rna-tools documentation

Release

Marcin Magnus

Feb 19, 2020
# User Documentation

## 1 Getting Started: I want to...

- **1.1** fetch a structure from the PDB database .............................................. 3
- **1.2** fetch a biologicaly assembly ........................................................... 3
- **1.3** get sequences of a bunch of PDB files ................................................. 4
- **1.4** get secondary structures of your PDB files ........................................... 4
- **1.5** delete a part of of your structure ....................................................... 5
- **1.6** get numbering of your structure and rename chains ............................... 5
- **1.7** edit your structure (rename chain) ..................................................... 6
- **1.8** find missing atoms in my structure ..................................................... 6
- **1.9** mutate residues .................................................................................... 6
- **1.10** add missing atoms .............................................................................. 8

## 2 RNA PDB Tools

- **2.1** rna_pdb_toolsx .................................................................................... 11
- **2.2** get RNAPuzzle ready ......................................................................... 14
- **2.3** get sequence ...................................................................................... 16
- **2.4** fetch .................................................................................................. 16
- **2.5** fetch Biological Assembly .................................................................. 17
- **2.6** delete .................................................................................................. 17
- **2.7** edit ..................................................................................................... 17
- **2.8** the library .......................................................................................... 18
- **2.9** PDB Edit Bfactor/Occupancy ............................................................... 23
- **2.10** Add chain to a file ............................................................................ 25
- **2.11** Measure distance between atoms ..................................................... 25

## 3 RNA Tools

- **3.1** RNA Sequence .................................................................................. 29
- **3.1.1** Installation ....................................................................................... 29
  - **3.1.1.1** ContextFold ................................................................................ 29
  - **3.1.1.2** ViennaRNA .............................................................................. 29
  - **3.1.1.3** ipknot OSX ............................................................................. 30
  - **3.1.1.4** RNA Structure ........................................................................ 30
  - **3.1.1.5** MC-Sym ................................................................................... 30
- **3.1.2** FAQ ................................................................................................. 30
- **3.1.3** TIPS ................................................................................................. 30
- **3.2** RNA Secondary Structure ................................................................. 34
- **3.2.1** rna_dot2ct.py ............................................................................... 36
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.2</td>
<td>Secondary structure format conversion</td>
<td>36</td>
</tr>
<tr>
<td>3.3</td>
<td>Search</td>
<td>38</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Blast PDB</td>
<td>38</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Rfam Search</td>
<td>38</td>
</tr>
<tr>
<td>3.4</td>
<td>RNA Alignment</td>
<td>39</td>
</tr>
<tr>
<td>3.4.1</td>
<td>RNASeq</td>
<td>40</td>
</tr>
<tr>
<td>3.4.2</td>
<td>RNAalignment</td>
<td>44</td>
</tr>
<tr>
<td>3.4.3</td>
<td>rna_alignment_get_species.py</td>
<td>49</td>
</tr>
<tr>
<td>3.4.4</td>
<td>rna_alignment_calc_energy.py</td>
<td>50</td>
</tr>
<tr>
<td>3.4.5</td>
<td>rna_align_get_ss_from_fasta.py</td>
<td>52</td>
</tr>
<tr>
<td>3.4.6</td>
<td>rna_align_get_ss_from_stk.py</td>
<td>52</td>
</tr>
<tr>
<td>3.4.7</td>
<td>rna_align_distance_to_seq.py</td>
<td>52</td>
</tr>
<tr>
<td>3.4.8</td>
<td>rna_align_foldability.py</td>
<td>53</td>
</tr>
<tr>
<td>3.4.9</td>
<td>Random assignment of nucleotides</td>
<td>57</td>
</tr>
<tr>
<td>3.4.10</td>
<td>CMAign</td>
<td>59</td>
</tr>
<tr>
<td>3.4.11</td>
<td>RChie</td>
<td>60</td>
</tr>
<tr>
<td>3.4.12</td>
<td>Renumber a pdb file according to alignment</td>
<td>62</td>
</tr>
<tr>
<td>3.5</td>
<td>Root Mean Square Deviation (RMSD)</td>
<td>63</td>
</tr>
<tr>
<td>3.5.1</td>
<td>rna_calc_rmsd</td>
<td>63</td>
</tr>
<tr>
<td>3.5.2</td>
<td>rna_calc_rmsd_multi_targets</td>
<td>67</td>
</tr>
<tr>
<td>3.5.3</td>
<td>rna_calc_rmsd_triaf</td>
<td>68</td>
</tr>
<tr>
<td>3.5.4</td>
<td>rna_calc_rmsd_all_vs_all</td>
<td>69</td>
</tr>
<tr>
<td>3.6</td>
<td>Interaction Network Fidelity (INF)</td>
<td>70</td>
</tr>
<tr>
<td>3.6.1</td>
<td>rna_calc_inf</td>
<td>70</td>
</tr>
<tr>
<td>3.6.2</td>
<td>rna_calc_dinf</td>
<td>71</td>
</tr>
<tr>
<td>3.7</td>
<td>RNA filter (DCA)</td>
<td>73</td>
</tr>
<tr>
<td>3.7.1</td>
<td>rna_filter.py - calculate distances based on given restraints on PDB files or SimRNA trajectories</td>
<td>73</td>
</tr>
<tr>
<td>3.7.2</td>
<td>rna_dca_mapping.py</td>
<td>74</td>
</tr>
<tr>
<td>3.7.3</td>
<td>show_dists - show distances in PyMOL</td>
<td>75</td>
</tr>
<tr>
<td>3.7.4</td>
<td>rna_pairs2SimRNArestrs.py - convert pairs to SimRNA restraints</td>
<td>76</td>
</tr>
<tr>
<td>3.7.5</td>
<td>rna_ss_get_bps.py - get a list of base pairs for a given “fasta ss” file</td>
<td>77</td>
</tr>
<tr>
<td>3.7.6</td>
<td>rna_pairs_diff.py - get a diff of pairs</td>
<td>77</td>
</tr>
<tr>
<td>3.8</td>
<td>Contacts classification &amp; secondary structure detection</td>
<td>78</td>
</tr>
<tr>
<td>3.8.1</td>
<td>3DNA (contacts classification &amp; secondary structure detection)</td>
<td>78</td>
</tr>
<tr>
<td>3.8.2</td>
<td>ClaRNA (contacts classification)</td>
<td>79</td>
</tr>
<tr>
<td>3.9</td>
<td>RNA 3D structure prediction</td>
<td>80</td>
</tr>
<tr>
<td>3.9.1</td>
<td>ROSETTA</td>
<td>80</td>
</tr>
<tr>
<td>3.9.1.1</td>
<td>Run (modeling)</td>
<td>80</td>
</tr>
<tr>
<td>3.9.1.2</td>
<td>Get a number of structures</td>
<td>82</td>
</tr>
<tr>
<td>3.9.1.3</td>
<td>Get a head of a Rosetta silent file</td>
<td>82</td>
</tr>
<tr>
<td>3.9.1.4</td>
<td>Cluster</td>
<td>83</td>
</tr>
<tr>
<td>3.9.1.5</td>
<td>Minimize</td>
<td>84</td>
</tr>
<tr>
<td>3.9.1.6</td>
<td>Extract lowscore decoy</td>
<td>85</td>
</tr>
<tr>
<td>3.9.1.7</td>
<td>Check progress</td>
<td>85</td>
</tr>
<tr>
<td>3.9.2</td>
<td>SimRNA</td>
<td>86</td>
</tr>
<tr>
<td>3.9.2.1</td>
<td>Select low energy frames</td>
<td>86</td>
</tr>
<tr>
<td>3.9.2.2</td>
<td>Extract</td>
<td>86</td>
</tr>
<tr>
<td>3.9.3</td>
<td>SimRNAweb</td>
<td>86</td>
</tr>
<tr>
<td>3.9.3.1</td>
<td>Download files of a SimRNAweb run</td>
<td>86</td>
</tr>
<tr>
<td>3.9.4</td>
<td>SimRNAtrajectory</td>
<td>88</td>
</tr>
<tr>
<td>3.10</td>
<td>RNA Refinement (QRNAS)</td>
<td>89</td>
</tr>
<tr>
<td>3.11</td>
<td>diffpdb</td>
<td>92</td>
</tr>
<tr>
<td>3.12</td>
<td>RNA clustering with CLANS (clanstix)</td>
<td>95</td>
</tr>
<tr>
<td>3.12.1</td>
<td>Color groups</td>
<td>96</td>
</tr>
</tbody>
</table>
rna-tools: a toolbox to analyze structures and simulations of RNA

The code of the project can be found at GitHub (https://github.com/mmagnus/rna-tools).

This documentation can be found at http://rna-tools.rtfd.io

If something does not work for you, please make an issue, using https://github.com/mmagnus/rna-tools/issues

The main documentation for the site is organized into sections:

user-docs Information about development is also available:

dev-docs
CHAPTER 1

Getting Started: I want to . . .

1.1 fetch a structure from the PDB database

Example:

```bash
$ rna_pdb_toolsx.py --fetch 1xjr
downloading...1xjr ok
```

1.2 fetch a biologicaly assembly

Example:

```bash
$ rna_pdb_toolsx.py --fetch_ba 1xjr
downloading...1xjr_ba.pdb ok
```

or over a list of pdb ids in a text file:

```bash
$ cat data/pdb_ids.txt
ly26
1fir

$ while read p; do rna_pdb_toolsx.py --fetch_ba $p; done < data/pdb_ids.txt
downloading...ly26_ba.pdb ok
downloading...1fir_ba.pdb ok

$ ls *.pdb
1fir_ba.pdb ly26_ba.pdb
```
1.3 get sequences of a bunch of PDB files

Example:

```bash
rna_pdb_toolsx.py --get_seq *.pdb
# 1xjr
> A:1-47
GGAGUUCECGGCAGGACGCGAGUACGAGGGUUGUUGAAUU
# 6TNA
> A:1-76
GCGGAUUAAcUCACAGuGGAGAGCCAGACUAgAucUGGAGgUCcUGUuCCaUCCACAGAAUUCGCACCA
# rp2_bujnicki_1_rpr
> A:1-15
CGCGAAACCUACUG
> B:1-10
CGCGCAGCCU
> C:1-15
CGCGAAACCUACUG
> D:1-10
CGCGCAGCCU
> E:1-15
CGCGAAACCUACUG
> F:1-10
CGCGCAGCCU
> G:1-15
CGCGAAACCUACUG
> H:1-10
CGCGCAGCCU
```

1.4 get secondary structures of your PDB files

Python parser to 3dna <http://x3dna.org/>.

Installation:

```
# install the code from http://forum.x3dna.org/downloads/3dna-download/
Create a copy of the rna_x3dna_config_local_sample.py (remove "_sample") present in
--rna-tools/rna_tools/tools/rna_x3dna folder.
Edit this line:
BINARY_PATH = <path to your x3dna-dssr file>
matching the path with the path of your x3dna-dssr file.
e.g. in my case: BINARY_PATH = ~/bin/x3dna-dssr.bin
```

For one structure you can run this script as:

```
[mm] py3dna$ git:(master) ./rna_x3dna.py test_data/1xjr.pdb
test_data/1xjr.pdb
>1xjr nts=47 [1xjr] -- secondary structure derived by DSSR
GGAGUUCECGGCAGGACGCGAGUACGAGGGUUGUUGAAUU
.(((((...(((.(((((......))..))..))).))))))
```

For multiple structures in the folder, run the script like this:

```
[mm] py3dna$ git:(master) ./rna_x3dna.py test_data/*
test_data/1xjr.pdb
```
>lxjr nts=47 [lxjr] -- secondary structure derived by DSSR
GAGUUCACCCCGAGCGGAGUACCGGGAACGGGUAAGUAAU
((((((((((((((........)))).)).))).))))))
test_data/6TNA.pdb

>6TNA nts=76 [6TNA] -- secondary structure derived by DSSR
GCGGAUUUAGCUAAGuGGGAGGcCAGAcUGApGGuUGGcUGCGuPCGAGUCCACAGAAUUCGACCA
(((((((((........)).)).))))))
test_data/rp2_bujnicki_1_rpr.pdb

>rp2_bujnicki_1_rpr nts=100 [rp2_bujnicki_1_rpr] -- secondary structure derived by DSSR
CCGGAGGAACUACUG&CCGGCAGCCU&CCGGAGGAACUACUG&CCGGCAGCCU&CCGGAGGAACUACUG&CCGGCAGCCU
[{
[{{{{}}}}}
]}

get_ion_water_report()

@todo File name: /tmp/tmp0pdNHS
	no. of DNA/RNA chains: 0
no. of nucleotides: 174
no. of waters: 793
no. of metals: 33
[Na=29, Mg=1, K=3]

get_modifications()

Run find_pair to find modifications.

get_secstruc()

Get secondary structure.

get_seq()

Get sequence.

Somehow 1bzt_1 x3dna UCAGACUUUUAAPCUGA, what is P? P -> u

run_x3dna (show_log=False)

1.5 delete a part of your structure

Examples:

$ for i in *pdb; do rna_pdb_toolsx.py --delete A:46-56 $i > ../rpr_rm_loop/$i ; done

go over all files in the current directory, remove a fragment of chain A, residues between 46-56 (including them) and save outputs to in the folder rpr_rm_loops.

1.6 get numbering of your structure and rename chains

Rename chain B in structure 4_das_1_rpr.pdb:

$ rna_pdb_toolsx.py --get_seq 4_das_1_rpr.pdb

> 4_das_1_rpr.pdb

GGCUUAUCAAGAGGGAGGCUGCCAGUUAACCCCGGACCCACACUGCUAGGUAGCGUCAGGUUCUGACUAGCGGUCAGGUGCGAAGCCAAUCCCGG

$ rna_pdb_toolsx.py --edit 'B:1-126>A:1-126' 4_das_1_rpr.pdb

1.5 delete a part of your structure

5
1.7 edit your structure (rename chain)

Examples:

$ rna_pdb_toolsx.py --edit 'A:3-21>A:1-19' 1f27_clean.pdb > 1f27_clean_A1-19.pdb

or even:

$ rna_pdb_toolsx.py --edit 'A:3-21>A:1-19,B:22-32>B:20-30' 1f27_clean.pdb > 1f27_clean_renumb.pdb

or even, even, do rename X chain to A only for Chen’s pdb structures in the folder, in place (so don’t create a new file):

for i in *Chen*; do rna_pdb_toolsx.py --edit 'X:1-125>A:1-125' $i > ${i}_temp; mv ${i}→_temp ${i}; done

# do only edit for Chen's pdb structures, in place.

1.8 find missing atoms in my structure

Run:

$ rna_pdb_toolsx.py --get_rnapuzzle_ready input/1_das_1_rpr_fixed.pdb

HEADER Generated with rna-pdb-tools
HEADER ver 91ed4f8-dirty
HEADER https://github.com/mmagnus/rna-pdb-tools
HEADER Sun Mar 5 10:58:07 2017
REMARK 000 Missing atoms:
REMARK 000 + P B <Residue C het= resseq=1 icode= > residue # 1
REMARK 000 + OP1 B <Residue C het= resseq=1 icode= > residue # 1
REMARK 000 + OP2 B <Residue C het= resseq=1 icode= > residue # 1
REMARK 000 + O5' B <Residue C het= resseq=1 icode= > residue # 1
ATOM 1 P C A 1 -16.936 -3.789 68.770 1.00 11.89 P
ATOM 2 OP1 C A 1 -17.105 -3.675 67.302 1.00 14.35 O
ATOM 3 OP2 C A 1 -15.666 -4.265 69.342 1.00 12.68 O
...

1.9 mutate residues

For example, to replace the first four residues of chain A into adenines and 13th A of chain B, run:

rna_pdb_toolsx.py --mutate 'A:1A+2A+3A+4A,B:13A' \ 
--inplace output/205d_rmh2o_mutA1234-B1_inplace.pdb
Figure. Input structure on the left, mutated structure on the right.

If, for whatever reason, the tool here does not do what you want, use the tool from MC-Fold|MC-Sym Pipeline (go there https://www.major.irc.ca/MC-Pipeline/ and scroll down to the Section: “RNA SEQUENCE MUTATION” at the very bottom of the page).

Moreover, you can also mutate interactively proteins and nucleic acids with PyMOL >2.

Learn more here https://pymolwiki.org/index.php/Mutagenesis

If you want to mutate with PyMOL with command-line see this https://pymolwiki.org/index.php/Rotkit

1.9. mutate residues
1.10 add missing atoms

The tool is using the function:

\[
\text{RNAStructure}.\text{get_rnapuzzle_ready}(\text{renumber_residues}=\text{True}, \text{ fix_missing_atoms}=\text{True}, \text{ rename_chains}=\text{True}, \text{ report_missing_atoms}=\text{True}, \text{ verbose}=\text{True})
\]

Get rnapuzzle (SimRNA) ready structure.
Clean up a structure, get current order of atoms.

- **Parameters**
  - `renumber_residues` – boolean, from 1 to …, second chain starts from 1 etc.
  - `fix_missing_atoms` – boolean, superimpose motifs from the minilibrary and copy-paste missing atoms, this is super crude, so should be used with caution.

Submission format @http://ahsoka.u-strasbg.fr/rnapuzzles/

Run `rna_tools.rna_tools.lib.RNAStructure.std_resn()` before this function to fix names.

- 170305 Merged with get_simrna_ready and fixing OP3 terminal added
- 170308 Fix missing atoms for bases, and O2’

Fig. Add missing O2’ atom (before and after).
Fig. The residue to fix is in cyan. The G base from the library in red. Atoms O4’, C2’, C1’ are shared between the sugar (in cyan) and the G base from the library (in red). These atoms are used to superimpose the G base on the sugar, and then all atoms from the base are copied to the residues.

Warning: It was only tested with the whole base missing!

Fig. Rebuild ACGU base-less. It's not perfect but good enough for some applications.

Warning: It was only tested with the whole base missing!
Warning: requires: Biopython
CHAPTER 2

RNA PDB Tools

2.1 rna_pdb_toolsx

rna_pdb_toolsx - a swiss army knife to manipulation of RNA pdb structures

Tricks:

```bash
for i in *pdb; do rna_pdb_toolsx.py --get-rnapuzzle-ready $i > ${i/.pdb/_rpr.pdb}; done
```

Usage:

```bash
$ for i in *pdb; do rna_pdb_toolsx.py --delete A:46-56 $i > ../rpr_rm_loop/$i ; done
$ rna_pdb_toolsx.py --get-seq *
# BujnickiLab_RNApuzzle14_n01bound
> A:1-61
# BujnickiLab_RNApuzzle14_n02bound
> A:1-61
CGUAGCCAGGAAACUGGGCGGAAGUAAGGCCCAUUGCACUCCGGGCCUGAAGCAACGCG
[...]
```

usage: rna_pdb_toolsx.py [-h] [--version] [-r] [--renum-atoms]
[-renum-atoms-dirty] [--delete-anisou]
[-split-alt-locations] [-c] [--is-pdb] [--is-nmr]
[--nmr-dir NMR_DIR] [--un-nmr] [--orgmode]
[--get-chain GET_CHAIN] [--fetch] [--fetch-ba]
[--rosetta2generic] [--get-rnapuzzle-ready] [--rpr]
[--no-hr] [--renumber-residues]
[--dont-rename-chains] [--dont-fix-missing-atoms]
[--dont-report-missing-atoms] [--collapsed-view]
[--cv] [-v] [--replace-hetatm] [--inplace]
[--mutate MUTATE] [--edit EDIT]
[--rename-chain RENAME_CHAIN]
[--swap-chains SWAP_CHAINS]
Positional arguments:

file

Options:

--version Undocumented
-r=False, --report=False get report
--renum-atoms=False renumber atoms, tested with --get-seq
--renum-residues-dirty=False Undocumented
--delete-anisou=False remove files with ANISOU records, works with --inplace
--split-alt-locations=False @todo
-c=False, --clean=False get clean structure
--is-pdb=False check if a file is in the pdb format
--is-nmr=False check if a file is NMR-style multiple model pdb
--nmr-dir make NMR-style multiple model pdb file from a set of files
rna_pdb_toolsx.py --nmr-dir . 'cwc15_u5_fragments*.pdb' > ~/Desktop/cwc15-u5.pdb
please use ‘’ for pattern file recognition, this is a hack to deal with folders
with thousands of models, if you used only *.pdb then the terminal will
complain that you selected to many files.
--un-nmr=False split NMR-style multiple model pdb files into individual models [biopython]
--orgmode=False get a structure in org-mode format <sick!>
--get-chain get chain, one or many, e.g. A, but now also ABC works
--fetch=False fetch file from the PDB db
--fetch-ba=False fetch biological assembly from the PDB db
--get-seq=False get seq
--get-ss=False hide warnings, works with --get-chain, it hides warnings that given
changes are not detected in a PDB file
--compact=False with --get-seq, get it in compact view’ $ rna_pdb_toolsx.py
--rosetta2generic=False convert ROSETTA-like format to a generic pdb
with --get-seq --compact *.pdb # 20_Bujnicki_1 ACCCGCAAGGCC-GACGCGCCCGCUGUGAUGCAAGGCACCGCUUGC-GCGUGGGCGCUCAUGGGU # A:1-68 # 20_Bujnicki_2 ACCCGCAAGGCCACCGGCACCGGCGCCCGCUGUGAUGCAAGGCACCGCUUGC-GCGUGGGCGCUCAUGGGU # A:1-68 # 20_Bujnicki_3 ACCCGCAAGGCCACCGGCACCGGCGCCCGCUGUGAUGCAAGGCACCGCUUGC-GCGUGGGCGCUCAUGGGU # A:1-68 # 20_Bujnicki_4
--get-ss=False get secondary structure
--get-rnapuzzle-ready=False  get RNApuzzle ready (keep only standard atoms).’ Be default it
does not renumber residues, use –renumber-residues [requires BioPython]

--rpr=False  alias to get_rnapuzzle ready)
--no-hr=False  do not insert the header into files
--renumber-residues=False  by default is false
--dont-rename-chains=False  used only with –get-rnapuzzle-ready. By default: –get-rnapuzzle-ready rename chains from ABC.. to stop behavior switch on this option
--dont-fix-missing-atoms=False  used only with –get-rnapuzzle-ready
--dont-report-missing-atoms=False  used only with –get-rnapuzzle-ready
--collapsed-view=False  Undocumented
--cv=False  alias to collapsed_view
-v=False, --verbose=False  tell me more what you’re doing, please!
--replace-hetatm=False  replace ‘HETATM’ with ‘ATOM’ [tested only with –get-rnapuzzle-ready]
--inplace=False  in place edit the file! [experimental, only for get_rnapuzzle_ready, delete,–get-ss, –get-seq, –edit-pdb]
--mutate  mutate residues, e.g., –mutate “A:1A+2A+3A+4A,B:1A” to mutate to
adenines the first four nucleotides of the chain A and the first nucleotide
of the chain B
--rename-chain  edit ‘A>B’ to rename chain A to chain B
--swap-chains  B>A, rename A to _, then B to A, then _ to B
--replace-chain=  a file PDB name with one chain that will be used to replace the chain in the
original PDB file, the chain id in this file has to be the same with the chain
id of the original chain
--delete=  delete the selected fragment, e.g. A:10-16, or for more than one fragment
–delete ‘A:1-25+30-57’
--extract=  extract the selected fragment, e.g. A:10-16, or for more than one fragment
–extract ‘A:1-25+30-57’
--extract-chain  extract chain, e.g. A
--uniq  rna_pdb_toolsx.py –get-seq –uniq ‘[5]’ –compact –chain-first * 
  sort A:1-121  ACCUUGCGCAACUGGCCAUCUGGAGCGCUGGCUUG- 
  CGCCCGCAGUACCC...CA # rp13nc3295_min.out.1  A:1-123 
  ACCUUGCGCAACUGGCCAUCUGGAGCGCUGGCUUGAGCGG- 
  GCUGUCC...AG # rp13cp0016_min.out.1  A:1-123 ACCUUGCGC- 
  GACUGCGAUCUGGAGUGCUUGAGCGCUUG...AG # zcp_6537608a_ALL-000001_AA  A:1-45 57-71 GGGUCGUGACUG- 
  GCGAACAGGUGGGAAACCACCGGGAGCGACCACCAGCCCGC- 
  CUGGCC # solution
--chain-first=False  Undocumented
--oneline=False  Undocumented
RNA PDB tools documentation, Release

--fasta=False

with --get-seq, show sequences in fasta format, can be combined with
--compact (mind, chains will be separated with ' ' in one line)

$ rna_pdb_toolsx.py --get-seq --fasta --compact input/20_Bujnicki_1.pdb
> 20_Bujnicki_1 ACCCGCAAGGCCGACGGC GCCGCCGCUGGUG-
CAAGUCCAGCCACGCUUCGGCGUGGGCGCUCAUGGGU

2.2 get RNAPuzzle ready

class rna_tools.rna_tools_lib.RNAStructure(fn)
    RNAStructure - handles an RNA pdb file.

