**PyRy3D (version 4.6) User Manual**

# 1. PyRy3D: Functionality

PyRy3D is a method for building multi-resolution models of large macromolecular complexes using restraints derived from biochemical data. The components (proteins, nucleic acids and any other type of physical objects including e.g. solid surfaces) can be represented as rigid bodies (e.g. based on atomic coordinates of structures determined experimentally or modeled computationally) or as flexible shapes (e.g. for parts, whose structure is dynamic or unknown).

PyRy3D run can be initiated with input files that include sequences of all complex components (in MultiFASTA format) and structures of some components (ar .tar archive containing files in PDB format). Additionally, information about restraints can be provided (in Filtrest3D format), as well as a complex density map (in CCP4 format) or SAXS curve (as .dat file).

The output from PyRy3D consists of a set of .pdb files with generated models and a file containing information about PyRy3D parameters used and PyRy3D scores for models saved on disk. All models generated by PyRy3D are saved in PDB format and the filenames contain information about a score assigned by the program, number of simulation step and current temperature (a parameter characteristic for simulated annealing algorithm). For the user's request, a trajectory of complex simulation trace (file with extension .trafl) can be produced and a text file with a history of all movements applied to all complex components. All these files can be used (or reused) in various post-processing tasks (such as models clustering).

Additional tools that accompany this distribution are explained in the Section 6 titled Additional Tools.

# *2.* Command line options:

To get help with PyRy3D command line just type:

```
python pyry3d.py - - help
```

and a list of all available options will appear with short descriptions.

```
-s input_file_sequence
```

The sequences of complex components are written in a single text file in MultiFASTA format, where each sequence is called as the corresponding chain ID in PDB file (the defline of each sequence starts with ">" and chain ID).

```
-c simulation_config_file
```

The config_file contains a variety of important program's settings: see section 2.0 for details.

```
-o output_files_basename
```

If this is not specified, then the program will create a "pyryresults" directory and copy output files there.

```
-r restraints_file
```

This is the file specifying restraints. (The details on the format of the file for restraints, Filtrest3D, are discussed below. The format is somewhat strict, and should be followed or an error will be issued.)

`-d archive_with_structures`
This is .tar or .tgz archive containing 3D coordinates of complex components. Required format is PDB and only one chain can be stored in a single PDB file.

`-m density_map_file`
This is the shape of the complex defined by the electron density map. Negative stain map can also be provided here. PyRy3D accepts files in .ccp4 and .mrc format.

`-y SAXS_curve_file`
This is the shape of the complex measured with SAXS and saved as a curve in .dat format.

`-f full_atom_representation`
This command allows to save output models in full atom representation, even if reduced representation was used during the simulation.

`-v history_of_movements`
This option allows to save a history of movements applied to components during the simulation. The file is also required to reconstruct full atom representation of the model.

`-e energy_plot`
This option creates PyRy3D score plot.

`-t trafl_file`
This option allows to save a simulation trajectory.

## *2.1. Format of sequence input file*

`-s input_file_sequence`
The sequences of complex components are written in a single text file in MultiFASTA format, where each sequence's header contains its corresponding chain ID in PDB file (the defline of each sequence starts with ">" and chain ID). For example:

```
>A
MMMET
>B
AAUUCCGG
>X_protein
WERTY
>D_DNA
AAATCG
```

"A" sequence corresponds to protein structure with chain id "A", B to RNA structure with chain id "B", "X_protein" to protein with unknown structure and "D_DNA" refers to DNA sequence with no 3D data (no PDB provided by the user for X and D sequences).

## *2.2. Format of structure input files*

`-d archive_with_structures`

This is a single .tar or .tgz archive containing 3D coordinates of all complex components. Required format is PDB and only one chain can be stored in a single PDB file.

**Comment:**

- The format of the PDB file has some very strict requirements: i.e., it should be a simple structure
- Chain IDs for all components must be unique (you cannot provide two structures with "A" chain ID)
- For each PDB file a sequence must be provided in MultiFASTA file
- Do not use heteroatoms (HETATOM), use 'ATOM ' instead
- Program supports non-standard base notations: the readable residues are A, C, G, U, T, but also most modified residues and some ligands are supported
- Be sure that there are no duplicated atoms or residues in provided structures, program will complain about it
- In a single PDB file only one chain should be provided – if more chains are given, only the first one will be considered by PyRy3D
- If one want to treat e.g. protein and DNA as a single component during simulation, the easiest way is to prepare two separate PDB files and define COVALENT_BONDS (explained further in this manual) for these two molecules in the configuration file provided for PyRy3D. In such a case PyRy3D will treat those two molecules as one e.g. if protein A is rotated, DNA is rotated the same way at the same time.

## *2.3. Format of long-range restraint files*

`-r restraints_file`

The input restraint file contains information on the type and weight (importance) of the restraints between entities (defined as atoms, sets of atoms or even (X,Y,Z) points in 3D space).

### Restraint options

A restraint can be thought of as a flexible tether that drives the selected atoms towards points located within a certain distance by applying a penalty for distances that deviate from that range. It can also provide a reward when a desired distance is achieved. The penalty and reward are positive and negative contributions to the total energy of the simulated system, respectively.

