting. Knowledge of interacting residues is particularly useful to avoid biases in statistics about structural features of bare RNA.

A cleaned version of the PDB - with modified residues converted to their canonical parent and non-RNA molecules removed is stored as Biopython chains in the `CoarseGrainRNA`'s `chains` attribute.

Operation

`forgi`[17] is compatible with Python 2.7, 3.5 and 3.6, should run on all operating systems where its dependencies are available and has been tested on Linux, Mac and Windows. It makes heavy use of the NumPy[23] library to speed-up array-based calculations and also depends on SciPy[24], NetworkX[25], Biopython[12,13], pandas[26,27] and `appdirs`, all available via the Python Package Index (PyPi) or Anaconda.

Helpful utility scripts. `forgi` comes with the two very useful scripts: `rnaConvert.py` can be used to convert between many common file formats for RNA structures, including the Vienna format and fasta variants, the bpseq format, the connectivity table (ct) format, MMCIF format and PDB files. `visualize_rna.py` can be used to display a coarse-grained representation of an RNA's secondary structure in PyMol alongside the all atom structure from PDB files, producing visualizations like those in Figure 3 and Figure 5.

Analysis of stacking geometries. To illustrate how the `forgi` library makes secondary structure elements usable for 3D structure analysis, we used it to analyze the stacking of adjacent helices in multi loops and pseudoknots in a representative set of RNA 3D structures[28] (version 3.36, available at http://rna.bgsu.edu/rna3dhub/nrlist/).

While loading these 3D structures into `forgi`, DSSR[7] (Version v1.7.1-2017nov01) was called to obtain the secondary structure and nucleotide level reference stacking annotations. We count a pair of connected helices as stacking, if at least one nucleotide of the first helix's closing/opening base pair is in a continuous stack with at least one nucleotide of the second helix's opening/closing base pair. This allows for any number of stacking nucleotides between the stems (not necessarily connected via the backbone) and for bulged out nucleotides which do not contribute to the stack. Our definition of stacking is more relaxed than stricter criteria used elsewhere[29].

For each pair of adjacent stems within a junction, we used `forgi` to calculate a number of properties, such as the angle between the stem vectors, the separation vector between the stems' ends and an offset value between the stems.

These properties are based on the axes that were fitted to the helices (see above) and the multi loop segment that connects the two stems. We define the stem vectors $\boldsymbol{v}_h$ as pointing away from the multi loop segment along the helix axis, where $\boldsymbol{s}_h$ and $\boldsymbol{e}_h$ are the helix's start and end coordinates.

$$\boldsymbol{v}_h = \begin{cases} \boldsymbol{e}_h - \boldsymbol{s}_h & \text{if } \boldsymbol{s}_h \text{ is at the side of the multiloop} \\ \boldsymbol{s}_h - \boldsymbol{e}_h & \text{if } \boldsymbol{e}_h \text{ is at the side of the multiloop} \end{cases}$$

The angle between the adjacent stems i and h is then defined as the angle between their stem vectors $\boldsymbol{v}_i$ and $\boldsymbol{v}_h$:

$$\cos(\alpha) = \frac{\boldsymbol{v}_h \cdot \boldsymbol{v}_i}{\|\boldsymbol{v}_h\| \cdot \|\boldsymbol{v}_i\|}$$

The offset between stems is calculated as distance between two rays R_h that start at the helix's end closer to the multi loop segment, $\boldsymbol{c}_h$, and extend the helix axis:

$$\boldsymbol{c}_h = \begin{cases} \boldsymbol{s}_h & \text{if } \boldsymbol{s}_h \text{ is at the side of the multiloop} \\ \boldsymbol{e}_h & \text{if } \boldsymbol{e}_h \text{ is at the side of the multiloop} \end{cases}$$

$$R_h = \left\{ \boldsymbol{X} \mid \boldsymbol{X} = \boldsymbol{c}_h + \lambda \boldsymbol{v}_h, \lambda \geq 0 \right\}$$

Depending on the helix orientation, the distance between these rays is either the distance between two lines or the distance between two points or between one point and one line, all of which can be calculated with standard formulas.