    Attributes:
    fn (string) : filename of the pdb file
    lines (list) : the PDB file is loaded and
                  ATOM/HETATM/TER/END go to self.lines

    get_rnapuzzle_ready (renumber_residues=True, fix_missing_atoms=True,
                         rename_chains=True, report_missing_atoms=True, verbose=True)
    Get rnapuzzle (SimRNA) ready structure.
    Clean up a structure, get current order of atoms.

    Parameters
    - renumber_residues – boolean, from 1 to …, second chain starts from 1 etc.
    - fix_missing_atoms – boolean, superimpose motifs from the minilibrary and copy-
                          paste missing atoms, this is super crude, so should be used with caution.

    Submission format @http://ahsoka.u-strasbg.fr/rnapuzzles/

    Run rna_tools.rna_tools.lib.RNAStructure.std_resn() before this function to fix
    names.

    - 170305 Merged with get_simrna_ready and fixing OP3 terminal added
    - 170308 Fix missing atoms for bases, and O2’

    /missing_o
    /missing_o_fixed
Fig. Add missing O2’ atom (before and after).

Fig. The residue to fix is in cyan. The G base from the library in red. Atoms O4’, C2’, C1’ are shared between the sugar (in cyan) and the G base from the library (in red). These atoms are used to superimpose the G base on the sugar, and then all atoms from the base are copied to the residues.

Fig. Rebuild ACGU base-less. It’s not perfect but good enough for some applications.

**Warning:** It was only tested with the whole base missing!
2.3 get sequence

Example:

```
$ rna_pdb toolsx.py --get-seq 5_solution_1.pdb
> 5_solution_1.pdb A:1-576
CAUCCGGUAUCCCAAGACAUCUGGGGUUGGGGAAGUAUCAUGGCUAAUCACCAUGAUGCAAUCGGGUUGAACACUUAAUUGGUAACGUGGU
```

```python
class rna_tools.rna_tools_lib.RNAStructure(fn)

RNAStructure - handles an RNA pdb file.

Attributes:

- fn (string) : filename of the pdb file
- lines (list) : the PDB file is loaded and ATOM/HETATM/TER/END go to self.lines

get_seq(compact=False, chainfirst=False, fasta=False, addfn="")

Get seq (v2) gets segments of chains with correct numbering

Run:

```
python rna_pdb_seq.py input/lykq_clx.pdb
> lykq_clx A:101-111
GGAGUCGCCC
> lykq_clx B:201-238
GGCCAGGCGGCGCCAGCUCUCGGAGCAUACUGGC
> 6_solution_0 A:1-19 26-113 117-172
GCCCGAGUGCGUGCCAGCGGAGUUGAAGCG
```

Chains is {'A': {'header': 'A:1-19 26-113 117-172', 'resi': [1, 2, 3, ..., 19, 26, 27, ..., 172], 'seq': ['G', 'G', 'C', 'G', ..., 'C']}, 'U': 'C'}

Chains are in other as the appear in the file.

```
Warning: take only ATOM and HETATM lines.
```

2.4 fetch

Example:

```
$ rna_pdb toolsx.py --fetch 1xjr
downloading...1xjr ok
```

```
rna_tools.rna_tools_lib.fetch(pdb_id, path='')
fetch pdb file from RCSB.org https://files.rcsb.org/download/1Y26.pdb
```
2.5 fetch Biological Assembly

Example:

```bash
$ rna_pdb_toolsx.py --fetch-ba 1xjr
downloading...1xjr_ba.pdb ok
```

or over a list of pdb ids in a text file:

```bash
$ cat data/pdb_ids.txt
ly26
lfir

$ while read p; do rna_pdb_toolsx.py --fetch-ba $p; done < data/pdb_ids.txt
downloading...ly26_ba.pdb ok
downloading...lfir_ba.pdb ok

$ ls *.pdb
lfir_ba.pdb ly26_ba.pdb
```

```python
rna_tools.rna_tools_lib.fetch_ba(pdb_id, path=':')
fetch biological assembly pdb file from RCSB.org
```

```bash
>>> fetch_ba('1xjr')
```

2.6 delete

Examples:

```bash
$ for i in *.pdb; do rna_pdb_toolsx.py --delete A:46-56 $i > ../rpr_rm_loop/$i ; done
```

go over all files in the current directory, remove a fragment of chain A, residues between 46-56 (including them) and save outputs to in the folder rpr_rm_loops.

2.7 edit

```python
rna_tools.rna_tools_lib.edit_pdb(f, args)
```

Edit your structure.

| The function can take A:3-21>A:1-19 or even syntax like this A:3-21>A:1-19,B:22-32>B:20-30 and will do an editing. |
| The output is printed, line by line. Only ATOM lines are edited! |

Examples:

```bash
$ rna_pdb_toolsx.py --edit 'A:3-21>A:1-19' 1f27_clean.pdb > 1f27_clean_A1-19.pdb
```

or even:

```bash
$ rna_pdb_toolsx.py --edit 'A:3-21>A:1-19,B:22-32>B:20-30' 1f27_clean.pdb > 1f27_clean_renumb.pdb
```
2.8 the library

rna-tools documentation, Release

rna_tools_lib.py - main lib file, many tools in this lib is using this file.

class rna_tools.rna_tools_lib.RNAStructure(fn)

RNAStructure - handles an RNA pdb file.

Attributes:

fn (string) : filename of the pdb file
lines (list) : the PDB file is loaded and ATOM/HETATM/TER/END go to self.lines

edit_occupancy_of_pdb (txt, pdb, pdb_out, v=False)

Make all atoms 1 (flexi) and then set occupancy 0 for selected atoms. Return False if error. True if OK

fix_O_in_UC()

fix_op_atoms()

Replace OXP’ to OPX1, e.g. ('O1P' -> 'OP1')

fix_with_qrnas (outfn='', verbose=False)

Add missing heavy atom.

A residue is recognized based on a residue names.

Copy QRNAS folder to curr folder, run QRNAS and remove QRNAS.

Warning: QRNAS required (http://genesilico.pl/QRNAS/QRNAS.tgz)

get_all_chain_ids()

Returns chain ids, e.g. set([‘A’, ‘B’])

Return type set

get_atom_code (line)

Get atom code from a line of a PDB file

get_atom_coords (line)

Get atom coordinates from a line of a PDB file

get_atom_num (line)

Extract atom number from a line of PDB file :param * line = ATOM line from a PDB file:

Output:
• atom number (int)

get_info_chains ()

return A:3-21 B:22-32

get_report ()

Returns report, messages collected on the way of parsing this file

Return type string

get_res_code (line)

Get residue code from a line of a PDB file

get_res_num (line)

Extract residue number from a line of PDB file :param * line = ATOM line from a PDB file:

Output:
• residue number as an integer
**get_rnapuzzle_ready** (renumber_residues=True, fix_missing_atoms=True, rename_chains=True, report_missing_atoms=True, verbose=True)

Get rnapuzzle (SimRNA) ready structure.

Clean up a structure, get current order of atoms.

**Parameters**

- **renumber_residues** – boolean, from 1 to . . . , second chain starts from 1 etc.
- **fix_missing_atoms** – boolean, superimpose motifs from the minilibrary and copy-paste missing atoms, this is super crude, so should be used with caution.

Submission format @http://ahsoka.u-strasbg.fr/rnapuzzles/

Run `rna_tools.rna_tools.lib.RNAStructure.std_resn()` before this function to fix names.

- 170305 Merged with get_simrna_ready and fixing OP3 terminal added
- 170308 Fix missing atoms for bases, and O2’

Fig. Add missing O2’ atom (before and after).
The residue to fix is in cyan. The G base from the library in red. Atoms O4’, C2’, C1’ are shared between the sugar (in cyan) and the G base from the library (in red). These atoms are used to superimpose the G base on the sugar, and then all atoms from the base are copied to the residues.

Fig. Rebuild ACGU base-less. It’s not perfect but good enough for some applications.

Warning: It was only tested with the whole base missing!

Warning: requires: Biopython

get_seq(compact=False, chainfirst=False, fasta=False, addfn="")
Get seq (v2) gets segments of chains with correct numbering

Run:
```
python rna_pdb_seq.py input/lykq_clx.pdb
> lykq_clx A:101-111
GGAGUCGCCC
> lykq_clx B:201-238
GGCGAGGCCGUGCAGUCUCUUCGAGCAAUACUGGC
> 6_solution_0 A:1-19 26-113 117-172
GGCGAGGUGUCUCUCCCCACGUCCGGAGUUAAGGGAAG

Chains is {'A': {'header': 'A:1-19 26-113 117-172', 'resi': [1, 2, 3, ..., 19, 26, 27, ..., 172], 'seq': ['G', 'G', 'C', 'G', ... 'C', 'G', 'U', 'C']}}

Chains are in other as the appear in the file.

Warning: take only ATOM and HETATM lines.

get_text(add_end=True)
works on self.lines.
is_amber_like()  
Use self.lines and check if there is XX line

is_mol2()  
Return True if is_mol2 based on the presence of `@<TRIPOS>`.

is_nmr()  
True if the file is an NMR-style multiple model pdb.

Returns  True or Fo  
Return type  boolean

is_pdb()  
Return True if the files is in PDB format.

If self.lines is empty it means that nothing was parsed into the PDB format.

remove (verbose)  
Delete file, self.fn

remove_ion()  
TER 1025 U A 47 HETATM 1026 MG MG A 101 42.664 34.395 50.249 1.00 70.99 MG  
HETATM 1027 MG MG A 201 47.865 33.919 48.090 1.00 67.09 MG  

type  object

remove_water()  
Remove HOH and TIP3

rename_chain (chain_id_old, chain_id_new, debug=False)  
Rename chains.

Parameters  
• chain_id_old (str) – e.g., A  
• chain_id_new (str) – e.g., B  
• debug (bool) – show some diagnostics

Returns  pdb content (txt) self.lines is updated with new lines

renum_atoms()  
Renum atoms, from 1 to X for line; ATOM/HETATM

set_atom_occupancy (line, occupancy)  
set occupancy for line

set_occupancy_atoms (occupancy)  

Parameters  occupancy –

set_res_code (line, code)  

Parameters  
• lines –  
• code –

path (str): The path of the file to wrap field_storage (FileStorage): The :class:Y instance to wrap temporary (bool): Whether or not to delete the file when the File instance is destructed

Returns  A buffered writable file descriptor  
Return type  BufferedFileStorage

std_resn()  
‘Fix’ residue names which means to change them to standard, e.g. RA5 -> A

Works on self.lines, and returns the result to self.lines.

Will change things like:
### un_nmr (startwith=True, verbose=False)
Un NMR - Split NMR-style multiple model pdb files into individual models.

Take self.fn and create new files in the way:

```
input/1a9l_NMR_1_2_models.pdb
input/1a9l_NMR_1_2_models_0.pdb
input/1a9l_NMR_1_2_models_1.pdb
```

**Warning:** This function requires biopython.

### write (outfn, v=True)
Write `self.lines` to a file (and END file)

```python
rna_tools.rna_tools_lib.collapsed_view(args)
```

Collapsed view of pdb file. Only lines with C5' atoms are shown and TER, MODEL, END.

**Example:**

```
[mm] rna_tools git:(master) $ python rna-pdb-tools.py --cv input/1f27.pdb
   C
ATOM 23 C5'  C   A   4  19.700  19.206  5.034  1.00 12.65
   C
ATOM 43 C5'  C   A   5  14.537  16.130  6.444  1.00  8.74
   C
ATOM 63 C5'  G   A   6  11.726  11.579  9.544  1.00  9.81
   C
ATOM 86 C5'  U   A   7  12.007  7.281 13.726  1.00 11.35
   C
ATOM 106 C5'  C   A   8 12.087  6.601 18.999  1.00 12.74
   C
TER
```

```python
rna_tools.rna_tools_lib.edit_pdb(f, args)
```

Edit your structure.

The function can take `A:3-21>A:1-19` or even syntax like this `A:3-21>A:1-19, B:22-32>B:20-30` and will do an editing.

The output is printed, line by line. Only ATOM lines are edited!

**Examples:**

```bash
$ rna_pdb_toolsx.py --edit 'A:3-21>A:1-19' 1f27_clean.pdb > 1f27_clean_A1-19.pdb
```

or even:

```bash
$ rna_pdb_toolsx.py --edit 'A:3-21>A:1-19,B:22-32>B:20-30' 1f27_clean_renumb.pdb
```
### rna_tools documentation, Release

**fetch** *(pdb_id, path=':')*

fetch pdb file from RCSB.org https://files.rcsb.org/download/1Y26.pdb

**fetch_ba** *(pdb_id, path=':')*

fetch biological assembly pdb file from RCSB.org

```python
>>> fetch_ba('1xjr')
...
```

**fetch_cif_ba** *(cif_id, path=':')*

fetch biological assembly cif file from RCSB.org

**get_version** *(currfn='', verbose=False)*

Get version of the tool based on state of the git repository. Return version. If currfn is empty, then the path is ':'. Hmm.. I think it will work. We will see. The version is not printed! https://github.com/m4rx9/curr_version/

**replace_chain** *(struc_fn, insert_fn, chain_id)*

Replace chain of the main file *(struc_fn)* with some new chain *(insert_fn)* of given chain id.

**Parameters**

- **struc_fn**(str) – path to the main PDB file
- **insert_fn**(str) – path to the file that will be injected in into the main PDB file
- **chain_id**(str) – chain that will be inserted into the main PDB file

**Returns** text in the PDB format

**sort_strings** *(l)*

Sort the given list in the way that humans expect. http://blog.codinghorror.com/sorting-for-humans-natural-sort-order/

### 2.9 PDB Edit Bfactor/Occupancy

**rna_pdb_edit_occupancy_bfactor.py** - edit occupancy or bfactor in PDB file.

**Example:**

```bash
rna_pdb_edit_occupancy_bfactor.py --occupancy --select A:1-40,B:1-22 --set-to 0 19_Bujnicki_Human_4_rpr_n0-000001.pdb
```

```bash
rna_pdb_edit_occupancy_bfactor.py --occupancy --select A:1-2 --select-atoms P+C4' --set-to 10 -o test_data/3w3s_homologymodel_out.pdb
```

### Usage:

```bash
```
Positional arguments:

file file

Options:

--bfactor=False set bfactor
--occupancy=False set occupancy
--select get chain, e.g A:1-10, works also for multiple chainse.g A:1-40,B:1-22
--set-to=1 set value to, default is 1
--set-not-selected-to=0 set value to, default is 0
-o, --output file output
--verbose=False be verbose
--select-atoms select only given atoms can be only one atom, e.g. P+ for prims, e.g. P+C4'

Change occupancy or bfactor of pdb file.

Load the structure, and first set everything to be set_not_selected_to and then set selected to set_to.

Parameters

- **txt** (str) – A:1-10, selection, what to change
- **pdb** (str) – filename to read as an input
- **pdb_out** (str) – filename to save an output
- **bfactor** (bool) – if edit bfactor
- **occupancy** (bool) – if edit occupancy
- **set_to** (float) – set to this value, if within selection
- **set_not_selected_to** (float) – set to this value, if not within selection
- **select_atoms** (str) – P, P+C4, use + as a separator
- **v** (bool) – be verbose

Returns if OK, save an output to pdb_out

Return type bool
Warning: this function requires BioPython

## 2.10 Add chain to a file

Example:

```
./rna_add_chain.py -c X ../../input/1msy_rnakbmd_decoy999_clx_noChain.pdb > ../../
˓→output/1msy_rnakbmd_decoy999_clx_noChain_Xchain.pdb
```

From:

<table>
<thead>
<tr>
<th>ATOM</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>O5'</td>
<td>U</td>
<td>1</td>
<td>42.778</td>
<td>25.208</td>
</tr>
<tr>
<td></td>
<td>C5'</td>
<td>U</td>
<td>1</td>
<td>42.780</td>
<td>26.630</td>
</tr>
<tr>
<td></td>
<td>C4'</td>
<td>U</td>
<td>1</td>
<td>42.080</td>
<td>27.526</td>
</tr>
<tr>
<td></td>
<td>O4'</td>
<td>U</td>
<td>1</td>
<td>43.013</td>
<td>28.044</td>
</tr>
<tr>
<td></td>
<td>C1'</td>
<td>U</td>
<td>1</td>
<td>42.706</td>
<td>29.395</td>
</tr>
<tr>
<td></td>
<td>N1</td>
<td>U</td>
<td>1</td>
<td>43.857</td>
<td>30.305</td>
</tr>
<tr>
<td></td>
<td>C6</td>
<td>U</td>
<td>1</td>
<td>45.057</td>
<td>29.857</td>
</tr>
<tr>
<td></td>
<td>C5</td>
<td>U</td>
<td>1</td>
<td>46.025</td>
<td>30.676</td>
</tr>
<tr>
<td></td>
<td>C4</td>
<td>U</td>
<td>1</td>
<td>45.720</td>
<td>32.110</td>
</tr>
<tr>
<td></td>
<td>O4</td>
<td>U</td>
<td>1</td>
<td>46.444</td>
<td>32.975</td>
</tr>
</tbody>
</table>

to:

<table>
<thead>
<tr>
<th>ATOM</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>O5'</td>
<td>U</td>
<td>1</td>
<td>42.778</td>
<td>25.208</td>
</tr>
<tr>
<td></td>
<td>C5'</td>
<td>U</td>
<td>1</td>
<td>42.780</td>
<td>26.630</td>
</tr>
<tr>
<td></td>
<td>C4'</td>
<td>U</td>
<td>1</td>
<td>42.080</td>
<td>27.526</td>
</tr>
<tr>
<td></td>
<td>O4'</td>
<td>U</td>
<td>1</td>
<td>43.013</td>
<td>28.044</td>
</tr>
<tr>
<td></td>
<td>C1'</td>
<td>U</td>
<td>X</td>
<td>1</td>
<td>42.706</td>
</tr>
<tr>
<td></td>
<td>N1</td>
<td>U</td>
<td>X</td>
<td>1</td>
<td>43.857</td>
</tr>
<tr>
<td></td>
<td>C6</td>
<td>U</td>
<td>X</td>
<td>1</td>
<td>45.057</td>
</tr>
<tr>
<td></td>
<td>C5</td>
<td>U</td>
<td>X</td>
<td>1</td>
<td>46.025</td>
</tr>
<tr>
<td></td>
<td>C4</td>
<td>U</td>
<td>X</td>
<td>1</td>
<td>45.720</td>
</tr>
<tr>
<td></td>
<td>O4</td>
<td>U</td>
<td>X</td>
<td>1</td>
<td>46.444</td>
</tr>
</tbody>
</table>

```

```

```

```

```

```python
rna_tools.tools.misc.rna_add_chain.get_parser()

```

usage: rna_add_chain [-h] [-c CHAIN] file

Positional arguments:

  file   file

Options:

  -c, --chain   a new chain, e.g. A

## 2.11 Measure distance between atoms

This is a quick and dirty method of comparison two RNA structures (stored in pdb files). It measures the distance between the relevant atoms (C4') for nucleotides defined as “x” in the sequence alignment.

author: F. Stefaniak, modified by A. Zyla, supervision of mmagnus
usage: pdbs_measure_atom_dists [-h] [-v] seqid1 seqid2 alignfn pdbfn1 pdbfn2

Positional arguments:

  seqid1 : seq1 id in the alignment
  seqid2 : seq2 id in the alignment
  alignfn : alignment in the Fasta format
  pdbfn1 : pdb file1
  pdbfn2 : pdb file2

Options:

  -v=False, --verbose=False  increase output verbosity

This is a quick and dirty method of comparison two RNA structures (stored in pdb files). It measures the distance between the relevant atoms (C4') for nucleotides defined as “x” in the sequence alignment.

author: F. Stefaniak, modified by A. Zyla, supervision of mmagnus

rna_tools.tools.pdbs_measure_atom_dists.pdbs_measure_atom_dists.find_core(seq_with_gaps1, seq_with_gaps2)

Parameters

  • seq_with_gaps1 (str) – a sequence 1 from the alignment
  • seq_with_gaps1 – a sequence 2 from the alignment

Usage:


>>> find_core('GUUCAG-------------------UGAC-', 'CUUCGCAGCCAUUGCACUCCGGCUGCGAUG')
'xxxxxx-------------------xxxx-'

Returns core="xxxxxx———-xxxx-"

rna_tools.tools.pdbs_measure_atom_dists.pdbs_measure_atom_dists.get_parser()

rna_tools.tools.pdbs_measure_atom_dists.pdbs_measure_atom_dists.get_seq(alignfn, seqid)

Get seq from an alignment with gaps.

Parameters

  • alignfn (str) – a path to an alignment
  • seqid (str) – seq id in an alignment

Usage:


>>> get_seq('test_data/ALN_OBJ1_OBJ2.fa', 'obj1')
SeqRecord(seq=SeqRecord(seq=Seq('GUUCAG-------------------UGAC-', SingleLetterAlphabet()), id='obj1', name='obj1', description='obj1', id=<unknown id>, name='<unknown name>', dbxrefs=[], description='<unknown description>'))

Returns SeqRecord
 Args: structure (pdb): PDB Biopython object: with a pdb structure

   Returns a list of coords for atoms
   structure1realNumber: a list of residues

Return type struct1dict

2.11. Measure distance between atoms

rna_tools.tools.pdbs_measure_atom_dists.pdbs_measure_atom_dists.map_coords_atom(structure)

Open pdb with Biopython.

Parameters pdbfn (str) – a path to a pdb structure

Returns with a pdb structure

Return type PDB Biopython object
3.1 RNA Sequence

RNA Sequence with secondary structure prediction methods.

This tool takes a given sequence and returns the secondary structure prediction provided by 5 different tools: RNAfold, RNAsubopt, ipknot, contextfold and centroid_fold. You must have these tools installed. You don’t have to install all tools if you want to use only one of the methods.

It’s easy to add more methods of your choice to this class.

3.1.1 Installation

Depends on what tools you want to use, follow the instructions below.

3.1.1.1 ContextFold

https://www.cs.bgu.ac.il/~negevcb/contextfold/

needs Java. Try this on Ubuntu 14-04 https://askubuntu.com/questions/521145/how-to-install-oracle-java-on-ubuntu-14-04 Single chain only!

3.1.1.2 ViennaRNA

https://www.tbi.univie.ac.at/RNA/

For OSX install from the binary Installer from the page.
3.1.3 ipknot OSX

https://github.com/satoken/homebrew-rnatools

If one encounters a problem:

```
[mm] Desktop$ /usr/local/opt/bin/ipknot
  dyld: Library not loaded: /usr/local/opt/glpk/lib/libglpk.40.dylib
        Referenced from: /usr/local/opt/bin/ipknot
        Reason: image not found
[1] 51654 abort /usr/local/opt/bin/ipknot
```

the solution is:

```
brew install glpk # on OSX
```

3.1.4 RNA Structure

http://rna.urmc.rochester.edu/


Download http://rna.urmc.rochester.edu/RNAstructureDownload.html and untar it in `<RNA_PDB_TOOLS>/opt/RNAstructure/`. run `make`, the tools will be compiled in a folder `exe`. Set up `DATPATH` in your `bashrc` to `<RNA_PDB_TOOLS>/opt/RNAstructure/data_tables DATAPATH=/home/magnus/work/src/rna-pdb-tools/opt/RNAstructure/data_tables/` (read more http://rna.urmc.rochester.edu/Text/Thermodynamics.html). RNAstructure can be run with SHAPE restraints, read more http://rna.urmc.rochester.edu/Text/File_Formats.html#Constraint about the format. The file format for SHAPE reactivity comprises two columns. The first column is the nucleotide number, and the second is the reactivity. Nucleotides for which there is no SHAPE data can either be left out of the file, or the reactivity can be entered as less than -500. Columns are separated by any white space.

3.1.5 MC-Sym

3.1.2 FAQ

• Does it work for more than one chain?? Hmm.. I think it’s not. You have to check on your own. –magnus

3.1.3 TIPS

Should you need to run it on a list of sequences, use the following script:

```
from rna_tools import Seq
f = open("listOfSequences.fasta")
for line in f:
    if line.startswith('>'):
        print line,
    else:
        print line,
        s = Seq.Seq(line.strip()) # module first Seq and class second Seq #without strip
        #this has two lines
        print s.predict_ss(method="contextfold"),
        #print s.predict_ss(method="centroid_fold")
```
Todo:

- This class should be renamed to RNASeq and merged with RNASeq class from RNAalignment

```python
todo
exception rna_tools.Seq.MethodNotChosen
class rna_tools.Seq.RNASequence(seq, ss='', name='rna_seq')
    RNASequence.

