### 1 Getting Started: I want to . . .
1.1 fetch a structure from the PDB database .......................................................... 3
1.2 fetch a biological 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 ......................................... 6
1.7 edit your structure (rename chain) ..................................................................... 6
1.8 extract part of your structure ............................................................................ 6
1.9 find missing atoms in my structure ................................................................... 7
1.10 mutate residues ............................................................................................... 7
1.11 add missing atoms ......................................................................................... 9

### 2 RNA PDB Tools
2.1 rna_pdb_toolsx ................................................................................................. 13
2.2 get RNAPuzzle ready ....................................................................................... 16
2.3 get sequence .................................................................................................. 19
2.4 fetch ............................................................................................................... 20
2.5 fetch Biological Assembly ............................................................................. 20
2.6 delete .............................................................................................................. 21
2.7 edit ................................................................................................................. 21
2.8 the library ...................................................................................................... 21
2.9 PDB Edit Bfactor/Occupancy ......................................................................... 28
2.10 Add chain to a file ......................................................................................... 30
2.11 Measure distance between atoms ................................................................. 31

### 3 RNA Tools
3.1 RNA Sequence .............................................................................................. 33
3.1.1 Installation .................................................................................................. 33
3.1.1.1 ContextFold ......................................................................................... 33
3.1.1.2 ViennaRNA .......................................................................................... 33
3.1.1.3 ipknot OSX ......................................................................................... 34
3.1.1.4 RNA Structure ..................................................................................... 34
3.1.1.5 MC-Sym ............................................................................................... 34
3.1.2 FAQ ........................................................................................................... 34
3.1.3 TIPS .......................................................................................................... 34
3.2 RNA Secondary Structure ............................................................................... 38
rna-tools: a toolbox to analyze structures and simulations of RNA

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

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

If something does not work for you, please make an issue, using https://github.com/mmagnus/rna-tools/issues.
CHAPTER 1

Getting Started: I want to . . .

1.1 fetch a structure from the PDB database

Example:

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

1.2 fetch a biologicaly assembly

Example:

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

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

$ cat data/pdb_ids.txt
ly26
lfir

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

$ ls *.pdb
lfir_ba.pdb ly26_ba.pdb
1.3 get sequences of a bunch of PDB files

Example:

```bash
$ rna_pdb_toolsx.py --get-seq *.pdb
# 1xjr
> A:1-47
GGAGUUACCACGAGCCACCGGAGAUACGAGAGGGGUACUGA
# 6TNA
> A:1-76
CCGGAGAUAaCUAGAUuGGAGAGCGccAGAcUqAucUGGAGgUCcUGUuCGaUCCACAGAAUUCGCACCA
# rp2_bujnicki_1_rpr
> A:1-15
CCGGAGAACUACUG
> B:1-10
CCGGCCACGU
> C:1-15
CCGGAGAACUACUG
> D:1-10
CCGGCCACGU
> E:1-15
CCGGAGAACUACUG
> F:1-10
CCGGCCACGU
> G:1-15
CCGGAGAACUACUG
> H:1-10
CCGGCCACGU
```

in some more fancy way ;-)  

```bash
$ rna_pdb_toolsx.py --get-seq 
   --oneline 3_bujnicki_1_rpr* 
   --color-seq --compact
```

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-tools/rna_tools/tools/rna_x3dna folder.
Edit this line:
BINARY_PATH = <path to your x3dna-dssr file>
matching the path with the path of your x3dna-dssr file.
e.g. in my case: BINARY_PATH = ~/bin/x3dna-dssr.bin
```

For one structure you can run this script as:
For multiple structures in the folder, run the script like this:

```bash
[mn] py3dna$ git:(master) ./rna_x3dna.py test_data/*
```

```
>6TNA nts=76 [6TNA] -- secondary structure derived by DSSR
GCGAUUUAaUCAGaaUUggGAAaAGcGaaCaGaaCAMuAgAaPcUGAAgUGcUGAGtPCGaaUCCACAGAAUUCGCAACAA

(((((.((....[[....]])))))))..(((((.((....[[....]])))))))..(((((.((....[[....]])))))))
```
1.6 get numbering of your structure and rename chains

Rename chain B in structure 4_das_1_rpr.pdb:

```bash
$ rna_pdb_toolsx.py --get-seq 4_das_1_rpr.pdb > 4_das_1_rpr.pdb
GGCUUAUCCACUAGUUGAGGAGGAUCAUGCGAUGCCAUAGUUGGAGGGACUGGCCCGAUGAAACCCGGCAACCACACUAGUCUAGCUUCCUGCUGACGCUUCCUGCUGAUAUCCUGG
$ rna_pdb_toolsx.py --edit 'B:1-126>A:1-126' 4_das_1_rpr.pdb > 4_das_1_rpr2.pdb
$ rna_pdb_toolsx.py --get-seq 4_das_1_rpr2.pdb > 4_das_1_rpr2.pdb
```

1.7 edit your structure (rename chain)

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
```

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):

```bash
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 extract part of your structure

Example:

```bash
$ rna_pdb_toolsx.py --extract A:1-4 13_Bujnicki_1_rpr.pdb
```

<table>
<thead>
<tr>
<th>ATOM</th>
<th>1</th>
<th>P</th>
<th>G</th>
<th>A</th>
<th>-16.883</th>
<th>-12.441</th>
<th>8.021</th>
<th>1.00</th>
<th>0.00</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATOM</td>
<td>2</td>
<td>O P1</td>
<td>G</td>
<td>A</td>
<td>-15.777</td>
<td>-12.225</td>
<td>8.969</td>
<td>1.00</td>
<td>0.00</td>
<td>O</td>
</tr>
<tr>
<td>ATOM</td>
<td>3</td>
<td>O P2</td>
<td>G</td>
<td>A</td>
<td>-16.752</td>
<td>-11.535</td>
<td>6.892</td>
<td>1.00</td>
<td>0.00</td>
<td>O</td>
</tr>
<tr>
<td>ATOM</td>
<td>4</td>
<td>O5'</td>
<td>G</td>
<td>A</td>
<td>-16.882</td>
<td>-13.822</td>
<td>7.219</td>
<td>1.00</td>
<td>0.00</td>
<td>O</td>
</tr>
<tr>
<td>ATOM</td>
<td>5</td>
<td>C5'</td>
<td>G</td>
<td>A</td>
<td>-16.092</td>
<td>-13.871</td>
<td>6.013</td>
<td>1.00</td>
<td>0.00</td>
<td>C</td>
</tr>
<tr>
<td>ATOM</td>
<td>6</td>
<td>C4'</td>
<td>G</td>
<td>A</td>
<td>-16.314</td>
<td>-15.160</td>
<td>5.206</td>
<td>1.00</td>
<td>0.00</td>
<td>C</td>
</tr>
<tr>
<td>ATOM</td>
<td>7</td>
<td>O4'</td>
<td>G</td>
<td>A</td>
<td>-17.723</td>
<td>-14.932</td>
<td>4.905</td>
<td>1.00</td>
<td>0.00</td>
<td>O</td>
</tr>
<tr>
<td>ATOM</td>
<td>8</td>
<td>C3'</td>
<td>G</td>
<td>A</td>
<td>-15.788</td>
<td>-15.216</td>
<td>3.752</td>
<td>1.00</td>
<td>0.00</td>
<td>C</td>
</tr>
<tr>
<td>ATOM</td>
<td>9</td>
<td>O3'</td>
<td>G</td>
<td>A</td>
<td>-14.461</td>
<td>-15.860</td>
<td>3.764</td>
<td>1.00</td>
<td>0.00</td>
<td>O</td>
</tr>
<tr>
<td>ATOM</td>
<td>10</td>
<td>C2'</td>
<td>G</td>
<td>A</td>
<td>-16.841</td>
<td>-15.946</td>
<td>2.969</td>
<td>1.00</td>
<td>0.00</td>
<td>C</td>
</tr>
<tr>
<td>(...)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATOM</td>
<td>84</td>
<td>O2</td>
<td>U</td>
<td>A</td>
<td>-14.553</td>
<td>-5.285</td>
<td>-7.938</td>
<td>1.00</td>
<td>0.00</td>
<td>O</td>
</tr>
</tbody>
</table>
1.9 find missing atoms in my structure

Run:

```
$ rna_pdb_toolsx.py --get-rnapuzzle-ready input/1_das_1_rpr_fixed.pdb
```

1.10 mutate residues

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

```
 rna_pdb_toolsx.py --mutate 'A:1A+2A+3A+4A,B:13A' \
   --inplace output/205d_rmH2o_mutA1234-B1_inplace.pdb
```
Figure. Input structure on the left, mutated structure on the right.

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

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

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

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

The tool is using the function:

```python
RNAStructure.get_rnapuzzle_ready(renumber_residues=True, fix_missing_atoms=True,
rename_chains=True, report_missing_atoms=True,
keep_hetatm=False, backbone_only=False,
no_backbone=False, bases_only=False, verbose=False)
```

Get rnapuzzle (SimRNA) ready structure.

Clean up a structure, get current order of atoms.

**Parameters**

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

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

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

Figure: (Starting from left) input structure, structure with rebuilted atoms, and reference. The B fragment is observed in the reference used here as a “benchmark”, fragment A is reconstructed atoms (not observed in the reference”). 201122

- 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
CHAPTER 2

RNA PDB Tools

2.1 rna_pdb_toolsx

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

Usage:

```
$ rna_pdb_toolsx.py --delete A:46-56 --inplace *.pdb

$ rna_pdb_toolsx.py --get-seq *

# BujnickiLab_RNApuzzle14_n01bound
> A:1-61
# BujnickiLab_RNApuzzle14_n02bound
> A:1-61
CGUUAGCCCAGAAACUGGGCGGAAGUAAGGCCCAUUGCACUCCGGCCUGAAGCAACGCG

[...]
```

```
usage: rna_pdb_toolsx.py [-h] [--version] [-r] [--no-progress-bar]
[--renum-atoms] [--renum-nmr]
[--renum-residues.dirty] [--undo] [--delete-anisou]
[--fix] [--to-mol2] [--split-alt-locations] [-c]
[--is-pdb] [--is-nmr] [--nrr-dir NMR_DIR] [--un-nmr]
[--orgmode] [--get-chain GET_CHAIN] [--fetch]
[--fetch-ba] [--fetch-chain] [--get-seq]
[--color-seq] [--ignore-files IGNORE_FILES]
[--compact] [--hide-warnings] [--get-ss]
[--rosetta2generic] [--no-hr] [--renumber-residues]
[--dont-rename-chains] [--dont-fix-missing-atoms]
[--inspect] [--collapsed-view] [--cv] [-v]
[--mutate MUTATE] [--edit EDIT]
[--rename-chain RENAME_CHAIN]
[--swap-chains SWAP_CHAINS] [--set-chain SET_CHAIN]
[--replace-chain REPLACE_CHAIN] [--delete DELETE]
[--extract EXTRACT] [--extract-chain EXTRACT_CHAIN]
[--uniq UNIQ] [--chain-first] [--oneline] [--fasta]
```
Positional arguments:

file

Options:

--version

Undocumented

-r=False, --report=False get report

--no-progress-bar=False for -no-progress-bar for -rpr

--renum-atoms=False renumber atoms, tested with --get-seq

--renum-nmr=False Undocumented

--renum-residues-dirty=False Undocumented

--undo=False undo operation of action done -inplace, , rename “backup files” .pdb~ to pdb, ALL files in the folder, not only ~ related to the last action (that you might want to revert, so be careful)

--delete-anisou=False remove files with ANISOU records, works with -inplace

--fix=False fix a PDB file, ! external program, pdbfixer used to fix missing atoms

--to-mol2=False fix a PDB file, ! external program, pdbfixer used to fix missing atoms

--split-alt-locations=False @todo

-c=False, --clean=False get clean structure

--is-pdb=False check if a file is in the pdb format

--is-nmr=False check if a file is NMR-style multiple model pdb

--nmr-dir make NMR-style multiple model pdb file from a set of files

rna_pdb_toolsx.py --nmr-dir . ‘cwc15_u5_fragments*.pdb’ > ~/Desktop/cwc15_u5.pdb

please use ‘’ for pattern file recognition, this is a hack to deal with folders with thousands of models, if you used only *.pdb then the terminal will complain that you selected to many files.

--un-nmr=False split NMR-style multiple model pdb files into individual models [biopython]

--orgmode=False get a structure in org-mode format <sick!>

--get-chain get chain, one or many, e.g, A, but now also ABC works

--fetch=False fetch file from the PDB db, e.g., 1xjr, use ‘rp’ to fetchthe RNA-Puzzles standardized_dataset [around 100 MB]

--fetch-ba=False fetch biological assembly from the PDB db

--fetch-chain=False fetch a structure in extract chain, e.g. 6bk8 H

--get-seq=False get seq
--color-seq=False color seq, works with --get-seq
--ignore-files files to be ignored, e.g. ‘solution’
--compact=False with --get-seq, get it in compact view
  $ rna_pdb_toolsx.py --get-seq --compact * .pdb
  # 20_Bujnicki_1 ACCCGCAAGGCCGACGGCGCCGCCGCCGCAUGGGU # A:1-68
  # 20_Bujnicki_2 ACCCGCAAGGCCGACGGCGCCGCCGCCGCAUGGGU # A:1-68
  # 20_Bujnicki_3 ACCCGCAAGGCCGACGGCGCCGCCGCCGCAUGGGU # A:1-68
  # 20_Bujnicki_4
--hide-warnings=False hide warnings, works with --get-chain, it hides warnings that given
  changes are not detected in a PDB file
--get-ss=False get secondary structure
--rosetta2generic=False convert ROSETTA-like format to a generic pdb
--no-hr=False do not insert the header into files
--renumber-residues=False by default is false
--dont-rename-chains=False used only with --get-mapuzzle-ready. By default: --get-mapuzzle-
  ready rename chains from ABC.. to stop behavior switch on this option
--dont-fix-missing-atoms=False used only with --get-mapuzzle-ready
--inspect=False inspect missing atoms (technically decorator to --get-mapuzzle-ready with-
  out actually doing anything but giving a report on problems)
--collapsed-view=False Undocumented
--cv=False alias to collapsed_view
--mutate mutate residues, e.g., --mutate “A:1A+2A+3A+4A,B:1A” to mutate to
  adenines the first four nucleotides of the chain A and the first nucleotide
  of the chain B
--rename-chain edit ‘A>B’ to rename chain A to chain B
--swap-chains B>A, rename A to _, then B to A, then _ to B
--set-chain set chain for all ATOM lines and TER (quite brutal function)
--replace-chain= a file PDB name with one chain that will be used to replace the chain in the
  original PDB file, the chain id in this file has to be the same with the chain
  id of the original chain
--delete= delete the selected fragment, e.g. A:10-16, or for more than one fragment
  --delete ‘A:1-25+30-57’
--extract= extract the selected fragment, e.g. A:10-16, or for more than one fragment
  --extract ‘A:1-25+30-57’
--extract-chain extract chain, e.g. A
--uniq rna_pdb_toolsx.py --get-seq --uniq ‘[:5]’ --compact --chain-first * l
  sort A:1-121 ACCUUUGCGCAACUUGGGAUCUUGGGAUGCGCGCGAGUACC...CA
  # rp13nc3295_min.out.1 A:1-123
ACCUUGCGGACUGGCGAAUCUCUGCUUUGAGCGGCUUUCG...AG # rp1356016_min.out.1 A:1-123 ACCUUGCGGACUGGCGAAUCUCUGCUUUGAGCGGCUUUCG...AG zcp_6537608a_ALL-000001_AA A:1-45 57-71 GGGUCGUGACUGGCGAACAGUUGGGAACCCACCGGGGACGGACCCGGCCGCCCACCCGCCG-CUGGGC # solution

--chain-first=False Undocumented
--oneline=False Undocumented
--fasta=False with --get-seq, show sequences in fasta format, can be combined with --compact (mind, chains will be separated with ' ' in one line)

$ rna_pdb_toolsx.py --get-seq --compact input/20_Bujnicki_1.pdb
> 20_Bujnicki_1 ACCCGCAAGGCCGACGGC GCCGCCGCUGGUGCAAGUCCAGCCACGCUUCGGCGUGGGCGCUCAUGGGU

--cif2pdb=False [PyMOL Python package required]
--pdb2cif=False [PyMOL Python package required]
--get-rnapuzzle-ready=False get RNApuzzle ready (keep only standard atoms). `Be default it does not renumber residues, use --renumber-residues [requires BioPython]
--rpr=False alias to get_rnapuzzle ready)
--keep-hetatm=False keep hetatoms
--inplace=False in place edit the file! [experimental, only for get_rnapuzzle_ready, --delete, --get-ss, --get-seq, --edit-pdb]
--suffix when used with --inplace allows you to change a name of a new file, --suffix del will give <file>_del.pdb (mind added _)

replace-hetatm=False replace `HETATM` with `ATOM` [tested only with --get-rnapuzzle-ready]
--dont-report-missing-atoms=False used only with --get-rnapuzzle-ready
--backbone-only=False used only with --get-rnapuzzle-ready, keep only backbone (= remove bases)
--no-backbone=False used only with --get-rnapuzzle-ready, remove atoms of backbone (define as P OP1 OP2 O5')
--bases-only=False used only with --get-rnapuzzle-ready, keep only atoms of bases

2.2 get RNApuzzle ready

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

    fn
    string - path to the structural file, e.g., "./rna_tools/input/4ts2.pdb"

    name
    string - filename of the structural file, "4ts2.pdb"

    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, keep_hetatom=False, backbone_only=False, no_backbone=False, bases_only=False, verbose=False)

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

![Figure: (Starting from left) input structure, structure with rebuilted atoms, and reference. The B fragment is observed in the reference used here as a “benchmark”, fragment A is reconstructed atoms (not observed in the reference”). 201122](image)

- 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

### 2.3 get sequence

Example:

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

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

    fn
        string – path to the structural file, e.g., “./rna_tools/input/4ts2.pdb”

    name
        string – filename of the structural file, “4ts2.pdb”

    lines
        list – the PDB file is loaded and ATOM/HETATM/TER/END go to self.lines

    get_seq(compact=False, chainfirst=False, fasta=False, addfn=”, color=False)
        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
```
Chains is {'A': {'header': 'A:1-19 26-113 117-172', 'resi': [1, 2, 3, ..., 19, 26, 27, ..., 172], 'seq': ['G', 'C', 'G', ... C', 'G', 'U', 'C']}}
Chains are in other as the appear in the file.

Warning: take only ATOM and HETATM lines.

### 2.4 fetch

Example:

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

```python
rna_tools.rna_tools_lib.fetch(pdb_id, path=')
fetch pdb file from RCSB.org https://files.rcsb.org/download/1Y26.pdb
Returns: a path to a file
```

### 2.5 fetch Biological Assembly

Example:

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

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

```
$ 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
```

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

Examples:

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

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

2.7 edit

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

Edit your structure.

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

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

Examples:

```
$ 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
```

2.8 the library

```
rna_tools_lib.py - main lib file, many tools in this lib is using this file.
class rna_tools.rna_tools_lib.RNAStructure(fn)
```

RNAStructure - handles an RNA pdb file.

```
fn
  string – path to the structural file, e.g., “../rna_tools/input/4ts2.pdb”

name
  string – filename of the structural file, “4ts2.pdb”

lines
  list – the PDB file is loaded and ATOM/HETATM/TER/END go to self.lines

edit_occupancy_of_pdb(txt, pdb, pdb_out, v=False)
  Make all atoms 1 (flexi) and then set occupancy 0 for seletected atoms. Return False if error. True if OK

fix_O_in_UC()

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

fix_with_qrnas(outfn=”, verbose=False)
  Add missing heavy atom.
```
A residue is recognized base on a residue names.