### Restraint command formats

The restraints format implemented in PyRy3D was based on Filtrest3D format. The description of the latter can be found under the following website: http://filtrest3d.genesilico.pl/readme.html. In PyRy3D some new types of restraints occurred (e.g. surface accessibility, distance from point in 3D space, relation or symmetry)

General restraint file syntax is:

```
// Comment
```

```
DEFAULT_WEIGHT <- 1.0

RESTRAINT_TYPE_NAME (
  RESTRAINT_DECLARATION
  RESTRAINT_DECLARATION
  ...
)
```

There are following restraint types (restraint type name in parentheses):

- distance: (*dist*);
- surface accessibility: (*surface_access*);
- solvent accessibility (*access*);
- distance from point in space: (*pointdist*);
- symmetry: (*symmetry*);
- relation between distances (*relation*);
- logical operators (*and, or*).

**Distance restraints**

*Distance* restraints allow to define the permitted range of distances between the residues (e.g., "5 Å between any atom of residue X and residue Y" or "5-10 Å between the Cα residue X in chain A and any residue of the fragment Y-Z in chain B") or sets of residues (e.g., "10 Å between β-sheet A171-C182 of chain A, and an α-helix A57-D62 of chain B").

- If C-alpha atoms of V10 and D34 should be in at least 4.5 Angstroms distance:

*dist ( (V10)- "A" (D34) "X" (>4.5 ) )*

- To restraint SG atom of the residue C140 to be within 4 angstroms from OE1 atom of residue E33:

*dist ( C140_near_E33: (C140) "A"-(E33) "E" (SG-#OE1#<=4) )*

- To define distance between two regions (here any residue in range 163-175 in chain Z should be within 4.8 Angstroms from any residue 306-310 in chain A)

*dist ( (E163-F175) "Z" -(306-310) "A" (<=4.8) )*

**Surface accessibility**

*Surface accessibility* allows for defining entities (atoms, residues, components fragments) that are positioned on the surface of the whole complex such as exposed enzyme active sites.

*surface_access ( (163-165) "Z" (<=1.0) )*

**Solvent accessibility**

The *residue burial/exposition to the solvent* may be given as a range of the relative accessible solvent area (ASA) expressed in percentages (e.g., "residue X is exposed in 40-50 %") to define residues exposed to solvent in a particular component.

- To define aminoacid 45th in chain "A" as having between 40% to 90% accessibility, Glycine 56th as having 30% to 60% accessibility, and alanine 20th as having between 80% to 90% accessibility, use the following formula:

*access (*

*45 “A” 40-90*
*Gly56 “A” 30-60*
*A20 80-90 “A” weight<-1.5*
*)*

**Relation**

*Relation* restraint is specific to PyRy3D (it is not available in Filtrest3D itself) and is used to compare two distances (e.g., data from FRET like “a distance between residues X-Y of chain A and residues U-Z of chain B” is larger/smaller/equal to “a distance between the Cα residue X in chain C and any residue of the fragment Y-Z in chain D”).

- Relation restraint is applied to define relation between two distances e.g. distance between residue 3 in chain A and residue 5 in chain B should be smaller than distance between residue 30 in chain A and residue 500 in chain B

*relation ( (3) "A" -(5) "B" () < (30) "A" -(500) "B" ())*

**Distance from point**

The distance can be defined between a residue or a set of residues and a point in 3D space (*PointDistance*):

*pointdist ( (100-101) "D"- (20.0, 20.0, 20.0) (<=5.5) )*

**Symmetry**

Symmetry allows to provide distances that should be of equal length in the complex. This particular restraint can be applied to introduce symmetry between identical copies of oligomeric assemblies.

*symmetry (*
*(1) "A" -(1) "B" ()*
*(1) "B"-(1) "C" (CA-CA)*
*(1) "A"-(1) "C" ()*
*)*

**Conjunction and alternative**

To declare composite restraint having score of minimum (“or”) or maximum (“and”) of two other restraints, the following restraint blocks can be used:

*and (*
*access (*
*Gly56 30-60*
*)*
*access (*
*Glu45 40-90*
*)*
*)*

**Restraint weights**

Default restraint weight is 1.0, but it can be changed:

DEFAULT_WEIGHT <- 2 //all penalties defined in the restraint file will be multiplied by 2

*dist (*
*(V10) “A”- (34-36) “B” (CA<4.5 weight<-2 // C-alpha penalty will be multiplied by 2.0*
*)*

**Comments:**

- the residue numbers specified in the restraints file must correspond to ones in input PDB file !!!

- chain id is required in all restraint types to indicate which component the user refers to

## *3.4. Format of the configuration file*

`-c simulation_config file`

The argument is a file containing simulation configuration parameters. With this option, the user can configure the simulation in a more advanced way.

The configuration file is a text file with one parameter specified in each line. There can be blank lines, each line must contain a command or must be empty. Comments starts with "#". Also, if no configuration file is provided by the user, PyRy3D is run with default parameters (see the tables below in this paragraph).

**Config file options**

These are some of the options that can be included in a typical configuration file.

`STEPS N`

This specifies the number of unified iterations in a simulated annealing simulation.

