The pdb2sql Python Package: Parsing, Manipulation and Analysis of PDB Files Using SQL Queries

Nicolas Renaud[1] **and Cunliang Geng**[1]

1 Netherlands eScience Center, Science Park 140 1098 XG Amsterdam, the Netherlands

DOI: 10.21105/joss.02077

Software

- Review
- Repository
- Archive

Editor: Lorena Pantano

Reviewers:

- @i-mtz
- @JoaoRodrigues

Submitted: 27 January 2020
Published: 06 February 2020

Summary

The analysis of biomolecular structures is a crucial task for a wide range of applications ranging from drug design to protein engineering. The Protein Data Bank (PDB) file format (Burley et al., 2019) is the most popular format to describe biomolecular structures such as proteins and nucleic acids. In this text-based format each line represents a given atom and entails its main properties such as atom name and identifier, residue name and identifier, chain identifier, coordinates, etc. Several solutions have been developed to parse PDB files in dedicated objects that facilitate the analysis and manipulation of biomolecular structures. This is for example the case of the `BioPython` parser (Cock et al., 2009,@biopdb) that loads PDB files in a nested dictionary whose structure mimics the hierarchical nature of the biomolecular structure. Selecting a given sub-part of the biomolecule can then be done by going through the dictionary and selecting the required atoms. Other packages, such as `ProDy` (Bakan, Meireles, & Bahar, 2011), `BioJava` (Lafita, 2019), `MMTK` (Hinsen, 2000) and `MDAnalysis` (Gowers et al., 2016) to cite a few, also offer solutions to parse PDB files. However these parsers are embedded in large code base that are sometimes difficult to integrate in new applications and are often geared toward the analysis of molecular dynamics simulations. Light-weight applications such as `pdb-tools` (Rodrigues, Teixeira, Trellet, & Bonvin, 2018) lack the capabilities to manipulate coordinates.

We present here the Python package `pdb2sql`, which loads individual PDB files in a relational database. Among different solutions the Structured Query Language (SQL) is a very popular solution to query a given database. However SQL queries are complex and domain scientists such as bioinformaticians are usually not familiar with them. This represents an important barrier for the adoption of SQL technology in bioinformatics. `pdb2sql` exposes complex SQL queries through simple Python methods that are intuitive for end users. Native `SQLite3` database can be used or `sqlalchemy` (Bayer, 2012) can be used to obtain an object oriented approach. As such, our package leverages the power of SQL queries and remove the barrier that SQL complexity represents. In addition, several advanced modules have also been built, for example to rotate or translate biomolecular structures, to characterize interface contacts, and to measure structure similarity between two protein complexes. Additional modules can easily be developed following the same scheme. As a consequence, `pdb2sql` is a light-weight and versatile PDB tool that is easy to extend and to integrate with new applications.

Capabilities of `pdb2sql`

`pdb2sql` allows to query, manipulate and process PDB files through a series of dedicated classes. We give an overview of these features and illustrate them with snippets of code. More examples can be found in the documentation (https://pdb2sql.readthedocs.io).

Extracting data from PDB files

pdb2sql allows to simply query the database using the get(attr, **kwargs) method. The attribute attr is here a list of or a single column name of the SQL database, see Table 1 for available attributes. The keyword argument kwargs can then be used to specify a sub-selection of atoms.

Table 1. Atom attributes and assoicated definitions in pdb2sql

Attribute	Definition
serial	Atom serial number
name	Atom name
altLoc	Alternate location indicator
resName	Residue name
chainID	Chain identifier
resSeq	Residue sequence number
iCode	Code for insertion of residues
x	Orthogonal coordinates for X in Angstroms
y	Orthogonal coordinates for Y in Angstroms
z	Orthogonal coordinates for Z in Angstroms
occ	Occupancy
temp	Temperature factor
element	Element symbol
model	Model serial number

Every attribute name can be used to select specific atoms and multiple conditions can be easily combined. For example, let's consider the following example :

```
from pdb2sql import pdb2sql
pdb = pdb2sql('1AK4.pdb')
atoms = pdb.get('x,y,z',
                name=['C','H'],
                resName=['VAL','LEU'],
                chainID='A')
```

This snippet extracts the coordinates of the carbon and hydrogen atoms that belong to all the valine and leucine residues of the chain labelled A in the PDB file. Atoms can also be excluded from the selection by appending the prefix no_ to the attribute name. This is the case in the following example :

```
from pdb2sql import pdb2sql
pdb = pdb2sql('1AK4.pdb')
atoms = pdb.get('name, resName',
                no_resName=['GLY', 'PHE'])
```

This snippet extracts the atom and residue names of all atoms except those belonging to the glycine and phenylalanine residues of the structure. Similar combinations of arguments can be designed to obtain complex selectiong rules that precisely select the desired atom properties.

