BCH2203 Python - 12. Protein structures

Ramses van Zon

April 6, 2022

So what about structure?

- When usin Biopython, we have focused mostly on sequences.
- However, The 3d structure, or shape, of a protein is essential to its biological function.
- Although the aminoacids in the sequence determine the shape in principle, there is no known easy relation between the sequence and the 3d structure.

  (not to mention that it depends on the surrounding liquid's content and conditions)

Computational approaches

- Molecular dynamics (MD) simulations starting from ‘random coil’ of protein sequence.

  One would never do MD in Python, performance is too critical, use GROMACS, NAMD, LAMMPS, . . . written in C++.

- All-atom MD simultations can’t reach time scales of protein folding, but can give ‘bits’ of the trajectory as input of more coarse grained models.
- Other computational studies start from the (experimentally) known structures.
- We can investigate the known structures in Python using Biopython, in particular, the PDB module.

Experimentally determined protein structures

- Experimental techniques like X-ray crystallography, NMR spectroscopy, and cryo-electron microscopy.
- Results are often submitted to, and can be retrieved from the Protein Databank.

  https://www.wwpdb.org
  https://www.ebi.ac.uk/pdbe
  https://www.rcsb.org
  https://bmrb.io
  https://pdbj.org

- You can download the whole database of ~1TB, weekly updated (but would you?).

Protein structure file formats

There are different file formats for protein structures:

- PDB: Legacy (i.e. old, unmaintained) because of limitations on what it can hold.
- PDBx/mmCif: Macro-Molecular Crystallographic Information File
- PDBML/XML format: An XML format
- MMTF: Highly compressed format
- Bundle: PDB formatted archive for large structures

To download, you'll need to know the PDB code.

This is an entry number is assigned to each structure. Typically it is a number followed by 3 letters (E.g. 2HVF).

Look up on website.

The same molecule can have multiple entries.

How to download structures

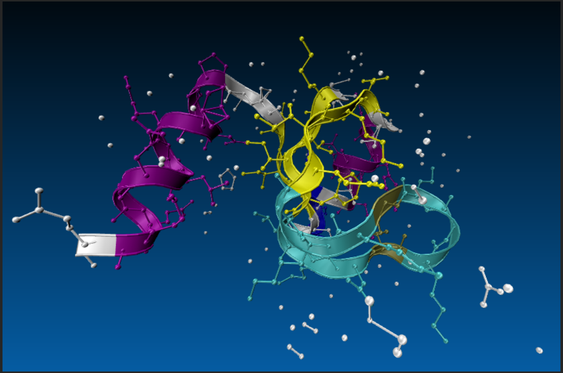
Use PDBList from Bio.PDB to download these.

```
>>> from Bio.PDB import PDBList

>>> pdbl = PDBList()

>>> filename = pdbl.retrieve_pdb_file("2HVF",
...                                   file_format="mmCif")
Downloading PDB structure '2HVF'...

>>> print(filename)
'/gpfs/fs1/home/s/scinet/rzon/hv/2hvf.cif'
```



Sidebar: How to visualize structures

- PyMOL

  This allows you to use python commands.
  https://sourceforge.net/projects/pymol

- Nglview

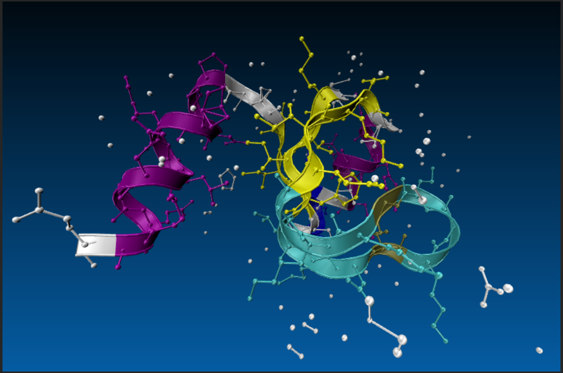
  Can embed visualizations in Jupyter notebooks.
  https://github.com/nglviewer/nglview

```
$ pip install nglview
$ jupyter-nbextension enable nglview --py
```

```
>>> view=nv.show_structure_file(filename)
... view.clear()
... view.add_cartoon()
... view
```

- VMD

  The picture on the right was created with it.
  (CPK + Ribbons, colour by Secondary Structure)
  https://www.ks.uiuc.edu/Research/vmd



There are many more molecular visualization packages.

Reading PDB files into Python

We can use PDB to load the file we just downloaded into a convenient data structure in Python:

```
>>> from Bio.PDB.MMCIFParser import MMCIFParser

>>> parser = MMCIFParser()

>>> structure = parser.get_structure("2hvf", filename)
```

There also exist `PDBParser` and `MMTFParser` for PDB Files and MMTF files

There are also functions for writing these formats, so you could e.g. filter out the water molecules and write the result to a file.