`WRITE_N_ITER N`

This specifies how many iterations should occur before the conformation is saved on disk or appended to the trajectory file. In general, there probably should be at least 10 000 iterations of PyRy3D before a write is done to generate a meaningful structural change. However, if the user desires to have a chain of quite similar structures, this value can be low.

`ANNTEMP float1 float2`

This specifies the temperature parameter of the simulation (and is applied only to Simulated Annealing algorithm). Float1 refers to the temperature at the beginning of the simulation, float2 to the temperature at the end.

`SIMMETHOD name_of_algorithm`

Algorithm to perform sampling of conformational space. Available options: "Genetic" for genetic algorithm or "SimulatedAnnealing" for simulated annealing (default) or "ReplicaExchange" for replica exchange.

**Comments:**

- !!! The final temperature can be lower or higher that initial temperature. In the second case, the system is being gradually heated. !!!
- !!! The final temperature can be also equal the initial temperature. In such a case, the program maintains same temperature during entire simulation !!!
- The order of parameters in configuration file is not important

The format of the file up to this point should be as follows

```
STEPS  1000
WRITE_N_ITER 20
ANNTEMP        1.35 0.90
SIMMETHOD SimulatedAnnealing
```

There are some additional terms that can be set up the use in the configuration file. These options include the following:

**Scoring function parameters:**

This is a group of parameters that control a scoring function and correspond to weights assigned for particular features of generated model such as number of clashes, violation of restraints or leaving empty spaces in density map.

| parameter name | default value | available values | description |
|---|---|---|---|
| OUTBOX | 1 | 0 to 10 | Weight of a penalty for atoms/residues outside simulation area |
| MAP_FREESPACE | 1 | 0 to 10 | Weight of a penalty for free spaces inside density map |
| CLASHES | 1 | 1 to 10 | Weight of a penalty for collisions (only CA and C4' atoms are considered) |
| CLASHES_ALLATOMS | 1 | 1 to 10 | Weight of a penalty for collisions (all atoms are considered) |
| RESTRAINTS | 1 | 0 to 10 | Weight of a penalty for violation of distance restraints |
| DENSITY | 1 | 0 to 10 | Weight of a penalty for occupation of map points with low density values |
| CHI2 | 1 | 0 to 10 | Weight of a penalty for disagreement with SAXS curve |
| RGE | 1 | 0 to 10 | Weight of a penalty for disagreement with user defined radius of gyration (RG_VAL) |
| SYMMETRY | 1 | 0 to 10 | Weight of a penalty for violation of symmetry restraints |

**Movements frequencies:**

Describe how often particular types of movements are applied to components

| parameter name | default value | available values | description |
|---|---|---|---|
| ROTATION_FREQ | 0.25 | 0 to 1 | frequency of rotations |
| ROTATION_COV_FREQ | 0.25 | 0 to 1 | frequency of rotations around covalent bonds |
| TRANSLATION_FREQ | 0.25 | 0 to 1 | frequency of translations |
| EXCHANGE_FREQ | 0.25 | 0 to 1 | frequency of components exchange |

| | | | |
|---|---|---|---|
| SIMUL_DD_FREQ | 0.25 | 0 to 1 | frequency of simulation of disordered fragments |
| ROTATION_ALL_FREQ | 0.25 | 0 to 1 | frequency of rotations where all components are moved simultaneously |
| ROTATION_WHOLE_FREQ | 0.25 | 0 to 1 | frequency of rotations where all components are moved simultaneously around common centre of mass |
| TRANSLATION_ALL_FREQ | 0.25 | 0 to 1 | frequency of rotations where all components are moved simultaneously |

**Information about complex:**

Set of different parameters describing modeling complex such as density threshold of density map, values measured with SAXS or definitions of mobile and immobile components

| parameter name | default value | available values | description |
|---|---|---|---|
| KVOL | 1 | 1 to 10 | how many complex volumes will describe density map, e.g. KVOL=2 indicated that map volume will be twice as big as complex volume calculated from its sequence |
| THRESHOLD | 0.0 | Value set must occur in a density map | float value describing density map threshold |
| SAXSRADIUS | 0.0 | positive float value | float value describing dammy atom radius |
| RG_VAL | 0.0 | positive float value | float value describing radius of gyration for a complex |
| CRYSOL_PATH | 0.0 | string | path to CRYSOL binaries |
| MOVE_STATE | no default values, parameter is optional | e.g. D movable 5 5 5 0.1 0.1 0.1 10 10 10 0.1 0.1 1 5 30 To fix a molecule use "fixed" parameter | Indicates limited values of moves (rotations, translations) for particular component |
| COVALENT_BONDS | no default values, parameter is optional | ChainName [ChainBound1, ChainBound2] [AtomBoundNumber1, AtomBoundName1] [AtomBoundNumber2, AtomBoundName2], e.g. | is used to indicate which components are linked by a covalent bond |

| | | | |
|---|---|---|---|
| | | A ['Z','D'] [10,'CA'] [11,'CA'] | |
| START_ORIENTATION | False | True/False | True if user sets start conformation, False if not!! |
| IDENTIFY_DISORDERS | False | True/False | True if user wants PyRy3D to add missing or disordered fragments into the structures, False if not!! |

