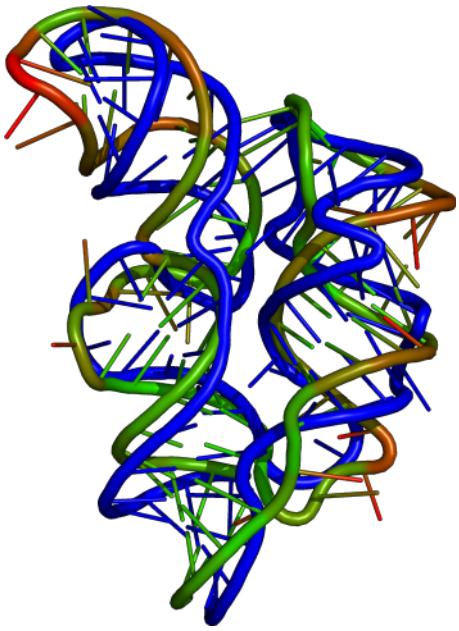

rna_predict Documentation

Release 0.1

Sebastian Ratz

September 30, 2015

Contents

The `rna_predict` Python package models a complete workflow for RNA tertiary structure prediction using the secondary structure information as well as optional, additional constraints.

The modelling is done using Rosetta and its `rna_denovo` protocol.

Additional constraints, for example from co-evolutional anaylsis, can be included to improve the prediction quality.

A utility to convert residue-residue contacts into atom-atom constraints is included.

`rna_predict` also comes with tools to cluster and visualize the prediction results and to compare them to a native crystal strcture, if available.

CHAPTER 1

Contents

1.1 Rosetta compilation and installation

`rna_predict` relies on a few modifications of the Rosetta source code to work.

1.1.1 Get Rosetta

Download Rosetta from the official website. The latest set of patches targets build `2015wk19` so it's advised to get a fairly recent weekly build.

1.1.2 Apply patches

The needed patches are included in the `contrib/rosetta_patches` directory.

To apply them `cd` to the unpacked Rosetta source and run:

```
patch -p1 < /path/to/patches/*.patch
```

If they don't apply cleanly, for example when targeting a newer version of Rosetta, they need to be applied manually. The changes are only minor so it should be possible to do without much effort.

1.1.3 Build and install

Build Rosetta normally according the official documentation.

1.1.4 `PATH` setup

`rna_predict` needs to find the Rosetta binaries as well as some of the helper scripts from the `tools` package.

Also, `rna_predict` does not specify the `-database` parameter when invoking Rosetta commands, so the database needs to reside in the default location or the `ROSETTA3_DB` environment variable needs to be set correctly.

The following lines in `.bashrc` or similar should be sufficient to make everything work:

```
export ROSETTA3_DB=/path/to/rosetta_database
export RNA_TOOLS=/path/to/rosetta_tools/rna_tools
export PYTHONPATH=$PYTHONPATH:$RNA_TOOLS/bin/
export PATH=$PATH:/path/to/rosetta_source/bin:$RNA_TOOLS/bin
```

1.2 Configuration

`rna_predict` reads the config file `~/.rna_predict/sysconfig` if present.

1.2.1 Rosetta path and suffix

By default, `rna_predict` searches for the Rosetta binaries in the `PATH` environment variable and assumes *.linuxgccrelease* as binary suffix. This can be overridden in the config file as follows:

```
[rosetta]
exe_path = /path/to/rosetta/directory/bin/
exe_suffix =
```

1.2.2 Other options

Currently, there is only one other option to control buffering of the output of called programs. To enable line-buffered output, set the following:

```
[rna_predict]
subprocess_buffsize = L
```

Note that this is done by prefixing all commands with `stdbuf -<option>` and therefore requires the `stdbuf` program to be available.

1.3 Usage

`rna_predict` comes with a command line interface. To see all available options run:

```
rna_predict --help
```

1.3.1 Peparation workflow

These steps need to be run only once for every structure.

1. Directory Preparation

The first step is to prepare a dedicated prediction directory. You need to have at least a fasta file containing the sequence and another file containing the secondary structure (in dot-bracket notation). Additionally, if available, you can supply a PDB file containing the native crystal structure. To prepare the directory, run the following:

```
rna_predict prepare \
    --name <name> \
    --sequence <fasta_file> \
    --secstruct <secstruct_file> \
    --native <native_pdb_file>
```

All the options are optional and default values are used if not given. See the output of `rna_predict prepare --help` for additional information.

This will parse out stems and motifs into the `preparation` subdirectory.

Note: When using the `--native` option, make sure the PDB file has neen adjusted to work with rosetta. This means that the residues have to be reordered starting at 1, so that Rosetta's `pdbslice.py` program uses the corrent ones. To achieve this, prepare the file using the Rosetta tool:

```
make_rna_rosetta_ready.py <native_pdb>
```

2. Create ideal helix models

For each helical part of the structure, an ideal helix model needs to be generated:

```
rna_predict create-helices
```

1.3.2 Main workflow

1. Constraints creation

When tertiary constraints are to be used, `.cst` file should be created and put int the `constraints` subdirectory. The syntax is explained in the Rosetta documentation.