The following code shows how `forgi` was used to calculate these properties for multi loops:

```
from forgi import load_rna
from forgi.threedee.utilities.vector import vec_angle, vec_distance
from forgi.threedee.utilities.vector import line_segment_distance

def main():
    # Load the PDB into CoarseGrainRNA instances
    # rnas is a list, because a PDB can contain multiple connected components
    # This also loads the DSSR json annotations.
    rnas = load_rna("path/to/file.cif", pdb_remove_pk=False,
                    pdb_annotation_tool="DSSR")
    for rna in rnas:
        for junction in rna.junctions:
            print_junction_parameters(rna, junction)

def print_junction_parameters(rna, junction):
    """
    Prints the angles and offsets of a junction.
    :param rna: A BulgeGraph Object
    :param junction: A list of element names (strings)
    """
    for ml in junction:
        # rna.edges holds adjacent nodes in the BulgeGraph
        stem1, stem2 = rna.edges[ml]
        # rna.coords holds the 3D coordinates of structure elements
        # rna.coords["s0"] returns a tuple start-, end-coordinates of stem "s0"
        # Furthermore rna.coords has the get_direction function to give the
        # vector pointing from the start to the end coordinate.
        direction1 = rna.coords.get_direction(stem1)
        direction2 = rna.coords.get_direction(stem2)
        # Get the indices into rna.coords for the stem sides closer to
        # and further away from the multiloop segment ml
        c1, f1 = rna.get_sides(stem1, ml)
        c2, f2 = rna.get_sides(stem2, ml)
        # Make sure the stem vector points away from the multiloop
        if c1 == 1:
            direction1 = - direction1
        if c2 == 1:
            direction2 = - direction2
        angle = vec_angle(direction1, direction2) # imported above
        is_stacking_dssr = (ml in rna.dssr.stacking_loops())
        closer1 = rna.coords[stem1][c1]
```

```
closer2 = rna.coords[stem2][c2]
# Calculate the offset as distance between rays.
offset = vec_distance(*line_segment_distance(
                            closer1, closer1+100000*direction1,
                            closer2, closer2+100000*direction2))
print("{}\t{}\t{}".format(angle, is_stacking_dssr, offset))
```

We then used pandas[26,27], Matplotlib[30] and a custom library (https://github.com/Bernhard10/filterAndView) to analyze and visualize the collected data. The results were collected for different classes of RNA independently. This was necessary, because the representative sets of RNA structures contain, by design, homologs of the same molecule in multiple species.

For the classification of pseudoknots, Reidys' concept[31] based on the definition of the mathematical genus is used. Classes of pseudoknots are defined based on their shadow representations, which contain only crossing base pairs and only one base pair each. On the level of these shadows, only four distinct classes exhibit genus 1, two of which are well known: the H-type pseudoknot and the kissing hairpin. pseudoknots with higher genus contain, among others, the case where a genus 1 pseudoknot is nested within another pseudoknot.

In combination with the `forgi` library, we wrote a tool that is able to convert structures to their shadow representation, identify and classify the pseudoknots within the structure and describe their helix arrangement in the 3D structure. This tool, called `pseudoknot_analyzer.py`, is distributed with the `forgi` library in the folder "examples". We used it to gather statistics about simple H-type and kissing hairpin pseudoknots. Figure 4a, b illustrates how we measured the angle between stems in pseudoknots via vector directions. In kissing hairpins the angles α and β restrict the possible values of the angle γ. We also include the representative structure of an intermolecular kissing hairpin interaction (lacking the green connection in Figure 4b) in our analysis. In this special case α and β are indistinguishable and were assigned arbitrarily.

Results

We used the helix-centered representation of the 3D structure to analyze the geometry of coaxially stacking helices. The relative geometry of two cylinders in 3D space can be described by five parameters: A separation vector (three parameters) and two angles for the relative orientation in 3D space. In Figure 2 and Figure 4, we show the single angle calculated between the vectors along the helix axes, as a proxy for these parameters.

The distribution of angles between adjacent stems in tRNAs multi loops (see Figure 2a) shows the expected bimodal distribution with one mode slightly below 90° and another mode between 160° and 170°, which fits to the known helix arrangement in the L-shaped tRNA. As confirmed by comparison to annotations with DSSR, the second peak is almost exclusively due to stacking helices, even at angles as small as 140°. In Figure 3 we show an example of such a coaxial stack that has been strongly bent (possibly by the tRNA synthetase) without completely breaking the stack or affecting the canonical tRNA secondary structure.

The second family of RNA molecules with lots of data available is ribosomal RNA. Here we find that the angle of adjacent stems in multi loops is almost uniformly distributed above 20° until a peak from 135° to 170°, where DSSR annotates roughly half of the geometries as stacking (see Figure 2c). 7% of the stacking geometries and

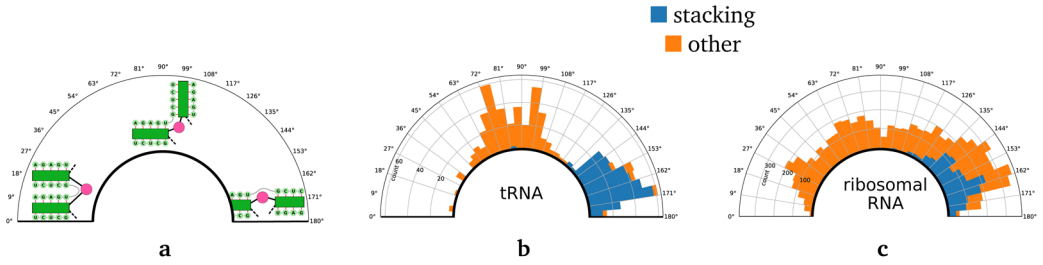

Figure 2. Distribution of angles between adjacent stems in multi loops. (**a**) Angles close to 0° mean parallel stems whereas angles close to 180° correspond to potentially stacking stems. Angle distributions in (**b**) tRNAs and (**c**) ribosomal RNA are shown as a histogram mapped onto a circular plot illustrating the angles. The (inner) blue bars are instances where DSSR detects stacking on the atom-level scale. The orange bars start at the top of the blue bars and indicate geometries where DSSR does not detect stacking.

