rna-pdb-tools documentation

Release 0.99

Marcin Magnus et al.

Dec 11, 2018
1 Getting Started: I want to . . . .......................................................... 3
   1.1 fetch a structure from the PDB database ........................................ 3
   1.2 fetch a biologically 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 add missing atoms ........................................................................... 6

2 rna-pdb-tools ...................................................................................... 11
   2.1 get RNAPuzzle ready ...................................................................... 11
   2.2 get sequence .................................................................................... 13
   2.3 fetch ................................................................................................. 14
   2.4 fetch Biological Assembly .................................................................. 14
   2.5 delete ................................................................................................. 15
   2.6 edit ..................................................................................................... 15
   2.7 the library .......................................................................................... 15

3 Utils ....................................................................................................... 23
   3.1 RNA Sequence .................................................................................. 23
   3.2 RNA Secondary Structure ................................................................. 25
   3.3 Blast PDB ........................................................................................... 27
   3.4 Blastn - select sequences from teh database matched by BLASTn ......... 27
   3.5 Rfam Search ........................................................................................ 27
   3.6 PDB Edit Bfactor/Occupancy ............................................................... 28
       3.6.1 Positional Arguments ................................................................. 29
       3.6.2 Named Arguments ..................................................................... 29
   3.7 RNA Alignment .................................................................................. 30
       3.7.1 RNASeq ...................................................................................... 31
       3.7.2 RNAalignment .......................................................................... 35
       3.7.3 Random assignment of nucleotides ............................................. 40
           3.7.3.1 Named Arguments ............................................................... 40
       3.7.4 CMAlign .................................................................................... 42
       3.7.5 RChie .......................................................................................... 43
       3.7.6 Renumber a pdb file according to alignment .................................... 44
9 Notebooks & rna-pdb-tools
  9.1 Share your notebooks ......................................................... 120
  9.2 Learn Jupyter ................................................................. 120

10 Emacs & rna-pdb-tools ...................................................... 121

11 Geekbook & rna-pdb-tools .................................................. 123
  11.1 Draw VARNA-based image of RNA secondary structure ............... 123

12 Git Quickref ........................................................................... 125
  12.1 Git sheet cheat ................................................................. 126
  12.2 Learn Git ........................................................................... 126
  12.3 Git GUI ............................................................................ 127

13 Install & Configure .............................................................. 129

14 Configuration .......................................................................... 131

15 Documentation ......................................................................... 133

16 Add a new util .......................................................................... 135

17 Indices and tables ................................................................... 137

Python Module Index .................................................................. 139
rna-pdb-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-pdb-tools).
If you find the tools helpful, you can cite the repo using the doi:


If something does not work for you, please make an issue, using https://github.com/mmagnus/rna-pdb-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
1y26
1fir

$ while read p; do rna_pdb_toolsx.py --fetch_ba $p; done < data/pdb_ids.txt
downloading...1y26_ba.pdb ok
downloading...1fir_ba.pdb ok

$ ls *.pdb
1fir_ba.pdb 1y26_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
GGAGUUCACCGCCACGCGGAGUACGAUCGAGGGUACAGUGAAUU
# 6TNA
> A:1-76
GCCGAUUUAgCUCAuUGGAGAGGcGCAGAcUgAUAGUCUGGUCuGCGaUCCACAGAAUUCGCACCA
# rp2_bujnicki_1_rpr
> A:1-15
CCGGAGGAACUACUG
> B:1-10
CCGGCAGCCU
> C:1-15
CCGGAGGAACUACUG
> D:1-10
CCGGCAGCCU
> E:1-15
CCGGAGGAACUACUG
> F:1-10
CCGGCAGCCU
> G:1-15
CCGGAGGAACUACUG
> H:1-10
CCGGCAGCCU
```

1.4 get secondary structures of your PDB files

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

Installation:

```bash
# 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-pdb-tools/rna_pdb_tools/utils/rna_x3dna folder.
Edit this line :
BINARY_PATH = <path to your x3dna-dssr file>
mapping 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:

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

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

```bash
[mm] py3dna$ git:(master) ./rna_x3dna.py test_data/*
test_data/1xjr.pdb
(continues on next page)
```
>1xjr nts=47 [1xjr] -- secondary structure derived by DSSR
gGAGUUCACCAGGCACGCGGAGUACGAUCGGGUACAGUGAAUU
..((((((.(((.))))))))))

>6TNA nts=76 [6TNA] -- secondary structure derived by DSSR
GCCGAUUAGUCAGGgUGGAGAAGcUaAGApUGGAGgUCcUGUgtPCGCaUCCACAGAUUUUCGCACCA
{(((((.(((((.))))))))))((((((((.)))))))))(((((.)))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))

test_data/rp2_bujnicki_1_rpr.pdb
>rp2_bujnicki_1_rpr nts=100 [rp2_bujnicki_1_rpr] -- secondary structure derived by DSSR
CCGGAGGAACUACUG&CCGGCAACCGCU&CCGGAGGAACUACUG&CCGGCAACCGCU&CCGGAGGAACUACUG&CCGGCAACCGCU&
CCGGAGGAACUACUG&CCGGCAACCGCU
{(((((.(((((.))))))))))&((((((.(((((.))))))))))))&((((((.(((((.))))))))))))&{((((((.(((((.))))))))))))&}
&((((((.(((((.))))))))))})

class rna_pdb_tools.utils.rna_x3dna.rna_x3dna.x3DNA(pdbfn)
        Attributes:
            curr_fn report
        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()
        exception rna_pdb_tools.utils.rna_x3dna.rna_x3dna.x3DNAMissingFile

1.5 delete a part of of your structure

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.

1.6 get numbering of your structure and rename chains

Rename chain B in structure 4_das_1_rpr.pdb:
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 add missing atoms

The tool is using the function:
Get mapuzzle (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_pdb_tools.rna_pdb_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.

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
2.1 get RNAPuzzle ready

class rna_pdb_tools.rna_pdb_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_pdb_tools.rna_pdb_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.
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

### 2.2 get sequence

Example:

```
$ rna_pdb_toolsx.py --get_seq 5_solution_1.pdb
> 5_solution_1.pdb A:1-576
CAUCCGGUAUCCCAAGACAAUCUCGGGUUGGGUUGGGAAGUAUCAUGGCUAAUCACCAUGAUGCAAUCGGGUUGAACACUUAAUUGGGUUAAAACGGUGGGGGACGAUCCCGUAACAUCCGUCCUAACGGCGACAGACUGCACGGCCCUGCCUCAGGUGUGUCCAAUGAACAGUCGUUCCGAAAGGAAG
```

```python
class rna_pdb_tools.rna_pdb_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=True)
  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
GGAGCUCGCC
> lykq_clx B:201-238
GGGCAGGCGCGUGCCAGCCUCUCGGAGCAGACUUACUCGC
```

(continues on next page)
Chains is


Chains are in other as the appear in the file.

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

### 2.3 fetch

Example:

```bash
$ rna_pdb_toolsx.py --fetch lxjr
downloading...lxjr ok
```

```python
rna_pdb_tools.rna_pdb_tools_lib.fetch(pdb_id, path='.')
```

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

### 2.4 fetch Biological Assembly

Example:

```bash
$ rna_pdb_toolsx.py --fetch_ba lxjr
downloading...lxjr_ba.pdb ok
```

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

```bash
$ cat data/pdb_ids.txt
1y26
1fir
$ while read p; do rna_pdb_toolsx.py --fetch_ba $p; done < data/pdb_ids.txt
downloading...1y26_ba.pdb ok
downloading...1fir_ba.pdb ok
```

```bash
$ ls *.pdb
1fir_ba.pdb 1y26_ba.pdb
```

```python
rna_pdb_tools.rna_pdb_tools_lib.fetch_ba(pdb_id, path='.')
```

fetch biological assembly pdb file from RCSB.org

```python
>>> fetch_ba('lxjr')
...
```
2.5 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`.

2.6 edit

```python
rna_pdb_tools.rna_pdb_tools_lib.edit_pdb(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
```

2.7 the library

```python
rna_pdb_tools_lib.py - main lib file, many tools in this lib is using this file.
```

exception rna_pdb_tools.rna_pdb_tools_lib.PDBFetchError

class rna_pdb_tools.rna_pdb_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

```python
edit_occupancy_of_pdb(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

```python
fix_O_in_UC()
```

```python
fix_op_atoms()
```

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

```python
fix_with_qrnas(outfn=", verbose=False)
```

Add missing heavy atom.

A residue is recognized base 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 as an integer

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_pdb_tools.rna_pdb_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.
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=True)
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
GGAGCUCGCC
> lykq_clx B:201-238
GGGCGAGGCCGUGCCAGCUCUUCGGAGCAAUACUCGCC
> 6_solution_0 A:1-19 26-113 117-172
GGCGGCGGCCGUCUCCCGACGUCGGGAGUUAAAAGGGAAG
```

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 False

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(verbos)
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

rtype object

remove_water()
Remove HOH and TIP3

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 –

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:

# URI -> U, URA -> U
1xjr_clx_charmm.pdb:ATOM 101 P URA A 5 58.180 39.153
→ 30.336 1.00 70.94
rp13_Dokholyan_1_URI_CYT_ADE_GUA_hydrogens.pdb:ATOM 82 P URI A
→ 4 501.633 506.561 506.256 1.00 0.00 P

un_nmr(verbos=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)
collapsed_view

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

Example:

```
[mm] rna_pdb_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
```

edit_pdb

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:

```
$ 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
```

fetch

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

fetch_ba

fetch biological assembly pdb file from RCSB.org

```
>>> fetch('1xjr')
...```

fetch_cif_ba

fetch biological assembly cif file from RCSB.org

get_version

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

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

Parameters
- \texttt{struc\_fn} (str) – path to the main PDB file
- \texttt{insert\_fn} (str) – path to the file that will be injected into the main PDB file
- \texttt{chain\_id} (str) – chain that will be inserted into the main PDB file

**Returns** text in the PDB format

**Return type** string
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.

Installation:


ViennaRNA (https://www.tbi.univie.ac.at/RNA/)

ipknot OSX (https://github.com/satoken/homebrew-rnatools)

RNAStructure (http://rna.urmc.rochester.edu/)

Works with 5.8.1; Jun 16, 2016.

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 DATAPATH 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.

FAQ:

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

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

```python
from rna_pdb_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
        print s.predict_ss(method="contextfold"),
        # print s.predict_ss(method="centroid_fold")
```

@todo should be renamed to RNASeq, and merged with RNASeq class from RNAalignment.