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

**Warning:** QRNA required (http://genesilico.pl/QRNA/QRNA.tgz)

---

**get_all_chain_ids()**

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

*Return type* set

**get_atom_code(line)**

Get atom code from a line of a PDB file

**get_atom_coords(line)**

Get atom coordinates from a line of a PDB file

**get_atom_num(line)**

Extract atom number from a line of PDB file

:param * line = ATOM line from a PDB file:

*Output:*

- atom number (int)

**get_info_chains()**

return A:3-21 B:22-32

**get_remarks_text()**

Get remarks as text for given file. This function re-open files, as define as self.fn to get remarks.

Example:

```
r = RNAstructure(fout)
remarks = r.get_remarks_txt()
r1 = r.get_res_txt(‘A’, 1)
r2 = r.get_res_txt(‘A’, 2)
r3 = r.get_res_txt(‘A’, 3)
with open(fout, ‘w’) as f:
    f.write(remarks)
    f.write(r1)
    f.write(r2)
    f.write(r3)

remarks is

REMARK 250 Model edited with rna-tools
REMARK 250 ver 3.5.4+63.g4338516.dirty
REMARK 250 https://github.com/mmagnus/rna-tools
REMARK 250 Fri Nov 13 10:15:19 2020
```

**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_res_text** *(chain_id, resi)*

Get a residue of given resi of chain_id and return as a text

**Parameters**

- **chain_id** *(str)* – e.g., 'A'
- **resi** *(int)* – e.g., 1

**Returns**

**Return type** *txt*

**Example:**

```python
r = RNAStructure(fn)
print(r.get_res_txt('A', 1))
```

```
ATOM 1 O5' G A 1  78.080 -14.909 -0.104 1.00 9.24
   → O
ATOM 2 C5' G A 1  79.070 -15.499 -0.956 1.00 9.70
   → C
ATOM 3 C4' G A 1  78.597 -16.765 -1.648 1.00 9.64
   → C
ATOM 4 O4' G A 1  78.180 -17.761 -0.672 1.00 9.88
   → O
(...)
```

**get_rnapuzzle_ready** *(renumber_residues=**True**, fix_missing_atoms=**True**, rename_chains=**True**, report_missing_atoms=**True**, keep_hetatm=False, backbone_only=False, no_backbone=False, bases_only=False, verbose=False)*

Get rnapuzzle (SimRNA) ready structure.

Clean up a structure, get current order of atoms.

**Parameters**

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

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

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

Figure: (Starting from left) input structure, structure with rebuilt atoms, and reference. The B fragment is observed in the reference used here as a “benchmark”, fragment A is reconstructed.
atoms (not observed in the reference”). 201122
- 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\(\text{(}\text{compact=}False, \text{chainfirst=}False, \text{fasta=}False, \text{addfn=}", \text{color=}False)\)

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

Run:

```bash
python rna_pdb_seq.py input/lykq_clx.pdb
> lykq_clx A:101-111
GGAGCUCGCC
> lykq_clx B:201-238
GGCCAGGCCGUGCCAGUCUCUUCGGAGCAAUACUGC
> 6_solution_0 A:1-19 26-113 117-172
GGCCAGGUGUCUCCCGAGUCGGAGUUAAAAGGAAG
```

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\(\text{(}\text{add_end=}True)\)

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 (verbose)
    Delete file, self.fn

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

remove_water()
    Remove HOH and TIP3

rename_chain (chain_id_old, chain_id_new, debug=False)
    Rename chains
    Parameters
    • chain_id_old (str) – e.g., A
    • chain_id_new (str) – e.g., B
    • debug (bool) – show some diagnostics
    Returns pdb content (txt) self.lines is updated with new lines

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

set_atom_occupancy (line, occupancy)
    set occupancy for line

set_occupancy_atoms (occupancy)
    Parameters occupancy –

set_res_code (line, code)
    Parameters
    • lines –
    • code –
    path (str): The path of the file to wrap field_storage (FileStorage): The :class:Y instance to wrap temporary (bool): Whether or not to delete the file when the File instance is destructed
    Returns A buffered writable file descriptor
    Return type BufferedFileStorage

std_resn()
    ‘Fix’ residue names which means to change them to standard, e.g. RA5 -> A
    Works on self.lines, and returns the result to self.lines.
    Will change things like:

# 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 5 501.633 506.561 506.295 1.00 0.00
un_nmr (startwith1=True, verbose=False)
Un NMR - Split NMR-style multiple model pdb files into individual models.
Take self.fn and create new files in the way:

```
input/la91_NMR_1_2_models.pdb
input/la91_NMR_1_2_models_0.pdb
input/la91_NMR_1_2_models_1.pdb
```

**Warning:** This function requires biopython.

write (outfn=", verbose=True)
Write `self.lines` to a file (and add END file)

**Parameters**
- **outfn (str)** – file to save, if outfn is "," then simply use self.fn
- **verbose (Boolean)** – be verbose or not

**Returns** None

rna_tools.rna_tools_lib.aa3to1(aaa)
based on https://pymolwiki.org/index.php/Aa_codes

rna_tools.rna_tools_lib.collapsed_view(args)
Collapsed view of pdb file. Only lines with C5' atoms are shown and TER, MODEL, END.

**example:**

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

rna_tools.rna_tools_lib.edit_pdb(f, args)
Edit your structure.

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

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

**Examples:**

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

or even:
```bash
→pdb > 1f27_clean_renumb.pdb
```

rna_tools.rna_tools_lib.fetch(pdb_id, path='')
fetch pdb file from RCSB.org https://files.rcsb.org/download/1Y26.pdb
Returns: a path to a file

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

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

rna_tools.rna_tools_lib.fetch_cif_ba(cif_id, path='')
fetch biological assembly cif file from RCSB.org

rna_tools.rna_tools_lib.load_rnas(path, verbose=True)
Load structural files (via glob) and return a list of RNAStructure objects.
Examples:

```python
rnas = rtl.load_rnas('../rna_tools/input/mq/*.pdb')
```

rna_tools.rna_tools_lib.replace_chain(struc_fn, insert_fn, chain_id)
Replace chain of the main file (struc_fn) with some new chain (insert_fn) of given chain id.

Parameters

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

Returns text in the PDB format

Return type string

rna_tools.rna_tools_lib.set_chain_for_struc(struc_fn, chain_id, save_file_inplace=False, skip_ter=True)
Quick & dirty function to set open a fn PDB format, set chain_id and save it to a file. Takes only lines with ATOM and TER.

rna_tools.rna_tools_lib.sort_strings(l)
Sort the given list in the way that humans expect. http://blog.codinghorror.com/sorting-for-humans-natural-sort-order/

### 2.9 PDB Edit Bfactor/Occupancy

rna_pdb_edit_occupancy_bfactor.py - edit occupancy or bfactor in PDB file.
Example:

```python
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
```
Change occupancy or bfactor of pdb file.

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

**Parameters**

- `txt` *(str)* – A:1-10, selection, what to change
- `pdb` *(str)* – filename to read as an input
- `pdb_out` *(str)* – filename to save an output

**Usage:**

```
rna_pdb_edit_occupancy_bfactor.py --occupancy \ 
   --select A:1-2 \ 
   --select-atoms P+C4\' \ 
   --set-to 10 \ 
   -o test_data/3w3s_homologymodel_out.PD 
   --set-not-selected-to 8 
   test_data/3w3s_homologymodel.pdb
```
• **bfactor** *(bool)* – if edit bfactor
• **occupancy** *(bool)* – if edit occupancy
• **set_to** *(float)* – set to this value, if within selection
• **set_not_selected_to** *(float)* – set to this value, if not within selection
• **select_atoms** *(str)* – P, P+C4’, use + as a separator
• **v** *(bool)* – be verbose

**Returns** if OK, save an output to pdb_out

**Return type**  *bool*

**Warning:** this function requires BioPython

### 2.10 Add chain to a file

**Example:**

```bash
./rna_add_chain.py -c X ../../input/1msy_rnakbmd_decoy999_clx_noChain.pdb > ../../
‐‐‐output/1msy_rnakbmd_decoy999_clx_noChain_Xchain.pdb
```

<table>
<thead>
<tr>
<th>From:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ATOM</td>
<td>1</td>
<td>O5’</td>
<td>U</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>C5’</td>
<td>U</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>C4’</td>
<td>U</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>O4’</td>
<td>U</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>C1’</td>
<td>U</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>N1</td>
<td>U</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>C6</td>
<td>U</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>C5</td>
<td>U</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>C4</td>
<td>U</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>O4</td>
<td>U</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>to:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ATOM</td>
<td>1</td>
<td>O5’</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>C5’</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>C4’</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>O4’</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>C1’</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>N1</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>C6</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>C5</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>C4</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>O4</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

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

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

2.11 Measure distance between atoms

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

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

```
usage: pdbs_measure_atom_dists [-h] [-v] seqid1 seqid2 alignfn pdbfn1 pdbfn2
```

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

Options:
- `-v=False, --verbose=False` increase output verbosity

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

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

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

Parameters

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

Usage:

```
>>> find_core('GUUCAG-------------------UGAC', 'CUUCGCAGCAUUGCAUCGGGCUGGAUG')
'xxxxxx-------------------xxxx-'
```

Returns core="xxxxxx———-xxxx-"

```
rna_tools.tools.pdbs_measure_atom_dists.pdbs_measure_atom_dists.get_parser()
```

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

Get seq from an alignment with gaps.

Parameters

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

2.11. Measure distance between atoms
Usage:

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

**Returns** SeqRecord

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

**Args:** structure (pdb): PDB Biopython object: with a pdb structure

**Returns** a list of coords for atoms structure

**Return type** struct1dict

```python
rna_tools.tools.pdbs_measure_atom_dists.pdbs_measure_atom_dists.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
3.1 RNA Sequence

RNA Sequence with secondary structure prediction methods.

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

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

3.1.1 Installation

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

3.1.1.1 ContextFold

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

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

3.1.1.2 ViennaRNA

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

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

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

If one encounters a problem:

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

the solution is:

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

3.1.1.4 RNA Structure

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


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

3.1.1.5 MC-Sym

3.1.2 FAQ

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

3.1.3 TIPS

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

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

34 Chapter 3. RNA Tools
Todo:

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

```
exception rna_tools.Seq.MethodNotChosen

class rna_tools.Seq.RNASequence(seq, ss='', name='rna_seq')
RNASequence.

Usage:

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

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

Parameters

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

Returns Energy (float)

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

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

An equivalent RNAeval command line call would have been:

```
RNAeval -v -d0 < input.txt
```

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

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

Steps:

- parse SS into basepairs,
- calculate foldability

Configuration:

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

Cmd wrapper in here:

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

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

Predict secondary structure of the seq.

**Parameters**

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

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

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

Methods that can be used with SHAPE contraints: RNAfold.

**ContextFold**

Example:

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

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

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

**RNAstructure**

Example:

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

and with the shape data:

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

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

**MC-Fold**

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

```plaintext
RNA Sequence
acucgcuaggcaguauauagccgucagccgcuagccucagcucaccccccccucugggugcagggcgaagggucg
(((............)))............((((((((((((((....)))))))))))))))))))))))))

```
explore

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


**3.2 RNA Secondary Structure**

Secondary structure analysis

```
exception rna_tools.SecondaryStructure.ExceptionOpenPairsProblem
```

```
rna_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:

```
>>> seq = 'GGAAACC'
>>> ss = '((...))'
>>> img_out = 'output/demo.png'
>>> draw_ss('rna', seq, ss, img_out)
>>> print('Made %s' % img_out)
Made output/demo.png
```
Can be used with http://geekbook.readthedocs.io/en/latest/rna.html

**rna_tools.SecondaryStructure.parse_vienna_to_pairs** *(ss, remove_gaps_in_ss=False)*

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

**Parameters**

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

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

```text
Traceback (most recent call last):
  File "/usr/lib/python2.7/doctest.py", line 1315, in __run
    compileflags, 1)
  in test.globs
File "<doctest __main__.parse_vienna_to_pairs[4]>", line 1, in <module>
```
3.2.1 rna_dot2ct.py

The output file is <input-file>.ct


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

Positional arguments:

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

Options:

  -v=False, --verbose=False  be verbose

3.2.2 Secondary structure format conversion

rna_convert_pseudoknot_formats

Run this as:

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

Convert:

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

to:

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

and:

>2 chains
(((....))).........
.....(((((............)))..{{. ...).....)}}.

and:

>2 chains
(((([[[[[)))).........
.....(((((............)))..{{. ...).....}}.

and:
to:

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

and it works with VARNA:

![VARNA GUI](varna_gui.png)

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

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

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

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

3.3.1 Blast PDB

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

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

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

Usage:

```python
>>> p = BlastPDB('GGGUCAGGCCGGCGAAAGUCGCCACAGUUUGGGGAAAGCUGUGCAGCCUGUAACCCCCCCACGAAAGUGGG')
>>> p.search()
>>> p.result
u'\n<TITLE>BLAST Search Results</TITLE>
...'
```

Parameters seq – string

search()
Search online the seq.

3.3.2 Rfam Search

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

class rna_tools.RfamSearch.RfamSearch
RfamSearch (local).

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

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

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

Setup:

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

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


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

Usage:

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

**Parameters** `seq` – string

**Returns** result

**Return type** string

**exception** `rna_tools.RfamSearch.RfamSearchError`

### 3.4 RNA Alignment

RNAalignment - a module to work with RNA sequence alignments.

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

Load an alignment in the Stockholm:

```python
alignment = ra.RNAalignment('test_data/RF00167.stockholm.sto')
or fasta format::

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

Parameters of the alignment:

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

Consensus SS:

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

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

### 3.4.1 RNASeq

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

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

**seq_no_gaps**

```python
str = seq.replace('-', '')
```

**ss_no_gaps**

```python
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
UCGGGGUGCCCCUCUGCUGUGG-----------------------------AAGGC-
UGAGAAUUACCGUG------------------------AUCACCUG-AUCUGGAU-AAUGC
XXXXXXXXXXXXXXXXXGXXGXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX-
XXXXXXXXXCUGAGAXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX------------------XXXXXXXX-
XXXXXXXX-XCUGAGAXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX------------------XXXXXXXX-
XXXXXXXX-ACXUG

get_distance_to(nseq)
Get distance of self.seq to nseq.

get_ss_std()

remove_columns(to_remove)
indexing from 0

remove_gaps(check_bps=True, only_canonical=True, allow_gu=True)
Remove gaps from seq and secondary structure of the seq.

Parameters

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

A residue “paired” with a gap.
paired with any residues (in the blue circle). If yes, then this residues is unpair (in this case $\triangleright$ $\triangleright$).
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.GcgG.cc.u.aUACAuACCC.GaAA.GGGAAUAggCc.gGc.gu.......CU.......
→uuugucgGUuUucAgCccCCgcCcCaCCcuuuu
((((((((....((.((...........(..............))))....)))))))))))......
```

→ ...))))........)))))........

gaps will be remove as well.

**ss_to_bps()**
Convert secondary structure into a list of basepairs.

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

**Return type** bps (list)

### 3.4.2 RNAalignment

**class** rna_tools.tools.rna_alignment.rna_alignment.RNAAlignment *(fn="", fetch="")*
RNA alignment - adapter class around BioPython to do RNA alignment stuff

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

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

**Parameters**
• **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 alignment, `-.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.

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

**Parameters**

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

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

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

**Parameters**

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

```python
seq = "ggaucgcugaacccgaaaggggcgggggacccagaaauggggcgaaucucuuccgaaaggaagaguaggguuacuccuucgacccgagcccgucagcuaaccucgc当地ccgcaagcguccgaaggagaauc"

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

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

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

**Parameters**

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

```python
find_seq_exact("seq, verbose=False")
```

**format_annotation** (*t*)

**get_clean_ss** (*ss*)

**get_distances** ()

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

With BioPython.

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

get_seq(seq_id)
get_seq_ss(seq_id)
get_seq_with_name(seq_name)
get_shift_seq_in_align()
RF_cons vs ‘=#GC RF’ ???

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

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

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

UAU-AACAUAUAAUUGACAAUUG-GUCAUA-GUUUCUACCGGAAUACC-GUAAAUUUCU—GACUAUG-
UAUA- (((.((.(((.(((((.))))))))).)))))))))))))))))))))))))))))))))))))))))))

get_the_closest_seq_to_ref_seq(verbos=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
---ACCUUGCGCGACUGGCGAAUCC-------AAU CP001644.1/756294-756165
--GCCUCUGCGCGACUGGCGACUUUG------------------...GAA CU234118.1/352539-352459
UGAGUUUUCUGCGACUGACGGAUUA--------------------...CUG BAAV01000055.1/2897-2982
GCCCGUUCUGCGACUGGCGCUAGU-------------------...CGA CP000927.1/5164264-5164343
-----GGGUCGUGACUUGCGCAACA--------------------...--- zmp
UCACCCCUGCGACUUGCGCGAUA-----------------------...GUU AP009385.1/718103-718202
```  

```python
>>> a.remove_empty_columns()
>>> print(a)
SingleLetterAlphabet() alignment with 6 rows and 138 columns
---ACCUUGCGCGACUGGCGAAUCC-UGAAGCUGCUUUG-AGCG...AAU CP001644.1/756294-756165
--GCCUCUGCGCGACUGGCGACUUUG------------------...GAA CU234118.1/352539-352459
UGAGUUUUCUGCGACUGACGGAUUA--------------------...CUG BAAV01000055.1/2897-2982
GCCCGUUCUGCGACUGGCGCUAGU-------------------...CGA CP000927.1/5164264-5164343
-----GGGUCGUGACUUGCGCAACA--------G-----------...--- zmp
UCACCCCUGCGACUUGCGCGAUA--------GAACCCUCGGGUU...GUU AP009385.1/718103-718202
```

go over all seq modifes 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: (((.(((((...))))))))........(((....)))))))....)))))))
```

`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-AAAAUAGGUUUCCAGGC.. #=GC SS_cons .(.((.((((((((((. . .
#=GC RF .g.gc.aGAGUAGggugccguc.. //

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

3.4.3 rna_alignment_get_species.py

This is an improved version of the script that uses the Rfam MySQL database online interface (thanks @akaped for this idea) (so you need to be connected to the Internet, of course). Redirect the output to the file.

Warning: This scripts needs mysql-connector-python-rf module to connect the Rfam MySQL server, so install it before using: pip install mysql-connector-python-rf.

Example:

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

Examples 2:

$ rna_alignment_get_species.py u5_rfam_u5only.stk --verbose
# STOCKHOLM 1.0
#=GF WK U5_spliceosomal_RNA
#=GF NC 39.90
#=GF RT The spliceosomal snRNAs of Caenorhabditis elegans.
#=GF TC 40.00
#=GF RN [3]
(...)
#=GF AC RF00020
#=GF SE Zwieb C, The uRNA database, PMID:9016512; PMID:18390578
#=GF GA 40.00
#=GF BM cmbuild -F CM SEED
#=GF TP Gene; snRNA; splicing;
Bos-taurus-(cattle) GAUC-GUAUAAAUCUUUCGCCUUUUACUAAAGA-
˓→UUUCCG----UGG-A--GA-G
Sorex-araneus-(European-shrew) GAUC-GUAUAAAUCUUUCGCCUUUUACUAAAGA-
˓→UUUCCG----UGG-A--GA-G
Ictidomys-tridecemlineatus-(thirteen-lined-ground- GAUC-GUAUAAAUCUUUCGCCUUUUACUAAAGA-
˓→UUUCCG----UGG-A--GA-G

3.4. RNA Alignment
Monodelphis-domestica-(gray-short-tailed-opossum)  GAUC-GUAUAAAUCUUUCGCCUUUUACUAAAGA-...
Oryctolagus-cuniculus-(rabbit)  GAUC-GUAUAAAUCUUUCGCCUUUUACUAAAGA-...
Cavia-porcellus-(domestic-guinea-pig)  GAUC-GUAUAAAUCUUUCGCCUUUUACUAAAGA-...
Ochotona-princeps-(American-pika)  GAUC-GUAUAAAUCUUUCGCCUUUUACUAAAGA-...

usage: rna_alignment_get_species [-h] [-v] [--debug] [--id-width ID_WIDTH]
[--evo-mapping EVO_MAPPING]
[--evo-mapping-default] [--one] [--osfn OSFN]
[--rfam]
alignment

Positional arguments:

    alignment  alignment

Options:

    -v=False, --verbose=False  be verbose
    --debug=False  Undocumented
    --id-width=50  define width of ids, trim species name when longer than this
    --evo-mapping  Undocumented
    --evo-mapping-default=False  Undocumented
    --one=False  Undocumented
    --osfn  cache file
    --rfam=False  Undocumented

3.4.4 rna_alignment_calc_energy.py

Calculate energy (.cet) format:

UGGC-CCCUCGCCCAA--GGAUGACA
(((...((....)))...))
(((...((**))...)))

Examples:

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

Takes cet files (calc-energy-templets):

$ rna_alignment_calc_energy.py --template test_data/u6-lower.cet --one test_data/u6-
only.stk -v # --loop-seq test_data/u6-only-loop-seq-u6-lower
N/A% (0 of 182) |
| Elapsed Time: 0.00.00 ETA: |

--:--:--
by parsing output from MC-Sym:

```plaintext
Score: -999.000 GAACAUGGUUCUUGCCUUUACCAGAACCAUCCGGGGUGUGUUG
Total number of MB structures with 3 stems: 16041
(overlaps: 0, !energy: 335585)
```

```plaintext
usage: rna_alignment_calc_energy [-h] [--debug] [--one] [--method METHOD]
       alignment
```

**Positional arguments:**

- `alignment`: an alignment in the Stockholm format
**3.4.5 rna_align_get_ss_from_fasta.py**

Input as a file:

```plaintext
>ade
GCU-U-CAUAUAAUCCUAUGAUAUGG-UUUGGGA-GUUUCUACCAAGAG-CC--UUAAA-CUCUU---GAUUAUG-AAGU- 
(....(((......(((((((((((((((((((((((((.))))))))))))))))))))))))....)))))))))))))))))
```
to get:

```plaintext
>ade
GCUUCAUUAUAAUCCUAUGAUAUGGUUGGAGGUUUUCUACCAAGAGCCUUAACUCUGAUAAUGAAGU 
(((((((((((((.......))))))))........(((((.......))))))...))))))))))))))))))
```

Usage: rna_align_get_ss_from_fasta [-h] file

Positional arguments:

- file subsection of an alignment

**3.4.6 rna_align_get_ss_from_stk.py**

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

Example:

```plaintext
$ rna_align_get_ss_from_stk.py aligns/gmp_RF01786.stockholm_p75_pk.sto
AAC00000007.1/31274-31356
AAGAAUUGAACACUGUGAGCGUGUUUUUUCUAAACCCUGUGAGACUGACGGGAGGCCGCGCCACAAACUAUCUA 
.(((....(((....(((((((((((((((((((((((((((((((((.)))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
CP000724.1/3560727-3560809
AAAAAUGUGACGCAAUGUCAUGAGGUGGACCCUUAAUCUAGGAGUGUGCGGUAACCCCAUACAAUU 
((...(((((.......(((((((((((((((((.)))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
```

Usage: rna_align_get_ss_from_stk [-h] file

Positional arguments:
3.4.7 rna_align_distance_to_seq.py

Calculate

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

Example:

```bash
$ rna_align_distance_to_seq.py test_data/gmp_ref.sto test_data/gmp_ref_distance.csv
```

<table>
<thead>
<tr>
<th>distance</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.00 gmp</td>
</tr>
<tr>
<td>1</td>
<td>0.69 AE000513.1/1919839-1919923</td>
</tr>
<tr>
<td>2</td>
<td>0.73 BA000004.3/387918-388001</td>
</tr>
<tr>
<td>3</td>
<td>0.69 ABFD02000011.1/154500-154585</td>
</tr>
<tr>
<td>4</td>
<td>0.73 AE015927.1/474745-474827</td>
</tr>
<tr>
<td>5</td>
<td>0.75 AAWL01000006.1</td>
</tr>
<tr>
<td>6</td>
<td>0.72 AM180355.1</td>
</tr>
<tr>
<td>7</td>
<td>0.72 CP001116.1/102374-102457</td>
</tr>
<tr>
<td>8</td>
<td>0.65 AJ965256.1/1260708-1260792</td>
</tr>
</tbody>
</table>

```python
seq
0 -----GCGCGGAAC--AAUGAUAAU--GGG--UUUA--AAUGGGC--...
1 CUGUCGAGAGAGC--GAUGAUAUC--GCC--CUGUAUUCGGCC--...
2 AAUCAUAUGGAGAC--AACGAAGACAU--AGC--CUUU--AAUGGAC--...
3 AAAUAUUAUGGAGAUAU--GUUGAAGUAU--AUU--CUAAUA--UUGGCC--...
4 AUUUAAGAGGAUAA--UUUGAACUAAU--AUA--CUU--AAUUGGCC--...
5 --UGCA--UGGUGU--GAUGAGUCC--GGA--CAGUAUGUGGCC--...
6 AUAUUU--UGAAGAC--UGAGAAGUAU--AUC--UAUAUA--UUGGCC--...
7 AUAACCGGCAGAG--AAUGAAUGU--UCG--AUGU--AACCGGGCC--...
8 AAAUUAAGGGGAAGC--GUUGAGCGGC--UAC--CAAU--AUGUGUUC--...
```

```python
ss:
0 (((((((((((.((................((((..(((.(................
1 (((((((((((.((................((((..(((.(................
2 (((((((((((.((................((((..(((.(................
3 (((((((((((.((................((((..(((.(................
4 (((((((((((.((................((((..(((.(................
5 (((((((((((.((................((((..(((.(................
6 (((((((((((.((................((((..(((.(................
7 (((((((((((.((................((((..(((.(................
8 (((((((((((.((................((((..(((.(................
```

usage: rna_align_distance_to_seq.py [-h] file output

Positional arguments:

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

3.4.8 rna_align_foldability.py

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>accuracy</td>
<td>1.000000</td>
</tr>
<tr>
<td>cfe</td>
<td>0.653813</td>
</tr>
<tr>
<td>foldability</td>
<td>0.622038</td>
</tr>
<tr>
<td>mfe</td>
<td>0.607340</td>
</tr>
<tr>
<td>efe</td>
<td>0.585077</td>
</tr>
<tr>
<td>diversity</td>
<td>0.404350</td>
</tr>
<tr>
<td>eval</td>
<td>0.349499</td>
</tr>
<tr>
<td>cfe_enforce</td>
<td>0.311744</td>
</tr>
<tr>
<td>mfe_enforce</td>
<td>0.302973</td>
</tr>
<tr>
<td>efe_enforce</td>
<td>0.280929</td>
</tr>
<tr>
<td>distance</td>
<td>0.256870</td>
</tr>
<tr>
<td>freq</td>
<td>0.037037</td>
</tr>
<tr>
<td>diversity_enforce</td>
<td>0.018429</td>
</tr>
<tr>
<td>mcsym</td>
<td>0.017533</td>
</tr>
<tr>
<td>freq_enforce</td>
<td>-0.037991</td>
</tr>
<tr>
<td>length</td>
<td>-0.340809</td>
</tr>
</tbody>
</table>

The data:

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

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


Example:

```
$ python rna_align_foldability.py test_data/gmp_ref.sto test_data/gmp_foldability.csv
cfess_enforce distance diversity
cfess_enforce 1.00 3.96
distance 0.69 5.56
0.73 3.84
0.69 5.92
0.73 7.49
0.75 7.92
0.72 5.83
0.72 7.35
0.65 4.86
diversity_enforce efe efe_enforce
diversity_enforce efe efe_enforce
diversity_enforce efe efe_enforce
diversity_enforce efe efe_enforce
diversity_enforce efe efe_enforce
```

3.4. RNA Alignment
### efess

<table>
<thead>
<tr>
<th>length</th>
<th>mcsym</th>
</tr>
</thead>
<tbody>
<tr>
<td>75.0</td>
<td>-39.73</td>
</tr>
<tr>
<td>85.0</td>
<td>-37.89</td>
</tr>
<tr>
<td>84.0</td>
<td>-35.40</td>
</tr>
<tr>
<td>86.0</td>
<td>-36.11</td>
</tr>
<tr>
<td>83.0</td>
<td>-37.37</td>
</tr>
<tr>
<td>80.0</td>
<td>-36.55</td>
</tr>
<tr>
<td>84.0</td>
<td>-42.95</td>
</tr>
<tr>
<td>85.0</td>
<td>-43.58</td>
</tr>
</tbody>
</table>

### mcsym comment

<table>
<thead>
<tr>
<th>energy</th>
<th>best dynamics</th>
<th>programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP energy</td>
<td>BP energy</td>
<td>BP energy</td>
</tr>
<tr>
<td>energy best dynamics</td>
<td>programming</td>
<td>programming</td>
</tr>
<tr>
<td>energy best dynamics</td>
<td>programming</td>
<td>programming</td>
</tr>
<tr>
<td>energy best dynamics</td>
<td>programming</td>
<td>programming</td>
</tr>
<tr>
<td>BP energy</td>
<td>BP energy</td>
<td>BP energy</td>
</tr>
<tr>
<td>energy best dynamics</td>
<td>programming</td>
<td>programming</td>
</tr>
</tbody>
</table>

- mcsym ss mfe mfe_enforce

<table>
<thead>
<tr>
<th>mcsym ss</th>
<th>mfe</th>
<th>mfe_enforce</th>
</tr>
</thead>
<tbody>
<tr>
<td>-13.9</td>
<td>-12.9</td>
<td></td>
</tr>
<tr>
<td>-18.0</td>
<td>-17.3</td>
<td></td>
</tr>
<tr>
<td>-14.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>-12.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>-7.2</td>
<td>-6.1</td>
<td></td>
</tr>
<tr>
<td>-18.6</td>
<td>-18.6</td>
<td></td>
</tr>
<tr>
<td>-10.5</td>
<td>-10.5</td>
<td></td>
</tr>
<tr>
<td>-12.8</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>-19.8</td>
<td>-19.8</td>
<td></td>
</tr>
</tbody>
</table>

### mfess

<table>
<thead>
<tr>
<th>mfess_enforce</th>
</tr>
</thead>
<tbody>
<tr>
<td>-19.8</td>
</tr>
</tbody>
</table>

---

Chapter 3. RNA Tools
```python
1  ((((((((...(........((((((................)))))))))))))......
2    error
3    error
4  (((...(((....((((((((((((((................)))))))))))))))...)
5  ((((((((....((((((((((((((((((................)))))))))))))))))...)
6  ((((....((((((((((((((((((((((................)))))))))))))))))...)
7  ........((((((((((((((((((((((((((................)))))))))))))))))...)
8  ...((((((((((((((((((((((((((((((................)))))))))))))))))...)

seq
0  GCGCGGAACAAUGAUGAAUGGGUUUAUUUGGCAUCUGACUAU...
1  CUGUGAAGAGCCGCAUGAAUCCCGCUGUUUAAUUCGGAUCCUC...
2  AAUCUUAGGGGAAGCAAGGCAUGCAUAGCUUAAUUGGACACUUGG...
3  AAAUAUAUAAGGAUGUGAAGUAUUUCACUAUAAUUGGACCACUAU...
4  AAAAAAGAAGAAAUUUGGAAACUAUAUACUAUUUUUGGCAUUUGU...
5  UGCAUGGUGUGAGAUGAACGAGGCAACAGUAAUGGGCACUUAAGUC...
6  AAAUUUAAAAGAAAUCAGAAGUAUCAUACUUAUUAAUUGGCAUCUGGA...
7  AAUAAACCAAGGAAUGGGAUUGGCAUGUAACCGGGACCUUAAU...
8  AAAUAAGGGGAAGGUUAGCCCGGUACACCAAUAAUGGGUUCACUGG...

ss
0  ((((((((...(((((((((................)))))))))...)...
1  ((((((....(((((((((................)))))))...)
2  .......((((((((((((((................)))))))))...)
3  ..((((((((((((((((................)))))))))))...)
4  (((((....((((((((((((((((................)))))))))))...)
5  ((((((((...((((((((((((................))))))))))))...)
6  ..(((((((((....(((((((((................))))))))))))...)
7  .......(((((((((......((((((((................)))))))))...)
8  ...(((((((((......(((((((((................)))))))))...)

[9 rows x 26 columns]
```

        file output

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

output csv pandas file

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

3.4.9 Random assignment of nucleotides

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

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

Usage:
```
random_assignment_of_nucleotides [-h] [-v] [--alignfn ALIGNFN]
                                  [--seqfn SEQFN] [--outfn OUTFN]
```

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

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

Author: A. Zyla - azyla

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

Get seq from an alignment with gaps.

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

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

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

Returns: alignment

```python
rna_tools.tools.rna_alignment.random_assignment_of_nucleotides.get_parser()
```

```python
rna_tools.tools.rna_alignment.random_assignment_of_nucleotides.get_sequences(seqfn)
    Get seq from an fasta file. :param seqfn: a path to a fasta file :type seqfn: str
```

Usage::

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

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

```python
rna_tools.tools.rna_alignment.random_assignment_of_nucleotides.write_align(align, outfn)
    Write cleaned alignment with Biopython. :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: none
```

```python
rna_tools.tools.rna_alignment.random_assignment_of_nucleotides.write_seq(seqfn, outfn)
    Write cleaned alignment with Biopython. :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: none
```

### 3.4.10 CMAAlign

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

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

Example:

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

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

### 3.4. RNA Alignment

63
Install [http://eddylab.org/infernal/](http://eddylab.org/infernal/)


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

Variables self.output – string

get_seq()
Variables self.output – output of cmalign, string

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

Parameters

- seq – seq string
- cm – cm fn

Run:

```
$ cmalign RF01831.cm 4lvv.seq
# STOCKHOLM 1.0
#=GF AU Infernal 1.1.2

4lvv

```

''
Warning: requires cmalign to be set in your shell

3.4.11 RChie

class rna_tools.tools.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:

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

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

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

```
<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
def 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 pseudoknots.
```

### 3.4.12 Renumber a pdb file according to alignment

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

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

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

```
```

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

**Options:**
- `--residue_index_start=1` renumber starting number (default: 1)
- `--outfn` output pdb file (default: pdbfn .pdb -> _out.pdb)

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

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

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

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

Get seq from an alignment with gaps.

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

Usage:

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

Returns SeqRecord

```
rna_tools.tools.renum_pdb_to_aln.renum_pdb_to_aln.open_pdb(pdbfn)
```

Open pdb with Biopython.

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

Returns with a pdb structure

Return type PDB Biopython object

```
rna_tools.tools.renum_pdb_to_aln.renum_pdb_to_aln.renumber(seq_with_gaps, struc, residue_index_start)
```

Renumber a pdb file.

Parameters

• **seq_with_gaps** *(str)* – a target sequence extracted from the alignment

• **struc** *(pdb)* – a structure

• **residue_index_start** *(int)* – starting number

Returns BioPython Structure object

```
rna_tools.tools.renum_pdb_to_aln.renum_pdb_to_aln.renum_pdb_to_aln.write_struc(struc, outfn)
```

Write renumbered pdb with Biopython.

Parameters

• **struc** *(pdb)* – a renumbered structure

• **outfn** *(str)* – a path to a new, renumbered pdb file

Returns writes to a file

Return type none

### 3.5 Root Mean Square Deviation (RMSD)

#### 3.5.1 rna_calc_rmsd

rna_calc_rmsd.py - calculate RMSDs of structures to the target

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

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

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

```
target_selection: A:1-48+52-63
model_selection: A:1-48+52-63
target_ignore_selection: A/57/O2'
model_ignore_selection:
# of models: 801
fn,rmsd_all
pistol_thrs0.50A_clust01-000001_AA.pdb,7.596
pistol_thrs0.50A_clust02-000001_AA.pdb,7.766
pistol_thrs0.50A_clust03-000001_AA.pdb,18.171
[...]
pistol_thrs0.50A_clust799-000001_AA.pdb,5.356
pistol_thrs0.50A_clust800-000001_AA.pdb,15.282
pistol_thrs0.50A_clust801-000001_AA.pdb,16.339
# of atoms used: 1237
csv was created! rmsds.csv
rmsd_calc_to_target.py -t 5k7c_clean_onechain_renumber_as_puzzle_srr.pdb
37.93s user 1.07s system 87% cpu 44.650 total
```

```
usage: rna_calc_rmsd [-h] [-t TARGET_FN] [--ignore-files IGNORE_FILES]
   [--target-column-name]
files [files ...]
```
Positional arguments:
files          files

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

rna_calc_rmsd.py - calculate RMSDs of structures to the target

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

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

Example #1:

```
$ 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
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:

```

3.5. Root Mean Square Deviation (RMSD)
time rmsd_calc_to_target.py
-t 5k7c_clean_onechain_renumber_as_puzzle_srr.pdb
--target-selection A:1-48+52-63
--model-selection A:1-48+52-63
--target-ignore-selection A/57/02`
clusters/*/AA.pdb

rmsd_calc_rmsd_to_target
-----------------------------------------------------------------------------------------------------------------------------------
target_selection: A:1-48+52-63
model_selection: A:1-48+52-63
target_ignore_selection: A/57/02`
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

Calculate RMSD between two XYZ files
by: Jimmy Charnley Kromann <jimmy@charnley.dk> and Lars Andersen Bratholm <lars-bratholm@gmail.com> project: https://github.com/charnley/rmsd license: https://github.com/charnley/rmsd/blob/master/LICENSE

a is model b is target

Params a = filename of structure a
Params b = filename of structure b

Returns rmsd, number of atoms

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

See:
• Align: <http://www.pymolwiki.org/index.php/Align>
• 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
**3.5. Root Mean Square Deviation (RMSD)**

- 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 brewsci/bio/pymol` or get

If you have a problem:

```bash
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 find BLOSUM62, e.g.:

```bash
mdfind -name BLOSUM62 | grep pymol
```

and then define `PYMOL_DATA` in your `.bashrc/.zshrc`, e.g.:

```bash
export PYMOL_DATA="/Users/magnus/work/opt/pymol-open-source/data/pymol/matrices/BLOSUM62"
```

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

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

**Parameters**

- `models` – a list of filenames

Example of the list:

```python
['test_data/rp17/2_restr1_Michal1.pdb_clean.pdb', 'test_data/rp17/2a_nonrestr2_Michal1.pdb_clean.pdb', 'test_data/rp17/3_nonrestr1_Michal1.pdb_clean.pdb', 'test_data/rp17/5_restr1_Michal3.pdb_clean.pdb']
```

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

Sort the given list in the way that humans expect.