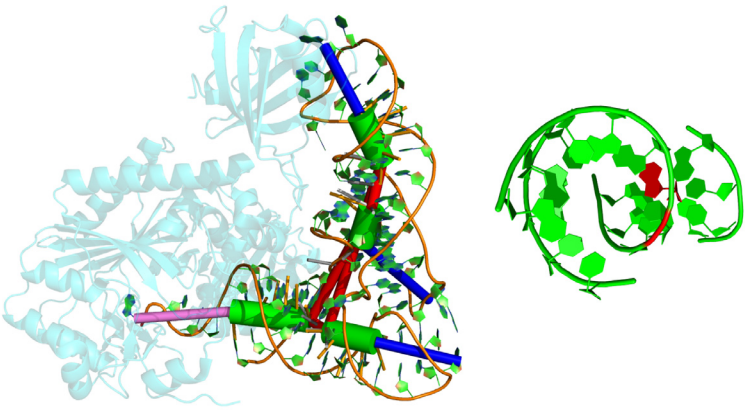

Figure 3. Example of a tRNA where stacking between two non-collinear stems occurs (PDB id 4WJ4[32]). Green cylinders were fitted to stems, blue cylinders represent hairpins and the pink cylinder the unpaired nucleotides at the 3' end. Red cylinders connect stems and indicate single-stranded connections between these stems (multi loop segments). Left: The stems of the anticodon arm (top) and the D-arm stack (according to DSSR) despite being at an angle of 143°. Right: View along the axis of the anticodon arm's stem. The red nucleotide is the unpaired multi loop segment which mediates stacking to the stem of the D-arm. This illustration was generated using PyMol[33] via the `visualize_rna.py` wrapper in `forgi`.

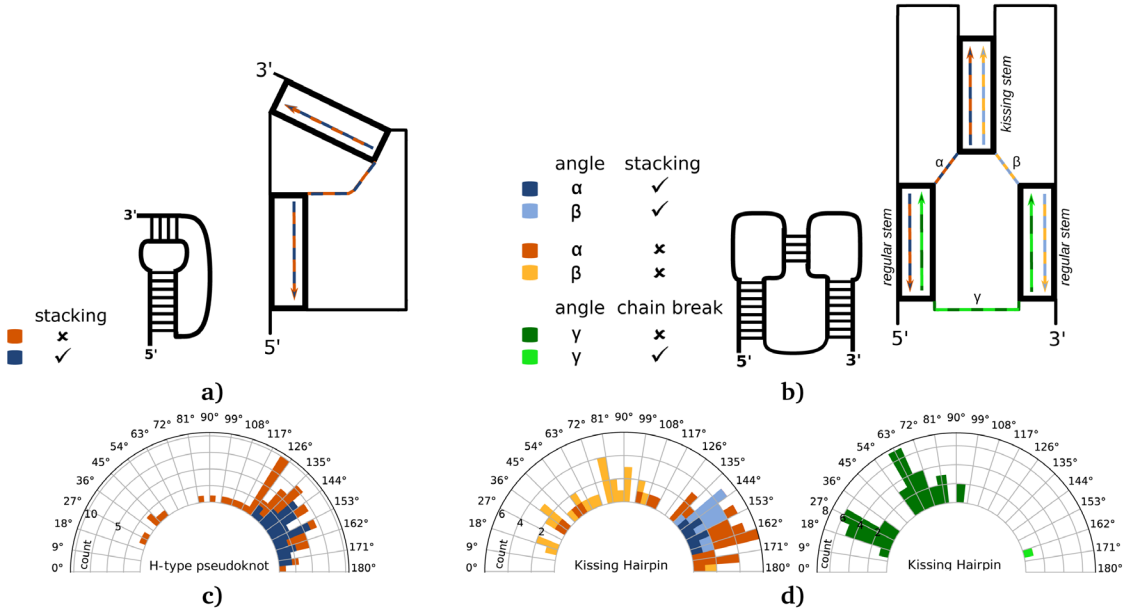

Figure 4. Angles between the stems in pseudoknots. (**a**) and (**b**) show the vector directions used for the angle measurement. The distributions of the angles between adjacent stems building **c**) an H-type pseudoknot or (**d**) a kissing hairpin are shown as histograms mapped to a circle. Furthermore, the distribution of angles between the regular stems of kissing hairpin pseudoknots is shown in the second panel of (**d**). Like mentioned in Figure 2, the angles in **c**) measured between stacking stems (detected via DSSR) are colored in blue. Geometries without stacking are colored in orange. The color scheme in (**d**) (shades of orange, blue and green) refer to the respective vectors/measured angles (α, β, γ) in the same color scheme as in (**b**). Blue bars stand for stacking between the two related stems, whereas orange ones for non-stacking stems. Dark green bars in the second kissing hairpin associated histogram show angles measured between a intramolecular interaction (kissing hairpin pseudoknot), whereas light green bars stand for angles measured between two RNA chains (kissing hairpin interaction).