```python
exception rna_pdb_tools.Seq.MethodNotChosen
class rna_pdb_tools.Seq.RNASequence(seq)
    RNASequence.

Usage:

```python
>>> seq = RNASequence("CCCCUUUUGGGG")
>>> seq.name = 'RNA03'
>>> print(seq.predict_ss("RNAfold", constraints="((((....))))"))
>RNA03
CCCCUUUUGGGG
((((....)))) ( -6.40)
```

**eval** *(no_dangling_end_energies=True, verbose=False)*

Evaluate energy of RNA sequence.

**Args:** no_dangling_end_energies (Boolean) verbose (Boolean)

**Returns:** Energy (flaot)

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](http://rna.tbi.univie.ac.at/cgi-bin/RNAWebSuite/RNAeval.cgi)

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

Predict secondary structure of the seq.

**Parameters**

- method
- constraints
- shapefn – path to a file with shape reactivites
• **verbose** –

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

**ContextFold:**

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

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

```bash
$ java -cp bin contextFold.app.Predict in:UUUUUUGGG
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10

# this is OK
$ java -cp bin contextFold.app.Predict in:CCCCUUUGGG
CCCUUUUGGGG
.(((.)))
```

**RNAstructure:**

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

and with the shape data:

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

the shape data:

```
1 10
2 1
3 1
```

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

### 3.2 RNA Secondary Structure

Secondary structure analysis

```python
>>> exception rna_pdb_tools.SecondaryStructure.ExceptionOpenPairsProblem
rna_pdb_tools.SecondaryStructure.draw_ss(title, seq, ss, img_out, resolution=4, verbose=False)
```

Draw Secondary Structure using VARNA (you need correct configuration for this).

If everything is OK, return None, if an error (=exception) return stderr.

**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
```

![Diagram of RNA structure](image)

**rnna**


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

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

**Parameters**

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

3.3 Blast PDB

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

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

Usage:

```python
>>> p = BlastPDB(
  'GGGUCAGGCCGGCGAAAGUCGCCACAGUUUGGGGAAAGCUGUGCAGCCUGUAACCCCCCCACGAAAGUGGG')
>>> p.search()
```  

Parameters  

- **seq** - string

search ()
Search online the seq.

3.4 Blastn - select sequences from teh database matched by BLASTn

A super-simple wrapper to parse by headers a BLASTn output (outfmt - 6) sequences in fasta from the database of sequences.

3.5 Rfam Search

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

class rna_pdb_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/](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-pdb-tools.


**cmscan** *(seq)*  
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_pdb_tools.RfamSearch.RfamSearchError

### 3.6 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\`
```

(continues on next page)
usage: rna_pdb_edit_occupancy_bfactor.py [-h] (--bfactor | --occupancy) 
[--select SELECT] [--set-to SET_TO] 
[--set-not-selected-to SET_NOT_SELECTED_TO] [-o OUTPUT] [--verbose] 
[--select-atoms SELECT_ATOMS] 
file

3.6.1 Positional Arguments

file

3.6.2 Named Arguments

--bfactor set bfactor
Default: False

--occupancy set occupancy
Default: False

--select get chain, e.g A:1-10, works also for multiple chains e.g. A:1-40,B:1-22

--set-to set value to, default is 1
Default: 1

--set-not-selected-to set value to, default is 0
Default: 0

-o, --output file output

--verbose be verbose
Default: False

--select-atoms select only given atomscan be only one atom, e.g. P+C4'

rna_pdb_tools.utils.rna_pdb_edit_occupancy_bfactor.rna_pdb_edit_occupancy_bfactor.edit_occupancy_of_pdb

3.6. PDB Edit Bfactor/Occupancy 29
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 sel_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

### 3.7 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/magnus/rna-pdb-tools/blob/master/rna_pdb_tools/utils/rna_alignment/rna_alignment.ipynb)

Load an alignment in the Stockholm 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]
```
3.7.1 RNASeq

class rna_pdb_tools.utils.rna_alignment.rna_alignment.RNASeq(id, seq, ss=None)
RNASeq.

Parameters

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

seq_no_gaps
str = seq.replace(‘-‘, ‘’)

ss_no_gaps
str = ss.replace(‘-‘, ‘’)

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 UCGGGGUGCCCUUCUGCGUG—AAGGC—UGAGAAAUACCCGU—AUCACCUG—AUCUGGAU—AAUGC
XXXXXXXXXXXXGXGXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX—XXXXXXXX
XXXXXXXX—ACXUG

3.7. RNA Alignment
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 (by default is True) then GU pair is also possible.
If you provide seq and secondary structure such as:

```
GgCcGGgg.GcggG.cc.u.AUACAAuACCC.GaAA.GGGAAUAaggCc.gGCc.gu........CU.......         
→uugugcgGUuUcaAgCccCCgCcCaCCcuuuu
(((((((.....(.(..................))))...((.((..............

``` 

...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)

### 3.7.2 RNAAlignment

**class**  rna_pdb_tools.utils.rna_alignment.rna_alignment.RNAAlignment (**fn=",**, **fetch="")

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

Usage (for more see IPython notebook https://github.com/mmagnus/rna-pdb-tools/blob/master/rna_pdb_tools/utils/rna_alignment/rna_alignment.ipynb)

```python
>>> a = RNAAlignment('test_data/RF00167.stockholm.sto')
>>> print(a.tail())
```

```bash
>>> 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:
and on the format itself

**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 alignemnt, `-GG.AGAGUA-GAUGAUUCGCGUUA

**Return type**
- str

### copy_ss_cons_to_all (verbose=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.
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 = "ggaucgcugaacccgaaaggggcgggggacccagaaauggggcgaaucucuuccgaaaggaagaguaggguuacuccuucgacccgagcccgucagcuaaccucgcaagcguccgaaggagaauc"

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

Match: AL939120.1/174742-174619
ID: AL939120.1/174742-174619
Name: AL939120.1
Description: AL939120.1/174742-174619
Number of features: 0
/start=174742
/end=174619
/accession=AL939120.1

Per letter annotation for: secondary_structure
Seq('CCAGGUAAGUCCG--G--ACCG---------------GUCA-----------...GGA',
SingleLetterAlphabet())

find_seq_exact (seq, verbose=False)
Find seq (also subsequences) and reverse in the alignment.

Parameters

- seq – string, 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.

blasting: 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_shift_seq_in_align()
    RF_cons vs ‘#=GC RF’ ???

get_ss_cons()
    Returns SS_cons_pk line or None if there is no SS_cons_pk.

get_ss_cons_pk()
    Returns SS_cons_pk line or None if there is no SS_cons_pk:

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

UUAU-AACAUUAUAUUAUGACAAUAUGG-GUCAUAA-GUUUCUACC-GUAAUAAUUCU—GACUAUGA

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’
• **resis** – list resis, [5,6]

map:

```
[4, 5, 6]
UAU-A
UAU-AA
UAU-AAC
[5, 6, 7]
CAC-U
CAC-U-
CAC-U-U
[4, None, 5]
```

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
---ACCUUGCCGACUUGCCGAUCC-------------------...AAU CP001644.1/756294-756165
--GCUUCUGCACUGGGCGACUUG-------------------...GAA CU234118.1/352539-352459
UGAGUUUUCUGCUGACGGCGAUAU-------------------...CUG BAAV01000055.1/2897-2982
GCCCUGCUGACUGCCCUGCGCGAUG-------------------...CGA CP000927.1/5164264-5164343
------GGGUGCAGGCGCGAACA-------------------...--- zmp
UCACCCCCUGGCUGACUGCGCGAUA-------------------...GUU AP009385.1/718103-718202
>>> a.remove_empty_columns()
>>> print(a)
SingleLetterAlphabet() alignment with 6 rows and 138 columns
---ACCUUGCCGACUUGCCGAUCC-UGAAGCUGCUUUG-AGCG...AAU CP001644.1/756294-756165
--GCUUCUGCACUGGGCGACUUG-------------------...GAA CU234118.1/352539-352459
UGAGUUUUCUGCUGACGGCGAUAU-------------------...CUG BAAV01000055.1/2897-2982
GCCCUGCUGACUGCCCUGCGCGAUG-------------------...CGA CP000927.1/5164264-5164343
------GGGUGCAGGCGCGAACA--------G-----------...--- zmp
UCACCCCCUGGCUGACUGCGCGAUA--------GAACCCUCGGGUU...GUU AP009385.1/718103-718202
```

Go over all seq modifies self.nss_cons

**ss_cons_std**

**ss_cons_with_pk**

Go over ss_cons and overwrite bp is there is pk (ss_cons_pk)

```
ss_cons: (((..(((...........(((((((((((((((((((((((((................))))))))))))))))))))))))))))))))))
..............[[..............................]]]........................
ss_cons_with_pk: (((..(((...........(((((((((((((((((((((((((................))))))))))))))))))))))))))))))))))))))
```

"return ss_cons_with_pk: string, e.g. (((..(((...........[..........])))))))

**ss_cons_with_pk_std**

subset *(ids, verbose=False)*

Get subset for ids:

```
# STOCKHOLM 1.0
#@GF WK Tetrahydrofolate_riboswitch
```
AAQK01002704.1/947-1059 -U-GC-AAAAUAGGUUCCAUGC.. #=GC SS_cons .(.().(—-((((((((((. . . .
#=GC RF .g.gc.aGAGUAGggugccgugc.. //

tail()
trimmed_rf_and_ss()
    Remove from RF and SS gaps.

    Returns  trf, tss - new RF and SS

    Return type  (str,str)

write(fn, verbose=False)
    Write the alignment to a file

### 3.7.3 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 sequences (arg –seqfn)

<table>
<thead>
<tr>
<th>R = G A (purine)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y = U C (pyrimidine)</td>
</tr>
<tr>
<td>K = G U (keto)</td>
</tr>
<tr>
<td>M = A C (amino)</td>
</tr>
<tr>
<td>S = G C (strong bonds)</td>
</tr>
<tr>
<td>W = A U (weak bonds)</td>
</tr>
<tr>
<td>B = G U C (all but A)</td>
</tr>
<tr>
<td>D = G A U (all but C)</td>
</tr>
<tr>
<td>H = A C U (all but G)</td>
</tr>
<tr>
<td>V = G C A (all but T)</td>
</tr>
<tr>
<td>N = A G C U (any)</td>
</tr>
</tbody>
</table>

author: A. Zyla - azyla

**Warning:** Tested only on fasta files! and requires Biopython (tested with v1.68)


#### 3.7.3.1 Named Arguments

- **-v, --verbose**  
  increase output verbosity  
  Default: False

- **--alignfn**  
  alignment in the Fasta format

- **--seqfn**  
  sequences in the Fasta format

- **--outfn**  
  output aln file (default: alnfn .fasta -> _out.fasta)
** Warning: ** Tested only on fasta files! and requires Biopython (tested with v1.68)

```python
get_align(alignfn)
```

Get seq from an alignment with gaps.

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

**Usage:**

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

**Returns:** alignment

```python
get_sequences(seqfn)
```

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

**Usage:**

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

**Returns**

```python
[obj1', description='obj1', id='<unknown id>', name='<unknown name>', description='<unknown description>', dbxrefs=[], ...]
```

```python
write_align(align, outfn)
```


**Returns** writes to a file in fasta format

**Return type** none
**write_seq**

Write cleaned alignment with Biopython.

```
write_seq(align, outfn)
```

- **param align**: a cleaned alignment
- **type align**: obj
- **param outfn**: a path to a new alignment file
- **type outfn**: str

**Returns**
writes to a file in fasta format

**Return type**
one

### 3.7.4 CMAAlign

```
class CMAAlign(outputfn=None)
```

CMAAlign class around cmalign (of Infernal).

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.CMAlign()
    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
```

```bash
    cmd cmalign -g RF00167.cm ade_seq.fa
```

```
# STOCKHOLM 1.0
#=GF AU Infernal 1.1.2
ade ----------------CGCUUCAUAUACCUCUUAGAUUGGAGUUGGAACAGAUGAG-
  ->CCUUAAA-CUCUUGAUUGGAGAUGAAGUGGGAUGGGAUGEUGGAGUGAGAGAGA-
  #=GR ade PP ................99*********************************************.
  ->***********.***************999............
  #=GC SS_cons :::::::::::::::::((((((((,,,<<<<<<<_______>>>>>>>,,,,,,,,<<<<<<<_____
  __>>>>>>>,,))))))))::::::::::::::
  #=GC RF
  aaaaaaaaaaaaaaaaaaaacccccCguUAAuuccgggAAAUUGGcgggaGUUUUCACGagcaqCGUAAAcugccuGACUAcGagggaaauuuuu
  //
  cma hit ----------------CGCUUCAUAUACCUCUUAGAUUGGAGUUGGAACAGAUGAG-
  ->CUCUUGAUUAUGGAGAUGAAGGGAUGGGAUGEUGGAGUGAGAGAGA-
  seq ------------------------CGCU-U-CAUAUAUACCUCUUAGAUUGGAGUUGGAACAGAUGAG-
  ->UUAAA-CUCU---GUUUUGGGA-AUGGGAUGGGAUGEUGGAGUGAGAGAGA-
  a.rf aaaaaaaauuuuuuuuuuuu.c.u.CguUAAuuccgggAAAUUGG.cggga.GUUUUCACGagcaqCGUAAAacugccuGACUAcGagggaaauuu.
```

Install http://eddylab.org/infernal/


**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 41vv.seq
# STOCKHOLM 1.0
# =GF AU Infernal 1.1.2
41vv
=GGAGAGUA-GAUGAUUCGCUUAGUGUGUGUGA-AUGGGAUGCG-UCACACAACGAAGC---
=GAGA---GCGCUGAUAUCAUG-CACUCCGUCUCA
# =GR 41vv PP .********.******************99999998.***********.8999999******8...
=5555...8********.*************
# =GC SS_cons ((((((((.,.,,,<<<<<<<<<<<<<>>>>)>>>>>>>,<<<<_-
________________>)__________))))))----------))
# =GC RF
---ggcaGAGUAAGggugccgugcGUuAAGUGccggcggAcGGAUGccgcggAcGGAAGggcaaaauugcccGCGgacgggaccCGCAUGGCCCCG

Warning: requires cmalign to be set in your shell

3.7.5 RChie

class rna_pdb_tools.utils.rna_alignment.rna_alignment.RChie

RChie - plotting arc diagrams of RNA secondary structures.

www.e-rna.org
http://www.e-rna.org/r-chie/
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_pdb_tools/rpt_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 --msafile 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
- `ss_cons` – a string of secondary structure consensus, use only ( ). Works with pseudo-knots.

- `show()`
- `write(outfn)`

### 3.7.6 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
```

### 3.7.6.1 Positional Arguments

- `seqid` seq id in the alignemnt
**alignfn**  
alignemnt in the Fasta format

**pdbfn**  
pdb file

### 3.7.6.2 Named Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| -v, --verbose | increase output verbosity  
Default: False |
| --residue_index_start | renumber starting number (default: 1)  
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_pdb_tools.utils.renum_pdb_to_aln.renum_pdb_to_aln.get_parser()
rna_pdb_tools.utils.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_pdb_tools.utils.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_pdb_tools.utils.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_pdb_tools.utils.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

### 3.8 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.)

**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](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.

#### 3.8.1 Color groups

You can color your groups:
To get colors, run a cmd like this:

```
rna_clastix.py rnapz17_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)

### 3.8.2 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_run.txt
```

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 1: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 | tee 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.


```python
class rna_pdb_tools.utils.clanstix.rna_clanstix.RNAStructClans(n=10)
    Clans run.

    Usage:
    >>> f = open('matrix.txt')
    >>> ids = f.readline().replace('#','').split()
    >>> c = RNAStructClans(n=len(ids)) # 2007
    >>> c.add_ids(ids)
    >>> c.dist_from_matrix(f)
    >>> print(c.txt)

    add_ids(ids)
    dist_from_matrix(lines, matrix=0, use_pv=False)
```

3.9 Calculate Root Mean Square Deviation (RMSD)

3.9.1 rna_calc_rmsd

`rna_calc_rmsd.py` - calculate RMSDs of structures to the target

If you still have problems 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:

```
$ rna_calc_rmsd.py -t 6_0_solution_4GXY_rpr.pdb --model_selection=A:1-17+24-110+115-
\rightarrow 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
6_Blanchet_5_rpr.pdb 24.17 3409
6_Blanchet_6_rpr.pdb 23.28 3409
6_Blanchet_7_rpr.pdb 22.26 3409
6_Bujnicki_1_rpr.pdb 36.95 3409
6_Bujnicki_2_rpr.pdb 30.9 3409
6_Bujnicki_3_rpr.pdb 32.1 3409
6_Bujnicki_4_rpr.pdb 32.04 3409
...
```

Example #2:

```
time rna_calc_rmsd.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_rmsd_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]
   [--model_selection MODEL_SELECTION]
   [--model_ignore_selection MODEL_IGNORE_SELECTION]
   [-m METHOD] [-o RMSDS_FN] [-v]

(continues on next page)
3.9.1.1 Positional Arguments

files

3.9.1.2 Named Arguments

- `-t`, `--target_fn`  
  pdb file  
  Default: ""

- `--target_selection`  
  selection, e.g. A:10-16+20, where #16 residue is included  
  Default: ""

- `--target_ignore_selection`  
  A/10/O2’  
  Default: ""

- `--model_selection`  
  selection, e.g. A:10-16+20, where #16 residue is included  
  Default: ""

- `--model_ignore_selection`  
  A/10/O2’  
  Default: ""

- `-m`, `--method`  
  align, fit  
  Default: “all-atom-built-in”

- `-o`, `--rmsds_fn`  
  output, matrix  
  Default: “rmsds.csv”

- `-v`, `--verbose`  
  verbose  
  Default: False

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
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
6_Blanchet_5_rpr.pdb 24.17 3409
```

(continues on next page)
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

**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/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
```

**Example #2:**

```bash
# 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
```

### 3.9. Calculate Root Mean Square Deviation (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

#### `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/
```

```
rna_pdb_tools.utils.rna_calc_rmsd.rna_calc_rmsd.get_parser()
rna_pdb_tools.utils.rna_calc_rmsd.rna_calc_rmsd.get_rna_models_from_dir(files)
```

**Parameters**

- `models` – a list of filenames

Example of the list:

```
['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']
```

```
rna_pdb_tools.utils.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/

### 3.9.2 rna_calc_evo_rmsd

### 3.9.3 rna_calc_rmsd_trafl

```
rmsd_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.pdb_rmsd_e
< 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
/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 17_Das_2_rpr.pdb_rmsd_e_17_Das_2_rpr_n0_rmsd_e
< rmsd_out: 17_Das_2_rpr.pdb_rmsd_e_17_Das_2_rpr_n0_rmsd_e
> struc: 5k7c_clean_onechain_renumber_as_puzzle_rpr_rmGapped_n0.trafl 2
> trafl: 17_Das_2_rpr.pdb.trafl 48
% saved: 17_Das_2_rpr.pdb.trafl_5k7c_clean_onechain_renumber_as_puzzle_rpr_rmGapped_n0.trafl
> 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_5k7c_clean_onechain_renumber_as_puzzle_rpr_rmGapped_n0.trafl 17_Das_2_rpr.pdb_rmsd_e_5k7c_clean_onechain_renumber_as_puzzle_rpr_rmGapped_n0_rmsd_e
```

(continues on next page)
< rmsd_out: 17_Das_2_rpr.pdb_rmsd_e_5k7c_clean_onechain_renumber_as_puzzle_rpr_remgapped_n0_rmsd_e
0.000 -695.634
0.000 -551.093
< rmsd_out: rp17_rmsd.txt

---

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] trafl struc1 struc2 rmsds_fn

3.9.3.1 Positional Arguments

- trafl: trafl
- struc1: structure A
- struc2: structure B
- rmsds_fn: output file

3.9.4 rna_calc_rmsd_all_vs_all

rna_calc_rmsd_all_vs_all.py - calculate RMSDs all vs all.

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)

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

3.9.4.1 Named Arguments

- -i, --input_dir: input folder with structures
  Default: ""
- -o, --matrix_fn: output, matrix
  Default: "matrix.txt"

rna_calc_rmsd_all_vs_all.py - calculate RMSDs all vs all.

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

rna_pdb_tools.utils.rna_calc_rmsd.rna_calc_rmsd_all_vs_all.calc_rmsd(a, b)
Calc rmsd.

rna_pdb_tools.utils.rna_calc_rmsd.rna_calc_rmsd_all_vs_all.get_parser()

rna_pdb_tools.utils.rna_calc_rmsd.rna_calc_rmsd_all_vs_all.get_rna_models_from_dir(directory)
Sort the given list in the way that humans expect. http://blog.codinghorror.com/sorting-for-humans-natural-sort-order/

3.10 Calculate Interaction Network Fidelity (INF) and not only

3.10.1 rna_calc_inf


3.10.1.1 Positional Arguments

files
files, e.g folder_with_pdb/#pdb

3.10.1.2 Named Arguments

-t, --target_fn pdb file
Default: “”

-m, --number_of_threads number of threads used for multiprocessing, if 1 then mp is not used (useful for debugging)
Default: 8

-s, --ss A:([[[ ]]])
Default: “”

-f, --force force to run ClaRNA even if <pdb>.outCR file is there
Default: False

-v, --verbose be verbose, tell me more what’re doing
Default: False

-o, --out_fn out csv file, be default inf.csv
Default: “inf.csv”
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` 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_pdb_tools.utils.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_pdb_tools.utils.rna_calc_inf.rna_calc_inf.get_parser()
```

### 3.10.2 rna_calc_dinf

Obtain a list of interaction in an RNA molecule where “Interaction” is purely distance based (defined by –cutoff).

Later, you can use it to calculate distance based INF, dINF ;-).

Example:

```
[rmm] rna_calc_inf$ 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:

```
[rmm] rna_calc_inf$ ./rna_calc_dinf.py test_output/1Y26.pdb > 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:

```
draw_dists([(13, 14), (13, 83), ... (82, 83)])
```
so you can plot all interacting bases:

Mind, that draw_dist 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).

3.10.2.1 Positional Arguments

file            a PDB file

3.10.2.2 Named Arguments

-d, --draw-dists  Default: False
-c, --cut-off     Default: 5
-v, --verbose     be verbose
                   Default: False

3.11 Measure distance between atoms

3.12 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/utils/diffpdb/diffpdb_conf.py` (create it if needed)

```
./diffpdb.py --names test_data/4/1duq.pdb test_data/4/1duq_decoy0171_amb_clx.pdb
```
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.

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/utils/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.

```
usage: rna_helix_vis [-h] [--names] [--names_and_resi] [--htmlout]
               [--method METHOD] f1 f2
```

### 3.12.1 Positional Arguments

- `f1` file
- `f2` file

### 3.12.2 Named Arguments

- `--names` take only atom residues names
  - Default: False
- `--names_and_resi` take only atom residues names
  - Default: False
- `--htmlout` take only atom residues names
  - Default: False
- `--method` method e.g. `diff`

### 3.13 RNA filter

#### 3.13.1 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:

$ 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

# $ 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)

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

3.13.1.1 Named Arguments

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

3.13.2 rna_dca_mapping.py

3.13.3 show_dists - show distances in PyMOL

show_dists - show distances in PyMOL

3.13. RNA filter
Usage:

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

### 3.13.4 `rna_ex2x.py` - analyze an evolutionary coupling file.

`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

```
--pairs:
```

```
$ 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],[15, 81],[34, 63],[56, 70],[58, 68],[35, 63],[26, 45],[35, 64],[32, 39],[54, 73],[24, 74],[16, 82],[24, 45],[24, 43],[32, 36],[25, 48],[48, 82],[36, 48],
```


#### 3.13.4.1 Positional Arguments

- `interaction_fn` interaction file
3.13.4.2 Named Arguments

- **--sep** separator
  Default: “,”
- **--chain** chain
  Default: “A”
- **--ec-pairs** Default: False
- **--ss-pairs** file with secondary structure base pairs
- **--pairs-delta** delta: ec-bp - ss-paris
  Default: False

3.13.5 rna_pairs2SimRNArestrs.py - convert pairs to SimRNA restraints

**Example:**

```
$ 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:** rna_pairs2SimRNArestrs.py [-h] [-o OFFSET] [-w WEIGHT] [-v]

3.13.5.1 Positional Arguments

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

3.13.5.2 Named Arguments

- **--offset** can be -10
  Default: 0
- **--weight** weight
  Default: 3
- **-v, --verbose** be verbose
  Default: False

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

**Example:**

```
rna_ss_get_bps.py -i input_file.faa -o output_file.bps
```

3.13. RNA filter
Usage:

```bash
$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: `rna_ss_get_bps [-h] [--offset OFFSET] [-v] file`

### 3.13.6.1 Positional Arguments

- `file` file in the Fasta format

### 3.13.6.2 Named Arguments

- `--offset` offset
- `-v, --verbose` be verbose

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

**Usage:**

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

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

### 3.13.7.1 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)

### 3.13.7.2 Named Arguments

- `-v, --verbose` be verbose
3.14 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:

```plaintext
> a
....((..((....)))..(((.(........)...))).)).((((((......))))

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

to:

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

and:

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

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

to:

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

and:

```plaintext
> b
..((....(((....))).(((.(........)..)))))).(((((......))))

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

to:

```plaintext
> b
..((.[.[.[([[[[]]]]].))).(((.(.[.{[[[.]]]].))).)).

and it works with VARNA:
```
Convert a secondary structure with a pk to the SimRNA format:

```
python rna_ss_pk_to_simrna.py test_data/ss_with_pk.ss
```

The code:

```python
rna_pdb_tools.utils.rna_convert_pseudoknot Formats.rna_ss_pk_to_simrna.get_multiple_lines(ss)
rna_pdb_tools.utils.rna_convert_pseudoknot Formats.rna_ss_pk_to_simrna.get_parser()
rna_pdb_tools.utils.rna_convert_pseudoknot Formats.rna_ss_pk_to_simrna.is_pk(ss)
```

### 3.15 x3DNA (contacts classification & secondary structure detection)

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-pdb-tools/rna_pdb_tools/utils/rna_x3dna folder.
Edit this line:
BINARY_PATH = <path to your x3dna-dssr file>
mapping 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
GAGGUUCACCGAGCCACCGGAAGUACGAGGGUAGUGAAUU
..(((((((...((((((......))))..)).))).).))))
```

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

```
[mm] py3dna$ git:(master) ./rna_x3dna.py test_data/ *
```
```
test_data/1xjr.pdb
>1xjr nts=47 [1xjr] -- secondary structure derived by DSSR
GAGGUUCACCGAGCCACCGGAAGUACGAGGGUAGUGAAUU
..(((((((...((((((......))))..)).))).).))))
```
```
test_data/6TNA.pdb
>6TNA nts=76 [6TNA] -- secondary structure derived by DSSR
GCGGAUUUAAGCUCAGuGGAGAAGAgCqCAACuUqAPcUGGAGuUCcUGuGtPCGCaUCCACAGAUUCGCACCA
(((((((...((((((......))))..)).))).).))))
```
```
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&
CCGGAGGAACUACUG&CCGGCAGCCU
[[][[[[[[[[[[[[[[[&]]]]]]]]]]]]]]]]]]]]]]]]]]
```

```py
rna_pdb_tools.utils.rna_x3dna.rna_x3dna.get_parser()
```

```py
class rna_pdb_tools.utils.rna_x3dna.rna_x3dna.x3DNA(pdbfn)
```

```py
curr_fn
```
```
report
```

```py
clean_up (verbose=False)
```

```py
get_ion_water_report ()
```

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

```py
get_modifications ()
```

```
Run find_pair to find modifications.
```

```py
get_secstruc ()
```
```
Get secondary structure.
```

```py
get_seq ()
```
```
Get sequence.
```
```
Somehow 1bzt_1 x3dna UCAGACUUUUAAPCUGA, what is P? P -> u
```

```py
run_x3dna ()
```

```
exception rna_pdb_tools.utils.rna_x3dna.rna_x3dna.rna_x3dna.x3DNAMissingFile
```

