Google Summer of Code 2011

O|B|F - Biopython Project Proposal

Mikael Trellet

Student Complete Contact Information

Full Name: Mikael Eric Trellet
Short Name: Mikael Trellet
Current Address: Berkenlaan 9, 3707 BA Zeist, Netherlands
Summer Address: 57 rue Georges Heren 91590 BAULNE, France
Email Address: mikael.trellet@gmail.com
Telephone Number: +33650607172

Title: Interface Analysis module for Biopython

Objective: Develop a module to produce significant and useful physico-chemical information on protein-protein interfaces.

On the author:

I am a master student currently involved in protein-peptide docking with the Computational Structural Biology group in Utrecht University. My work involves simulating and analyzing protein-peptide complexes with the HADDOCK docking program. This internship is a part of my Master in "Biology and Computer Science Engineering" at the University of Evry Val d'Essonne. I previously worked with Michael Nilges at *Institut Pasteur*, on modelling and molecular dynamics. This work led to a publication in Science: "*Cell-contact induced posttranslational modification of type IV pilin triggers Neisseria meningitidis dissemination".*

I find my academic background appropriate as I focused on biology, physics and chemistry for 3 years and in computational sciences for the last 2. This translates to a strong knowledge of the PDB file format and of structural biology in general which allows me to distinguish and perform independent research of interest for the broader community. Thanks to my internships, I increased my knowledge of the Python programming language and become familiar with the Biopython module. I have used it to write several scripts for structure analysis, more accurately protein-peptide/protein complexes analysis.

My daily tasks involve manipulating PDB files and this is one of the main reasons of this project proposal for Google Summer Code. While Biopython suffices for most basic tasks, I found it lacking when dealing with molecular complexes, in particular interface analysis, a subject of interest to the structural biology community.

Finally, I am sharing the same lab with a previous GSOC participant, and I find his work and my proposal complimentary, thus bringing a sense of continuity to the enhancement and development of Biopython Bio.PDB structural biology module.

Rationale:

Analysis of protein-protein complexes interfaces at a residue level yields significant information on the overall binding process. Such information can be broadly used for

example in binding affinity studies, interface design, and enzymology. To tap into it, there is a need for tools that systematically and automatically analyze protein structures, or that provide means to this end. Protorop (http://www.bioinformatics.sussex.ac.uk/protorp/) is an example of such a tool and the elevated number of citations the server has had since its publication acknowledge its importance. However, being a webserver, Protorop is not suited for large-scale analysis and it leaves the community dependent on its maintainers to keep the service available. On the other hand, Biopython's structural biology module, Bio.PDB, provides the ideal parsing machinery and programmatic structures for the development of an offline, open-source library for interface analysis. Such a library could be easily used in large-scale analysis of protein-protein interfaces, for example in the CAPRI experiment evaluation or in benchmark statistics. It would be also reasonable, if time permits, to extend this module to deal with protein-DNA or protein-RNA complexes, as Biopython supports nucleic acids already.

Main ideas:

1. Extended Residue object:
    a. Specific object containing physico-chemical information.
    b. Inherits from Residue class
2. Interface object:
    a. Container object.
    b. Inherits from Model
    c. Holds several functions that operate on the different Extended Residues to produce statistical information.

Timeline:

1. Study current Bio.PDB code base
    a. Evaluate possible code reuse.
    b. Evaluate best location of the new code: Bio.PDB.InterfaceAnalysis (?)
2. Extend IUPAC.Data module with residue information:
    a. Weight (already there). Deduce from Atom.element instead (?)
    b. Polar/Charge character (dictionary)
    c. Hydrophobicity Scale
3. Implement Extended Residue class as a subclass of Residue.
    a. Build Extended Residue on the fly or have it hard-coded?
    b. Allow regular operations on Residue to be performed seamlessly in Extended Residue (should come with inheritance)
    c. Develop Tests
4. Implement InterfaceAnalysis module:
    a. Develop Interface class as a subclass of Model.
    b. Develop method to automatically extract Interface from parsed structure upon class instantiation:
        i. e.g. I = Interface(Structure);
        ii. Allow threshold for distance
        iii. Allow chain pairs to ignore (to avoid intra-molecule contacts)
    c. Develop Tests
5. **== Mid-term Evaluation ==**
6. Develop functions for interface analysis:

    a. Calculation of interface polar character statistics (% of polar residues, apolar, etc)
    b. Calculation of BSA calling MSMS or HSA.
    c. Calculation of SS element statistics in the interface through DSSP
    d. …
    e. Tests
7. Develop functions for Interface comparison
    a. Perhaps adapt current RMSD functions to allow usage of Interface Residues.
    b. Otherwise, should be called through something like Ia.rmsd_to(Ib) where Ia and IB are interface objects.
    c. Calculation of iRMSD
    d. Calculation of FCC (Fraction of Common Contacts)
    e. Rough Identity and Similarity %s
    f. …
    g. Tests
8. Code organization and final testing.