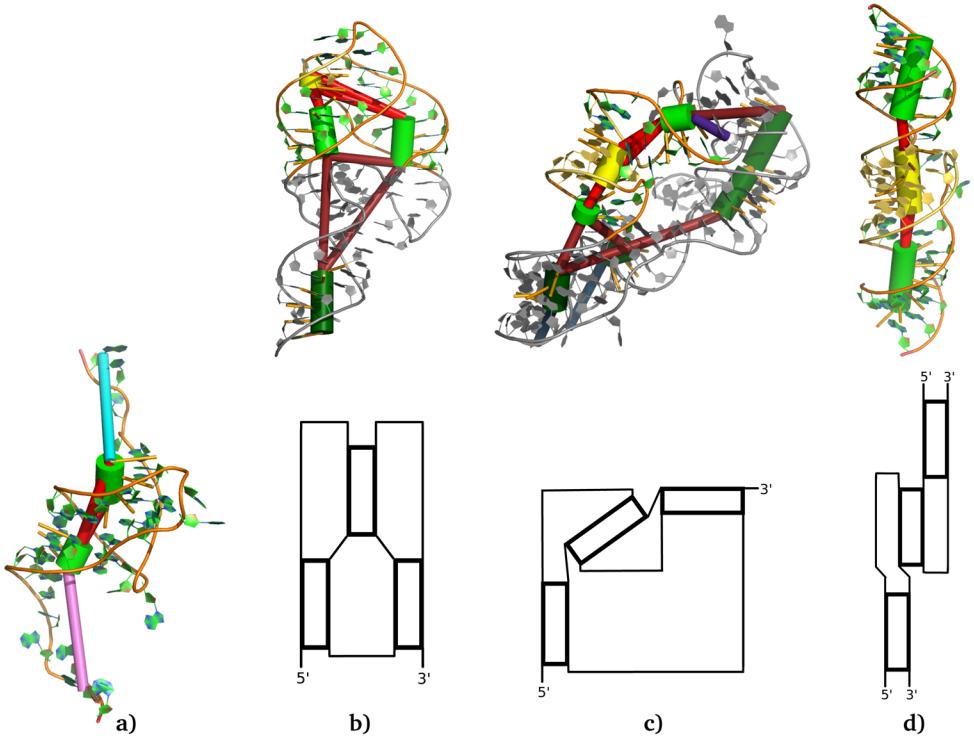

Figure 5. Examples of coaxial stacking within (**a**) an H-type pseudoknot (PDB id 2XD0) and (**b**-**d**) the three major structural families of kissing hairpins (PDB ids 5KPY, 4FRN and 1ZCI, from left to right). In all four representations the green cylinders were fitted to stems, the turquoise and pink cylinder represent the unpaired nucleotides at the 5' and 3' end respectively. Red cylinders connect stems and indicate single-stranded connections between these stems (pseudoknot or multi loop segments). The light-colored stems in (**b**–**d**) represent the regular stems of kissing hairpins, whereas the kissing stem is colored yellow. Note that panel (**d**) (PDB id 1ZCI) shows a kissing hairpin interaction between two RNA chains. Below the 3-dimensional representation of the kissing hairpins, we show a schematic sketch of the structural family's helix arrangement. These illustrations were generated using PyMol[33] via the `visualize_rna.py` wrapper in `forgi`.

67% of the non-stacking geometries with an angle above 140° also have an offset above 10Å between the extended stem axes.

Additionally we analyzed the angle distribution between the stems forming simple H-type pseudoknots or kissing hairpins. Most angles measured within an H-type pseudoknot are between 120° and 180° (see Figure 4c). In this range, 35 out of 67 instances correspond to stacking. One example within this range is shown in Figure 5a, which represents a processed non-coding RNA, which regulates a bacterial antiviral system (PDB id 2XD0[34]).

The distribution illustrated in Figure 4d shows mainly values between 130° and 180° for the angles β and especially α. One additional angle measurement between the two regular stems (see Figure 4d, angle γ) shows one peak at about 20° and one at about 65°. Within the class of kissing hairpins we often find coaxial stacking, but most of the time only one of the two regular stems stacks with the kissing stem in the middle. With the help of the coarse grained representation we were able to divide the class of kissing hairpins into three different structural families (see Figure 5b-d).

The first family is especially common among (A-)riboswitches. Here the two regular stems are oriented almost parallel (γ below to 32°) with the second one (counting from the 5' end) stacking onto the kissing stem. The angles α and β are above 130°. One example is the structure with PDB id 5KPY[35] shown in Figure 5b. Here, the link from the first regular stem stacks onto it and interacts with the kissing stem via base multiplets and A-Minor interactions. This way, the roughly parallel orientation of all 3 stems is stabilized by stacking and base-pairing interactions. Interestingly, the kissing stem is more parallel to the first stem than to the second stem, onto which it stacks directly ($\alpha > \beta$).