### 3.16 ClaRNA (contacts classification)

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

```bash
./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_pdb_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))

**Usage:**

```python
rna_pdb_tools.utils.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
```

Example of the list:

```python
5k7c_clean_onechain_renumber_as_puzzle_srr.pdb  pistol_thrs0.50A_clust01-
→000001_AA.pdb
0.642 NA 0.874 0.000 0.944 0.810 0.000 0.
→000s
```

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

```python
rna_pdb_tools.utils.clarna_app.clarna_app.clarna_run(fn, force=True)
```

Run ClaRNA run

**fn:** str filename to analyze

**Returns** a filename to ClaRNA output (fn + `.outCR`)
rna_pdb_tools.utils.clarna_app.clarna_app.get_ClaRNA_output_from_dot_bracket(ss, temp=True, verbose=False)

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

Parameters ss (string) – secondary structure

Returns a filename to ClaRNA output

rna_pdb_tools.utils.clarna_app.clarna_app.get_dot_bracket_from_ClaRNA_output(inCR, verbose=False)

In inCR file

rna_pdb_tools.utils.clarna_app.clarna_app.get_parser()

### 3.17 SimRNA

#### 3.17.1 Select low energy frames

rna_simrna_lowest.py

Select lowest energy frames out of a SimRNA trajectory file. This code uses heavily the SimRNATrajectory class. By default 100 lowest energy frames is exported.

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

#### 3.17.1.1 Positional Arguments

- **trafl** 
  
  SimRNA trafl file

#### 3.17.1.2 Named Arguments

- **-n, --nstruc** 
  
  SimRNA trafl file
  
  Default: 100

#### 3.17.2 Extract

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

Options:

SIMRNA_DATA_PATH has to be properly defined in rpt_config_local.

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

3.17. SimRNA
### 3.17.2.1 Named Arguments

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

### 3.18 SimRNAweb

#### 3.18.1 Download files of a SimRNAweb run

### 3.19 SimRNATrajectory

SimRNATrajectory module.

SimRNATrajectory / Frame / Residue / Atom

```python
class rna_pdb_tools.utils.simrna_trajectory.simrna_trajectory.Atom(name, x, y, z)
x y z coord
get_coord()
    Return coords (np.array).

class rna_pdb_tools.utils.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_trafl.py to fix the trajectory first.

```python
class rna_pdb_tools.utils.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

```python
get_atoms()
Return all atoms
```

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

class rna_pdb_tools.utils.simrna_trajectory.simrna_trajectory.SimRNATrajectory

load_from_file (fn, debug_break=False, top_level=False, only_first_frame=False)
Create a trajectory based on give 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 trajectory file into memory, and get stuck. Use this if you want to compute e.g. distances between atoms, get the positions of specified atoms etc. If you can not process your trajectory use top_level=True or look at load_from_string() to load a frame by frame from a file.

h(eader), l(line), f(ile)

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 \( (fn, \text{verbose}=True) \)
Save the trajectory to file.

sort ()
Sort frames within the trajectory according to energy.

3.20 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.

code

\begin{verbatim}
rna_pdb_tools.utils.rnakb_utils.rnakb_utils.fix_gromacs_gro(path, \text{verbose}=False)
\end{verbatim}

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

code

\begin{verbatim}
rna_pdb_tools.utils.rnakb_utils.rnakb_utils.fix_gromacs_ndx(path)
\end{verbatim}

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\(\ast\) –

Output:

- index is overwritten with a corrected one

```python
rna_pdb_tools.utils.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

```python
rna_pdb_tools.utils.rnakb_utils.rnakb_utils.get_res_code(line)
```

Get residue code from a line of a PDB file

```python
rna_pdb_tools.utils.rnakb_utils.rnakb_utils.get_res_num(line)
```

Extract residue number from a line of PDB file

Parameters **line** = ATOM line from a PDB file\(\ast\) –

Output:

- residue number as an integer

```python
rna_pdb_tools.utils.rnakb_utils.rnakb_utils.make_rna_gromacs_ready(pdb_string, verbose=False)
```

GROMACS has some special requirements for PDB files.

Parameters **pdb_string** = contents of PDB file as a string\(\ast\) –

Output:

- new PDB returned as a string

```python
(!!!) # hmm... [ RA5 ] will not be detected based on it (!?) Hmm.. because it detects if the structure is already prepared.
```

```python
rna_pdb_tools.utils.rnakb_utils.rnakb_utils.make_rna_rnakb_ready(pdb_string, verbose=False)
```

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.

Parameters **pdb_string** = contents of PDB file as a string\(\ast\) –

Output:

- new PDB returned as a string

```python
rna_pdb_tools.utils.rnakb_utils.rnakb_utils.prepare_groups(fn, gr_fn, potential='aa', verbose=False)
```

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

Get seq and uniq & sort it. ['RG5', 'RA', 'RA', 'RA', 'RG', 'RU', 'RA', 'RA', 'RC3']

set(['RU', 'RG', 'RC3', 'RG5', 'RA'])

@todo RG5 etc – done!

gtxt:
3.21 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_clust01-000001_AA.pdb.
RNA Refinement (QRNAS)

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 just:
- QRNA -c <configfile>

**Installation of this util**

Set up in your bashrc:

```bash
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 queueing system (this one with qsub ;-)):

```bash
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`

### 3.2.1 Positional Arguments

| fn       | input pdb file |

### 3.2.2 Named Arguments

| -s, --steps | # of steps, default: 20k |
| Default: 20000 |
| -o, --output_file | output pdb file |
3.22 ROSETTA


3.22.1 Run (modeling)

`run_rosetta` - wrapper to ROSETTA tools for RNA modeling


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'
> | sed 's#<[^>]*>##g' | grep " | column -t
```

(https://stackoverflow.com/questions/26104116/qstat-and-long-job-names) so there is now need to shorted my job ids.

Run:

```
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!
Sequence: GUGGG CCCAU
Secstruc: ((((())))))
Writing to fasta file: helix2.fasta  # next helix #2

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 charactes.

[--sandbox SANDBOX]
file

3.22.1.1 Positional Arguments

file file: > a04
UAUAACAUAAAUUUUGACAAUAUGGGUCAAUAGUUUCUACCUGAAUACCUGUAAAUUUCA
(((.(((....(((.((((....)))....))))....)))))))

3.22.1.2 Named Arguments

-i, --init
prepare _folder with seq and ss
Default: False
-e, --helices
produce h(E)lices
Default: False
-r, --rosetta
prepare rosetta files (still you need go to send jobs to a cluster)
Default: False
-g, --go
send jobs to a cluster(run qsubMINI)
Default: False
-n, --nstruc
# of structures you want to get
Default: 10000
-c, --cpus
# of cpus to be used
Default: 200
--sandbox
where to run it (default: RNA_ROSETTA_RUN_ROOT_DIR_MODELING
Default: “/home/magnus/rosetta-runs”
### 3.22.2 Get a number of structures

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

**Example:**

```bash
$ rna_roseta_n.py ade.out
21594
```

**usage:** rna_roseta_n.py [-h] [-v] file

#### 3.22.2.1 Positional Arguments

- **file**
  - ade.out

#### 3.22.2.2 Named Arguments

- **-v, --verbose**
  - Default: False

### 3.22.3 Get a head of a Rosetta silent file

**rna_roseta_head.py** - get a head of a Rosetta silent file.

**Example:**

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

Silent file:

```plaintext
[peyote2] thf head -n 100 thf.out
SEQUENCE:
→ ggagaguagauucguuagugugagaauggauucagucaacagaagcgagcggugauucguacauccgucuca
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
REMARK BINARY SILENFILE
SCORE: -601.975 0.000 31.113 -16.960 ...
→ -3.888 20.501 -302.742 -38.531 -158.004 ...
→ -80.764 -110.053 23.750 0.000 ...
→ 33.601 32 6 86 0.000001_000
FOLD_TREE EDGE 1 4 -1 JEDGE 4 85 1 C4 C2 END EDGE 4 5 -1 EDGE 85 80 -1 EDGE...
→ 85 89 -1 JEDGE 80 40 5 C4 C2 END EDGE 80 78 -1 EDGE 40 43 -1 EDGE 40 33 -1 ...
→ JEDGE 33 45 4 C4 C2 END EDGE 45 54 -1 EDGE 45 44 -1 EDGE 33 20 -1 JEDGE ...
→ 20 65 3 C2 C4 END EDGE 65 67 -1 EDGE 20 17 -1 EDGE 65 63 -1 JEDGE 17 69 2 ...
→ C4 C2 END JEDGE 63 58 6 C4 C2 END EDGE 17 6 -1 EDGE 58 59 -1 EDGE 63 ...
→ 60 -1 EDGE 69 77 -1 EDGE 58 55 -1 EDGE 69 68 -1 0.000001_000
RT -0.987743 0.139354 0.0703103 -0.0509304 -0.0766626 -0.135963 0.989404 -0.0509304 ...
→ 996224 6.25631 0.10354 0.0647696 0.000001_000
RT -0.98312 0.1587 -0.091045 0.166923 0.981743 -0.0912024 0.074909 -0.10486 ...
→ 5.89962 -1.95819 -0.1075 0.000001_000
RT -0.987645 0.154078 0.0285994 0.153854 0.988044 -0.00989514 -0.0297821 -0.000001_000 ...
→ 0.999542 6.13138 1.047 0.115722 0.000001_000
```

(continues on next page)
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
[peyote2] thf rna_rosetta_n.py thf_10000.out
10000
```

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

3.22.3.1 Positional Arguments

- `file` ade.out

3.22.3.2 Named Arguments

- `-v, --verbose` Default: False
- `-n, --nstruc` Default: 10000

3.22.4 Cluster

```
usage: rna_rosetta_cluster.py [-h] [--no_select]
    [--radius-inc-step RADIUS_INC_STEP]
    [--limit-clusters LIMIT_CLUSTERS]
    file n
```

3.22.4.1 Positional Arguments

- `file` ade.out
- `n` # of total structures

3.22.4.2 Named Arguments

- `--no_select` Don’t run selection once again. Use selected.out in the current folder
A wrapper to ROSETTA tools for RNA modeling


```
rna_rosetta_cluster.py ade_min.out 20000
```

Take n * 0.005 (0.5%) of all frames and put them into selected.out. Then the tool clusters this selected.out.