Usage:

>>> seq = RNASequence("CCCCUUUUGGGG")
>>> seq.name = 'RNA03'
>>> print(seq.predict_ss("RNAfold", constraints="{{{{....}}}"")
>RNA03
CCCCUUUUGGGG

(eval (ss='', no_dangling_end_energies=False, verbose=False)
Evaluate energy of RNA sequence.

Parameters

- **ss** (optional)
- **no_dangling_end_energies** (Boolean)
- **verbose** (Boolean)

Returns

Energy (float)

The RNAeval web server calculates the energy of a RNA sequence on a given secondary structure. You can use it to get a detailed thermodynamic description (loop free-energy decomposition) of your RNA structures.

Simply paste or upload your sequence below and click Proceed. To get more information on the meaning of the options click the help symbols. You can test the server using this sample sequence/structure pair.

An equivalent RNAeval command line call would have been:

RNAeval -v -d0 < input.txt

Read more: <http://rna.tbi.univie.ac.at/cgi-bin/RNAWebSuite/RNAeval.cgi>

get_foldability (ss='', verbose=False)
Calculate foldability based on EntRNA.

Steps:

- parse SS into basepairs,
- calculate foldability

Configuration:

- Set ENTRNA_PATH to the folder where ENTRNA_predict.py is.

Cmd wrapper in here:

```bash
python ENTRNA_predict.py --seq_file pseudoknotted_seq.txt --str_file pseudoknotted_str.txt
```

**predict_ss** *(method='RNAfold', constraints='', enforce_constraint=False, shapefn='', explore='', verbose=0)*

Predict secondary structure of the seq.

**Parameters**

- **method**
- **constraints**
- **shapefn**(str) – path to a file with shape reactivites
- **verbose**(boolean)

It creates a seq fasta file and runs various methods for secondary structure prediction. You can provide also a constraints file for RNAfold and RNAsubopt.

Methods that can be used with contraints: RNAsubopt, RNAfold, mcfold.

Methods that can be used with SHAPE contraints: RNAfold.

**ContextFold**

Example:

```java
$ java -cp bin contextFold.app.Predict in:CCCCUUUGGGGG
(CCCCCUGGG)
```

It seems that a seq has to be longer than 9. Otherwise:

```java
$ java -cp bin contextFold.app.Predict in:UUUUGGG
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10
# this is OK
$ java -cp bin contextFold.app.Predict in:CCCCUUUGGG
(CCCCUGGG)
```

**RNAstructure**

Example:

```python
>>> seq = RNASequence("GGGGUUUCCC")
>>> print(seq.predict_ss("rnastructure"))
> ENERGY = -4.4  rna_seq
GGGGUUUCCC
(......)
```

and with the shape data:

```python
>>> print(seq.predict_ss("rnastructure", shapefn="data/shape.txt"))
> ENERGY = -0.2  rna_seq
GGGGUUUCCC
(......)
```

the shape data:
You can easily see that the first G is unpaired right now! The reactivity of this G was set to 10. Worked!

**MC-Fold**

MC-Fold uses the online version of the tool, this is very powerful with constraints:

```
acr_seq
acucggcuaggcaguauaaauagcgucagccuagcgcgucagccuagcccuucuugggcugggcgaagggucggg
((((.........))))........(((..................((.))))))(((((........))))........(((..................((.))))))
curl -Y 0 -y 300 -F "pass=lucy" -F mask="((((.........))))........(((..................((.))))))(((((........))))........(((..................((.))))))" -F sequence="acucggcuaggcaguauaaauagcgucagccuagcgcgucagccuagcccuucuugggcugggcgaagggucggg" -F "https://www.major.iric.ca/cgi-bin/MC-Fold/mcfold.static.cgi"
mcfold::energy best dynamics programming: -53.91 (-53.91, '((((.........))))........(((..................((.))))))(((((........))))........(((..................((.))))))')
curl -Y 0 -y 300 -F "pass=lucy" -F mask="((((.........))))........(((..................((.))))))(((((........))))........(((..................((.))))))" -F sequence="acucggcuaggcaguauaaauagcgucagccuagcgcgucagccuagcccuucuugggcugggcgaagggucggg" -F "https://www.major.iric.ca/cgi-bin/MC-Fold/mcfold.static.cgi"
mcfold::energy best dynamics programming: -34.77 (-34.77, '((((.........))))........(((..................((.))))))(((((........))))........(((..................((.))))))')
```
explore

The sub-optimal search space can be constrained within a percentage of the minimum free energy structure, as MC-fold makes use of the Waterman-Byers algorithm [18, 19]. Because the exploration has an exponential time complexity, increasing this value can have a dramatic effect on MC-Fold’s run time.


3.2 RNA Secondary Structure

Secondary structure analysis

```python
rna_tools.Seq.load_fasta_ss_into_RNAseqs(fn, debug=True)
```

3.2.1 RNA Secondary Structure

exception rna_tools.SecondaryStructure.ExceptionOpenPairsProblem

```python
rna_tools.SecondaryStructure.draw_ss(title, seq, ss, img_out, resolution=4, verbose=False)
```

`draw_ss` method can be used to draw secondary structures using VARNa. It returns None if everything is OK, or raises an exception otherwise.

Usage:

```python
>>> seq = 'GGAAACC'
>>> ss = '((...))'
>>> img_out = 'output/demo.png'
>>> draw_ss('rna', seq, ss, img_out)
>>> print('Made %s' % img_out)
Made output/demo.png
```
rna

Can be used with http://geekbook.readthedocs.io/en/latest/rna.html

```python
rna_tools.SecondaryStructure.parse_vienna_to_pairs(ss, remove_gaps_in_ss=False)
```

Parse Vienna (dot-bracket notation) to get pairs.

**Parameters**

- `ss` *(str)* – secondary stucture in Vienna (dot-bracket notation) notation
- `remove_gaps_in_ss` *(bool)* – remove - from ss or not, design for DCA (tpp case
  

```python
ss = "((((((((((.(((((.....))))))......------)....." works with pk
of the first level, [[]]
```

**Returns** (pairs, pairs_pk)

**Return type** list of two lists

**Examples:**

```python
>>> parse_vienna_to_pairs('((..))')
[[1, 6], [2, 5]]

>>> parse_vienna_to_pairs('(([[[]])])]
[[1, 6], [2, 5], [3, 8], [4, 7]]

>>> parse_vienna_to_pairs('((--))')
[[1, 6], [2, 5], []]

>>> parse_vienna_to_pairs('((--))', remove_gaps_in_ss=True)
[[1, 4], [2, 3], []]

>>> parse_vienna_to_pairs('((((......')
Traceback (most recent call last):
  File "/usr/lib/python2.7/doctest.py", line 1315, in __run
    compileflags, 1)
in test.globs
  File "<doctest __main__.parse_vienna_to_pairs[4]>", line 1, in <module>
```

3.2. RNA Secondary Structure
parse_vienna_to_pairs('((((......')
File "./SecondaryStructure.py", line 106, in parse_vienna_to_pairs
raise ExceptionOpenPairsProblem('Too many open pairs (()) in structure')
ExceptionOpenPairsProblem: Too many open pairs (()) in structure

3.2.1 rna_dot2ct.py

The output file is <input-file>.ct


usage: rna_dot2ct.py [-h] [-v] file

Positional arguments:
  file          Input is:   >seq aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
                   ((...(((((.....))...))))))

Options:
  -v=False, --verbose=False  be verbose

3.2.2 Secondary structure format conversion

rna_convert_pseudoknot_formats

Run this as:

```python
python rna-pk-simrna-to-one-line.py test_data/simrna.ss
```

Convert:

```
> a
....(((.(((((....))))..(((((.(...........))))....))).))))....(((.......))
--------------------------------------------------------------------------------
```

to:

```
> a
....((.(((((....))))..(((((.[[[.]]]]))....))).)))(((.......)))
```

and:

```
>2 chains
(((((((()))))))))........(.((....(....)...((. .))).))
.....((((((((())))))))))........ ...........
```

to:

```
>2 chains
(((((([[[[[[))))))........(.((....([]))))[[[.]]])..((. .))...))
```

and:
to:

```bash
> b
..([.[.[.[.(......)].]]).
```  

and it works with VARNA:

![VARNA GUI](image)

Convert a secondary structure with a pk to the SimRNA format:

```bash
rna_convert_pseudoknot_formats qit:(master) python rna_ss_pk_to_simrna.py test_data/
˓ss_with_pk.ss
((((([][[[[[]]]]]))))........((([[[[[[[]]]]]]]))...(((. .)))...))

(((((......))))........((....(.......)..(((. .)))...))).

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

```python
rna_tools.tools.rna_convert_pseudoknot_formats.rna_ss_pk_to_simrna.get_multiple_lines(ss)
rna_tools.tools.rna_convert_pseudoknot_formats.rna_ss_pk_to_simrna.get_parser()
rna_tools.tools.rna_convert_pseudoknot_formats.rna_ss_pk_to_simrna.is_pk(ss)
```
3.3 Search

3.3.1 Blast PDB

A super-simple wrapper around Blast on the PDB db (online).

class rna_tools.BlastPDB.BlastPDB(seq)
BlastPDB - run Blast online on the PDB database.

This can be used in Jupiter based RNA notebooks, e.g. https://github.com/mmagnus/rna-pdb-tools/blob/master/rp18.ipynb

Usage:

```python
>>> p = BlastPDB('GGGUCAGGCCGGCGAAAGUCGCCACAGUUUGGGAAAGCUGUGCAGCCUGUAACCCCCCCACGAAAGUGGG')
>>> p.search()
>>> p.result
u'<HTML>
<TITLE>BLAST Search Results</TITLE>
...</html>
```

Parameters seq - string

search()
Search online the seq.

3.3.2 Rfam Search

A super-simple wrapper around cmscan (Infernal) on local Rfam database.

class rna_tools.RfamSearch.RfamSearch
RfamSearch (local).

Rfam is a collection of multiple sequence alignments and covariance models representing non-coding RNA families. Rfam is available on the web http://rfam.xfam.org/. The website allow the user to search a query sequence against a library of covariance models, and view multiple sequence alignments and family annotation. The database can also be downloaded in flatfile form and searched locally using the INFERNAL package (http://infernal.wustl.edu/). The first release of Rfam (1.0) contains 25 families, which annotate over 50 000 non-coding RNA genes in the taxonomic divisions of the EMBL nucleotide database.

Infernal (“INFERence of RNA ALignment”) is for searching DNA sequence databases for RNA structure and sequence similarities. It is an implementation of a special case of profile stochastic context-free grammars called covariance models (CMs). A CM is like a sequence profile, but it scores a combination of sequence consensus and RNA secondary structure consensus, so in many cases, it is more capable of identifying RNA homologs that conserve their secondary structure more than their primary sequence.

Infernal cmscan is used to search the CM-format Rfam database.

Setup:

- download the database from ftp://ftp.ebi.ac.uk/pub/databases/Rfam/CURRENT (file: Rfam.cm.gz, ~30mb)
- install http://eddylab.org/infernal/
- set up RFAM_DB_PATH in the config file of rna-tools.
- compress Rfam.cm
Example of compressing the database:

```bash
$ cmpress Rfam.cm
Working... done.
Pressed and indexed 3016 CMs and p7 HMM filters (3016 names and 3016 accessions).
Covariance models and p7 filters pressed into binary file: Rfam.cm.i1m
SSI index for binary covariance model file: Rfam.cm.i1i
Optimized p7 filter profiles (MSV part) pressed into: Rfam.cm.i1f
Optimized p7 filter profiles (remainder) pressed into: Rfam.cm.i1p
```


**cmscan** *(seq, verbose=False)*
Run cmscan on the seq.

Usage:

```python
>>> seq = RNASequence("GGCGCGGCACCGUCCGCGGAACAAACGG")
>>> rs = RfamSearch()
>>> hit = rs.cmscan(seq)
>>> print(hit)
# cmscan :: search sequence(s) against a CM database...
```

**Parameters**  
*seq* – string  
**Returns**  
result  
**Return type**  
string

**exception**  
rna_tools.RfamSearch.RfamSearchError

### 3.4 RNA Alignment

RNAalignment - a module to work with RNA sequence alignments.

To see a full demo what you can do with this util, please take a look at the jupiter notebook (https://github.com/mmagnus/rna-pdb-tools/blob/master/rna_tools/tools/rna_alignment/rna_alignment.ipynb)

Load an alignment in the Stockholm:

```python
alignment = ra.RNAalignment('test_data/RF00167.stockholm.sto')
```

or fasta format::

```python
import rna_alignment as ra
alignment = ra.fasta2stokholm(alignment.fasta)
alignment = ra.RNAalignment
```

Parameters of the alignment:

```python
print(alignment.describe())
```

Consensus SS:

```python
print(alignment.ss_cons_with_pk)
```
Get sequence/s from the alignment:

```python
>>> seq = a.io[0]
```

### 3.4.1 RNASeq

```python
class rna_tools.tools.rna_alignment.rna_alignment.RNASeq(id, seq, ss=None)
```

**Parameters**

- `id` *(str)* – id of a sequence
- `seq` *(str)* – seq, it be uppercased.
- `ss` *(str)* – secondary structure, default None

```python
seq_no_gaps
str – seq.replace('-', '')
```

```python
ss_no_gaps
str – ss.replace('-', '')
```

```python
draw_ss (title=", verbose=False, resolution=1.5)
```

Draw secondary structure of RNA with VARNA.

VARNA: Visualization Applet for RNA A Java lightweight component and applet for drawing the RNA secondary structure

Cite: VARNA: Interactive drawing and editing of the RNA secondary structure Kevin Darty, Alain Denise and Yann Ponty Bioinformatics, pp. 1974-197, Vol. 25, no. 15, 2009

http://varna.lri.fr/
**get_conserved** *(consensus, start=0, to_pymol=True, offset=0)*

Start

UCGGGGUCCUUCUUGCGUG———AAGGC-
UGAGAAUACCCGU———AUCACCUG-AUCUGAU-AAUGC
XXX...XXXXXXXXGXXGXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX——
XXX...XCUGAGAXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX—XXX-
XXX...ACXUG

**get_distance_to** *(nseq)*

Get distance of self.seq to nseq.

**get_ss_std** *

**remove_columns** *(to_remove)*

Indexing from 0

**remove_gaps** *(check_bps=True, only_canonical=True, allow_gu=True)*

Remove gaps from seq and secondary structure of the seq.

**Parameters**

- **check_bps** *(bool)* – fix mistakes as
- **only_canonical** *(bool)* – keep in ss only pairs GC, AU
- **allow_gu** *(bool)* – keep in ss also GU pair

A residue “paired” with a gap.
paired with any residues (in the blue circle). If yes, then this residues is unpair (in this case).
if `only_canonical` (by default) is True then only GC, AU can be paired.

If `allow_gu` is False (be default is True) then GU pair is also possible.
If you provide seq and secondary structure such as:

```
GgCcGggG.GcggG.cc.u.aUACAuACC.GaAA.GGGAAUAaggCc.gGCc.gu......CU.......
→uuqucgGUuUcaAgCccCCgGcCcCceuuu
(((((((((....(.(.(..................(................))).((((.((..............
→.))))))))))))).....)))))))))........
```
gaps will be remove as well.

**ss_to_bps**

Convert secondary structure into a list of basepairs.

- **Returns** a list of base pairs, e.g. 
  
  ```
  [[0, 80], [1, 79], [2, 78], [4, 77], [6, 75], [7, 74], ...]
  ```

- **Return type** bps (list)

### 3.4.2 RNAalignment

**class** rna_tools.tools.rna_alignment.rna_alignment.RNAAlignment *(fn='*, fetch='*)

RNA alignment - adapter class around BioPython to do RNA alignment stuff


```python
>>> a = RNAAlignment('test_data/RF00167.stockholm.sto')
>>> print(a.tail())
>>> print(a.ss_cons)
```
- **fn**(str) – Filename
- **io**(Bio.AlignIO) – AlignIO.read(fn, “stockholm”)
- **lines**(list) – List of all lines of fn
- **seqs**(list) – List of all sequences as class:RNASeq objects
- **rf**(str) – ReFerence annotation, the consensus RNA sequence

Read more:

**Warning:** fetch requires urllib3

align_seq(seq)
Align seq to the alignment.
Using self.rf.

Parameters
- **seq**(str) – sequence, e.g., `-GGAGAGUA-GAUGAUUCGCGUUAAGUGUGUGA-AUGGGAUGUC.

Returns
- seq that can be inserted into alignment, `.GG.AGAGUA-GAUGAUUCGCGUUA` ! . ->`

Return type str

copy_ss_cons_to_all(verbos=False)
copy_ss_cons_to_all_editing_sequence(seq_id, before, after)
Change a sequence’s sec structure.

Parameters
- **seq_id** – string, sequence id to change, eg: AE009948.1/1094322-1094400
- **before** – string, character to change from, eg: ,
- **after** – string, character to change to, eg: .

**Warning:** before and after has to be one character long

describe()
Describe the alignment.

> print(a.describe()) SingleLetterAlphabet() alignment with 13 rows and 82 columns

find_core(ids=None)
Find common core for ids.

3.4. RNA Alignment
Fig. By core, we understand columns that have all homologous residues. The core is here marked by $x$.

**Parameters**

- `id` – list, ids of seq in the alignment to use

**find_seq**(seq, verbose=False)

Find seq (also subsequences) and reverse in the alignment.

**Parameters**

- `seq` (str) – seq is upper()
- `verbose` (bool) – be verbose

```python
seq = "ggaucgcugacccgaaaggggcgggggacccagaaauggggcgaaucucuuccgaaaggaagagguaggguuacuccgacccgagcccgucagcuaaccucgcaagcguccgaaggauc"

hit = a.find_seq(seq, verbose=False)
ggaucgcugacccgaaaggggcgggggacccagaaauggggcgaaucucuuccgaaaggaagagguaggguuacuccgacccgagcccgucagcuaaccucgcaagcguccgaaggauc
```

**find_seq_exact**(seq, verbose=False)

Find seq (also subsequences) and reverse in the alignment.

**Parameters**

- `seq` – string, seq, seq is upper()
- `verbose` – boolean, be verbose or not

**format_annotation**(t)

**get_clean_ss**(ss)

**get_distances()**

Get distances (seq identity) all-vs-all.

With BioPython.

```
blastn: Bad alphabet 'U' in sequence 'AE008922.1/409481-409568' at position '7' only for DNA?
```
get_gc_rf()
    Return (str) #=GC RF or ‘’ if this line is not in the alignment.

get_seq(seq_id)

get_seq_ss(seq_id)

get_seq_with_name(seq_name)

g_shift_seq_in_align()
    RF_cons vs ‘#=GC RF’ ???

get_ss_cons()
    Returns SS_cons_pk line or None if there is now SS_cons_pk.

get_ss_cons_pk()
    Returns SS_cons_pk line or None if there is now SS_cons_pk:

get_ss_remove_gaps(seq, ss)
    Parameters
    • seq – string, sequence
    • ss – string, ss

UAU-AACAUAUAUUUGACAAUAUGG-GUCAUAA-GUUUUCACC-GUAAAUAUUCU—GACUAUG-UAUA- (((..(((.((((((________)))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))

get_the_closest_seq_to_ref_seq(verbose=False)
    Example:

>>> a = RNAalignment("test_data/RF02221.stockholm.sto")
>>> a.get_the_closest_seq_to_ref_seq()
AF421314.1/431-344

head()

map_seq_on_align(seq_id, resis, v=True)
    Parameters
    • seqid – seq_id, ‘CP000721.1/2204691-2204775’
    • resis – list resis, [5,6]

maps:

[5, 6, 8]
CAC-U
CAC-U-
CAC-U-UA
[4, None, 6]

map_seq_on_seq(seq_id, seq_id_target, resis, v=True)
    Parameters
    • seq_id – seq_id, ‘AAML04000013.1/228868-228953’
    • seq_id_target – seq_id of target, ‘CP000721.1/2204691-2204778’
map:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[4, 5, 6]</td>
<td>UAU-A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UAU-AA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UAU-AAC</td>
<td></td>
</tr>
<tr>
<td>[5, 6, 7]</td>
<td>CAC-U</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CAC-U-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CAC-U-U</td>
<td></td>
</tr>
<tr>
<td>[4, None, 5]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

plot (plot_fn='rchie.png')

reload_alignment()

remove_empty_columns (verbose=False)
Remove empty columns in place.

Example:

```python
>>> a = RNAalignment("test_data/zmp.stk")
>>> print(a)
SingleLetterAlphabet() alignment with 6 rows and 319 columns
---ACCUUGCGCACUGCGGCAUCC---------------...AAU CP001644.1/756294-756165
--GCUCUUGCGACUUGCGGCAUUG-------------...GAA CU234118.1/352539-352459
UGAGUUUUCUGCACUGCGGCAUUA------------...CUG BAAV01000055.1/2897-2982
GCCCGGUUCUGACUGCCGCUCGUAGU-----------------...CGA CP000927.1/5164264-5164343
-----GGGUCGUGACUGCGGCAACA---------------...--- zmp
UCACCCCUGCGACUGCGGCAUA-----------------...GUU AP009385.1/718103-718202
```
3.4.3 rna_alignment_get_species.py

The output you simply get from the screen, save it it to a file.

Example:

```
rna_alignment_get_species.py RF00004.stockholm.stk
# STOCKHOLM 1.0
Sorex-araneus-(European-shrew)   AUCGCU-UCU----CGGCC--UUU-U
```

Examples 2:

```
[dhcp177-lan203] Desktop$ rna_alignment_get_species.py u5_rfam_u5only.stk --verbose
# STOCKHOLM 1.0
#=GF WK U5_spliceosomal_RNA
#=GF NC 39.90
#=GF RT The spliceosomal snRNAs of Caenorhabditis elegans.
#=GF TC 40.00
#=GF RN [3]
#=GF RM 2339054
#=GF AU Gardner PP; 0000-0002-7808-1213
#=GF CC methylation.
#=GF CB cmcalibrate --mpi CM
#=GF DR GO; 0046540; U4/U6 x U5 tri-snRNP complex;
#=GF ID U5
#=GF SS Published; PMID:2339054; Griffiths-Jones SR
#=GF RA Thomas J, Lea K, Zucker-Aprison E, Blumenthal T
#=GF SQ 180
#=GF SM cmsearch --cpu 4 --verbose --nohmmonly -E 1000 -Z 549862.597050 CM SEQDB
#=GF DE U5 spliceosomal RNA
#=GF AC RF00020
#=GF SE Zwieb C, The uRNA database, PMID:9016512; PMID:18390578
#=GF GA 40.00
#=GF BM cmbuild -F CM SEED
#=GF TP Gene; snRNA; splicing;
Bos-taurus-(cattle)   GAUC-GUAUAAAUCUUUCGCCUUUUACUAAAGA-
-->UUUCGG----UGG-A--GA-G
Sorex-araneus-(European-shrew)   GAUC-GUAUAAAUCUUUCGCCUUUUACUAAAGA-
-->UUUCGG----UGG-A--GA-G
Ictidomys-tridecemlineatus-(thirteen-lined-ground- GAUC-GUAUAAAUCUUUCGCCUUUUACUAAAGA-
-->UUUCGG----UGG-A--GA-G
Monodelphis-domestica-(gray-short-tailed-opossum)  GAUC-GUAUAAAUCUUUCGCCUUUUACUAAAGA-
-->UUUCGG----UGG-A--GA-G
```

3.4. RNA Alignment
Note: This code has way more code than the name of the script says. This is customized script based on some script that did way more.

Usage:
```
usage: rna_alignment_get_species [-h] [-v] [--debug] [--id-width ID_WIDTH]
                                [--evo-mapping EVO_MAPPING]
                                [--evo-mapping-default] [--one] [--u5]
                                [--calc-energy] [--osfn OSFN]
                                alignment
```

Positional arguments:
- alignment: Undocumented

Options:
- -v=False, --verbose=False: be verbose
- --debug=False: Undocumented
- --id-width=50: Undocumented
- --evo-mapping: Undocumented
- --evo-mapping-default=False: Undocumented
- --one=False: Undocumented
- --u5=False: Undocumented
- --calc-energy=False: Undocumented
- --osfn: Undocumented

3.4.4 rna_alignment_calc_energy.py

Calculate energy (.cet) format:
```
UGGC-CCCUGCCGCAA-GGAUGACA
(((..((((......)).))..))))
(((..(((((***)).))..))))
```

Examples:
```
$ rna_alignment_calc_energy.py --template alignments/u6-lower.cet alignments/u6-only-
--RemovedGapped.stk -v
  --loop-upper guaa --loop-lower guaa
  --loop-upper-cst '(..)' --loop-lower-cst '(..)' calc-energy2.py --template u6atac-template.txt u6atac_u6only.sto -v
./calc-energy2.py --template alignments/u6-lower.cet --one alignments/u6-lower-stem-
--only.sto
```
Takes cet files (calc-energy-templats).