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

### 3.5.2 rna_calc_rmsd_multi_targets

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

```
$ rna_calc_rmsd_multi_targets.py --models multi_targets/rp21/*pdb
   --targets multi_targets/rp21/solutions/*pdb
   --target-selection A:1-27+29-41
   --model-selection A:1-27+29-41
```
CSV table produced:

<table>
<thead>
<tr>
<th>PDB Name</th>
<th>RMSD 12.17</th>
<th>RMSD 12.11</th>
<th>RMSD 12.13</th>
<th>RMSD 12.11</th>
<th>RMSD 12.12</th>
<th>RMSD 12.17</th>
<th>RMSD 0.03</th>
</tr>
</thead>
<tbody>
<tr>
<td>21_solution_0_ChainA.pdb</td>
<td>12.17</td>
<td>12.11</td>
<td>12.13</td>
<td>12.11</td>
<td>12.12</td>
<td>12.17</td>
<td>0.03</td>
</tr>
<tr>
<td>21_solution_0_ChainB.pdb</td>
<td>21_solution_1_ChainA.pdb</td>
<td>21_solution_1_ChainB.pdb</td>
<td>21_solution_2.pdb</td>
<td>mean</td>
<td>min</td>
<td>max</td>
<td>sd</td>
</tr>
<tr>
<td>21_3dRNA_1_rpr.pdb</td>
<td>12.17</td>
<td>12.11</td>
<td>12.13</td>
<td>12.11</td>
<td>12.12</td>
<td>12.17</td>
<td>0.03</td>
</tr>
<tr>
<td>21_Adamiak_1_rpr.pdb</td>
<td>4.64</td>
<td>4.61</td>
<td>4.64</td>
<td>4.63</td>
<td>4.61</td>
<td>4.64</td>
<td>0.01</td>
</tr>
<tr>
<td>21_ChenHighLig_1_rpr.pdb</td>
<td>4.01</td>
<td>3.97</td>
<td>4.07</td>
<td>4.01</td>
<td>3.97</td>
<td>4.07</td>
<td>0.04</td>
</tr>
<tr>
<td>21_Das_1_rpr.pdb</td>
<td>5.71</td>
<td>5.60</td>
<td>5.65</td>
<td>5.60</td>
<td>5.71</td>
<td>5.71</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Save rna_calc_rmsd_multi_targets_output.csv

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

Options:
-v=False, --verbose=False  be verbose
--models  Undocumented
--targets  Undocumented
--output-csv=rna_calc_rmsd_multi_targets_output.csv  Undocumented
--model-selection= selection, e.g. A:10-16+20, where #16 residue is included
--target-selection= selection, e.g. A:10-16+20, where #16 residue is included

3.5.3 rna_calc_rmsd_trafl

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

After this script, run:
to get a plot like this:

Prepare structures:

```
$ SimRNA -p 17_Das_2_rpr.pdb -n 0 -o 17_Das_2_rpr_n0 # no trafi, trafi 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_Trafi.py 17_Das_2_rpr.pdb.trafi 17_Das_2_rpr_n0.trafi 5k7c_clean_onechain_renumber_as_puzzle_rpr_rmGapped_n0.trafi rp17_rmsd.txt
> calc_rmsd_to_lst_frame
  /Users/magnus/work/opt/simrna/SimRNA_64bitIntel_MacOSX_staticLibs/calc_rmsd_to_lst_frame 17_Das_2_rpr.pdb.trafi 17_Das_2_rpr.n0.pdb.rmsd_e
  < rmsd_out: 17_Das_2_rpr.pdb_rmsd_e
  > struc: 17_Das_2_rpr_n0.trafi 2
  > trafi: 17_Das_2_rpr.pdb.trafi 48
  % saved: 17_Das_2_rpr.pdb.trafi_17_Das_2_rpr_n0.trafi
> calc_rmsd_to_lst_frame
  /Users/magnus/work/opt/simrna/SimRNA_64bitIntel_MacOSX_staticLibs/calc_rmsd_to_lst_frame 17_Das_2_rpr.pdb.trafi 17_Das_2_rpr_n0.pdb.rmsd_e 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.trafi 2
  > trafi: 17_Das_2_rpr.pdb.trafi 48
  % saved: 17_Das_2_rpr.pdb.trafi_5k7c_clean_onechain_renumber_as_puzzle_rpr_rmGapped_n0.trafi
> calc_rmsd_to_lst_frame
  /Users/magnus/work/opt/simrna/SimRNA_64bitIntel_MacOSX_staticLibs/calc_rmsd_to_lst_frame 17_Das_2_rpr.pdb.trafi_17_Das_2_rpr_n0.pdb.rmsd_e_17_Das_2_rpr.pdb_rmsd_e_17_Das_2_rpr_n0_rmsd_e
```

### 3.5. Root Mean Square Deviation (RMSD)

---

[Diagram of RNA structure]
usage: rna_calc_evo_rmsd [-h] trafl struc1 struc2 rmsds_fn

Positional arguments:

    trafl       trafil
    struc1      structure A
    struc2      structure B
    rmsds_fn    output file

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

usage: rna_cal_rmsd_trafl_plot [-h] file

Positional arguments:

    file         rmsd.txt

3.5.4 rna_calc_rmsd_all_vs_all

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

Examples:

rna_calc_rmsd_all_vs_all.py -i test_data -o test_output/rmsd_calc_dir.tsv
    # of models: 4
    ... 1 test_data/struc1.pdb
    ... 2 test_data/struc2.pdb
    ... 3 test_data/struc3.pdb
    ... 4 test_data/struc4.pdb

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

You can also use PyMOL to do align or fit:

python rna_calc_rmsd_all_vs_all.py -i test_data -o test_output/rmsd_calc_dir_align.
    -m mat -m align
    # of models: 5
    # test_data/2nd_triplex_FB_1AUA3_rpr.pdb test_data/struc1.pdb test_data/struc2.pdb
    test_data/struc3.pdb test_data/struc4.pdb
0.0 4.13 4.922 4.358 4.368
4.13 0.0 11.092 4.707 3.46
4.922 11.092 0.0 11.609 11.785
4.358 4.707 11.609 0.0 2.759
4.368 3.46 11.785 2.759 0.0
matrix was created! test

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

Options:
- i=, --input-dir= input folder with structures
- o=matrix.txt, --matrix-fn=matrix.txt output, matrix
- m=all-atom, --method=all-atom all-atom, pymol: align, fit

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

Examples:

```
rna_calc_rmsd_all_vs_all.py -i test_data -o test_output/rmsd_calc_dir.tsv
# of models: 4
... 1 test_data/struc1.pdb
... 2 test_data/struc2.pdb
... 3 test_data/struc3.pdb
... 4 test_data/struc4.pdb
```

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

You can also use PyMOL to do align or fit:

```
python rna_calc_rmsd_all_vs_all.py -i test_data -o test_output/rmsd_calc_dir_align.mat -m align
# of models: 5
# test_data/2nd_triplex_FB_1AUA3_rpr.pdb test_data/struc1.pdb test_data/struc2.pdb
# test_data/struc3.pdb test_data/struc4.pdb
0.0 4.13 4.922 4.358 4.368
4.13 0.0 11.092 4.707 3.46
4.922 11.092 0.0 11.609 11.785
4.358 4.707 11.609 0.0 2.759
4.368 3.46 11.785 2.759 0.0
matrix was created! test
```

```
approval: rna-rna_calc_rmsd_all_vs_all.rna_calc_rmsd_all_vs_all.calc_rmsd(a, b)
Calc rmsd.
```

```
approval: rna_rna_calc_rmsd_all_vs_all.rna_calc_rmsd_all_vs_all.get_parser()
```

```
approval: rna_rna_calc_rmsd_all_vs_all.rna_calc_rmsd_all_vs_all.get_rna_models_from_dir(directory)
```

```
approval: rna_rna_calc_rmsd_all_vs_all.rna_calc_rmsd_all_vs_all.sort_nicely(l)
Sort the given list in the way that humans expect. http://blog.codinghorror.com/sorting-for-humans-natural-sort-order/
```

3.5. Root Mean Square Deviation (RMSD)
3.6 Interaction Network Fidelity (INF)

3.6.1 rna_calc_inf

**Warning:** rna_calc_inf is using ClaRNA (included in the rna-tools packages). However, ClaRNA requires some extra libraries, not required by other tools in the package, that’s why they are not included in the main installation configuration. To install these libraries in proper version, type “pip install rna-tools simplejson networkx==1.8.1”.

```
usage: rna_calc_inf [-h] [-t TARGET_FN] [-m NT] [--ignore-files IGNORE_FILES]
                   [-s SS] [--no-stacking] [--debug] [-pr] [-sr]
                   [--method METHOD] [--target-selection TARGET_SELECTION]
                   [--model-selection MODEL_SELECTION] [--renumber-residues]
                   [--dont-remove-sel-files] [-f] [-v] [-o OUT_FN]
                   files [files ...]

Positional arguments:
files
files, e.g folder_with_pdb/*pdbs

Options:
-t=, --target_fn=           pdb file
-m=3, --number-of-threads=3 number of threads used for multiprocessing, if 1 then mp is not
                              used (useful for debugging!)
--ignore-files=             files to be ignored, e.g. ‘solution’
-s=, --ss=                 A:(([]]), works only for single chain (the chain is A by default)
--no-stacking=False        default: use stacking, if this option on, don’t take into account stacking,
                           WARNING/BUG: inf_all will be incorrectly calculated if stacking is off
--debug=False              Undocumented
-pr=False, --print-results=False Undocumented
-sr=False, --sort-results=False Undocumented
--method=clarna             you can use mcannotate* or clarna (right now only clarna is tested)
--target-selection=         selection, e.g. A:10-16+20, where #16 residue is included
--model-selection=          selection, e.g. A:10-16+20, where #16 residue is included
--renumber-residues=        renumber residues from 1 to X for comparison with selection
--dont-remove-sel-files=False don’t remove temp files created based on target|model-
                              selectionforce
-f=False, --force=False     force to run ClaRNA even if <pdb>.outCR file is there, for will be auto
                           True when selection defined
-v=False, --verbose=False   be verbose, tell me more what’re doing
-o=inf.csv, --out_fn=inf.csv out csv file, be 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 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` in `rna_calc_inf.py` to re-run ClaRNA.

```python
rna_tools.tools.rna_calc_inf.rna_calc_inf.do_job(i, method='clarna')
```

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

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

### 3.6.2 rna_calc_dinf

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

Example:

```
[mm] 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:

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

You can use `--d` to get a list of all interacting bases, something like:

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

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

Positional arguments:
    file         a PDB file

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

3.7 RNA filter (DCA)

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

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

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

[PREVIOUS DOCUMENTATION - TO BE REMOVED]

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

    [mm] rp06$ git:(master) $ rna_filter.py -s 4gxy_rpr.pdb -r rp06_MohPairs.rfrestrs d:A5-A42
    100.0 measured: 26.7465763417 [x] d:A11-A26 100.0 measured: 19.2863696104 [x]
Traceback (most recent call last):
  File "~/home/magnus/work-src/rna-pdb-tools/bin/rna_filter.py", line 270, in <module>
calc_scores_for_pdbbs(args.structures, restraints, args.verbose)
  File "~/home/magnus/work-src/rna-pdb-tools/bin/rna_filter.py", line 221, in calc_scores_for_pdbbs
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)
restraints [('A1', 'A2', '<', '10.0', '1'), ('A2', 'A1', '<=', '10', '1')]

Frame #1 e:1252.26
  mb for A1 [ 54.729 28.9375 41.421 ]
  mb for A2 [ 55.3425 35.3605 42.7455]
  d:A1-A2 6.58677550096
  mb for A2 [ 55.3425 35.3605 42.7455]
  mb for A1 [ 54.729 28.9375 41.421 ]
  d:A2-A1 6.58677550096
# this ^ is off right now

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

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

3.7.2 rna_dca_mapping.py

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

Options:
--seq seq fn in Fasta format
--gseq gapped sequence and secondary structure (like in the alignment used for DCA) in Fasta format
--dca file with parsed interactions
--offset offset
--noss=False filter out ss from plot
--mss=False ss every each line
--verbose=False be verbose
--noshort=False filter out short interactions, dist in seq < 6 nt

3.7.3 show_dists - show distances in PyMOL

show_dists - show distances in PyMOL

Usage:

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

rna_ex2x.py - analyze an evolutionary coupling file.
Files can be downloaded from https://marks.hms.harvard.edu/ev_rna/, e.g. RF00167.EC.interaction.csv

3.7. RNA filter (DCA)
interaction_fn

Positional arguments:

interaction_fn interaction file

Options:

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

3.7.4 rna_pairs2SimRNArestrs.py - convert pairs to SimRNA restraints

rna_pairs2SimRNArestrs.py - convert pairs to SimRNA restraints

Example:

$ 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] [--offset OFFSET] [--weight WEIGHT]
[--dist DIST] [--well] [-v]
pairs

Positional arguments:

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

Options:

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

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

Input file:
```
cat ade_ss.fa >1y26
CGCUUCAUAUAUCCUAUAUGUUGUUGGAGUUUCUACCAAGAGCCUAAACUCUUGAUUAUGAAGUG
(((((((((...((((((.............))))))))))))))))))))))))))))))))))))%```

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

Now it also work with pseudoknots.
```
usage: rna_ss_get_bps [-h] [--offset OFFSET] [-v] file
```

Positional arguments:
```
file
```
file in the Fasta format

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

3.7.6  rna_pairs_diff.py - get a diff of pairs

rna_pairs_diff.py - get a diff of pairs

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

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

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

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


#### 3.8.1 3DNA (contacts classification & secondary structure detection)

Python parser to 3dna [http://x3dna.org/](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-tools/rna_tools/tools/rna_x3dna folder.
Edit this line:
BINARY_PATH = <path to your x3dna-dssr file>
making the path with the path of your x3dna-dssr file.
e.g. in my case:  BINARY_PATH = ~/bin/x3dna-dssr
```

For one structure you can run this script as:

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

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

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

```
import rna_tools

class rna_tools.tools.rna_x3dna.rna_x3dna.get_parser()

    Attributes:
        curr_fn
        report
        clean_up
            (verbose=False)

    get_ion_water_report()
```

File name: /tmp/tmp0pdNHS

no. of DNA/RNA chains: 0
no. of nucleotides: 174
no. of waters: 793
no. of metals: 33
[Na=29, Mg=1, K=3]
get_modifications()
Run find_pair to find modifications.

get_secstruc()
Get secondary structure.

get_seq()
Get sequence.

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

run_x3dna(show_log=False)

3.8.2 ClaRNA (contacts classification)

Warning: ClaRNA requires some extra libraries, not required by other tools in the package, that's why they are not included in the main installation configuration. To install these libraries in proper version, type “pip install rna-tools simplejson networkx==1.8.1”.

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

To run ClaRNA, see the documentation below:

Usage:

```
$ rna_clarna_app.py ../../input/5k7c_clean_onechain_renumber_as_puzzle_srr.pdb
→ ../../input/5k7c_clean_onechain_renumber_as_puzzle_srr.pdb
(((([[[[[]]]]]))........((((.....]])]]].)))..(((.....)))..)))
```

Example

```python
from rna_tools.utils.clarna_app import clarna_app
if __name__ == '__main__':
    ss = '(((([[[[[]]]]]))........((((.....]])]]].)))..(((.....)))..)))
    fnCRref = clarna_app.get_ClaRNA_output_from_dot_bracket(ss)
    f = '../rna_calc_rmsd/test_data/pistol/5k7c_clean_onechain_renumber_as_puzzle_srr.pdb'
    fnCR = clarna_app.clarna_run(f, force=False)
    results = clarna_app.clarna_compare(fnCRref, fnCR)
    print results
    #tmp_242i_..pdb.outCR 5k7c_clean_onechain_renumber_as_puzzle_srr.pdb.outCR
    → 0.706 NA 0.865 NA 0.842 0.889 NA 0.000
```

Run ClaRNA compare.

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

Scores:

<table>
<thead>
<tr>
<th>inf_all</th>
<th>inf_stack</th>
<th>inf_WC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.706</td>
<td>-999.999</td>
<td>0.865</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>inf_nWC</th>
<th>~999.999 -&gt; NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNS_WC</td>
<td>0.842</td>
</tr>
<tr>
<td>PPV_WC</td>
<td>0.889</td>
</tr>
<tr>
<td>SNS_nWC</td>
<td>NA</td>
</tr>
<tr>
<td>PPV_nWC</td>
<td>0.000</td>
</tr>
</tbody>
</table>

```
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_tools.tools.clarna_app.clarna_app.clarna_run(fn, force=True, stacking=True, verbose=False)
```

Run ClaRNA run

- **Parameters** `fn (str)` – filename to analyze
- **Returns** a filename to ClaRNA output (fn + `.outCR`)
- **Return type** `str`

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

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

- **Parameters** `ss (string)` – secondary structure
- **Returns** a filename to ClaRNA output

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

In inCR file

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

## 3.9 RNA 3D model quality assessment

Wrappers behind the server, [http://genesilico.pl/mqapRNA](http://genesilico.pl/mqapRNA) (in short, mq)

### 3.9.1 RASP

This module contains functions for computing RASP potential

```python
class rna_tools.tools.mq.RASP.RASP(sequence="", seq_name="", job_id=None)
```

Wrapper class for running RASP automatically.

- **colour_by_local_score** *(path_to_pdb, potential_type='all')*
- **executable** = `['rasp_fd', 'rasp_profile_fd']`
- **program_name** = `['rasp_fd', 'rasp_profile_fd']`
run (path_to_pdb, global_energy_score=True, potentials=['c3', 'bb', 'bbr', 'all'], handler=True, verbose=False)
Compute RASP potential for a single file

**rasp_fd generates:** output_all.txt:

| -9869.61 | 66447 | -0.148534 | 0 | 0 | 0 |

**rasp_profile_fd generates:**

**profile_all.txt:**

```
C 1 R -775.038
C 2 R -1164.22
U 3 R -2054.17
G 4 R -1601.13
[..]
```

Input:

- path_to_pdb = path to PDB file
- potential_type = all, bbr or c3
- global_energy_score = True/False (See Output), default=True
- handler = True (if you use PYRO)/FALSE otherwise

Output:

- the output depends on global_energy_score value T/F You might get:
  -9869.61
  profile [(‘C’, 1, ‘R’, -775.03800000000001), (‘C’, 2, ‘R’, -1164.22) ...

### 3.9.2 Dfire

This module contains functions for computing Dfire potential

**class rna_tools.tools.mq.Dfire.Dfire.Dfire** (sequence, seq_name, job_id=None)

Wrapper class for Dfire.

- **max_seq_len = 100000**
- **run (path_to_pdb, verbose=False)**

### 3.9.3 RNAkb

This module contains functions for computing RNAkb potential

It seems that this is impossible to run RNAkb in full atom mode. So this works only in 5 pt (5 points/atom per residue) mode.


**class rna_tools.tools.mq.RNAkb.RNAkb** (sequence=", seq_name=", job_id=", sandbox=False)

Wrapper class for running RNAkb automatically.
exe = 'mdrun'

executable = ['/usr/local/gromacs/bin/pdb2gmx', '/usr/local/gromacs/bin/make_ndx', '/usr/local/gromacs/bin/editconf', '/usr/local/gromacs/bin/grompp', '/usr/local/gromacs/bin/mdrun']

log_stdout_stderr()

run(name, potential_type, verbose=False)

Compute RNAkb potential for a single file

Parameters

- **path** (str) – name of a PDB file
- **potential_type** (str) – ‘5pt’ for 5 point or ‘aa’ for all atom aa is off

Returns a list of energies (strings) ['2.57116e+05', '1.62131e+03', '7.82459e+02', '3.00789e-01', '0.00000e+00', '0.00000e+00', '-2.51238e-03', '0.00000e+00', '2.59520e+05', '2.54174e-09', '2.59520e+05']

**Warning:** ‘aa’ does not work because of “a line longer than 4095 characters”

The function parses:

```
Statistics over 1 steps using 1 frames
Energies (kJ/mol)
F     Bond   Angle       Proper Dih.  Improper Dih.       LJ-14
2.44111e+05  1.76069e+03  8.12947e+02  1.82656e-01  0.00000e+00
0.00000e+00 -1.94624e-03  0.00000e+00  2.46685e+05  2.43227e-09
Total Energy  Temperature  Pressure (bar)
2.46685e+05   6.67884e-10   -5.94232e+04
Total Virial
```

Standalone tool to run the RNAkb class in the terminal.