```
rna_pdb_tools.utils.rna_rosetta.rna_rosetta_cluster.cluster(radius, limit_clusters)
```

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

```
rna_pdb_tools.utils.rna_rosetta.rna_rosetta_cluster.cluster_loop(ns, 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.

```
rna_pdb_tools.utils.rna_rosetta.rna_rosetta_cluster.extract()
```

```
rna_pdb_tools.utils.rna_rosetta.rna_rosetta_cluster.get_no_structures(file)
```

```
rna_pdb_tools.utils.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)

```
rna_pdb_tools.utils.rna_rosetta.rna_rosetta_cluster.get_parser()
```

```
rna_pdb_tools.utils.rna_rosetta.rna_rosetta_cluster.get_selected(file, nc)
```

Get selected for clustering

```
rna_pdb_tools.utils.rna_rosetta.rna_rosetta_cluster.run()
```

Pipeine for modeling RNA

#### 3.22.5 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.

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.

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

3.22. ROSETTA
rosetta_submit.py min_cmdline min_out 1 24

rosetta_submit.py min_cmdline min_out [1] [16] The first number states how many processors to use for each line in min_cmdline. Here, enter 1 for the first input so that the total number of processors used will be equal to the number of processors entered with the “-proc” flag in command line [12], above. The second number states the maximum time each job will be allowed to run (walltime). Start the run with the appropriate command listed by the out-put above (e.g., source qsubMPI for the Stampede cluster).

E.g. for 20k silet file, 1/6 will be minimized = 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

3.22.5.1 Positional Arguments

file ade.out

3.22.5.2 Named Arguments

-g, --go Default: False
-c, --cpus default: 200

3.22.6 Check progress

rna_rosetta_cluster.py - a script to cluster a silent file

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] [-k] dir

3.22.6.1 Positional Arguments

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

Default: “/home/magnus/rosetta-runs”
3.22.6.2 Named Arguments

- **-v, --verbose**  
  be verbose  
  Default: False

- **-m, --min-only**  
  check only for mo folder  
  Default: False

- **-k, --kill**  
  kill (qdel) jobs if your reach limit (nstruct) of structure that you want, right now is 10000 structures  
  Default: False

3.23 Plotting

3.24 Misc

3.24.1 rna_sali2dotbracket

```
usage: rna_sali2dotbracket [-h] filename
```

3.24.1.1 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_pdb_tools.utils.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:

```
[', ', ',', ',', ',', ',', ',', ',', ',', '--...<<<[[[...]]]..>>>>', ',', ', ']
```

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

\textbf{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.

\texttt{rna_pdb_tools.utils.rna_sali2dotbracket.rna_sali2dotbracket.get_parser()}
\texttt{repl(m)}

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

\subsection*{3.24.2 rna_add_chain}

Example:

\begin{verbatim}
.\!/rna_add_chain.py -c X ../../../input/1msy_rnakbmd_decoy999_clx_noChain.pdb > ../../../output/1msy_rnakbmd_decoy999_clx_noChain_Xchain.pdb
\end{verbatim}

\begin{table}[h]
\centering
\begin{tabular}{cccccccc}
\hline
\textbf{ATOM} & \textbf{O5'} & \textbf{U} & \textbf{1} & \textbf{42.778} & \textbf{25.208} & \textbf{46.287} & \textbf{1.00} & \textbf{0.00} \\
\hline
\texttt{ATOM} & \texttt{2} & \texttt{C5'} & \texttt{U} & \texttt{1} & \texttt{42.780} & \texttt{26.630} & \texttt{45.876} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{3} & \texttt{C4'} & \texttt{U} & \texttt{1} & \texttt{42.080} & \texttt{27.526} & \texttt{46.956} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{4} & \texttt{O4'} & \texttt{U} & \texttt{1} & \texttt{43.013} & \texttt{28.044} & \texttt{47.963} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{5} & \texttt{C1'} & \texttt{U} & \texttt{1} & \texttt{42.706} & \texttt{29.395} & \texttt{48.257} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{6} & \texttt{N1} & \texttt{U} & \texttt{1} & \texttt{43.857} & \texttt{30.305} & \texttt{47.703} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{7} & \texttt{C6} & \texttt{U} & \texttt{1} & \texttt{45.057} & \texttt{29.857} & \texttt{47.308} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{8} & \texttt{C5} & \texttt{U} & \texttt{1} & \texttt{46.025} & \texttt{30.676} & \texttt{46.763} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{9} & \texttt{C4} & \texttt{U} & \texttt{1} & \texttt{45.720} & \texttt{32.110} & \texttt{46.702} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{10} & \texttt{O4} & \texttt{U} & \texttt{1} & \texttt{46.444} & \texttt{32.975} & \texttt{46.256} & \texttt{1.00} & \texttt{0.00} \\
\hline
\end{tabular}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{cccccccc}
\hline
\textbf{ATOM} & \textbf{O5'} & \textbf{U} & \textbf{X} & \textbf{1} & \textbf{42.778} & \textbf{25.208} & \textbf{46.287} & \textbf{1.00} & \textbf{0.00} \\
\hline
\texttt{ATOM} & \texttt{2} & \texttt{C5'} & \texttt{U} & \texttt{X} & \texttt{1} & \texttt{42.780} & \texttt{26.630} & \texttt{45.876} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{3} & \texttt{C4'} & \texttt{U} & \texttt{X} & \texttt{1} & \texttt{42.080} & \texttt{27.526} & \texttt{46.956} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{4} & \texttt{O4'} & \texttt{U} & \texttt{X} & \texttt{1} & \texttt{43.013} & \texttt{28.044} & \texttt{47.963} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{5} & \texttt{C1'} & \texttt{U} & \texttt{X} & \texttt{1} & \texttt{42.706} & \texttt{29.395} & \texttt{48.257} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{6} & \texttt{N1} & \texttt{U} & \texttt{X} & \texttt{1} & \texttt{43.857} & \texttt{30.305} & \texttt{47.703} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{7} & \texttt{C6} & \texttt{U} & \texttt{X} & \texttt{1} & \texttt{45.057} & \texttt{29.857} & \texttt{47.308} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{8} & \texttt{C5} & \texttt{U} & \texttt{X} & \texttt{1} & \texttt{46.025} & \texttt{30.676} & \texttt{46.763} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{9} & \texttt{C4} & \texttt{U} & \texttt{X} & \texttt{1} & \texttt{45.720} & \texttt{32.110} & \texttt{46.702} & \texttt{1.00} & \texttt{0.00} \\
\texttt{ATOM} & \texttt{10} & \texttt{O4} & \texttt{U} & \texttt{X} & \texttt{1} & \texttt{46.444} & \texttt{32.975} & \texttt{46.256} & \texttt{1.00} & \texttt{0.00} \\
\hline
\end{tabular}
\end{table}

\texttt{rna_pdb_tools.utils.misc.rna_add_chain.get_parser()}  

\texttt{usage: rna_add_chain [-h] [-c CHAIN] file}
3.24.2.1 Positional Arguments

file          file

3.24.2.2 Named Arguments

-\texttt{c}, --chain \quad \text{a new chain, e.g. A}

3.24.3 Cluster load

A very simple tool to see your cluster load per user:

\begin{verbatim}
MAX_JOBS:  1000
#jobs cluster 917 load:  0.917 to use:  83
#jobs you    749 load:  0.749 to use: 251
{'deepak': 160, 'azyla': 8, 'magnus': 749}
1 azyla     r  8
20 magnus   r 10
16 deepak   r 10
329 magnus  r  1
22 magnus   qw 10
\end{verbatim}

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. FARN\_A runs.).

\begin{verbatim}
rna\_pdb\_tools.utils.cluster\_load.cluster\_load.get\_parser()
rna\_pdb\_tools.utils.cluster\_load.cluster\_load.per\_user()
    get stats (#cpus) per user
rna\_pdb\_tools.utils.cluster\_load.cluster\_load.stats\_for\_cluster()
    get stats (#jobs) per cluster
rna\_pdb\_tools.utils.cluster\_load.cluster\_load.stats\_for\_user()
    get stats (#jobs) per user
\end{verbatim}
Take txt such as A:1-31+B:1-11 and parse into:

```
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

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

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

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

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

**Warning:** e.g. for A:1-31, resi 31 is included
CHAPTER 5

Workflows

5.1 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 2655 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 struct/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:
5.2 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
```
5.3 Example #3

Starting structure doesn’t have chain id:

```
# 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_rnabnnc_decoy0473_amb.clx/A/ 1
5
11 14 16 21 26
G U G G G G G C A G G C U G G G G G C A G C

/1duq_rnabnnc_decoy0008_amb.clx/A/ 1
5
11 /B/14 15 21 26
```

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

You try to calculate RMSDs for RNA Puzzles #1:

```
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`.

