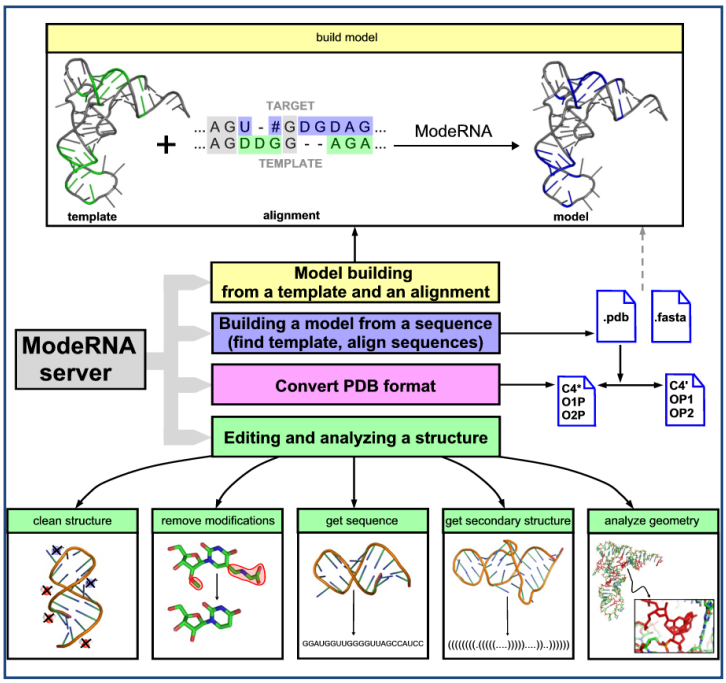
RNA tertiary structure prediction with ModeRNA:

A model of tRNA^{Thr} from *E. coli*



Aim of the tutorial

In this tutorial we are explain the usage of the ModeRNA program and web server. The ModeRNA tool uses the comparative modeling approach to build 3D models of RNA, and can be applied when a structural template is available and an alignment of reasonable quality can be prepared.

Summary: You will construct a 76-nt model of tRNA^{Thr} from *E. coli using*

- *the ModeRNA program (http://iimcb.genesilico.pl/moderna/) and*
- *the ModeRNA web server (http://genesilico.pl/modernaserver)*.*

* ModeRNA can be freely downloaded from http://iimcb.genesilico.pl/moderna/ under the conditions of the General Public License. It runs under LINUX, Windows, and Mac OS.

Authors of the tutorial: Kaja Milanowska, Joanna Kasprzak with the help of Magdalena Rother and Kristian Rother.

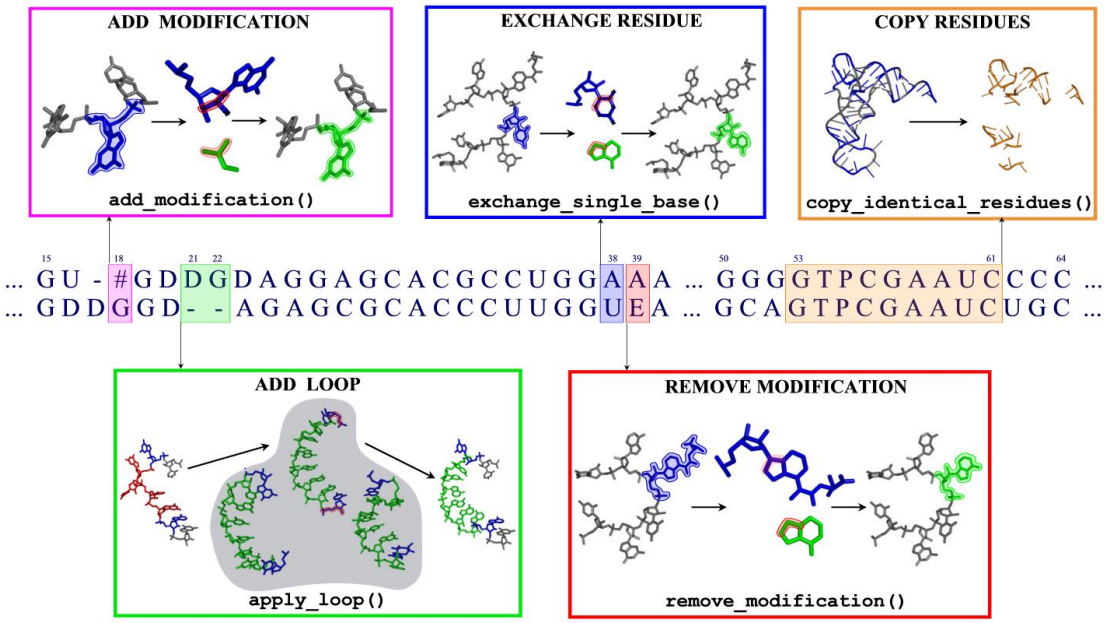
The tutorial is based on the publication: RNA tertiary structure prediction with ModeRNA. Rother M, Rother K, Puton T, Bujnicki JM. Nettab [under revision]

Target sequence: The 76-nt sequence of tRNA^{Thr} from *E. coli* (the target) can be found at: http://genesilico.pl/modernaserver/tutorial/tutorial.fasta

Overview: Comparative modeling of RNA structures

In analogy to proteins, the function of RNA depends on its structure and dynamics, which are encoded in the linear sequence. Thus, the function of both proteins and RNAs depends on the three-dimensional structure and dynamics, which in turn is encoded in the linear sequence of individual molecules. The knowledge of structure is very important for the understanding of RNA and protein function. However, experimental sequence determination of genes and entire genomes, from which the sequences of RNAs can be reliably inferred, is much cheaper and simpler than experimental determination of structures. As a consequence, the rate of macromolecular structure determination lags behind the rate of determination of new sequences and the gap between the number of known structures and known sequences continues to widen. This is why a few theoretical methods for structure determination of RNA appeared.

ModeRNA is a program for comparative modeling of RNA 3D structures. It requires a pairwise sequence alignment and a structural template to generate 3D structural model of the target RNA sequence via either a fully automated or script-based approaches. ModeRNA is capable of handling 115 different nucleotide modifications and bridging gaps using fragments derived from an extensive fragment library.



Picture 0. Basic functions of ModeRNA program.

Apart from that, ModeRNA offers a multitude of functions to examine and modify RNA structure files. ModeRNA can be integrated into other programs via Python scripting.

Table of Contents

STEP 1: TEMPLATE SEARCH

➔ The first step in homology modeling is to find an experimentally solved RNA structure, for which we have strong reason to believe that it shares a similar structure with the target molecule.

➔ The most obvious candidate with a potentially similar structure would be some evolutionary relative (a homolog), as molecules derived from a common ancestor are believed to preserve their structure despite accumulation of mutations.

- On the server site go to **'*Submit*' -> '*Find template*'**. Enter title and sequence of a target. You can also provide your e-mail address if you wish to receive a notification when the job is finished. Then press the **'*Find template*'**. (the procedure by which templates are searched is described in the APPENDIX).