Analyzing structures with Biopython's PDB module

`structure` is structured as a "SMCRA" hierarchy:

Structures contain models

Models contain chains

Chains contain residues

Residues contain atoms

An Atom contains a vector (its position) and other properties, but does not have children

```
>>> structure = parser.get_structure("2hvf", filename)

>>> model = structure[0]

>>> chain = model["A"]

>>> residue = chain[1]

>>> atom = residue['CA']
```

More about the structure of `structure`

https://biopython.org/wiki/The_Biopython_Structural_Bioinformatics_FAQ

Th. Hamelryck and B. Manderick,
PDB file parser and structure class implemented in Python, *Bioinformatics* 19, Issue 17, pp. 2308–2310.

Metadata

The structure also contains meta data in the “header” dict:

```
>>> for metadata,value in structure.header.items():
...     print(metadata,"=",value)

name = Crystal Structure of N-terminal Domain of Ribosomal Protein L9 (NTL9), G34dA
head = STRUCTURAL PROTEIN
idcode = 2HVF
deposition_date = 2006-07-28
structure_method = X-RAY DIFFRACTION
resolution = 1.57
```

Exploring the structure

What models?

Likely, at first, you won't know what IDs are valid to descend into the hierarchy.

The get_models() method returns an iterator over the models, e.g.

```
>>> models = [model for model in structure.get_models()]

>>> models
[<Model id=0>]

>>> type(models[0])
<class 'Bio.PDB.Model.Model'>
```

Exploring the structure

What chains?

Similarly, what chains are there in that model?

```
>>> model = models[0]
>>> chains = [chain for chain in model.get_chains()]

>>> chains
[<Chain id=A>]

>>> model["A"]
[<Chain id=A>]

>>> type(chain[0])
<class 'Bio.PDB.Chain.Chain'>
```

Exploring the structure

What residues?

Similarly, what residues are there in that chain?

```
>>> chain = chains[0]
>>> residues = [residue for residue in chain.get_residues()]

>>> residues
[<Residue MET het= resseq=1 icode= >, <Residue LYS het= resseq=2
icode= >, <Residue VAL het= resseq=3 icode= >, <Residue ILE het=
resseq=4 icode= >, <Residue PHE het= resseq=5 icode= >, <Residue LEU
het= resseq=6 icode= >, <Residue LYS het= resseq=7 icode= >, <Residue
ASP het= resseq=8 icode= >, <Residue VAL het= resseq=9 icode= >,
<Residue LYS het= resseq=10 icode= >, <Residue GLY het= resseq=11
icode= >, <Residue LYS het= resseq=12 icode= >, <Residue GLY het=
resseq=13 icode= >, <Residue LYS het= resseq=14 icode= >, <Residue LYS
het= resseq=15 icode= >, <Residue GLY het= resseq=16 ico
...

>>> residues[0]
<Residue MET het=  resseq=1 icode= >
```

Exploring the structure

What atoms?

Similarly, what atoms are there in that residue?

```
>>> residue = residues[0]
>>> atoms = [atom for atom in residue.get_atoms()]

>>> atoms
[<Atom N>, <Atom CA>, <Atom C>, <Atom O>, <Atom CB>, <Atom CG>, <Atom SD>, <Atom CE>]

>>> type(atoms[0])
<class 'Bio.PDB.Atom.Atom'>
```

What’s in the atom?

```
>>> atom = atoms[0]
>>> atom.get_name()      # atom name (spaces stripped, e.g. "CA"), also get_id()
'N'
>>> atom.get_coord()     # atomic coordinates in Angstroms (10^-10 m)
array([10.634, 21.097,  9.875], dtype=float32)
>>> atom.get_bfactor()   # B factor
14.65
>>> atom.get_occupancy() # occupancy
1.0
>>> atom.get_altloc()    # alternative location specified
' '
>>> atom.get_sigatm()    # std. dev. of atomic parameters
>>> atom.get_siguij()    # std. dev. of anisotropic B factor
>>> atom.get_anisou()    # anisotropic B factor
array([ 0.1718, -0.0211,  0.0061,  0.1908, -0.0004,  0.1937],
      dtype=float32)
>>> atom.get_fullname()  # full atom name (may or may not be equal to get_name())
'N'
```

Conclusion

This is the end. . .

We're at the last lecture.

Congrats everyone for getting to this point!

What did we do again?

Python programming

- Data structures
- File I/O
- Regular expressions
- Visualization
- Profiling
- Machine learning
- Randomness

Biochem specific

- Biopython
- Sequences
- Alignment
- Protein Structure

What's next?

There are quite a few topics we did not get into, e.g.

- Object oriented programming
- Generators
- Database interfaces
- Dates/times
- Parallel processing
- GUIs
- Debugging

For me:

I'll be getting on with the grading.

For you:

One more assignment.

Then, there should be a course evaluation coming your way.

Feel free to tell me how the course could be improved.

Although this was a short course, I hope to have given you enough to start you off on your Python Biochem journey.