PDB Manip Py Documentation

Samuel Murail

Mar 27, 2023

TABLE OF CONTENTS:

Pdb_Manip_py is a python library allowing simple operations on pdb coor files.

- **Online Documentation:**
  https://pdb-manip-py.readthedocs.io
- **Source code repository:**
  https://github.com/samuelmurail/pdb_manip_py

CHAPTER

ONE

MAIN FEATURES:

- **Basic pdb operations**
  - pdb wrapper
  - selection tools
  - rotation
  - translation
  - alignement
  - RMSD calculation
  - insertion of molecules in solvant
  - …

CHAPTER

TWO

INSTALLATION

2.1 Using Pypi

```
pip3 install pdb_manip_py
```

2.2 Using Conda

```
conda install pdb_manip_py -c conda-forge
```

2.3 From source code

Get the os_command_py library from github.

```
git clone https://github.com/samuelmurail/pdb_manip_py.git
./setup.py install --user
```

CHAPTER

THREE

AUTHOR

- Samuel Murail, Associate Professor - Université Paris Diderot, CMPLI.

See also the list of contributors who participated in this project.

CHAPTER

FOUR

LICENSE

This project is licensed under the GNU General Public License v2.0 - see the `LICENSE` file for details.

CHAPTER

FIVE

INSTALLATION

5.1 Using Pypi

```
pip3 install pdb_manip_py
```

5.2 Using Conda

```
conda install pdb_manip_py -c conda-forge
```

5.3 From source code

The sources for Docking Python can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/samuelmurail/pdb_manip_py
```

Or download the tarball:

```
$ curl -OJL https://github.com/samuelmurail/pdb_manip_py/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

5.4 Test installation

Launch test with doctest, will check that module's docstrings are up-to-date by verifying that all interactive examples still work as documented.

```
$ pytest
============================= test session starts ==============================
platform darwin -- Python 3.8.2, pytest-5.4.1, py-1.8.1, pluggy-0.13.1
rootdir: /Users/smurail/Documents/Code/pdb_manip_py, inifile: pytest.ini
plugins: cov-2.8.1
```

(continues on next page)

(continued from previous page)

```
collected 40 items

pdb_manip_py/pdb2pqr.py .                                                [  2%]
pdb_manip_py/pdb_manip.py ......................................         [100%]

=============================== warnings summary ===============================
pdb_manip_py/pdb_manip.py::pdb_manip_py.pdb_manip.Coor.align_principal_axis
  /Users/smurail/opt/miniconda3/envs/docking/lib/python3.8/site-packages/scipy/spatial/
↪transform/rotation.py:1953: UserWarning: Optimal rotation is not uniquely or poorly␣
↪defined for the given sets of vectors.
    warnings.warn("Optimal rotation is not uniquely or poorly defined "

-- Docs: https://docs.pytest.org/en/latest/warnings.html
======================== 40 passed, 1 warning in 11.64s ========================
```

CHAPTER

SIX

USAGE

To use PDB Manip Python in a project:

```
[1]: from pdb_manip_py import pdb_manip
     pdb_manip.show_log()
```

6.1 Create a `Coor()` object

You can either get the coordinates from the Protein Data Bank:

```
[2]: coor_1hsg = pdb_manip.Coor()
     coor_1hsg.get_PDB('1hsg')
```

```
Succeed to read file 1hsg.pdb ,  1686 atoms found
```

Or load a local stored file:

```
[3]: coor_1hsg = pdb_manip.Coor('./1hsg.pdb')
```

```
Succeed to read file 1hsg.pdb ,  1686 atoms found
```

6.2 Extract selection of coordinates

You can extract a selection of coordinates, here we will use the `1hsg.pdb` PDB file and extract the coordinates of L-735,524 an inhibitor of the HIV proteases (resname `MK1`):

```
[4]: # Select res_name MK1
     lig_coor = coor_1hsg.select_part_dict(selec_dict={'res_name': ['MK1']})
```

The obtain selection can be saved using the `write_pdb()` function:

```
[5]: # Save the ligand coordinates
     lig_coor.write_pdb('1hsg_lig.pdb')
```

```
Succeed to save file 1hsg_lig.pdb
```

For selection you can use :

- `name` for atom name
- `alter_loc` alternative location

- `res_name` residue name
- `chain` chain ID
- `res_num` residue number
- `uniq_resid` a unique residue number starting from 0
- `insert_res` insert residue
- `xyz` x, y, z, coordinates
- `occ` occupation
- `beta` beta factor

Selector can be combined, *eg.* to select residue names Proline and Alanine from chain A you can use:

[6]:
```
PRO_ALA_A_coor = coor_1hsg.select_part_dict(
    selec_dict={'res_name': ['PRO', 'ALA'], 'chain': ['A']})
```

6.2.1 *note:*

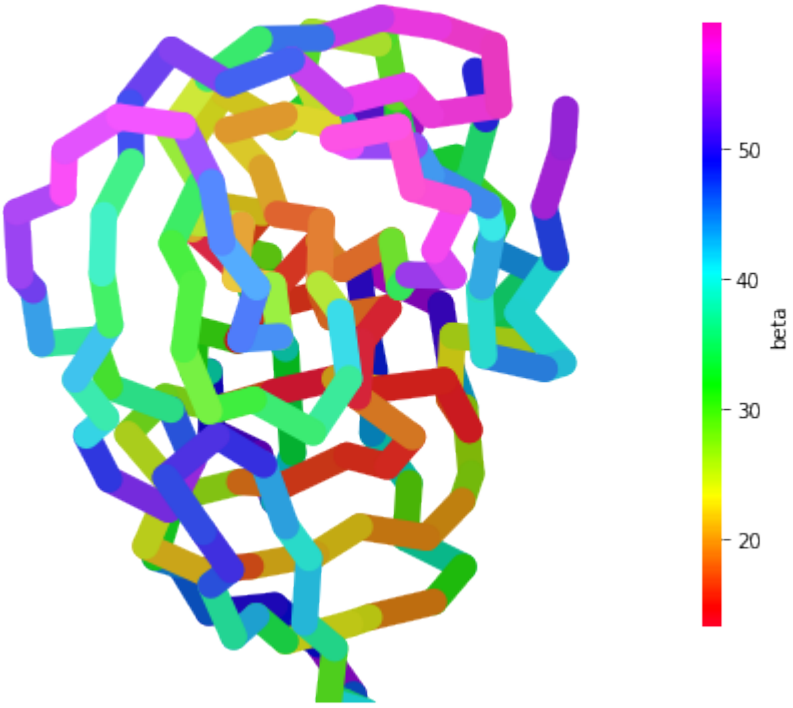
To select protein atoms you can use the PROTEIN_AA variable containing protein residue name, giving this selector:

```
selec_dict={'res_name': pdb_manip.PROTEIN_AA)}
```

6.3 Visualize coordinates in pseudo 3D

- using matplotlib
- Inspired from sokrypton in https://github.com/sokrypton/ColabFold

[7]:
```
_ = coor_1hsg.plot_pseudo_3D('beta')
```



6.4 Visualize coordinates in 3D

You can use the `nglview` library to vizualise in 3D your protein.

[8]:
```
view = coor_1hsg.view
view
```

```
_ColormakerRegistry()
```

```
NGLWidget()
```

[11]:
```
IFrame(src='../_static/1hsg.html', width=600, height=300)
```

[11]:
```
<IPython.lib.display.IFrame at 0x7f89d8f58150>
```

CHAPTER

SEVEN

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

7.1 Types of Contributions

7.1.1 Report Bugs

Report bugs at https://github.com/samuelmurail/pdb_manip_py/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

7.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

7.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

7.1.4 Write Documentation

Docking Python could always use more documentation, whether as part of the official Docking Python docs, in docstrings, or even on the web in blog posts, articles, and such.

7.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/samuelmurail/pdb_manip_py/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

7.2 Get Started!

Ready to contribute? Here's how to set up *pdb_manip_py* for local development.

1. Fork the *pdb_manip_py* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/pdb_manip_py.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv pdb_manip_py
$ cd pdb_manip_py/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 pdb_manip_py tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

7.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/samuelmurail/pdb_manip_py/pull_requests and make sure that the tests pass for all supported Python versions.

7.4 Tips

To run a subset of tests:

```
$ pytest tests.test_pdb_manip_py
```

7.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

EIGHT

COOR CLASS

class pdb_manip_py.pdb_manip.**Coor**(*coor_in=None*, *pdb_lines=None*)

Topologie base on coordinates like pdb or gro.

