The Biopython

Judging from requests for features and information, Bio.PDB is also used by several LPCs (Large Pharmaceutical Companies :-).

4 Is there a Bio.PDB reference?

Yes, and I'd appreciate it if you would refer to Bio.PDB in publications if you make use of it. The reference is:

> Hamelryck, T., Manderick, B. (2003) PDB parser and structure class implemented in Python.

Can I access the header information?

Thanks to Christian Rother you can access some information from the PDB header.

How do I extract a specific `Atom/Residue/Chain/Model` from a Structure?

Easy. Here are some examples:

```
model=structure[0]
chain=model['A']
residue=chain[100]
atom=residue['CA']
```

Note that you can use a shortcut:

```
atom=structure[0]['A'][100]['CA']
```

What is a model id?

The model id is an integer which denotes the rank of the model in the PDB/mmCIF file. The model is starts at 0. Crystal structures generally have only one model (with id 0), while NMR files usually have several models.

What is a chain id?

The chain id is specified in the PDB/mmCIF file, and is a single character (typically a letter).

What is a residue id?

How is disorder handled?

This is one of the strong points of Bio.PDB. It can handle both disordered atoms and point mutations (ie. a Gly and an Ala residue in the same position).

Disorder should be dealt with from two points of view: the atom and the residue

What about B factors?

Well, yes! Bio.PDB supports isotropic and anisotropic B factors, and also deals with standard deviations of anisotropic B factor if present (see 8.4).

What about standard deviation of atomic positions?

Yup, supported. See section 8.4.

I think the SMCRA data structure is not flexible/sexy/whatever enough...

Sure, sure. Everybody is always conIngays widy mostIbody

How do I measure angles?

This can easily be done via the vector representation of the atomic coordinates, and the `calc_angle` function from the `Vector` module:

```
vector1=atom1.get_vector()
vector2=atom2.get_vector()
vector3=atom3.get_vector()
angle=calc_angle(vector1, vector2, vector3)
```

How do I measure torsion angles?

Again, this can easily be done via the vector representation of the atomic coordinates, this time using the `calc_dihedral` function from the `Vector` module:

```
vector1=atom1.get_vector()
vector2=atom2.get_vector()
vector3=atom3.get_vector()
vector4=atom4.get_vector()
angle=calc_dihedral(vector1, vector2, vector3, vector4)
```

```
# Calculate classical coordination number exp_fs=hse.calc_fs_exposure(model)
# Print HSEalpha for a residue
print exp_ca[some_residue]
```

How do I map the residues of two related structures onto each other?

First, create an alignment file in FASTA format, then use the `StructureAlignment`

in the `rotran` attribute of the `Superimposer` object (note that the rotation is right multiplying!). The RMSD is stored in the `rmsd` attribute.

The algorithm used by `Superimposer` comes from *Matrix computations, 2nd ed. Golub, G. & Van Loan (1989)* Superimposer