All atom mode does not really work, see the documentation of the RNAkb class.

```
```

**Positional arguments:**
- **file** a PDB file, one or more

**Options:**
- **-v=False, --verbose=False** be verbose
3.9.4 QRNA

This module contains functions for computing QRNA.

Output:

```plaintext
(...)
Missing atom added: ATOM   34  H3T RA3 A   77   39.444   67.315   58.412   1.00  0.
    00  H
Missing atom added: ATOM   23  OP3 G A   1   46.987   84.037   61.665   1.00  0.00
Number of atoms: 2470
Number of residues: 77
Building intrarresidual bonds...
Building intrarresidual angles...
Building intrarresidual dihedrals...
Building intrarresidual impropers...
Building interresidual bonds...
Building interresidual angles...
Building interresidual dihedrals...
Number of bonds built: 2662
Number of angles built: 4764
Number of dihedrals built: 7406
Number of impropers built: 524
End of molecule (chain)

Building interresidual nonbonded pairs & H-bonds (it may take a while)...  
Number of electrostatic pairs: 0
Number of van der Waals pairs: 332101
Number of H-bonds built: 64
Number of spring restraints: 0
Number of positional restraints: 0
Number of base pairs built: 20
Number of molecules (chains) read: 1

Performing minimization step: 1. Total energy = 28783.5671 kcal/mol (28753.2335 without restraints)
Writing PDB file: /var/folders/yc/ssr9692s5fzf7k165grnhpk80000gp/T/tmpic_m3bc/qrna/query_out.pdb ...Done.
```

```python
class rna_tools.tools.mq.QRNA.QRNA(sequence='', seq_name='', job_id=None)
    Wrapper class for AMBER force field via QRNA of Julisz Stasiewicz

    executable = 'QRNA'
    program_name = 'QRNA'
    run()

    Returns e.g., [30307.1088, '28783.5671']

    run_one(filename, numSteps, electrostatics=True, verbose=False)
    Get AMBER energy via QRNA

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

3.9. RNA 3D model quality assessment
instance is destructed

**Returns**  A buffered writable file descriptor

**Return type**  BufferedFileStorage

**Parameters**  name = name(*) – of a PDB file

**Output:**
- global energy as a floating point number

```
rna_tools.tools.mq.QRNA.QRNA.main()
```

### 3.9.5 eSCORE

This module contains functions for computing eSCORE

**Install:**

```
pip install barnaba
```

**Output:**

```
$ baRNAba ESCORE --pdb .../test/1a9n.pdb --ff /Users/magnus/work/opt/barnaba/barnaba_→201128/test/data/1S72.pdb
# your output will be written to files with prefix outfile.ESCORE
# KDE computed. Bandwidth= 0.25 using 10655 base-pairs# Loaded sample .../test/1a9n. →pdb
#     Frame   ESCORE
    0   4.1693e-01
```

The Escore could be also accessed via Python:

```python
from barnaba import escore
Escore = escore.Escore([path_to_pdb])
(..)
# see example_12_escore.ipynb of barnaba package https://github.com/srnas/barnaba
```

```
class rna_tools.tools.mq.eSCORE.eSCORE.eSCORE
    Wrapper class for eSCORE
    run(path_to_pdb, verbose=True)
rna_tools.tools.mq.eSCORE.eSCORE.main()
```

### 3.9.6 3dRNAscore

This module contains functions for computing 3dRNAscore

**Install:**

```
make  # make clean if you don’t have clean
```

At Mac there is a problem (201128):
3.9.7 RNA3DCNN

This module contains functions for computing RNA3DCNN potential

Output:

Trainable params: 4,282,801
Non-trainable params: 0

Scores for each nucleotide in /var/folders/yc/ssr9692s5fzf7k165grnhpk80000gp/T/tmptg6jy2ud/query.pdb:

```
[0.02462262]
[0.03271335]
[0.06199259]
[0.02006263]
[0.05937254]
[0.12025979]
[0.20201728]
[0.24463326]
[0.43518737]
[0.7260638 ]
[0.6140108 ]
[0.6588027 ]
[0.7668936 ]
[0.4776191 ]
[0.0560993 ]
[0.05285829]
[0.0167731 ]
[0.01759553]
[0.02143204]
[-0.01818037]
```

Total score for /var/folders/yc/ssr9692s5fzf7k165grnhpk80000gp/T/tmptg6jy2ud/query.pdb is 6.3262305

If missing atoms:

```
Total params: 4,282,801
Trainable params: 4,282,801
Non-trainable params: 0

There is no atom O5' in residue 620A in chain A in PDB /var/folders/yc/ssr9692s5fzf7k165grnhpk80000gp/T/tmptg6jy2ud/query.pdb
```

3.9. RNA 3D model quality assessment
There is no atom O5' in residue 635A in chain B in PDB /var/folders/yc/-ssr9692s5fzf7k165grnhhk80000gp/T/tmpx87uus6x/query.pdb.
There is no atom O5' in residue 1750G in chain C in PDB /var/folders/yc/-ssr9692s5fzf7k165grnhhk80000gp/T/tmpx87uus6x/query.pdb.

Scores for each nucleotide in /var/folders/yc/ssr9692s5fzf7k165grnhhk80000gp/T/
˓→tmpx87uus6x/query.pdb:
[]

Total score for /var/folders/yc/ssr9692s5fzf7k165grnhhk80000gp/T/tmpx87uus6x/query.
˓→pdb is 0.0

class rna_tools.tools.mq.RNA3DCNN.RNA3DCNN.RNA3DCNN
    Wrapper class for RNA3DCNN.
    max_seq_len = 100000
    run(path_to_pdb, verbose=False)

rna_tools.tools.mq.RNA3DCNN.RNA3DCNN.main()

3.9.8 FARNA

Wrapper for ROSETTA software for structure prediction of small RNA sequences

class rna_tools.tools.mq.FARNA.FARNA
    Wrapper class for running ROSETTA scoring function automatically.
    best_energy = ''
    cleanup()
    db_path = ''
    executable = 'rna_minimize'
    get_result()
        Parse and get result from score file created during ROSETTA run
        All results are kept in self.result, but only global score is returned
    input_file = ''
    input_fn = 'seq.fasta'
    mqap(pdb)
        Total weighted score:s+(?P<ROSETTA_SCORE>[-d.]+)
    program_name = 'farna'
    run(pdb_file, hires)
        Compute FARNA potential for a single file

Parameters
    • pdb_file = path to pdb file(*)-
    • global_energy_score = True/False(*)-

Output:
    • A list of energies, e.g:
### 3.9.9 ClashScore

This module contains functions for computing NAST potential via Pyro.

```python
class rna_tools.tools.mq.ClashScore.ClashScore:
    ClashScore: Wrapper class for running phenix.clashscore

cleanup()

run(fn, verbose=False)
```

**Parameters**

- `fn (string)` – path to a file
- `verbose (bool)` – be verbose

**Returns** clashscore (float)

**Example:**

```bash
/Applications/phenix-1.18.2-3874/build/bin/phenix.clashscore test/1xjrA.pdb
Using electron cloud x-H distances and vdW radii

Adding H/D atoms with reduce...

Bad Clashes >= 0.4 Angstrom:
A 17 A N1 A 34 G N1 :0.430
A 41 G H2' A 42 U OP1 :0.738
A 25 U H6 A 25 U HO2' :0.659
A 46 U H2' A 47 U OP2 :0.656
A 29 A H5'' A 30 U O2' :0.623
(...)
A 24 G O2' A 26 A C8 :0.410
```

```bash
/Applications/phenix-1.18.2-3874/build/bin/phenix.clashscore test/1xjrA_M1.pdb
Using electron cloud x-H distances and vdW radii

Adding H/D atoms with reduce...

Bad Clashes >= 0.4 Angstrom:
A 41 G H2' A 42 U OP1 :0.738
A 25 U H6 A 25 U HO2' :0.659
A 46 U H2' A 47 U OP2 :0.656
A 29 A H5'' A 30 U O2' :0.623
(...)
A 24 G O2' A 26 A C8 :0.410
```
rna-tools documentation, Release 3.6.x

<table>
<thead>
<tr>
<th>42</th>
<th>U</th>
<th>O2'</th>
<th>A</th>
<th>43</th>
<th>G</th>
<th>H8 :0.409</th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
<td>G</td>
<td>H2'</td>
<td>A</td>
<td>44</td>
<td>A</td>
<td>O4' :0.408</td>
</tr>
<tr>
<td>28</td>
<td>G</td>
<td>C8</td>
<td>A</td>
<td>28</td>
<td>G</td>
<td>O2 ' :0.403</td>
</tr>
</tbody>
</table>

clashscore = 15.45
test/1xjrA_M1.pdb

# so the output is
clashscore = 15.45

rna_tools.tools.mq.ClashScore.ClashScore.test()

### 3.9.10 AnalyzeGeometry

This module contains functions for computing AnalyzeGeometry.

```python
class rna_tools.tools.mq.AnalyzeGeometry.AnalyzeGeometry>AnalyzeGeometry(verbosel=False)
```

Wrapper class for running clashscore

**Note:** Sequence is required to get the length to calculate % of correct residues.

```python
cleanup()
run(name, verbose=False)
```

**Parameters**

- `name (str)` – the path of the file to wrap
- `verbose (boolen)` – be verbose

**Returns** score; % of Backbone torsion suites (# of them per seq)

**Return type** float

**Output:**

```
----------Backbone torsion suites----------

<table>
<thead>
<tr>
<th>Suite ID</th>
<th>suite</th>
<th>suiteness</th>
<th>triaged angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>A A 3</td>
<td>!!</td>
<td>0.000</td>
<td>delta</td>
</tr>
<tr>
<td>G A 4</td>
<td>!!</td>
<td>0.000</td>
<td>epsilon-1</td>
</tr>
<tr>
<td>G A 11</td>
<td>!!</td>
<td>0.000</td>
<td>None</td>
</tr>
<tr>
<td>C A 20</td>
<td>!!</td>
<td>0.000</td>
<td>gamma</td>
</tr>
<tr>
<td>U A 25</td>
<td>!!</td>
<td>0.000</td>
<td>delta</td>
</tr>
<tr>
<td>A A 26</td>
<td>!!</td>
<td>0.000</td>
<td>delta-1</td>
</tr>
<tr>
<td>A A 29</td>
<td>!!</td>
<td>0.000</td>
<td>None</td>
</tr>
<tr>
<td>U A 30</td>
<td>!!</td>
<td>0.000</td>
<td>epsilon-1</td>
</tr>
<tr>
<td>G A 41</td>
<td>!!</td>
<td>0.000</td>
<td>delta</td>
</tr>
<tr>
<td>U A 42</td>
<td>!!</td>
<td>0.000</td>
<td>delta-1</td>
</tr>
<tr>
<td>A A 45</td>
<td>!!</td>
<td>0.000</td>
<td>delta</td>
</tr>
<tr>
<td>U A 46</td>
<td>!!</td>
<td>0.000</td>
<td>epsilon-1</td>
</tr>
<tr>
<td>U A 47</td>
<td>!!</td>
<td>0.000</td>
<td>delta-1</td>
</tr>
</tbody>
</table>

11 suites triaged and 0 incomplete leaving 35 suites
13/46 suite outliers present
Average suiteness: 0.490
```

13 # count lines after 'traged angle' and minus 3 (# of last lines)
### Output for a perfect structure:

```
CGACGCUAGCGUACGCUAGCGUCG
```

All bonds within 4.0 sigma of ideal values.

All angles within 4.0 sigma of ideal values.

All puckers have reasonable geometry.

0 suites triaged and 0 incomplete leaving 24 suites

All RNA torsion suites are reasonable.

Average suiteness: 0.766

---

### 3.10 RNA 3D structure prediction

#### 3.10.1 ROSETTA


#### 3.10.1.1 Run (modeling)

**rna_rosetta_run.py** - prepare & run ROSETTA simulations


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

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

I discovered this:

```
qstat -xml | tr '\n' '' | sed 's$<job_list[^>]*>$
```

> | 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
```

```
# prepare a folder for a run
>cp20
AUUAUCAAGAAUCUCAAGAGGAGAUAGCAACCUGCAUAAACGAGCAAGGUGCUAAAAUAUGAGAAUGCCAUAUUGGAAAGUA
.((((((.(((((((........)))))).........................))))))))((((..)))).......
→
[peyote2] ~ rna_rosetta_run.py -i cp20.fa
run rosetta for:
    cp20
AUUAUCAAGAAUCUCAAGAGGAGAUAGCAACCUGCAUAAACGAGCAAGGUGCUAAAAUAUGAGAAUGCCAUAUUGGAAAGUA
.((((((.(((((((........)))))).........................))))))))((((..)))).......
/home / magnus // cp20 / created
Seq & ss created
```

Troubleshooting.

If one of the helices is missing you will get:

```
IOError: [Errno 2] No such file or directory: 'helix1.out'
rosetta_submit.py README_FARFAR o 500 100 taf
Could not find: README_FARFAR
```

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 characters
```

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

Positional arguments:

```
file
    file: > a04 UAUAACAUUAUUUUGACAUAUAUGGGAUCAUAAGU-UUCUACCCGAAAUAUUCUGACUAUGU/AUA
        (((((((((...(((((........)))))).........................)))))))))))
```
Options:

- **-i=False, --init=False**  prepare _folder with seq and ss
- **-e=False, --helices=False**  produce h(E)lices
- **-r=False, --rosetta=False**  prepare rosetta files (still you need ‘go’ to send jobs to a cluster)
- **-g=False, --go=False**  send jobs to a cluster(run qsubMINI)
- **-n=10000, --nstruct=10000**  # of structures you want to get
- **-c=200, --cpus=200**  # of cpus to be used
- **--sandbox=**  where to run it (default: RNA_ROSETTA_RUN_ROOT_DIR_MODELING)

```
qstat -xml | tr \n '' | sed 's#<job_list[^>]*>##g' |
> | 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.

**Helix**

Run:

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

```
-i:

# prepare a folder for a run
>cp20
AUUAUCAAGAAUCUCAAGAGAGAGAGCAACCCUGGACGUAACCGCAGGACGUAACUCGGCACAUAGGAAGCAAGCCAAUUGGAAAAAUGUAA
.((((((((((((((((())))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))

[peyote2] ~ rna_rosetta_run.py -i cp20.fa
run rosetta for:

```
cp20
AUUAUCAAGAAUCUCAAGAGAGAGAGCAACCCUGGACGUAACCGCAGGACGUAACUCGGCACAUAGGAAGCAAGCCAAUUGGAAAAAUGUAA
.((((((((((((((((())))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
```
```
/home / magnus // cp20 / created
Seq & ss created
```

3.10. RNA 3D structure prediction 97
Troubleshooting.

If one of the helices is missing you will get:

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

Could not find: README_FARFAR

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 characters
```

```
class rna_tools.tools.rna_rosetta.rna_rosetta_run.CustomFormatter (prog, indent_increment=2, max_help_position=24, width=None)

rna_tools.tools.rna_rosetta.rna_rosetta_run.get_parser()
rna_tools.tools.rna_rosetta.rna_rosetta_run.go()
    send jobs to a cluster(run qsubMINI)
rna_tools.tools.rna_rosetta.rna_rosetta_run.main()
Pipeline for modeling RNA
rna_tools.tools.rna_rosetta.rna_rosetta_run.prepare_folder (args, header, seq, ss, path)
    Make folder for you job, with seq.fa, ss.fa, input file is copied as input.fa to the folder.
```

For ss lowercase is needed when used with motifs, otherwise:

```
[rpeyote2] aa20$ rna_rosetta_run.py -r -m E-loop_lq9a mutated_no_flanks_renumber_for_acy20.pdb ~/aa20.fa
2019-05-03 21:31:30,842 rpt_config_local.py::<module>::rpt_config_local loading...
run rosetta for:
    aa20
    UACGUUCAUCCUUGGAUGACGGAAUGCUAAGCGAAAGCGAAAGGAACGGAUG
    .((((.((((((.)))))))))[..((((.((.)))))))))[[[ ...
    rna_denovo_setup.py -fasta seq.fa -secstruct_file ss.fa -cycles 20000 -no_
    -minimize -nstruct 50 -s E-loop_lq9a mutated_no_flanks_renumber_for_acy20.pdb -
    -silent helix0.out helix1.out helix2.out -input_silent_res 3-7 47-51 9-14 19-24 -
    33-35 40-42
Sequence: UACGUUCAUCCUUGGAUGACGGAAUGCUAAGCGAAAGCGAAAGGAACGGAUG
Secstruc: .((((.((((((.)))))))))[..((((.((.)))))))))[[[...
```
aaguagaag
AAGUAGAAG
Traceback (most recent call last):
  →rna_denovo_setup.py", line 170, in <module>
    raise ValueError('The sequence in %s does not match input sequence!!' % pdb)
ValueError: The sequence in E-loop_1q9a_mutated_no_flanks_renumber_for_acy20.pdb
  →does not match input sequence!!
rosetta_submit.py README_FARFAR o 200 100 aa20_
Could not find: README_FARFAR

rna_tools.tools.rna_rosetta.rna_rosetta_run.\texttt{prepare\_helices}()
Make helices(wrapper around `helix\_preassemble\_setup.py`)

\textbf{Warning:} I think multiprocessing of helixX.run does not work.

rna_tools.tools.rna_rosetta.rna_rosetta_run.\texttt{prepare\_rosetta}(header, cpus, motif, nstruc)
Prepare ROSETTA using rna_denovo_setup.py
cpus is used to calc nstruc per job to get 10k structures per full run:

\begin{verbatim}
:param nstruc: how many structures you want to obtain
:type nstruc: int
:param nstruct = int: math.floor(20000 / cpus)
:param motif: motif file; e.g., -s E-loop_1q9a_mutated_no_flanks_renumber.pdb
:param 50:
:type 50: nstruc) = 10k / 200 (cpus
\end{verbatim}

\subsection*{3.10.1.2 Get a number of structures}

\texttt{rna\_roseta\_n.py} - show me # of structure in a silent file
Example:

\$ rna_roseta_n.py ade.out
21594

\texttt{usage: rna\_roseta\_n.py [-h] [-v] file}

Positional arguments:

\begin{verbatim}
file ade.out
\end{verbatim}

Options:

\begin{verbatim}
-v=False, --verbose=False  Undocumented
\end{verbatim}

\texttt{rna\_roseta\_n.py} - show me # of structure in a silent file
Example:

\$ rna_roseta_n.py ade.out
21594

\texttt{rna_tools.tools.rna_rosetta.rna_rosetta_n.get\_no\_structures(file)
3.10.1.3 Get a head of a Rosetta silent file

Example:

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

It seems to work:

```bash
-rw-r--r-- 1 magnus users 474M 2017-08-06 05:25 thf_10000.out
-rw -rw-r-- 1 magnus users 474M 2017-08-06 05:24 thf.out
```
usage: rna_rosetta_n.py [-h] [-v] [-n NSTRUC] file

Positional arguments:
  file               ade.out

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

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

Example:

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

Silent file:

```plaintext
[peyote2] thf head -n 1000 thf.out
SEQUENCE:
-ggagaguagauggauaguggauugucacacaacgagcgagcgugagaugcaucauggauguuacgcggcca

REMARK BINARY SILENTFILE
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 NWC N_WC N_BS description

FOLD_TREE EDGE 1 4 -1 JEDGE 4 85 1 C4 C2 END EDGE 4 5 -1 EDGE 85 80 -1 EDGE

RT -0.987743 0.139354 0.0703103 0.135963 0.989404 -0.0509304 -0.0766626 -0.0407466 -0.

RT -0.98312 0.1587 -0.091045 0.166923 0.981743 -0.0912024 0.074909 -0.10486 -0.991662

RT -0.987645 0.154078 0.0180536 0.153854 0.989532 -0.0348618 -0.0227723 -0.0319807 -0.

RT -0.990412 0.140722 0.00546261 0.137927 0.990299 -0.0168644 -0.00773751 -0.0159492 -0.

ANNOTATED_SEQUENCE: g[RGU:LowerRNA:Virtual_Phosphate]gaga[RAD:rna_cutpoint_
  lower]g[RGU:rna_cutpoint_upper]ugagauuagguguaguguguagggaguc[RCY:rna_
  cutpoint_upper]cauccgcggca[RAD:UpperRNA] S_000001_000
```
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
```

```python
rna_tools.tools.rna_rosetta.rna_rosetta_head.get_parser()

rna_tools.tools.rna_rosetta.rna_rosetta_head.run()
```

Pipeline for modeling RNA.

### 3.10.1.4 Cluster

**rna_rosetta_cluster.py** - cluster silent Rosetta files

A wrapper to ROSETTA tools for RNA modeling


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

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

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

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

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

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

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

```python
rna_tools.tools.rna_rosetta.rna_rosetta_cluster.get_no_structures(file)
```

```python
rna_tools.tools.rna_rosetta.rna_rosetta_cluster.get_no_structures_in_first_cluster(fn)
```

Get # of structures in a silent file.

**Parameters** `fn` *(string)* – a filename to a silent file

**Returns** int: # of structures in a silent file

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

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

Get selected for clustering

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

Pipeline for modeling RNA
3.10.1.5 Minimize

rna_rosetta_min.py - a script to do minimization

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

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


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

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

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

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

E.g. for 20k silent file, 1/6 will be minimized = 3.3k:

```bash
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
```

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

Positional arguments:

- `file` ade.out

Options:

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

rna_rosetta_min.py - a script to do minimization

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

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


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

The first number states how many processors to use for the run, while the second number is 1/6 the total number of previously generated FARNA models. If you are running on a supercomputer that only allows specific multiples of processors, use an appropriate number for the first input:
**RNA tools documentation, Release 3.6.x**

```bash
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 output 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
```

**rna_tools.tools.rna_rosetta.rna_rosetta_min.get_no_structures(file)**

Get a number of structures in a silent file

**rna_tools.tools.rna_rosetta.rna_rosetta_min.get_parser()**

**rna_tools.tools.rna_rosetta.rna_rosetta_min.min(silent_file, take_n, cpus, go)**

Run parallel_min_setup (to MINIMIZE file), rosetta_submit.py, and qsubMINI.

Fix on the way, qsub files:

```bash
-out:file:silent mo/0/mo/123/tha_min.out -> -out:file:silent mo/123/tha_min.out
```

I don’t know why mo/0/ is there. I might be because of my changes in rosetta_submit.py (?).

**rna_tools.tools.rna_rosetta.rna_rosetta_min.run()**

### 3.10.1.6 Extract lowsore decoy

**rna_rosetta_extract_lowscore_decoys.py** - a simple wrapper to extract_lowscore_decoys.py

To be used in Jupyter notebooks and other scripts.

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

**Positional arguments:**

- `nstruc` # of low score structures to obtained
- `file` silent file

**Options:**

- `-v=False, --verbose=False` be verbose

**rna_rosetta_extract_lowscore_decoys.py** - a simple wrapper to extract_lowscore_decoys.py

To be used in Jupyter notebooks and other scripts.

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

**Positional arguments:**

- `nstruc` # of low score structures to obtained
- `file` silent file

**Options:**

- `-v=False, --verbose=False` be verbose
3.10.1.7 Check progress

rna_rosetta_check_progress.py - check progress for many simulations of Rosetta

Example:

```
[peyote2] rosetta_jobs rna_rosetta_check_progress.py .
jobs #curr #todo #decoys done
0 ./rp17s223 200 0 407 [ ]
1 ./rp17hcf 0 0 0 [ ]
```

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

Positional arguments:

```
dir directory with rosetta runs, define by RNA_ROSETTA_RUN_ROOT_DIR_MODELING right now:
```

Options:

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

rna_rosetta_check_progress.py - check progress for many simulations of Rosetta

Example:

```
[peyote2] rosetta_jobs rna_rosetta_check_progress.py .
jobs #curr #todo #decoys done
0 ./rp17s223 200 0 407 [ ]
1 ./rp17hcf 0 0 0 [ ]
```

```
rna_tools.tools.rna_rosetta.rna_rosetta_check_progress.get_parser()
rna_tools.tools.rna_rosetta.rna_rosetta_check_progress.run_cmd(cmd)
```

3.10.2 SimRNA

3.10.2.1 Select low energy frames

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

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

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

Positional arguments:

```
trafi  SimRNA trajectory file
```

Options:

```
-n=100, --nstruc=100  SimRNA trajectory file
```
3.10.2.2 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]
```

Options:

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

3.10.3 SimRNAweb

3.10.3.1 Download files of a SimRNAweb run

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

Usage:

```
rp17pk$ rna_pdb_download_simrna_job.py 27b5093d -m -t -x
# download more clusters, trajectory, extract100

cp771_pk$ rna_pdb_download_simrna_job.py -t -x -m cf8f8bb2 -p cp771_pk
# download with a trajectory, and cluster #4 and #5, add to all pdb files
# prefix: cp771_pk

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

Example:

```
rna_pdb_download_simrna_job.py -t -x -m 20569fa1 -p zmp_pk

[m] zmp_pk ls
20569fa1_ALL_1001ow.trafl
_20569fa1-thrs7.10A_clust04
_20569fa1-thrs7.10A_clust05
_20569fa1_ALL_1001ow
data
rna_simrna_extract.log
subset.png
zmp_pk_20569fa1-thrs7.10A_clust01X.pdb
zmp_pk_20569fa1-thrs7.10A_clust02X.pdb
zmp_pk_20569fa1-thrs7.10A_clust03X.pdb
zmp_pk_20569fa1-thrs7.10A_clust04X.pdb
zmp_pk_20569fa1-thrs7.10A_clust05X.pdb
```

Positional arguments:

  job_id  job_id

Options:

  -p, --prefix    prefix to the name, without _, be careful with this. If you have already some files with the given folder, their names might be changed.

  -n=100, --nstruc=100  extract nstruc the lowest energy, this option must go with –web

  -e=False, --extract=False  extract nstruc the lowest energy, this option must go with –web

  -m=False, --more_clusters=False  download also cluster 4 and 5

  -r=False, --remove-trajectory=False  remove trajectory after analysis

  -c=False, --cluster=False  get trajectory from cluster OR local on your computer (mdfind for macOS)

  -d=False, --download-trajectory=False  web

  --top100=False  download top100 trajectory

  --top200=False  download top200 trajectory

  --web-models=False  web models download

3.10.4 SimRNATrajectory

SimRNATrajectory module.

SimRNATrajectory / Frame / Residue / Atom

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

  get_coord()  
  Return coords (np.array).

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

Syntax of header:

  • write_number
  • replica_id
  • total_energy
  • energy_without_restraints
  • temperature

Warning: If there is an invalid frame, please use repair_trafl.py to fix the trajectory first.
\texttt{rmsd\_to(frame, verbose=False)}

\begin{verbatim}
class rna_tools.tools.simrna_trajectory.simrna_trajectory.Residue(id, p, c4p, n1n9, b1, b2)

Create Residue object.
Each residue in SimRNA coarse-grained represantation consists only 5 coarse-grained atoms:
- backbone: \( p = \text{phosphate group, c4p = sugar moiety} \)
- nucleotide: \( n1n9 = N1 \text{ for pyrimidines, N9 for purines, b1 = C2 for purines and pyrimidines, b2 = C4 for pyrimidines, C6 for purines} \)
\end{verbatim}

\begin{verbatim}
def get_atoms()
    Return all atoms

def get_center()
    Return MB for residue \( ((\text{self.n1n9 + self.b2) / 2}) \)
\end{verbatim}

\begin{verbatim}
class rna_tools.tools.simrna_trajectory.simrna_trajectory.SimRNATrajectory

load\_from\_file(fn, debug\_break=False, top\_level=False, only\_first\_frame=False)
Create a trajectory based on 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

\begin{verbatim}
Warning: Loads up whole trafl 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.
\end{verbatim}

\begin{verbatim}
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 (plot\_fn='plot.png')
Plots the SimRNA energy of the trajectory.
\end{verbatim}
Save the trajectory to file.

Sort frames within the trajectory according to energy.

### 3.11 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, after 3k, ~10min 20A_clust01-000001_AA.pdb.
after 10k steps, around 30min
after 20k steps, around 1h.

**Installation of QRNAS**

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

```
./qrnamake sequential
```

This should create an executable version of QRNAS.

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

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

**Usage of QRNA:**

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

To use type:

```
QRNA -i <input PDBfile> [-o <output PDBfile>] [-c <configfile>] [-p] [-m <restraintsfile>]
```

OR specify `<input PDBfile>`, `<output PDBfile>` and `<restraintsfile>` in `<configfile>` and `--type` just:

```
QRNA --c <configfile>
```

**Installation of this util**

Set up in your bashrc:

```
export QRNAS_PATH=<your path to qrnas> # e.g. /home/magnus/src/rna-pdb-tools/opt/qrnas
```

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

**QRNAS at Peyote2**

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

To run it at a cluster with the Sun Grid Engine queuing system (this one with qusb ;-)):  

3.11. RNA Refinement (QRNAS)
for p in *.pdb; do echo "rna_refinement.py $p $p.log" | qsub -cwd -V -pe mpi 1 -N "r_$p" ; done

DONE:

• [x] clean up the output structure
• [x] configuration should not be hardcoded

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

Positional arguments:

    fn  input pdb file

Options:

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

3.12 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/tools/diffpdb/diffpdb_conf.py (create it if needed)
./diffpdb.py --names test_data/4/1duq.pdb test_data/4/1duq_decoy0171_amb_clx.pdb
and on the Mac (using **diffmerge**):

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

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

3.13 RNA clustering with CLANS (clanstix)

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

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

Quickref:

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

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

The RMSDs between structures are converted into p-values based on the method from the Dokholyan lab or some hacky way developed by mmagnus.
3.13.1 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](http://www.rapidtables.com/web/color/RGB_Color.htm))

3.13.2 How to use ClanstixRNA?

1. Get a matrix of distances, save it as e.g. matrix.txt (see Comment below)
2. run ClanstixRNA on this matrix to get an input file to Clans (e.g. clans_rna.txt):
   ```
rna_clanstix.py test_data/matrix.txt
   # clans.input will be created by default
   ```
3. open CLANS and click File -> Load run and load clans_run.txt
4. You’re done! :-)

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

```
rna_calc_rmsd_all_vs_all.py -i rp18 -o rp18_rmsd.csv
rna_clastix.py --groups 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  clans.in
```
rna_clastix.py --groups 100+100+100+100+100+100+100+100:1:native rp18_rmsd.csv

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_tools.tools.clanstix.rna_clanstix.RNAStructClans (n=10, dotsize=10)
Clans run.

Usage:

```
3.14 Misc

3.14.1 Plotting

rna_plot_hist.py - generate a histogram

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

Example:

```
file
fn    rmsd_all
19_Bujnicki_Human_4_rpr_n0-000001.pdb-000001_A... 14.73
19_Bujnicki_Human_4_rpr_n0-000001.pdb.trafl_19... 0.46
19_Bujnicki_Human_4_rpr_n0-000001.pdb.trafl_19... 14.73
19_Bujnicki_Human_4_rpr_n0-000001.pdb_thrs0.50... 0.73
19_Bujnicki_Human_4_rpr_n0-000001.pdb_thrs0.50... 0.83
```

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

               file

Positional arguments:

```markdown
file
rmsd.txt
```

Options:
rna_plot_density.py - generate a density plot

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

Example:

```plaintext
# file

<table>
<thead>
<tr>
<th>fn</th>
<th>rmsd_all</th>
</tr>
</thead>
<tbody>
<tr>
<td>19_Bujnicki_Human_4_rpr_n0-000001.pdb-000001_A...</td>
<td>14.73</td>
</tr>
<tr>
<td>19_Bujnicki_Human_4_rpr_n0-000001.pdb.trafl_19...</td>
<td>0.46</td>
</tr>
<tr>
<td>19_Bujnicki_Human_4_rpr_n0-000001.pdb.trafl_19...</td>
<td>14.73</td>
</tr>
<tr>
<td>19_Bujnicki_Human_4_rpr_n0-000001.pdb_thrs0.50...</td>
<td>0.73</td>
</tr>
<tr>
<td>19_Bujnicki_Human_4_rpr_n0-000001.pdb_thrs0.50...</td>
<td>0.83</td>
</tr>
</tbody>
</table>
```

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

```

Positional arguments:

  file                     rmsd.txt

Options:

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

### 3.14.2 rna_sali2dotbracket

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

**Positional arguments:**

  filename  file in the Sali format

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

Author: Catarina Almeida

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

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

**To get the ss**

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

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

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 sometimes 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:

```python
re.sub(r'\d+', repl, temp)
re.sub(r'\d+\dnt', repl, temp)
re.sub(r'(?P<smthBeautiful>(\d+))', repl, temp)
```

**To get the sequence**

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

```python
rna_tools.tools.rna_sali2dotbracket.rna_sali2dotbracket.get_parser()
```

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

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

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

```
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
```

A super simple script to get some statistics of who is running at a cluster

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

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

3.14.4 RNAkb

RNAkb (previous Gromacs) utils.

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

Marcin Magnus Albert Bogdanowicz

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

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

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

**Parameters**

- `path` = path to gro file

**Output:**

- file is overwritten with a corrected one

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

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

**Parameters**

- `path` = path to index file

**Output:**

- index is overwritten with a corrected one
Get a template score mdp and replace energygrps (it can be generated with prepare_groups) and energygrp_table

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

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

Get residue code from a line of a PDB file

Extract residue number from a line of PDB file

Parameters

- residue number as an integer

GROMACS has some special requirements for PDB files.

Parameters

- new PDB returned as a string

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

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

- new PDB returned as a string

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:
return, gtxt (groups_txt), energygrps. The result is saved to g_fn. energygrps: ['uP', 'uC4s', 'uC2', 'uC4', 'uC6', 'gP', 'gC4s', 'gC2', 'gC4', 'aP', 'aC4s', 'aC2', 'aC4', 'aC6'] gtxt: RA del 1 r RU & a P name 1 uP r RU & a C4s name 2 uC4s r RU & a C2 name 3 uC2 r RU & a C4 [..] r RA & a C6 name 15 aC6 1|2|3|4|5|6|7|8|9|10|11|12|13|14|15 name 16 RNA_5pt 0 & ! 16 name 17 other q

*Set residue code from a line of a PDB file*

```python
set_res_code(line, code)
```
See also https://github.com/mmagnus/rna-tools/tree/master/rna_tools/tools/PyMOL4RNA

4.1 Inspect structure

There is new `--inspect` function for rna_pdb_toolsx.py that tells you what are missing atoms in your structure file.

You can run it also from PyMOL, so in here you can see missing atoms directly in PyMOL (the object comes green and the residues missing atoms comes in red).

PyMOL:

```
inspect <object name>
```
and here you can compare the input file vs file after `--rpr` (of rna_pdb_toolsx.py) to see what was rebuilt (in pink).
4.1. Inspect structure
4.2 Show base pair edges

Leontis/Westhof classification of base pairings. (A) RNA bases - adenine (A), cytosine (C), guanine (G) and uracil (U) - involve one of three distinct edges: the Watson–Crick (W) edge, the Hoogsteen (H) edge, and the Sugar (S) edge. (B) Each pair of can interact in either cis or trans orientations with respect to the glycosidic bonds. (C) For these reasons, all base pairs can be grouped into twelve geometric base pair families and eighteen pairing relationships (bases are represented as triangles). Each pair is represented by a symbol that can be used in a secondary structure and a tertiary structure diagrams. Filled symbols mean cis base pair configuration, and open symbols, trans base pair. (D) Interestingly, bases can form triples and they have own classification devised by Leontis and coworkers (Creative Commons License) (A. S. Abu Almakarem, A. I. Petrov, J. Stombaugh, C. L. Zirbel, and N. B. Leontis, “Comprehensive survey and geometric classification of base triples in RNA structures,” Nucleic Acids Research, vol. 40, no. 4, pp. 1407–1423, Feb. 2012.)
**Warning:** This PyMOL function is very simple and works on standard atom naming, so to make sure that everything will work, get your structure in the RNA Puzzle ready format, See https://rna-tools.readthedocs.io/en/latest/main.html#get-rnapuzzle-ready

**Type:**

`edges (<selection>)`

e.g.:

`edges (all)`

to show the edges drawn on bases.

4.2. **Show base pair edges**
4.3 PyMOL4Spliceosome: all spliceosome structures in one PyMOL session

Download one PyMOL session with structures for each step.

See own project [https://github.com/mmagnus/PyMOL4Spliceosome](https://github.com/mmagnus/PyMOL4Spliceosome)

4.4 PyMOL: Color by conservation

Show conserved regions of proteins in PyMOL.
4.5 PyMOL Drawing

RNA-tools documentation, Release 3.6.x

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 \([0,1]\).

\( w \) Line width of the circle

RETURNS the CGO object (it also loads it into PyMOL, too).

See more: https://github.com/mmagnus/rna-tools/tree/master/rna_tools/tools/pymol_color_by_conserv
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.

Install PyMOL plugin to view the interactions with PyMOL:

```python
run <path>rna-tools/tools/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], [6, 67], [12, 23], [52, 62], [30, 40], [49, 65], [27, 43], [11, 24], [1, 72], [10, 25], [15, 48], [53, 61], [19, 56], [13, 22], [36, 37], [18, 19], [22, 46], [35, 73], [32, 38], [9, 13], [19, 20], [18, 20], [54, 60], [9, 23], [34, 35], [36, 38], [53, 54], [20, 56], [9, 12], [26, 44], [18, 55], [54, 61], [32, 36]])
```
4.6 Install

After you install rna-tools, run these two lines your terminal:

```bash
$ echo "sys.path.append('`rna_tools_which.py --site`')" >> ~/.pymolrc
$ echo "PyMOL4RNA_PATH='`rna_tools_which.py`/tools/PyMOL4RNA/PyMOL4RNA.py'; cmd.run(PyMOL4RNA_PATH)" >> ~/.pymolrc
```

so something like this will be added to your ~/.pymolrc:

```python
sys.path.append('/Users/magnus/work/src/rna-tools')
PyMOL4RNA_PATH='/Users/magnus/work/src/rna-tools/rna_tools/tools/PyMOL4RNA/PyMOL4RNA.py'; cmd.run(PyMOL4RNA_PATH)
```
These functions are intended to be imported to tools to give a unified framework for making selections.

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

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

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

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

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

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 6

Workflows

6.1 Example #1

The native:

```
[mq] md_1msy_clx cat 1msy_clean.pdb.out
Classifier: Clarna
chains: A 2647 2673

A 2648 A 2672 bp G U WW_cis 0.8732
A 2649 A 2671 bp C G WW_cis 0.9160
A 2650 A 2670 bp U A WW_cis 0.9289
A 2651 A 2669 bp C G WW_cis 0.9439
A 2652 A 2668 bp C G WW_cis 0.9281
A 2655 A 2656 bp G U SH_cis 0.9227
A 2656 A 2665 bp U A WH_tran 0.8526
A 2657 A 2664 bp A G HS_tran 0.8513
A 2658 A 2663 bp C G WW_cis 0.9421
A 2659 A 2662 bp G A SH_tran 0.7619
```

but analyzed structures are like:

```
[mq] md_1msy_clx cat struc/1msy_rnakbmd_decoy1478_clx.pdb.out
Classifier: Clarna
chains: A 1 27

2 26 bp G U WW_cis 0.7196
3 25 bp C G WW_cis 0.6702
4 24 bp U A WW_cis 0.8911
5 23 bp C G WW_cis 0.8925
6 22 bp C G WW_cis 0.9026
9 10 bp G U SH_cis 0.8714
10 19 bp U A WH_tran 0.7279
11 18 bp A G HS_tran 0.8810
12 17 bp C G WW_cis 0.9115
```

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

Listing:

```bash
$ rna_pdb_toolsx.py --edit 'A:2647-2673>A:1:17' lmsy_clean.pdb > lmsy_clean_renumb.pdb

```

```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
```

For Educational Use Only
6.3 Example #3

Starting structure doesn’t have chain id:

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

```
/1duq_rnakbmn_decay0473Amb_clx/A/                     /1duq_rnakbmn_decay0008Amb_clx/A/
  1         6             5          11 14 16   21   26
  C U G G G F G C G A G G C U G G G G U C A A G C
```

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

You try to calculate RMSDs for RNA Puzzles #1:

```bash
rna_calc_rmsd.py -t 1_solution_0_rpr.pdb *.pdb
method: all-atom-built-in
# of models: 15
l_bujnicki_1_rpr.pdb 5.71 978
l_bujnicki_2_rpr.pdb 6.16 978
l_bujnicki_3_rpr.pdb 5.3 978
l_bujnicki_4_rpr.pdb 4.95 978
l_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.