The coor object containt a dictionnary of atoms indexed on the atom num and the crystal packing info.

Parameters

- **atom_dict** (*dict*) – dictionnary of atom
- **crystal_pack** (*str*) – crystal packing

Atom dictionnary parameters

Parameters

- **field** (*str*) – pdb field
- **num** (*int*) – atom number
- **name** (*str*) – atom name
- **alter_loc** (*str*) – atom number
- **res_name** (*str*) – residue name (3 letters)
- **chain** (*str*) – chain ID
- **res_num** (*int*) – residue number (based on pdb file)
- **uniq_resid** (*int*) – unique residue number
- **insert_res** (*str*) – atom number
- **xyz** – coordinate
- **occ** (*float*) – occupation
- **beta** (*float*) – beta flactor

Note: The atom num index in the dictionnary, is not the same as the `atom_num` field of the dictionnary.

Note: Files necessary for testing : ../test/input/1y0m.pdb, ../test/input/1rxz.pdb and ../test/input/4n1m.pdb. To do the unitary test, execute pdb_mani.py (-v for verbose mode)

Todo: Add an atom class ?

add_zinc_finger(*ZN_pdb*, *cutoff=3.2*)

Change protonation state of cysteins and histidine coordinating Zinc atoms. To do after *correct_his_name* and *correct_cys_name*, in order that protonation is recognize by pdb2gmx.

Example

```
>>> try:
...   print("Start import")
...   from . import pdb2pqr
... except ImportError:
...   import pdb2pqr
Start import...
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> show_log()
>>> # Read the pdb 1jd4 and keep only chain A
>>> input_pdb = Coor(os.path.join(TEST_PATH, '1jd4.pdb'))
Succeed to read file ...1jd4.pdb ,  1586 atoms found
>>> chain_A = input_pdb.select_part_dict(selec_dict={'chain': ['A']})
>>> chain_A.write_pdb(os.path.join(TEST_OUT, '1jd4_A.pdb'))
Succeed to save file ...1jd4_A.pdb
>>>
>>> # Compute protonation with pdb2pqr:
>>> pdb2pqr.compute_pdb2pqr(os.path.join(TEST_OUT, '1jd4_A.pdb'),os.path.
↪join(TEST_OUT, '1jd4.pqr'))
Succeed to read file ...1jd4_A.pdb ,  793 atoms found
Succeed to save file ...tmp_pdb2pqr.pdb
pdb2pqr30... --ff CHARMM --ffout CHARMM --keep-chain --titration-state-
↪method=propka --with-ph=7.00...tmp_pdb2pqr.pdb ...1jd4.pqr
0
>>> prot_coor = Coor(os.path.join(TEST_OUT, '1jd4.pqr'))
Succeed to read file ...1jd4.pqr ,  1549 atoms found
>>> prot_coor.correct_cys_name()
<...Coor object at 0x...
>>> prot_coor.correct_his_name()
<...Coor object at 0x...
>>> prot_coor.correct_chain()
Chain: A  Residue: 0 to 95
<...Coor object at 0x...
>>> ZN_index = prot_coor.get_index_selection({'name': ['ZN']})
>>> print(len(ZN_index))
0
>>> prot_coor.add_zinc_finger(os.path.join(TEST_OUT, '1jd4_A.pdb'))
Succeed to read file ...1jd4_A.pdb ,  793 atoms found
Presence of 1 Zinc detected
change cystein residue(s) : [48, 51, 75]
change histidine residue(s) : [68]
True
>>> ZN_index = prot_coor.get_index_selection({'name': ['ZN']})
>>> print(len(ZN_index))
1
```

Note: This function seems useless. Since last version of pdb2pqr residue name seems correct.

align_principal_axis(*axis=2*, *vector=[0, 0, 1]*, *selec_dict={}*)

Align principal axis with index *axis* with *vector*.

Taken from: https://github.com/MDAnalysis/mdanalysis/blob/develop/package/MDAnalysis/core/topologyattrs.py

Parameters

- **axis** (*int (Default 2)*) – principal axis to align (0, 1, or 2)
- **vector** (*list (Default z axis)*) – vector to align
- **selec_dict** (*dict, default={'name': ['CA']}*) – selection dictionnary

Example

```
>>> show_log()
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> prot_coor.align_principal_axis()
Do a rotation of 66.43°
>>> prot_coor.align_principal_axis()
Do a rotation of 0.00°
```

static align_seq(*seq_1*, *seq_2*, *gap_cost=-8*, *gap_extension=-2*)

Align two amino acid sequences using the Waterman - Smith Algorithm.

Parameters

- **seq_1** (*str*) – amino acid sequence 1
- **seq_2** (*str*) – amino acid sequence 2
- **gap_cost** (*int (Default -8)*) – Gap cost
- **gap_extension** (*int (Default -2)*) – Gap extension cost

Returns
the two aligned sequences

Return type
str, str

Example

```
>>> seq_1 =
↪'AQDMVSPPPPIADEPLTVNTGIYLIECYSLDDKAETFKVNAFLSLSWKDRRLAFDPVRSGVRVKTYEPEAIWIPEIRFVNVENARDADVVDI
↪'
>>> seq_2 =
↪'APSEFLDKLMGKVSGYDARIRPNFKGPPVNVTCNIFINSFGSIAETTMDYRVNIFLRQQWNDPRLAYSEYPDDSLDLDPSMLDSIWKPDLFI
↪'
>>> align_seq_1, align_seq_2 = Coor.align_seq(seq_1, seq_2)
>>> Coor.print_align_seq(align_seq_1, align_seq_2)
-------------AQDMVSPPPPIADEPLTVNTGIYLIECYSLDDKAETFKVNAFLSLSWKDRRLAFDPVRS-GVRVKTY
             |   |   * |   *||*| |*|| |  *| | |  ||** **  |*|* ***||   | || |
APSEFLDKLMGKVSGYDARIRPNFKGPPVNVTCNIFINSFGSIAETTMDYRVNIFLRQQWNDPRLAYSEYPDDSLDLDPS

EPEAIWIPEIRFVNVENARDADVV----DISVSPDGTVQYLERFSARVLSPLDFRRYPFDSQTLHIYLIVRSVDTRNIVL
  ||** *|| *|* ||*|  |*|     | |* |*|* *  *||  |  *|*||||*|* **  | *   |   |||||
MLDSIWKPDLFFANEKGANFHEVTTDNKLLRISKNGNVLYSIRITLVLACPMDLKNFPMDVQTCIMQLESFGYTMNDLIF
```

(continues on next page)

(continued from previous page)

```
AVDLEKVGKNDDVFLTGWDIESFTAVV-KPANFALEDRLESK---LDYQLRISRQYFSYIPNIILPMLFILFISWTAFW-
  * ** *  |     *  | |*     *  ||| |    |*   || |||||**   *| || |* *|*|||**||**
EWD-EK-GAVQ--VADGLTLPQFILKEEKDLRYCTKHYNTGKFTCIEARFHLERQMGYYLIQMYIPSLLIVILSWVSFWI

-STSYEANVTLVVSTLIAHIAFNILVETNLPKTPYMTYTGAIIFMIYLFYFVAVIEVTVQHYL-KVESQP--ARAASITR
   |  *|* * ||*|||  | |   |||***| *|      | |  ** * *||* || ||| || ||   |**  *
NMDAAPARVGLGITTVLTMTTQSSGSRASLPKVSYVKAIDIWMAVCLLFVFSALLEYAAVNFIARAGTKLFISRAKRIDT

ASRIAFPVVFLLANIILAFLFFGF-----
|**|***|***| **   * || |
VSRVAFPLVFLIFNI---FYWITYKLVPR
```

align_seq_coor_to(*atom_sel_2*, *chain_1=['A']*, *chain_2=['A']*, *back_names=['CA']*, *align=True*, *rmsd_flag=True*, *tmscore_flag=False*)

Align 2 strucures, using a sequence alignement to determine which residue to align. Compute RMSD between two atom_dict Then return the RMSD value.