**Simulation parameters:**

| parameter name | default value | available values | description |
|---|---|---|---|
| SIMMETHOD | Simulated Annealing | "Genetic" or "Simulated Annealing" for simulated annealing (default) or "ReplicaExchange" for replica exchange | simulation algorithm |
| REDUCTMETHOD | roulette | roulette, cutoff, tournament | is available only for genetic simulation mode |
| REPLICAEXCHANGE_FREQ | 2 | integers | each X steps replicas will be exchanged; by default every 10% of simulation steps |
| REPLICATEMPERATURES | 400 375 350 325 300 275 250 225 200 175 150 125 100 | list of integer values of any size | list of temperatures for all replicas |
| ANNTEMP | 100 | from range 1 to 100 | annealing temperature in simulated annealing procedure |
| STEPS | 100 | min 1, max number is not limited | Corresponds to number of simulation steps to perform |
| COMPONENT_REPRESENTATION | ca | CA - only calfas/c4' (default); cacb - coarse grain, 3p - 3points, fa - | Type of structure representation |

| | | | |
|---|---|---|---|
| | | full atom | |
| GRIDRADIUS | 1.0 | no limits set | Radius of single grid cell. |
| SIMBOX | 2.0 | no limits set | Parameters indicated how many times simulation box diameters is bigger than the longest distance inside a density map |
| MAXROT | 5 | from 1 to 360 | Maximal rotation angle for single component move |
| MAXTRANS | 5 5 5 | no limits set | Maximal translation vector for single component move |

**Specific parameters to control simulation progress:**

| parameter name | default value | available values | description |
|---|---|---|---|
| WRITE_N_ITER | 100 | - | how often PyRy3D save a model on disk |
| PARAMSCALINGRANGES | 0 25 50 | - | at what point of simulation should parameter scaling ranges kick in |
| PARAMSCALINGR1 | 50 100 | - | first scaling range (numbers refer to steps) |
| PARAMSCALINGR2 | 25 50 | - | second scaling range (numbers refer to steps) |
| PARAMSCALINGR3 | 0 25 | - | third scaling range (numbers refer to steps) |

**Some examples:**

#use density map volume equal to 10 complex volumes
```
KVOL 10
```

#use density map with density values >= threshold equal to 1.5
```
THRESHOLD 1.5
```

#perform 50 000 simulation steps
```
STEPS 50 000
```

#assign default value for a particular component
```
THRESHOLD X
```

#do not move component “B” during simulation
```
MOVE_STATE B fixed
```

#limit movements of component "B"
```
MOVE_STATE B 5 5 5 1 1 1 20 20 20 100 100 100 0 0
```

5 5 5 - refers to rotation around X, Y, Z axis in single move (X - up and down; Y- right, left, Z - back, forward)
1 1 1 - refers to translation in single move
20 20 20 - refers to maximal change in X, Y, Z coordinates during simulation (due to rotations)

100 100 100 - refers to maximal change in X, Y, Z coordinates during simulation (due to translations)
0 0 - are specific to rotation around covalents bonds

#start simulation from components orientations as provided in PDB files
`START_ORIENTATION True`

#include disorder/flexibility while modeling
`IDENTIFY_DISORDERS True`
`SIMUL_DD_FREQ 0.5 0.5`

4. PyRy3D**: Output**

As a result of the simulation, PyRy3D returns a set of PDB files comprising the best-scored conformations generated during the modeling. Files are saved on disc with frequency defined by the user (parameter WRITE_N_ITER in configuration file) and they are named according to the following formula:

`Outname_score_iteration_temperature.pdb`

Where:
`Outname` – name of output assigned by the user in "–o" option (by default "pyryresult" )
`Score` – score assigned by PyRy3D, the closer to zero, the more model fulfills the restraints
`Iteration` – number of iteration in which the particular model was created
`Temperature` – value of temperature parameter indicating the annealing progress in iteration when this model was saved

5. **Examples**

In this chapter we present how to run a command-line tool to perform some specific modeling tasks. The same operations can be done with server and GUI. We would like to recommend:

- A command-line tool for extensive runs of PyRy3D (multiple runs with large number of steps)
- GUI – for learning how PyRy3D works, but also to select the most accurate values of settings parameters
- Server – for running single runs of PyRy3D (they might consist of large numer of steps). Multiple runs of server might lead to increase a queue.

1. Docking structures with restraints

```
>>python pyry3d.py –fast –d coordinates.tar –r restraints.txt
```

Where:
`Coordinates.tar` contains PDB files of complex components
`Restraints.txt` contains information about docking restraints

`NOTES:`

- In this case MultiFASTA file with sequences of components is generated automatically. Sequences are derived from PDB files found in `coordinates.tar` archive
- Bear in mind that structures in `coordinates.tar` archive must be prepared

before running PyRy3D – each file should contain different chain name and the atoms/residues defined in `restraints.txt` file should occur in PDB files (the same names and numbers in PDB file and restraints file)

- If configuration file is not delivered by the user, PyRy3D is run with default parameters (see 3.4 section of this Manual to check their values)
- If name of output is not provided, PyRy3D puts all models into "pyryresults" folder

2. Docking into electron density map

```
>>python pyry3d.py –fast –d coordinates.tar –m
densitymap.ccp4
```

Where:
`Coordinates.tar` contains PDB files of complex components
`Densitymap.ccp4` is the map with electron density map (or one from negative stain) in CCP4 format

`NOTES:`

- In this case FASTA file with sequences of components is generated automatically from PDB files found in `coordinates.tar` archive
- Docking of components into the map will start from random locations of components inside the map

3. Docking into density map with use of restraints

```
>>python pyry3d.py –fast –d coordinates.tar –m
densitymap.ccp4 –r restraints.txt
```

`NOTES:`

- In this case be sure that residues used in restraint file are provided in corresponding PDB file with structures.