Results for the structures of the resolution better than 2.5 Å:

Rfam Family	PDB ID	Alingment	Identity	Similarity	.aln file	.pdb file
RF00005	1EHZ_A	>Target GCCGAUAUAGCUCAGDDGGDAGAGCAGCGCAUUCGUEAUGCGAAG7UCGUAGGTPCGACUCCUAUUAUCGGCACCA >1EHZ A GCGGAUUUALCUCAGDDGGGAGAGCRCCAGABU#AAYAP?UGGAG7UC?UGUGTPCG"UCCACAGAAUUCGCACCA	0.618	0.733		
RF00005	1EVV_A	>Target GCCGAUAUAGCUCAGDDGGDAGAGCAGCGCAUUCGUEAUGCGAAG7UCGUAGGTPCGACUCCUAUUAUCGGCACCA >1EVV A GCGGAUUUALCUCAGDDGGGAGAGCRCCAGABU#AAYAP?UGGAG7UC?UGUGTPCG"UCCACAGAAUUCGCACCA	0.618	0.733		
RF00005	1QTQ_B	>Target GCCGAUAUAGCUCAGDDGGDAGAGCAGCGCAUUCGUEAUGCGAAG7UCGUAGGTPCGACUCCUAUUAUCGGCACCA >1QTQ B -GGGGUAUCGCCAAGC-GGUAAGGCACCGGAUUCUGAUUCCGGCAUUCCGAGGUUCGAAUCCUCGUACCCCAGCCA	0.539	0.646		
RF00005	1J1U_B	>Target GCCGAUAUAGCUCAGD-DGGDAGAGCAGCGCAUUCGUEAUGCGAAG7UCGUAGGTPCGACUCCUAUUAUCGGCACCA >1J1U B CCGGCGGUAGUUCAGCCUGGUAGAACGGCGGACUGUAGAUCCGCAUGUCGCUGGUUCAAAUCCGGCCCGCCGG---A	0.5	0.638		
RF00005	1QU2_T	>Target GCCGAUAUAGCUCAGDDGGD-AGAGCAGCGCAUUCGUEAUGCGAAG7UCGUAGGTPCGACUCCUAUUAUCGGCACCA- >1QU2 T GGGCUUGUAGCUCAGGUGGUUAGAGCGCACCCCUGAUAAGGGUGAGGUCGGUGGUUCAAGUCCACUCAGGCCC---AC	0.5	0.617		
RF00005	1FFY_T	>Target GCCGAUAUAGCUCAGDDGGD-AGAGCAGCGCAUUCGUEAUGCGAAG7UCGUAGGTPCGACUCCUAUUAUCGGCACCA- >1FFY T GGGCUUGUAGCUCAGGUGGUUAGAGCGCACCCCUGAUAAGGGUGAGGUCGGUGGUUCAAGUCCACUCAGGCCC---AC	0.5	0.617		
RF00005	1C0A_B	>Target GCCGAUAUAGCUCAGDDGGD-AGAGCAGCGCAUUCGUEAUGCGAAG7UCGUAGGTPCGACUCCUAUUAUCGGCACCA >1C0A B GGAGCGG4AGUUCAGDCGGDDAGAAUACCUGCCUQUCACGCAGGGG7UCGCGGGTPCGAGUCCCGPCCGUUCCGCCA	0.494	0.632		

Figure 1. Output of '*Find template*'.

- After a while (it will take a few minutes, when the server is busy with other jobs) you will receive a table with structures that are possibly homologous to the target – Please note that the server returns only structures with a resolution better than 2.5 Å. (Figure 1)
- While waiting for the server to finish, continue with **EXCURSUS: Structure analysis.**
- Having a set of template candidates, you have to choose the best template (see Template selection rules).

EXCURSUS: STRUCTURE ANALYSIS WITH MODERNA

➔ This section shows the basic usage of the ModeRNA scripting interface: loading a structure, basic features of ModeRNA structures and structure analysis.

➔ All the commands of ModeRNA program with description can be found here: http://www.genesilico.pl/moderna/commands/ or in the APPENDIX

- Download the structure 1Y0Q from the PDB (www.pdb.org).
- Open a terminal.
- Write: **python** (or **ipython** if you are using an interactive environment).
- To load a structure use the following commands:

```
In [10]: from moderna import *

In [11]: structure = load_template('1Y0Q.pdb', 'A')
```

- Check the properties of ModeRNA structure object:

```
In [12]: dir(structure)
Out[12]:
['__doc__',
 '__getitem__',
 '__init__',
 '__iter__',
 '__len__',
 '__module__',
 '__nonzero__',
 '__repr__',
 '_create_moderna_residues',
 '_get_chain_from_struct',
 '_get_residues_in_region',
 '_get_struct_from_file',
 'add_residue',
 'are_residues_connected',
 'chain_name',
 'change_sequence',
 'check_letters_in_residue_numeration',
 'cmp_for_moderna_residues',
 'contains_pseudoknot',
 'find_residues_in_range',
 'find_residues_not_in_range',
 'first_resi',
 'fix_backbone',
 'fix_backbone_after_resi',
 'fix_backbone_before_resi',
 'fix_backbone_between_resis',
 'get_all_atoms',
 'get_base_pairs',
 'get_modified_residues',
 'get_region',
 'get_resi_after',
 'get_resi_before',
 'get_secstruc',
 'get_sequence',
 'get_structure',
 'index',
 'is_chain_continuous',
 'last_resi',
 'moderna_residues',
 'remove_empty_residues',
 'remove_residue',
 'remove_residues_from_range',
 'renumber_chain',
 'renumber_residue',
 'set_template_numeration',
 'sort_residues',
 'template_residues',
 'write_pdb_file',
 'write_secstruc']
```

- Analyse the structure (you can also use ***examine_structure***):

```
In [3]: get_sequence(structure)
Out[3]: GAGCCUUUAUACAGUAAUGUAUAUCGAAAAAUCCUCUAAUUCAGGGAACACCUA_AGGCAAUCCUGAGCUAAGCUCUUAGUAAUAAGAGAAAG
UGCAACGACUAUUCCGAUAGGAAGUAGGGUCAAGUGACUCGAAAUGGGGAUUACCCUUCUAGGGUAGUGAUAUAGUCUGAACAUAUAUGGAAACAUAUAGA
AGGAUAGGAGUAACGAACCUAUCCGUAACAUAAUUG_._._._._._.

In [4]: seq = get_sequence(structure)

In [5]: print seq
GAGCCUUUAUACAGUAAUGUAUAUCGAAAAAUCCUCUAAUUCAGGGAACACCUA_AGGCAAUCCUGAGCUAAGCUCUUAGUAAUAAGAGAAAGUGCAACGA
CUAUUCCGAUAGGAAGUAGGGUCAAGUGACUCGAAAUGGGGAUUACCCUUCUAGGGUAGUGAUAUAGUCUGAACAUAUAUGGAAACAUAUAGAAGGAUAGG
AGUAACGAACCUAUCCGUAACAUAAUUG_._._._._._.

In [6]: get_secstruc(structure)
Out[6]: '.......((((((....))))))........((((((....(((((......(...).....))))))(((...((((((....))))))..))
).........((((....))))...(((((....)))))).....))))))((((.....)))))..............((..((((((....)))))()..
.(((((((.........)))))))..........)).......'

In [7]: secstruc = get_secstruc(structure)

In [8]: print secstruc
.......((((((....))))))........((((((....(((((......(...).....))))))(((...((((((....))))))..))).........
..((((....))))...(((((....)))))).....))))))((((.....)))))..............((..((((((....)))))()...(((((((.
........)))))))..........)).......

In [9]: analyze_geometry(structure)
Out[9]:

Unusual dihedral angles:
- Residues 33 --- 34 (X:O3',X+1:P,X+1:O5',X+1:C5'): 354.52
- Residues 135 --- 134 (X:O3',X+1:P,X+1:O5',X+1:C5'): 353.88
- Residues 186 --- 187 (X:O3',X+1:P,X+1:O5',X+1:C5'): 347.35
- Residues 241 --- 242 (X:O3',X+1:P,X+1:O5',X+1:C5'): 337.10
- Residues 37 --- 36 (X:C4',X:C3',X:O3',X+1:P): 149.21
- Residues 119 --- 118 (X:C4',X:C3',X:O3',X+1:P): 97.24
- Residues 119 --- 120 (X:C4',X:C3',X:O3',X+1:P): 81.63
- Residues 121 --- 120 (X:C4',X:C3',X:O3',X+1:P): 89.06
- Residues 140 --- 139 (X:C4',X:C3',X:O3',X+1:P): 93.76
- Residues 148 --- 149 (X:C4',X:C3',X:O3',X+1:P): 148.71
- Residues 160 --- 159 (X:C4',X:C3',X:O3',X+1:P): 84.04
- Residues 173 --- 174 (X:C4',X:C3',X:O3',X+1:P): 74.20
- Residues 176 --- 175 (X:C4',X:C3',X:O3',X+1:P): 140.58
- Residues 184 --- 185 (X:C4',X:C3',X:O3',X+1:P): 348.09
- Residues 186 --- 185 (X:C4',X:C3',X:O3',X+1:P): 54.59
- Residues 188 --- 187 (X:C4',X:C3',X:O3',X+1:P): 78.28
- Residues 194 --- 195 (X:C4',X:C3',X:O3',X+1:P): 133.81
- Residues 217 --- 216 (X:C4',X:C3',X:O3',X+1:P): 23.39
- Residues 218 --- 217 (X:C4',X:C3',X:O3',X+1:P): 20.92
- Residues 247 --- 246 (X:C4',X:C3',X:O3',X+1:P): 68.86
- Residues 252 --- 251 (X:C4',X:C3',X:O3',X+1:P): 142.05
- Residue 159 (X:O5',X:C5',X:C4',X:C3'): 230.02
- Residue 176 (X:O5',X:C5',X:C4',X:C3'): 96.92
- Residue 249 (X:O5',X:C5',X:C4',X:C3'): 129.83
- Residues 158 --- 159 (X:C3',X:O3',X+1:P,X+1:O5'): 353.71
- Residues 195 --- 196 (X:C3',X:O3',X+1:P,X+1:O5'): 344.95
- Residues 205 --- 206 (X:C3',X:O3',X+1:P,X+1:O5'): 335.69
- Residue 36 (X:P,X:O5',X:C5',X:C4'): 292.20
```