Parameters

- **atom_sel_2** (*dict*) – atom dictionnary
- **chain_1** (*list*) – list of chain
- **chain_2** (*list*) – list of chain

Returns
rmsd and alignement index

Return type
float and list

Example

```
>>> prot_1_coor = Coor(os.path.join(TEST_PATH, '1jd4.pdb'))
Succeed to read file ...1jd4.pdb ,  1586 atoms found
>>> prot_2_coor = Coor(os.path.join(TEST_PATH, '1dpx.pdb'))
Succeed to read file ...1dpx.pdb ,  1192 atoms found
>>> rmsd, align_sel = prot_1_coor.align_seq_coor_to(prot_2_coor)
------NYFPQYPEYAIETARLRTFEAWPRNLKQKPHQLAEAGFFYTGVGDRVRCFSCGGGLMDW-NDNDEPWEQHALWL
      |      |||||| *  ||  *      * ||  ||     ||   |       |   * **|  * ||| *
KVFGRCELAAAMKRHGLDNYRGYSLGNW--VCAAKFESNFNTQATNRNTDGSTDYGILQINSRWWCNDGRTPGSRN-LCN

SQCRFVKLMKGQLYIDTVAAKPV------------------------------
  *    *|||||  || |** |
IPCS--ALLSSDITASVNCAKKIVSDGNGMNAWVAWRNRCKGTDVQAWIRGCRL

>>> print('RMSD = {:.2f} Å'.format(rmsd))
RMSD = 12.63 Å
```

align_to(*atom_sel_2*, *selec_dict={'name': ['CA']}*, *index_list=None*, *rot_kabsch=True*)

Align structure to an Coor object.

Parameters

- **atom_sel_1** (*dict*) – atom dictionnary

- **atom_sel_2** (*dict*) – atom dictionnary
- **selec_dict** (*dict, default={'name': ['CA']}*) – selection dictionnary
- **rot_kabsch** (*bool, default=True*) – method for rotation kabsh, if not quaternion

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1jd4.pdb'))
Succeed to read file ...1jd4.pdb ,  1586 atoms found
>>> chain_A = prot_coor.select_part_dict(selec_dict={'chain': ['A']})
>>> chain_B = prot_coor.select_part_dict(selec_dict={'chain': ['B']})
>>> rmsd = chain_A.compute_rmsd_to(chain_B)
>>> print('RMSD before alignement is {:.2f} Å'.format(rmsd))
RMSD before alignement is 37.47 Å
>>> chain_A.align_to(chain_B)
>>> rmsd = chain_A.compute_rmsd_to(chain_B)
>>> print('RMSD after alignement is {:.2f} Å'.format(rmsd))
RMSD after alignement is 0.06 Å
>>> chain_A.align_to(chain_B)
>>> rmsd = chain_A.compute_rmsd_to(chain_B)
>>> print('RMSD after 2nd alignement is {:.2f} Å'.format(rmsd))
RMSD after 2nd alignement is 0.06 Å
```

static angle_vec(*vec_a*, *vec_b*)

Compute angle between two vectors.

Parameters

- **vec_a** (*list*) – vector
- **vec_b** (*list*) – vector

Returns
angle in radian

Return type
float

Example

```
>>> angle = Coor.angle_vec([1, 0, 0], [0, 1, 0])
>>> print('angle = {:.2f}'.format(np.degrees(angle)))
angle = 90.00
>>> angle = Coor.angle_vec([1, 0, 0], [1, 0, 0])
>>> print('angle = {:.2f}'.format(np.degrees(angle)))
angle = 0.00
>>> angle = Coor.angle_vec([1, 0, 0], [1, 1, 0])
>>> print('angle = {:.2f}'.format(np.degrees(angle)))
angle = 45.00
>>> angle = Coor.angle_vec([1, 0, 0], [-1, 0, 0])
>>> print('angle = {:.2f}'.format(np.degrees(angle)))
angle = 180.00
```

static atom_angle(*atom_a*, *atom_b*, *atom_c*)

Compute the anlge between 3 atoms.

Parameters

- **atom_a** (*dict*) – atom dictionnary
- **atom_b** (*dict*) – atom dictionnary
- **atom_c** (*dict*) – atom dictionnary

Returns
angle (degrees)

Return type
float

Example

```
>>> atom_1 = {'xyz': np.array([0.0, 0.0, 0.0])}
>>> atom_2 = {'xyz': np.array([0.0, 1.0, 0.0])}
>>> atom_3 = {'xyz': np.array([1.0, 1.0, 1.0])}
>>> Coor.atom_angle(atom_1, atom_2, atom_3)
90.0
>>> print('{:.3f}'.format(Coor.atom_angle(atom_1, atom_3, atom_2)))
35.264
```

static atom_dihed_angle(*atom_a*, *atom_b*, *atom_c*, *atom_d*)

Compute the dihedral anlge using 4 atoms.

Parameters

- **atom_a** (*dict*) – atom dictionnary
- **atom_b** (*dict*) – atom dictionnary
- **atom_c** (*dict*) – atom dictionnary
- **atom_d** (*dict*) – atom dictionnary

Returns
dihedral angle

Return type
float

Example

```
>>> atom_1 = {'xyz': np.array([0.0, -1.0, 0.0])}
>>> atom_2 = {'xyz': np.array([0.0, 0.0, 0.0])}
>>> atom_3 = {'xyz': np.array([1.0, 0.0, 0.0])}
>>> atom_4 = {'xyz': np.array([1.0, 1.0, 0.0])}
>>> atom_5 = {'xyz': np.array([1.0, -1.0, 0.0])}
>>> atom_6 = {'xyz': np.array([1.0, -1.0, 1.0])}
>>> angle_1 = Coor.atom_dihed_angle(atom_1, atom_2, atom_3, atom_4)
>>> print('{:.3f}'.format(angle_1))
180.000
>>> angle_2 = Coor.atom_dihed_angle(atom_1, atom_2, atom_3, atom_5)
>>> print('{:.3f}'.format(angle_2))
0.000
>>> angle_3 = Coor.atom_dihed_angle(atom_1, atom_2, atom_3, atom_6)
>>> print('{:.3f}'.format(angle_3))
-45.000
```

static atom_dist(*atom_a*, *atom_b*)

Compute the distance between 2 atoms.

Parameters

- **atom_a** (*dict*) – atom dictionnary
- **atom_b** (*dict*) – atom dictionnary

Returns
distance

Return type
float

Example

```
>>> atom_1 = {'xyz': np.array([0.0, 0.0, 0.0])}
>>> atom_2 = {'xyz': np.array([0.0, 1.0, 0.0])}
>>> atom_3 = {'xyz': np.array([1.0, 1.0, 1.0])}
>>> Coor.atom_dist(atom_1, atom_2)
1.0
>>> Coor.atom_dist(atom_1, atom_3)
1.7320508075688772
```

center_of_mass(*selec_dict={}*)

Compute the center of mass of a selection Avoid using atoms with 2 letters atom name like NA Cl ... If selection is empy, take all atoms.

Parameters
selec_dict (*dict, default={}*) – selection dictionnary

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> com_1y0m = prot_coor.center_of_mass()
>>> print("x:{:.2f} y:{:.2f} z:{:.2f}".format(*com_1y0m))
x:16.01 y:0.45 z:8.57
>>> com_1y0m_ca = prot_coor.center_of_mass({'name': ['CA']})
>>> print("x:{:.2f} y:{:.2f} z:{:.2f}".format(*com_1y0m_ca))
x:15.95 y:0.72 z:8.96
```

Warning: Atom name must start with its type letter (H, C, N, O, P, S).

centroid(*selec_dict={}*)

Compute the centroid of a selection If selection is empy, take all atoms.

Parameters
selec_dict (*dict, default={}*) – selection dictionnary

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> com_1y0m = prot_coor.centroid()
```

(continues on next page)

(continued from previous page)

```
>>> print("x:{:.2f} y:{:.2f} z:{:.2f}".format(*com_1y0m))
x:16.03 y:0.44 z:8.57
>>> com_1y0m_ca = prot_coor.centroid(selec_dict={'name': ['CA']})
>>> print("x:{:.2f} y:{:.2f} z:{:.2f}".format(*com_1y0m_ca))
x:15.95 y:0.72 z:8.96
```

change_index_pdb_field(*index_list*, *change_dict*)

Change all atom field of a part of coor object defined by `index`, the change is based on the change_dict dictionnary.

Parameters

- **index_list** (*list*) – list of atom index to change
- **change_dict** (*dict*) – change ditionnay eg. {"chain" : "A"}

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> res_826_852 = prot_coor.get_index_selection({'res_num' :
... range(826,852)})
>>> prot_coor.change_index_pdb_field(index_list=res_826_852,
... change_dict={"chain" : "B"})
<...Coor object at ...>
>>> prot_seq = prot_coor.get_aa_seq()
>>> prot_seq == {'A': 'TFKSAVKALFDYKAQREDELTFTKSAIIQNVEKQD',
... 'B': 'GGWWRGDYGGKKQLWFPSNYVEEMIN'}
True
```

change_pdb_field(*change_dict*)

Change all atom field of a coor object, the change is based on the change_dict dictionnary.