The file can be created by hand or using the following tools:

1a. Generation from DCA files

To create a tertiary constraints from DCA predictions, use the following command:

```
rna_predict make-constraints \
    --dca-file <dca.dat> \
    --mapping-mode <mapping_mode> \
    --dca-count <count> \
    --cst-function <function> \
    --cst-out-file <output_name> \
    --filter <filter>
```

The input dca file should contain at least two columns containing the residue numbers of the contact. The file should be sorted by DCA score. Optionally, the first line of the file may contain a comment specifying a PDB-mapping such as the following:

```
# pdb-mapping: 10-12,44,80-90
```

This defines how the residue numbers in the DCA file are to be mapped. Rosetta always uses 1,2,3,... internally, so the mapping above would, for example, result in the residue number 12 in the DCA file to be mapped to prediction residue 3.

The `--mapping-mode` parameter specifies the method to map residue-residue contacts to atom-atom contacts. Options are:

- minAtom
- pOnly

For details about the mapping, see *Residue-residue to atom-atom mapping*.

The `--dca-count` option limits the number of predictions in the DCA input file.

The `--cst-function` sets the Rosetta function to use. See https://www.rosettacommons.org/docs/latest/constraint-file.html#Function-Types for details. The default function for constraints creation (`FADE -100 26 20 -2 2`) uses

a spline smoothed square well potential (represented by the "FADE" function) and a default parameter set. After the generation of the cst file, it can of course be fine-tuned by further modification in any text editor.

The `--cst-out-file` option specifies an output filename.

The `--filter` option allows the DCA contacts to be passed through a chain of filters first. For the filter documentation see *DCA Filtering Syntax*.

1b. Editing existing file

To simply replace the Rosetta function in an existing `.cst` file you can use:

```
rna_predict edit-constraints \
    --cst <input_cst> \
    --cst-function <function> \
    --cst-out-file <output_name>
```

For option explanation see above.

This is pretty much the same as using search-and-replace in any text editor.

2. Prepare constraints

To run a simulation with a specific set of constraints (or none), another preparation step needs to be run:

```
rna_predict prepare-cst \
    --cst <cst> \
    --override-motifs-cst <motif_cst>
```

The `--cst` option selects the constraints from the `constraints` directory to be prepared. If not given, a prediction called 'none' for no tertiary constraints is created.

Optionally, it is possible to use a different set of motifs for the assembly. For example you can create a common set of motif models and use this in all future assemblies. To do this, specify the `--override-motifs-cst` option.

3. Motif creation

For all non-helical parts (loop regions, etc.) multiple models need to be created. To do this, run the following:

```
rna_predict create-motifs \
    --cst <cst> \
    --cycles <cycles> \
    --nstruct <nstruct> \
    --seed <random_seed> \
    --use-native
```

As always, `--cst` selects the constraints.

The `--cycles` option sets the number of monte-carlo cycles to run for generating each model.

The `--nstruct` option sets the number of models created for each motif.

To override the initial random seed, you can specify `--seed`.

And to have Rosetta automatically calculate RMSD values to a native structure you can supply the `--use-native` option.

4. Assembly

To combine helix and motif models an assembly simulation is run:

```
rna_predict assemble \
    --cst <cst> \
    --cycles <cycles> \
    --nstruct <nstruct> \
    --seed <random_seed> \
    --use-native
```

The options are the same as the ones for `create-motifs`, but their default values vary.

Note: The assembly step does not check how many models have already been created so far.

5. Evaluation

5a. Evaluation using Rosetta clustering and scoring

When the assembly has finished, you can evaluate the simulation. This means:

- Cluster the models
- Calculate RMSD values to the native structure, if available, and to the model with the best score.

Usage:

```
rna_predict evaluate \
    --cst <cst> \
    --cluster-cutoff <cutoff> \
    --cluster-limit <limit> \
    --full-eval
```

The `--cluster-cutoff` option specifies the RMSD radius in angstrom after which to create a new cluster.

The `--cluster-limit` option limits the maximum number of clusters to be created.

The `--full-eval` option forces the whole evaluation to be run again, and ignore any previous results stored.

5b. Custom scoring

Due to the fact that DCA predictions are not perfect, a custom scoring method was created. For each DCA prediction neighboring residues are included and if the distance between any of these residue paris are in contact the score is increased. Usage:

```
rna_predict evaluate-custom \
    --cst <cst> \
    --dca-file <dca_file> \
    --dca-count <count> \
    --radius <radius> \
    --threshold <threshold> \
    --full-eval
```

For the `--dca-file` and `--dca-count` options see `make-constraints`.

The `--radius` option sets the number of neighboring residues to take into account.

The `--threshold` option sets the distance threshold under which a residue pair is treated as in-contact.

1.3.3 Utilities

Status information

To print a summary of all predictions and their current state, run:

```
rna_predict status \
    [--compare] \
    [cst [cst ...]
```

The output table contains the following columns:

- P: preparation step
- M: motif generation:
- A: assembly:
- E: evaluation

If a step is completed, *X* is shown, - otherwise. For motif generation a * * * may be shown to indicate that models from a different set of constraints are used.

