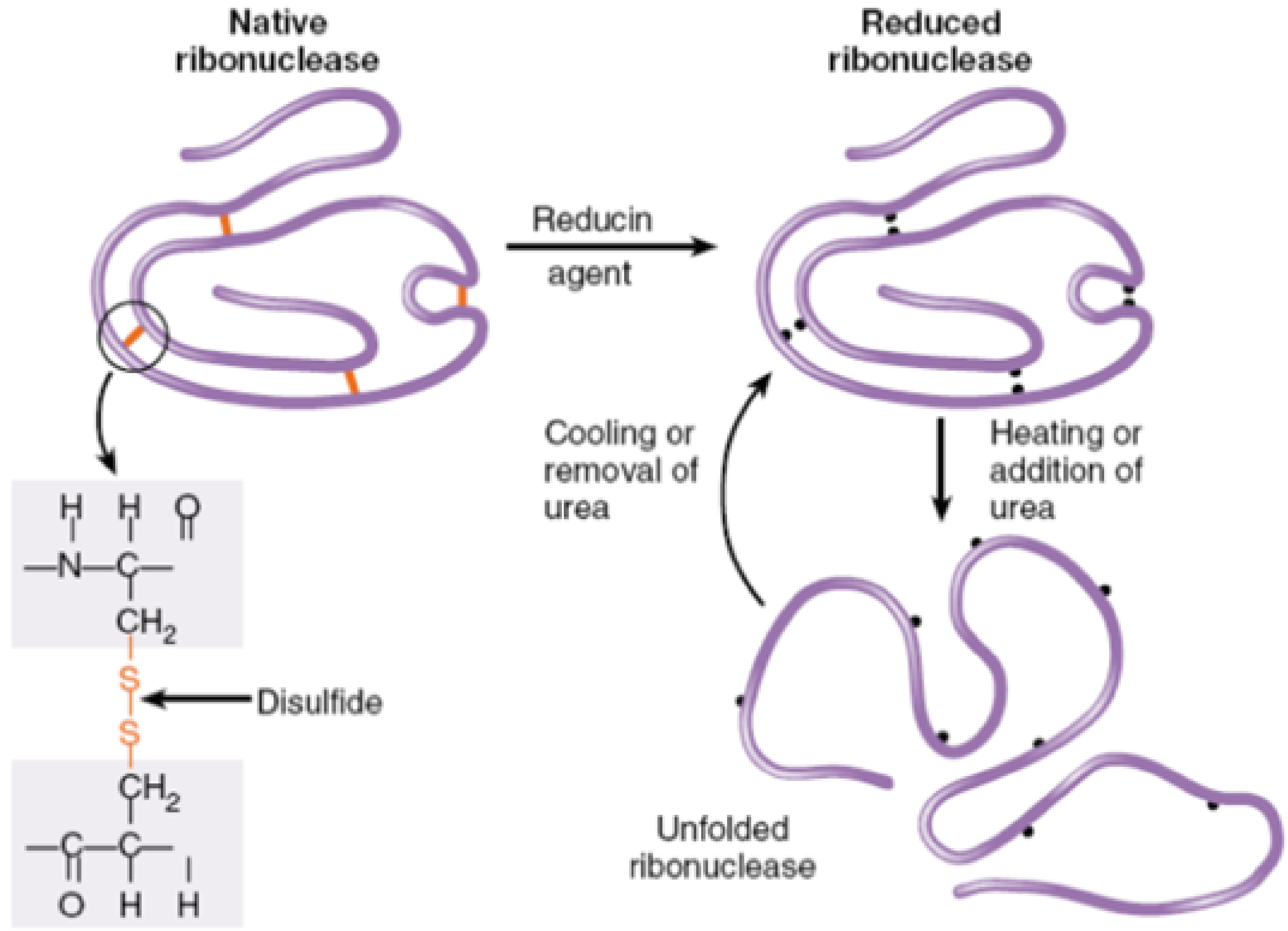
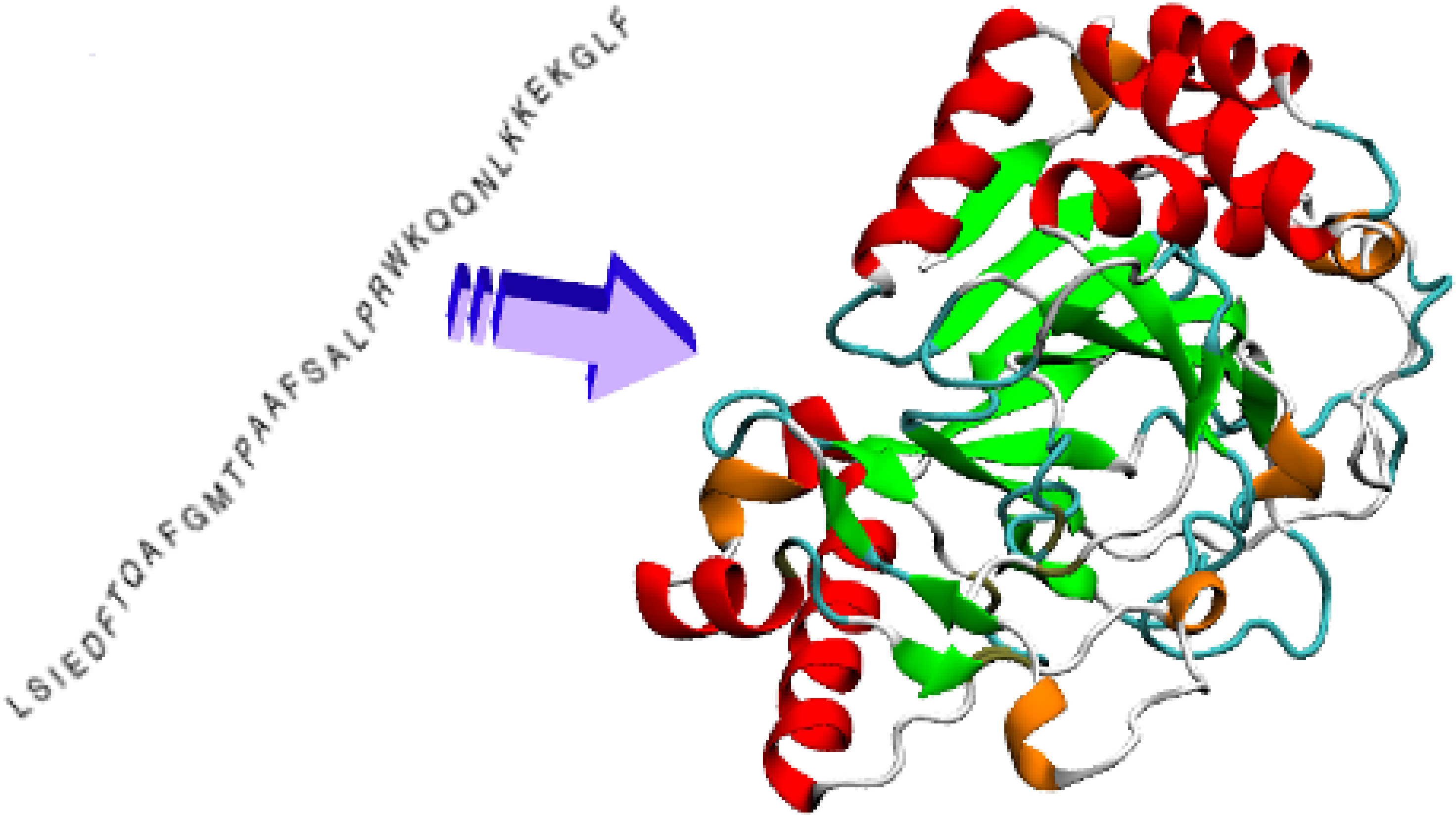
Generation of 3D Structural Descriptors of Protein Conformations from NetCDF-formatted MD Trajectory Files