```bash
$ rna_alignment_calc_energy.py --template test_data/u6-lower.cet --one test_data/u6-only.stk -v 
--loop-seq test_data/u6-only-loop-seq-u6-lower N/A% (0 of 182) | Elapsed Time: 0:00:00 ETA:
```

I

```text
id low_energy low_seq low_ss up_energy up_seq up_ss
0 AB010698.1/46467-46488 -10.64 UGGugcgcaACA (((((..))))) -9.6 CCCugcgcaAGG (((((..)))))) Done:
```

Is

**domains have 5451 elements.** 10:47:16 up 141 days, 26 min, 0 users, load average: 1.45, 1.30, 1.56

Score: -999.000 GAACAUGUUCUUGCCUUUACCAGAACCAUGUGGUGUUUGUUG Total number of MB structures with 3 stems: 16041 (overlaps: 0, energy: 33585) <pre>&lt;/p&gt;&lt;p&gt;&lt;/p&gt;&lt;h2&gt;Sorting the structures... &lt;p&gt;&lt;/h2&gt;&lt;p&gt;&lt;/p&gt;&lt;p&gt;Filtered and Sorted solutions:&lt;p&gt;&lt;p&gt;&lt;/p&gt;&lt;/h2&gt;&lt;pre&gt;
&lt;/pre&gt;&lt;/p&gt;&lt;p&gt;&lt;/p&gt;&lt;ca HREF="http://biwww2.informatik.uni-freiburg.de/Software/MARNA/index.html" target="_blank">MARNA&lt;/a&gt;-formatted:&lt;p&gt;&lt;p&gt;&lt;/p&gt;&lt;h2&gt;&lt;p&gt;&lt;/pre&gt;
GAACAUGUUCUUGCCUUUACCAGAACCAUGUGGUGUUUG ((((((((...)))))))(((((...))))))) -33.20 (-0.69 (((((...))))))) -33.17 (-0.69) (((((...))))))) -32.40 (+0.00)
```

Backtracking with 2 variables (stems): domains have 5451 elements.

10:47:16 up 141 days, 26 min, 0 users, load average: 1.45, 1.30, 1.56

Score: -999.000 GAACAUGUUCUUGCCUUUACCAGAACCAUGUGGUGUUUGUUG Total number of MB structures with 2 stems: 9555 (overlaps: 0, energy: 16582) <pre>&lt;/h2&gt;&lt;/p&gt;&lt;p&gt;&lt;/p&gt;&lt;p&gt;Filtered and Sorted solutions:&lt;p&gt;&lt;p&gt;&lt;/p&gt;&lt;/h2&gt;&lt;pre&gt;
&lt;/pre&gt;&lt;/p&gt;&lt;p&gt;&lt;/p&gt;&lt;ca HREF="http://biwww2.informatik.uni-freiburg.de/Software/MARNA/index.html" target="_blank">MARNA&lt;/a&gt;-formatted:&lt;p&gt;&lt;p&gt;&lt;/p&gt;&lt;h2&gt;&lt;p&gt;&lt;/pre&gt;
GAACAUGUUCUUGCCUUUACCAGAACCAUGUGGUGUUUG ((((((((...)))))))(((((...))))))) -33.20 (-0.69 (((((...))))))) -33.17 (-0.69) (((((...))))))) -32.40 (+0.00)
```

```
usage: rna_alignment_calc_energy [-h] [--debug] [--one] [--method METHOD]
[--csv CSV] [--loop-seq]
[--template TEMPLATE] [--flanks FLANKS] [-v]
alignment
```

### Positional arguments:
- `alignment`: an alignment in the Stockholm format

### Options:
- `--debug=FALSE`: Undocumented
- `--one=FALSE`: one only for the first seq
- `--method=mcfold`: mcfold or rnastructure_CycleFold
- `--csv`: Undocumented
- `--loop-seq=FALSE`: Undocumented
- `--template`: Undocumented
- `--flanks`: GC be default

#### 3.4. RNA Alignment

51
3.4.5 rna_align_get_ss_from fasta.py

Input as a file:

```
>ade
GCU--UCAUAUUCCUAUGAUUGG--UUUGGGA-GUUUCUACCAAGAG--UUAAA--CUCUU---GAUUAUG--AAGU--
(((((((.((((((.((((((.)))))))))))))))))))))))))))))))))))))))))))))
```
to get:

```
>ade
GCUUCAUAUUCCUAUGAUUGGUUGGAGUUUCUACCAAGACCCUUAAACUUGAUUAUGAAAG
(((((((((.((((((.))))))))))))))))))))))))))))))))))))))))))))))))))
```

usage: rna_align_get_ss_from_fasta [-h] file

Positional arguments:

- file subsection of an alignment

3.4.6 rna_align_get_ss_from stk.py

Process an alignment in the Stockholm format to get sequences and secondary structures:

Example:

```
$ rna_align_get_ss_from_stk.py aligns/gmp_RF01786.stockholm_p75_pk.sto
AAOX0100007.1/31274-31356
AAGAAUAUAGAACACUGUGAUGAGCGGUUUUUAUUUGCACUUUAAACCGCUUGGAGUGACUAGUGCAGCGCCGCAAUGAUCUA
.(((((((.(((((.((((((.)))))))))))))))))))))..(((((((((.)))))))))))..))..)
CP000724.1/3560727-3560809
AAAAAUGUAGAGCAAAUGAACUGCAGGUAUACAUGGACGCCUUAAACUGCAGGGAUGUAGUGCGUAACCGACUAACAAUAAUU
.((.(((.((((.(((((((((((((.)))))))))))))))))))))))))..(((((((.)))))))))))..))..)
AACY023761929.1/1009-1091
AUAUUGUGGCGGCUUGAUGUGCCCUUUGAUCUGGUCUGGCGUGAGUGCAUAGCGCCGCAAUGCGAACGCGCGCGUGC
.((.(((((((((.(((((((((.)))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
```

usage: rna_align_get_ss_from_stk [-h] file

Positional arguments:

- file subsection of an alignment

3.4.7 rna_align_distance_to_seq.py

Calculate

"Process an alignment in the Stockholm format to get sequences and secondary structures:

Example:
usage: rna_align_distance_to_seq.py [-h] file output

Positional arguments:

    file           an alignment in the Stokholm format, the first seq will be used to calculate
distance to (#TODO pick any seq)

    output       csv pandas file

3.4.8 rna_align_foldability.py

Calculate statistics of foldability on an alignment.

The tool uses ENTRANA [1] to calculate, what the authors called, foldability (column: “foldability”) of a given
sequence into a given secondary structure.

Next, MC-Fold [2] is executed to calculate free energy (column: “mcsym”) on the sequence and the secondary struc-
ture obtained based on the alignment. The secondary structure is used as constraints.

The third used program is RNAfold from the Vienna package [3]. Also, in this case the secondary structure obtained
with rna-tools from the RNA alignment is used as constraints, columns: “mfe” (minimum free energy), “mfess”
(secondary structure for minimum free energy state), “cfe” (minimum free energy of centroid), “cfess” (secondary
structure for centroid, “diversity” (ensemble diversity), “efe” (free energy of the thermodynamic ensemble), “efess”
(secondary structure for the thermodynamic ensemble), “freq” (frequency of mfe structure in ensemble). RNAfold is also executed in with “--enforceConstraint” where the constraints are enforced. This run gives analogous values as the default RNAfold, to all RNAfold column “_enforce” is added.

The tool is able to calculate the distance Levenshtein (the difference between the two sequences)(column: “distance”) from the target sequence and all sequence in the alignment to test if there is a bias in the accuracy towards the most similar sequences.

Another tool used from the Vienna package is RNAeval. The tool calculates free energy for a given sequence and secondary structure.

The accuracy is expressed as the median of core RMSD of 10% the lowest core RMSD models for the given sequences.

The correlations:

<table>
<thead>
<tr>
<th>Correlation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>accuracy</td>
<td>1.000000</td>
</tr>
<tr>
<td>cfe</td>
<td>0.653813</td>
</tr>
<tr>
<td>foldability</td>
<td>0.622038</td>
</tr>
<tr>
<td>mfe</td>
<td>0.607340</td>
</tr>
<tr>
<td>efe</td>
<td>0.585077</td>
</tr>
<tr>
<td>diversity</td>
<td>0.404350</td>
</tr>
<tr>
<td>eval</td>
<td>0.349499</td>
</tr>
<tr>
<td>cfe_enforce</td>
<td>0.311744</td>
</tr>
</tbody>
</table>
The data:

We tested correlations between the above-mentioned statistics, and the highest correlation, 0.65 () was achieved to the centroid free energy calculated with RNAFold, which suggests that to some extent this metric could be used to pick sequence from the alignment to pick sequences that are more likely to fold.

However, this needs further investigation and the detailed analysis an all cases and more folded sequences.


Example:

```
$ python rna_align_foldability.py test_data/gmp_ref_sto test_data/gmp_foldability.csv
cfess_enforce distance diversity
0 ((((( .... (((((((((((((((((............)))))))... 1.00 3.96
1 (((((( .... (((((((((((((.(((((.......)))...... 0.69 5.56
2 0.73 3.84
3 0.69 5.92
4 (((.... (((((((..((......(((((.(((((.......)))....... 0.73 7.49
5 (((....... (((((.(((((.......)))))... 0.75 7.92
6 ....... (((((.(((..(...(((((((((..............))... 0.72 5.83
7 0.72 7.35
8 )))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))... 0.65 4.86
```

```
diversity_enforce efe efe_enforce
cfess 2.89 -14.77 -13.75
1 3.70 -19.52 -18.25
2 0.00 -15.41 0.00
3 0.00 -13.55 0.00
4 2.46 -8.58 -6.91
5 6.37 -20.72 -20.08
6 2.92 -11.87 -11.38
7 0.00 -14.59 0.00
8 3.83 -21.16 -20.64
```

3.4. RNA Alignment
### RNA Tools Documentation

#### Chapter 3. RNA Tools

<table>
<thead>
<tr>
<th>Length</th>
<th>McSym</th>
<th>McSym Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.0</td>
<td>39.73</td>
</tr>
<tr>
<td>2</td>
<td>8.0</td>
<td>35.40</td>
</tr>
<tr>
<td>3</td>
<td>8.0</td>
<td>36.11</td>
</tr>
<tr>
<td>4</td>
<td>8.0</td>
<td>37.37</td>
</tr>
<tr>
<td>5</td>
<td>8.0</td>
<td>43.59</td>
</tr>
<tr>
<td>6</td>
<td>8.0</td>
<td>42.95</td>
</tr>
<tr>
<td>7</td>
<td>8.0</td>
<td>35.55</td>
</tr>
<tr>
<td>8</td>
<td>8.0</td>
<td>43.58</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>McSym</th>
<th>McSym SS</th>
<th>McSym MFE</th>
<th>McSym MFE Enforce</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>13.9</td>
<td>12.9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18.0</td>
<td>17.3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>14.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7.2</td>
<td>-6.1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>18.6</td>
<td>-18.6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10.5</td>
<td>-10.5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>12.8</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>19.8</td>
<td>-19.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>McSym</th>
<th>McSym SS</th>
<th>McSym MFE</th>
<th>McSym MFE Enforce</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>13.9</td>
<td>12.9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18.0</td>
<td>17.3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>14.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7.2</td>
<td>-6.1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>18.6</td>
<td>-18.6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10.5</td>
<td>-10.5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>12.8</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>19.8</td>
<td>-19.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>McSym</th>
<th>McSym SS</th>
<th>McSym MFE</th>
<th>McSym MFE Enforce</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>13.9</td>
<td>12.9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18.0</td>
<td>17.3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>14.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7.2</td>
<td>-6.1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>18.6</td>
<td>-18.6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10.5</td>
<td>-10.5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>12.8</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>19.8</td>
<td>-19.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>McSym</th>
<th>McSym SS</th>
<th>McSym MFE</th>
<th>McSym MFE Enforce</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>13.9</td>
<td>12.9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18.0</td>
<td>17.3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>14.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7.2</td>
<td>-6.1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>18.6</td>
<td>-18.6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10.5</td>
<td>-10.5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>12.8</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>19.8</td>
<td>-19.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>McSym</th>
<th>McSym SS</th>
<th>McSym MFE</th>
<th>McSym MFE Enforce</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>13.9</td>
<td>12.9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18.0</td>
<td>17.3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>14.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7.2</td>
<td>-6.1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>18.6</td>
<td>-18.6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10.5</td>
<td>-10.5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>12.8</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>19.8</td>
<td>-19.8</td>
<td></td>
</tr>
</tbody>
</table>

Positional arguments:

file an alignment in the Stockholm format, the first seq will be used to calculate
distance to (#TODO pick any seq)

output csv pandas file

Options:

--all-stars=False this takes usully super long
--dev=False Undocumented
--skip-mcfold=False Undocumented
-v=False, --verbose=False be verbose

3.4.9 Random assignment of nucleotides

random_assignment_of_nucleotides.py - Random assignment of nucleotides for non-typical characters in the sequence
alignment (arg –alignfn or fasta file with sequneces (arg –seqfn)

<table>
<thead>
<tr>
<th>R</th>
<th>G</th>
<th>A</th>
<th>(purine)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>U</td>
<td>C</td>
<td>(pyrimidine)</td>
</tr>
<tr>
<td>K</td>
<td>G</td>
<td>U</td>
<td>(keto)</td>
</tr>
<tr>
<td>M</td>
<td>A</td>
<td>C</td>
<td>(amino)</td>
</tr>
<tr>
<td>S</td>
<td>G</td>
<td>C</td>
<td>(strong bonds)</td>
</tr>
<tr>
<td>W</td>
<td>A</td>
<td>U</td>
<td>(weak bonds)</td>
</tr>
<tr>
<td>B</td>
<td>G</td>
<td>U</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>G</td>
<td>A</td>
<td>U</td>
</tr>
<tr>
<td>H</td>
<td>A</td>
<td>C</td>
<td>U</td>
</tr>
</tbody>
</table>
Usage:
```
random_assignment_of_nucleotides [-h] [-v] [--alignfn ALIGNFN]
                              [--seqfn SEQFN] [--outfn OUTFN]
```

Options:
- `-v=False, --verbose=False` increase output verbosity
- `--alignfn` alignment in the Fasta format
- `--seqfn` sequences in the Fasta format
- `--outfn` output aln file (default: alnfn .fasta -> _out.fasta)

random_assignment_of_nucleotides.py - Random assignment of nucleotides for non-typical characters in the sequence alignment (arg –alignfn or fasta file with sequences (arg –seqfn)

get_align(alignfn)
Get seq from an alignment with gaps.

Args: 
alignfn (str): a path to an alignment

Usage:
```
>>> get_align('test_data/aln1.fasta')
SingleLetterAlphabet() alignment with 2 rows and 13 columns
AGGGGGACAGNYU 1
CYGA------CGG 2
```

Returns: alignment
Get seq from an fasta file. :param seqfn: a path to a fasta file :type seqfn: str

Usage::

```python
>>> get_align('test_data/fasta.fasta')
```

Returns [SeqRecord(seq=Seq('GGGYYGCCNRW', SingleLetterAlphabet()), id='1', name='1',
  description='1', dbxrefs=[]), SeqRecord(seq=Seq('GGRGYYGCCUURW AA', SingleLetterAlphabet()),
  id='1', name='1', description='1', dbxrefs=[])]


Returns writes to a file in fasta format

Return type none


Returns writes to a file in fasta format

Return type none

3.4.10 CMAAlign

class rna_tools.tools.rna_alignment.rna_alignment.CMAAlign(outputfn=None)
CMAalign class around cmalign (of Inferal).

cmalign - aligns the RNA sequences in <seqfile> to the covariance model (CM) in <cmfile>. The new alignment is output to stdout in Stockholm format.

Example:

```python
cma = ra.CMAAlign()
cma.run_cmalign("ade_seq.fa", "RF00167.cm")
seq = cma.get_seq()
print 'cma hit ', seq
print 'seq ', a.align_seq(seq)
print 'a.rf ', a.rf

```

```text
# STOCKHOLM 1.0
#=GF AU Infernal 1.1.2
ade ----------------CGCUUCAUAUACUAAUGAUUGGUGUUGGGAGGUUCUCAAAGAG-
  CCUUAAA--CUUUAGAUAUGAUGAG-----------------CCUUAA--CCUUAAUAGAUGA--
#=#=GR ade PP ..........................99.................................
  --------.+++++++++++++++++++999..............
  #=#=GC SS_cons :........................((((((/,<<<<<______)>>>>>>>,,,,,,,<vvv
  __>>>>)>>>),)))))))))))))))))))))))))))))))))))))))))))
```
Install http://eddylab.org/infernal/


```python
# get_gc_rf()
get #=GC RF.

Variables self.output – string

get_seq()

Variables self.output – output of cmalign, string

run_cmalign (seq, cm, verbose=True)
Run cmalign and process the result.

Parameters
• seq – seq string
• cm – cm fn

Run:
```
$ cmalign RF01831.cm 4lvv.seq
# STOCKHOLM 1.0
#=GF AU Infernal 1.1.2
4lvv -GGAGAGUA-GAUGAUUCGCGUUAAGUGUGUGUGA-AUGGGAUGUCG-UCACACAACGAAGC---
→GAGA---GCAGGUGAACAUU-GCAUCCGCUCCA
#=GR 4lvv PP .********.************9999998.**********.8999999******8...
→5555...8************.************
#=GC SS_cons ((((((((----(((((((((((,,,,,<<-<<<<<<<<___________>>>>>>>>>>,,,<<<<_)
→_____>>>>,,,)))))))))))-------)))))
#=GC RF
→ggcaGAGUAGggugccgcgcGGuuAGUGccggcggAcGGGgAUUUGccgccggACGAAAgggcaaaaauuggcgcGGguacggcaccCGG
```

Warning: requires cmalign to be set in your shell

### 3.4.11 RChie

```python
class rna_tools.tools.rna_alignment.rna_alignment.RChie
RChie - plotting arc diagrams of RNA secondary structures.
```
The offline version of R-chie, which requires first installing R4RNA is available here, or clone our git repository here How to install it:

- Ensure R is installed already, or download it freely from http://www.r-project.org/
- Download the R4RNA (https://github.com/jujubix/r-chie), open R and install the package:

```r
install.packages("<path_to_file>/R4RNA", repos = NULL, type="source")
# Install the optparse and RColorBrewer
install.packages('optparse')
install.packages('RColorBrewer')
```

- Go to rna_tools/rna_tools_config_local.py and set RCHIE_PATH to the folder with RChie, e.g. "/home/magnus/work/opt/r-chie/".

To test if Rchie works on your machine (from rna_align folder):

```bash
<path to your rchie>/rchie.R --msafie test_data/rchie_test_files/fasta.txt test_→
data/rchie_test_files/helix.txt
```

you should have rchie.png file in the folder.

More at http://www.e-rna.org/r-chie/download.cgi


```python
plot_cov(seqs, ss_cons, plot_fn='rchie.png', verbose=False)
```
Plot an RChie plot_conv.

**Parameters**

- `seqs` – seqs in the fasta format

### 3.4. RNA Alignment
• **ss_cons** – a string of secondary structure consensus, use only ( ). Works with pseudo-knots.

```python
show()
write(outfn)
```

### 3.4.12 Renumber a pdb file according to alignment

renum_pdb_to_aln.py - renumber a pdb file based on the alignment.

**author:** A. Zyla under supervision of mmagnus

**Warning:** works only for single chain! and requires Biopython (tested with v1.68)

```bash
```

**Positional arguments:**

- `seqid` : seq id in the alignment
- `alignfn` : alignment in the Fasta format
- `pdbfn` : pdb file

**Options:**

```
-v=False, --verbose=False  increase output verbosity
--residue_index_start=1  renumber starting number (default: 1)
--outfn                   output pdb file (default: pdbfn .pdb -> _out.pdb)
```

renum_pdb_to_aln.py - renumber a pdb file based on the alignment.

**author:** A. Zyla under supervision of mmagnus

**Warning:** works only for single chain! and requires Biopython (tested with v1.68)

```python
rna_tools.tools.renum_pdb_to_aln.renum_pdb_to_aln.get_parser()
```

```python
rna_tools.tools.renum_pdb_to_aln.renum_pdb_to_aln.get_seq(alignfn, seqid)
```

Get seq from an alignment with gaps.

**Parameters**

- `alignfn` *(str)* – a path to an alignment
- `seqid` *(str)* – seq id in an alignment

**Usage:**

```python
>>> get_seq('test_data/ALN_OBJ1_OBJ2.fa', 'obj1')
SeqRecord(seq=SeqRecord(seq=Seq('GUUCAG-------------------UGAC-', SingleLetterAlphabet()), id='obj1', name='obj1', description='obj1', dbxrefs=[]), id='<unknown id>', name='<unknown name>', description='<unknown description>', dbxrefs=[])```
Returns SeqRecord

```python
rna_tools.tools.renum_pdb_to_aln.renum_pdb_to_aln.open_pdb(pdbfn)
```
Open pdb with Biopython.

Parameters `pdbfn (str)` – a path to a pdb structure

Returns with a pdb structure

Return type PDB Biopython object

```python
rna_tools.tools.renum_pdb_to_aln.renum_pdb_to_aln.renumber(seq_with_gaps, struc, residue_index_start)
```
Renumber a pdb file.

Parameters

- `seq_with_gaps (str)` – a target sequence extracted from the alignment
- `struc (pdb)` – a structure
- `residue_index_start (int)` – starting number

Returns BioPython Structure object

```python
rna_tools.tools.renum_pdb_to_aln.renum_pdb_to_aln.write_struc(struc, outfn)
```
Write renumbered pdb with Biopython.

Parameters

- `struc (pdb)` – a renumbered structure
- `outfn (str)` – a path to a new, renumbered pdb file

Returns writes to a file

Return type none

3.5 Root Mean Square Deviation (RMSD)

3.5.1 rna_calc_rmsd

rna_calc_rmsd.py - calculate RMSDs of structures to the target

If you still have problem with various number of atoms, check out this issue: get_rnapuzzle_ready: how to deal with `Alternate location indicator (https://github.com/mmagnus/rna-pdb-tools/issues/30).

The program is using (https://github.com/charnley/rmsd).

Example #1:

```bash
$ rna_calc_rmsd.py -t 6_0_solution_4GXY_rpr.pdb --model-selection=A:1-17+24-110+115-168 *.pdb
rmsd_calc_rmsd_to_target
-method: all-atom-built-in
# of models: 35
6_0_solution_4GXY_rpr.pdb 0.0 3409
6_Blanchet_1_rpr.pdb 22.31 3409
6_Blanchet_2_rpr.pdb 21.76 3409
6_Blanchet_3_rpr.pdb 21.32 3409
6_Blanchet_4_rpr.pdb 22.22 3409
```

3.5. Root Mean Square Deviation (RMSD)
Example #2:

```

time rmsd_calc_to_target.py
   -t 5k7c_clean_onechain_renumber_as_puzzle_srr.pdb
   --target-selection A:1-48+52-63
   --model-selection A:1-48+52-63
   --target-ignore-selection A/57/O2'
   clusters/*_AA.pdb

rmsd_calc_rmsd_to_target
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

   target_selection: A:1-48+52-63
   model_selection: A:1-48+52-63
   target_ignore_selection: A/57/O2'
   model_ignore_selection: 
   # of models: 801
   fn,rmsd_all
   pistol_thrs0.50A_clust01-000001_AA.pdb,7.596
   pistol_thrs0.50A_clust02-000001_AA.pdb,7.766
   pistol_thrs0.50A_clust03-000001_AA.pdb,18.171
   [...]  
   pistol_thrs0.50A_clust799-000001_AA.pdb,5.356
   pistol_thrs0.50A_clust800-000001_AA.pdb,15.282
   pistol_thrs0.50A_clust801-000001_AA.pdb,16.339
   # of atoms used: 1237
   csv was created! rmsds.csv

rmsd_calc_to_target.py -t 5k7c_clean_onechain_renumber_as_puzzle_srr.pdb 
37.93s user 1.07s system 87% cpu 44.650 total
```

usage: rna_calc_rmsd [-h] [-t TARGET_FN] [--target-selection TARGET_SELECTION]
   [--target-ignore-selection TARGET_IGNORE_SELECTION]
   [--target-column-name]
   files [files ...]

Positional arguments:
   files

Options:
   -t=, --target-fn= pdb file
   --target-selection= selection, e.g. A:10-16+20, where #16 residue is included
   --target-ignore-selection= A/10/O2'
   --model-selection= selection, e.g. A:10-16+20, where #16 residue is included
--model-ignore-selection=  A/10/O2'
-m=all-atom-built-in, --method=all-atom-built-in  align, fit
-o=rmsds.csv, --rmsds-fn=rmsds.csv  output, matrix
-v=False, --verbose=False  verbose
-pr=False, --print-results=False  Undocumented
-sr=False, --sort-results=False  Undocumented
-pp=False, --print-progress=False  Undocumented
--target-column-name=False  Undocumented

rna_calc_rmsd.py - calculate RMSDs of structures to the target

If you still have problem with various number of atoms, check out this issue: get_rnapuzzle_ready: how to deal with


The program is using (https://github.com/charnley/rmsd).

Example #1:

```bash
$ rna_calc_rmsd.py -t 6_0_solution_4GXY_rpr.pdb --model-selection=A:1-17+24-110+115-
→ 168 *.pdb
```

<table>
<thead>
<tr>
<th>method: all-atom-built-in</th>
</tr>
</thead>
<tbody>
<tr>
<td># of models: 35</td>
</tr>
<tr>
<td>6_0_solution_4GXY_rpr.pdb 0.0 3409</td>
</tr>
<tr>
<td>6_Blanchet_1_rpr.pdb 22.31 3409</td>
</tr>
<tr>
<td>6_Blanchet_2_rpr.pdb 21.76 3409</td>
</tr>
<tr>
<td>6_Blanchet_3_rpr.pdb 21.32 3409</td>
</tr>
<tr>
<td>6_Blanchet_4_rpr.pdb 22.22 3409</td>
</tr>
<tr>
<td>6_Blanchet_5_rpr.pdb 24.17 3409</td>
</tr>
<tr>
<td>6_Blanchet_6_rpr.pdb 23.28 3409</td>
</tr>
<tr>
<td>6_Blanchet_7_rpr.pdb 22.62 3409</td>
</tr>
<tr>
<td>6_Bujnicki_1_rpr.pdb 36.95 3409</td>
</tr>
<tr>
<td>6_Bujnicki_2_rpr.pdb 30.9 3409</td>
</tr>
<tr>
<td>6_Bujnicki_3_rpr.pdb 32.1 3409</td>
</tr>
<tr>
<td>6_Bujnicki_4_rpr.pdb 32.04 3409</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Example #2:

```bash
time rmsd_calc_to_target.py
 -t 5k7c_clean_onechain_renumber_as_puzzle_srr.pdb
 --target-selection A:1-48+52-63
 --model-selection A:1-48+52-63
 --target-ignore-selection A/57/02'
 clusters/*_AA.pdb
```

<table>
<thead>
<tr>
<th>target_selection: A:1-48+52-63</th>
</tr>
</thead>
<tbody>
<tr>
<td>model_selection: A:1-48+52-63</td>
</tr>
<tr>
<td>target_ignore_selection: A/57/02'</td>
</tr>
<tr>
<td>model_ignore_selection:</td>
</tr>
<tr>
<td># of models: 801</td>
</tr>
<tr>
<td>fn,rmsd_all</td>
</tr>
</tbody>
</table>

3.5. Root Mean Square Deviation (RMSD) 65
RNA tools documentation, Release

```plaintext
pistol_thrs0.50A_clust01-000001_AA.pdb, 7.596
pistol_thrs0.50A_clust02-000001_AA.pdb, 7.766
pistol_thrs0.50A_clust03-000001_AA.pdb, 18.171
[..]
pistol_thrs0.50A_clust799-000001_AA.pdb, 5.356
pistol_thrs0.50A_clust800-000001_AA.pdb, 15.282
pistol_thrs0.50A_clust801-000001_AA.pdb, 16.339
# of atoms used: 1237
csv was created! rmsds.csv
rmsd_calc_to_target.py -t 5k7c_clean_onechain_renumber_as_puzzle_srr.pdb
37.93s user 1.07s system 87% cpu 44.650 total
```

```python
rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd.calc_rmsd(a, b, target_selection, target_ignore_selection, model_selection, model_ignore_selection, verbose)
```

*a is model b is target*

**Params**

- `a` = filename of structure `a`
- `b` = filename of structure `b`

**Returns**

- `rmsd`, number of atoms

```python
rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd.calc_rmsd_pymol(pdb1, pdb2, method)
```

Calculate rmsd using PyMOL. Two methods are available: align and fit

**See:**

- Fit: [http://www.pymolwiki.org/index.php/Fit](http://www.pymolwiki.org/index.php/Fit)

Align can return a list with 7 items:

- RMSD after refinement
- Number of aligned atoms after refinement
- Number of refinement cycles
- RMSD before refinement
- Number of aligned atoms before refinement
- Raw alignment score
- Number of residues aligned

In this version of function, the function returns `RMSD before refinement`.

**Install on OSX:** *brew install homebrew/science/pymol* and set `PYTHONPATH` to your PyMOL packages, e.g.

```
PYTHONPATH=$PYTHONPATH:/opt/local/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/site-packages
```

**If problem:**

```
Match-Error: unable to open matrix file '/opt/local/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/site-packages/data/pymol/matrices/BLOSUM62'
```

*then define `PYMOL_PATH` in your .bashrc, e.g.:*

```
export PYMOL_PATH=/opt/local/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/site-packages/pymol/
```

```python
rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd.get_parser()
```

```python
rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd.get_rna_models_from_dir(files)
```

66 Chapter 3. RNA Tools
Parameters **models** – a list of filenames

Example of the list:

```python
['test_data/rp17/2_restr1_Michal1.pdb_clean.pdb', 'test_data/rp17/2a_nonrestr2_→Michal1.pdb_clean.pdb', 'test_data/rp17/3_nonrestr1_Michal1.pdb_clean.pdb', 'test_data/rp17/5_restr1_→Michal3.pdb_clean.pdb']
```

```python
rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd.sort_nicely(l)
```

Sort the given list in the way that humans expect.

http://blog.codinghorror.com/sorting-for-humans-natural-sort-order/

### 3.5.2 rna_calc_rmsd_multi_targets

rna_calc_rmsd_multi_targets.py - calculate RMSDs of structures to multiple targets:

```bash
$ rna_calc_rmsd_multi_targets.py --models multi-targets/rp21/*.pdb
   --targets multi-targets/rp21/solutions/*.pdb
   --target-selection A:1-27+29-41
   --model-selection A:1-27+29-41
```

CSV table produced:

```
<table>
<thead>
<tr>
<th></th>
<th>21_solution_0_ChainA.pdb</th>
<th>21_solution_0_ChainB.pdb</th>
<th>21_solution_1_ChainA.pdb</th>
<th>21_solution_1_ChainB.pdb</th>
<th>21_solution_2.pdb</th>
<th>mean</th>
<th>min</th>
<th>max</th>
<th>sd</th>
</tr>
</thead>
<tbody>
<tr>
<td>fn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21_3dRNA_1.rpr.pdb</td>
<td>12.17</td>
<td>12.17</td>
<td>12.11</td>
<td>12.11</td>
<td>12.13</td>
<td>12.11</td>
<td>12.17</td>
<td></td>
<td>0.03</td>
</tr>
<tr>
<td>21_Adamiak_1.rpr.pdb</td>
<td>4.64</td>
<td>4.64</td>
<td>4.61</td>
<td>4.61</td>
<td>4.63</td>
<td>4.61</td>
<td>4.64</td>
<td></td>
<td>0.01</td>
</tr>
<tr>
<td>21_ChenHighLig_1.rpr.pdb</td>
<td>4.01</td>
<td>3.97</td>
<td>4.07</td>
<td>4.01</td>
<td>4.07</td>
<td>3.97</td>
<td>4.07</td>
<td></td>
<td>0.04</td>
</tr>
<tr>
<td>21_Das_1.rpr.pdb</td>
<td>5.71</td>
<td>5.60</td>
<td>5.61</td>
<td>5.65</td>
<td>5.60</td>
<td>5.60</td>
<td>5.71</td>
<td></td>
<td>0.05</td>
</tr>
</tbody>
</table>
```

Save `rna_calc_rmsd_multi_targets_output.csv`

```
usage: rna_calc_evo_rmsd [-h] [-v] [--models MODELS [MODELS ...]]
   [--targets TARGETS [TARGETS ...]]
   [--output-csv OUTPUT_CSV]
   [--model-selection MODEL_SELECTION]
   [--target-selection TARGET_SELECTION]
```

Options:

- `-v=False, --verbose=False`  be verbose
- `--models`  Undocumented
- `--targets`  Undocumented
- `--output-csv=rna_calc_rmsd_multi_targets_output.csv`  Undocumented
3.5.3 rna_calc_rmsd_trafl

rmtd_calc_trafl - calculate RMSD of transition A->B based on a SimRNA trajectory

After this script, run:

```
rna_cal_rmsd_trafl_plot.py rmsd.txt
```

to get a plot like this:

Prepare structures:

```
$ SimRNA -p 17_Das_2_rpr.pdb -n 0 -o 17_Das_2_rpr_n0 # no trafl, trafl will be added
$ SimRNA -p 5k7c_clean_onechain_renumber_as_puzzle_rpr_rmGapped.pdb -n 0 -o 5k7c_clean_onechain_renumber_as_puzzle_rpr_rmGapped
#(struc must be (~CG~) nope. It has to be a trajectory!)
```

and run:

```
$ rmsd_calc_trafl.py 17_Das_2_rpr.pdb.trafl 17_Das_2_rpr_n0.trafl 5k7c_clean_onechain_renumber_as_puzzle_rpr_rmGapped_n0.trafl rp17_rmsd.txt
> calc_rmsd_to_1st_frame
/Users/magnus/work/opt/simrna/SimRNA_64bitIntel_MacOSX_staticLibs/calc_rmsd_to_1st_frame 17_Das_2_rpr.pdb.trafl 17_Das_2_rpr_n0.trafl 5k7c_clean_onechain_renumber_as_puzzle_rpr_rmGapped_n0.trafl rp17_rmsd.txt
< rmsd_out: 17_Das_2_rpr.pdb_rmsd_e
> struc: 17_Das_2_rpr_n0.trafl 2
> trafl: 17_Das_2_rpr.pdb.trafl 48
% saved: 17_Das_2_rpr.pdb.trafl_17_Das_2_rpr_n0.trafl
> calc_rmsd_to_1st_frame
```
Warning: calc_rmsd_to_1st_frame (SimRNA) is required and the path to the binary file is defined in config_local.

usage: rna_calc_evo_rmsd [-h] traf1 struc1 struc2 rmsds_fn

Positional arguments:
  traf1       trafil
  struc1      structure A
  struc2      structure B
  rmsds_fn    output file

rna_cal_rmsd_trafl_plot - generate a plot based of <rmsd.txt> of rna_calc_evo_rmsd.py.

usage: rna_cal_rmsd_trafl_plot [-h] file

Positional arguments:
  file       rmsd.txt

3.5.4 rna_calc_rmsd_all_vs_all

rna_calc_rmsd_all_vs_all.py - calculate RMSDs all vs all and save it to a matrix

Examples:

rna_calc_rmsd_all_vs_all.py -i test_data -o test_output/rmsd_calc_dir.tsv
  # of models: 4
  ... 1 test_data/struc1.pdb
  ... 2 test_data/struc2.pdb
  ... 3 test_data/struc3.pdb
  ... 4 test_data/struc4.pdb
The program is using (https://github.com/charnley/rmsd)

```bash
usage: rna_calc_rmsd_all_vs_all [-h] [-i INPUT_DIR] [-o MATRIX_FN]
```

Options:

- `i=`, `--input-dir=` input folder with structures
- `o=`, `--matrix-fn=` output, matrix

`rna_calc_rmsd_all_vs_all.py` - calculate RMSDs all vs all and save it to a matrix

Examples:

```bash
rna_calc_rmsd_all_vs_all.py -i test_data -o test_output/rmsd_calc_dir.tsv
```

```plaintext
# of models: 4
... 1 test_data/struc1.pdb
... 2 test_data/struc2.pdb
... 3 test_data/struc3.pdb
... 4 test_data/struc4.pdb
```

The program is using (https://github.com/charnley/rmsd)

```python
import rna_tools

rna_tool = rna_tools.rna_calc_rmsd.rna_calc_rmsd_all_vs_all()
```

3.6 Interaction Network Fidelity (INF)

3.6.1 `rna_calc_inf`

```bash
files [files ...]
```

Positional arguments:

- `files` files, e.g folder_with_pdb/*pdb

Options:

- `-t=`, `--target_fn=` pdb file
- `-m=3`, `--number_of_threads=3` number of threads used for multiprocessing, if 1 then mp is not used (useful for debugging)
- `-s=`, `--ss=` A:(([]))
- `--no-stacking=False` default: use stacking, if this option on, don’t take into account stacking, WARNING/BUG: inf_all will be incorrectly calculated if stacking is off
- `--debug=False` Undocumented
A tool to calc inf_all, inf_stack, inf_WC, inf_nWC, SNS_WC, PPV_WC, SNS_nWC, PPV_nWC between two structures.

Mind, that ClaRNA is pretty slow, it takes even a few seconds to analyze a structure, so for, say, 1000 models you need a few hours.

How to make it faster? First, you can use --number_of_threads to specify the number of cores used for multiprocessing.

Second, the procedure implemented in here is composed of two steps, first for each structure ClaRNA is used to generate an output with contacts, then these files are used for comparisons. So, if you want to re-run your analysis, you don’t have to run re-run ClaRNA itself. Thus, be default ClaRNA is not executed if <model>.outCR is found next to the analyzed files. To change this behavior force (--force) rna_cal_inf.py to re-run ClaRNA.

ClaRNA_play required! https://gitlab.genesilico.pl/RNA/ClaRNA_play (internal GS gitlab server). Contact <magnus@genesilico.pl>.

import progressbar (in version 2) is required!

```python
rna_tools.tools.rna_calc_inf.rna_calc_inf.do_job(i)
```

Run ClaRNA & Compare, add 1 to the counter, write output to csv file (keeping it locked)

```python
rna_tools.tools.rna_calc_inf.rna_calc_inf.get_parser()
```

### 3.6.2 rna_calc_dinf

Obtain a list of interaction in an RNA molecule where “Interaction” is purely distance based (defined by –cuf-off). Later, you can use it to calculate distance based INF, dINF :-).

Example:

```bash
[mm] rna_calc_dinf$ git:(master)$ ./rna_calc_dinf.py test_output/1Y26.pdb
X 13 X 14 bp G C WW_cis 1
X 13 X 83 bp G C WW_cis 1
X 13 X 82 bp U C WW_cis 1
X 14 X 15 bp C G WW_cis 1
X 14 X 83 bp G G WW_cis 1
X 14 X 81 bp G G WW_cis 1
X 14 X 82 bp U G WW_cis 1
```

use clarna_compare.py:

```bash
[mm] rna_calc_dinf$ ./rna_calc_dinf.py test_output/1Y26.pdb > 1Y26.pdb.outCR
[mm] rna_calc_dinf$ clarna_compare.py -iref 1Y26.pdb.outCR -ichk 1Y26.pdb.outCR
1Y26.pdb.outCR 1Y26.pdb.outCR 1.000 0.000
→ 1.000 1.000 1.000 1.000 1.000 1.000
```

You can use –d to get a list of all interacting bases, something like:
so you can plot all interacting bases:

Mind, that `draw_dists` works on C2 atoms, that might be different from atoms detected with the program (e.g. different base atom could be detected to make an interaction).

Positional arguments:
  file              a PDB file

Options:
  -d=False, --draw-dists=False  Undocumented
  -c=5, --cut-off=5            Undocumented
  -v=False, --verbose=False    be verbose

3.7 RNA filter (DCA)

3.7.1 rna_filter.py - calculate distances based on given restraints on PDB files or SimRNA trajectories

rna_filter.py - calculate distances based on given restraints on PDB files or SimRNA trajectories.

Changes: weight is always 1 (at least for now). >,=,=,<=

[PREVIOUS DOCUMENTATION - TO BE REMOVED]

    rna_filter.py -s 4gxy_rpr.pdb -r rp06_MohPairs.rfrestrs d:A5-A42 100.0 measured: 26.7465763417 [x]
    d:A11-A26 100.0 measured: 19.2863696104 [x]

    [mm] rp06$ git:(master) $ rna_filter.py -s 4gxy_rpr.pdb -r rp06_MohPairs.rfrestrs d:A5-A42 100.0 measured: 26.7465763417 [x] d:A11-A26 100.0 measured: 19.2863696104 [x]
Traceback (most recent call last):
  File “/home/magnus/work-src/rna-pdb-tools/bin/rna_filter.py”, line 270, in <module>
    calc_scores_for_pdbs(args.structures, restraints, args.verbose)
  File “/home/magnus/work-src/rna-pdb-tools/bin/rna_filter.py”, line 221, in calc_scores_for_pdbs
    dist = get_distance(residues[h[0]]['mb'], residues[h[1]]['mb'])
KeyError: 'A24'

correct, there is no A24 in this structure:

The format of restraints:

(d:A1-A2 < 10.0 1) = if distance between A1 and A2 lower than 10.0, score it with 1

Usage:

```bash
$ python rna_filter.py -r test_data/restraints.txt -s test_data/CG.pdb
d:A1-A2 10.0 measured: 6.58677550096 [x]
test_data/CG.pdb 1.0 1 out of 1
```

```bash
# $ python rna_filter.py -r test_data/restraints.txt -t test_data/CG.trafl
(d:A1-A2 < 10.0 1)|(d:A2-A1 <= 10 1)
restraints [('A1', 'A2', '<', '10.0', '1'), ('A2', 'A1', '<=', '10', '1')]
```

Frame #1 e:1252.26
mb for A1 [54.729 28.9375 41.421]
mb for A2 [55.3425 35.3605 42.7455]
d:A1-A2 6.58677550096
mb for A2 [55.3425 35.3605 42.7455]
mb for A1 [54.729 28.9375 41.421]
d:A2-A1 6.58677550096

# this ^ is off right now

Usage:

```bash
usage: rna_filter.py [-h] -r RESTRAINTS_FN [-v]
                    [-s STRUCTURES [STRUCTURES ...]] [-offset OFFSET]
                    [-t TRAJECTORY]
```

Options:

- `r`, `--restraints_fn`  restraints_fn: Format: (d:A9-A41 < 10.0 1)|(d:A41-A9 <= 10 1)
- `v=False, --verbose=False`  be verbose
- `s`  structures
- `--offset=0`  use offset to adjust your restraints to numbering in PDB files, ade (1y26)pdb starts with 13, so offset is -12
- `t`  SimRNA trajectory

3.7.2 rna_dca_mapping.py

Usage:

```bash
usage: rna_dca_mapping.py [-h] --seq SEQ --gseq GSEQ --dca DCA
                          [-offset OFFSET] [-noss] [-mss] [-verbose]
                          [-noshort]
```

Options:
**--seq**  seq fn in Fasta format

**--gseq**  gapped sequence and secondary structure (like in the alignment used for DCA) in Fasta format

**--dca**  file with parsed interactions

**--offset**  offset

**--noss=False**  filter out ss from plot

**--mss=False**  ss every each line

**--verbose=False**  be verbose

**--noshort=False**  filter out short interactions, dist in seq < 6 nt

### 3.7.3 show_dists - show distances in PyMOL

show_dists - show distances in PyMOL.

```python
PyMOL> show_dists([[1,2]])
1, 2, 3.41
```

**Usage:**

```python
PyMOL> show_dists([[1,2]])
1, 2, 3.41
```

**rna_ex2x.py** - analyze an evolutionary coupling file.

Files can be downloaded from [https://marks.hms.harvard.edu/ev_rna/](https://marks.hms.harvard.edu/ev_rna/), e.g. RF00167.EC.interaction.csv

### 3.7. RNA filter (DCA)
--pairs:

```bash
$ rna_ex2x.py RF00167.EC.interaction_LbyN.csv --pairs
[18, 78], [31, 39], [21, 75], [30, 40], [28, 42], [27, 43], [59, 67], [54, 72], [57, 69], [25, 45], [29, 41], [17, 79], [26, 44], [16, 80], [14, 82], [19, 77], [55, 71], ...
```

Usage:

```bash
```

Positional arguments:

- `interaction_fn` interaction file

Options:

- `--sep=,` separator
- `--chain=A` chain
- `--ec-pairs=False` Undocumented
- `--ss-pairs` file with secondary structure base pairs
- `--pairs-delta=False` delta: ec-bp - ss-paris

### 3.7.4 rna_pairs2SimRNArestrs.py - convert pairs to SimRNA restraints

rna_pairs2SimRNArestrs.py - convert pairs to SimRNA restraints

Example:

```bash
$ rna_pairs2SimRNArestrs.py rp06_pairs_delta.txt -v
# of pairs: 42
SLOPE A/2/MB A/172/MB 0 6 1
SLOPE A/2/MB A/172/MB 0 7 -1
SLOPE A/3/MB A/169/MB 0 6 1
SLOPE A/3/MB A/169/MB 0 7 -1
SLOPE A/12/MB A/32/MB 0 6 1
```

Usage:

```bash
```

Positional arguments:

- `pairs` a file with [[2, 172], [3, 169], [12, 32], [13, 31]]

Options:

- `--offset=0` can be -10
- `--weight=3` weight
- `--dist=7` distances, for MOHCA use 25
- `--well=False` well instead of slope
- `-v=False, --verbose=False` be verbose
### 3.7.5 rna_ss_get_bps.py - get a list of base pairs for a given “fasta ss” file.

**rna_ss_get_bps.py** - get a list of base pairs for a given “fasta ss” file.

**Input file:**

```bash
cat ade_ss.fa
>1y26
CGCUUCAUAUAUCCUAUUGAUUGGUUUGGAGUUUCUACCAAGACGCUUAACUCUUGAUUAUAGUG
(((((((.(((.........................))))))))........(((........)))))))))%
```

**Usage:**

```
$ rna_ss_get_bps.py ade_ss.fa --offset 12
[[13, 83], [14, 82], [15, 81], [16, 80], [17, 79], [18, 78], [19, 77], [20, 76], [21, 75], [25, 45], [26, 44], [27, 43], [28, 42], [29, 41], [30, 40], [54, 72], [55, 71], [56, 70], [57, 69], [58, 68], [59, 67]]
```

Now it also work with pseudoknots.

**Usage:**

```
usage: rna_ss_get_bps [-h] [--offset OFFSET] [-v] file
```

**Positional arguments:**
- **file**
  - file in the Fasta format

**Options:**
- **--offset**
  - offset
- **-v, --verbose**
  - be verbose

### 3.7.6 rna_pairs_diff.py - get a diff of pairs

**rna_pairs_diff.py** - get a diff of pairs

**Usage:**

```
$ rna_pairs_diff.py pistol_dca_all.bp pistol.bp
# of ec_paris: 31
# of ssbps : 18
delta#: 13
[[4, 32], [6, 9], [6, 36], [6, 39], [9, 39], [13, 32], [16, 17], [17, 18], [22, 49], [29, 58]]
```

**Usage:**

```
usage: rna_pairs_diff.py [-h] [-v] pairs1 pairs2
```

**Positional arguments:**
- **pairs1**
  - a list of pairs, A
- **pairs2**
  - a list of pairs to subtract, A-B, results in C(all pairs that are in A and are not in B)

**Options:**
- **-v, --verbose**
  - be verbose
3.8 Contacts classification & secondary structure detection

3.8.1 3DNA (contacts classification & secondary structure detection)

Python parser to 3dna <http://x3dna.org/>.

Installation:

```sh
$ install the code from http://forum.x3dna.org/downloads/3dna-download/
Create a copy of the rna_x3dna_config_local_sample.py (remove "_sample") present in
---rna-tools/rna_tools/tools/rna_x3dna folder.
Edit this line:
BINARY_PATH = <path to your x3dna-dssr file>
matching the path with the path of your x3dna-dssr file.
e.g. in my case: BINARY_PATH = ~/bin/x3dna-dssr.bin
```

For one structure you can run this script as:

```sh
$ py3dna$ git:(master) ./rna_x3dna.py test_data/1xjr.pdb
>1xjr nts=47 [1xjr] -- secondary structure derived by DSSR
..(((((((...((((.((((.....))..))..))).).))))))).)))))))))))))))))))))))))))))))))))))
```

For multiple structures in the folder, run the script like this:

```sh
$ py3dna$ git:(master) ./rna_x3dna.py test_data/
>1xjr nts=47 [1xjr] -- secondary structure derived by DSSR
..(((((((...((((.((((.....))..))..))).).))))))).)))))))))))))))))))))))))))))))))))))
>6TNA nts=76 [6TNA] -- secondary structure derived by DSSR
..(((((((...((((.((((.....))..))..))).).))))))).)))))))))))))))))))))))))))))))))))))
```

```py
rna_tools.tools.rna_x3dna.rna_x3dna.get_parser()

class rna_tools.tools.rna_x3dna.rna_x3dna.x3DNA (pdbfn, show_log=False)