When the `--compare` option is given, comparison to the native structure is performed and the output is extended with the following columns:

- 1: Native RMSD score of the first cluster
- 5: Lowest native RMSD score of the first 5 clusters
- 10: Lowest native RMSD score of the first 10 clusters
- n: Number of models

Model information and extraction

To print model information or extract PDB files use the following subcommands:

```
rna_predict print-models|extract-models \
    --cst <cst> \
    --mode <selection_mode> \
    model [model ...]
```

The `--mode` option selects the way to look up models:

- cluster: Cluster number to reference the cluster primary model
- cluster_ntop: Clusters sorted by the RMSDs of their representatives
- ntop: Models sorted by RMSD to native structure
- tag: Internal model name
- top: Models sorted by Rosetta score

The `model` options may be string (if `mode` is 'tag'), or numbers. For `mode=cluster_ntop` it may also be in the form of `n/m`, meaning the `n``th best cluster out of the first ``m` clusters.

Examples:

```
rna_predict extract-models --mode=tag S_000289 S_000100  # extract two models by tag
rna_predict extract-models --mode=top 1 2 3 4 5  # extract the two best-scoring models
rna_predict extract-models --mode=ntop 1  # extract the model with the lowest native RMSD
```

```
rna_predict extract-models --mode=cluster 1 2 3 4 5  # extract the cluster primaries of the first 5 c
rna_predict extract-models --mode=cluster_ntop 1/5  # extract the lowest native RMSD cluster out of t
```

Evaluation tools

Plot generation and other tools can be accessed using:

```
rna_predict tools <tool> ...
```

plot-clusters

Plot score over native RMSD. Usage:

```
rna_predict tools plot-clusters \
    --score-weights <score:weight,score:weight,...> \
    --max-models <max> \
    cst
```

The `--score-weights` options allows to calculate a different total model score using the individual Rosetta scores. The score name "default" can be given to set a default value for all other, non-specified scores.

For example, to only visualize the score of additional constraints, use:

```
--score-weights default:0,atom_pair_constraint:1
```

For a list of score names, refer to the Rosetta documentation or use the `print-models` command.

The `--max-models` option limits the number of models by either specifying a number (greater than 1) or a fraction (smaller than or equal to 1.0).

plot-constraint-quality

This visualizes the distances of constraints by comparing it to a reference (native) PDB structure. Usage:

```
rna_predict tools plot-constraint-quality \
    --dca-mode \
    reference-pdb \
    cst|dca|filter [cst|dca|filter ...]
```

When `--dca-mode` is given, residue-residue distances are plotted, atom-atom contacts otherwise.

For the `filter`-syntax see *DCA Filtering Syntax*.

plot-contact-atoms

Plots atoms involved in forming nucleotide contacts that satisfy the cutoff condition in the contact database. Usage:

```
rna_predict tools plot-contact-atoms \
    --mean-cutoff <cutoff> \
    --std-cutoff <cutoff>
```

The `--mean-cutoff` and `--std-cutoff` options select the limits for the average contact distances standard deviations.

plot-contact-distances

Plots histogram for each nucleotide pair contact containing the distances of the atoms involved. Usage:

```
rna_predict tools plot-contact-distances
```

plot-dca-contacts-in-pdb

Visualizes how well DCA contacts are fullfiled in PDB files. Usage:

```
rna_predict tools plot-dca-contacts-in-pdb \
    dca-file \
    pdb-file [pdb-file -..]
```

plot-pdb-comparison

Compare PDB files by plotting the distances of the residues. Usage:

```
rna_predict tools plot-pdb-comparison \
    ref-pdb \
    sample-pdb [sample-pdb ...]
```

plot-gdt

Create a GDT (gloabl distance test) plot.

This plots a distance cutoff on the y-axis and the percent of residues which are below the cutoff on the x-axis.

Usage:

```
rna_predict tools plot-gdt \
    ref-pdb \
    sample-pdb [sample-pdb ...]
```

1.4 Residue-residue to atom-atom mapping

The result of the DCA analysis is a list of nucleotide contacts. These need to be mapped to atom-atom contacts to be usable as constraints in Rosetta. Two options are implemented.

1.4.1 P-Only mapping: `pOnly`

A simple variant is to take the P atom of each nucleotide backbone and add these as atom-atom constraints.

1.4.2 Complex atom-atom mapping using a contact database: `minAtom`

This method relies on pre-built contact lists included in the prediction package. For each type of nucleotide contact (A-A, A-C, A-G, A-U, ...) a list of representatives is stored, containing the PDB code of an entry with known native structure and the numbers of the residues. From these lists a database is built containing all possible atom-to-atom combinations for the two residues, as well as their distances extracted from the crystal structures. Next, to reduce

this database to the relevant atoms involved in forming a contact, a second database is created, containing only those atom-to-atom contacts that satisfy a mean and standard deviation condition. The defaults for those are set to lower than 6Å and 3Å respectively. DCA contacts such as "A-U" are then looked up in the mean database, and all atom-to-atom combinations found are added as constraints.

1.5 DCA Filtering Syntax

Sometimes it may be necessary to filter and modify DCA contacts.

To specify filters on the command line the following syntax is used:

```
filter:<arg>:<arg>:<arg>:...
```

The number of arguments depends on the filter being used.