Parameters

change_dict (*dict*) – change ditionnay eg. {"chain": "A"}

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> prot_coor.get_aa_seq()
{'A': 'TFKSAVKALFDYKAQREDELTFTKSAIIQNVEKQDGGWWRGDYGGKKQLWFPSNYVEEMIN'}
>>> prot_coor.change_pdb_field(change_dict = {"chain": "B"})
<...Coor object at ...>
>>> prot_coor.get_aa_seq()
{'B': 'TFKSAVKALFDYKAQREDELTFTKSAIIQNVEKQDGGWWRGDYGGKKQLWFPSNYVEEMIN'}
```

compute_TMscore_to(*atom_sel_2*, *ltarget*, *index_list=None*)

Compute TMscore between two atom_dict Then return the TMscore.

Parameters

- **atom_sel_1** (*dict*) – atom dictionnary
- **atom_sel_2** (*dict*) – atom dictionnary
- **selec_dict** (*dict, default={}*) – selection dictionnary

Returns
TMscore

Return type
float

Ref: Y. Zhang, J. Skolnick, Scoring function for automated assessment of protein structure template quality, Proteins, 57: 702-710 (2004)

TMscore = max(1/Ltarget * Sum(1/(1 + (di/do(Ltarget))**2)))

with Ltarget is length of model protein and: d0(Ltarget) = 1.24 * (Ltarget-15)**(1/3) - 1.8

compute_rmsd_to(*atom_sel_2*, *selec_dict={'name': ['CA']}*, *index_list=None*)

Compute RMSD between two atom_dict Then return the RMSD value.

Parameters

- **atom_sel_1** (*dict*) – atom dictionnary
- **atom_sel_2** (*dict*) – atom dictionnary
- **selec_dict** (*dict, default={}*) – selection dictionnary

Returns
RMSD

Return type
float

static concat_pdb(**pdb_in_files*, *pdb_out*, *check_file_out=True*)

Concat a list of pdb files in one.

Parameters

- **pdb_in_files** (*list*) – list of pdb files
- **pdb_out** (*dict*) – atom dictionnary

Example

```
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> Coor.concat_pdb(os.path.join(TEST_PATH, '1y0m.pdb'),
...                 os.path.join(TEST_PATH, '1rxz.pdb'),
...                 pdb_out = os.path.join(TEST_OUT, 'tmp_2.pdb'))
Succeed to save concat file: ...tmp_2.pdb
```

correct_chain(*Ca_cutoff=4.5*)

Correct the chain ID's of a coor object, by checking consecutive Calphas atoms distance. If the distance is higher than `Ca_cutoff` , the former atoms are considered as in a different chain.

Parameters
Ca_cutoff (*float, default=4.5*) – cutoff for distances between Calphas atoms (X)

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> res_810 = prot_coor.get_index_selection({'res_num': [810]})
>>> prot_coor = prot_coor.del_atom_index(index_list=res_810)
>>> prot_coor.get_aa_seq()
Residue A:THR:811 is not consecutive, there might be missing residues
```

(continues on next page)

(continued from previous page)

```
{'A': 'TFKSAVKALFDYKAQREDE-TFTKSAIIQNVEKQDGGWWRGDYGGKKQLWFPSNYVEEMIN'}
>>> prot_coor.correct_chain()
Chain: A  Residue: 0 to 18
Chain: B  Residue: 20 to 60
<...Coor object at ...>
>>> # As a residue is missing, Calphas after residue 18 is no
>>> # more consecutive
```

Note: This is specially usefull for pdb2gmx which cut the protein chains based on the chain ID's.

correct_cys_name()

Correct the CYS resname from pdb2pqr

Example

```
>>> try:
...   print("Start import")
...   from . import pdb2pqr
... except ImportError:
...   import pdb2pqr
Start import...
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> # Compute protonation with pdb2pqr:
>>> pdb2pqr.compute_pdb2pqr(os.path.join(TEST_PATH, '1dpx.pdb'),os.path.
↪join(TEST_OUT, '1dpx.pqr'))
Succeed to read file ...1dpx.pdb ,  1192 atoms found
Succeed to save file ...tmp_pdb2pqr.pdb
pdb2pqr30... --ff CHARMM --ffout CHARMM --keep-chain --titration-state-
↪method=propka --with-ph=7.00 ...tmp_pdb2pqr.pdb ...1dpx.pqr
0
>>> prot_coor = Coor(os.path.join(TEST_OUT, '1dpx.pqr'))
Succeed to read file ...1dpx.pqr ,  1960 atoms found
>>> Isu_index = prot_coor.get_index_selection({'res_name': ['DISU']})
>>> print(len(Isu_index))
16
>>> prot_coor.correct_cys_name()
<...Coor object at 0x...
>>> Isu_index = prot_coor.get_index_selection({'res_name': ['DISU']})
>>> print(len(Isu_index))
0
```

correct_his_name()

Get his protonation state from pdb2pqr and replace HIS resname with HSE, HSD, HSP resname. To do after pdb2pqr, in order that protonation is recognize by pdb2gmx.

Example

```
>>> try:
...   print("Start import")
...   from . import pdb2pqr
... except ImportError:
```

(continues on next page)

(continued from previous page)

```
...   import pdb2pqr
Start import...
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> # Compute protonation with pdb2pqr:
>>> pdb2pqr.compute_pdb2pqr(os.path.join(TEST_PATH, '4n1m.pdb'),os.path.
↪join(TEST_OUT, '4n1m.pqr'))
Succeed to read file ...4n1m.pdb ,  2530 atoms found
Succeed to save file ...tmp_pdb2pqr.pdb
pdb2pqr30... --ff CHARMM --ffout CHARMM --keep-chain --titration-state-
↪method=propka --with-ph=7.00 ...tmp_pdb2pqr.pdb ...4n1m.pqr
0
>>> prot_coor = Coor(os.path.join(TEST_OUT, '4n1m.pqr'))
Succeed to read file ...4n1m.pqr ,  2549 atoms found
>>> HSD_index = prot_coor.get_index_selection({'res_name': ['HSD'],'name': ['CA
↪']})
>>> print(len(HSD_index))
4
>>> HSE_index = prot_coor.get_index_selection({'res_name': ['HSE'],'name': ['CA
↪']})
>>> print(len(HSE_index))
0
>>> HSP_index = prot_coor.get_index_selection({'res_name': ['HSP'],'name': ['CA
↪']})
>>> print(len(HSP_index))
1
>>> prot_coor.correct_his_name()
<...Coor object at 0x...
>>> HIS_index = prot_coor.get_index_selection({'res_name': ['HIS'],'name': ['CA
↪']})
>>> print(len(HIS_index))
0
```

Note: This function seems useless. Since last version of pdb2pqr residue name seems correct.

correct_ion_octa(*ion_name_list*, *dist=0.9*)

For specified ion, create an octahedral dummy model described by a set of 6 cationic dummy atoms connected around a central metal atom. From Duarte et al. J Phys Chem B 2014.

Parameters

ion_name (*str*) – name of metal present in .pdb to transform in octahedral dummy model

Example

```
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1jd4.pdb'))
Succeed to read file ...1jd4.pdb ,  1586 atoms found
>>> ion_index = prot_coor.get_index_selection({'res_name' : ['ZN']})
>>> print(len(ion_index))
2
>>> prot_coor.correct_ion_octa(['ZN'])
<...Coor object at 0x...
```

(continues on next page)

(continued from previous page)

```
>>> ion_index = prot_coor.get_index_selection({'res_name' : ['ZN']})
>>> print(len(ion_index))
14
```

correct_protonated_res()

Correct protonated residues names, to avoid a mix of residue names from pdb2pqr, like GLUP and GLU or ASPP and ASP. To do after pdb2pqr, in order that protonation is recognize by pdb2gmx.