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

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

you can tail the files:

```
tail *.pdb
```

```
=> l_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
...

=> l_chen_1_rpr.pdb <=
ATOM  971  N7  G  B  23  -14.462  -1.101  79.998  1.00  0.00  N
ATOM  972  C5  G  B  23  -14.952  -0.485  78.839  1.00  0.00  C
ATOM  973  C6  G  B  23  -15.577  -1.020  77.655  1.00  0.00  C
ATOM  974  O6  G  B  23  -15.822  -2.189  77.351  1.00  0.00  O
ATOM  975  N1  G  B  23  -15.972  -0.576  76.763  1.00  0.00  N
ATOM  976  C2  G  B  23  -15.787  1.274  76.944  1.00  0.00  C
ATOM  977  N2  G  B  23  -16.269  2.059  76.021  1.00  0.00  N
ATOM  978  N3  G  B  23  -15.224  1.822  78.022  1.00  0.00  N
ATOM  979  C4  G  B  23  -14.818  0.884  78.935  1.00  0.00  C
```
so now you can see that the files look the same. Let’s try to calculate RMSDs again:

```
rna_calc_rmsd.py -t 1_solution_0_rpr.pdb *.pdb
method: all-atom-built-in
# of models: 16
  l_bujnicki_1_rpr.pdb 5.71 978
  l_bujnicki_2_rpr.pdb 6.16 978
  l_bujnicki_3_rpr.pdb 5.3 978
  l_bujnicki_4_rpr.pdb 4.95 978
  l_bujnicki_5_rpr.pdb 5.1 978
  l_chen_1_rpr.pdb 4.35 978
  l_chen_1_rpr_v2.pdb 4.35 978
  l_das_1_rpr.pdb 3.97 978
  l_das_2_rpr.pdb 4.48 978
  l_das_3_rpr.pdb 3.43 978
  l_das_4_rpr.pdb 3.92 978
  l_das_5_rpr.pdb 4.57 978
  l_dokholyan_1_rpr.pdb 7.25 978
  l_major_1_rpr.pdb 4.34 978
  l_santalucia_1_rpr.pdb 5.76 978
  l_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.

---

6.4. Example #4 Calculate RMSDs of unstandardized structures (RNA Puzzle #1)
CHAPTER 7

Tips

7.1 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
# ex2
$ parallel "rna_clashscore.py {} > {}.csv" ::: *.pdb
```

7.2 Using sed

sed (stream editor) is a Unix utility that parses and transforms text, using a simple, compact programming language. You can used sed to find & replace parts of text files:

```
$ head lmsy_rnakbmd_decoy1661_clx.pdb.outCR
Classifier: Clarna
chains: 1 27
   2   26   bp G U     WW_cis  0.8500
   3   25   bp C G     WW_cis  0.8114
   4   24   bp U A     WW_cis  0.9222
   5   23   bp C G     WW_cis  0.9038
   6   22   bp C G     WW_cis  0.8913
```
9  10   bp G U     SH_cis  0.8563
10  19   bp U A     WH_tran 0.7826
11  18   bp A G     HS_tran 0.7620

$ sed 's/chains: /chains: A/' lmsy_rnakbmd_decoy1661_clx.pdb.outCR
Classifier: Clarna
chains: A 1 27
  2  26   bp G U     WW_cis  0.8500
  3  25   bp C G     WW_cis  0.8114
  4  24   bp U A     WW_cis  0.9222
  5  23   bp C G     WW_cis  0.9038
  6  22   bp C G     WW_cis  0.8913
  9  10   bp G U     SH_cis  0.8563
 10  19   bp U A     WH_tran 0.7826
 11  18   bp A G     HS_tran 0.7620
 12  17   bp C G     WW_cis  0.7242

Read more about sed.

### 7.3 In PyMOL

Quickref:

```plaintext
set ignore_case, off
```

Rename a chain:

```plaintext
PyMOL> alter (sele), chain="B"
Alter: modified 708 atoms.
PyMOL> sort
```

don’t forget about sort.

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

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

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

To renumber residues:

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

Read more.

The example of the pistol ribozyme editing.
Run:

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

To get residue index use:

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

Quickref:

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

Tips:

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

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

Numbering:

```
|1.......|10.......|20.......|30.......|40.......|50.......|60.......|70.......|80.......|90.......|
└──────────┘          └──────────┘          └──────────┘          └──────────┘          └──────────┘          └──────────┘          └──────────┘          └──────────┘          └──────────┘
123456789112345678921234567893123456789412345678951234567896123456789712345678981234567899123456789
```

7.7 TER format

Example of pro TER:

```
ATOM 72307 C4 U x 9 304.768 147.960 320.897 1.00218.84 C
ATOM 72308 O4 U x 9 304.171 146.902 321.104 1.00225.09 O
ATOM 72309 C5 U x 9 304.960 150.336 320.668 1.00205.76 C
ATOM 72310 C6 U x 9 304.190 149.269 320.912 1.00211.91 C
TER 72311 U x 9
```

7.8 Add missing atoms

Add missing atoms etc.
Read more:

- https://github.com/openmm/pdbfixer
RNA Puzzle Submission

The RNA Puzzle organizers required ONE file with your submissions in the NMR-style multiple model PDB format.

First, prepare your structures in the folder and run to get them RNApuzzle ready (_rpr):

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

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

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

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

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

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

```
$ 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
```

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

```
[mm] rnapuzzle_sender$ git:(master) ./rnapuzzle_sender.py
rnapuzzle_sender
Usage: rnapuzzle_sender.py
```
Options:
- `h, --help` show this help message and exit
- `d DIR, --dir=DIR`
- `s EMAIL_SUBJECT, --email_subject=EMAIL_SUBJECT`

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

```
In [2]: seq = Seq.Seq("GGUGCACGCGGCAGAAAGCGACAGUGCCGAAAGCUUGCAACUGCCUAAACCCCAACGAAGUGGS")
In [3]: seq
Out[3]: GGGUGCACGCGGCAGAAAGCGACAGUGCCGAAAGCUUGCAACUGCCUAAACCCCAACGAAGUGGS
```

### Secondary structure prediction

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

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

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

GGUGUGCACGCGGCAGAAAGCGACAGUGCCGAAAGCUUGCAACUGCCUAAACCCCAACGAAGUGGS -33.10 100
(((((...........)))(................))....)...)

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

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

In [15]: print seq.predict_ss(method='centroid_fold')

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

In [16]: print seq.predict_ss(method='contextfold')

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

9.1 Share your notebooks

You can share Jupyter notebooks from your Google Drive using the new Jupyter Drive. This allows you share Jupyter notebooks like NBViewer with all the access control that Google Drive provides. You can also push your notebooks to a Github repository, so then can be rendered for easy viewing.


9.2 Learn Jupyter

This is a quick introduction to jupyter which is the IPython version 3. It covers some of the new and interesting features about Jupyter.

https://www.youtube.com/watch?v=Rc4JQWowG5I
Warning: Proof of concept

RNA-tools can be used side-by-side https://github.com/mmagnus/emacs-pdb-mode to edit files structural files in the PDB format.

Emacs /imæks/ and its derivatives are a family of text editors that are characterized by their extensibility. 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/099YFbLSVRw
<table>
<thead>
<tr>
<th>Residue</th>
<th>Type</th>
<th>Chain</th>
<th>ID</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>HETATM</td>
<td>26</td>
<td>N2</td>
<td>GTP</td>
<td>X</td>
<td>10</td>
<td>-20.755</td>
<td>-4.612</td>
</tr>
<tr>
<td>HETATM</td>
<td>27</td>
<td>N3</td>
<td>GTP</td>
<td>X</td>
<td>10</td>
<td>-20.820</td>
<td>-6.290</td>
</tr>
<tr>
<td>HETATM</td>
<td>28</td>
<td>C4</td>
<td>GTP</td>
<td>X</td>
<td>10</td>
<td>-21.574</td>
<td>-6.775</td>
</tr>
<tr>
<td>ATOM</td>
<td>29</td>
<td>P</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-17.189</td>
<td>-6.642</td>
</tr>
<tr>
<td>ATOM</td>
<td>30</td>
<td>O1</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-16.067</td>
<td>-6.893</td>
</tr>
<tr>
<td>ATOM</td>
<td>31</td>
<td>O2</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-18.436</td>
<td>-6.124</td>
</tr>
<tr>
<td>ATOM</td>
<td>32</td>
<td>O3'</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-15.783</td>
<td>-5.042</td>
</tr>
<tr>
<td>ATOM</td>
<td>33</td>
<td>O5'</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-15.744</td>
<td>-5.986</td>
</tr>
<tr>
<td>ATOM</td>
<td>34</td>
<td>C5'</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-15.722</td>
<td>-5.012</td>
</tr>
<tr>
<td>ATOM</td>
<td>35</td>
<td>O4'</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-16.947</td>
<td>-5.133</td>
</tr>
<tr>
<td>ATOM</td>
<td>36</td>
<td>C3</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-15.685</td>
<td>-3.342</td>
</tr>
<tr>
<td>ATOM</td>
<td>37</td>
<td>O3'</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-14.365</td>
<td>-2.130</td>
</tr>
<tr>
<td>ATOM</td>
<td>38</td>
<td>C2'</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-16.159</td>
<td>-2.013</td>
</tr>
<tr>
<td>ATOM</td>
<td>39</td>
<td>O2'</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-15.151</td>
<td>-2.917</td>
</tr>
<tr>
<td>ATOM</td>
<td>40</td>
<td>C1'</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-17.280</td>
<td>-3.878</td>
</tr>
<tr>
<td>ATOM</td>
<td>41</td>
<td>N9</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-18.593</td>
<td>-3.458</td>
</tr>
<tr>
<td>ATOM</td>
<td>42</td>
<td>C8</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-19.297</td>
<td>-4.004</td>
</tr>
<tr>
<td>ATOM</td>
<td>43</td>
<td>N7</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-20.447</td>
<td>-3.426</td>
</tr>
<tr>
<td>ATOM</td>
<td>44</td>
<td>C7</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-20.512</td>
<td>-2.438</td>
</tr>
<tr>
<td>ATOM</td>
<td>45</td>
<td>C6</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-21.528</td>
<td>-1.488</td>
</tr>
<tr>
<td>ATOM</td>
<td>46</td>
<td>O6</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-27.606</td>
<td>-1.320</td>
</tr>
<tr>
<td>ATOM</td>
<td>47</td>
<td>N1</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-21.194</td>
<td>-0.655</td>
</tr>
<tr>
<td>ATOM</td>
<td>48</td>
<td>C2</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-20.027</td>
<td>-0.748</td>
</tr>
<tr>
<td>ATOM</td>
<td>49</td>
<td>N2</td>
<td>G</td>
<td>X</td>
<td>11</td>
<td>-19.880</td>
<td>0.141</td>
</tr>
</tbody>
</table>
Warning: Proof of concept

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:

```markdown
[ss: rna]
UUUCUGUAUAUGCCGAUAUAAGGUUCGGCAGUUUCUACCAAACAGCCGUAAACUGUUUGACUACAGUAA
{.((.(((...)))))...((((((.))))))..)}..)
</pre>
```

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

The syntax:

```markdown
<pre>
[ss:/name of your seq/]
</pre>
```
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.
To install kind of vanilla version rna-tools by use pip:

$ pip install rna-tools

Test if the package is installed:

$ python -c 'import rna_tools'

if nothing appears that's good. If you see this:

$ python -c 'import rna_tools'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ImportError: No module named rna_tools

then go to custom installation: <http://rna-tools.readthedocs.io/en/latest/install-dev.html>
13.1 Pip as a developer

You can get rna-tools and install them from the current directory with this pip:

```bash
pip install -e git+http://github.com/mmagnus/rna-tools.git#egg=rna-tools
```

This way is better (than `pip install rna-tools`) if you’re going to do some coding in the tools.

You can even first get rna-tools with git and then do:

```bash
$ pip install -e .
Obtaining file:///Users/magnus/work/src/rna-tools
```

to “install” with pip rna-tools based on a set of links to the current directory.

13.2 Test if installed

Test if the package is installed:

```bash
$ python -c 'import rna_tools'
# ipython -c 'import rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd' (to test specific tools)
```

if nothing appears that’s good. If you see this:

```bash
$ python -c 'import rna_tools'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ImportError: No module named rna_tools
```

then go to your config file for the shell (~/.zshrc or ~/.bashrc) add add:
```bash
PYTHONPATH=<path to rna-tools>:/$PYTHONPATH
export PYTHONPATH

so in my case this is:

PYTHONPATH=<path to rna-tools>/home/magnus/src/rna-tools:/$PYTHONPATH
export PYTHONPATH
```

### 13.3 Configuration

To set up your own configuration, create ~/.rna_tools.py in your HOME directory and redefine variables, e.g.:

```python
(py37) [mx] rna-tools$ git:(master) cat ~/.rna_tools.py
VARNA_PATH = '/Users/magnus/work/opt/varna'
VARNA_JAR_NAME = 'VARNA.jar'
SIMRNA_DATA_PATH = '/Users/magnus/work/opt/simRNA/SimRNA_64bitIntel_MacOSX_staticLibs/
data'
QRNAS_PATH = '/Users/magnus/work/opt/qrnas/
RCHIE_PATH = '/Users/magnus/work/opt/r-chie/
RFAM_DB_PATH = '/Users/magnus/work/db/rfam/Rfam.cm'
CONTEXTFOLD_PATH = '/Users/magnus/work/opt/ContextFold_1_00/
DIFF_TOOL = "open -a diffmerge"
CPUS_CLUSTER = 630
RNA_STRUCTURE_PATH = '/Users/magnus/work/opt/RNAstructure/6.1/
ENTRNA_PATH = '/Users/magnus/work/opt/ENTRNA"
```

### 13.4 All requirements

To get ALL requirements, use pip:

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

Be default rna-tools will not install all requirements, because some of them are heavy or might cause various problems, so you will be asked to install them when needed. For example, installing `matplotlib` is not essential for many other tools, `python-Levenshtein` is only used in one function.

### 13.5 Test all

This is still under active development.

To test (almost) all rna-tools functionality, you can run rna_tools_test_all.py to see if you got any errors, this should look like:

```bash
(py37) [mm] rna-tools$ git:(master) rna_tools_test_all.py
BlastPDB requires urllib3
Python: 3.7.4 (default, Aug 13 2019, 15:17:50) [Clang 4.0.1 (tags/RELEASE_401/
final)]
RNA_TOOLS_PATH set to /home/magnus/work-src/rna-tools/
```
or for Python 2:

```bash
(base) [mm] rna-tools$ git:(master) rna_tools_test_all.py
- Python: 2.7.16 |Anaconda, Inc.| (default, Mar 14 2019, 16:24:02) [GCC 4.2.1Compatible Clang 4.0.1 (tags/RELEASE_401.final)]
- rna-tools: py2-78-g3b3dd5f
- RNA_TOOLS_PATH set to /home/magnus/work-src/rna-tools/
- See full list of tools <https://github.com/mmagnus/rna-tools/blob/master/rna-tools--index.csv
Seems OK
```

For crude testing you can also use `./test.sh` script and then see for errors in output and also check output/ folder to see if there are differences between your output and output committed to GitHub by me.

### 13.6 PyMOL4RNA: adv config

For some extra functions you might also follow this. Open your `~/.pymolrc` and set up following variables as you need:

```bash
# rna-tools
EXECUTABLE="/bin/zsh" # set up your shell, usually /bin/bash or /bin/zsh
SOURCE="source ~/.zshrc" # set up the path to the file where you keep your shell variables

CLARNA_RUN="/Users/magnus/work-src/clarna_play/clarna_run.py" # if you want to run clarna_run.py set up the path

run ~work-src/rna-tools/rna_tools/tools/pymol_drawing/pymol_drawing.py
run ~work-src/rna-tools/rna_tools/tools/rna_filter/pymol_dists.py
```

The plugins have been tested with MacPyMOL version 1.7.4.5 Edu.
For Git in a scientific environment, please read:


Git is a version control system that is used for software development that helps you to keep track of versions of your program. To start using git you have to know only these two commands below. If you want to contribute to the package you need a few more, but it’s not important right now :-)

To get the package for the first time on your computer go to the selected package and on the top left corner select clone or download. Copy the link and run it on console with git clone:

```
$ git clone git@github.com:mmagnus/rna-pdb-tools.git
```

and if you want to update the package later run:

```
$ git pull # be in the folder like ~/src/rna-pdb-tools/ <here>
```

if you see something like this:

```
$ git pull
Already up-to-date.
```

it means that your version of the package is up to date, congrats! :-)
If you see something like this:

```bash
$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 2), reused 3 (delta 2), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:mmagnus/rna-pdb-tools
  69c4ee3..7f90739 master     -> origin/master
Updating 69c4ee3..7f90739
Fast-forward
install_links_bin.sh | 1 +
1 file changed, 1 insertion(+)
```

it means that there is a small change in `install_links_bin.sh` and you are up to date, congrats as well! You might need to run `./install_links_bin.sh` to “install” new tools that were added to the packages (if this is the case). If you get any error then talk to me magnus@genesilico.pl.