Multiple filters may be chained by separating them with `,`.

The following filters are implemented:

1.5.1 none

The `none` filter works as a dummy filter and does nothing. It takes no options.

1.5.2 threshold

The `threshold` filter looks up DCA contacts in a target PDB file, calculates their distances, and discards or includes them depending on the threshold setting.

The syntax is the following:

```
threshold:<n>:<cst>:<mode>:<moodel>
```

Arguments:

- `n`: threshold (< 0: keep below, > 0: keep above)
- `cst`: prediction / constraints selection to look up PDB file
- `mode`, `model`: model selection mode (see *README <README.rst>* for details)

Example:

```
threshold:8.0:100dca_FADE_-100_26_20_-2_2:cluster:1,threshold:-6.0:100dca_FADE_-100_26_20_-2_2:cluste
```

This uses the first cluster of the `100dca_FADE_-100_26_20_-2_2` predicton as target PDB file. All DCA contacts whose distance is between 6Å and 8Å in the target PDB are kept.

1.6 rna_predict package

1.6.1 Submodules

1.6.2 rna_predict.dcatools module

class `rna_predict.dcatools.`**`DcaContact`**(*res1*, *res2*, *use_contact=True*, *weight=1*)

Bases: `object`

Class representing a DCA contact.

__init__(*res1*, *res2*, *use_contact=True*, *weight=1*)
Create new DCA contact.

Parameters

- **res1** – number of first residue (from 1)
- **res2** – number of second residue (from 1)
- **use_contact** – True or False if contact is to be used
- **weight** – assign a different weight to the contact (default 1)

get_rosetta_function(*function='FADE -100 26 20 -2 2'*)
Return Rosetta function of the contact as a list while applying weight.

Parameters **function** – rosetta function as text

Returns rosetta function as a list with applied weight

exception rna_predict.dcatools.**DcaException**
Bases: exceptions.Exception

Custom exception class used for foreseeable, DCA related errors.

class rna_predict.dcatools.**DcaFilter**
Bases: object

Filter base class.

apply(*contact*, *quiet=False*)
Apply filter to a DCA contact.

Parameters

- **contact** – DCA contact
- **quiet** – reduce verbosity

class rna_predict.dcatools.**DcaFilterThreshold**(*pdb_chain*, *threshold*, *keep_below=True*, *mode='minimum_heavy'*)
Bases: rna_predict.dcatools.DcaFilter

Filter to skips DCA contact if below or above a threshold.

__init__(*pdb_chain*, *threshold*, *keep_below=True*, *mode='minimum_heavy'*)
Create a new threshold filter.

Parameters

- **pdb_chain** – PDB chain
- **threshold** – threshold below or above to keep a contact
- **keep_below** – True to keep below threshold, False to keep above
- **mode** – What distance to compare to (average_heavy, minimum_heavy)

apply(*contact*, *quiet=False*)

rna_predict.dcatools.**build_cst_info_from_dca_contacts**(*dca_data*, *sequence*, *mapping_mode*, *cst_function*, *number_dca_predictions*, *quiet=False*)

Maps DCA residue contacts to atom-atom constraints.

Parameters

- **dca_data** – list od DcaContacts
- **sequence** – sequence as text
- **mapping_mode** – atom-to-atom mapping mode to use, supported values: "minAtom" or "pOnly"
- **cst_function** – rosetta function and parameters as text string
- **number_dca_predictions** – maximum number of DCA predictions to use
- **quiet** – reduce output verbosity

Returns list of constraint information

rna_predict.dcatools.**filter_dca_data**(*dca_data*, *dca_filter_chain*, *quiet=False*)

Run list of DCA contacts through a chain of filters.

Parameters

- **dca_data** – list of DcaContact objects
- **dca_filter_chain** – list of DcaFilter objects
- **quiet** – reduce output verbosity

rna_predict.dcatools.**get_atoms_for_res**(*res*, *term_phosphate=False*)

Get list of atoms for residue.

Parameters

- **res** – nucleotide (A,U,G,C)
- **term_phosphate** – add P atoms

Returns list of atoms

rna_predict.dcatools.**get_atoms_for_res_sequence**(*sequence*)

Get list of atoms for a sequence of nucleotides

Parameters **sequence** – sequence as text

Returns list of atoms

rna_predict.dcatools.**get_contact_distance_map**(*structure_directory='/home/sebastian/.rna_predict/structure_info'*, *westhof_vector=None*, *force_rebuild=False*)

Returns contact distance map

The contact distance map is cached it in the user directory and updated when newer files are found.

Parameters

- **structure_directory** – directory to look up structure information text files
- **westhof_vector** – list of factors to apply different weights to the bonding family classes (defaults to `[1, 1, ... ]`)
- **force_rebuild** – force rebuilding the distance map

rna_predict.dcatools.**get_contact_distance_map_mean**(*distance_map*, *mean_cutoff=None*, *std_cutoff=None*)

Return an average distance map containing only those contacts with average distance and standard deviation satisfiying a cutoff.