The second structural family is shown in Figure 5c. Here the two regular stems are close to 90°, whereas the kissing stem is along the arc between them. This conformation is found in several groups of non coding RNA including some 23S ribosomal RNA and the cobalamin riboswitch regulatory element (PDB id 4FRN[36]) shown in Figure 5c.

The two families as well as some conformations in between are observable within intramolecular pseudoknots. As mentioned, `forgi` is also able to model multiple RNA chains that interact with each other forming an intermolecular complex. One example is the dimerization initiation site of the HIV type 1 (PDB id 1ZCI[37]), illustrated in Figure 5d and being the only example of the third family of kissing hairpin pseudoknots. Here the kissing hairpin interaction shows a nearly perfect coaxial stacking between all three involved stems. In this case β is close to 0°, because stacking takes place at the other side of the kissing stem and thus the complementary angle is close to 180°.

Discussion

Using the `forgi` library[17] we have analyzed the relative orientation of helices in junctions and pseudoknots. While stacking between adjacent helices is common, we found that their orientation often deviates significantly from co-linearity, suggesting that junctions introduce flexibility into the RNA structure while still maintaining the benefit of stacking interactions. Previous studies that assign coaxial geometries based on manual inspection[38] miss this deviation from coaxiality that becomes apparent once you actually calculate the helix axis. Our approach allowed us to perform this analysis on the scale of stems, as opposed to the smaller all-atom scale used by the program DSSR and in previous surveys[39] or the level of networks of (noncanonical) base pairs[38].

Conversely, we found that out of 1257 pairs of adjacent stems in ribosomal RNA structures with an angle above 140°, only roughly half (608) are annotated as stacking by DSSR. This becomes clear if we consider that the angle between stems is only a proxy for a 5-dimensional orientation parameter. In particular, a large offset means that stem axes with an angle close to 180° can be parallel without being coaxially stacked. Indeed, it is not uncommon that all three stems in a 3-way junction are roughly parallel, with two stems forming a coaxial stack and the third having a higher offset[40].

There is growing interest in predicting pseudoknots in RNA structures as they are involved in a variety of biological functions[41]. The `forgi` library allows us to easily identify pseudoknots in RNA 3D structures and gather statistics on the frequency of pseudoknot types, sizes, and composition. Kissing hairpins formed between two RNA molecules are known to often form continuous stacks between all three helices, such as the pseudoknot in Figure 5d. In contrast to these intermolecular kissing hairpins, we find that stacking between all three helices is seldom possible in intramolecular pseudoknots. Instead we find that the limited length of all loop segments makes it nearly impossible to form coaxial stacking of all three stems in one line in a single RNA chain. Thus, in the main families of kissing hairpin conformations, the regular (outer) stems are oriented parallelly or almost perpendicular with the kissing helix stacking onto at most one of the two regular stems. These structures are often further stabilized by base triplets like those found in Figure 5b.

Similar to stacking conformations in multi loops, many stacking helices in pseudoknots form angles below 180°. A deviation of the coaxiality between stems can have multiple reasons such as a higher flexibility of short helices within a pseudoknot as well as strain from short stem connections. But there are also structures that show more classical coaxial stacking like PDB id 2XD0, see Figure 5a.

Results and challenges of the application of `forgi` to the analysis of pseudoknots are described in more detail in the thesis by Beckmann[42].

Varying additional parameters like the minimal number of base pairs required to count an interaction as a helix allow for a better understanding about the principles of the formation of three-dimensional RNA structures. The insight in the world of pseudoknots and multi loops presented here can support the improvement of the prediction of RNA structures and help identify unrealistic multi-loop conformations predicted by current RNA structures prediction tools. We are now in the process of implementing additional features for the RNA 3D structure prediction program Ernwin[20] based on our findings about stacking and pseudoknots.

Conclusions

`forgi`[17] is a multi-purpose Python library for dealing with RNA on the levels of sequence, secondary structure and tertiary structure. By providing our code as a Python library, we give users of our tool more flexibility than a single executable program could provide. Furthermore, our code is fully open source, giving researchers the possibility to comprehend and where needed extend the inner workings of the code.

`forgi` is well documented, with a tutorial available at https://ViennaRNA.github.io/forgi/ and a complete API documentation following the Python standard of docstrings parsable by Sphinx. It contains an extensive automatic test suite (unit and integration tests) and has been optimized for speed, maintainability and usefulness.

Data availability

Underlying data

The PDB ids analyzed were taken from version 3.36 of the representative sets of RNA 3D structures[28] at a cutoff of 4 Å, http://rna.bgsu.edu/rna3dhub/nrlist/release/3.36/4.0A.

Extended data

Open Science Framework: 3D based on 2D: `forgi` 2.0 Extended Data. https://doi.org/10.17605/OSF.IO/HDJRU[43]. This project contains the following extended data files:

- **assignment_of_classes_to_pdbids.csv**: Assignment of PDB ids to RNA type. These annotations were generated in a semi-manual way.
- **multiloop_angles.csv**: Angles and offsets measured in PDB structures for multi loops, generated with the script `describe_rna.py`, which is available in the examples folder of `forgi`.
- **pseudoknot_angles.tsv**: Angles measured in pseudoknots, generated with `pseudoknot_analyzer.py`, which is available in the examples folder of `forgi`.