Abdul-Rashid B. Sampaco III

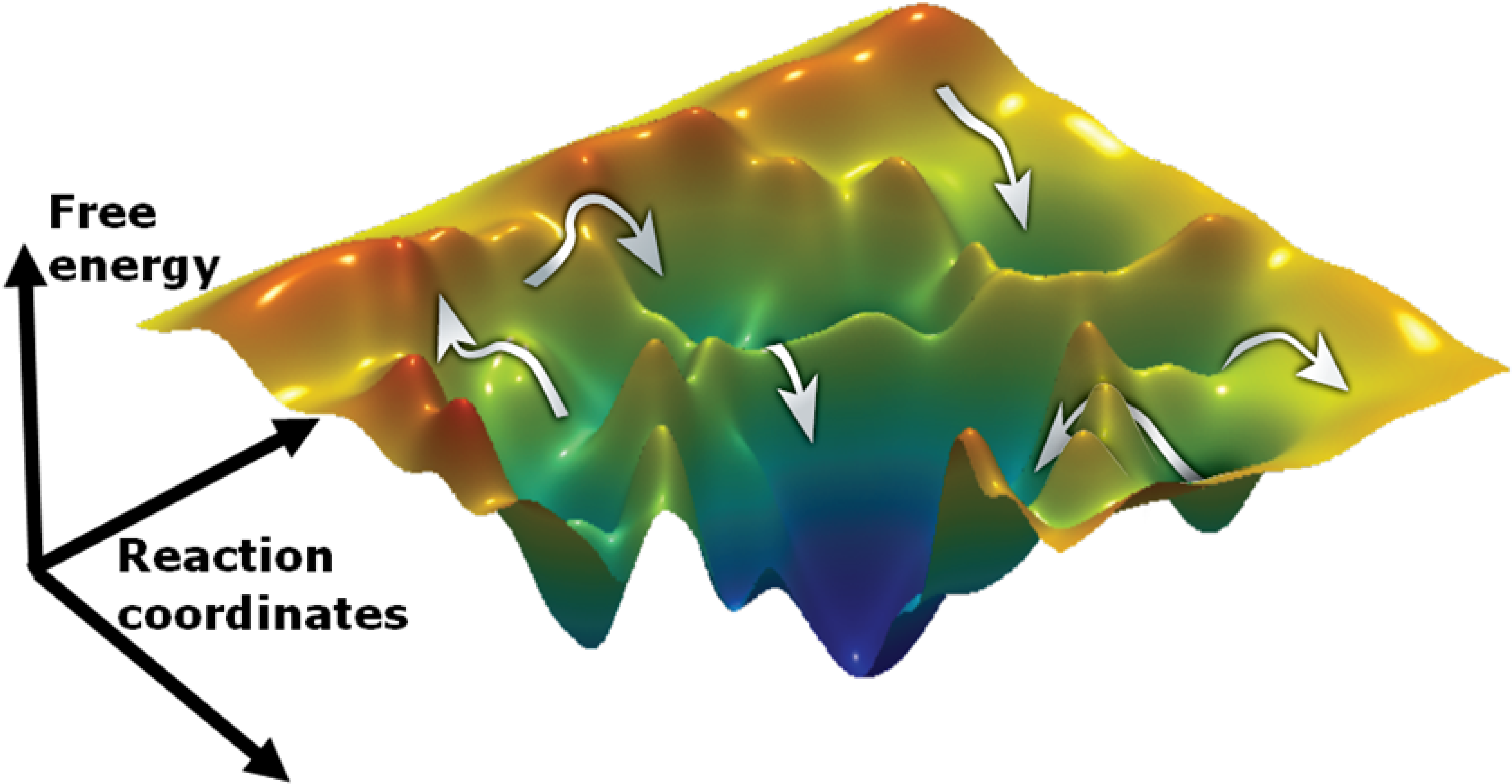
Anfinsen’s Experiment



Anfinsen’s Experiment



Free Energy Surface



Problem

- There is a need to derive other scoring functions that may be able to distinguish native conformations from a large of protein structure data, e.g. MD trajectories.