Parameters

- **distance_map** – full distance map
- **mean_cutoff** – limit for average
- **std_cutoff** – limit for standard deviation

Returns average distance map

rna_predict.dcatools.**get_contact_information_in_pdb_chain**(*dca_contact*, *pdb_chain*, *heavy_only=True*)

Returns distance information about a DCA contact in a realized PDB chain

Return value is a tuple of:

•Average distance: Mean distance of all atoms in the contacts.

•Minimum distance: Minimum distance between two atoms in the contact.

•Minimum pair: List of `[atom1, atom2]` forming the minimal contact

Parameters

- **dca_contact** – DcaContact object
- **pdb_chain** – PDB chain structure object
- **heavy_only** – Only use heavy atoms

Returns tuple `(average_dist, minimum_dist, minimum_pair)`. In case the contact cannot be found in the PDB file `(0, 0, None)` is returned.

rna_predict.dcatools.**parse_dca_data**(*dca_prediction_filename*)

Read a DCA file, adjust the sequence numbers to match the alignment of the PDB file, and create a list of DcaContacts

Parameters **dca_prediction_filename** – DCA input filename

Returns list of DcaContact objects

rna_predict.dcatools.**read_pdb_mapping_from_file**(*dca_prediction_filename*)

Read a PDB mapping from DCA file if present and return it as text

Parameters **dca_prediction_filename** – DCA input filename

Returns PDB mapping text

1.6.3 rna_predict.main module

rna_predict.main.**main**()

main commandline parser.

1.6.4 rna_predict.pdbtools module

rna_predict.pdbtools.**align_structure**(*ref_pdb*, *moving_pdb*, *assign_b_factors=True*)

Align one PDB structure to another.

Parameters

- **ref_pdb** – reference PDB structure object

- **moving_pdb** – moving PDB structure object
- **assign_b_factors** – place distance values in the b-factor field

Returns tuple (res_dists, atom_dists, rmsd, transformation_matrix)

rna_predict.pdbtools.**filter_atoms**(*atoms*, *heavy_only=False*, *p_only=False*)
Filter list of atoms.

Parameters

- **atoms** – list of atoms
- **heavy_only** – only keep heavy atoms
- **p_only** – only keep P atoms

Returns filtered list of atoms

rna_predict.pdbtools.**get_center_of_res**(*res*)
Calculate the center of a residue.

Parameters **res** – residue object

Returns center coordinates

rna_predict.pdbtools.**get_pdb_by_code**(*pdb_code*, *pdb_directory='/home/sebastian/.rna_predict/pdbs'*)
Get PDB file by code. Download if necessary.

Parameters

- **pdb_code** – PDB code to download
- **pdb_directory** – directory lookup and store the PDB file, defaults to the rna_predict PDB cache directory

Returns PDB filename

rna_predict.pdbtools.**parse_pdb**(*pdb_code*, *pdb_file*)
Parse PDB file using Biopyhon.

Parameters

- **pdb_code** – internal id
- **pdb_file** – PDB filename

Returns PDB structure object

1.6.5 rna_predict.simulation module

class rna_predict.simulation.**Command**(*command*, *add_suffix=None*, *dry_run=False*, *print_commands=True*, *stdin=None*, *quiet=False*)
Bases: object

Helper class to store external program calls plus additional flags.

__init__(*command*, *add_suffix=None*, *dry_run=False*, *print_commands=True*, *stdin=None*, *quiet=False*)
Create a new command wrapper.

Parameters

- **command** – list of external command and parameters

- **add_suffix** – type of suffix to first entry in command. currently only `None` or `rosetta` are supported
- **dry_run** – don't actually execute anything
- **print_commands** – print commandline when running
- **stdin** – optional text to use as standard input
- **quiet** – hide output of external program

get_full_command(*sysconfig*)
If necessary, add the suffix to the command. Depending on user / system configuration.

Parameters **sysconfig** – instance of SysConfig

Returns suffixed command as a list

class rna_predict.simulation.**EvalData**
Bases: object

Evaluation data storing model and cluster information.

__init__()
Create empty EvalData object.

get_model_count()
Returns the number of models in the vaulation data

Returns number of models

get_models(*model_list*, *kind='tag'*)
Returns a selection of models with specific properties

Parameters

- **model_list** – list of models to get, might be a list of numbers, or a list of model names
- **kind** – model selection mode:
  - `tag`: str: internal name such as S_000123_5
  - `top`: int: models ordered by score
  - `ntop`: int: models ordered by rmsd_native
  - `cluster`: int: nth cluster decoy
  - `cluster_ntop`: n[/m]: nth cluster ordered by native rmsd [of first m]

Returns list of selected models

static get_weighted_model_score(*model*, *score_weights*)
Calculate a model score based on different weights for the invidiual Rosetta scores

Parameters

- **model** – model to reweight
- **score_weights** – dict of rosetta score names and their weights. `default` to set the default weight for all scores. Example: `{'default':0, 'atom_pair_constraint': 1}` to only use atom pair constraints.

static load_from_cst(*constraints*)
Load evaluation data from a prediction.

Parameters **constraints** – constraints selection

static load_from_file(*filename*)
Load evaluation data from a file.