Software availability

Software available from: https://pypi.org/project/forgi/ (`pip install forgi`) and Bioconda.

Archived source code at time of publication: https://doi.org/10.5281/zenodo.2582870[17].

License: GNU General Public License 3.0.

Grant information

This work was partly funded by the Austrian science fund (FWF) projects F 43 "RNA regulation of the transcriptome" and I 2874 "Prediction of RNA-RNA interactions" and Doctoral College W 1207 "RNA Biology".

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgments

We thank all contributors, who submitted a pull-request or filed an issue in the `forgi` repository on github. We thank Roman Ochsenreiter for proofreading, Gregor Entzian for useful inputs about many quirks in the PDB format and for proofreading and Lorena Pantano for creating the first version of a Bioconda recipe for `forgi`.

References

1. Dawson WK, Maciejczyk M, Jankowska EJ, *et al.*: **Coarse-grained modeling of RNA 3D structure.** *Methods.* 2016; **103**: 138–156.
PubMed Abstract | Publisher Full Text

2. Berman H, Henrick K, Nakamura H, *et al.*: **The worldwide Protein Data Bank (wwPDB): ensuring a single, uniform archive of PDB data.** *Nucleic Acids Res.* 2007; **35**(Database issue): D301–D303.
PubMed Abstract | Publisher Full Text | Free Full Text

3. Leontis NB, Westhof E: **Geometric nomenclature and classification of RNA base pairs.** *RNA.* 2001; **7**(4): 499–512.
PubMed Abstract | Publisher Full Text | Free Full Text

4. Gendron P, Lemieux S, Major F: **Quantitative analysis of nucleic acid three-dimensional structures.** *J Mol Biol.* 2001; **308**(5): 919–936.
PubMed Abstract | Publisher Full Text

5. Yang H, Jossinet F, Leontis N, *et al.*: **Tools for the automatic identification and classification of RNA base pairs.** *Nucleic Acids Res.* 2003; **31**(13): 3450–3460.
PubMed Abstract | Publisher Full Text | Free Full Text

6. Sarver M, Zirbel CL, Stombaugh J, *et al.*: **FR3D: finding local and composite recurrent structural motifs in RNA 3D structures.**

J Math Biol. 2008; **56**(1–2): 215–252.
PubMed Abstract | Publisher Full Text | Free Full Text

7. Lu XJ, Bussemake HJ, Olson WK: **DSSR: an integrated software tool for dissecting the spatial structure of RNA.** *Nucleic Acids Res.* 2015; **43**(21): e142.
PubMed Abstract | Publisher Full Text | Free Full Text

8. Zok T, Antczak M, Zurkowski M, *et al.*: **RNApdbee 2.0: multifunctional tool for RNA structure annotation.** *Nucleic Acids Res.* 2018; **46**(W1): W30–W35.
PubMed Abstract | Publisher Full Text | Free Full Text

9. Mustoe AM, Brooks CL, Al-Hashimi HM: **Hierarchy of RNA functional dynamics.** *Annu Rev Biochem.* 2014; **83**(1): 441–466.
PubMed Abstract | Publisher Full Text | Free Full Text

10. Thiel BC, Flamm C, Hofacker IL: **RNA structure prediction: from 2D to 3D.** *Emerg Top Life Sci.* 2017; **1**(3): 275–285.
Publisher Full Text

11. Lorenz R, Bernhart SH, Höner Zu Siederdissen C, *et al.*: **ViennaRNA Package 2.0.** *Algorithms Mol Biol.* 2011; **6**(1): 26.
PubMed Abstract | Publisher Full Text | Free Full Text

12. Cock PJ, Antao T, Chang JT, *et al.*: **Biopython: freely available Python tools for computational molecular biology and bioinformatics.** *Bioinformatics.* 2009; **25**(11): 1422–1423.
PubMed Abstract | Publisher Full Text | Free Full Text

13. Hamelryck T, Manderick B: **PDB file parser and structure class implemented in Python.** *Bioinformatics.* 2003; **19**(17): 2308–2310.
PubMed Abstract | Publisher Full Text

14. Knight R, Maxwell P, Birmingham A, *et al.*: **PyCogent: a toolkit for making sense from sequence.** *Genome Biol.* 2007; **8**(8): R171.
PubMed Abstract | Publisher Full Text | Free Full Text

15. Smit S, Rother K, Heringa J, *et al.*: **From knotted to nested RNA structures: a variety of computational methods for pseudoknot removal.** *RNA.* 2008; **14**(3): 410–416.
PubMed Abstract | Publisher Full Text | Free Full Text

16. Rother M, Rother K, Puton T, *et al.*: **ModeRNA: a tool for comparative modeling of RNA 3D structure.** *Nucleic Acids Res.* 2011; **39**(10): 4007–4022.
PubMed Abstract | Publisher Full Text | Free Full Text