- The problem is to convert the information stored in the sampled structures to something that is both quantifiable and physically meaningful and then correlate said quantities with closeness to the native state.

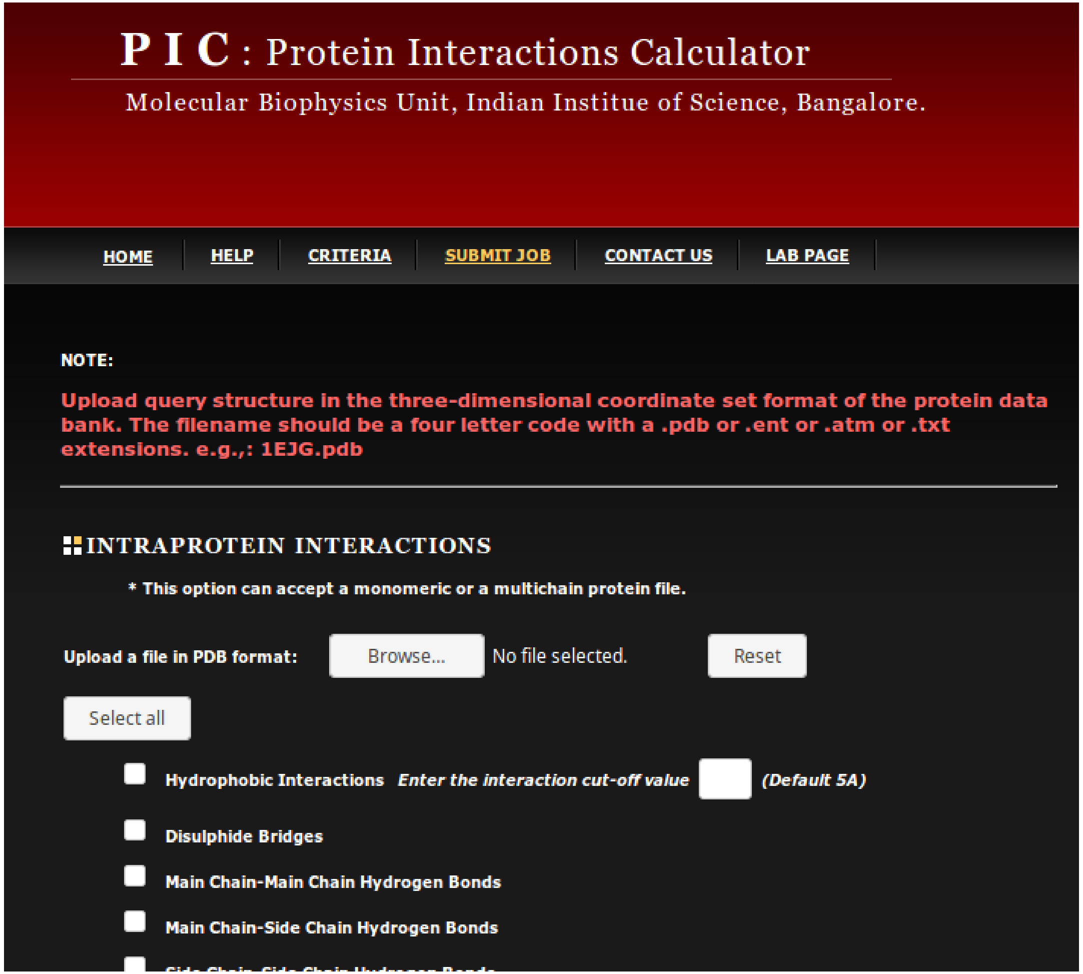
Proposed Solution

- Generate structural descriptors that vary according to a protein’s conformation.

- Possible descriptors may be counts of noncovalent intramolecular interactions, i.e. hydrogen bonds, ionic interactions, aromatic interactions ,etc.