Example

```
>>> try:
...   print("Start import")
...   from . import pdb2pqr
... except ImportError:
...   import pdb2pqr
Start import...
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> # Compute protonation with pdb2pqr:
>>> pdb2pqr.compute_pdb2pqr(os.path.join(TEST_PATH, '4n1m.pdb'),os.path.
↪join(TEST_OUT, '4n1m.pqr'), ph=3.0)
Succeed to read file ...4n1m.pdb ,  2530 atoms found
Succeed to save file ...tmp_pdb2pqr.pdb
pdb2pqr30... --ff CHARMM --ffout CHARMM --keep-chain --titration-state-
↪method=propka --with-ph=3.00 ...tmp_pdb2pqr.pdb ...4n1m.pqr
0
>>> prot_coor = Coor(os.path.join(TEST_OUT, '4n1m.pqr'))
Succeed to read file ...4n1m.pqr ,  2564 atoms found
>>> HSD_index = prot_coor.get_index_selection({'res_name': ['HSD'],'name': ['CA
↪']})
>>> print(len(HSD_index))
0
>>> HSE_index = prot_coor.get_index_selection({'res_name': ['HSE'],'name': ['CA
↪']})
>>> print(len(HSE_index))
0
>>> HSP_index = prot_coor.get_index_selection({'res_name': ['HSP'],'name': ['CA
↪']})
>>> print(len(HSP_index))
5
>>> GLUP_index = prot_coor.get_index_selection({'res_name' : ['GLUP'],'name': [
↪'CA']})
>>> print(len(GLUP_index))
0
>>> ASPP_index = prot_coor.get_index_selection({'res_name': ['ASPP'],'name': [
↪'CA']})
>>> print(len(ASPP_index))
0
>>> prot_coor.correct_protonated_res()
<...Coor object at 0x...
>>> GLUP_index = prot_coor.get_index_selection({'res_name': ['GLUP'],'name': [
↪'CA']})
>>> print(len(GLUP_index))
```

(continues on next page)

(continued from previous page)

```
9
>>> GLU_index = prot_coor.get_index_selection({'res_name': ['GLU'],'name': ['CA
↪']})
>>> print(len(GLU_index))
3
>>> ASPP_index = prot_coor.get_index_selection({'res_name': ['ASPP'],'name': [
↪'CA']})
>>> print(len(ASPP_index))
2
>>> ASP_index = prot_coor.get_index_selection({'res_name': ['ASP'],'name': ['CA
↪']})
>>> print(len(ASP_index))
5
```

correct_water_name()

Correct the water resname from pdb2pqr

Example

```
>>> try:
...   print("Start import")
...   from . import pdb2pqr
... except ImportError:
...   import pdb2pqr
Start import...
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1dpx.pdb'))
Succeed to read file ...1dpx.pdb ,  1192 atoms found
>>> prot_coor.water_to_ATOM()
<...Coor object at 0x...
>>> prot_coor.write_pdb(os.path.join(TEST_OUT, '1dpx_water.pdb'))
Succeed to save file ...1dpx_water.pdb
>>> # Compute protonation with pdb2pqr:
>>> pdb2pqr.compute_pdb2pqr(
... os.path.join(TEST_OUT, '1dpx_water.pdb'),
... os.path.join(TEST_OUT, '1dpx_water.pqr'))
Succeed to read file ...1dpx_water.pdb ,  1192 atoms found
Succeed to save file ...tmp_pdb2pqr.pdb
pdb2pqr30... --ff CHARMM --ffout CHARMM --keep-chain --titration-state-
↪method=propka --with-ph=7.00...tmp_pdb2pqr.pdb ...1dpx_water.pqr
0
>>> prot_coor = Coor(os.path.join(
... TEST_OUT, '1dpx_water.pqr'))
Succeed to read file ...1dpx_water.pqr ,  2492 atoms found
>>> water_index = prot_coor.get_index_selection(
... {'res_name': ['TP3M'], 'name': ['OH2']})
>>> print(len(water_index))
177
```

cryst_convert(*format_out='pdb'*)

PDB format: https://www.wwpdb.org/documentation/file-format-content/format33/sect8.html

Gro to pdb: https://mailman-1.sys.kth.se/pipermail/gromacs.org_gmx-users/2008-May/033944.html

https://en.wikipedia.org/wiki/Fractional_coordinates

```
>>> prot_coor = Coor()
>>> prot_coor.read_file(os.path.join(TEST_PATH, '1y0m.gro'))
Succeed to read file ...1y0m.gro ,  648 atoms found
>>> prot_coor.cryst_convert(format_out='pdb')
'CRYST1   28.748   30.978   29.753  90.00  92.12  90.00 P 1           1\n'
>>> prot_coor = Coor()
>>> prot_coor.read_file(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> prot_coor.cryst_convert(format_out='gro')
'   2.87480   3.09780   2.97326   0.00000   0.00000   0.00000   0.00000  -0.
↪11006   0.00000\n'
```

del_atom_index(*index_list*)

Delete atoms of a coor object defined by their `index`.

Parameters
index_list (*list*) – list of atom index to delete

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> prot_coor.get_aa_seq()
{'A': 'TFKSAVKALFDYKAQREDELTFTKSAIIQNVEKQDGGWWRGDYGGKKQLWFPSNYVEEMIN'}
>>> res_810_852 = prot_coor.get_index_selection({'res_num': range(810,852)})
>>> prot_coor.del_atom_index(index_list=res_810_852)
<...Coor object at ...>
>>> prot_coor.get_aa_seq()
{'A': 'TFKSAVKALFDYKAQREDE'}
```

dist_under_index(*atom_sel_2*, *cutoff=10.0*)

Check is distance between atoms of self.coor is under cutoff with atoms of group 1. Then return list of index of atoms of self.coor under ctuoff ditance.

Parameters

- **atom_sel_1** (*dict*) – atom dictionnary
- **atom_sel_2** (*dict*) – atom dictionnary
- **cutoff** (*float, default=10.0*) – distance cutoff

Returns
array of index

Return type
np.array

get_PDB(*pdb_ID*, *out_file=None*, *check_file_out=True*)

Get a pdb file from the PDB using its ID and return a Coor object.

Parameters

- **pdb_ID** (*str*) – Protein Data Bank structure ID
- **out_file** (*str, optional, default=None*) – path of the pdb file to save

- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.

Example

```
>>> show_log()
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> prot_coor = Coor()
>>> prot_coor.get_PDB('3EAM', os.path.join(TEST_OUT, '3eam.pdb'))
Succeed to read file ...3eam.pdb ,  13505 atoms found
```

get_aa_DL_seq(*gap_in_seq=True*)

Get the amino acid sequence from a coor object. if amino acid is in D form it will be in lower case.

L or D form is determined using CA-N-C-CB angle Angle should take values around +34° and 34° for L- and D-amino acid residues.

Reference: https://onlinelibrary.wiley.com/doi/full/10.1002/prot.10320

Returns
dictionnary of sequence indexed by the chain ID

Return type
dict

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> prot_coor.get_aa_DL_seq()
{'A': 'TFKSAVKALFDYKAQREDELTFTKSAIIQNVEKQDGGWWRGDYGGKKQLWFPSNYVEEMIN'}
>>> prot_coor = Coor(os.path.join(TEST_PATH, '6be9_frame_0.pdb'))
Succeed to read file ...6be9_frame_0.pdb ,  104 atoms found
>>> prot_coor.get_aa_DL_seq()
Residue K2 is in D form
Residue N6 is in D form
Residue P7 is in D form
{'A': 'TkNDTnp'}
```

Warning: If atom chains are not arranged sequentialy (A,A,A,B,B,A,A,A ...), the first atom seq will be overwritten by the last one.

get_aa_num()

Get the amino acid number of a coor object.

Returns
Number of residues

Return type
int

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
```

(continues on next page)

(continued from previous page)

```
>>> prot_coor.get_aa_num()
61
```

Note: Only count Ca atoms, this may not be the best choice ?

get_aa_seq(*gap_in_seq=True*)

Get the amino acid sequence from a coor object.

Returns
dictionnary of sequence indexed by the chain ID

Return type
dict

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> prot_coor.get_aa_seq()
{'A': 'TFKSAVKALFDYKAQREDELTFTKSAIIQNVEKQDGGWWRGDYGGKKQLWFPSNYVEEMIN'}
```

Warning: If atom chains are not arranged sequentialy (A,A,A,B,B,A,A,A ...), the first atom seq will be overwritten by the last one.

get_array(*field='xyz'*, *index_list=None*)

Convert atom dict as a numpy array.