17. Thiel B, Kerpedjiev P, tcarlile, *et al.*: **Viennarna/forgi: Forgi version 2.0**. 2019.
http://www.doi.org/10.5281/zenodo.2582870

18. Licht K, Jantsch MF: **Rapid and dynamic transcriptome regulation by RNA editing and RNA modifications.** *J Cell Biol.* 2016; **213**(1): 15–22.
PubMed Abstract | Publisher Full Text | Free Full Text

19. Dimitropoulos D, Ionides J, Henrick K: **Using MSDchem to search the PDB ligand dictionary.** *Curr Protoc Bioinformatics.* 2006; Chapter 14: Unit14.3.
PubMed Abstract | Publisher Full Text

20. Kerpedjiev P, Höner Zu Siederdissen C, Hofacker IL: **Predicting RNA 3D structure using a coarse-grain helix-centered model.** *RNA.* 2015; **21**(6): 1110–1121.
PubMed Abstract | Publisher Full Text | Free Full Text

21. Schlick T: **Adventures with RNA graphs.** *Methods.* 2018; **143**: 16–33.
PubMed Abstract | Publisher Full Text | Free Full Text

22. Steffen P, Voss B, Rehmsmeier M, *et al.*: **RNAshapes: an integrated RNA analysis package based on abstract shapes.** *Bioinformatics.* 2006; **22**(4): 500–503.
PubMed Abstract | Publisher Full Text

23. Oliphant TE: **Guide to NumPy**. 2nd Edition. CreateSpace Independent Publishing Platform, 2015.
Reference Source

24. Jones E, Oliphant T, Peterson T, *et al.*: **SciPy: Open source scientific tools for Python, since 2001**. [online, accessed 8 Oct 2018].
Reference Source

25. Hagberg AA, Schult DA, Swart PJ: **Exploring network structure, dynamics, and function using networkx**. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*. Pasadena, CA USA. 2008; 11–15.
Reference Source

26. McKinney W: **Data structures for statistical computing in python**. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference.* 2010; 51–56.
Reference Source

27. McKinney W: **pandas: a foundational python library for data analysis and statistics**. *Python for High Performance and Scientific Computing.* 2011; 1–9.
Reference Source

28. Leontis NB, Zirbel CL: **Nonredundant 3D structure datasets for RNA knowledge extraction and benchmarking**. In *Nucleic Acids and Molecular Biology*. Springer Berlin Heidelberg. 2012; 281–298.
Publisher Full Text

29. Tyagi R, Mathews DH: **Predicting helical coaxial stacking in RNA multibranch loops.** *RNA.* 2007; **13**(7): 939–951.
PubMed Abstract | Publisher Full Text | Free Full Text

30. Hunter JD: **Matplotlib: A 2D graphics environment.** *Comput Sci Eng.* 2007; **9**(3): 90–95.
Publisher Full Text

31. Reidys CM, Huang FW, Andersen JE, *et al.*: **Topology and prediction of RNA pseudoknots.** *Bioinformatics.* 2011; **27**(8): 1076–1085.
PubMed Abstract | Publisher Full Text

32. Suzuki T, Nakamura A, Kato K, *et al.*: **Structure of the *Pseudomonas aeruginosa* transamidosome reveals unique aspects of bacterial tRNA-dependent asparagine biosynthesis.** *Proc Natl Acad Sci U S A.* 2015; **112**(2): 382–387.
PubMed Abstract | Publisher Full Text | Free Full Text

33. Schrödinger, LLC: **The PyMOL molecular graphics system.** version 1.8. 2015.

34. Blower TR, Pei XY, Short FL, *et al.*: **A processed noncoding RNA regulates an altruistic bacterial antiviral system.** *Nat Struct Mol Biol.* 2011; **18**(2): 185–191.
PubMed Abstract | Publisher Full Text | Free Full Text

35. Porter EB, Polaski JT, Morck MM, *et al.*: **Recurrent RNA motifs as scaffolds for genetically encodable small-molecule biosensors.** *Nat Chem Biol.* 2017; **13**(3): 295–301.
PubMed Abstract | Publisher Full Text | Free Full Text

36. Johnson JE Jr, Reyes FE, Polaski JT, *et al.*: **B_{12} cofactors directly stabilize an mRNA regulatory switch** *Nature.* 2012; **492**(7427): 133–137.
PubMed Abstract | Publisher Full Text | Free Full Text

37. Ennifar E, Dumas P: **Polymorphism of bulged-out residues in HIV-1 RNA DIS kissing complex and structure comparison with solution studies.** *J Mol Biol.* 2006; **356**(3): 771–782.
PubMed Abstract | Publisher Full Text

38. Laing C, Schlick T: **Analysis of four-way junctions in RNA structures.** *J Mol Biol.* 2009; **390**(3): 547–559.
PubMed Abstract | Publisher Full Text | Free Full Text

39. Holbrook SR: **Structural principles from large RNAs.** *Annu Rev Biophys.* 2008; **37**(1): 445–464.
PubMed Abstract | Publisher Full Text