Attributes:

curr_fn report

clean_up (verbose=False)

get_ion_water_report ()
@todo File name: /tmp/tmp0pdNHS

no. of DNA/RNA chains: 0 [] no. of nucleotides: 174 no. of waters: 793 no. of metals: 33
[Na=29, Mg=1, K=3]

get_modifications ()
Run find_pair to find modifications.
```
get_secstruc()
Get secondary structure.

get_seq()
Get sequence.

Somehow 1bzt_1 x3dna UCAGACUUUAAPCUGA, what is P? P -> u

run_x3dna (show_log=False)

exception rna_tools.tools.rna_x3dna.rna_x3dna.x3DNAMissingFile

### 3.8.2 ClaRNA (contacts classification)

If you want to calculate “Interaction Network Fidelity (INF) and not only” see rna_calc_inf in the Utils.

usage:
```
./clarna_app.py ../../input/5k7c_clean_onechain_renumber_as_puzzle_srr.pdb
../../input/5k7c_clean_onechain_renumber_as_puzzle_srr.pdb
(((((((......))))))))........(((......)))......(((((((......))))))....)
```

**Example**

```python
from rna_tools.utils.clarna_app import clarna_app
if __name__ == '__main__':
    ss = '((((.([[[[.]]]]))))........((((.....[[[.]]]]))...(((((....)))))..))))'
    fnCRref = clarna_app.get_ClaRNA_output_from_dot_bracket(ss)
    f = '...'./rna_calc_rmsd/test_data/pistol/5k7c_clean_onechain_renumber_as_puzzle_srr.pdb'
    fnCR = clarna_app.clarna_run(f, force=False)
    results = clarna_app.clarna_compare(fnCRref, fnCR)
    print results
```

```
#tmp_Z42i_..pdb.outCR 5k7c_clean_onechain_renumber_as_puzzle_srr.pdb.outCR
    → 0.706 NA 0.865 NA 0.842 0.889 NA 0.000
```

**Warning:** Setup a bash variable: ClaRNA_play_path, and add ClaRNA_play to your $PATH (install ClaRNA_play https://gitlab.genesilico.pl/RNA/ClaRNA_play (internal GS gitlab server))

**rna_tools.tools.clarna_app.clarna_app.clarna_compare** (target_cl_fn, i_cl_fn, verbose=False)

Run ClaRNA compare.

**Returns** a list target, fn, scores

Scores:
```
inf_all 0.706
inf_stack -999.999 -> NA
inf_WC 0.865
inf_nWC -999.999 -> NA
SNS_WC 0.842
PPV_WC 0.889
SNS_nWC NA
PPV_nWC 0.000
```

3.8. Contacts classification & secondary structure detection
Example of the list:

```
5k7c_clean_onechain_renumber_as_puzzle_srr.pdb  pistol_thrs0.50A_clust01-
    000001_AA.pdb
0.642   NA   0.874   0.944   0.810   0.000   0.
```

use results.split()[4] to get inf_WC

```python
rna_tools.tools.clarna_app.clarna_app.clarna_run(fn, force=True, stacking=True)
```

Run ClaRNA run

**fn**: str  filename to analyze

**Returns**  a filename to ClaRNA output (fn + ‘.outCR’)

```python
rna_tools.tools.clarna_app.clarna_app.get_ClaRNA_output_from_dot_bracket(ss, temp=True, verbose=False)
```

Get dummy ClaRNA output out of dat bracket secondary structure (ss)

**Parameters**  **ss** (string) – secondary structure

**Returns**  a filename to ClaRNA output

```python
rna_tools.tools.clarna_app.clarna_app.get_dot_bracket_from_ClaRNAoutput(inCR, verbose=False)
```

In inCR file

```python
rna_tools.tools.clarna_app.clarna_app.get_parser()
```

### 3.9 RNA 3D structure prediction

#### 3.9.1 ROSETTA


#### 3.9.1.1 Run (modeling)

```python
rna_rosetta_run.py - prepare & run ROSETTA simulations
```


The script makes(1) a folder for you job, with seq.fa, ss.fa, input file is copied as input.fa to the folder(2) make helices(3) prepare rosetta input files(4) sends jobs to the cluster.

The header is take from the fast file(`` > /header /``) not from the filename of your Fasta file.

I discovered this:

```
qstat -xml | tr ‘\n’ ‘ ’ | sed s$<job_list[^>]>*$\n\n'g'
```
Run:

```bash
rna_rosetta_run.py -i -e -r -g -c 200 cp20.fa
```

Troubleshooting.

If one of the helices is missing you will get:

```
IOError: [Errno 2] No such file or directory: 'helix1.out'
```

and the problem was a1 and g8 pairing:

```
outputting command line to: helix0.RUN # previous helix #0
Sequence: AUGG CCGG
Secstruc: (((()))))
Not complementary at positions a1 and g8!
```

My case with a modeling of rp12

Sequence: cc gc Secstruc: (()) Not complementary at positions 1 and 4!

edit the secondary structure, run the program with -i (init, to overwrite seq.fa, ss.fa) and then it works.

Notes:

```
rp17hc 6 characters
```

Usage:
```
usage: rna_rosetta_run.py [-h] [-i] [-e] [-r] [-g] [-m MOTIF] [-n NSTRUC]
[-c CPUS] [--sandbox SANDBOX]
file
```

Positional arguments:
file: > a04 UAUAACAUAAUAUUUGACAAUAUGGUCAUAAGU- UUCUACCGAAAUACCGUAAUAUUCUGACUAUGUAU ((((.(((((.(((.(((((.........)))))).........(((.(((((.(((((.........)))))))))))))))))))))))

Options:

-i=False, --init=False  prepare _folder with seq and ss
-e=False, --helices=False  produce h(E)lices
-r=False, --rosetta=False  prepare rosetta files (still you need ‘go’ to send jobs to a cluster)
-g=False, --go=False  send jobs to a cluster(run qsubMINI)
-m, --motif  include a motif file, e.g. -s E-loop_1q9a_mutated_no_flanks_renumber.pdb
-n=10000, --nstruc=10000  # of structures you want to get
-c=200, --cpus=200  # of cpus to be used
--sandbox=/home/magnus/rosetta-runs  where to run it (default: RNA_ROSETTA_RUN_ROOT_DIR_MODELING)

3.9.1.2 Get a number of structures

rna_rosetta_n.py - show me # of structure in a silent file

Example:

```
$ rna_rosetta_n.py ade.out
21594
```

```
usage: rna_rosetta_n.py [-h] [-v] file

Positional arguments:
  file ade.out

Options:
  -v=False, --verbose=False  Undocumented
```

3.9.1.3 Get a head of a Rosetta silent file

rna_rosetta_head.py - get a head of a Rosetta silent file.

Example:

```
$ rna_rosetta_head.py -n 10000 thf.out
# a new file will be created, thf_10000.out
```

Silent file:

```
[peyote2] thf head -n 100 thf.out
SEQUENCE:
˓→ggagaguagauuucgguuagaugggauagucacacaaacgaagcgagcgagcguguuguuauuccuuccca
SCORE: score rna_data_backbone rna_vdw rna_base_backbone rna_backbone␣˓→backbone rna_repulsive rna_base_pair rna_base_axis rna_base_stagger ˓→rna_base_stack rna_base_stack_axis rna_rg atom_pair_constraint linear␣˓→chainbreak N_WC N_NWC N_BS description
```
It seems to work:

```
-rw-rw-r-- 1 magnus users 474M 2017-08-06 05:25 thf_10000.out
-rw-rw-r-- 1 magnus users 947M 2017-08-06 04:54 thf.out
```

```
usage: rna_rosetta_n.py [-h] [-v] [-n NSTRUC] file

Positional arguments:
  file         ade.out

Options:
  -v=False, --verbose=False  Undocumented
  -n=10000, --nstruc=10000  Undocumented

3.9.1.4 Cluster

rna_rosetta_cluster.py - cluster silent Rosetta files

A wrapper to ROSETTA tools for RNA modeling


3.9. RNA 3D structure prediction 83
Take $n \times 0.005$ (0.5%) of all frames and put them into `selected.out`. Then the tool clusters this `selected.out`.

```python
rna_tools.tools.rna_rosetta.rna_rosetta_cluster.cluster(radius, limit_clusters)
```

Internal function of `cluster_loop`: It removes `cluster.out` first.

```python
rna_tools.tools.rna_rosetta.rna_rosetta_cluster.cluster_loop(ns, radius, radius_inc_step, limit_clusters)
```

Go from radius 1 to get 1/6 of structures of `ns` (# of selected structures) in the first cluster, then it stops.

```python
rna_tools.tools.rna_rosetta.rna_rosetta_cluster.extract()
rna_tools.tools.rna_rosetta.rna_rosetta_cluster.get_no_structures(file)
rna_tools.tools.rna_rosetta.rna_rosetta_cluster.get_no_structures_in_first_cluster(fn)
```

Get # of structures in a silent file.

**Parameters**

- `fn (string)`: a filename to a silent file

**Returns**

- `int`: # of structures in a silent file

```python
rna_tools.tools.rna_rosetta.rna_rosetta_cluster.get_parser()
rna_tools.tools.rna_rosetta.rna_rosetta_cluster.get_selected(file, nc)
```

Get selected for clustering

```python
rna_tools.tools.rna_rosetta.rna_rosetta_cluster.run()
```

Pipeline for modeling RNA

### 3.9.1.5 Minimize

**rna_rosetta_min.py - a script to do minimization**

The script takes the number of structures and the analyzed silence file and does the maths.

Job names will be as your silent file preceding with ~, e.g. `~tha`.


```bash
ade$ rna_rosetta_cluster.py ade.out
```

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

```bash
rosetta_submit.py min_cmdline min_out 1 24
```

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

E.g. for 20k silent file, 1/6 will be minimized $\approx$ 3.3k:
parallel_min_setup.py -silent rp21cr62.out -tag rp21cr62_min -proc 200 -nstruct 3200 -out_folder mo -out_script MINIMIZE " -ignore_zero_occupancy false "
rosetta_submit.py MINIMIZE mo 1 100 m

[peyote2] rp21 easy_cat.py mo
Catting into: rp21_min.out ... from 200 primary files. Found 3200 decoys.

# on 200 cpus it took around ~30min

usage: rna_rosetta_min.py [-h] [-g] [-c CPUS] file

Positional arguments:
  file ade.out

Options:
  -g=False, --go=False  Undocumented
  -c=200, --cpus=200  default: 200

3.9.1.6 Extract lowscore decoy

rna_rosetta_extract_lowscore_decoys.py - a simple wrapper to extract_lowscore_decoys.py
To be used in Jupter notebooks and other scripts.

usage: rna_rosetta_extract_lowscore_decoys.py [-h] [-v] nstruc file

Positional arguments:
  nstruc # of low score structures to obtained
  file silent file

Options:
  -v=False, --verbose=False  be verbose

3.9.1.7 Check progress

rna_rosetta_check_progress.py - check progress for many simulations of Rosetta
Example:

[peyote2] rosetta_jobs rna_rosetta_check_progress.py .

jobs #curr #todo #decoys done
0 ./rp17s223 200 0 407 [ ]
1 ./rp17hcf 0 0 0 [ ]

# curr 232 #todo 0

usage: rna_rosetta_cluster.py [-h] [-v] [-m] [-s SELECT] [-k] dir

Positional arguments:
  dir directory with rosetta runs, define by RNA_ROSETTA_RUN_ROOT_DIR_MODELING right now:
  /home/magnus/rosetta-runs

3.9. RNA 3D structure prediction
Options:

- \(v=\text{False}, --\text{verbose}=\text{False}\) be verbose
- \(m=\text{False}, --\text{min-only}=\text{False}\) check only for mo folder
- \(s=, --\text{select}=\) select for analysis only jobs with this phrase, .e.g., evoseq_
- \(k=\text{False}, --\text{kill}=\text{False}\) kill (qdel) jobs if your reach limit (nstruc) of structure that you want, right now is 10000 structures

### 3.9.2 SimRNA

#### 3.9.2.1 Select low energy frames

**rna_simrna_lowest.py** - get lowest energy frames out of a SimRNA trajectory file

This code uses heavily the SimRNATrajectory class. Be default 100 lowest energy frames is exported.

```bash
usage: rna_simrna_lowest.py [-h] [-n NSTRUC] trafl
```

Positional arguments:
```
trafl           SimRNA trafl file
```

Options:
```
-n=100, --nstruc=100  SimRNA trafl file
```

#### 3.9.2.2 Extract

**rna_simrna_extract.py** - extract full atom structures from a SimRNA trajectory file

Options:

**SIMRNA_DATA_PATH** has to be properly defined in `rpt_config_local`.

```bash
usage: rna_simrna_extract.py [-h] -t TEMPLATE -f TRAFL [-c] [-n NUMBER_OF_STRUCTURES]
```

Options:
```
-t, --template template PDB file used for reconstruction to full atom models
-f, --trafl SimRNA trafl file
-c=False, --cleanup=False  Keep only *_AA.pdb files, move *.ss_detected and *.pdbto _<traj name folder>
-n=100, --number_of_structures=100  Undocumented
```

### 3.9.3 SimRNAweb

#### 3.9.3.1 Download files of a SimRNAweb run

**rna_simrnaweb_download_job.py** - download model files, trajectory for a given SimRNAweb job.

Usage:
Example:

```
mlr $ rna_pdb_download_simrna_job.py 27b5093d -m -t -x
# download more clusters, trajectory, extract100

mlrp $ rna_pdb_download_simrna_job.py -t -x -m cf8f8bb2 -p mlrp
# download with a trajectory, and cluster #4 and #5, add to all pdb files
# prefix: mlrp

$ rna_simrnaweb_download_job.py --web-models rp17_well_d10_e1-a43d3ab5 --prefix tar
# prefix added will be tar_XXXX
```

```
job_id
```

Positional arguments:

```
job_id job_id
```

Options:

```
-p, --prefix
prefix to the name, withouth _, be careful with this.If you have already
some files with the given folder, their names mightbe changed.

-n=100, --nstruc=100 extract nstruc the lowest energy, this option must go with --web
-e=False, --extract=False extract nstruc the lowest energy, this option must go with --web
-m=False, --more_clusters=False download also cluster 4 and 5
-r=False, --remove-trajectory=False remove trajectory after analysis
-c=False, --cluster=False get trajectory from cluster OR local on your computer (mdfind for
macOS)
-d=False, --download-trajectory=False web
--top100=False download top100 trajectory
--top200=False download top200 trajectory
--web-models=False web models download
```
3.9.4 SimRNATrajectory

SimRNATrajectory module.

SimRNATrajectory / Frame / Residue / Atom

class rna_tools.tools.simrna_trajectory.simrna_trajectory.**Atom**(*name*, *x*, *y*, *z*)

    get_coord()

        Return coords (np.array).

class rna_tools.tools.simrna_trajectory.simrna_trajectory.**Frame**(*id*, *header*, *coords*, *top_level=False*)

    Syntax of header:
        • write_number
        • replica_id
        • total_energy
        • energy_without_restraints
        • temperature

    **Warning:** If there is an invalid frame, please use **repair_traf.py** to fix the trajectory first.

class rna_tools.tools.simrna_trajectory.simrna_trajectory.**Residue**(*id*, *p*, *c4p*, *n1n9*, *b1*, *b2*)

    Create Residue object.

    Each residue in SimRNA coarse-grained representation consists only 5 coarse-grained atoms:
        • backbone: *p* = phosphate group, *c4p* = sugar moiety
        • nucleotide: *n1n9* = N1 for pyrimidines, N9 for purines, *b1* = C2 for purines and pyrimidines, *b2* = C4 for pyrimidines, C6 for purines

    get_atoms()

        Return all atoms

    get_center()

        Return MB for residue `((self.n1n9 + self.b2) / 2)`

class rna_tools.tools.simrna_trajectory.simrna_trajectory.**SimRNATrajectory**

    load_from_file(*fn*, *debug_break=False*, *top_level=False*, *only_first_frame=False*)

        Create a trajectory based on given filename.

        **Parameters**
        top_level - top_level = True, don’t make a huge tree of objects (Residues/Atoms) == amazing speed up! Useful if you need only frames, energies and coords as text. You only get the info that is in header of each frame.

        top_level = False, makes huge tree of objects (Residues/Atoms) == very slow for a huge trajectories
**Warning:** Loads up whole traffic file into memory, and gets stuck. Use this if you want to compute e.g. distances between atoms, get the positions of specified atoms etc. If you cannot process your trajectory use top_level=True or look at load_from_string() to load a frame by frame from a file.

**load_from_list** *(frames)*

**load_from_string** *(c, txt)*
Create a trajectory based on given string (txt) with id given by c.
Faster method, loads only one frame at a time to memory, and after computations loads the next frame (memory efficient).

**plot_energy** *(plotfn='plot.png ')*
Plots the SimRNA energy of the trajectory.

**save** *(fn, verbose=True)*
Save the trajectory to file.

**sort** *(inplace=True)*
Sort frames within the trajectory according to energy.

### 3.10 RNA Refinement (QRNAS)

rna_refinement - RNA refinement with QRNAS.
Models of RNA 3D structures obtained by modeling methods often suffer from local inaccuracies such as clashes or physically improbable bond lengths, backbone conformations, or sugar puckers. To ensure high quality of models, a procedure of refinement should be applied as a final step in the modeling pipeline. The software tool QRNAS was developed in our laboratory to perform local refinement of nucleic acid structures based on an extended version of the AMBER force field. The extensions consist of energy terms associated with introduction of explicit hydrogen bonds, idealization of base pair planarity and regularization of backbone conformation.


Right now, there is 20k steps of refinement. The initial structure, 179c48aa-c0d3-4bd6-8e06-12081da22998_ALL_thrs6.20A_clust01-00001_AA.pdb.

Analysis output: all-atomic contacts and geometry for 179c48aa-c0d3-4bd6-8e06-12081da22998_ALL_thrs6.20A_clust01-00001_AA.pdb.

Summary statistics

<table>
<thead>
<tr>
<th>All-Atom Contacts</th>
<th>Clashscore, all atoms:</th>
<th>66.7</th>
<th>10th percentile (N=1784, all residues)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clashscore is the number of serious steric overlaps (&gt; 0.4 Å) per 1000 atoms.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nucleic Acid Geometry</td>
<td>Probably wrong sugar puckers:</td>
<td>7</td>
<td>11.29%</td>
</tr>
<tr>
<td></td>
<td>Probably wrong sugar puckers:</td>
<td>20</td>
<td>32.79%</td>
</tr>
<tr>
<td></td>
<td>Bad backbone conformations:</td>
<td>24 / 1458</td>
<td>1.65%</td>
</tr>
<tr>
<td></td>
<td>Bad angles:</td>
<td>235 / 2270</td>
<td>10.35%</td>
</tr>
</tbody>
</table>

In the two column results, the left column gives the raw count, right column gives the percentage.

1 10th percentile is the best among structures of comparable resolution; 9th percentile is the worst. For clashscore, the comparative set of structures was selected by sequence identity.

2 RNA backbone was recently shown to be natomeric. Outliers are RNA suites that don’t fall into recognized natomers.
After 3k steps, ~10min
After 10k steps, around 30min
After 20k steps, around 1h.

Installation of QRNAS

Download the QRNAS package from http://genesilico.pl/qrnas/, unzip the archive, and compile it with the following command:

```
./qrnamake sequential
```

This should create an executable version of QRNAS.

**Warning:** Please, change the name of the binary file from QRNA to QRNAS!

Be default the script searches QRNAS in <rna-pdb-tools>/opt/qrnas/.

Usage of QRNA:

```
QRNA - Quick Refinement of Nucleic Acids (0.2 alpha)
by Juliusz Stasiewicz (jstasiewicz@genesilico.pl)

To use type:
    QRNA -i <input PDBfile> [-o <output PDBfile>] [-c <configfile>] [-p] [-m <restraintsfile>]
OR specify <input PDBfile>, <output PDBfile> and <restraintsfile> in <configfile> and
    --type just:
```

3.10. RNA Refinement (QRNAS)
QRNA -c <configfile>

Installation of this util
Set up in your bashrc:
```
export QRNAS_PATH=<your path to qrnas> # e.g. /home/magnus/src/rna-pdb-tools/opt/qrnas
```

but default rna-pdb-tools searches for qrnas in <rna-pdb-tools>/opt/qrnas.

QRNAS at Peyote2
There is no problem to run QRNAS at our Genesilico cluster, peyote2. Tested by mmagnus –170822. Copy files of QRNAS to peyote and run ./qrnamake sequential.

To run it at a cluster with the Sun Grid Engine queuing system (this one with qusb :-)):
```
for p in *.pdb; do echo "rna_refinement.py $p >& $p.log" | qsub -cwd -V -pe mpi 1 -N "r_\$p" ; done
```

DONE:
- [x] clean up the output structure
- [x] configuration should not be hardcoded

usage: rna_refinement.py [-h] [-s STEPS] [-o OUTPUT_FILE] fn

Positional arguments:
```
fn input pdb file
```

Options:
- `s=20000, --steps=20000` # of steps, default: 20k
- `o, --output_file` output pdb file

3.11 diffpdb

diffpdb - a simple tool to compare text-content of PDB files

The method is quick-and-dirty, but works!

The script takes first 31 characters of lines (or only atom names and residue names) starting with HETATM or ATOM and save these lines to a <filename>.out file.

One file is created per pdb. In the final step DIFF_TOOL is executed on these two output files. You get a diff output. That’s it! Enjoy!

Configuration:
- `DIFF_TOOL="open -a diffmerge"` or `DIFF_TOOL="kompare"` to set up what tool would you like to use to diff files in the file rna-pdb-tools/tools/diffpdb/diffpdb_conf.py (create it if needed)
/diffpdb.py --names test_data/4/1duq.pdb test_data/4/1duq_decoy0171_amb_clx.pdb
and on the Mac (using `diffmerge`):

One of the differences that can be detected with the script is variants of atoms.

or a detection of missing atom.
or a detection of missing OP3 at the beginning.

3.12 RNA clustering with CLANS (clanstix)

rna_clanstix - a tool for visualizing RNA 3D structures based on pairwise structural similarity with Clans.

We hacked Clans thus instead of BLAST-based distances between sequences, you can analyze distances between structures described as p-values of rmsd (based on the method from the Dokholyan lab.)

Quickref:

```bash
rna_clanstix.py --groups-auto 10 --color-by-homolog --shape-by-source thf_ref mapping_pk_refX.txt input2.clans
```

Running Clans: To run CLANS you need to have Java 1.4 or better installed (Java can be downloaded HERE). For full functionality you will also need the NCBI BLAST, PSI-BLAST and formatdb executables (NCBI). For command line parameters and basic help please refer to the README file. (source: http://www.eb.tuebingen.mpg.de/research/departments/protein-evolution/software/clans.html)

The RMSDs between structures are converted into p-values based on the method from the Dokholyan lab or some hacky way developed by mmagnus.
3.12.1 Color groups

You can color your groups:

To get colors, run a cmd like this:

```
rna_clastix.py rnazp17_matrix_farfar_HelSeedCst.txt --groups
  →20:seq1+20+20+20+20+20+20:seq10
```

where with the + sign you separate groups. Each group has to have a number of structures. Optionally it can have a name, e.g., 20:seq1, use : as a separator. If a provided name is native then this group will be shown as starts.

Get inspiration for more colors (http://www.rapidtables.com/web/color/RGB_Color.htm)

3.12.2 How to use ClanstixRNA?

1. Get a matrix of distances, save it as e.g. matrix.txt (see Comment below)
2. run ClanstixRNA on this matrix to get an input file to Clans (e.g. clans_rna.txt):

```
rna_clanstix.py test_data/matrix.txt
```

# clans.input will be created by default

3. open CLANS and click File -> Load run and load clans_run.txt
4. You’re done! :-)

Comment: To get this matrix you can use for example another tool from the rna-pdb-tools packages:

```
rna_calc_rmsd_all_vs_all.py -i rp18 -o rp18_rmsd.csv
rna_clastix.py --groups l:native+5:3dRNA+
  5:Chen+3:Dokh+5:Feng+5:LeeASModel+
  5:Lee+5:RNAComposer+10:RW3D+5:Rhiju+
  1:YagoubAli+3:SimRNA
```

```
   rp18_rmsd.csv clans.in
```
where `rp18` is a folder with structure and `rp18_rmsd.csv` is a matrix of all-vs-all rmsds.


An output of this tool can be viewed using CLANS.


class rna_tools.tools.clanstix.rna_clanstix.RNAStructClans (n=10, dotsize=10)

Clans run.

Usage:

```python
>>> f = open('matrix.txt')
>>> ids = f.readline().replace('#', '').split()
>>> c = RNAStructClans(n=len(ids))  # 200?
>>> c.add_ids(ids)
>>> c.dist_from_matrix(f)
>>> print(c.txt)
```

add_ids(ids)

dist_from_matrix (rmsds, matrix=0, use_pv=False, dont_calc=False, debug=False)

dist_from_matrix_mp (output_pmatrix_fn, max, min, lines, pmat=False, use_pv=False, debug=False)

rna_tools.tools.clanstix.rna_clanstix.check_symmetric (a, rtol=1e-05, atol=1e-08)

https://stackoverflow.com/questions/42908334/checking-if-a-matrix-is-symmetric-in-numpy

rna_tools.tools.clanstix.rna_clanstix.get_parser()
3.13 Misc

3.13.1 Plotting

rna_plot_hist.py - generate a histogram

Don’t open Excel, Jupyter. Simple plot a histogram of one column and save it to a file.