Existing tools that can accomplish the task

- Protein Interactions Calculator (PIC)
- A webserver that takes PDB files as input and calculates interactions based on empirical and semi-empirical rules.



PIC : Protein Interactions Calculator

Molecular Biophysics Unit, Indian Institue of Science, Bangalore.

NOTE:

Upload query structure in the three-dimensional coordinate set format of the protein data bank. The filename should be a four letter code with a .pdb or .ent or .atm or .txt extensions. e.g.,: 1EJG.pdb

INTRAPROTEIN INTERACTIONS

* This option can accept a monomeric or a multichain protein file.

Upload a file in PDB format: No file selected.

□ Hydrophobic Interactions *Enter the interaction cut-off value* □ *(Default 5A)*

□ Disulphide Bridges

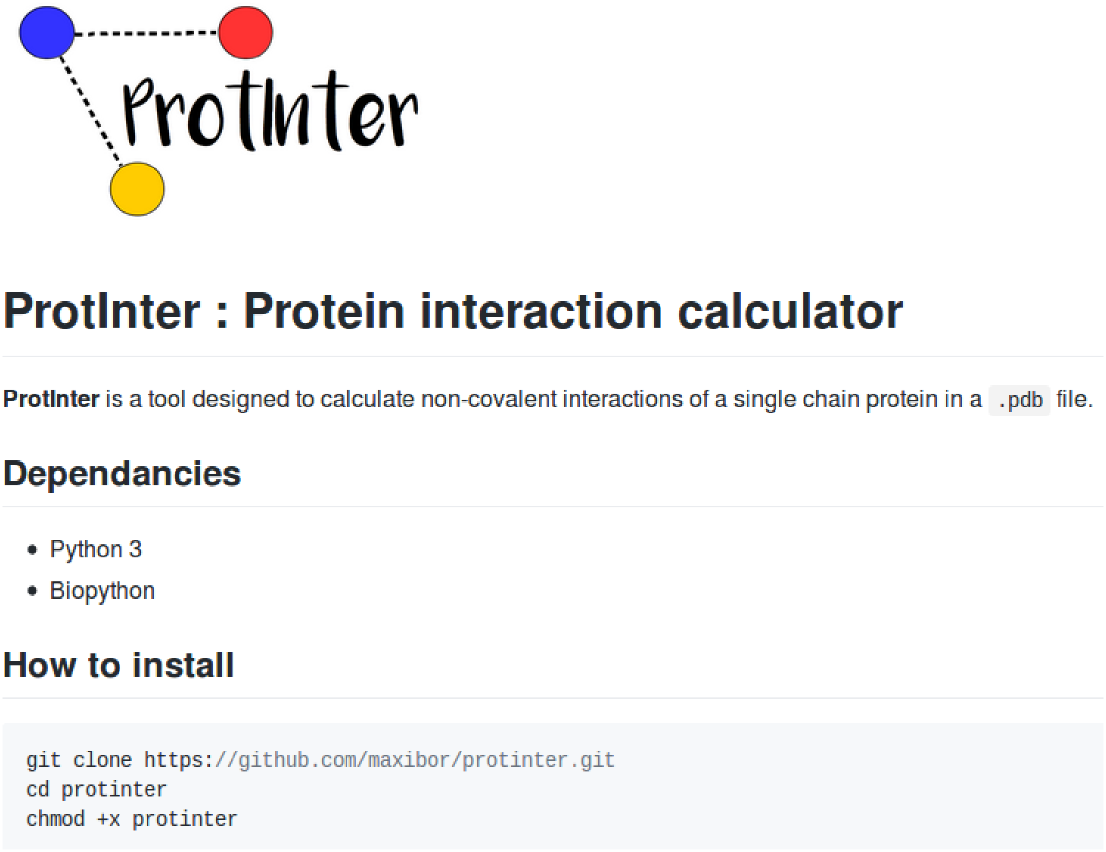
□ Main Chain-Main Chain Hydrogen Bonds

□ Main Chain-Side Chain Hydrogen Bonds

□ Side Chain-Side Chain Hydrogen Bonds

Existing tools that can accomplish the task

- ProtInter
- An open-source command line interface tool written in Python3 that utilizes the library BioPython to parse through PDB files.