STEP 2: TEMPLATE SELECTION

Template selection rules:

- ➔ Verify the identity and similarity of the target and the template sequences. The higher the better.
- ➔ Investigate the target-template alignment – (most of the sequences are aligned, not too many gaps). A high quality alignment has sequences aligned over the full length and only short gaps occur.
- ➔ Check the resolution of the template structure – the server provides you only the structures with a resolution better than 2.5 Å – a template selection rule here is the higher resolution, the more accurate template structure and in consequence a better starting point for modeling.
- ➔ Look at the Rfam family – it should be closely related to the target sequence, and check the template structure – its origin and function. It should have the same or similar function as the target.
- ➔ Check if the template structure contains any features that could cause problems during modeling such as water or ions – here you can use the ModeRNA server feature '***Analyse structure***' – e.g. water and ions will be visible as a '_._._._._' tail in the sequence. Eventually remove them.

- Select the best template bearing in mind the aforementioned rules; download the structure and the alignment files from the server results page.
- Now using the ModeRNA program installed on your computer analyse the template that you have chosen – check the geometry, sequence and secondary structure (see the EXCURSUS) .

  You can also analyse the structure using ModeRNA server: go to '***Submit*' -> '*Analyse structure***'. Here use as an input the downloaded structure file, insert a title and chain id (the last letter or number in the name of the downloaded file) and check the boxes: '***get sequence***', '***get secondary structure***' and '***analyse geometry***' to find out if there are any problems with the structure → press '***Analyse structure***' (the output would look as in Figure 2).
- If there were any characters such as '_._._' (it means that water ions are present in the structure or there are breaks in a chain) in a sequence – it means that you have to clean the structure with the '***clean structure***' feature and run the analysis again.

```
In [26]: clean_structure(structure, write_structure = True)
Out[26]: Chain OK
```

- Analyse the output (secondary structure and geometry). Are there any problems with the structure such as unusual bonds, unusual dihedral angles or

unusual secondary structure? If yes, select another template.

- You could analyse the template structure even with more details – an example analysis is shown in Figure 3.

Structure after analysis:
[tutorial_1.pdb]

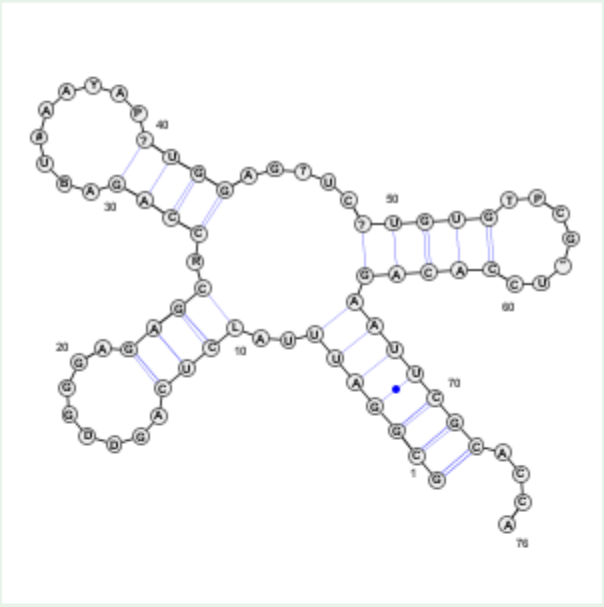
Sequence of a given structure:

```
GCGGAUUUALCUCAGDDGGGAGAGCRCCAGABU#AAYAP?UGGAGUC?UG
UGTPCG"UCCACAGAAUUCGCACCA
```

Download: **[.fasta file with the sequence]**

Secondary structure of a given structure:

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



Download: **[.vienna file with the secondary structure]**

Geometry of the structure:

```
Unusual dihedral angles:
- Residues 15 --- 16 (X:O3',X+1:P,X+1:O5',X+1:C5'): 353.87
- Residue 14 (X:O5',X:C5',X:C4',X:C3'): 245.41
```

Figure 2. '*Analyse structure*' output. 1EHZ possesses the cloverleaf structure typical for tRNA. In the tertiary structure some uncommon values of dihedral angles were found in residues 14-16. These residues are located in the DHU loop, which has a flexible conformation due to the presence of two dihydrouridines. According to observations, a few nonstandard dihedral angles can be found in many PDB structures, and especially in this position they are not unusual.