4. Modeling with usage of SAXS data

PyRy3D can work either with *ab initio* reconstruction or with raw data (SAXS curve). In first case the input is regular PDB file, in the second .DAT file.

```
>>python pyry3d.py –fast –d coordinates.tar –x abinitio.pdb
```

`OR`

```
>>python pyry3d.py –fast –d coordinates.tar –y curve.dat
```

Where:
`Abinitio.pdb` is the *ab initio* reconstruction calculated from SAXS data
`Curve.dat` is SAXS curve

`NOTES:`

- When *ab initio* model is used for modeling, PyRy3D performs docking in similar way as in case of a density map (the algorithm of scoring agreement with complex shape is similar)
- When SAXS curve is used, PyRy3D runs CRYSOL and scores agreement of theoretical curve calculated from obtained model with the curve delivered by the user

- For -y option CRYSOL must be available on disk and CRYSOL_PATH should be given in configuration file (to folder with *crysol.bin*)

5. Reproduce full-atom model from a reduced representation

```
>>python pyry3d.py –fast –d coordinates.tar –f fullatom.pdb –
v history.txt
```

To reduce computational time of simulation the user can use reduced representation of complex structure. In order to reproduce full-atom representation of final model generated by PyRy3D –f and –v options can be used. PyRy3D will rebuilt the full-atom model from the history of movements used during simulation.

Where:
`history.txt` is a name of file where history of modeling movements is saved in format recognizable by PyRy3D
`Fullatom.pdb` is a name for PDB file where the fullatom model will be saved

NOTES:

- Fullatom representation is reproduced only for the best-scored model generated during a single run of PyRy3D
- `Fullatom.pdb` and `history.txt` files are also used in PyRy3D GUI to record a movie of simulation

6. Starting modeling procedure from predefined orientations of some components

```
>>python pyry3d.py –fast –d coordinates.tar –c
config_file.txt
```

If the user wants to start a simulation from predefined model it is possible to achieve it with PyRy3D in two ways:

- If all components should have predefined starting positions, set the START_ORIENTATION parameter in configuration file as True (by default is False and modeling starts with random positions of components):
  ```
  START_ORIENTATION True
  ```

- If only some components should have predefined starting positions – use MOVE_STATE parameter and define how much these components can change their positions during simulation:
  ```
  MOVE_STATE movable A 5 5 5 10 10 10 360 360 360 1000
  1000 1000 5 30
  ```

7. Limiting movements of some components

```
>>python pyry3d.py –fast –d coordinates.tar –c config.txt
```

Movements of components can be either disabled:

- `MOVE_STATE B fixed #(in configuration file MOVE_STATE parameter is set as fixed for chain B)`

Or only limited:

- `MOVE_STATE movable A 5 5 5 0.1 0.1 0.1 10 10 10 0.1 0.1 1 5 30 #(chain A can move but not more than defined`

```
ranges)
```

8. Modeling disordered/flexible fragments of some components

```
>>python pyry3d.py –fast –d coordinates.tar –s
sequences.fasta –c config_file.txt
```

In this particular case, the user must provide several files:

`coordinates.tar` – an archive with PDB coordinates of components

`sequences.fasta` – MultiFASTA file with full-length sequences of all components

config_file.txt – a text file with chosen values of PyRy3D parameters.

When modeling disorder/flexibility the user must set two parameters:

`IDENTIFY_DISORDERS True`

Which turns on the procedures to find missing fragments in PDB files (that will be treated as disordered/flexible during the simulation)

`SIMUL_DD_FREQ 0.5 0.1`

Which defines how often a conformation of disordered/flexible parts will be changed at the beginning and at the end of simulation (0.5 and 0.1 respectively)

NOTES:

Be very careful when preparing input data for modeling with flexible/disordered regions. Follow the guidelines below:

- Sequences in MultiFASTA file must contain full length sequences (with sequence of disordered fragments included)
- Structures in PDB files must not contain any coordinates for disordered/flexible regions (remove any lines corresponding to those atoms)
- Numbering of residues in PDB files must be in agreement with the numbering in MultiFASTA file. Here are some examples:
    - if residues 1-5 in chain A are disordered their sequence is provided in FASTA file, but PDB file containing chain A starts from residue number 6
    - if residues 50-55 are disordered their sequence is provided in FASTA file, but PDB file contains residues 1-49 and then from 56 till the end of the chain
    - if a structure of entire component is unknown, then provide the full length sequence in MultiFASTA file, but call this sequence by chain name followed by "_" and a molecule type e.g. "A_protein", "B_RNA", "C_DNA" to indicate that this component will be treated as a simulated volume. Also no PDB file is provided for such a component.