Parameters **filename** – path to a file containing the evaluation data

print_models(*model_list*, *kind='tag'*)
Print a list of models

Parameters

- **model_list** – see get_models
- **kind** – see get_models

save_to_file(*filename*)
Dump evaldata to a file.

Parameters **filename** – path to a file to store the data

class rna_predict.simulation.**RNAPrediction**(*sysconfig*)
Bases: object

Base class used for prediction simulation

CONFIG_FILE = '.config'

__init__(*sysconfig*)
Create a prediction simulation for the current directory and try to load an existing configuration.

assemble(*nstruct=50000*, *cycles=20000*, *constraints=None*, *dry_run=False*, *seed=None*, *use_native_information=False*, *threads=1*)
Assembly step. Assemble helices and motifs into complete models.

Parameters

- **nstruct** – number of models to create
- **cycles** – number of monte-carlo cycles per model
- **constraints** – constraints selection
- **dry_run** – don't actually run any external command
- **seed** – optionally override random seed
- **use_native_information** – use native pdb file to calculate rmsd for each model
- **threads** – number of threads to use

check_config()
Check if current directory contains a valid configuraion.

create_helices(*dry_run=False*, *threads=1*)
Helix creation step. Create one ideal helix model for each helix.

Parameters

- **dry_run** – don't actually run any external command
- **threads** – number of threads to use

create_motifs(*nstruct=50000*, *cycles=20000*, *dry_run=False*, *seed=None*, *use_native_information=False*, *threads=1*, *constraints=None*, *motif_subset=None*)
Motif generation step. Generate models for each motif.

Parameters

- **nstruct** – number of models to create for each motif

- **cycles** – number of monte-carlo cycles per model
- **dry_run** – don't actually run any external command
- **seed** – optionally override random seed
- **use_native_information** – use native pdb file to calculate rmsd for each model
- **threads** – number of threads to use
- **constraints** – constraints selection
- **motif_subset** – sepcify a list of motifs to generate instead of all, counting starts at 1 (i.e. [1, 3, 4])

edit_constraints(*constraints*, *output_filename=None*, *cst_function='FADE -100 26 20 -2 2'*)
Edit an existing .cst file, replacing the rosetta function.

Parameters

- **constraints** – constraints selection
- **output_filename** – fixed output filename or None to automatically create
- **cst_function** – rosetta function and parameters as text string

evaluate(*constraints=None*, *cluster_limit=10*, *cluster_cutoff=4.1*, *full_evaluation=False*)
Evaluation step. Extract model information, cluster models and calculate rmsd values.

Parameters

- **constraints** – constraints selection
- **cluster_limit** – maximum number of clusters to produce
- **cluster_cutoff** – rmsd distance in A at which a new cluster is created
- **full_evaluation** – discard existing evaluation data, re-extract model information and re-calculate rmsd values.

evaluate_custom(*constraints=None*, *dca_prediction_filename='dca/dca.txt'*, *full_evaluation=False*, *threshold=7.5*, *radius=2*, *number_dca_predictions=100*, *threads=4*)
Custom scoring algorithmy by inspecting neighboring residues of dca contact pairs

Parameters

- **constraints** – constraints selection
- **dca_prediction_filename** – dca filename
- **full_evaluation** – discard existing evaluation data and extracted models, re-extract and re-calculate distance information
- **threshold** – threshold in A to count a contact
- **radius** – number of adjacent residues to inspect
- **number_dca_predictions** – maximum number of DCA predictions to use
- **threads** – number of threads to use for extraction

execute_command(*command*, *add_suffix=None*, *dry_run=False*, *print_commands=True*, *stdin=None*, *quiet=False*)
Execute an external command.

Parameters

- **command** – list of external command and parameters

- **add_suffix** – type of suffix to first entry in command. currently only `None` or `rosetta` are supported
- **dry_run** – don't actually execute anything
- **print_commands** – print commandline when running
- **stdin** – optional text to use as standard input
- **quiet** – hide output of external program

execute_command_and_capture(*command*, *add_suffix=None*, *dry_run=False*, *print_commands=True*, *stdin=None*, *quiet=False*)

Execute an external command while capturing output.

Parameters

- **command** – list of external command and parameters
- **add_suffix** – type of suffix to first entry in command. currently only `None` or `rosetta` are supported
- **dry_run** – don't actually execute anything
- **print_commands** – print commandline when running
- **stdin** – optional text to use as standard input
- **quiet** – hide output of external program

Returns generator over lines of standard output

execute_commands(*commands*, *threads=1*)

Execute a list of commands parallelly.

Parameters

- **commands** – list of Commands
- **threads** – number of parallel invocations

extract_models(*constraints*, *model_list*, *kind='tag'*)

Extract PDB files of a set of models

Parameters

- **constraints** – constraints selection
- **model_list** – see EvalData.print_models
- **kind** – see EvalData.print_models

extract_pdb(*constraints*, *model*)

Extract PDB file of a model to the tmp directory.

Parameters

- **constraints** – constraints selection
- **model** – tag of the model

Returns path to the extracted PDB file

get_csts()

Returns a list of all constraints.