Parameters

- **field** (*str (Default='xyz')*) – field to extract
- **index_list** (*list (Default=None)*) – list of index to extract

Returns
coordinates array

Return type
np.array

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> coor_array = prot_coor.get_array()
>>> print(coor_array[1:4])
[[-0.971  9.213 12.734]
 [-0.185  7.901 12.631]
 [ 0.456  7.524 13.61 ]]
>>> coor_array = prot_coor.get_array(field='name')
>>> print(coor_array[1:4])
['CA' 'C' 'O']
```

get_attribute_selection(*selec_dict={}*, *attribute='uniq_resid'*, *index_list=None*)

Select atom of a coor object based on the change_dict dictionnary. Return the list of unique attribute of the selected atoms.

Parameters
selec_dict (*dict*) – select ditionnay eg. {"chain": ["A","G"]}

Returns
list of atom index

Return type
list of int

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> prot_coor.get_attribute_selection({'res_num': [826,827]},attribute='uniq_
↪resid')
[35, 36]
```

get_box_dim(*selec_dict={}*)

Compute the x, y, z dimension of a selection

Parameters
selec_dict (*dict, default={}*) – selection dictionnary

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> box_1y0m = prot_coor.get_box_dim()
>>> print("x:{:.2f} y:{:.2f} z:{:.2f}".format(*box_1y0m))
x:39.14 y:29.36 z:31.19
```

Warning: Atom name must start with its type letter (H, C, N, O, P, S).

get_center_residue(*selec_dict={'res_name': ['GLY', 'HIS', 'HSP', 'HSE', 'HSD', 'HIP', 'HIE', 'HID', 'ARG', 'LYS', 'ASP', 'ASPP', 'GLU', 'GLUP', 'SER', 'THR', 'ASN', 'GLN', 'CYS', 'SEC', 'PRO', 'ALA', 'ILE', 'PHE', 'TYR', 'TRP', 'VAL', 'LEU', 'MET', 'DA5', 'DA3', 'DAN', 'DA', 'DT5', 'DT3', 'DTN', 'DT', 'DC5', 'DC3', 'DCN', 'DC', 'DG5', 'DG3', 'DGN', 'DG', 'RA5', 'RA3', 'RAN', 'RA', 'RU5', 'RU3', 'RUN', 'RU', 'RC5', 'RC3', 'RCN', 'RC', 'RG5', 'RG3', 'RGN', 'RG']}*, *field='res_num'*)

Find the closet residue to protein center of mass. Return the residue index.

Parameters
selec_res_list (*list, default={'res_name': PROTEIN_RES}*) – selection residue

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> center_res = prot_coor.get_center_residue()
Minimal distance to protein COM is 3.00 Å with residue "res_num": 841
>>> print("Center residue is {:d}".format(center_res))
Center residue is 841
```

get_common_atom(*atom_sel_2*, *chain_1=['A']*, *chain_2=['A']*, *back_names=['C', 'N', 'O', 'CA']*)

Get atom selection in common for two atom_dict based on sequence alignement.

get_gro_structure_string()

Return a coor object as a pdb string.

Example

```
>>> prot_coor = Coor()
>>> prot_coor.read_pdb(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> gro_str = prot_coor.get_gro_structure_string()
>>> print('Number of caracters: {}'.format(len(gro_str)))
Number of caracters: 29283
```

get_index_dist_between(*atom_sel_2*, *cutoff_min=0*, *cutoff_max=10*)

Check is distance between atoms of self.atom_dict is under cutoff with the atoms of group 1. Then return list of index of atoms of self.coor under cutoff ditance.

Parameters

- **atom_sel_1** (*dict*) – atom dictionnary
- **atom_sel_2** (*dict*) – atom dictionnary
- **cutoff_min** (*float, default=0.0*) – maximum distance cutoff
- **cutoff_max** (*float, default=10.0*) – minimum distance cutoff

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> res_810 = prot_coor.select_part_dict({'res_num': [810]})
>>> close_r810 = prot_coor.get_index_dist_between(
... res_810, cutoff_min=3, cutoff_max=5)
>>> print(len(close_r810))
65
```

get_index_selection(*selec_dict*)

Select atom of a coor object based on the change_dict dictionnary. Return the list of index of selected atoms.

Parameters

selec_dict (*dict*) – select ditionnay eg. {“chain”: [“A”,”G”]}

Returns

list of atom index

Return type

list of int

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> prot_coor.get_index_selection({'res_num': [826,827]})
[297, 298, 299, 300, 301, 302, 303, 304]
```

get_mass_array()

Extract mass of each *atom_dict* and return it as an numpy array Avoid using atoms with 2 letters atom name like NA Cl ...

Returns
mass array

Return type
np.array

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> mass_1y0m = prot_coor.get_mass_array()
>>> print("Mass of 10 first atoms: {}".format(mass_1y0m[:10]))
Mass of 10 first atoms: [7 6 6 8 6 8 6 7 6 6]
>>> prot_coor_ca = prot_coor.select_part_dict({'name': ['CA']})
>>> mass_1y0m_ca = prot_coor_ca.get_mass_array()
>>> print("Mass of 10 first atoms: {}".format(mass_1y0m_ca[:10]))
Mass of 10 first atoms: [6 6 6 6 6 6 6 6 6 6]
```

get_max_size()

Get maximum size of a molecule.

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> print('Maximum size without alignment is {:.2f} Å'.format(
... np.ceil(prot_coor.get_box_dim()).max()))
Maximum size without alignment is 40.00 Å
>>> max_size = prot_coor.get_max_size()
Do a rotation of 66.43°
>>> print('Maximum size is {:.2f} Å'.format(max_size))
Maximum size is 43.00 Å
```

get_pqr_structure_string()

Return a coor object as a pqr string.

Example

```
>>> prot_coor = Coor()
>>> prot_coor.read_pdb(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> pqr_str = prot_coor.get_pqr_structure_string()
>>> print('Number of caracters: {}'.format(len(pqr_str)))
Number of caracters: 46728
```

get_structure_string()

Return a coor object as a pdb string.

Example

```
>>> prot_coor = Coor()
>>> prot_coor.read_pdb(os.path.join(TEST_PATH, '1y0m.pdb'))
```

(continues on next page)

(continued from previous page)

```
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> pdb_str = prot_coor.get_structure_string()
>>> print('Number of caracters: {}'.format(len(pdb_str)))
Number of caracters: 51264
```

insert_mol(*pdb_out*, *out_folder*, *mol_chain*, *mol_num*, *check_file_out=True*, *prot_atom_name=['CA']*, *sol_res_name=['SOL']*, *sol_atom_name=['OW']*)

Insert molecules defined by chain ID `mol_chain` in a water solvant. Check which water molecules are within `cutoff_prot_off=12.0` X and `cutoff_prot_in=15.0` X of protein and peptide C alpha atoms. Move the molecules to be inserted at the position of water molecules. Then delete all water molecules within `cutoff_water_clash=1.2` X of the inserted molecule atoms.

Parameters

- **pdb_out** (*str*) – name of output pdb file
- **out_folder** (*str*) – path of the ouput directory
- **mol_chain** (*str*) – chain ID of the molecule to be inserted,
- **mol_num** (*int*) – Number of molecule to be inserted,
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.

Warning: self.atom_dict file must contain alredy a concatenated system with a ligand (chain: `mol_chain`) and a solvated system. Molecules to insert should have different residue number or at least non consecutive.

static kabsch(*coor_1*, *coor_2*)

Source: https://github.com/charnley/rmsd/blob/master/rmsd/ calculate_rmsd.py Using the Kabsch algorithm with two sets of paired point P and Q, centered around the centroid. Each vector set is represented as an NxD matrix, where D is the the dimension of the space.

The algorithm works in three steps: - a centroid translation of P and Q (assumed done before this function call) - the computation of a covariance matrix C - computation of the optimal rotation matrix U For more info see http://en.wikipedia.org/wiki/Kabsch_algorithm

Parameters

- **coor_1** (*np.array*) – coordinates array of size (N, D), where N is points and D is dimension.
- **coor_2** (*np.array*) – coordinates array of size (N, D), where N is points and D is dimension.

Returns
rotation matrix

Return type
np.array of size (D, D)

static makeQ(*r1*, *r2*, *r3*, *r4=0*)

Source: https://github.com/charnley/rmsd/blob/master/rmsd/ calculate_rmsd.py matrix involved in quaternion rotation

static makeW(*r1*, *r2*, *r3*, *r4=0*)

Source: https://github.com/charnley/rmsd/blob/master/rmsd/ calculate_rmsd.py matrix involved in quaternion rotation

make_peptide(*sequence*, *pdb_out*, *check_file_out=True*)

Create a linear peptide structure.