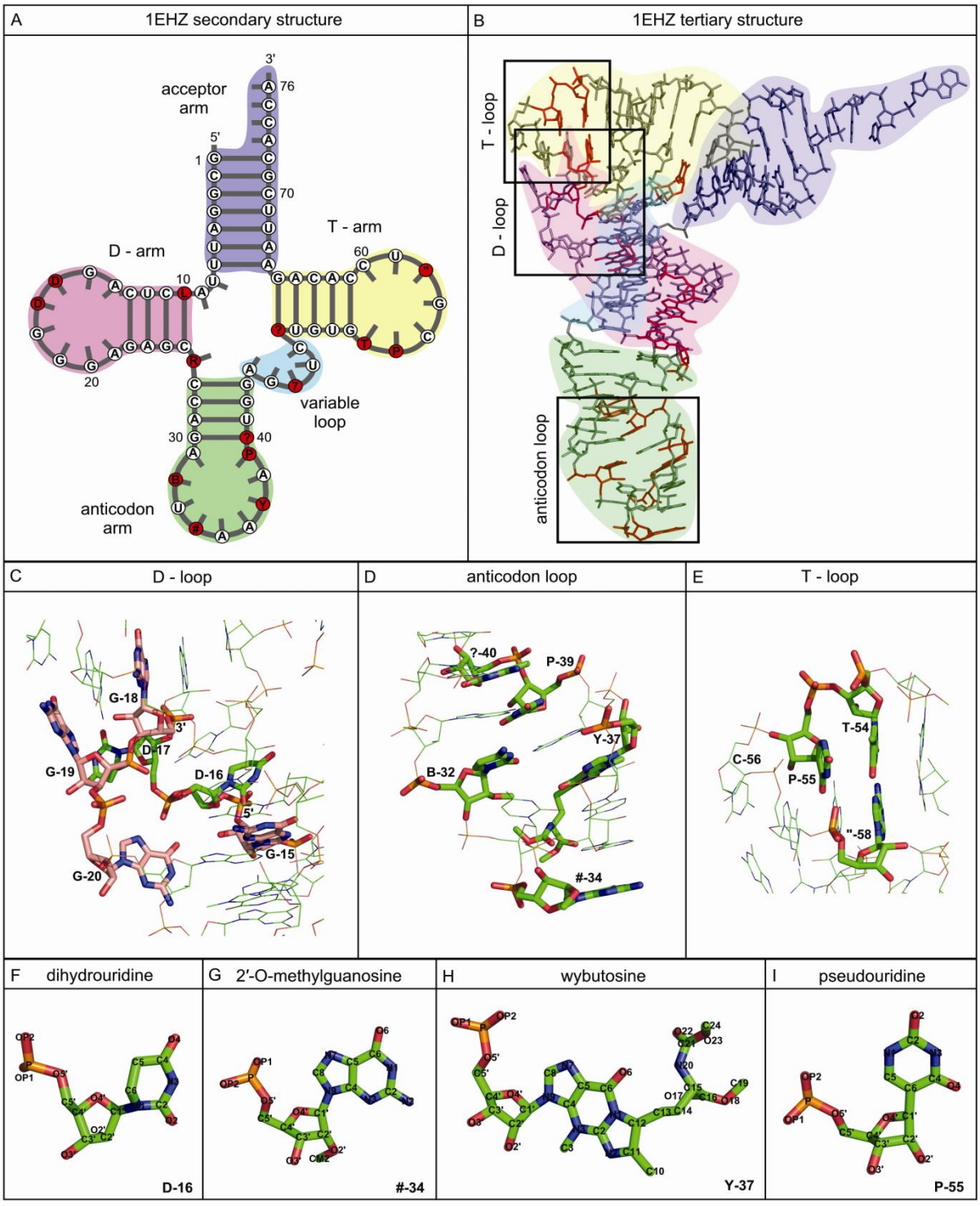


Figure 3. Structure of the template 1EHZ (E. coli tRNA(Phe)); A) secondary structure; B) tertiary structure; colours in A and B are corresponding, modified nucleotides are marked in red; C) D-loop; D) anticodon loop; E) T-loop; in D and E modified nucleotides are shown as sticks; F-I) examples of modified nucleotides occurring in 1EHZ: F) dihydrouridine; G) 2′-O-methylguanosine; H) wybutosine; I) pseudouridine; in all panels a one-letter nomenclature is applied for standard and modified residues.

STEP 3: TARGET -TEMPLATE ALIGNMENT

➔ In the next step we need to generate a high quality alignment of the template and the target, i.e. have a high number of confidently identifiable homologous residues between the target and template. This typically requires a high sequence identity, and a low number of indels.

- You have the template pdb file and the target sequence – you also have the automatically generated alignment file, from the ***'Find template'*** routine performed on the ModeRNAserver.
- In case you want to prepare your own alignment please take advantage of the ***'Prepare alignment'*** feature available at the server (***'Submit' -> 'Prepare alignment'***). In the '***Find template***' procedure, the server uses Infernal with covariance models taken from RFAM database, and in the **'Prepare alignment'** procedure the server uses the RCoffee program. Alternatively, you can use a program for aligning the sequences of target and template by yourself, e.g. Clustal: http://www.clustal.org/, RCoffee http://www.tcoffee.org/Projects_home_page/r_coffee_home_page.html or Infernal: http://infernal.janelia.org/.
- To investigate your alignment in detail, open it with a text editor.
- If you have chosen 1EHZ as the template - the alignment returned by the server, you should see that it contains no insertions or deletions.
- In Figure 4 you can see an alternative alignment of the target sequence, and two templates, 1EHZ and 1C0A (the structure that is on the seventh position in the ***'Find template'*** result).

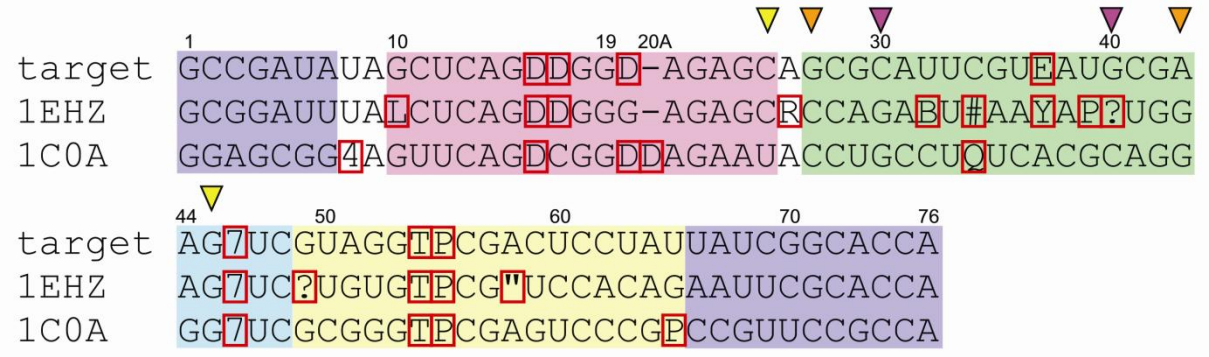


Figure 4. Alignment between the target sequence (*E. coli* tRNA$^{(Thr)}$), the template sequence from structure 1EHZ, chain A (*S. cerevisiae* tRNA$^{(Phe)}$), and the template sequence from structure 1C0A,chain B (*E. coli* tRNA$^{(Asp)}$). In the latter template one additional residue is present and is numbered as 20A according to the tRNA sequence nomenclature. The shading of the alignment corresponds to the particular parts of tRNA: acceptor arm (dark blue); D-loop (purple); anticodon arm (green); variable loop (cyan); T-loop (yellow); modified nucleotides were highlighted with red frames. The triangles over the alignment indicate inserted fragments at the AC loop of length 9 (purple), 15(orange), and 19 (yellow).

STEP 4: MODEL BUILDING

Modeling steps:

During modeling, ModeRNA analyses the alignment and assigns particular operations for each position.

- ➔ In cases where residues were identical, the atomic coordinates are directly copied from the template.
- ➔ In case of substitutions, backbone and ribose atoms are copied and the proper base is added according to superposition of three atoms closest to the glycosidic bond.
- ➔ For each insertion and deletion, ModeRNA searches for a fragment in a library created from a non-redundant set of RNA structures, based on the geometry of residues to be connected (so called “anchor residues”).
- ➔ In case a fragment did not fit ideally between the anchors, the backbone coordinates were optimized using the FCCD Loop Closer algorithm, which closes small breaks in the backbone.

- Prepare a model by loading the template structure (cleaned and analysed) and the alignment file.

```
In [1]: from moderna import *

In [2]: t = load_template('1EHZ_A.pdb', 'A')

In [3]: a = load_alignment('1EHZ A.aln')

In [4]: match_template_with_alignment(t,a)
Out[4]: True

In [5]: m = create_model(t,a)

In [6]: write_model(m, 'model.pdb')
```

- You can also prepare a model with the server feature **‘*Build model*’ (‘*Submit*’ -> ’*Build model*’**). Here you have to submit a template file with the appropriate chain ID. If you have chosen 1EHZ, the chain ID is A. You also have to provide an alignment file – the one you have downloaded together with the template structure. you can also check the boxes: ***clean structure*** and ***analyse geometry*** and click the **‘*Build model*’** button.

- Analyzing the alignment, we can see that in this particular modelling exercise the alignment is 76 positions long, with 47 identical target-template residue pairs. The 29 remaining positions contain mismatches. We have no insertions or deletions. In some of these positions, modified nucleosides occur (**CHECK** the procedure of modeling modifications described and the operations made by ModeRNA in Table 1).