Returns list of all constraints names

static get_models(*constraints*, *model_list*, *kind='tag'*)
Returns a selection of models with specific properties

Parameters

- **constraints** – constraints selection
- **model_list** – see EvalData.print_models
- **kind** – see EvalData.get_models

get_status(*constraints=None*, *include_evaluation_data=False*, *include_motif_model_count=False*, *include_assembly_model_count=False*)
Returns a dict containing status information for a cst.

Parameters

- **constraints** – constraints selection
- **include_motif_model_count** – set to True to include model count (longer processing time)
- **include_assembly_model_count** – set to True to include model count (longer processing time)
- **include_evaluation_data** – set to True to include evaluation model count and native RMSD values (best of first 1,5,10 clusters) if available (longer processing time)

Returns status dict

load_config()
Load simulation configuration of current directory

make_constraints(*dca_prediction_filename='dca/dca.txt'*, *output_filename=None*, *number_dca_predictions=100*, *cst_function='FADE -100 26 20 -2 2'*, *filter_text=None*, *mapping_mode='minAtom'*)
Create a set of constraints from a DCA prediction.

Parameters

- **dca_prediction_filename** – DCA input file
- **output_filename** – fixed output filename or None to automatically create
- **number_dca_predictions** – maximum number of DCA predictions to use
- **cst_function** – rosetta function and parameters as text string
- **filter_text** – optional: List of DCA filters (see parse_dca_filter_string for details)
- **mapping_mode** – atom-to-atom mapping mode to use, supported values: `minAtom` or `pOnly`

modify_config(*key*, *value*)
Modify configuration entry.

Parameters

- **key** – setting to modify
- **value** – new value ("-" to store None)

static parse_cst_file(*constraints_file*)
Parse .cst file as a list of constraints

Parameters **constraints_file** – path to a .cst file

Returns list of constraints

static parse_cst_name_and_filename(*constraints*)
Find and clean up constraints by name or filename

Parameters **constraints** – constraints name or filename

Returns tuple of constraints name and filename

parse_dca_filter_string(*line*)
Parse a text string and turn it tinto a list of DCA filters

Multiple filters are separated by command and have the folling format:

`<filter>:<arg>:<arg>:...`

Threshold filter: Lookup dca contacts in a PDB file, discard or keep contact depending on whether the contact is realized:

Format: `threshold:<n>:<cst>:<mode>:<moodel>`

- n: threshold (< 0: keep below, > 0: keep above)
- cst: constraints selection to look up PDB file
- mode, model: model selection mode (see EvalData.get_models for details)

Example: `threshold:8.0:100rnaDCA_FADE_-100_26_20_-2_2:cluster:1,threshold:-6.0:100rna`

None filter: Empty filter

Format: `none`

Parameters **line** – string to parse

Returns list of DcaFilter objects

prepare(*fasta_file='sequence.fasta'*, *params_file='secstruct.txt'*, *native_pdb_file=None*, *data_file=None*, *torsions_file=None*, *name=None*)
Preparation step. Parse out stems and motifs from sequence and secondary structure information and create necessary base files.

Parameters

- **fasta_file** – fasta file containing the sequence
- **params_file** – text file containing the secondary structure
- **native_pdb_file** – native pdb file if available
- **data_file** – additional data file if available
- **torsions_file** – additional torsions file if available
- **name** – optional name for this set of predictions

prepare_cst(*constraints=None*, *motifs_override=None*)
Constraints preparation step. Prepare constraints files for motif generation and assembly.

Parameters

- **constraints** – constraints selection
- **motifs_override** – optional name of a different set of constraints to use as motifs.

print_config()
Print simulation configuration.

static print_models(*constraints*, *model_list*, *kind='tag'*)
Print a set of models

Parameters

- **constraints** – constraints selection
- **model_list** – see EvalData.print_models
- **kind** – see EvalData.print_models

print_status(*native_compare=False*, *csts=None*)
Print summary of predictions and their current status.

Output format always contains the following columns:

•P: preparation step

•M: motif generation

•A: assembly

•E: evaluation

If a step is completed, X is shown, − otherwise.

For motif generation a ⋆ may be shown to indicate that models from a different set of constraints are used.

If *native_compare* is set to True another 4 columns are printed:

•1: native rmsd score of the first cluster

•5: lowest native rmsd score of the first 5 clusters

•10: lowest native rmsd score of the first 10 clusters

•n: number of models

Parameters

- **native_compare** – print rmsd comparison to native structure
- **csts** – list of constraints to include in output (default: all)

save_config()
Save simulation configuration of current directorx

exception rna_predict.simulation.**SimulationException**
Bases: exceptions.Exception

Custom exception class for foreseeable prediction errors.

rna_predict.simulation.**check_dir_existence**(*path*, *alternative_error_text=None*)
Make sure a directory exists and raise an exception otherwise.

Parameters

- **path** – directory to check
- **alternative_error_text** – alternative error text passed to exception

rna_predict.simulation.**check_file_existence**(*path*, *alternative_error_text=None*)
Make sure a file exists and raise an exception otherwise.

Parameters

- **path** – filename to check
- **alternative_error_text** – alternative error text passed to exception

rna_predict.simulation.**delete_glob**(*pattern*, *print_notice=True*)
Helper function to delete files while expanding shell globs.