ProtInter : Protein interaction calculator

ProtInter is a tool designed to calculate non-covalent interactions of a single chain protein in a `.pdb` file.

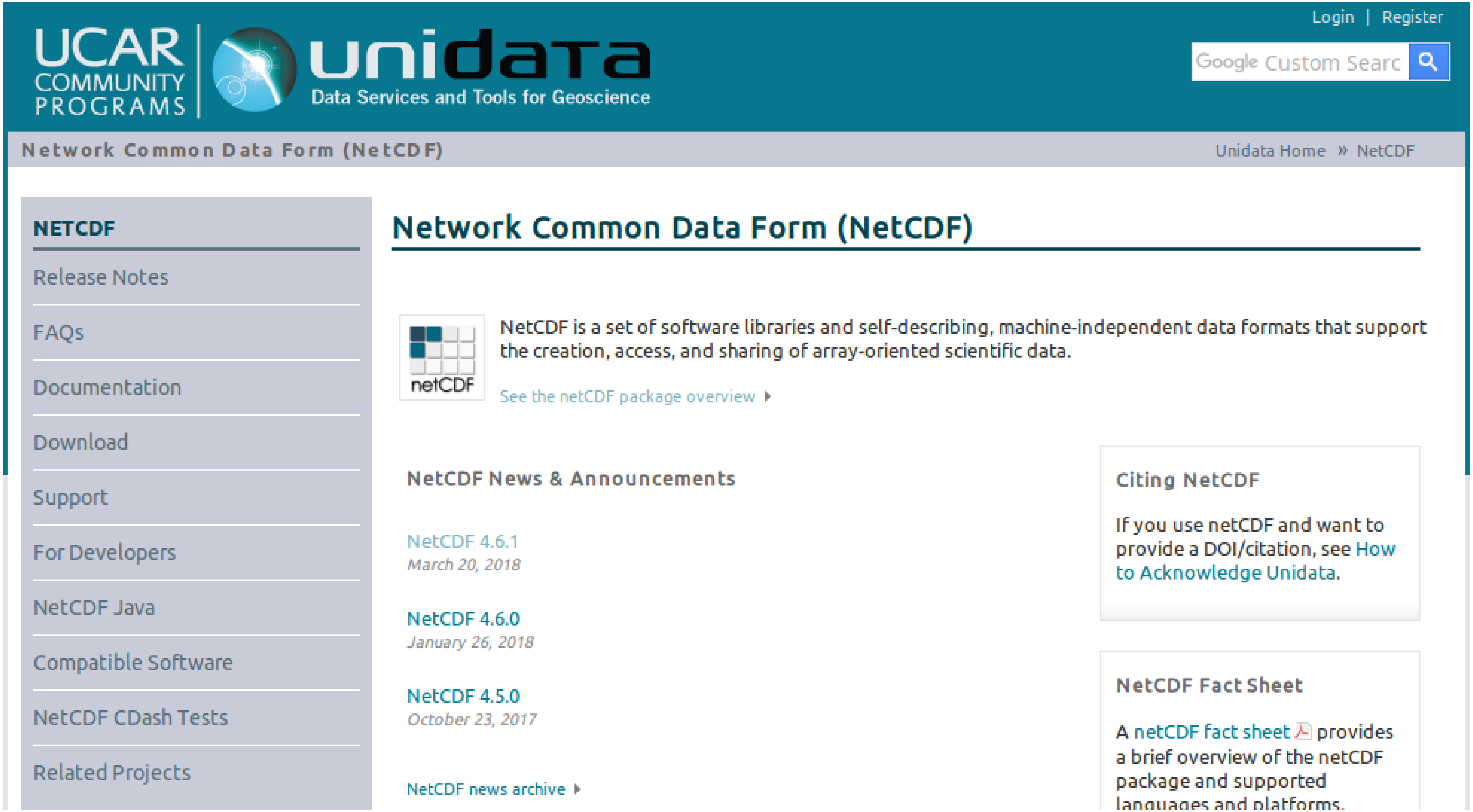
Dependancies

- Python 3
- Biopython

How to install

```
git clone https://github.com/maxibor/protinter.git
cd protinter
chmod +x protinter
```

NetCDF



Network Common Data Form (NetCDF)

NetCDF is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data.