Manipulating PDB files

The data contained in the SQL database can also be modified using the update(attr, vals, **kwargs) method. The attributes and keyword arguments are identical to those in the get method. The vals argument should contain a numpy array whose dimension should match the selection criteria. For example :

```
import numpy as np
from pdb2sql import pdb2sql

pdb = pdb2sql('1AK4.pdb')
xyz = pdb.get('x,y,z', chainID='A', resSeq=1)
xyz = np.array(xyz)
xyz -= np.mean(xyz)
pdb.update('x,y,z', xyz, chainID='A', resSeq=1)
```

This snippet first extracts the coordinates of atoms in the first residue of chain A, then translates this fragment to the origin and updates the coordinate values in the database. pdb2sql also provides a convinent class transform to easily tanslate or rotate strcutures. For example, to translate the first residue of the strcuture 5 Å along the Y-axis,

```
import numpy as np
from pdb2sql import pdb2sql
from pdb2sql import transform

pdb = pdb2sql('1AK4.pdb')
trans_vec = np.array([0,5,0])
transform.translation(pdb, trans_vec, resSeq=1, chainID='A')
```

One can also rotate a given selection around a given axis with the rotate_axis method :

```
angle = np.pi
axis = (1., 0., 0.)
transform.rot_axis(pdb, axis, angle, resSeq=1, chainID='A')
```

Identifying interface

The interface class is derived from the pdb2sql class and offers functionalities to identify contact atoms or residues between two different chains with a given contact distance. It is useful for extracting and analysing the interface of e.g. protein-protein complexes. The following example snippet returns all the atoms and all the residues of the interface of '1AK4.pdb' defined by a contact distance of 6 Å.

```
from pdb2sql import interface

pdb = interface('1AK4.pdb')
atoms = pdb.get_contact_atoms(cutoff=6.0)
res = pdb.get_contact_residues(cutoff=6.0)
```

Computing Structure Similarity

The StructureSimilarity class allows to compute similarity measures between two protein-protein complexes. Several popular measures used to classifiy qualities of protein complex structures in the CAPRI (Critical Assessment of PRedicted Interactions) challenges (Méndez, Leplae, Maria, & Wodak, 2003) have been implemented: interface rmsd, ligand rmsd, fraction of native contacts and DockQ(Basu & Wallner, 2016). The approach implemented to compute the interface rmsd and ligand rmsd is identical to the well-known package ProFit (Martin & Porter, 2009). All the methods required to superimpose structures have been implemented in the transform class and therefore relies on no external dependencies. The following snippet shows how to compute these measures:

```
from pdb2sql import StructureSimilarity

sim = StructureSimilarity(decoy = '1AK4_model.pdb',
                          ref = '1AK4_xray.pdb')

irmsd = sim.compute_irmsd_fast()
lrmsd = sim.compute_lrmsd_fast()
fnat = sim.compute_fnat_fast()
dockQ = sim.compute_DockQScore(fnat, lrmsd, irmsd)
```

Application

psb2sql has been used at the Netherlands eScience center for bioinformatics projects. This is for example the case of iScore (Geng et al., 2019) that uses graph kernels and support vector machines to rank protein-protein interface. We illustrate here the use of the pacakge by computing the interface rmsd and ligand rmsd of a series of structural models using the experimental structure as a reference. This is a common task for protein-protein docking where a large number of docked conformations are generated and have then to be compared to a ground truth to identify the best generated poses. This calculation is usually done using the ProFit software and we therefore compare our results with those obtained with ProFit. The code do compute the similarity measure for different decoys is simple:

```
from pdb2sql import StructureSimilarity

ref = '1AK4.pdb'
decoys = os.listdir('./decoys')
irmsd = {}

for d in decoys:
    sim = StructureSimilarity(d, ref)
    irmsd[d] = sim.compute_irmsd_fast(method='svd', izone='1AK4.izone')
```

Note that the method will compute the i-zone, i.e. the zone of the proteins that form the interface in a similar way than ProFit. This is done for the first calculations and the i-zone is then reused for the subsequent calculations. The comparison of our interface rmsd values to those given by ProFit are shown in Fig 1.