Example:

```bash
$ rna_plot_hist.py rmsds.csv --column rmsd_all
```

                   file

Positional arguments:
  file           rmsd.txt

Options:
  --column       column of file to plot
  --sep=,        separator, be default
RNA-tools documentation, Release

-o, --output  Undocumented
--bins=10  Undocumented

**rna_plot_density.py** - generate a density plot

Don’t open Excel, Jupyter. Simple plot a density of one column and save it to a file.

**Example:**

```bash
# file

| 0 | 19_Bujnicki_Human_4_rpr_n0-000001.pdb-000001_A... | 14.73 |
| 1 | 19_Bujnicki_Human_4_rpr_n0-000001.pdb.trafl_19... | 0.46  |
| 2 | 19_Bujnicki_Human_4_rpr_n0-000001.pdb.trafl_19... | 14.73 |
| 3 | 19_Bujnicki_Human_4_rpr_n0-000001.pdb_thrs0.50... | 0.73  |
| 4 | 19_Bujnicki_Human_4_rpr_n0-000001.pdb_thrs0.50... | 0.83  |

$ rna_plot_hist.py rmsds.csv --column rmsd_all
```

**usage:** `rna_plot_density [-h] [--column COLUMN] [--sep SEP] [-o OUTPUT] file`

**Positional arguments:**

- `file`  rmsd.txt

**Options:**

- `--column`  column of file to plot
- `--sep=`  separator, be default
- `-o, --output`  Undocumented
3.13.2 rna_sali2dotbracket

usage: rna_sali2dotbracket [-h] filename

Positional arguments:

    filename            file in the Sali format

This beauty here will go to sali notation and convert it to dotbracket notation. The file name should be xxxx.sali

Author: Catarina Almeida

rna_tools.tools.rna_sali2dotbracket.rna_sali2dotbracket.convert_sali2dotbracket (fn)

The function needs a filename in the Sali format. This function will get the secondary structure of the sequence, then get its identifier and then the sequence itself.

To get the ss

The line with the secondary structure is a list and will look like this:

\['', '', '', '', '', '', '', '', '', '', '----<<<[[...]]>>>', '', '', '\n']

In this case, the ss is in the 11th position. But in some files it may be in the 12th, 13th, 10th, etc..

If the longest element from the list is extracted, then this problem is overcome.

The ss will some times have patterns of repeated gaps, which will come in the form of:

1. x
2. xnt
3. ( x )

With x being any number, from 1 to 1000. These must be converted to the correspondent number of gaps (-) in the converted ss. This conversion is done by:

1 - Identifying the pattern with regex
2 - Replacing it with repl function.

As such, the following expressions will replace the previously mentioned patterns:

1. re.sub(r'\d+\d', repl, temp)
2. re.sub(r'\d+\dnt', repl, temp)
3. re.sub(r'(\(?P<smthBeautiful>\d+\))', repl, temp)

To get the sequence

The sequence, much like the ss, can sometimes be in a different position in the list. Like in the ss, the longest element will be selected. Also, like in the ss, patterns for repeated gaps appear. So these must also be removed.

rna_tools.tools.rna_sali2dotbracket.rna_sali2dotbracket.get_parser()

rna_tools.tools.rna_sali2dotbracket.rna_sali2dotbracket.repl(m)

This function will replace the length of a given string by the correspondent number of dashes. The expression qwerty will be replaced by -----

3.13.3 Cluster load

A very simple tool to see your cluster load per user:
A super simple script to get some statistics of who is running at a cluster

Set MAX_JOBS to calc % of usage, it’s an approximation of max number of jobs, e.g. peyote ~1k (rather 700, e.g. FARNA runs.).

```python
rna_tools.tools.cluster_load.cluster_load.get_parser()
rna_tools.tools.cluster_load.cluster_load.per_user()
get stats (#cpus) per user
rna_tools.tools.cluster_load.cluster_load.stats_for_cluster()
get stats (#jobs) per cluster
rna_tools.tools.cluster_load.cluster_load.stats_for_user()
get stats (#jobs) per user
```

### 3.13.4 RNAkb

RNAkb (previous Gromacs) utils.

A module with different functions needed for Gromacs/RNAkb merriage.

Marcin Magnus Albert Bogdanowicz

1. prepare groups and then (2) mdp score file.

```python
rna_tools.tools.rnakb_utils.rnakb_utils.fix_gromacs_gro(path, verbose=False)
```

It’s probably a bug in GROMACS, but box coordinates in gro files are not always separated by spaces. This function guesses how it should be separated and inserts spaces.

**Parameters**

```
path = path to gro file(*=)
```

**Output:**

- file is overwritten with a corrected one

```python
rna_tools.tools.rnakb_utils.rnakb_utils.fix_gromacs_ndx(path)
```

Sometimes, GROMACS index has some atoms in more than one group, or doesn’t have all the groups grompp requires. This function fixes that.

**Parameters**

```
path = path to index file(*=)
```

**Output:**

- index is overwritten with a corrected one

```python
rna_tools.tools.rnakb_utils.rnakb_utils.format_score_mdp(mdp_out, energygrps, seq, verbose=False)
```

Get a template score mdp and replace energygrps (it can be generated with prepare_groups) and energygrp_table
**get_res_code**

Get residue code from a line of a PDB file.

**get_res_num**

Extract residue number from a line of a PDB file.

**make_rna_gromacs_ready**

GROMACS has some special requirements for PDB files.

**make_rna_rnakb_ready**

RNAkb read (difference between this function and make_rna_gromacs_ready is ignoring R5U etc. RNAkb does not treat them differently so there is no point to distinguish them.

**prepare_groups**

Prepare an index for fn file. gr_fn is a file where gtxt is saved in.

```
@todo RG5 etc – done!
```

gtxt:

```
1
  r RU & a C1
  name 1 uc1s
  r RU & a C2
  name 2 uc2
  r RU & a C2'
  name 3 uc2s
  ...
```

return, gtxt (groups_txt), energygrps . The result is saved to g_fn. energygrps: ['uP', 'uc4s', 'uc2', 'uc4', 'uc6', 'gP', 'gc4s', 'gc2', 'gc4', 'gc6', 'ap', 'ac4s', 'ac2', 'ac4', 'ac6'] gtxt: RA del 1 r RU & a P name 1 uP r RU & a C4s name 2 uc4s r RU & a C2 name 3 uc2 r RU & a C4 [..] r RA & a C6 name 15 aC6 11213145/6/7891011112131415 name 16 RNA_5pt 0 & ! 16 name 17 other q
Set residue code from a line of a PDB file
4.1 PyMOL4Spliceosome: all spliceosome structures in one PyMOL session

Download one PyMOL session with structures for each step.
See own project https://github.com/mmagnus/PyMOL4Spliceosome

4.2 PyMOL: color by conservation

Show conserved regions of proteins in PyMOL.
4.3 PyMOL Drawing

```python
draw_circle(x, y, z, r=8.0, cr=1.0, cg=0.4, cb=0.8, w=2.0)
```

Create a CGO circle

**PARAMS**

- `x, y, z` X, Y and Z coordinates of the origin
- `r` Radius of the circle
- `cr, cg, cb` Color triplet, [r,g,b] where r,g,b are all [0.0,1.0].
- `w` Line width of the circle

**RETURNS** the CGO object (it also loads it into PyMOL, too).
RNA-Tools documentation, Release

RNA-Tools tools.pymol_drawing.pymol_drawing.draw_circle_selection (selName, r=None, cr=1.0, cg=0.4, cb=0.8, w=2.0)

circleSelection – draws a cgo circle around a given selection or object

PARAMS

selName Name of the thing to encircle.

r Radius of circle. DEFAULT: This script automatically defines the radius for you. If you select one atom and the resultant circle is too small, then you can override the script’s calculation of r and specify your own.

cr, cg, cb red, green and blue coloring, each a value in the range [0.0, 1.0]

RETURNS The circle object.

RNA-Tools tools.pymol_drawing.pymol_drawing.draw_dist (54.729, 28.9375, 41.421, 55.342, 35.3605, 42.745)

https://sourceforge.net/p/pymol/mailman/message/25795427/

RNA-Tools tools.pymol_drawing.pymol_drawing.draw_dists (interactions)

RNA-Tools tools.pymol_drawing.pymol_drawing.draw_vector (x1, y1, z1, x2, y2, z2)

https://pymolwiki.org/index.php/CGOCylinder

Install PyMOL plugin to view the interactions with PyMOL:

run <path> rna-tools/tools/pymol_drawing/pymol_dists.py

and type:

draw_dists([[29, 41], [7, 66], [28, 42], [51, 63], [50, 64], [2, 71], [5, 68], [3, 70], [31, 39], [4, 69], [6, 67], [12, 23], [52, 62], [30, 40], [49, 65], [27, 43], [11, 24], [1, 72], [10, 25], [15, 48], [53, 61], [19, 56], [13, 22], [36, 37], [18, 19], [22, 46], [35, 73], [32, 38], [9, 13], [19, 20], [18, 20], [54, 60], [9, 23], [34, 35], [36, 38], [53, 54], [20, 56], [9, 12], [26, 44], [18, 55], [54, 61], [32, 36]])

4.3. PyMOL Drawing 107
4.4 Install

Open your ~/.pymolrc and set up following variables as you need:

```bash
# rna-tools
RNA_TOOLS="/Users/magnus/work-src/rna-tools"
EXECUTABLE="/bin/zsh" # set up your shell, usually /bin/bash or /bin/zsh
SOURCE="source ~/.zshrc" # set up the path to the file where you keep your shell variables
CLARNA_RUN="/Users/magnus/work-src/clarna_play/clarna_run.py" # if you want to run clarna_run.py set up the path
sys.path.append('/Users/magnus/work-src/rna-tools') # if you want to run the plugins with PyMOL
run ~/work-src/rna-tools/rna_tools/tools/PyMOL4RNA/PyMOL4RNA.py
run ~/work-src/rna-tools/rna_tools/tools/pymol_drawing/pymol_drawing.py
run ~/work-src/rna-tools/rna_tools/tools/rna_filter/pymol_dists.py
```

The plugins have been tested with MacPyMOL version 1.7.4.5 Edu.
These functions are intended to be imported to tools to give a unified framework for making selections.

```python
rna_tools.tools.extra_functions.select_fragment.select_pdb_fragment(txt, separator='-', splitting='[:\+]', verbose=False)
```

Take `txt` such as `A:1-31+B:1-11` and parse into:

```python
OrderedDict([('A', [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]), ('B', [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]))
```

**Warning:** e.g. for A:1-31, resi 31 is included

```python
rna_tools.tools.extra_functions.selectFragment.select_pdb_fragment_pymol_style(txt)
```

Take `txt` such as `A/10-15/P` and parse into:

```python
A/57/O2' -> ['A', ['57'], "O2'"]
```

If you want to combine a few subselections, please use `,`:

```python
--model_ignore_selection "A/57/O2',A/58/O2'"
```

**Warning:** e.g. for A:1-31, resi 31 is included
6.1 Example #1

The native:

```
[mq] md_1msy_clx cat 1msy_clean.pdb.outCR
Classifier: Clarna
chains: A 2647 2673
A 2648 A 2672 bp G U WW_cis 0.8732
A 2649 A 2671 bp C G WW_cis 0.9160
A 2650 A 2670 bp U A WW_cis 0.9289
A 2651 A 2669 bp C G WW_cis 0.9439
A 2652 A 2668 bp C G WW_cis 0.9281
A 2655 A 2656 bp G U SH_cis 0.9227
A 2656 A 2665 bp U A WH_tran 0.8526
A 2657 A 2664 bp A G HS_tran 0.8513
A 2658 A 2663 bp C G WW_cis 0.9421
A 2659 A 2662 bp G A SH_tran 0.7619
```

but analyzed structures are like:

```
[mq] md_1msy_clx cat struc/1msy_rnakbmd_decoy1478_clx.pdb.outCR
Classifier: Clarna
chains: A 1 27
2 26 bp G U WW_cis 0.7196
3 25 bp C G WW_cis 0.6702
4 24 bp U A WW_cis 0.8911
5 23 bp C G WW_cis 0.8925
6 22 bp C G WW_cis 0.9026
9 10 bp G U SH_cis 0.8714
10 19 bp U A WH_tran 0.7279
11 18 bp A G HS_tran 0.8810
12 17 bp C G WW_cis 0.9115
```

You have to renumber 1msy_clean.pdb to 1:27:
6.2 Example #2

Listing:

```bash
$ rna_pdb_toolsx.py --get-seq 1nuj_rnakbmd_decoy1000_clx.pdb
  1nuj_rnakbmd_decoy1000_clx.pdb A:1-13
  CGGACCGAGCCAG
  1nuj_rnakbmd_decoy1000_clx.pdb B:14-24
  GCUGGGAGUCC

$ rna_pdb_toolsx.py --get-seq 1nuj_clean.pdb
  1nuj_clean.pdb A:18-30
  CGGACCGAGCCAG
  1nuj_clean.pdb B:39-49
  GCUGGGAGUCC


$ rna_pdb_toolsx.py --get-seq 1nuj_clean_renumber.pdb
  1nuj_clean_renumber.pdb A:1-13
  CGGACCGAGCCAG
  1nuj_clean_renumber.pdb B:14-24
  GCUGGGAGUCC
```
6.3 Example #3

Starting structure doesn’t have chain id:

```bash
# add chain A
$ parallel "rna_add_chain.py -c A {} > ../struc_with_chain/{}" ::: *.pdb
# edit the second part of the new chain A as B
$ parallel "rna_pdb_gtools.py --edit 'A:14-27>B:14-27' {} > out/{}" ::: *.pdb
```

```
/1duq_rnakbnn_decay0273_amb_clx/A/ 1  CUGGCAGG
/1duq_rnakbnn_decay0008_amb_clx/A/ 1  CUGGCAGG
```

6.4 Example #4 Calculate RMSDs of unstandardized structures (RNA Puzzle #1)

You try to calculate RMSDs for RNA Puzzles #1:

```bash
rna_calc_rmsd.py -t 1_solution_0_rpr.pdb *.pdb
method: all-atom-built-in
# of models: 15
1_bujnicki_1_rpr.pdb 5.71 978
1_bujnicki_2_rpr.pdb 6.16 978
1_bujnicki_3_rpr.pdb 5.3 978
1_bujnicki_4_rpr.pdb 4.95 978
1_bujnicki_5_rpr.pdb 5.1 978
Error: # of atoms is not equal target (1_solution_0_rpr.pdb):978 vs model (1_chen_1_rpr.pdb):975
```
you can see that there is a different number of atoms in $1\_solution\_0\_rpr.pdb$ and $1\_chen\_1\_rpr.pdb$.

To see more you can run `diffpdb`.

![Comparison of pdb files](image)

you can see that something is wrong. To fix it, run:

```bash
$ rna_pdb_toolsx.py --rpr --inplace *.pdb
```

you can tail the files:

```bash
tail *.pdb
```

```bash
==> 1_bujnicki_1_rpr.pdb <==
ATOM 971 N7  G B 23 -16.558 -3.375 78.345 1.00 0.00 N
ATOM 972 C5  G B 23 -17.169 -2.575 77.384 1.00 0.00 C
ATOM 973 C6  G B 23 -17.589 -2.874 76.053 1.00 0.00 C
ATOM 974 O6  G B 23 -17.497 -3.930 75.430 1.00 0.00 O
ATOM 975 N1  G B 23 -18.234 -1.800 75.459 1.00 0.00 N
ATOM 976 C2  G B 23 -18.441 -0.576 76.049 1.00 0.00 C
ATOM 977 N2  G B 23 -19.127 0.345 75.382 1.00 0.00 N
ATOM 978 N3  G B 23 -18.053 -0.282 77.292 1.00 0.00 N
ATOM 979 C4  G B 23 -17.419 -1.324 77.898 1.00 0.00 C
...  
==> 1_chen_1_rpr.pdb <==
ATOM 971 N7  G B 23 -14.462 -1.101 79.998 1.00 0.00 N
ATOM 972 C5  G B 23 -14.952 -0.485 78.839 1.00 0.00 C
ATOM 973 C6  G B 23 -15.577 -1.020 77.655 1.00 0.00 C
ATOM 974 O6  G B 23 -15.822 -2.189 77.351 1.00 0.00 O
ATOM 975 N1  G B 23 -15.972 -0.576 76.049 1.00 0.00 N
ATOM 976 C2  G B 23 -15.787 1.274 76.944 1.00 0.00 C
ATOM 977 N2  G B 23 -15.224 1.822 78.022 1.00 0.00 N
ATOM 978 N3  G B 23 -14.818 0.884 78.935 1.00 0.00 C
```
so now you can see that the files look the same. Let’s try to calculate RMSDs again:

```
rna_calc_rmsd.py -t 1_solution_0_rpr.pdb *.pdb
method: all-atom-built-in
# of models: 16
1_bujnicki_1_rpr.pdb 5.71 978
1_bujnicki_2_rpr.pdb 6.16 978
1_bujnicki_3_rpr.pdb 5.3 978
1_bujnicki_4_rpr.pdb 4.95 978
1_bujnicki_5_rpr.pdb 5.1 978
1_chen_1_rpr.pdb 4.35 978
1_chen_2_rpr.pdb 4.35 978
1_das_1_rpr.pdb 3.97 978
1_das_2_rpr.pdb 4.48 978
1_das_3_rpr.pdb 3.43 978
1_das_4_rpr.pdb 3.92 978
1_das_5_rpr.pdb 4.57 978
1_dokholyan_1_rpr.pdb 7.25 978
1_major_1_rpr.pdb 4.34 978
1_santalucia_1_rpr.pdb 5.76 978
1_solution_0_rpr.pdb 0.0 978
# of atoms used: 978
csv was created! rmsds.csv
```

worked! :-)  

This is a real-life case, https://github.com/mmagnus/RNA-Puzzles-Normalized-submissions/tree/master/rp01.
7.1 Run in batch

You can easily run a single tool in batch and rename new files:

```bash
$ for i in *.pdb; do rna_pdb_toolsx.py --get_rnapuzzle_ready $i > ${i/.pdb/_rpr.pdb}; done
```

or write new files in a different folder (out):

```bash
$ for i in *.pdb; do rna_pdb_toolsx.py --get_rnapuzzle_ready $i > ../out/$i; done
```

You can also easily run a single tool parallel using `parallel`:

```bash
$ parallel "rna_add_chain.py -c A {} > ../nchain/{}" ::: *.pdb
```

7.2 Using sed

sed (stream editor) is a Unix utility that parses and transforms text, using a simple, compact programming language.

You can used sed to find & replace parts of text files:

```bash
$ head lmsy_rnakbmd_decoy1661_clx.pdb.outCR
Classifier: Clarna
chains:  1 27
  2   26   bp G U  WW_cis  0.8500
  3   25   bp C G  WW_cis  0.8114
  4   24   bp U A  WW_cis  0.9222
  5   23   bp C G  WW_cis  0.9038
  6   22   bp C G  WW_cis  0.8913
  9   10   bp G U  SH_cis  0.8563
 10   19   bp U A  WH_tran 0.7826
```
$ sed 's/chains: /chains: A/' lmsy_rnakbmd_decoy1661_clx.pdb.outCR
Classifier: Clarna

chains: A 1 27
  2   26  bp G U  WW_cis  0.8500
  3   25  bp C G  WW_cis  0.8114
  4   24  bp U A  WW_cis  0.9222
  5   23  bp C G  WW_cis  0.9038
  6   22  bp C G  WW_cis  0.8913
  9   10  bp G U  SH_cis  0.8563
 10   19  bp U A  WH_tran  0.7826
 11   18  bp A G  HS_tran  0.7620
 12   17  bp C G  WW_cis  0.7242

Read more about sed.

### 7.3 In PyMOL

Quickref:

```plaintext
set ignore_case, off
```

Rename a chain:

```plaintext
PyMOL> alter (sele), chain="B"
Alter: modified 708 atoms.
PyMOL> sort
don’t forget about sort.
```

To select all, use `PyMOL> alter all, resv -= 12`.

To renumber a fragment starting with 24 to 29, select the fragment and:

```plaintext
PyMOL> alter (sele), resv += 5
Alter: modified 109 atoms.
```

To renumber residues:

```plaintext
PyMOL> alter (chain B), resv -= 44
Alter: modified 708 atoms.
PyMOL> sort
```

Read more.

The example of the pistol ribozyme editing.
Run:

PyMOL> alter (sele), chain="B"
  Alter: modified 236 atoms.
PyMOL> alter (chain B), resv -= 51
  Alter: modified 236 atoms.
PyMOL> sort

7.3. In PyMOL
7.4 In Python

To get residue index use:

```python
resi = int(l[22:26].strip())
```

Quickref:

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>PYTHON</th>
<th>DATA TYPE</th>
<th>FIELD</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 6</td>
<td>[0:6]</td>
<td>Record name</td>
<td>“ATOM”</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>[16]</td>
<td>Character</td>
<td>altLoc</td>
<td>Alternate location indicator.</td>
</tr>
<tr>
<td>18 - 20</td>
<td>[17:20]</td>
<td>Residue name</td>
<td>resName</td>
<td>Residue name.</td>
</tr>
<tr>
<td>23 - 26</td>
<td>[22:26]</td>
<td>Integer</td>
<td>resSeq</td>
<td>Residue sequence number.</td>
</tr>
<tr>
<td>31 - 38</td>
<td>[30:38]</td>
<td>Real(8.3)</td>
<td>x</td>
<td>Orthogonal coordinates for X in → Angstroms.</td>
</tr>
<tr>
<td>39 - 46</td>
<td>[38:46]</td>
<td>Real(8.3)</td>
<td>y</td>
<td>Orthogonal coordinates for Y in → Angstroms.</td>
</tr>
<tr>
<td>47 - 54</td>
<td>[46:54]</td>
<td>Real(8.3)</td>
<td>z</td>
<td>Orthogonal coordinates for Z in → Angstroms.</td>
</tr>
<tr>
<td>55 - 60</td>
<td>[54:60]</td>
<td>Real(6.2)</td>
<td>occupancy</td>
<td>Occupancy.</td>
</tr>
<tr>
<td>77 - 78</td>
<td>[76:78]</td>
<td>LString(2)</td>
<td>element</td>
<td>Element symbol, right-justified. #</td>
</tr>
<tr>
<td>79 - 80</td>
<td>[78:80]</td>
<td>LString(2)</td>
<td>charge</td>
<td>Charge on the atom.</td>
</tr>
</tbody>
</table>
7.5 Working with cluster

Tips:

```
# get your pdb files
[mm] ade rsync -v peyote2:'~/ade/*' . # ' ' is required!
```

See long name with qstat:
7.6 Numbering line used in my flat-file notes

Numbering:

```
|1.......|10.......|20.......|30.......|40.......|50.......|60.......|70.......|80....
--+--------+--------+--------+--------+--------+--------+--------+--------+--------+
|123456789112345678921234567893123456789412345678951234567896123456789712345678981234567899123456789
```

7.7 TER format

Example of pro TER:

```
ATOM  72307  C4  U   x9     304.768 147.960 320.897 1.00218.84 C
ATOM  72308  O4  U   x9     304.171 146.902 321.104 1.00225.09 O
ATOM  72309  C5  U   x9     304.190 149.269 320.912 1.00211.91 C
ATOM  72310  C6  U   x9     304.960 150.336 320.668 1.00205.76 C
TER   72311  U   x9
```
CHAPTER 8