40. Lescoute A, Westhof E: **Topology of three-way junctions in folded RNAs.** *RNA.* 2006; **12**(1): 83–93.
PubMed Abstract | Publisher Full Text | Free Full Text

41. Staple DW, Butcher SE: **Pseudoknots: RNA structures with diverse functions.** *PLoS Biol.* 2005; **3**(6): e213.
PubMed Abstract | Publisher Full Text | Free Full Text

42. Beckmann IK: **Identification and Classification of Pseudoknots and their Impact on RNA 3D Structure Prediction**. Master's thesis, University of Vienna, 2018.
Reference Source

43. Thiel BC, Beckmann IK, Kerpedjiev P, *et al.*: **3D based on 2D: Forgi 2.0 Extended Data**. 2019.
https://www.doi.org/10.17605/OSF.IO/HDJRU

Open Peer Review

Current Referee Status: ✔ ✔

Version 2

Referee Report 23 April 2019

https://doi.org/10.5256/f1000research.20631.r47470

✔ **Marta Szachniuk**
Institute of Computing Science & European Centre for Bioinformatics and Genomics, Poznan University of Technology, Poznań, Poland

I'm satisfied with all modifications the authors made.

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Structural bioinformatics

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Referee Report 27 March 2019

https://doi.org/10.5256/f1000research.20195.r45731

✔ **Jerome Waldispühl** , **Roman Sarrazin-Gendron**
School of Computer Science, McGill University, Montreal, QC, Canada

The authors present a new version of the forgi software, a python library to analyze the tertiary structure of secondary structure elements in RNA. It takes as input almost any format of 2D or 3D information.

This manuscript describes a new method for easily obtaining the angles and stacking patterns in a multi-branched loop and a pseudoknot/kissing hairpin. The authors validate their results by testing the software on known pseudo-knots and junctions, and obtain results that are consistent with the state-of-the-art knowledge of RNA local structure organization.

The authors highlight significant improvements compared to previously published packages and specifically all-atom based methods. For instance, a deviation from coaxiality is very common but can be easily missed if a manual inspection is performed without doing computations.

The software is clearly presented, well-validated, freely available and open source. The results presented

are significant and the tool will be very useful to the RNA bioinformatics community.

Although, this manuscript does not discuss extensively the analysis of the results, the latter is included in the first author's thesis, and freely accessible as further reading.
`

Minor remarks

- Page 4: ''with and without missing residues'', it is easy to imagine how the graph without the missing residue is, but how are the ''missing residues'' added? (i.e. are you inferring which residues are missing, and if yes, how?)
- Page 5: The authors mention a ''cleaned version'' of the PDB. Is the only ''cleaning'' performed is the conversion of the residue names?
- Page 5: It would be helpful to clarify what is a ''simplified example'' of the code? What sort of simplification was applied to it? Is the real code significantly more complicated to use than the example? If it is not, the authors might want to consider rewording this as it appears to undermine the real simplicity of use of forgi (which is very user friendly).

Is the rationale for developing the new software tool clearly explained?
Yes

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: RNA bioinformatics

We have read this submission. We believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 01 Apr 2019

Bernhard Thiel, University of Vienna, Austria

Thank you for the constructive comments.

Our PDB and CIF parsers extract the information about missing residues from the files. For this purpose we have patched the biopython PDB parsing routine; this has already been merged into the biopython source. Following your input, we have revised the text to reflect this.

Yes, the cleaned version only has modified residues renamed and non-rna molecules removed. We hope this is now clear from the revised text.

We have expanded the sample code to compute the proper direction of helices, thus it is now fully functional and no longer "simplified".
We have also added the code used to calculate the offset.

Competing Interests: No competing interests were disclosed.

Referee Report 27 March 2019

https://doi.org/10.5256/f1000research.20195.r45753

Marta Szachniuk
Institute of Computing Science & European Centre for Bioinformatics and Genomics, Poznan University of Technology, Poznań, Poland

The article is written in a clear and concise way. Illustrations perfectly complement the content. I have only 3 comments to the authors and I would like them to be taken into account in the revision:

1. For unknown reasons, the authors introduced a new spelling of the name pseudoknot. Throughout the whole article, they use spelling with space: "pseudo knots". I haven't met such a form before. The authors of the article also used the term "pseudoknot" and not "pseudo knot" in their previous works. So, the spelling should be corrected.
2. On page 3, the authors write about FR3D suite, but they do not cite the paper about the tool. Please, include the appropriate citation here.

3. As the paper subject is calculating helix angles and stacking parameters, I would expect to see the formulas used to calculate these data (or - at least - the pseudocode showing how they are calculated). The lack of such details is serious negligence.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Structural bioinformatics

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 01 Apr 2019

Bernhard Thiel, University of Vienna, Austria

Thank you for the constructive comments.

We have now included the exact formulas for the helix angles as requested. In particular this makes clear what sign we use for the helix vectors.

The FR3D reference was present in our LaTeX source and the resulting HTML version, but got lost in the pdf conversion. This has been fixed in the revised version.

Competing Interests: No competing interests were disclosed.