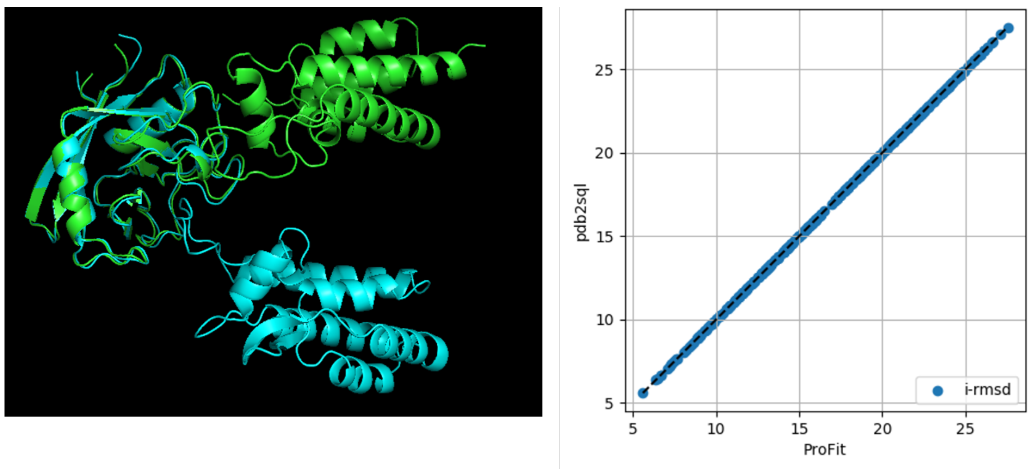

Figure 1. Left - Superimposed model (green) and reference (cyan) structures. Right - comparison of interface rmsd values given by `pdb2sql` and by `ProFit`.

Acknowledgements

We acknowlege contributions from Li Xue, Sonja Georgievska and Lars Ridder.

References

Bakan, A., Meireles, L. M., & Bahar, I. (2011). ProDy: Protein Dynamics Inferred from Theory and Experiments. *Bioinformatics*, *27*(11), 1575–1577. doi:10.1093/bioinformatics/btr168

Basu, S., & Wallner, B. (2016). DockQ: A quality measure for protein-protein docking models. *PLOS ONE*, *11*(8), e0161879. doi:10.1371/journal.pone.0161879

Bayer, M. (2012). SQLAlchemy. In A. Brown & G. Wilson (Eds.), *The architecture of open source applications volume ii: Structure, scale, and a few more fearless hacks*. aosabook.org. Retrieved from http://aosabook.org/en/sqlalchemy.html

Burley, S. K., Berman, H. M., Bhikadiya, C., Bi, C., Chen, L., Costanzo, L. D., Christie, C., et al. (2019). Protein data bank: The single global archive for 3D macromolecular structure data. *Nucleic Acids Research*, *47*(D1), D520–D528. doi:10.1093/nar/gky949

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., et al. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, *25*(11), 1422–1423. doi:10.1093/bioinformatics/btp163

Geng, C., Jung, Y., Renaud, N., Honavar, V., Bonvin, A. M. J. J., & Xue, L. C. (2019). iScore: A novel graph kernel-based function for scoring protein-protein docking models. *Bioinformatics*. doi:10.1093/bioinformatics/btz496

Gowers, Linke, Barnoud, Reddy, Melo, Seyler, Domański, et al. (2016). MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. In Sebastian Benthall & Scott Rostrup (Eds.), *Proceedings of the 15th Python in Science Conference* (pp. 98–105). doi:10.25080/Majora-629e541a-00e

Hamelryck, T., & Manderick, B. (2003). PDB file parser and structure class implemented in Python. *Bioinformatics*, *19*(17), 2308–2310. doi:10.1093/bioinformatics/btg299

Hinsen, K. (2000). The molecular modeling toolkit: A new approach to molecular simulations. *Journal of Computational Chemistry*, *21*(2), 79–85. doi:10.1002/(SICI)1096-987X(20000130)21:2<79::AID-JCC1>3.0.CO;2-B

Lafita, S. A. P., Aleix AND Bliven. (2019). BioJava 5: A community driven open-source bioinformatics library. *PLOS Computational Biology*, *15*(2), 1–8. doi:10.1371/journal.pcbi.1006791

Martin, A. C. R., & Porter, C. T. (2009). *ProFit3.1*. Retrieved from http://www.bioinf.org.uk/software/profit/

Méndez, R., Leplae, R., Maria, L. D., & Wodak, S. J. (2003). Assessment of blind predictions of protein–protein interactions: Current status of docking methods. *Proteins: Structure, Function, and Bioinformatics*, *52*(1), 51–67. doi:10.1002/prot.10393

Rodrigues, J. P. G. L. M., Teixeira, J. M. C., Trellet, M., & Bonvin, A. M. J. J. (2018). Pdb-tools: A swiss army knife for molecular structures. *bioRxiv*. doi:10.1101/483305