Modeling of modified nucleosides

- ➔ The target sequence and template structure contain modified nucleosides (many tRNA structures in the PDB are however modeled without modifications).
- ➔ The structure 1EHZ contains methylations (e.g. 2′-O-methylguanosine in position 34, Picture 3G) and more complicated chemical groups (e.g. wybutosine in position 37, Picture 3H). Dihydrouridine (residue 16 in 1EHZ, Picture 3F) is a unique modification, as it possesses a non-planar ring. It is important to note that the modification pattern between the target and the template differs (see Table 1).
- ➔ In cases where modifications from the template match those from the target, they are simply copied in the same manner as unmodified residues (e.g. the dihydrouridines in position 16 and 17).
- ➔ Some modifications need to be changed into unmodified nucleosides (e.g. position 10). In such cases, the unmodified base is introduced by superposition of the three atoms nearest to the glycosidic bond onto the modified base to be replaced.
- ➔ In the opposite situation, i.e. when an unmodified residue need to be changed into a modified one (e.g. in position 20), one or more structural fragments containing the additional chemical groups are added to the base or ribose.
- ➔ When one modified residue needs to be replaced by another modified residue (e.g. in position 37), the first operation introduces a new unmodified residue, followed by addition of the new modification.
- ➔ ModeRNA contains a set of 70 structural fragments that enable building 115 known modifications. Addition of the fragments is guided by a set of rules describing the atom triplets used for superposition and atoms to be added and removed.
- ➔ In order to add dihydrouridine, the entire base needs to be exchanged due to the non-planarity of the partially saturated ring. The same applies to pseudouridine, because the base ring needs to be rotated and connected with ribose via the C6 atom instead of N1.
- ➔ ModeRNA automatically identifies and executes the proper rules for adding these modifications.
- ➔ ModeRNA uses the modification nomenclature implemented in the MODOMICS database (http://modomics.genesilico.pl). For each modification it stores a full name, a one-letter and a few-letter abbreviation, a common PDB residue name, and a numerical code. In the alignment, the one-letter abbreviations are used, if available. Because there are more modifications than reasonably usable ASCII characters, the numerical code can be used alternatively, e.g. 001U for pseudouridine.

Residue number	Template (1EHZ) residue	Target residue	ModeRNA operation
10	N2-methylguanosine (m2G, L, 2G)	guanosine (G)	remove modification
16	dihydrouridine (D, 6U)	dihydrouridine (D, 6U)	copy modified nucleotide
17	dihydrouridine (D, 6U)	dihydrouridine (D, 6U)	copy modified nucleotide
20	guanosine (G)	dihydrouridine (D, 6U)	replace G with U, add modification
26	N2,N2-dimethylguanosine (m22G, R, 3G)	adenosine (A)	remove modification, replace G with A
32	2′-O-methylcitidine (Cm, B, 0C)	uridine (U)	remove modification, replace C with U
34	2′-O-methylguanosine (Gm, #, 0G)	cytidine (C)	remove modification, replace G with C
37	wybutosine (yW, Y, 16G)	N6-methyl-N6-threnylcarbamoyladenosine (m6t6A, E, 15A)	remove modification, replace A with G, add modification
39	pseudouridine (Y, P, 1U)	uridine (U)	remove modification
40	5-methylcytidine (m5C, ?, 5C)	guanosine (G)	remove modification, replace C with G
46	7-methylguanosine (m7G, 7, 7G)	7-methylguanosine (m7G, 7, 7G)	copy modified nucleotide
49	5-methylcytidine (m5C, ?, 5C)	guanosine (G)	remove modification, replace C with G
54	5-methyluridine (m5U, T, 5U)	5-methyluridine (m5U, T, 5U)	copy modified nucleotide
55	pseudouridine (Y, P, 1U)	pseudouridine (Y, P, 1U)	copy modified nucleotide
58	1-methyladenosine (m1A, “, 1A)	adenosine (A)	remove modification

Table 1. Operations made by ModeRNA during the modelling of our target using 1EHZ template.

- The results of the modelling can be seen in Figure 5. You can visualize your model in SwissPDBViewer or PyMOL.

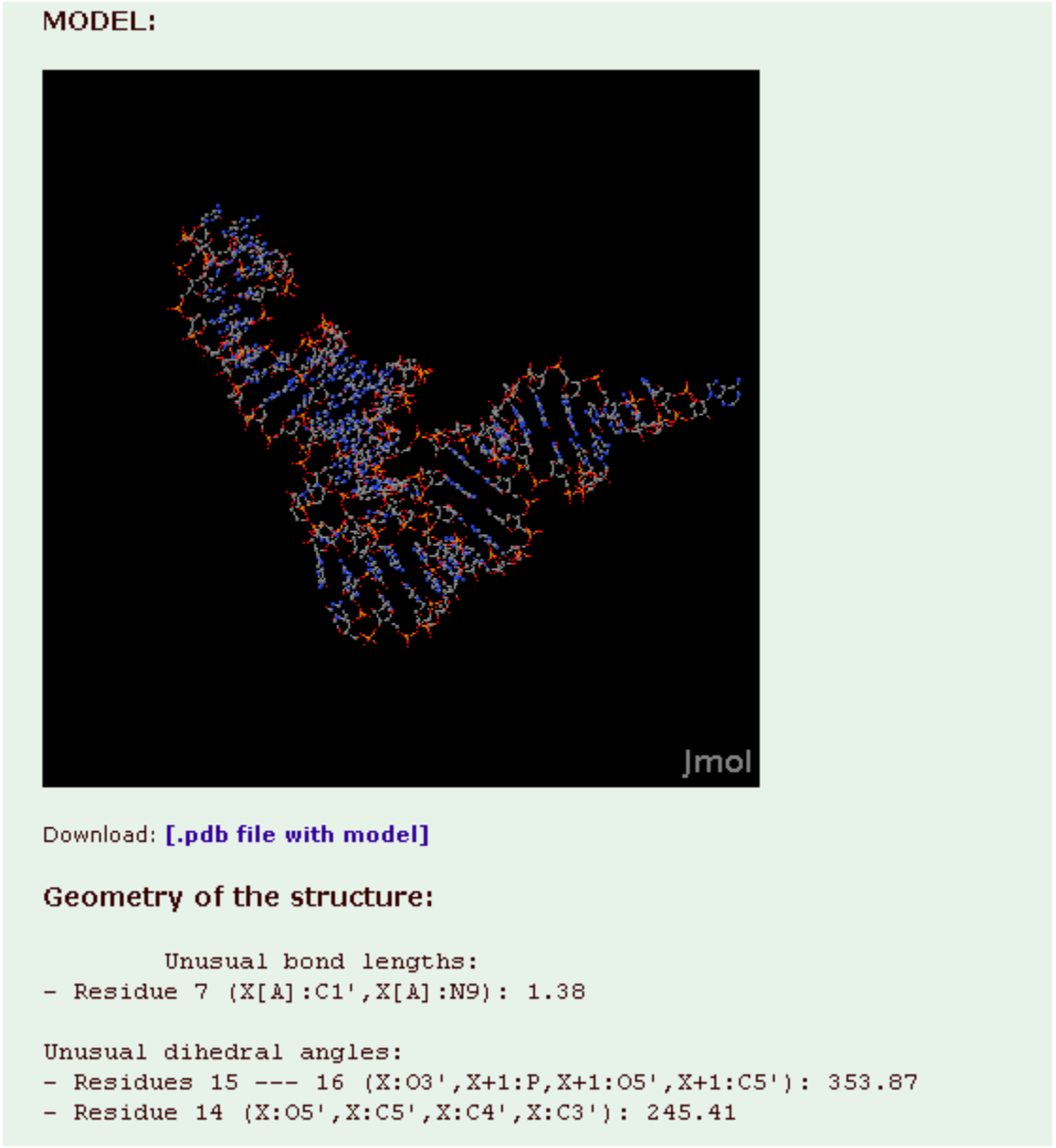


Figure 5. Result of the modelling.

- After building the model one should evaluate it, because there are many errors which could occur in the model. We are now going to check if the model is correct and reasonable.

STEP 5: EVALUATION OF THE MODEL

Benchmarks of structural similarity between the model and original structure.