General remarks:

- The number of structures you will obtain from a single run is always: STEPS/WRITE_N_ITER + 1.
- PyRy3D can be run without providing a configuration file by the user. In such a case, the program is run with default parameters (with values as mentioned in 3.4 Chapter of this Manual).
- For each run PyRy3D automatically generates pyry.log text file with values of parameters used and detailed scores assigned for models saved on disk.

## 6. **Additional Tools**

Additional tools are also included with the distribution.

### *6.3 Clustering*

Clustering is a tool for processing a set of alternative models by finding and grouping similar structures together into groups (called clusters) etc.. Benchmarks demonstrated that clustering provides is very useful in identification of most probable solutions.

The input for clustering tool delivered with PyRy3D is just a folder containing a set of models saved in PDB files. Typically those PDB files originate from multi-instance runs of PyRy3D. The output of clustering is a set of PDB files corresponding to subsequent clusters.

**Usage:**

```
>> cluster_complexes.py [<options>] [-o output] [-i input] [-t threshold] [-n struct_number]…
```

**Example usage:**

```
>> cluster_complexes.py –i alternativemodels –t 10 –n 1000 –o clustering_result.txt
```

**where:**

`alternativemodels` is a name of folder containing alternative models to be clustered,
`10` is the RMSD threshold in the set for clustering
`1000` is the number of best scored models to be considered while clustering
`clustering_result.txt` is a name of text file with clustering results

**there are also some additional options:**

`-s` for selecting best N structures criterion from selection of pyry3d (for score assigned to models by PyRy3D), cc (for cross-correlation coefficiency)
`-m` for selecting clustering criterion from selection of RMSD, GDT_TS and TMSCORE
`-r` for choosing representation of models, here clustering can be performed for full-atom models (-r fa), only Calfa or C4 ′ atoms (-r ca) or for centers of mass (-r sphere)
`--sort` this option generates new folders for five largest clusters containing PDB files assigned to those groups

**As a result (when –o –sort options were used to run the tool), a user will obtain:**
new folders for five largest clusters containing PDB files assigned to those groups
text file containing cluster members:

```
Clustered conformers number   3
poz2dna43_-238.979_225000_4.46819e-06.pdb     score   -238.979
poz2dna46_-240.066_295000_1.44237e-08.pdb     score   -240.066
poz2dna4_-240.49_30000_0.0154917.pdb  score   -240.49
Clustered conformers number   3
poz2dna24_-244.255_40000_0.00280232.pdb       score   -244.255
poz2dna39_-245.184_285000_0.000100337.pdb     score   -245.184
poz2dna26_-248.681_125000_0.000107724.pdb     score   -248.681
Clustered conformers number   2
poz2dna34_-234.2963_300000_4.13923698943e-05.pdb   score   -234.2963
poz2dna54_-235.7814_300000_6.45055979476e-06.pdb   score   -235.7814
Clustered conformers number   2
```

poz2dna92_-239.959_80000_0.00641644.pdb score -239.959
poz2dna45_-244.967_160000_8.1394e-07.pdb score -244.967

where for each cluster a number of members is given (3, 3, 2 and 2 in the above example) and names of PDB files belonging to these groups with corresponding scores assigned by PyRy3D.

NOTE: the current implementation of the clustering tool doesn't provide any cut-off for a number of clusters. Actually it identifies subsequent clusters until it exhausts the input data. In such cases, only the first few clusters are meaningful: the remaining clusters are usually insignificant. The user should inspect the clustering output to identify the percentage of structures that are contained in the 1st, 2nd, 3rd, and so on, set of clusters.

### *6.4* Ranking

Using rank_models.py tool attached to PyRy3D distribution, the user can create a ranking of alternative models according to PyRy3D scoring function. For instance, let's consider that a user generated a large set of alternative complex models using docking tool that does not allow to define distance restraints (or other restraints such as symmetry, solvent accessibility etc.). With help of rank_models.py tool one can evaluate these structures according to their agreement with distance restraints from chemical crosslinking, FRET or some other sources. Also, other types of restraints can be considered for this evaluation. Additionally, a ranking can be created with different PyRy3D configuration parameters values such as weights for clashes penalty or for quality of fit into the map (e.g. to check which model from the set of alternatives fits most accurately to the map).

**Usage:**

```
>> rank_models.py [<options>] [-m map] [-d structures] [-s
sequences] [-c config] [-r restraints]
```

**Example usage:**

```
>> rank_models.py -m map.ccp4 -d structures -s sequences.fasta -
c config.txt -r restraints.txt
```

**where:**

`-m map.ccp4` – electron density map of the complex (optional)
`-d structures` – name of a folder containing models to be ranked
`-s sequences.fasta` – name of file with sequences of components
`-c config.txt` – file with PyRy3D configuration parameters
`-r restraints.txt` – file with restraints

As a result (when –o –sort options were used), the user receives a text file with model names sorted according to PyRy3D score.

NOTES:

- all input files required for this tool should follow the same format rules as PyRy3D itself.
- much more options of ranking models are available in PyRy3D GUI launched with UCSF Chimera Extension (see chapter 6.6 for details). Also GUI provides graphical interface to run all these options.