See the netCDF package overview ▸

NetCDF News & Announcements

NetCDF 4.6.1
March 20, 2018

NetCDF 4.6.0
January 26, 2018

NetCDF 4.5.0
October 23, 2017

NetCDF news archive ▸

Citing NetCDF

If you use netCDF and want to provide a DOI/citation, see How to Acknowledge Unidata.

NetCDF Fact Sheet

A netCDF fact sheet provides a brief overview of the netCDF package and supported languages and platforms.

Why reinvent the wheel?

- The existing tools are not specialized for handling large amounts of data.
- The file format I work on is not compatible with the existing tools.
- I want to be able to extend my analyses to other structural descriptors, and not limit analyses to intramolecular interactions.

Control Flow

1. Parse through NetCDF files (trajectory files).
2. Parse through parameter and topology files (AMBER uses prmtop format).
3. Calculate structural descriptors.
4. Write results to output.

Calculations Supported

1. Hydrogen Bonds
2. Ionic Interactions
3. Aromatic-Aromatic Interactions
4. Cation-Pi Interactions
5. Aromatic-Sulfur Interactions
6. Disulfide Bonds
7. Hydrophobic Interactions
8. Hydrophobic over Hydrophilic Ratio

Code Description

Two Modules

1. describe.cpp, describe.hpp – contains the data types and class that performs the calculations.
2. vecop.cpp, vecop.hpp – contains functions for manipulation of vectors (vector operations).

Main program

3. main.cpp – Instantiates a Class *Describe* taking input from the stream, and calls all functions from the class.

describe.cpp

- Custom Data Type *traj* – where the information from both input files are stored. It contains:

1. *coords* – three-dimensional vector of coordinates. (number of frames * number of atoms * xyz)
2. *atoms* – string array of atom names.
3. *residues* – string array of residue names.
4. *charges* – float vector of partial atomic charges.
5. *masses* – float vector of atomic masses.

describe.cpp

- Class *Describe* – takes care of parsing the input files and calculation of the desired quantities.

  Functions:

1. Constructor(string, string)
2. HydrogenBonds(string, string)
3. IonicInteractions(string, string)
4. AromaticAromatic(string, string)
5. CationPi(string, string)
6. AromaticSulphur(string, string)
7. Disulfide(string, string)
8. Hydrophobic(string, string)
9. HydrophobicOverHydrophilic(string, string)

vecop.cpp

Functions

1. Distance(vector<float>, vector<float>)
2. Angle(vector<float>, vector<float>, vector<float>)
3. Scale(vector<float>, float)(vector<float>, int)
4. VectorSum(vector<float>, vector<float>)
5. DotProduct(vector<float>, vector<float>)
6. FindLargestNegativeInteger(vector<int>)
7. Translate(vector<float>, int)(vector<float>, float)(vector<int>, int)(vector<int>, float)
8. CenterOfMass(vector<vector<float>>, vector<float>)
9. IsIn(int, vector<int>)(string, vector<string>)
10. Where(string, vector<string>)
11. VectorSummation(vector<float>)
12. VectorError

Future Plans

- Writing program of the program was rushed, so it is not at its optimum state.
- Most functions were implemented brute-force without referring to existing functions in the standard library.
- First action point is to clean the code to make it run faster and manage memory more efficiently.
- The program is envisioned to be further extended to support a wider selection of structural descriptors.
- Parsers for other file formats will also be implemented as needed.