- ➔ **The root mean square deviation (RMSD)** is the oldest measure of similarity between two 3D structures, and is the one most frequently found in the literature. Generally, the smaller the RMSD, the more similar the compared structures are (with 0.0 corresponding to identity), however this relationship may not hold for structures with very different sizes, or those exhibiting conformational changes. It can be calculated in two ways: using all heavy atoms (all-atom RMSD) or only two backbone atoms (P and C4′ RMSD).
- ➔ **The template modeling (TM)** score is a normalized measure of the overall structure similarity that does not depend on the structure size. It ranges from 1.0 (identity) to 0.0 (no similarity) and follows an extreme value distribution; for protein structures values above 0.5 typically indicate similar structures, while values below 0.5 are characteristic of dissimilar structures.
- ➔ **Global Distance Test – Total Score (GDT-TS)** is another measure whose value ranges between 0.0 (no identity) to 1.0 (100% identity) and indicates the number of corresponding atom pairs between the compared structures found below four different distance thresholds: 1Å, 2Å, 4Å and 8Å divided by four times the total number of atoms.
- ➔ **The DI (deformation index)** indicates how well the base pairing and stacking interactions were modeled, with 0.0 being the ideal value.
- ➔ **DP (deformation profile metrics)** highlights the dissimilarity between two structures and is calculated by superimposing each pair of corresponding residues and generating a matrix of per-residue RMSD values (hence, the lower the DP, the more similar the structures).

- To evaluate the model we need to investigate such structural features as unusual geometry and interatomic clashes (we can see it in the result of the modelling because we have checked the box ***'analyse geometry'***) - the test revealed that the model exhibited minor geometrical problems.
- Open the model you have obtained in PyMOL and check results retrieved for model evaluation. We can see that none of the problems is very serious, moreover the model did not accumulate a particularly high level of errors during modelling.
- As the structure of *E. coli* $tRNA^{Thr}$ interacting with its cognate aaRS enzyme has been experimentally solved and is available as a PDB entry 1QF6 (chain B), we were able to assess the real accuracy of our models. To do so, we can apply six different benchmarks of structural similarity (Table 2).

- Download the crystal structure of 1QF6 from www.pdb.org.
- In PyMOL, open 1QF6 – as you can see there is RNA and protein molecule there – hide the protein using command "***sele chain chain_id***" (eg. "***sele chain A***") → chain ID of the protein can be found in the PDB record of 1QF6 or clicking in PyMOL on the right site near 1QF6 on ***L*** (as Label) and ***chain***. After selecting proper chain click near (sele) ***A*** (as Action) → ***extract object*** (now you have new protein object). Near obj01 click ***A*** → ***delete object*** and near (sele) ***A*** → ***delete selection***. Now you can see only the RNA molecule.
- Now load your model and the template we used in modelling procedure and the original crystal structure of $tRNA^{Thr}$ from *E. coli*. (***File*** -> ***Open***).
- Use "***align***" command to superimpose these structures → "***align name_of_molecule_1, name_of_molecule_2.***" (e.g. "***align 1QF6, test***" → detailed description of the command can be found here: http://www.pymolwiki.org/index.php/Align). We can restrict the alignment only to one cycle – to do that add to command after ',' '***cycles=0***' → in that way we can see the all atoms RMSD value.
- For each pair of structures check the RMSD value (by aligning them → you can see the RMSD value in the terminal). Investigate the differences between the original crystal structure, the model from ModeRNA and the template used for homology modelling.
- **Answer the question:** is the RMSD value similar to the one in the Table 2? Is the model correct? Why? Why not?

Benchmark	Interatomic clashes	Unusual bond lengths	Unusual bond angles	Unusual dihedral angles	**All-atom RMSD**	**P, C4′ RMSD**	**GDT-TS score**	**DI**	**Average DP**	**TM score**
Model based on 1EHZ	1	2	0	2	**5.07**	**4.37**	**0.55**	**0.82**	**12.70**	**0.56**

Table 2. Geometry analyses and benchmark values of the model based on the template 1EHZ.

STEP 6: ADVANCED MODELING OF AN ANTICODON LOOP

Other options of ModeRNA.

Automatic model building is the simplest way to obtain a model. However, in some cases it is not sufficient and some additional editing is required. ModeRNA provides many commands that enable changes on different levels: the entire molecule, a particular region, and a single residue.

- A model was built based on the template 1EHZ. This is an RNA structure in a state not bound to the protein. The differences between the model and original structure you observed originated from the fact that the native structure is in the state bound to the protein (the conformation of the anticodon loop is different from the unbound one).
- We would like to model a loop in a state bound with the protein. In the results of the '***Find template***' procedure on the web server, find a template with a reasonable similarity and identity bound to a protein.
- We will be using the **1C0A** structure as a template for the anticodon loop. Open the structures: 1C0A, your model and 1EHZ with the program of your choice (chimera, PyMOL), prepare a structural alignment and analyse the anticodon loop. How many residues from 1C0A should be used to remodel the anticodon loop of the model?

- We will try 3 different options (we have to add the two terminal residues of each fragment as anchor residues for superposition with the corresponding residues from the model):
  1) Just the anticodon loop - 9 residues (G-630 to C-640 (GCCUQUCACGC))
  2) Almost the entire anticodon arm plus one terminal base pair – 15 residues (C-627 to G-643 (CCUGCCUQUCACGCAGG)).
  3) A fragment with the anticodon arm and a few additional nucleotides from an adjacent helical stem – 19 residues (U-625 to G-645 (UACCUGCCUQUCACGCAGGGG)).

- Cut out parts of 1C0A and save them as separate PDB files – to extract residues from 1C0A:
  - load structure (***load_model*).**
  - check the residue numbers **(*_get_residues_in_region***) – pay attention to the residue number 620 → it is labelled 620A, and at the residues 676, and 701 – there are no other residues between those two.

```
In [37]: m = load_model('1C0A.pdb', 'B')

In [38]: m._get_residues_in_region()
Out[38]:
[<Residue 601 G>,
 <Residue 602 G>,
 <Residue 603 A>,
 <Residue 604 G>,
 <Residue 605 C>,
 <Residue 606 G>,
 <Residue 607 G>,
 <Residue 608 s4U>,
 <Residue 609 A>,
 <Residue 610 G>,
 <Residue 611 U>,
 <Residue 612 U>,
 <Residue 613 C>,
 <Residue 614 A>,
 <Residue 615 G>,
 <Residue 616 D>,
 <Residue 617 C>,
 <Residue 618 G>,
 <Residue 619 G>,
 <Residue 620 D>,
 <Residue 620A D>,
```