### *6.5* Input Generator

Preparing input files for PyRy3D can be sometimes hard and time consuming. For this reason

we implemented a tool that automatically generates files ready for PyRy3D run called InputGenerator.py (distributed together with PyRy3D source).

The simplest way to use it is by running regular command-line program without delivering MultiFASTA sequences for components. In this case InputGenerator.py is called straightaway and PyRy3D generates this file for the user based on coordinates in PDB files.

However this tool offers a lot more options such as:

- renaming residues and chains of 3D coordinates of components
- defining disordered/flexible regions
- coming back to original names and numbering of components (operation reverse to generating input for PyRy3D)
- renaming chains and renumbering residues in files with restraints according to changes applied to structures and sequences
- and many more

NOTES:

- much more options of automatic generation of input files for PyRy3D program are available in PyRy3D GUI launched with UCSF Chimera Extension (see chapter 6.6 for details).

## *6.6 GUI*

We have created a tool that associates PyRy3D with UCSF Chimera, a popular program for interactive visualization and analysis of molecular structures. PyRy3D Chimera Extension is a plugin, that provides a user-friendly graphical interface, letting the user to generate a set of PyRy3D input files interactively, or to calculate a score for a set of different components' arrangements, based on default or user-defined parameters, directly from the extension's interface. It also offers many features designed to make the understanding and interpretation of PyRy3D results much easier.

**PyRy3D Chimera Extension installation**

The graphical user interface that we developed for PyRy3D is an extension to a popular molecular viewer, called UCSF Chimera. It is not a standalone program, so in order to use it, you must first install the viewer.

STEP 1: Download and install UCSF Chimera, version 1.9 or higher

STEP 2: Download BioPython 1.63 here and unpack it. Do not install it yet.

UCSF Chimera comes with its own Python copy. In order to use BioPython within Chimera on Linux, you need to install the library using Chimera's Python copy.

for Windows: Install BioPython 1.63 using the regular installer.

for Linux: Install BioPython 1.63 using Chimera's Python copy (Yes install BioPython, even if you have BioPython on your system already!):

1. Open the terminal.
2. Change your working directory to the previously unpacked BioPython folder (it must contain

the "setup.py" file)

3. Type the following command ("CHIMERA" here is the location where you have your Chimera installed):

   CHIMERA/bin/chimera --nogui --silent --script "setup.py install"

4. To check if the library was installed correctly, open Chimera, go to Tools -> General Controls -> IDLE and type:

```
>>import Bio
```

If it doesn't return any exception, it means BioPython is installed correctly.

STEP 3: Place the "PyRy3D_Extension" folder in `share` directory, located in the main UCSF Chimera installation directory.

## Loading input files

The Extension's first tab, called "*Models*" allows you to load your input files for further analysis. Use "*Add density map...*" and "*Add structures...*" buttons to list your shape descriptor and components of the complex. After you're done with preparing the list of your files (which can be seen in "*Maps ready to open*" and "*Structures ready to open*" fields), click the "*Load data*" button to load them into Chimera.

## Defining simulation / evaluation parameters

Before running an automatic complex evaluation or a classic PyRy3D simulation, a user can modify the default set of parameters. In order to do so, go to the Extension's second tab ("*Sim/Score*"). There, a user can use the "*parameters window*" to manually modify different parameters, or load his/her own configuration file using the "*Get parameters from configuration file*" option.

## Automatic complex scoring

Aside from performing PyRy3D modeling, one can use PyRy3D Chimera Extension to evaluate particular arrangement of components, that is set interactively in UCSF Chimera. This feature lets to use knowledge and/or intuition to manually arrange the components in the space and then check how it would be scored by PyRy3D.

After loading your input files using the "*Models*" tab, go to the Extension's second tab ("*Sim/Score*"), choose your output folder using the "*Browse*" button next to the "*Output directory*" field, and click "*Calculate PyRy3D score for displayed complex*".

## Performing PyRy3D simulation

The main purpose of PyRy3D software is conducting Monte Carlo simulations in order to find the best arrangement of components in a macromolecular complex. PyRy3D Chimera Extension also allows for using this feature directly from the Extension's interface. After the simulation is finished, a diagram is being displayed, showing the gradual changes of the complex score with the simulation progress.

After loading input files using the "*Models*" tab, go to the Extension's second tab ("*Sim/Score*"), choose your output folder using the "*Browse*" button next to the "*Output directory*" field, and

click "*Perform PyRy3D Simulation*".

## Results Analysis

One of the most important goals of PyRy3D Chimera Extension is to make the PyRy3D's results analysis easier and more understandable. In order to achieve this goal, we've created a set of features that allow for visualizing how PyRy3D scores particular structures with given set of parameters. The user can show every element of PyRy3D scoring function, such as regions outside the simulation box, or regions inside the density map.

The "*Results display*" window pops up after finishing the automatic evaluation of visualized complex and after finishing a PyRy3D simulation. It contains a PyRy3D score (and its individual components), and a set of buttons allowing the user to:

- Display the simulation box
- Display the simulation grid
- Highlight the complexes disordered regions
- Highlight collisions between components
- Highlight regions outside the simulation box
- Highlight regions inside the density map
- Highlight regions of the map that are empty
- Highlight regions outside of the density map
- Display user-defined restraints (if provided)

Each feature can be visualized in user's color of choice. After a user picks a color from the palette, the square button will displayed the user-chosen color.