Parameters

- **sequence** (*str*) – peptide sequence
- **pdb_out** (*str*) – name of output pdb file
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.

moment_inertia()

Tensor moment of inertia relative to center of mass.

Taken from: https://github.com/MDAnalysis/mdanalysis/blob/develop/package/MDAnalysis/core/topologyattrs.py

Returns
tensor matrix 3*3

Return type
np.array

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> tensor = prot_coor.moment_inertia()
>>> print(tensor)
[[343695.37614973  31289.34303697  40416.71166176]
 [ 31289.34303697 478561.60329129  -9443.32698326]
 [ 40416.71166176  -9443.32698326 413840.08602987]]
```

parse_gro_lines(*gro_lines*)

Parse a gro file and return atom informations as a dictionnary indexed on the atom num.

Parameters
gro_in (*list of str*) – lines to parse

Example

```
>>> prot_coor = Coor()
>>> f = open(os.path.join(TEST_PATH, '1y0m.gro'))
>>> lines = f.readlines()
>>> prot_coor.parse_gro_lines(lines)
>>> prot_coor.num
648
```

parse_pdb_lines(*pdb_lines*, *pqr_format=False*)

Parse the pdb lines and return atom informations as a dictionnary indexed on the atom num. The fonction can also read pqr files if specified with `pqr_format = True`, it will only change the column format of beta and occ factors.

Parameters
pdb_lines (*list of str*) – lines to parse

Example

```
>>> prot_coor = Coor()
>>> f = open(os.path.join(TEST_PATH, '1y0m.pdb'))
>>> lines = f.readlines()
>>> prot_coor.parse_pdb_lines(lines)
>>> prot_coor.num
648
```

plot_pseudo_3D(*c_field=None*, *cmap='gist_rainbow'*, *line_w=1.5*, *chainbreak=5.0*, *sel={'name': ['CA']}*, *fig_size=(7, 7)*)

Plot alpha Carbon trace of protein in pseudo 3D.

Inspired from Colab fold plot_pseudo_3D() function from sokrypton https://github.com/sokrypton/ColabFold

Parameters

- **c_field** (*str (Default=None or 'index')*) – field used to color figure
- **cmap** (*str (Default="gist_rainbow")*) – Color map
- **line_w** (*flt (Default=1.5)*) – Line width
- **chainbreak** (*flt (Default=5.0)*) – Distance for chain break (in Å)
- **sel** (*dict (Default={'name': ['CA']})*) – selection to plot
- **fig_size** (*tupple (Default=(7, 7))*) – Figure size

Returns
matplotlib ax

Return type
ax

Example

principal_axis()

Calculate the principal axes from the moment of inertia.

Taken from: https://github.com/MDAnalysis/mdanalysis/blob/develop/package/MDAnalysis/core/topologyattrs.py

Returns
3 principal axis

Return type
list of np.array

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> princ_axis = prot_coor.principal_axis()
>>> print(princ_axis)
[[-0.21318784 -0.97697417  0.00850932]
 [ 0.39228202 -0.07761763  0.91656441]
 [-0.89479929  0.19873844  0.39979654]]
```

static quaternion_rotate(*X*, *Y*)

Source: https://github.com/charnley/rmsd/blob/master/rmsd/ calculate_rmsd.py Calculate the rotation

Parameters

- **coor_1** (*np.array*) – coordinates array of size (N, D), where N is points and D is dimension.
- **coor_2** (*np.array*) – coordinates array of size (N, D), where N is points and D is dimension.

Returns
rotation matrix

Return type
np.array of size (D, D)

static quaternion_transform(*r*)

Source: https://github.com/charnley/rmsd/blob/master/rmsd/ calculate_rmsd.py Get optimal rotation note: translation will be zero when the centroids of each molecule are the same.

read_file(*file_in*)

Read a pdb file and return atom informations as a dictionnary indexed on the atom num. The fonction can also read pqr files if specified with `pqr_format = True`, it will only change the column format of beta and occ factors.

Parameters

- **pdb_in** (*str*) – path of the pdb file to read
- **pqr_format** (*bool, default=False*) – Flag for .pqr file format reading.

Example

```
>>> prot_coor = Coor()
>>> prot_coor.read_file(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> prot_coor.read_file(os.path.join(TEST_PATH, '1y0m.gro'))
Succeed to read file ...1y0m.gro ,  648 atoms found
```

read_pdb(*pdb_in*, *pqr_format=False*)

Read a pdb file and return atom informations as a dictionnary indexed on the atom num. The fonction can also read pqr files if specified with `pqr_format = True`, it will only change the column format of beta and occ factors.

Parameters

- **pdb_in** (*str*) – path of the pdb file to read
- **pqr_format** (*bool, default=False*) – Flag for .pqr file format reading.

Example

```
>>> prot_coor = Coor()
>>> prot_coor.read_pdb(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
```

remove_alter_position()

Remove alternative position.

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '4n1m.pdb'))
Succeed to read file ...4n1m.pdb ,  2530 atoms found
>>> print('Atom num = {}'.format(prot_coor.num))
Atom num = 2530
>>> prot_coor.remove_alter_position()
>>> print('Atom num = {}'.format(prot_coor.num))
Atom num = 2475
```

rotation_angle(*tau_x*, *tau_y*, *tau_z*)

Compute coordinates of a system after a rotation on x, y and z axis.

Parameters

- **tau_x** (*float*) – angle of rotation (degrees) on the x axis
- **tau_y** (*float*) – angle of rotation (degrees) on the y axis
- **tau_z** (*float*) – angle of rotation (degrees) on the z axis

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> com_1y0m = prot_coor.center_of_mass()
>>> print("x:{:.2f} y:{:.2f} z:{:.2f}".format(*com_1y0m))
x:16.01 y:0.45 z:8.57
>>> prot_coor.rotation_angle(90, 90, 90)
>>> com_1y0m = prot_coor.center_of_mass()
>>> print("x:{:.2f} y:{:.2f} z:{:.2f}".format(*com_1y0m))
x:9.98 y:-4.03 z:-14.63
```

select_from_index(*index_list*)

Select atom of a coor object from an atom index list. Return a new coor object.

Parameters
index_list (*list*) – list of index eg. [0, 4]

Returns
a new coor object

Return type
coor

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> index_list = prot_coor.get_index_selection({'res_num': [826,827]})
>>> index_list
[297, 298, 299, 300, 301, 302, 303, 304]
>>> prot_sel = prot_coor.select_from_index(index_list)
>>> len(prot_sel.atom_dict) == len(index_list)
True
```

select_part_dict(*selec_dict*)

Select atom of a coor object defined, the selection is based on the change_dict dictionnary. Return a new coor object.

Parameters
selec_dict (*dict*) – change ditionnay eg. {"chain" : ["A","G"]}

Returns
a new coor object

Return type
coor

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> prot_coor.get_aa_num()
61
>>> prot_20_coor = prot_coor.select_part_dict(selec_dict={'res_num':␣
↪list(range(791,800))})
>>> prot_20_coor.get_aa_seq()
{'A': 'TFKSAVKAL'}
>>> prot_20_coor.get_aa_num()
9
>>> prot_N_atom = prot_coor.select_part_dict(selec_dict={'name': ['ZN']})
>>> # WARNING using selec_dict = {'name': 'ZN'} will
>>> # give you 61 residues !!
>>> print(prot_N_atom.num)
0
>>> # Select only protein atoms
>>> print(prot_coor.num)
648
>>> prot_only = prot_coor.select_part_dict(selec_dict={'res_name': PROTEIN_RES})
>>> print(prot_only.num)
526
```

translate(*vector*)

Translate all atoms of a coor object by a given `vector`

Parameters
vector (*list*) – 3d translation vector

Example

```
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> com_1y0m = prot_coor.center_of_mass()
>>> print("x:{:.2f} y:{:.2f} z:{:.2f}".format(*com_1y0m))
x:16.01 y:0.45 z:8.57
>>> prot_coor.translate(-com_1y0m)
>>> com_1y0m = prot_coor.center_of_mass()
>>> print("x:{:.2f} y:{:.2f} z:{:.2f}".format(*com_1y0m))
x:-0.00 y:0.00 z:0.00
```

property view

Return a *nglview* object to be view in a jupyter notebook.