  - Check the sequence before deleting residues (***get_sequence***).
  - Delete residues that are unwanted (***delete_residue***) – together with residue 620A and 701.
  - Check the sequence after deletion (***get_sequence***).
  - If the sequence is the same as given above – clean the fragment (***clean_structure***) and save the structure (***write_model***).
  - Repeat the procedure for the three fragments listed above.

```
In [149]: t_1COA = load_model('1C0A.pdb', 'B')

In [150]: get_sequence(t_1COA)
Out[150]: GGAGCGG4AGUUCAGDCGGDDAGAAUACCUGCCUQUCACGCAGGGG7UCGCGGGTPCGAGUCCCGPCCGUUCCGCCA_._._._._
._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._
._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._
._._._._._._._.

In [151]: for residue in range(601, 630):
   .....:         delete_residue(str(residue), t_1COA)
   .....:

In [152]: for residue in range(641, 677):
   .....:         delete_residue(str(residue), t_1COA)
   .....:

In [153]: delete_residue('620A', t_1COA)

In [154]: delete_residue('701', t_1COA)

In [155]: get_sequence(t_1COA)
Out[155]: GCCUQUCACGC_._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._
._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._
._._._._._._._._._._._._._._._._._._._._._.

In [156]: clean_structure(t_1COA)
Out[156]: Chain OK

In [157]: get_sequence(t_1COA)
Out[157]: GCCUQUCACGC

In [158]: write_model(t_1COA, 'fragment_9_residues.pdb')
```

```
In [159]: t_1COA = load_model('1C0A.pdb', 'B')

In [160]: get_sequence(t_1COA)
Out[160]: GGAGCGG4AGUUCAGDCGGDDAGAAUACCUGCCUQUCACGCAGGGG7UCGCGGGTPCGAGUCCCGPCCGUUCCGCCA_._._._._
._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._
._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._
._._._._._._._.

In [161]: for residue in range(601, 627):
   .....:         delete_residue(str(residue), t_1COA)
   .....:

In [162]: for residue in range(644, 677):
   .....:         delete_residue(str(residue), t_1COA)
   .....:

In [163]: delete_residue('620A', t_1COA)

In [164]: delete_residue('701', t_1COA)

In [165]: get_sequence(t_1COA)
Out[165]: CCUGCCUQUCACGCAGG_._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._
._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._
._._._._._._._._._._._._._._._._._._._._._._._.

In [166]: clean_structure(t_1COA)
Out[166]: Chain OK

In [167]: get_sequence(t_1COA)
Out[167]: CCUGCCUQUCACGCAGG

In [168]: write_model(t_1COA, 'fragment_15_residues.pdb')
```

```
In [169]: t_1COA = load_model('1C0A.pdb', 'B')

In [170]: get_sequence(t_1COA)
Out[170]: GGAGCGG4AGUUCAGDCGGDDAGAAUACCUGCCUQUCACGCAGGGG7UCGCGGGTPCGAGUCCCGPCCGUUCCGCCA_._._._._._.
_._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._
._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._.

In [171]: for residue in range(601, 625):
   .....:         delete_residue(str(residue), t_1COA)
   .....:

In [172]: for residue in range(646, 677):
   .....:         delete_residue(str(residue), t_1COA)
   .....:

In [173]: delete_residue('620A', t_1COA)

In [174]: delete_residue('701', t_1COA)

In [175]: get_sequence(t_1COA)
Out[175]: UACCUGCCUQUCACGCAGGGG_._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._.
_._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._._
._._._._._._._._._._._._._._._._._._._._._.

In [176]: clean_structure(t_1COA)
Out[176]: Chain OK

In [177]: get_sequence(t_1COA)
Out[177]: UACCUGCCUQUCACGCAGGGG

In [178]: write_model(t_1COA, 'fragment_19_residues.pdb')
```

- Use the prepared PDB files to create fragment objects (***create_fragment***):

- Load a model that you have prepared earlier on the template 1EHZ (***load_model***) and check the properties of the model (numbers of residues and a sequence) – you will need it to prepare fragments with the anchors.

```
In [94]: m_1EHZ = load_model('model.pdb', 'A')

In [95]: get_sequence(m_1EHZ)
Out[95]: GCCGAUAUAGCUCAGDDGGDAGAGCAGCGCAUUCGUEAUGCGAAG7UCGUAGGTPCGACUCCUAUUAUCGGCACCA

In [96]: m_1EHZ._get_residues_in_region()
Out[96]:
[<Residue 1 G>,
 <Residue 2 C>,
 <Residue 3 C>,
 <Residue 4 G>,
 <Residue 5 A>,
 <Residue 6 U>,
 <Residue 7 A>,
 <Residue 8 U>,
 <Residue 9 A>,
 <Residue 10 G>,
 <Residue 11 C>,
 <Residue 12 U>,
 <Residue 13 C>,
 <Residue 14 A>,
```

- Create fragments for the prepared PDB files (***create_fragment***) - specify two anchor residues from the model to guide the insertion and a new sequence for the anticodon fragment (sequence of the fragment to be modeled should be the same as in the target – so the sequence is taken from the model and then modeled into it with the usage of specified fragment) – for the anchor position check in PyMOL which positions of the model correspond to 1C0A residues depicted above.
- Insert the fragments into the loaded model (***insert_fragment***).
- Save the built models (***write_model***).

```
In [194]: m_1EHZ = load_model('model.pdb', 'A')

In [195]: fragment_9 = create_fragment('fragment_9_residues.pdb', m_1EHZ['30'], m_1EHZ['40'], chain_n
ame='B', sequence='AUUCGUEAU')

In [196]: insert_fragment(m_1EHZ, fragment_9)

In [197]: write_model(m_1EHZ, 'model_1EHZ_insertion_9nt.pdb')
```

```
In [190]: m_1EHZ = load_model('model.pdb', 'A')

In [191]: fragment_15 = create_fragment('fragment_15_residues.pdb', m_1EHZ['27'], m_1EHZ['43'], chain
_name='B', sequence='CGCAUUCGUEAUGCG')

In [192]: insert_fragment(m_1EHZ, fragment_15)

In [193]: write_model(m_1EHZ, 'model_1EHZ_insertion_15nt.pdb')
```

```
In [182]: m_1EHZ = load_model('model.pdb', 'A')

In [183]: fragment_19 = create_fragment('fragment_19_residues.pdb', m_1EHZ['25'], m_1EHZ['45'], chain
_name='B', sequence='AGCGCAUUCGUEAUGCGAA')

In [184]: insert_fragment(m_1EHZ, fragment_19)

In [185]: write_model(m_1EHZ, 'model_1EHZ_insertion_19nt.pdb')
```

Inserting fragments – How ModeRNA does it

➔ During the insertion process the fragments are superimposed using the anchor residues from fragment and model (the fragment is mobile and the model stays in a fixed position). In particular the atoms O3′, C3′, C4′, C1′, and N1 or N9 on the 5′-end and C5′, C4′, C3′, C1′, N1 or N9, and O5′ on 3′-end are used for superposition. All residues present in the model between the anchors specified during fragment creation are removed and new residues from the fragment are added.

➔ The insertion of a fragment with a non-ideal match with the anchor residues can result in minor gaps in the backbone of the model. You can fix it using function ***fix_backbone***.

- Analyse all the models you have created with PyMOL as described above (align each model with the original structure 1QF6) and analyse using ModeRNA the geometry of each model. What is the RMSD now? Which model is the best? Check with the values in Table 3.

Benchmark	Model based on 1EHZ	Model based on 1C0A	Model based on 1EHZ + 9nt loop	Model based on 1EHZ + 15nt loop	Model based on 1EHZ + 19nt loop
Interatomic clashes	**1**	2	**1**	**1**	**1**
Unusual bond lengths	**2**	7	3	4	3
Unusual bond angles	**0**	3	1	1	2
Unusual dihedral angles	**2**	3	4	**2**	**2**
All-atom RMSD	5.07	**3.38**	4.33	4.01	3.78

Table 3. Geometry analysis and benchmark values of all the models.

After modeling

The further use of RNA 3D structure models, e.g. to model interactions with other molecules, requires the use of other bioinformatics tools.

➔ The next step could require the acquisition of a model of the protein partner in the appropriate functional state, which can be achieved by an analogous modeling protocol, with protein-specific tools such as SwissModel or Modeller.

➔ The assembly of a complex can be guided by homology (e.g. by superposition onto another related complex) or *de novo*, by protein-RNA docking, e.g. with HADDOCK. Protein-RNA docking is an emerging field, and currently no standard protocols exist, especially for analyzing RNA molecules with modified residues. For instance HADDOCK cannot automatically process modified residues, and such analysis would require “demodification” of the RNA with ModeRNA prior to docking.

APPENDIX

analyze_geometry(struc, file_name=None)

Analyses geometry of a structure. Checks whether all angles and bonds length values are close to reference values determined from the PDB. Writes the result of the analysis to log file or to an output file when specified.

Arguments:

RNAModel object

Name of an output file

```
t = load_template('1QF6_B_tRNA.pdb', 'B')
analyze_geometry(t)
```

clean_structure(structure, write_structure=False)

Eliminates features that may cause problems during modeling from a template or model structure:
water molecules, ions, amino acids, and unidentified residues are deleted.
old atom names are replaced (C1* becomes C1', O1P becomes OP1).
missing phosphate groups are added (also adds the OP3 atom for these)
reports whether the chain is continuous.
In case some feature cannot be fixed (e.g. chain discontinuity) this is written to the logfile. It is recommended to include such features in the alignment ('.' characters for strange residues, and '_' for backbone breaks).