## Generating input files

PyRy3D Chimera Extension offers a possibility to quickly generate files that can be used to perform simulations using the PyRy3D server or standalone version. Even though you can run PyRy3D simulations from the Chimera Extension's interface, it's much more efficient to perform the simulations using the original, command line - based, PyRy3D software.

The Extension's third tab ("*Input Gen*") contains a set of checkboxes. By checking and un-checking those checkboxes, one can choose the types of input files that will be saved in the older of choice (selected with "*Generate files in*" option) and clicking the "*Generate defined input files*".

The types of input files are:

- Structures (generated using the .PDB files loaded into Chimera using the first tab)
- Density map (simple copy of map loaded into Chimera using the first tab)
- Sequences (automatically generated based on the structures loaded into Chimera)
- Restraints (file is copied from a localization indicated in "*Restraints file*" field)
- Configuration file (generated using the parameters defined by the user in the "*Parameters window*")

## Generating movies

After the simulation is finished, a user can investigate the whole process thanks to the animation generator. The tool lets to generate animations that can be saved in one of many popular video file formats.

A user can find the parameters necessary to generate an animation on the fourth tab, called "*Animations*".. The camera's position does not change during the entire animation's recording. For this reason, the user has to first define the camera's angle by opening and positioning his density map in the Chimera window (using standard manipulation tools).

To generate a move, a user is supposed to define:

- "*PyRy3D output files*" folder – the folder that contains a set of models generated by a single PyRy3D simulation. These models will serve as frames for the animation.
- A fraction of models that will serve as frames – in the "*Record 1/n of the files*", the user can define the fraction of models that will constitute the animation's frames. For example, if the user has generated 1000 files, but wants the animation to be faster and only show 100 of them, the "n" number should equal 10 ("Record 1/10 of the files"). After defining the fraction, the "*Number of frames*" label should inform the user about the resulting total number of frames that will be recorded.
- The output movie's file path – using the "*Save movie file as*" field.
- The output movie's format – using the "*Movie format*" menu.
- The folder that will store frame images – before generating a movie file, Chimera must first take screenshots of each of the frames and store all of them in one location. The user can define this location using the "*Save images in*" field.
- "*Movie size*" – the width and height of the output movie.

**Create ranking of multiple complexes**

The PyRy3D Chimera Extension allows for a quick evaluation of multiple complexes in order to create a ranking based on criteria chosen by the user (e.g. distance restraints). Thanks to this tool a user can evaluate complexes based on different set of PyRy3D configuration parameters (level of clashes, quality of fit into map) or verify structures built with other tools (imagine having 1000 models generated with docking tool (without any prior data about their possible interactions) and then creating a ranking with PyRy3D that sorts models according to their agreement with external data such as distances from chemical crosslinking or FRET).

The "*Ranking*" tab is the fifth tab of the PyRy3D Chimera Extension. Within this tab, a user can define:

- *The ranked models' source* – using the radio buttons in the first field of the tab, a user can choose to load the input models from Chimera's window, or from a previously prepared folder. In the latter case, the files should be already tared (using, for example, the Extension's input generator)
- *The configuration file's source* – using the tools available in the second field, the user can define if the configuration file is supposed to be generated automatically based on the "Parameters window", or if the user's previously prepared file should be used.
- If experimental restraints will be used – if so, the user can check the "*Use restraints file from hard drive*" radio button, and point to the file's location using the "*Restraints file*" field.
- *The sequences file's source* – using the tools available in the fourth field, the user can define whether the sequences are supposed to be generated automatically based on the structures loaded into Chimera, or if they should be loaded from a previously prepared file.

- *Map file* – an electron density map.
- Sorting methods – using the "*Sort results by*" menu on the bottom of the tab, the user can choose if the ranking should be based on the overall PyRy3D score, or on one of its components.

After defining all the necessary parameters, the user can click the "*Create ranking*" button to start evaluating multiple complexes.

**Clustering complexes**

An option to cluster models generated from multiple PyRy3D simulations is also available from the PyRy3D Chimera Extension.

The last tab of the Extension, titled "*Clustering*", contains a set of parameters necessary to perform a clustering of multiple complexes:
- "*Input directory*" – the directory that contains the files that will be clustered.
- "*Map file*" - electron density map file
- "*Score type*" - the score that will be used to arrange the files in the output file
- "*Measure type*" - the measure that will be used to compare the structures
- "*Representation*" - the models can be compared based on their full structures ("fa"), on their backbones only ("ca"), or on their approximate shapes ("sphere")
- "*Number of structures*" - the number of highest scored structures from the indicated directory, that will be shown in the clustering output
- "*Clustering threshold*" - the threshold that will be used to group similar models into clusters
- "*Output directory*" - the directory that will contain the output files.

After defining all the necessary parameters, press the "*Start clustering*" button to start the clustering process.

More materials about PyRy3D Chimera Extension can be found on the following website: http://genesilico.pl/pyry3d/tutorials