Parameters

- **pattern** – pattern to delete
- **print_notice** – if True print verbose notice

rna_predict.simulation.**fix_atom_names_in_cst**(*cst_info*, *sequence*)
Switches N1 and N3 atom names in a residue.

Parameters

- **cst_info** – list of constraints
- **sequence** – residue sequence in lower case

Returns modified list of constraints

rna_predict.simulation.**get_model_count_in_silent_file**(*silent_file*)
Returns the number of models present in a Rosetta silent file.

Parameters **silent_file** – filename

Returns model count

rna_predict.simulation.**merge_silent_files**(*target*, *sources*)
Merges rosetta silent files (.out).

All source files are appended to target. Model numbers are incremented uniquely. Source files are deleted after a successful merge.

Parameters

- **target** – target filename
- **sources** – source filenames

Returns model count

rna_predict.simulation.**natural_sort_key**(*s*, *_nsre=<_sre.SRE_Pattern object at 0x7fc8a47b0df0>*)
Helper function to be used as sort key in sorted() to naturally sort numeric parts.

1.6.6 rna_predict.sysconfig module

class rna_predict.sysconfig.**SysConfig**
Bases: object

Stores user configuration

SYSCONFIG_FILE = '/home/sebastian/.rna_predict/sysconfig'

SYSCONFIG_LOCATION = '/home/sebastian/.rna_predict'

__init__()
Load system configuration.

check_sysconfig()
Check if all external programs are accessible.

Returns dict containing lists of failed (fail) and successful (success) programm accesses

load_sysconfig()
Load system configuration.

print_sysconfig()
Pretty-print configuration.

1.6.7 rna_predict.tools module

rna_predict.tools.**plot_clusters**(*cst*, *max_models=0.99*, *score_weights=None*)
Plot score over native rmsd.

Parameters

- **cst** – constraints
- **max_models** – limit to number of models if > 1, or relative percentage if <= 1
- **score_weights** – see EvalData.get_weighted_model_score

rna_predict.tools.**plot_constraint_quality**(*comparison_pdb*, *sources*, *dca_mode=False*)
Plot constraint quality.

This visualizes the distances of constraints by comparing it to a reference (native) PDB structure.

Parameters

- **comparison_pdb** – filename to a pdb file to compare to
- **sources** – list of dca files or constraints (depending on dca_mode). may also use 'filter:' to filter on-the-fly
- **dca_mode** – visualize residue-residue DCA instead of atom-atom constraints

rna_predict.tools.**plot_contact_atoms**(*mean_cutoff*, *std_cutoff*)
Plots atoms involved in forming nucleotide contacts that satisfy the cutoff condition in the contact database.

Parameters

- **mean_cutoff** – limit for average distance
- **std_cutoff** – limit for the standard deviation

rna_predict.tools.**plot_contact_distances**()
Plots histogram for each nucleotide pair contact containing the distances of the atoms involved.

rna_predict.tools.**plot_contact_map**(*native_filename='native.pdb'*, *first_filename='dca/dca.txt'*, *second_filename='dca/mi.txt'*, *native_cutoff=8.0*)

rna_predict.tools.**plot_dca_contacts_in_pdb**(*dca_prediction_filename*, *pdb_files*)
Visualize how well DCA contacts are fullfiled in PDB files.

This plots the distances of dca contacts in one or more PDB files.

Parameters

- **dca_prediction_filename** – input DCA filename
- **pdb_files** – list of PDB filenames

rna_predict.tools.**plot_gdt**(*pdb_ref_filename*, *pdbs_sample_filenames*)

rna_predict.tools.**plot_pdb_comparison**(*pdb_ref_filename*, *pdbs_sample_filenames*)
Compare PDB files by plotting the distance of the residues.

Parameters

- **pdb_ref_filename** – reference PDB filename

- **pdbs_sample_filenames** – list of sample PDB filenames

rna_predict.tools.**plot_tp_rate**(*pdb_ref_filename*, *dca_filenames*, *tp_cutoff=8.0*)

1.6.8 rna_predict.utils module

rna_predict.utils.**comma_separated_entries_to_dict**(*s*, *type_key*, *type_value*)

Parses a string containing comma separated key-value paris (colon-separated) into a dict with fixed key and value types.

Example: Turns `foo:3,bar:4` into `{"foo": 3, "bar": 4}`

Parameters

- **s** – input string
- **type_key** – type of the keys
- **type_value** – type of the values

Returns dict

rna_predict.utils.**comma_separated_ranges_to_list**(*s*)

Parses a string containing comma separated ranges to a list.

Example: Turns `1-3,10,20-22` into `[1, 2, 3, 10, 20, 21, 22]`

Parameters **s** – comma separated ranges

Returns list of ints

rna_predict.utils.**mkdir_p**(*path*)

Creates directories recursively and does not error when they already exist.

Parameters **path** – directory to create

rna_predict.utils.**read_file_line_by_line**(*filename*, *skip_empty=True*)

Yields lines in a file while stripping whitespace.

Parameters

- **filename** – filename to read
- **skip_empty** – True if empty lines should be skipped

Returns next line in file

1.6.9 Module contents

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*

r

Symbols

A

B

C

D

E

F

G

L

M

N

P

R

S