Example:

```
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> prot_coor = Coor()
>>> prot_coor.get_PDB('3EAM', os.path.join(TEST_OUT, '3eam.pdb'))
Succeed to read file ...3eam.pdb ,  13505 atoms found
>>> view = prot_coor.view
>>> view
```

water_to_ATOM()

Change *HETATM* field of water to *ATOM*, as pdb2pqr only use ATOM field.

Example

```
>>> try:
...   print("Start import")
...   from . import pdb2pqr
... except ImportError:
...   import pdb2pqr
Start import...
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1dpx.pdb'))
Succeed to read file ...1dpx.pdb ,  1192 atoms found
>>> hetatm_index = prot_coor.get_index_selection({'field': ['HETATM']})
>>> print(len(hetatm_index))
179
>>> prot_coor.water_to_ATOM()
<...Coor object at 0x...
>>> hetatm_index = prot_coor.get_index_selection({'field': ['HETATM']})
>>> print(len(hetatm_index))
2
>>> water_index = prot_coor.get_index_selection({'res_name': ['HOH']})
>>> print(len(water_index))
177
```

write_pdb(*pdb_out*, *check_file_out=True*)

Write a pdb file.

Parameters

- **pdb_out** (*str*) – path of the pdb file to write
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.

Example

```
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> prot_coor.write_pdb(os.path.join(TEST_OUT, 'tmp.pdb'))
Succeed to save file ...tmp.pdb
```

write_pqr(*pqr_out*, *check_file_out=True*)

Write a pdb file.

Parameters

- **pdb_out** (*str*) – path of the pdb file to write

- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.

Example

```
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> prot_coor = Coor(os.path.join(TEST_PATH, '1y0m.pdb'))
Succeed to read file ...1y0m.pdb ,  648 atoms found
>>> prot_coor.write_pdb(os.path.join(TEST_OUT, 'tmp.pdb'))
Succeed to save file ...tmp.pdb
```

CHAPTER

NINE

MULTI COOR CLASS

class pdb_manip_py.pdb_manip.**Multi_Coor**(*pdb_in=None*, *pqr_format=False*)

Topologie base on coordinates like pdb or gro.

The coor object containt a a list of dictionnary of atoms indexed on the atom num and the crystal packing info.

Parameters

- **coor_list** – list of dictionnary of atom
- **crystal_pack** (*str*) – crystal packing

Note: Files necessary for testing : * ../test/input/1y0m.pdb, ../test/input/1rxz.pdb and ../test/input/4n1m.pdb. To do the unitary test, execute pdb_mani.py (-v for verbose mode)

---

compute_rmsd_to(*atom_sel_2*, *selec_dict={'name': ['CA']}*)

Compute RMSD between two atom_dict Then return the RMSD value.

Parameters

- **atom_sel_1** (*dict*) – atom dictionnary
- **atom_sel_2** (*dict*) – atom dictionnary

Returns
distance

Return type
float

Example

```
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> VIP_coor = Multi_Coor(os.path.join(TEST_PATH, '2rri.pdb'))
Read 20 Model(s)
Succeed to read file ...2rri.pdb, 479 atoms found
>>> aa_seq = VIP_coor.coor_list[0].get_aa_seq()['A']
>>> print(aa_seq)
HSDAVFTDNYTRLRKQMAVKKYLNSILNG
>>> linear_pep = Coor()
>>> pep_out = os.path.join(TEST_OUT, 'tmp_pep.pdb')
>>> pdb_pep = linear_pep.make_peptide(aa_seq, pep_out)
-Make peptide: HSDAVFTDNYTRLRKQMAVKKYLNSILNG
residue name:X
residue name:H
```

(continues on next page)

(continued from previous page)

```
residue name:S
residue name:D
residue name:A
residue name:V
residue name:F
residue name:T
residue name:D
residue name:N
residue name:Y
residue name:T
residue name:R
residue name:L
residue name:R
residue name:K
residue name:Q
residue name:M
residue name:A
residue name:V
residue name:K
residue name:K
residue name:Y
residue name:L
residue name:N
residue name:S
residue name:I
residue name:L
residue name:N
residue name:G
Succeed to save file ...tmp_pep.pdb
>>> linear_pep.read_pdb(pdb_pep)
Succeed to read file ...tmp_pep.pdb ,  240 atoms found
>>> rmsd_list = VIP_coor.compute_rmsd_to(linear_pep)
>>> rmsd_str = ['{:.2f}'.format(i) for i in rmsd_list]
>>> rmsd_str
['58.57', '58.40', '58.74', '58.35', '58.60', '58.53', '58.49', '58.40', '58.45
↪', '58.27', '58.52', '58.34', '58.57', '58.33', '58.34', '58.63', '58.61',
↪'58.40', '58.55', '58.32']
```

read_pdb(*pdb_in*, *pqr_format=False*)

Read a pdb file and return atom informations as a dictionnary indexed on the atom num. The fonction can also read pqr files if specified with `pqr_format = True`, it will only change the column format of beta and occ factors.

Parameters

- **pdb_in** (*str*) – path of the pdb file to read
- **pqr_format** (*bool, default=False*) – Flag for .pqr file format reading.

Example

```
>>> VIP_coor = Multi_Coor(os.path.join(TEST_PATH, '2rri.pdb'))
Read 20 Model(s)
Succeed to read file ...2rri.pdb, 479 atoms found
```

write_pdb(*pdb_out*, *check_file_out=True*)

Write a pdb file.

Parameters

- **pdb_out** (*str*) – path of the pdb file to write
- **check_file_out** – flag to check if output file already exists

Example

```
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> VIP_coor = Multi_Coor(os.path.join(TEST_PATH, '2rri.pdb'))
Read 20 Model(s)
Succeed to read file ...2rri.pdb, 479 atoms found
>>> VIP_coor.write_pdb(os.path.join(TEST_OUT, 'tmp.pdb'))
Succeed to save file ...tmp.pdb
```

CHAPTER

TEN

PDB2PQR MODULE

pdb_manip_py.pdb2pqr.**compute_pdb2pqr**(*pdb_in*, *pdb_out*, *ff='CHARMM'*, *method='propka'*, *ph=7.0*, *check_file_out=True*)

Use pdb2pqr to define protonation state of each residue of a protein.

Parameters

- **pdb_in** (*str*) – path of input pdb file
- **pdb_out** (*str*) – path of output pdb file
- **ff** (*str, optional, default="CHARMM"*) – forcefield nomenclature for atom names
- **method** (*str, optional, default="propka"*) – Method used to calculate ph values (propka or pdb2pka).
- **ph** (*float, optional, default=7.0*) – pH value to define AA residue
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.

Example

```
>>> pdb_manip.show_log()
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> # Compute protonation with pdb2pqr:
>>> compute_pdb2pqr(os.path.join(TEST_PATH,'4n1m.pdb'),
... os.path.join(TEST_OUT, '4n1m.pqr'))
Succeed to read file ...4n1m.pdb ,  2530 atoms found
Succeed to save file ...tmp_pdb2pqr.pdb
pdb2pqr30... --ff CHARMM --ffout CHARMM --keep-chain --titration-state-
↪method=propka --with-ph=7.00 ...tmp_pdb2pqr.pdb ...4n1m.pqr
0
>>> prot_coor = pdb_manip.Coor()
>>> prot_coor.read_pdb(os.path.join(TEST_OUT, '4n1m.pqr'), pqr_format = True)
Succeed to read file ...4n1m.pqr ,  2549 atoms found
>>> HSD_index = prot_coor.get_index_selection({'res_name' : ['HSD'],
... 'name':['CA']})
>>> print(len(HSD_index))
4
>>> HSE_index = prot_coor.get_index_selection({'res_name' : ['HSE'],
... 'name':['CA']})
>>> print(len(HSE_index))
0
>>> HSP_index = prot_coor.get_index_selection({'res_name' : ['HSP'],
```

(continues on next page)

(continued from previous page)

```
... 'name':['CA']})
>>> print(len(HSP_index))
1
```

Note: Idealy I would need a pdb file with 3 different histidine protonation. I couldn't find one.

CHAPTER

ELEVEN

CREDITS

11.1 Development Lead

- Samuel Murail, Université de Paris <samuel.murail@u-paris.fr>

11.2 Contributors

- Pierre Tuffery, INSERM

We are open to any contribution.

CHAPTER

TWELVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

I

K

M

P

Q

R

S

T

V

W