RNA Puzzle Submission

The RNA Puzzle organizers required ONE file with your submissions in the NMR-style multiple model PDB format. First, prepare your structures in the folder and run to get them RNApuzzle ready (_rpr):}

```bash
$ for i in *.pdb; do rna_pdb_toolsx.py --get-rnapuzzle-ready $i > ${i/.pdb/_rpr.pdb}; done
```

merge them as one file in the order as you like (or use *):

```bash
$ rna_pdb_merge_into_one.py 02_19pz_v1_SimRNA3.22_thrs6.60A_clust02-000001_AA_out_rpr.pdb \ 
   09_19pz_v2_SimRNA3.22_thrs6.60A_clust03-000001_AA_out_rpr.pdb \ 
   d311d821-a075-4df0-bd7d-1dcf7669dad9_ALL_thrs6.20A_clust01-000001_AA_out_rpr.pdb \ 
   d311d821-a075-4df0-bd7d-1dcf7669dad9_ALL_thrs6.20A_clust03-000001_AA_out_rpr.pdb \ 
   05_19pz_v1_SimRNA4.xx_thrs6.60A_clust02-000001_AA_out_rpr.pdb > rp19_bujnicki.pdb
```

and verify your file with the template provided by the organizers (if provided):

```bash
$ diffpdb --method diff Reference_19.pdb rp19_bujnicki.pdb
#<empty = no difference but xyz columns, OK!>
```

diffpdb is a part of the rna-tools package.
In terminal:

```bash
$ head -n 5 Reference_19.pdb rp19_bujnicki.pdb
===> Reference_19.pdb ====
MODEL 1
ATOM 1 P G A 1 0.000 0.000 0.000 1.00 0.00 P
ATOM 2 OP1 G A 1 0.000 0.000 0.000 1.00 0.00 O
ATOM 3 OP2 G A 1 0.000 0.000 0.000 1.00 0.00 O
ATOM 4 O5' G A 1 0.000 0.000 0.000 1.00 0.00 O
===> rp19_bujnicki.pdb ====
MODEL 1
ATOM 1 P G A 1 31.463 14.180 -0.676 1.00 0.00 P
ATOM 2 OP1 G A 1 31.412 12.806 -1.223 1.00 0.00 O
ATOM 3 OP2 G A 1 30.646 15.083 -1.517 1.00 0.00 O
ATOM 4 O5' G A 1 30.955 14.212 0.842 1.00 0.00 O

$ tail -n 5 Reference_19.pdb rp19_bujnicki.pdb
===> Reference_19.pdb ====
ATOM 1325 C5 C B 22 0.000 0.000 0.000 1.00 0.00 C
ATOM 1326 C6 C B 22 0.000 0.000 0.000 1.00 0.00 C
TER 1327 C B 22
ENMDL
END
===> rp19_bujnicki.pdb ====
ATOM 1325 C5 C B 22 29.927 21.506 -6.542 1.00 0.00 C
ATOM 1326 C6 C B 22 29.822 22.338 -5.500 1.00 0.00 C
TER 1327 C B 22
ENMDL
END
```

The Bujnicki lab is using this script to process the results and send the results, you can hack it for your own case:

```
[mm] rnapuzzle_sender$ git:(master) ./rnapuzzle_sender.py
rnapuzzle_sender

Usage: rnapuzzle_sender.py
```
Options:
- `h, --help`  show this help message and exit
- `d DIR, --dir=DIR`
- `s EMAIL_SUBJECT, --email_subject=EMAIL_SUBJECT`  
  email subject

CHAPTER 9

Jupyter Notebooks & rna-tools

The tools collected here as rna-tools, ideally could be also used in IPython/Jupyter (https://ipython.org/notebook.html) Notebooks. We believe it would be valuable if we could, as scientists share our notebooks used for RNA structure analysis, e.g., protocols of modeling used in the RNA Puzzle challenge.

```
In [2]: seq = Seq.Seq("GGUGAGGCUCGGAGAAAGUCACAGUUUGGGGAAAGCGUGCAUACCACCCACGAAGUGG")
In [3]:
Out[3]: GGGUGAGGCUCGGAGAAAGUCACAGUUUGGGGAAAGCGUGCAUACCACCCACGAAGUGG
```

**Secondary structure prediction**

```
In [12]: print seq.predict_ss()

(((((........((((..)))))(((........))))))....)...))....)

In [13]: print seq.predict_ss(method='RNAsubopt')

GGUGAGGCUCGGAGAAAGUCACAGUUUGGGGAAAGCGUGCAUACCACCCACGAAGUGG 331.0 100
(((((........((((..)))))(((........))))))....)...))....)

In [14]: print seq.predict_ss(method='ipKnot')

(((((........((((..)))))(((........))))))....)...))....)

In [15]: print seq.predict_ss(method='centroid_fold')

(((((........((((..)))))(((........))))))....)...))....) (g=1, th=0.5, e=27.26)

In [16]: print seq.predict_ss(method='contextfold')

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

9.1 Share your notebooks

You can share Jupyter notebooks from your Google Drive using the new Jupyter Drive. This allows you share Jupyter notebooks like NBViewr with all the access control that Google Drive provides. You can also push your notebooks to a Github repository, so then can be rendered for easy viewing.


9.2 Learn Jupyter

This is a quick introduction to jupyter which is the IPython version 3. It covers some of the new and interesting features about Jupyter.

https://www.youtube.com/watch?v=Rc4JQWowG5I
Emacs & rna-tools

rna-tools can be used side-by-side https://github.com/mmagnus/emacs-pdb-mode to edit files structural files in the PDB format.

Emacs /imæks/ and its derivatives are a family of text editors that are characterized by their extensibility.[2] The manual for the most widely used variant, GNU Emacs, describes it as “the extensible, customizable, self-documenting, real-time display editor”. Development of the first Emacs began in the mid-1970s and continues actively as of 2017. (https://en.wikipedia.org/wiki/Emacs)

pdb-mode (https://github.com/mmagnus/emacs-pdb-mode not authored by me, I’m a maintainer, and a beginner developer) is an emacs-lisp minor mode for Emacs to perform a number of useful editing functions on Protein DataBank (PDB) formatted files. XEmacs and/or GNU Emacs are available for most computing platforms.

Youtube video: https://www.youtube.com/embed/o99YFbLSVRw
Warning: Still a very proof of concept. More soon.
11.1 Draw VARNA-based image of RNA secondary structure

Type:

```markdown
<pre>
[ss:rna]
UUUCUGUAUAUGGCCAUAUAAGGUUCGCGAUACUACUAACACAGCCGUAACUGUUUGACUACAGUAA
{{.(((((((......))))))))..}}..}}{{.((((.........)))))}.{{.((...((....)...))}}}..)
</pre>
```

**Warning:** Keep exactly the same syntax as in the example above and below.

The syntax:

```markdown
<pre>
[ss:/name of your seq/]
/seq/
/seq/
</pre>
```

# ^ not <pre/> nor <pre>. Keep a new line after this syntax. So don't do:
Warning: This plugin will change your Markdown file, so make sure that your editor will detect this change and ask you to reload the file!

to get a VARNA-drawn image of secondary structure.
Download rna-tools by clicking here https://github.com/mmagnus/rna-tools/archive/master.zip (~30 MB) and unpack the zip file, enter the folder:

```
$ cd rna-tools-master
```

OR use git:

```
$ git clone https://github.com/mmagnus/rna-tools.git
$ cd rna-tools
```

`git` is better if you want to contribute to the package or/and you want to get pretty frequent updates.

The first step is “zero” because not all requirements are needed to start working with rna-tools. If anything is missing you can install it later.

To install the full set of requirements, use `pip`:

1. **Setup the package paths by adding to your ~/.bashrc or ~/.zshrc following code:**

   ```
   export RNA_TOOLS_PATH=<PATH TO YOUR RNA_TOOLS>
   export PYTHONPATH=$PYTHONPATH:$RNA_TOOLS_PATH
   export PATH=$PATH:$RNA_TOOLS_PATH'/bin/
   ```

   For example in my case it looks as:

   ```
   export RNA_TOOLS_PATH=/home/magnus/work-src/rna-tools/
   export PYTHONPATH=$PYTHONPATH:$RNA_TOOLS_PATH
   export PATH=$PATH:$RNA_TOOLS_PATH'/bin/
   ```

2. **And run the installation script:**

   ```
   rna-tools git:(master) ./install_links_bin.sh
   Installed in ./bin
   rmsd_calc_to_target.py
   ```
3. Run rna_tools_test_all.py to see if you got any errors, this should look like:

```python
(py37) [mm] rna-tools$ git:(master) rna_tools_test_all.py
BlastPDB requires urllib3
- Python: 3.7.4 (default, Aug 13 2019, 15:17:50) [Clang 4.0.1 (tags/RELEASE_401/→final)]
- rna-tools: b'py2-78-g3b3dd5f'
- RNA_TOOLS_PATH set to /home/magnus/work-src/rna-tools/
- See full list of tools <https://github.com/mmagnus/rna-tools/blob/master/rna-→tools-index.csv>
Seems OK
```

or for Python 2:

```python
(base) [mm] rna-tools$ git:(master) rna_tools_test_all.py
- Python: 2.7.16 |Anaconda, Inc.| (default, Mar 14 2019, 16:24:02) [GCC 4.2.1←Compatible Clang 4.0.1 (tags/RELEASE_401/final)]
- rna-tools: py2-78-g3b3dd5f
- RNA_TOOLS_PATH set to /home/magnus/work-src/rna-tools/
- See full list of tools <https://github.com/mmagnus/rna-tools/blob/master/rna-tools-→index.csv>
Seems OK
```

To set you own configuration, please first:

```bash
cp rna_tools_config_local.py_sample rna_tools_config_local.py # in rna-tools/rna_tools
```

and then edit rna_tools_config_local.py as you need. In my case it is:

```bash
rna_tools git:(master) cat rna_tools_config_local.py
VARNA_PATH = '/Users/magnus/skills/rnax/varna_tut/
VARNA_JAR_NAME = 'VARNA.jar'
```
For Git in a scientific environment, please read:


Git is a version control system that is used for software development that helps you to keep track of versions of your program. To start using git you have to know only these two commands below. If you want to contribute to the package you need a few more, but it’s not important right now :-)

To get the package for the first time on your computer go to the selected package and on the top left corner select clone or download. Copy the link and run it on console with git clone:

```
$ git clone git@github.com:mmagnus/rna-pdb-tools.git
```

and if you want to update the package later run:

```
$ git pull # be in the folder like ~/src/rna-pdb-tools/ <here>
```

if you see something like this:

```
$ git pull
Already up-to-date.
```

it means that your version of the package is up to date, congrats! :-)}
If you see something like this:

```bash
$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 2), reused 3 (delta 2), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:mmagnus/rna-pdb-tools
   69c4ee3..7f90739 master    -> origin/master
Updating 69c4ee3..7f90739
Fast-forward
install_links_bin.sh | 1 +
1 file changed, 1 insertion(+)
```

it means that there is a small change in `install_links_bin.sh` and you are up to date, congrats as well! You might need to run `./install_links_bin.sh` to “install” new tools that were added to the packages (if this is the case). If you get any error then talk to me magnus@genesilico.pl.

### 13.1 Git sheet cheat

```bash
$ git init # start git repo in a folder
$ git add <file> # add file to stage area
$ git commit -m <text> # send file to
$ git push # sent this remote
$ git gui # install `apt-get install git-gui`
$ git log #
$ git status # get status of your repo

$ git clone git@gitlab.genesilico.pl:magnus/git_crash_course.git

$ git remote add origin git@gitlab.genesilico.pl:magnus/git_crash_course.git

$ git push -u origin master
Counting objects: 45, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (41/41), done.
Writing objects: 100% (45/45), 4.97 KiB | 0 bytes/s, done.
Total 45 (delta 12), reused 0 (delta 0)
To git@gitlab.genesilico.pl:magnus/git_crash_course.git
 * [new branch] master -> master
Branch master set up to track remote branch master from origin.
```

### 13.2 Learn Git

Model used by Git vs Svn:

```
file <-> stage area <-> local repo <-> git repo/gitlab
file --------------------------------------> SVM repo
```
For more see, git - the simple guide (just a simple guide for getting started with git. no deep shit :))
http://rogerdudler.github.io/git-guide/


If you want to read more, we highly recommend a book our collages, Kristian (http://www.apress.com/us/book/9781484222409).

To understand the principles of Git, a bit more advance, by Linus Torvals (an author of Git) https://www.youtube.com/watch?v=4XpnKHJAok8

How to learn Git in 20min https://www.youtube.com/watch?v=Y9XZQO1n_7c

### 13.3 Git GUI

You don’t have to use terminal to work with git. Git comes with `git gui`. 
You can also use qgit (http://sourceforge.net/projects/qgit/) and much more, a list of tools: https://git-scm.com/download/gui/linux.
Keep configuration syntax like:

```python
from rna_tools.rna_tools_config import CPUS_CLUSTER
# since we use export PYTHONPATH=$PYTHONPATH:/home/magnus/src/rna-tools/
```

vs:

```python
try:
    RNA_ROSETTA_RUN_ROOT_DIR_MODELING = os.environ['RNA_ROSETTA_RUN_ROOT_DIR_MODELING']
except:
    print ('Set up RNA_ROSETTA_RUN_ROOT_DIR_MODELING in .bashrc')
```
We are using (at least we are moving towards) the Google style docstrings via Napoleon. Napoleon is a Sphinx Extensions that enables Sphinx to parse both NumPy and Google style docstrings - the style recommended by Khan Academy. http://www.sphinx-doc.org/en/stable/ext/napoleon.html#type-annotations
Add a new tool to the package

1. Create a new folder in `rna-tools/rna_tools/tools` with your tool. The folder will be seen online after your push at https://github.com/mmagnus/rna-tools/tree/master/rna_tools/tools. We will walk you through this simple example https://github.com/mmagnus/rna-tools/tree/master/rna_tools/tools/renum_pdb_to_aln.

2. Make sure that there is a simple test as `test.sh`:

   ```bash
   #!/bin/bash
   python renum_pdb_to_aln.py --residue_index_start 1 obj1 test_data/ALN_OBJ1_OBJ2.fa test_data/obj01.pdb
   
   and there is a `test_data` folder with some test inputs and outputs. See the example.

3. Add your tool to `install_links_bin.sh` at the top folder of `rna-tools`:

   ```bash
   ln -s $curr_dir/rna_tools /tools/<tool folder>/<util script name with .py> $curr_dir/bin/<util script name with .py>
   
   e.g.
   
   ln -s $curr_dir/rnatools/utils/renum_pdb_to_aln/renum_pdb_to_aln.py $curr_dir/bin/renum_pdb_to_aln.py
   
   This will “install” your script in bin directory of the project so it can be used system-wide.

   Run this script to see if there is any error, `./install_links_bin.sh`.

4. Add your tool to the documentation. The tool has to be “importable”, so don’t forget to create `__init__.py` inside your tool directory. Next, go to `rna-tools/docs/source` and edit `tools.rst`. Add, wherever you think your tool will fit, lines like:

   ```rst
   Renumber a pdb file according to alignment
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   .. argparse::
      :ref: rna_tools.tools.<tool folder>.<tool script name>.get_parser
      :prog: <util script name>
   ```
.. automodule:: rna_tools.tool.<tool folder>.<tool script name>
   :members:

.. _:members:

.. _:members:
e.g.:

Reumber a pdb file according to alignment
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

.. argparse::
   :ref: rna_tools.tool.renum_pdb_to_aln.renum_pdb_to_aln.get_parser
   :prog: renum_pdb_to_aln

.. automodule:: rna_tools.tool.renum_pdb_to_aln.renum_pdb_to_aln
   :members:

and run `make html` in the folder to check if the documentation is compiled without any errors.

If you are using any external library such as `scipy`, please make sure that they are listed in `rna-tools/docs/requirements.txt`. If the library is not there, please add it. This file is read by the Read The Docs to compile the documentation online and also by Travis for continuous testing.


5. The very last step is to add your tool `test.sh` to the main testing script. Edit `rna-tools/test.sh` and add

```bash
 cd ./tools/<tool folder>/
 ./test.sh
cd ..

e.g.

cd ./tools/renum_pdb_to_aln/
./test.sh
cd ..
```

6. Run this main test (`./test.sh`) and see if the tool works as expected.

7. Now we are ready to push the changes. In the terminal, type:

```bash
$ git pull
$ git add <files> # or use git gui
$ git commit -m <desc the tool>
$ git push
```

to commit all your changes and push it to the Github repository!

**Warning:** This testing is very, very rough and we are moving to have more test in py.test at some point.
CHAPTER 17

Indices and tables

• genindex
• modindex
• search
Python Module Index

rna_tools.BlastPDB, 38
rna_tools.RfamSearch, 38
rna_tools.rna_tools_lib, 18
rna_tools.SecondaryStructure, 34
rna_tools.Seq, 29
rna_tools.tools.clanstix.rna_clanstix, 95
rna_tools.tools.clarna_app.clarna_app, 79
rna_tools.tools.cluster_load.cluster_load, 101
rna_tools.tools.diffpdb.diffpdb, 92
rna_tools.tools.extra_functions.select_fragment, 109
rna_tools.tools.misc.rna_add_chain, 25
rna_tools.tools.pdbs_measure_atom_dists.pdbs_measure_atom_dists, 26
rna_tools.tools.pymol_drawing.pymol_drawing, 106
rna_tools.tools.renum_pdb_to_aln.renum_pdb_to_aln, 62
rna_tools.tools.rna_alignment.random_assignment_of_nucleotides, 58
rna_tools.tools.rna_alignment.rna_alignment, 39
rna_tools.tools.rna_calc_inf.rna_calc_inf, 71
rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd, 65
rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd_all_vs_all, 70
rna_tools.tools.rna_convert_pseudoknot_formats.rna_ss_pk_to_simrna, 37
rna_tools.tools.rna_filter.pymol_dists, 75
rna_tools.tools.rna_rosetta.rna_rosetta_cluster, 83
rna_tools.tools.rna_sali2dotbracket.rna_sali2dotbracket, 100
rna_tools.tools.rna_x3dna.rna_x3dna, 4
rna_tools.tools.rnakb_utils.rnakb_utils, 101
rna_tools.tools.simrna_trajectory.simrna_trajectory, 88
Index

A
add_ids() (rna_tools.tools.clanstix.rna_clanstix.RNAStructClans method), 97
align_seq() (rna_tools.tools.rna_alignment.rna_alignment.RNAalignment method), 45
Atom (class in rna_tools.tools.simrna_trajectory.simrna_trajectory), 88

B
BlastPDB (class in rna_tools.BlastPDB), 38

C
calc_rmsd() (in module rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd), 66
calc_rmsd() (in module rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd_all_vs_all), 70
calc_rmsd_pymol() (in module rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd), 66
check_symmetric() (in module rna_tools.tools.clanstix.rna_clanstix), 97
clama_compare() (in module rna_tools.tools.clarna_app.clarna_app), 79
clama_run() (in module rna_tools.tools.clarna_app.clarna_app), 80
clean_up() (rna_tools.tools.rna_x3dna.rna_x3dna.rna_x3DNA method), 78
cluster() (in module rna_tools.tools.rna_rosetta.rna_rosetta_cluster), 84
cluster_loop() (in module rna_tools.tools.rna_rosetta.rna_rosetta_cluster), 84
CMAlign (class in rna_tools.tools.rna_alignment.rna_alignment), 59
cmscan() (rna_tools.RfamSearch.RfamSearch method), 39
collapsed_view() (in module rna_tools.rna_tools_lib), 22
convert_sali2dotbracket() (in module rna_tools.tools.rna_sali2dotbracket.rna_sali2dotbracket), 100
copy_ss_cons_to_all() (rna_tools.tools.rna_alignment.rna_alignment.RNAalignment method), 45
copy_ss_cons_to_all_editing_sequence() (rna_tools.tools.rna_alignment.rna_alignment.RNAalignment method), 45

draw_circle() (in module rna_tools.tools.pymol_drawing.pymol_drawing), 106
draw_circle_selection() (in module rna_tools.tools.pymol_drawing.pymol_drawing), 106
draw_dist() (in module rna_tools.tools.pymol_drawing.pymol_drawing), 107
draw_dists() (in module rna_tools.tools.pymol_drawing.pymol_drawing), 107
draw_3s() (in module rna_tools.SecondaryStructure), 34
draw_ss() (rna_tools.tools.rna_alignment.rna_alignment.RNASeq method), 40
draw_vector() (in module rna_tools.tools.pymol_drawing.pymol_drawing), 107

e
edit_occupancy_of_pdb() (in module rna_tools.tools.rna_pdb_edit_occupancy_bfactor.rna_pdb_edit_occupancy_bfactor), 149
edit_occupancy_of_pdb()
    (rna_tools.rna_tools_lib.RNAStructure method), 18
edit_pdb() (in module rna_tools.rna_tools_lib), 17, 22
eval() (rna_tools.Seq.RNASequence method), 31
ExceptionOpenPairsProblem, 34
extract() (in module rna_tools.tools.rna_rosetta.rna_rosetta_cluster),
    84
fetch() (in module rna_tools.rna_tools_lib), 16, 22
fetch_ba() (in module rna_tools.rna_tools_lib), 17, 23
fetch_cif_ba() (in module rna_tools.rna_tools_lib), 23
find_core()
    (in module rna_tools.tools.pdbs_measure_atom_dists.pdbs_measure_atom_dists),
    26
find_core() (rna_tools.tools.rna_alignment.rna_alignment.RNAalignment method), 45
find_seq() (rna_tools.tools.rna_alignment.rna_alignment.RNAalignment method), 46
find_seq_exact() (rna_tools.tools.rna_alignment.rna_alignment.RNAalignment method), 46
fix_gromacs_gro()
    (in module rna_tools.tools.rnakb_utils.rnakb_utils), 101
fix_gromacs_ndx()
    (in module rna_tools.tools.rnakb_utils.rnakb_utils), 101
fix_O_in_UC()
    (rna_tools.rna_tools_lib.RNAStructure method), 18
fix_op_atoms()
    (rna_tools.rna_tools_lib.RNAStructure method), 18
fix_with_qrnas()
    (rna_tools.rna_tools_lib.RNAStructure method), 18
format_annotation() (rna_tools.tools.rna_alignment.rna_alignment.RNAalignment method), 46
format_score_mdp()
    (in module rna_tools.tools.rnakb_utils.rnakb_utils), 101
Frame (class in rna_tools.tools.simrna_trajectory.simrna_trajectory), 88
get_align()
    (in module rna_tools.tools.rna_alignment.random_assignment_of_nucleotides),
    58
get_all_chain_ids() (rna_tools.rna_tools_lib.RNAStructure method), 18
get_atom_code()
    (rna_tools.rna_tools_lib.RNAStructure method), 18
get_atom_coords()
    (rna_tools.rna_tools_lib.RNAStructure method), 18
get_atom_num()
    (rna_tools.rna_tools_lib.RNAStructure method), 18
get_atoms()
    (rna_tools.tools.simrna_trajectory.simrna_trajectory.Residue method), 88
get_center()
    (rna_tools.tools.simrna_trajectory.simrna_trajectory.Residue method), 88
get_ClaRNA_output_from_dot_bracket()
    (in module rna_tools.tools.clarna_app.clarna_app), 80
generate() (rna_tools.rna_alignment.rna_alignment.CMAlign method), 60
generate() (rna_tools.rna_alignment.rna_alignment.RNAalignment method), 47
generate() (rna_tools.rna_alignment.rna_alignment.RNASeq method), 47
generate() (rna_tools.rna_alignment.rna_alignment.RNASeq method), 47
get_clean_ss() (rna_tools.tools.rna_alignment.rna_alignment.RNAalignment method), 46
get_conserved() (rna_tools.tools.rna_alignment.rna_alignment.RNASeq method), 40
get_coord() (rna_tools.tools.simrna_trajectory.simrna_trajectory.Atom method), 88
get_distance_to() (rna_tools.tools.rna_alignment.rna_alignment.RNASeq method), 41
get_distances()
    (rna_tools.tools.rna_alignment.rna_alignment.RNASeq method), 41
get_dot_bracket_from_ClaRNAoutput()
    (in module rna_tools.tools.pdbs_measure_atom_dists.pdbs_measure_atom_dists), 80
get_foldability() (rna_tools.Seq.RNASequence method), 45
get_gc_rf() (rna_tools.tools.rna_alignment.rna_alignment.CMAlign method), 60
get_gc_rf() (rna_tools.tools.rna_alignment.rna_alignment.RNASeq method), 47
get_info_chains()
    (rna_tools.rna_tools_lib.RNAStructure method), 18
generate() (rna_tools.tools.clanstix.rna_clanstix), 97
get_is_defined() (rna_tools.rna_alignment.rna_alignment.RandomAssignmentOfNucleotides method), 37
get_modifications() (rna_tools.tools.rna_alignment.random_assignment_of_nucleotides), 58
get_no_structures() (rna_tools.tools.rna_alignment.rna_alignment.RNASeq method), 47
get_no_structures() (rna_tools.rna_alignment.rna_alignment.RNASeq method), 47
get_no_structures_in_first_cluster() (rna_tools.rna_alignment.rna_alignment.RNASeq method), 47
get_parser() (rna_tools.clanstix.rna_clanstix), 97
get_parser() (rna_tools.clanstix.rna_clanstix), 97
get_parser() (rna_tools.clanstix.rna_clanstix), 97
get_parser() (rna_tools.clanstix.rna_clanstix), 97
get_parser() (rna_tools.clanstix.rna_clanstix), 97
get_parser() (rna_tools.clanstix.rna_clanstix), 97
get_parser() (rna_tools.clanstix.rna_clanstix), 97
get_parser() (rna_tools.clanstix.rna_clanstix), 97
get_parser() (rna_tools.clanstix.rna_clanstix), 97
get_parser() (rna_tools.clanstix.rna_clanstix), 97
get_parser() (rna_tools.clanstix.rna_clanstix), 97