![DiffMerge](image)

you 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  -15.589  -2.874  76.053  1.00  0.00  C
ATOM   974  O6  G  B  23  -15.497  -3.930  75.430  1.00  0.00  O
ATOM   975  N1  G  B  23  -16.234  -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
```
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_1_rpr_v2.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.
6.1 Run in batch

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

```
$ 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):

```
$ 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`:

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

6.2 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:

```
$ head lmsy_rnakbmd_decoy1661_cix.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.out

Classifier: Clarna

chains: A 1 27

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>18</td>
<td>bp A G</td>
<td>HS_tran</td>
<td>0.7620</td>
</tr>
<tr>
<td>12</td>
<td>17</td>
<td>bp C G</td>
<td>WW_cis</td>
<td>0.7242</td>
</tr>
<tr>
<td>10</td>
<td>19</td>
<td>bp U A</td>
<td>WH_tran</td>
<td>0.7826</td>
</tr>
<tr>
<td>11</td>
<td>18</td>
<td>bp A G</td>
<td>SH_tran</td>
<td>0.7620</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>bp G U</td>
<td>SH_cis</td>
<td>0.8563</td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>bp C G</td>
<td>WW_cis</td>
<td>0.8913</td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>bp C G</td>
<td>WW_cis</td>
<td>0.9038</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>bp U A</td>
<td>WW_cis</td>
<td>0.9222</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>bp C G</td>
<td>WW_cis</td>
<td>0.8114</td>
</tr>
<tr>
<td>2</td>
<td>26</td>
<td>bp G U</td>
<td>WW_cis</td>
<td>0.8500</td>
</tr>
</tbody>
</table>

Read more about *sed*.

### 6.3 In PyMOL

**Rename a chain:**

```
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:

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

To renumber residues:

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

Read more.

The example of the pistol ribozyme editing.
Run:

```
PyMOL> alter (sel), chain="B"
  Alter: modified 236 atoms.
PyMOL> alter (chain B), resv -= 51
  Alter: modified 236 atoms.
PyMOL> sort
```
6.4 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>&quot;ATOM &quot;</td>
<td></td>
</tr>
<tr>
<td>7 - 11</td>
<td>6:11</td>
<td>Integer</td>
<td>serial</td>
<td>Atom serial number.</td>
</tr>
<tr>
<td>13 - 16</td>
<td>12:16</td>
<td>Atom</td>
<td>name</td>
<td>Atom name.</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 (\text{\AA}).</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 (\text{\AA}).</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 (\text{\AA}).</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. #1</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>
6.5 Working with cluster

Tips:

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

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

Numbering:

```
|1.......|10.......|20.......|30.......|40.......|50.......|60.......|70.......|80.......|
|  ...    |         |         |         |         |         |         |         |
|123456789|123456789|123456789|123456789|123456789|123456789|123456789|123456789|123456789|
```
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-1dce7669dad9_ALL_thrs6.20A_clust01-000001_AA_out_rpr.pdb 
d311d821-a075-4df0-bd7d-1dce7669dad9_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-pdb-tools package.
In terminal:

```
$ 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
ENDMDL
END
===> rp19_bujnicki.pdb <==
ATOM 1325 C5 C B 22 29.827 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
ENDMDL
END
```
Read for more interesting functions https://daslab.stanford.edu/site_data/docs_pymol_rhiju.pdf

```python
rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA.align_all(subset=[])  
Superimpose all open models onto the first one. This may not work well with selections.

This function is probably taken from https://daslab.stanford.edu/site_data/docs_pymol_rhiju.pdf
```

```python
rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA.clarna()  
Get contacts classification of the selected fragment based on ClaRNA.
```
rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA.color_aa_types()
    Color aminoacides types like in Cider (http://pappulab.wustl.edu/CIDER/)

rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA.color_by_text(txt)
    Helper function used for color-coding based on residue indexes ranges.

rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA.color_obj(rainbow=0)
    stolen from :)

    AUTHOR     Gareth Stockwell
    USAGE      color_obj(rainbow=0)

        This function colours each object currently in the PyMOL heirarchy with a different colour. Colours used
        are either the 22 named colours used by PyMOL (in which case the 23rd object, if it exists, gets the same
        colour as the first), or are the colours of the rainbow

rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA.color_rbw(rainbow=0)
    similar to color_obj() but this time colors every object as rainbow

rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA.exe(cmd, verbose=False)
    Helper function to run cmd. Using in this Python module.

rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA.get_pdb()
    Get PDB content of selection.
Get contacts classification based on ClaRNA.

For Educational Use Only

```python
rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA.get_seq()
```

Get contacts classification based on ClaRNA.
Sphäre and yellow inorganic, such as Mg.

A shortcut for putting a seq at the bottom. Pretty cool for screenshots with names of objects.
rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA.rcomp()

RNA like in papers ;-)  
Similar to rc() but this time it colors each (and every) structure in different colour. Great on viewing-comparing superimposed structures.

rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA.rgyration(selection='(all)', quiet=1)

[PyMOL] RES: radius of gyration From: Tsjerk Wassenaar <tsjerkw@gm...> - 2011-03-31 14:07:03 https://sourceforge.net/p/pymol/mailman/message/27288491/

DESCRIPTION

Calculate radius of gyration

USAGE

rgyrate [ selection ]

:::warning::: if nothing is selected function is calculating radius of gyration for all pdbs in current Pymol session

rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA.rna_cartoon()

http://www-cryst.bioc.cam.ac.uk/members/zbyszek/figures_pymol
Represent your RNA.

Color-coding for secondary structure elements for the RNA Puzzle 17.

For the variant:

```
CGUGGUUAGGGCCACGUUAAUAGUUGCUUAAAGGCCUAAGCGAGGGAACCCAUAAUCAACAGGGCAAA
(((.[[[[[.]]]].))........((((.....))))))....((((.......)))).....))..)))))
```

# len 62-nt
Color-coding for secondary structure elements for the RNA Puzzle 17.

For the variant:

```text
CGUGGUUAGGCCCAGUUAAUAGUUGCUUAAGCCCCUAAGCGUUAUCAGGUGCAA
((((.[[[[.]]]]))))...(((([[[.]]]])))...(((([[[[.]]]]))))
# len 58-nt
```

See rp17()

The function creates super-cool cartoon-like RNA and colors each structure as a rainbow. Good to view aligned structures in a grid.
Get Secondary Structure of (sele) based on py3dna.py.

```python
rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA.spli()
rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA.ss()
```

Get Secondary Structure of (sele) based on py3dna.py.
The same as ss() but for all objects.

**8.1 PyMOL Drawing**

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).

```python
rna_pdb_tools.utils.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.

```python
rna_pdb_tools.utils.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/

```python
rna_pdb_tools.utils.pymol_drawing.pymol_drawing.draw_dists(interactions)
```

```python
rna_pdb_tools.utils.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:

```bash
run <path>rna-pdb-tools/utils/pymol_drawing/pymol_dists.py
```

and type:

```python
draw_dists([[29, 41], [7, 66], [28, 42], [51, 63], [50, 64], [2, 71], [5, 68], [3, 70], [31, 39], [4, 69], [5, 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]])
```
8.2 RNA Helix Vis (draw helices using PyMOL)

8.3 Install

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

```bash
# rna-pdb-tools
RNA_PDB_TOOLS="/Users/magnus/work-src/rna-pdb-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/clarna_play/clarna_run.py')
run ~/work-src/rna-pdb-tools/rna_pdb_tools/utils/PyMOL4RNA/PyMOL4RNA.py
run ~/work-src/rna-pdb-tools/rna_pdb_tools/utils/pymol_drawing/pymol_drawing.py
run ~/work-src/rna-pdb-tools/rna_pdb_tools/utils/rna_filter/pymol_dists.py
```

The plugins have been tested with MacPyMOL version 1.7.4.5 Edu.
The tools collected here as rna-pdb-tools, ideally could be also used in IPython/Jupyter (https://ipython.org/notebook.html) notebooks. We believe it would be extremely 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.

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.

E.g https://github.com/mmagnus/rna-pdb-tools/blob/master/rp18.ipynb

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-pdb-tools

RNA-PDB-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.
G33KB00K - eXtreme eXtendable note taking system for nerds/geeks (including scientists!) (= beautiful html generator of your markdown-based notes) docs: http://geekbook.rtfd.io

Marcin Magnus (mmagnus) & Pietro Boccaletto (akaped)

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

A neat way how to combine Emacs/Atom/Sublime editor + Markdown Syntax + Git + Html engine (bootstrap/python) to get the best notes-talking experience ever. Highly customizable with plugins written in Python. What’s the most important, under the hood it’s just a set of Markdown files.. you can do with them whatever you want, e.g. you can Pandoc (http://pandoc.org/epub.html) them to epub (that’s origin of “book” part of the name).