### 14.1 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.
```

### 14.2 Learn Git

Model used by Git vs Svn:

```
file <-> stage area <-> local repo <-> git repo/gitlab
file ----------------------------------------> SVM repo
```
For more see, git - the simple guide (just a simple guide for getting started with git. no deep shit :))

http://rogerdudler.github.io/git-guide/


If you want to read more, we highly recommend a book our collages, Kristian (http://www.apress.com/us/book/9781484222409).

To understand the principles of Git, a bit more advance, by Linus Torvals (an author of Git) https://www.youtube.com/watch?v=4XpnKHJAok8

How to learn Git in 20min https://www.youtube.com/watch?v=Y9XZQO1n_7c

14.3 Git GUI

You don’t have to use terminal to work with git. Git comes with git gui.
You can also use qgit (http://sourceforge.net/projects/qgit/) and much more, a list of tools: https://git-scm.com/download/gui/linux.
Keep configuration syntax like:

```python
from rna_tools.rna_tools_config import CPUS_CLUSTER
# since we use export PYTHONPATH=$PYTHONPATH:/home/magnus/src/rna-tools/
```

vs:

```python
try:
    RNA_ROSETTA_RUN_ROOT_DIR_MODELING = os.environ['RNA_ROSETTA_RUN_ROOT_DIR_MODELING']
except:
    print ('Set up RNA_ROSETTA_RUN_ROOT_DIR_MODELING in .bashrc')
```
We are using (at least we are moving towards) the Google style docstrings via Napoleon. Napoleon is a Sphinx Extensions that enables Sphinx to parse both NumPy and Google style docstrings - the style recommended by Khan Academy. http://www.sphinx-doc.org/en/stable/ext/napoleon.html#type-annotations

16.1 Generate docs

At the moment the generation of the docs is quite tricky, because of old dependencies on Python2 and old version of Sphinx. To generate docs locally please switch to Python2 and then run `cd docs/; make clean; make html`.

It should be updated at some point.
1. Create a new folder in `rna-tools/rna_tools/tools` with your tool. The folder will be seen online after your push at https://github.com/mmagnus/rna-tools/tree/master/rna_tools/tools. We will walk you through this simple example https://github.com/mmagnus/rna-tools/tree/master/rna_tools/tools/renum_pdb_to_aln.

2. Make sure that there is a simple test as `test.sh`:

```bash
#!/bin/bash
python renum_pdb_to_aln.py --residue_index_start 1 obj1 test_data/ALN_OBJ1_OBJ2.fa test_data/obj01.pdb
```

and there is a `test_data` folder with some test inputs and outputs. See the example.

3. Add your tool to `install_links_bin.sh` at the top folder of `rna-tools`:

```bash
ln -s $curr_dir/rna_tools /tools/<tool folder>/<util script name with .py> $curr_dir/bin/<util script name with .py>
```

   e.g.

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

```rst
Renumber a pdb file according to alignment
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.. argparse::
    :ref: rna_tools.tools.<tool folder>.<tool script name>.get_parser
    :prog: <util script name>
```
.. automodule:: rna_tools.tools.<tool folder>.<tool script name>
   :members:

e.g.:

Renumber a pdb file according to alignment
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.. argparse::
   :ref: rna_tools.tools.renum_pdb_to_aln.renum_pdb_to_aln.get_parser
   :prog: renum_pdb_to_aln

.. automodule:: rna_tools.tools.renum_pdb_to_aln.renum_pdb_to_aln
   :members:

and run `make html` in the folder to check if the documentation is compiled without any errors.

If you are using any external library such as `scipy`, please make sure that they are listed in `rna-tools/docs/requirements.txt`. If the library is not there, please add it. This file is read by the Read The Docs to compile the documentation online and also by Travis for continuous testing.

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

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

```bash
cd ./tools/<tool folder>/
./test.sh
cd ../..
```

6. Run this main test (`./test.sh`) and see if the tool works as expected.

7. Now we are ready to push the changes. In the terminal, type:

```bash
$ git pull
$ git add <files> # or use git gui
$ git commit -m <desc the tool>
$ git push
```

to commit all your changes and push it to the Github repository!

**Warning:** This testing is very, very rough and we are moving to have more test in py.test at some point.
CHAPTER 18

Indices and tables

• genindex
• modindex
• search
Python Module Index

r
rna_tools.BlastPDB, 42
rna_tools.RfamSearch, 42
rna_tools.rna_tools_lib, 21
rna_tools.SecondaryStructure, 38
rna_tools.Seq, 33
rna_tools.tools.clanstix.rna_clanstix, 115
rna_tools.tools.clarna_app.clarna_app, 85
rna_tools.tools.cluster_load.cluster_load, 121
rna_tools.tools.diffpdb.diffpdb, 112
rna_tools.tools.extra_functions.select_fragment, 135
rna_tools.tools.misc.rna_add_chain, 30
rna_tools.tools.mq.AnalyzeGeometry.AnalyzeGeometry, 94
rna_tools.tools.mq.ClashScore.ClashScore, 93
rna_tools.tools.mq.Dfire.Dfire, 87
rna_tools.tools.mq.eSCORE.eSCORE, 90
rna_tools.tools.mq.FARNA.FARNA, 92
rna_tools.tools.mq.QRNA.QRNA, 89
rna_tools.tools.mq.RASP.RASP, 86
rna_tools.tools.mq.RNA3DCNN.RNA3DCNN, 91
rna_tools.tools.mq.RNAkb.RNAkb, 87
rna_tools.tools.mq.RNAScore.RNAScore, 90
rna_tools.tools.pdbs_measure_atom_dists.pdbs_measure_atom_dists, 31
rna_tools.tools.pymol_drawing.pymol_drawing, 131
rna_tools.tools.renum_pdb_to_aln.renum_pdb_to_aln, 66
rna_tools.tools.rna_alignment.random_assignment_of_nucleotides, 62
rna_tools.tools.rna_alignment.rna_alignment,
A
aa3to1() (in module rna_tools.rna_tools_lib), 27
add_ids() (rna_tools.tools.clanstix.rna_clanstix.RNAStructClans
method), 117
align_seq() (rna_tools.tools.rna_alignment.rna_alignment.RNAalignment
method), 49
AnalyzeGeometry (class in
rna_tools.tools.mq.AnalyzeGeometry.AnalyzeGeometry), 94
Atom (class in rna_tools.simrna_trajectory.simrna_trajectory), 107

B
best_energy (rna_tools.tools.mq.FARNA.FARNA.FARNA
attribute), 92
BlastPDB (class in rna_tools.BlastPDB), 42

C
calc_rmsd() (in module
rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd), 70
calc_rmsd() (in module
rna_tools.tools.rna_calc_rmsd_all_vs_all), 75
calc_rmsd_pymol() (in module
rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd), 70
check_symmetric() (in module
rna_tools.tools.clanstix.rna_clanstix), 117
clarna_compare() (in module
rna_tools.tools.clarna_app.clarna_app), 85
clarna_run() (in module
rna_tools.tools.clarna_app.clarna_app), 86
ClashScore (class in rna_tools.tools.mq.ClashScore.ClashScore), 93
cleanup() (rna_tools.tools.mq.ClashScore.ClashScore.ClashScore
method), 93
cleanup() (rna_tools.tools.mq.FARNA.FARNA.FARNA
method), 92
cluster() (in module rna_tools.tools.rna_rosetta.rna_rosetta_cluster), 102
cluster_loop() (in module
rna_tools.tools.rna_rosetta.rna_rosetta_cluster), 102
CMAlign (class in rna_tools.tools.rna_alignment.rna_alignment), 63
cmscan() (rna_tools.RfamSearch.RfamSearch method), 43
collapsed_view() (in module rna_tools.rna_tools_lib), 27
colour_by_local_score() (rna_tools.tools.mq.RASP.RASP.RASP
method), 86
copy_ss_cons_to_all() (rna_tools.tools.rna_alignment.rna_alignment.RNAAlignment
method), 49
copy_ss_cons_to_all_editing_sequence()
(rna_tools.tools.rna_alignment.rna_alignment.RNAAlignment
method), 49
CustomFormatter (class in
rna_tools.tools.mq.ClashScore.ClashScore.ClashScore), 98

D
db_path (rna_tools.tools.mq.FARNA.FARNA.FARNA
attribute), 92
describe() (rna_tools.tools.rna_alignment.rna_alignment.RNAAlignment
method), 49
Dfire (class in rna_tools.tools.mq.Dfire.Dfire), 87
dist_from_matrix() (rna_tools.tools.clanstix.rna_clanstix.RNAStructClans
method), 117
dist_from_matrix_mp() (rna_tools.tools.clanstix.rna_clanstix.RNAStructClans
method), 117
do_inh() (in module rna_tools.tools.rna_calc_inf.rna_calc_inf), 77
get_distance_to() (rna_tools.tools.rna_alignment.rna_alignment method), 45
get_distances() (rna_tools.tools.rna_alignment.rna_alignment method), 50
get_dot_bracket_from_ClaRNAoutput() (in module rna_tools.tools.clarna_app.clarna_app), 86
get_foldability() (rna_tools.Seq.RNASequence method), 35
get_gc_rf() (rna_tools.tools.rna_alignment.rna_alignment.CMAAlign method), 41
get_gc_rf() (rna_tools.tools.rna_alignment.rna_alignment.RNAalignment method), 51
get_info_chains() (rna_tools.rna_tools_lib.RNAStructure method), 22
get_ion_water_report() (rna_tools.tools.rna_x3dna.rna_x3dna.x3DNA method), 102
get_modifications() (rna_tools.tools.rna_x3dna.rna_x3dna.x3DNA method), 5, 85
get_multiple_lines() (in module rna_tools.tools.rna_convert_pseudoknot_formats.rna_ss_pk_to_simrna), 41
get_no_structures() (in module rna_tools.tools.rna_rosetta.rna_rosetta_cluster), 102
get_no_structures() (in module rna_tools.tools.rna_rosetta.rna_rosetta_min), 104
get_no_structures() (in module rna_tools.tools.rna_rosetta.rna_rosetta_n), 100
get_no_structures_in_first_cluster() (in module rna_tools.tools.rna_rosetta.rna_rosetta_cluster), 98
get_parser() (in module rna_tools.tools.clanstix.rna_clanstix), 117
get_parser() (in module rna_tools.tools.clarna_app.clarna_app), 86
get_parser() (in module rna_tools.tools.cluster_load.cluster_load), 121
get_parser() (in module rna_tools.tools.mq.FARNA.FARNA), 92
get_parser() (in module rna_tools.tools.mq.FARNA.FARNAMethod), 98
get_parser() (in module rna_tools.tools.mq.FARNA.FARNAMethod), 98
get_parser() (in module rna_tools.tools.mq.FARNA.FARNAMethod), 98
get_result() (rna_tools.tools.mq.FARNA.FARNA, 92
get_rna_models_from_dir() (in module rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd), 75
get_res_code() (rna_tools.tools.rna_calc_inf.rna_calc_inf), 77
get_res_text() (rna_tools.tools.rna_calc_inf.rna_calc_inf), 77
get_res_text() (rna_tools.tools.rna_alignment.random_assignment_of_nucleotides), 63
get_result() (rna_tools.tools.mq.FARNA.FARNA.FARNA method), 92
get_result() (rna_tools.tools.mq.FARNA.FARNA.FARNA method), 92
get_result() (rna_tools.tools.mq.FARNA.FARNA.FARNA method), 92
get_result() (rna_tools.tools.mq.FARNA.FARNA.FARNA method), 92
get_rna_models_from_dir() (in module rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd),
<table>
<thead>
<tr>
<th>Name</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_seq_len</td>
<td>rna_tools.tools.mq.Dfire.Dfire.Dfire attribute, 87</td>
</tr>
<tr>
<td>max_seq_len (RNA3DCNN)</td>
<td>rna_tools.tools.mq.RNA3DCNN.RNA3DCNN.RNA3DCNN attribute, 92</td>
</tr>
<tr>
<td>MethodNotChosen</td>
<td></td>
</tr>
<tr>
<td>min() (RChie)</td>
<td></td>
</tr>
<tr>
<td>mqap()</td>
<td>rna_tools.tools.mq.FARNA.FARNA.FARNA method, 92</td>
</tr>
<tr>
<td>N</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>rna_tools.ma_tools_lib.RNAStructure attribute, 16, 19, 21</td>
</tr>
<tr>
<td>O</td>
<td></td>
</tr>
<tr>
<td>open_pdb()</td>
<td>rna_tools.tools.pdbs_measure_atom_dists.pdbs_measure_atom_dists, 32</td>
</tr>
<tr>
<td>open_pdb()</td>
<td>rna_tools.tools.renum_pdb_to_aln.renum_pdb_to_aln, 67</td>
</tr>
<tr>
<td>P</td>
<td></td>
</tr>
<tr>
<td>parse_vienna_to_pairs()</td>
<td>rna_tools.SecondaryStructure, 39</td>
</tr>
<tr>
<td>per_user()</td>
<td>rna_tools.tools.cluster_load.cluster_load, 121</td>
</tr>
<tr>
<td>plot()</td>
<td>rna_tools.rna_alignment.rna_alignment.RNAAlignment method, 52</td>
</tr>
<tr>
<td>plot_cov()</td>
<td>rna_tools.rna_alignment.rna_alignment.RChie method, 66</td>
</tr>
<tr>
<td>plot_energy()</td>
<td>rna_tools.simrna_trajectory.simrna_trajectory.rsimrna_trajectory.method, 108</td>
</tr>
<tr>
<td>predict_ss()</td>
<td>rna_tools.Seq.RNASequence method, 36</td>
</tr>
<tr>
<td>prepare_folder()</td>
<td>rna_tools.tools.rna_rosetta.rna_rosetta_run, 98</td>
</tr>
<tr>
<td>prepare_groups()</td>
<td>rna_tools.tools.rnakb_utils.rnakb_utils, 122</td>
</tr>
<tr>
<td>prepare_helices()</td>
<td>rna_tools.tools.rna_rosetta.rna_rosetta_run, 99</td>
</tr>
<tr>
<td>prepare_rosetta()</td>
<td>rna_tools.tools.rna_rosetta.rna_rosetta_run, 99</td>
</tr>
<tr>
<td>program_name</td>
<td>rna_tools.tools.mq.FARNA.FARNA.FARNA attribute, 92</td>
</tr>
<tr>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>QRNA</td>
<td>rna_tools.tools.mq.QRNA.QRNA.QRNA, 89</td>
</tr>
</tbody>
</table>

**Index**

| RASP (RASP)          | 86                                                      |
| RChie (RChie)        | 65                                                      |
| reload_alignment()   | rna_tools.tools.rna_alignment.rna_alignment.RNAAlignment method, 52 |
| remove()             | rna_tools.ma_tools_lib.RNAStructure method, 52           |
| remove_columns()     | rna_tools.tools.rna_alignment.rna_alignment.RNASeq method, 45 |
| remove_empty_columns() | rna_tools.tools.rna_alignment.rna_alignment.RNAAlignment method, 52 |
| remove_gaps()        | rna_tools.ma_tools_lib.RNAStructure method, 52           |
| remove_ion()         | rna_tools.ma_tools_lib.RNAStructure method, 52           |
| remove_water()       | rna_tools.ma_tools_lib.RNAStructure method, 52           |
| rename_chain()       | rna_tools.ma_tools_lib.RNAStructure method, 52           |
| repl()               | rna_tools.tools.rna_sali2dotbracket.rna_sali2dotbracket, 120 |
| remove()             | rna_tools.tools.rna_alignment.rna_alignment.RNASeq method, 45 |
| RfamSearch (RfamSearch) | 42                                                  |
| RfamSimRNAtrajectory |                                                                        |
| rmsd_to()            | rna_tools.simrna_trajectory.simrna_trajectory.Frame.method, 107     |
| RNA3DCNN (RNA3DCNN)  | rna_tools.ma_tools_lib.RNA3DCNN.RNA3DCNN, 92             |

| RNA3DCNN (RNA3DCNN)  | rna_tools.BlastPDB (module), 42                           |
| RNA3DCNN (RNA3DCNN)  | rna_tools.RfamSearch (module), 42                         |
| RNA3DCNN (RNA3DCNN)  | rna_tools.ma_tools_lib (module), 21                       |
| RNA3DCNN (RNA3DCNN)  | rna_tools.SecondaryStructure (module), 38                 |
| RNA3DCNN (RNA3DCNN)  | rna_tools.Seq (module), 33                                |
| RNA3DCNN (RNA3DCNN)  | rna_tools.tools.clanstix.rna_clanstix (module), 115       |
| RNA3DCNN (RNA3DCNN)  | rna_tools.tools.clarna_app.clarna_app (module), 85        |
| RNA3DCNN (RNA3DCNN)  | rna_tools.tools.cluster_load.cluster_load (module), 121   |
| RNA3DCNN (RNA3DCNN)  | rna_tools.tools.difpdb.difpdb (module), 112               |
| RNA3DCNN (RNA3DCNN)  | rna_tools.tools.extra_functions.select_fragment (module), 135 |
| RNA3DCNN (RNA3DCNN)  | rna_tools.tools.misc.rna_add_chain (module), 30           |
| RNA3DCNN (RNA3DCNN)  | rna_tools.ma_tools análisisGeometry.AnalyzeGeometry (module), 94 |
| RNA3DCNN (RNA3DCNN)  | rna_tools.ma_tools.ClarinScore.ClarinScore (module), 93   |

**RNA3DCNN (RNA3DCNN)**

| RNA3DCNN (RNA3DCNN)  | rna_tools.ma_tools.Dfire.Dfire (module), 87               |

**RNA3DCNN (RNA3DCNN)**

| RNA3DCNN (RNA3DCNN)  | rna_tools.ma_tools.Dfire.Dfire (module), 87               |
seq_no_gaps (rna_tools.tools.rna_alignment.rna_alignment.RNASeq attribute), 44
set_atom_occupancy () (rna_tools.rna_tools_lib.RNAStructure method), 26
set_chain_for_struct () (in module rna_tools.rna_tools_lib), 28
set_occupancy_atoms () (rna_tools.rna_tools_lib.RNAStructure method), 26
set_res_code () (rna_tools.rna_tools_lib.RNAStructure method), 26
set_res_code () (rna_tools.makb_utils.makb_utils), 123
show () (rna_tools.tools.rna_alignment.rna_alignment.RNASeq method), 48
SimRNATrajectory () (class in rna_tools.simrna_trajectory.simrna_trajectory), 108
sort () (rna_tools.tools.rna_alignment.rna_alignment.RNASeq method), 53
sort_nicely () (in module rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd), 71
sort_nicely () (in module rna_tools.tools.rna_calc_rmsd.rna_calc_rmsd_all_vs_all), 75
sort_strings () (module rna_tools.rna_tools_lib), 28
src_bin (rna_tools.tools.mq.FARNA.FARNA attribute), 93
ss_cons_std (rna_tools.tools.rna_alignment.rna_alignment.RNAAlignment attribute), 52
ss_cons_with_pk (rna_tools.tools.rna_alignment.rna_alignment.RNAAlignment attribute), 52
ss_cons_with_pk_std (rna_tools.tools.rna_alignment.rna_alignment.RNAAlignment attribute), 52
ss_no_gaps (rna_tools.tools.rna_alignment.rna_alignment.RNASeq attribute), 44
ss_to_bps () (rna_tools.tools.rna_alignment.rna_alignment.RNASeq method), 48
stats_for_cluster () (in module rna_tools.cluster_load.cluster_load), 121
stats_for_user () (in module rna_tools.cluster_load.cluster_load), 121
std_resn () (rna_tools.rna_tools_lib.RNAStructure method), 26
subset () (rna_tools.tools.rna_alignment.rna_alignment.RNAAlignment method), 52

tail () (rna_tools.tools.rna_alignment.rna_alignment.RNAAlignment method), 53
test () (module rna_tools.tools.mq.ClashScore.ClashScore), 94