Arguments:

Stucture object (RnaModel or Template)

True/False - whether structure should be written to a PDB file (optional)

```
# cleaning up a loaded template:
# removes water, ions, amino acids, and unknown residues
# replaces O1P and O2P in atom names by OP1 and OP2
# replaces * in atom names by '
clean_structure(t)
```

create_fragment(pdb_path, anchor5=None, anchor3=None, chain_name='A', sequence=None)

To insert small pieces of custom PDB structures into models, fragments can be used. This command loads a fragment, and defines one or two connection points, to which the fragment will be inserted by superposition. It returns a ModernaFragment object.
To add a ModernaFragment object to a model, the insert_fragment command should be used. The scenarios for adding either involve superposition of a single residue with the model on the 5' end of the fragment, or on the 3' end, or both.

Arguments:

path to pdb file with the fragment

residue to which the 5' end of the fragment will be superimposed

residue to which the 3' end of the fragment will be superimposed

chain name of the fragment in the file (optional; A by default)

sequence that should be modeled onto the fragment upon insertion (optional; the sequence should not include the 1-2 anchor residues, it therefore has to be shorter than the fragment)

```
f = create_fragment('single_strand.pdb',anchor5=m['20'])
f = create_fragment('single_strand.pdb', chain_name='A', anchor3=m['1'])
f = create_fragment('hairpin.pdb', anchor5=m['12'], anchor3=m['15'])
f = create_fragment('hairpin.pdb', anchor5=m['12'], anchor3=m['15'],
sequence='AG')
```

create_model

create_model(template=None, alignment=None, model_chain_name=None)

Creates a RNA structure model. Produces a RnaModel object that can be saved in a variable (see example).

If no arguments are given, an empty RNAModel is created. If both a Template and an Alignment are given as arguments, the complete model is built automatically. The chain id that the model should have can be specified optionally.

Arguments:

Template object (optional)

Alignment object (optional)
chain id of the model to be built

```
m = create_model()
m = create_model(model_chain_name='K')
m = create_model(t,a)
m = create_model(t,a,model_chain_name='K')
```

delete_residue(residue_number, model)

Removes a residue from a RNAModel object.
Arguments:
residue identifier e.g. '1', '6', '6A'
RnaModel object

```
delete_residue('6',m)
```

examine_structure(structure)

Checks whether the given structure has any features that may cause any problems during the modeling process. The user needs to give a structure object, and optionally a name of the file the report is written to.
Arguments:
Stucture object
name of logfile (optional)

```
# examine a loaded template for irregularities:
examine_structure(t)
examine_structure(t,'logfile.log')
```

fix_backbone(structure)

Fixes interrupted backbones between adjacent residues.
This function either examines the backbone of all nt in a structure (and repairs them if broken), or only does this for the two residues specified. For repairing, first only the phosphate and adjacent oxygens are remodeled. In case this fails, the FCCD Loop Closing Algorithm (Boomsma/Hamelryck 2005) is used for the entire O3'..C4' segment. If the two residues are far from each other or oriented in a weird angle, the result will be awkward.
Arguments:
RnaModel object
residue number on the 5' side of the broken backbone (optional).
residue number on the 3' side of the broken backbone (optional).

```
m = load_model('broken_bb.pdb','A')
print m.get_sequence()
# check and fix the entire model
fix_backbone(m)
print m.get_sequence()
# check and fix the connection between residues 4 and 5
fix_backbone(m, '4', '5')
```

get_secstruc(structure)

Retrieves the dot-bracket secondary structure from the coordinates of a structure (template or model). Base pairs (only Watson-Crick pairings AU and GC, and GU Wobble pairs) are indicated by round brackets, all other residues by dots.
Arguments:
structure - a Template or RNAModel object

```
get_secstruc(t)
ss = get_secstruc(m)
```

get_sequence(structure)

Retrieves the one-letter-sequence from the coordinates of a structure (template or model). In the sequence, standard RNA bases are denoted by upper case letters, DNA bases by lowercase. For many modifications, one-letter ASCII abbreviations exist (according to McCloskey). All nucleotides that

do not have a one-letter abbreviation are represented by the x letter. Nucleotides that cannot be recognized (e.g. base missing) are represented by the '.' character.
For determining the sequence, the residues are processed according to their numbers. If an unusually long bond is found anywhere in the backbone between two bases, an additional "_" symbol is inserted in the sequence to mark this discontinuity.
Also see http://www.genesilico.pl/modomics
Arguments:
structure - a Template or RnaModel object

```
get_sequence(t)
seq = get_sequence(m)
```

insert_fragment(model, fragment)

Inserts a fragment object into the model. The model should be the same object, from which the two anchor residues were defined when the fragment was created. (See also documentation of the create_fragment/find_fragment functions.)
Arguments:
RnaModel object
ModernaFragment or LirHit object

```
insert_fragment(m,f)
```

load_alignment(file_path)

Loads a sequence alignment from a FASTA file. Produces an Alignment object that can be saved in a variable.
ModeRNA expects, that the alignment file contains exactly two sequences. The first is for the target for which the model is to be built, the second for the structural template.
Standard RNA bases should be written in upper case (ACGU). Standard DNA bases should be written in lower case (acgt). For modified bases, see the 'concepts' section of the manual.
Arguments:
path+filename of a FASTA file

```
a = load_alignment('alignment_1F1T.fasta')
```

load_model(file_path, chain_name='A', data_type='file', template=None, alignment=None)

Loads a structure model that has been built previously, or any PDB struture that is then to be modified. Produces a RnaModel object that can be saved in a variable. Each model in ModeRNA contains only one chain. Multi-chain structures can be modeled by using more than one Template/Alignment/RNAModel at a time.
By default, RNAModels are created by reading files. They can also be created from BioPython PDB objects (precisely Bio.PDB.Structure.Structure and Bio.PDB.Chain.Chain objects)
Arguments:
path+filename of a PDB structure file; or a Structure or Chain object from BioPython.
chain id (by default 'A')
data type ('file' by default; if set to 'structure' or 'chain', Bio.PDB objects are read)
Template object to be used for this model (optional)
Alignment object to be used for this model (optional)

```
m = load_model('1F1T.pdb')
m = load_model('1F1T.pdb','A')
m = load_model(biopy_struc, data_type='structure')
m = load_model(biopy_struc[0].child_list[0], data_type='chain')
```

load_template(file_path, chain_name='A')

Loads a template structure from a PDB file. Produces a Template object that can be saved in a variable.
Each template in ModeRNA has only one chain. By default, the chain with id 'A' is loaded. Another chain id can be specified optionally
Arguments:
path+filename of a PDB structure file
chain id (by default 'A')

```
t = load_template('1F1T.pdb')
```

```
t = load_template('1F1T.pdb','A')
```

match_template_with_alignment(template, alignment)

Checks, if the sequence of the template structure is equal to the second sequence in the alignment. Writes an according message and returns True or False. Small inconsistencies between both sequences, e.g. backbone breaks, or missing modification symbols are corrected automatically in the alignment, and changes are reported in the logfile.
Both sequences also count as equal if one has modified nucleotides, and the other the corresponding unmodified nucleotides in the same position, or if one of the sequences contains the wildcard symbol '.'. Thus, the sequence "AGU" is equal to both "A7Y" and "A.U".
Arguments:
Template object
Alignment object

```
match_template_with_alignment(t,a)
boolean = match_template_with_alignment(t,a)
```

write_model(model, pdb_file_name='moderna_model.pdb')

Writes a model to a PDB file. The residues in the file are sorted. All residues keep their numbers as last assigned.
Arguments:
Structure object (model or template)
name of the PDB file (optional; by default moderna_model.pdb)

```
write_model(m)
write_model(m, 'output.pdb')
write_model(m, 'output.pdb', 'log.txt')
```

Find template procedure used on a ModeRNA server