## 11.1 Draw VARNA-based image of RNA secondary structure

Type:

```plaintext
[ss: rna]
UUUCGUUAUGCCGUAUAAGGUUCGGCAGUUUCUACCACAGCCGUAAACUGUUUGACUACAGUAA
((.(((((...(............))).......(((((............))....)))))).)))
</pre>
```

**Warning:** Keep exactly the same syntax as in the example above and below.

The syntax:

```
<pre>
[ss:/name of your seq/]
/seq/
/ss/
</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.
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.

### 12.1 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.genesico.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.
```

### 12.2 Learn Git

Model used by Git vs Svn:

```plaintext
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

### 12.3 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.
Download rna-pdb-tools by clicking here [https://github.com/mmagnus/rna-pdb-tools/archive/master.zip](https://github.com/mmagnus/rna-pdb-tools/archive/master.zip) or using git:

```bash
$ git clone https://github.com/mmagnus/rna-pdb-tools.git
$ cd rna-pdb-tools
```

(git is better if you want to contribute to the package and if you want to get pretty frequent updates).

The first step is “zero” because not all requirements are needed to start working with rna-pdb-tools.

To install the full set of requirements, use pip:

1. `pip install -r docs/requirements.txt`

and install the package itself in three steps:

1. add the path to the package to your PYTHONPATH (in ~/.bashrc), e.g. `export PYTHONPATH=$PYTHONPATH:/home/magnus/src/rna-pdb-tools/`
2. add the path to the bin folder of the package to your PATH (in ~/.bashrc), e.g. `export PATH=$PATH:/home/magnus/src/rna-pdb-tools/bin/`
3. add the path to the bin folder of the package to your PATH (in ~/.bashrc), e.g. `export RNA_PDB_TOOLS=/home/magnus/src/rna-pdb-tools/`
4. and run the install script:

```bash
  rna-pdb-tools git:(master) ./install_links_bin.sh
  Installed in ./bin
  rmsd_calc_to_target.py
```

should be OK now :-)

To set your own configuration, please first:

```bash
  cp rpt_config_local.py_sample rpt_config_local.py # in rna-pdb-tools/rna_pdb_tools
```

and then edit `rpt_config_local.py` as you need. In my case it is:
VARNA_PATH = '/Users/magnus/skills/rnax/varna_tut/'
VARNA_JAR_NAME = 'VARNA.jar'
CHAPTER 14

Configuration

Keep configuration syntax like:

```python
from rna_pdb_tools.rpt_config import CPUS_CLUSTER
# since we use export PYTHONPATH=$PYTHONPATH:/home/magnus/src/rna-pdb-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](http://www.sphinx-doc.org/en/stable/ext/napoleon.html#type-annotations)
CHAPTER 16

Add a new util

1. Create a new folder in rna-pdb-tools/rna_pdb_tools/utils with your util. The folder will be seen online after your push at https://github.com/mmagnus/rna-pdb-tools/tree/master/rna_pdb_tools/utils. We will walk you through this simple example https://github.com/mmagnus/rna-pdb-tools/tree/master/rna_pdb_tools/utils/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 util to install_links_bin.sh at the top folder of rna-pdb-tools:

```bash
ln -s $curr_dir/rna_pdb_tools /utils/<util folder>/<util script name with .py>
$curr_dir/bin/<util script name with .py>
```

E.g.

```bash
ln -s $curr_dir/rna_pdb_tools/utils/renum_pdb_to_aln/renum_pdb_to_aln.py $curr_dir/bin/rna_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 util to the documentation. The util has to be “importable”, so don’t forget to create __init__.py inside your util directory. Next, go to rna-pdb-tools/docs/source and edit utils.rst. Add wherever you think your tool will fit - lines like:

```rst
Renumber a pdb file according to alignment
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.. argparse::
  :ref: rna_pdb_tools.utils.<util folder>.<util script name>.get_parser
```

(continues on next page)
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-pdb-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 and also by Travis for continuous testing.

You can open the documentation compiled locally under a link such as `file://<path to rna-pdb>/rna-pdb-tools/docs/build/html/index.html`, e.g. `file:///Users/magnus/work/src/rna-pdb-tools/docs/build/html/index.html`.

5. The very last step is to add your util `test.sh` to the main testing script. Edit `rna-pdb-tools/test.sh` and add:

```bash
cd ./utils/<util folder>/
./test.sh
cd ../..
```

6. Run this main test (`./test.sh`) and see if the util 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 util"
$ git push
```

This testing is very, very rough and we are moving to have more test in `py.test`.

**Warning:**
CHAPTER 17

Indices and tables

- genindex
- modindex
- search
Python Module Index
A

add_ids() (rna_pdb_tools.utils.clanstix.rna_clanstix.RNAStructClans method), 48
align_all() (in module rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA), 107
align_seq() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNAalignment method), 36
Atom (class in rna_pdb_tools.utils.simrna_trajectory.simrna_trajectory), 74

calc_rmsd() (in module rna_pdb_tools.utils.rna_calc_rmsd.rna_calc_rmsd), 51
calc_rmsd() (in module rna_pdb_tools.utils.rna_calc_rmsd.rna_calc_rmsd_all_vs_all), 55
calc_rmsd_pymol() (in module rna_pdb_tools.utils.rna_calc_rmsd.rna_calc_rmsd), 51
clama() (in module rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA), 107
clarna() (in module rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA),
collapse_view() (in module rna_pdb_tools.utils.rna_pdb_tools_lib), 19
color_aa() (in module rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA), 108
color_by_text() (in module rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA), 108
color_obj() (in module rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA), 108
color_rbw() (in module rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA), 108
cmscan() (rna_pdb_tools.RfamSearch.RfamSearch method), 28
copy_ss_cons_to_all() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNAalignment method), 36
copy_ss_cons_to_all_editing_sequence() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNAalignment method), 36
draw_circle() (in module rna_pdb_tools.utils.pymol_drawing.pymol_drawing), 115
draw_circle_selection() (in module rna_pdb_tools.utils.pymol_drawing.pymol_drawing), 116
draw_dist() (in module rna_pdb_tools.utils.pymol_drawing.pymol_drawing),
describe() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNAalignment method), 36
do_job() (in module rna_pdb_tools.utils.rna_calc_inf.rna_calc_inf), 56
dist_from_matrix() (rna_pdb_tools.utils.clanstix.rna_clanstix.RNAStructClans method), 48
clarna_compare() (in module rna_pdb_tools.utils.clarna_app.clarna_app), 72
clarna_run() (in module rna_pdb_tools.utils.clarna_app.clarna_app), 72
clean_up() (rna_pdb_tools.utils.rna_x3dna.rna_x3dna.x3DNA method), 71
cluster() (in module rna_pdb_tools.utils.rna_rosetta.rna_rosetta_cluster), 85
cluster_loop() (in module rna_pdb_tools.utils.rna_rosetta.rna_rosetta_cluster), 85
CMAlign (class in rna_pdb_tools.utils.rna_alignment.rna_alignment), 42

color_rbw() (in module rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA), 108
Index
get_parser() (in module rna_pdb_tools.utils.clanstix.rna_clanstix), 48
get_parser() (in module rna_pdb_tools.utils.clarna_app.clarna_app), 73
get_parser() (in module rna_pdb_tools.utils.cluster_load.cluster_load), 89
get_parser() (in module rna_pdb_tools.utils.misc.rna_add_chain), 88
get_parser() (in module rna_pdb_tools.utils.renum_pdb_to_aln.renum_pdb_to_aln), 45
get_parser() (in module rna_pdb_tools.utils.rna_alignment.random_assignment_of_nucleotides), 41
get_parser() (in module rna_pdb_tools.utils.rna_calc_inf.rna_calc_inf), 56
get_parser() (in module rna_pdb_tools.utils.rna_calc_rmsd.rna_calc_rmsd), 52
get_models_from_dir() (in module rna_pdb_tools.utils.rna_calc_rmsd.rna_calc_rmsd_all_vs_all), 55
get_mapuzzle_ready() (rna_pdb_tools.utils.rna_pdb_tools_lib.RNAStructure method), 6, 11, 16
get_secstruct() (rna_pdb_tools.utils.rna_x3dna.rna_x3dna.x3DNA method), 5, 71
get_selected() (in module rna_pdb_tools.utils.rna_rosetta.rna_rosetta_cluster), 85
get_seq() (in module rna_pdb_tools.utils.PyMOL4RNA.PyMOL4RNA), 109
get_seq() (in module rna_pdb_tools.utils.renum_pdb_to_aln.renum_pdb_to_aln), 45
get_seq() (rna_pdb_tools.utils.rna_pdb_tools_lib.RNAStructure method), 13, 18
get_seq() (rna_pdb_tools.utils.rna_alignment.rna_alignment.CMAAlign method), 42
get_seq() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNASeq method), 38
get_seq() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNAalignment method), 38
get_seq_ss() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNASeq method), 38
get_shift_seq_in_align() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNAalignment method), 38
get_ss_cons() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNASeq method), 38
get_ss_cons_pk() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNASeq method), 38
get_ss_remove_gaps() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNASeq method), 38
get_version() (rna_pdb_tools.utils.rna_pdb_tools_lib), 20
get_pdb() (rna_pdb_tools.utils.rna_pdb_tools_lib.RNAStructure method), 31
get_text() (rna_pdb_tools.utils.rna_pdb_tools_lib.RNAStructure method), 18
get_the_closest_seq_to_ref_seq() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNASeq method), 38
get_res_code() (rna_pdb_tools.utils.rna_pdb_tools_lib.RNAStructure method), 16
get_res_num() (rna_pdb_tools.utils.rna_pdb_tools_lib.RNAStructure method), 16
get_rna_models_from_dir() (rna_pdb_tools.utils.rna_pdb_tools_lib.RNAStructure method), 14
ss_no_gaps (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNASeq attribute), 31
ss_to_bps() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNASeq method), 35
stats_for_cluster() (in module rna_pdb_tools.utils.cluster_load.cluster_load), 89
stats_for_user() (in module rna_pdb_tools.utils.cluster_load.cluster_load), 89
std_resn() (rna_pdb_tools.rna_pdb_tools_lib.RNAStructure method), 19
subset() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNAAlignment method), 39

T
tail() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNAAlignment method), 40
trimmed_rf_and_ss() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNAAlignment method), 40

U
un_nmr() (rna_pdb_tools.rna_pdb_tools_lib.RNAStructure method), 19

W
write() (rna_pdb_tools.rna_pdb_tools_lib.RNAStructure method), 19
write() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RChie method), 44
write() (rna_pdb_tools.utils.rna_alignment.rna_alignment.RNAAlignment method), 40
write_align() (in module rna_pdb_tools.utils.rna_alignment.random_assignment_of_nucleotides), 41
write_seq() (in module rna_pdb_tools.utils.rna_alignment.random_assignment_of_nucleotides), 41
write_struc() (in module rna_pdb_tools.utils.renum_pdb_to_aln.renum_pdb_to_aln), 46

X
x3DNA (class in rna_pdb_tools.utils.rna_x3dna.rna_x3dna), 5, 71
x3DNAMissingFile, 5, 71