Report_SBI_PYT_Project

Júlia Mir, Helena Rodriguez

April 3rd, 2018

Modelling protein complexes

Aim of the project

The aim of this project is to create a program in python that is able to reconstruct a whole protein or complex from PDB fragments of pairs of interacting chains. In this case, first aligning the sequences to find good templates for the model and then superimposing them.
We will work with the PDB data a researcher gave us with interactions between protein chains and try to assemble the whole protein or complex.

Theory

Theoretical background

Some proteins, and specially protein complexes, are made from the interaction of many protein chains. These chains usually interact using the lateral chains of their amino-acids, usually with weak forces like Van der Waals forces.
Something very important in protein-protein interactions, and related to the forces and bonds of the interaction, is the distance between atoms. Distances between interacting atoms (from different chains, non-consecutive) are around 5 Angstroms, even if the number can be smaller if we want to be more restrictive. For clashes (two atoms too close to each other) the distance is around 0.4 Angstroms, but this can be bigger in a more restrictive approach.

PDB

PDB (Protein Data Bank) is a database with all known structures for sequenced proteins, nucleic acids and complexes. It includes data in several formats, like .pdb or .ent, to work with structure, or .fasta, to work with protein sequences.

Methods used and why

Since we have PDB data, we already know the secondary structure (maintained by H-bonds), and we only need to join all complexes together.
To do so, we used different methods:
BioPython: We used biopython because it includes many functions and classes useful to work with biological data, and allowed us to use other programs, like ClustalW or PSI-BLAST from python itself.
PSI-BLAST: Position-Specific Iterative Basic Local Alignment Search Tool is a tool derived from NCBI-BLAST. It creates a Position-Specific Scoring Matrix (PSSM) from a multiple sequence alignment using protein-protein Blast. Then uses this matrices again to look for more matching sequences in PDB, allowing to find distant matching proteins. This can be done several times, until we have the desired number and quality of templates to ensemble our complex. Generally, uses SwissProt (another protein structure database) to create the PSSMs and PDB for the final BLAST. We decided to use PSI-BLAST because normal BLAST uses a BLOSUM matrix to look for matches. Using PSI-BLAST we use our own matrix, so we can obtain more accurate matches with a higher statistical significance (higher e-value).

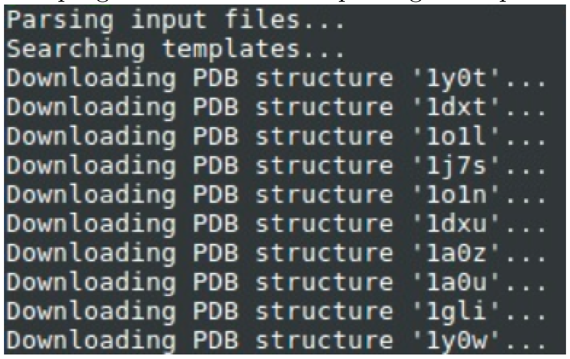
ClustalW: ClustalW is a program written in C++ to make multiple alignments using the command line interface. In our case we used to align the sequences of our query protein's chains and the template's chains in fasta format.
We selected ClustalW for being a fast program to make alignments, and one of the most commonly used.

The problem was solved using superimposition. As we already have the structure of each chain, we perform a rotation and translation of the atoms in order to superimpose the input chains with their corresponding template chains, that way, we reconstruct the whole complex and we can also find new interactions (interactions with chains that are not given in the input files).

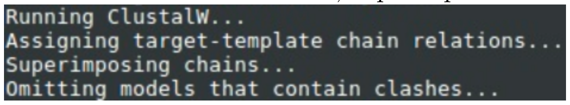
Installation

To use this program you need several dependencies installed. Below you can find a list on where to find the required programs:
-**Clustal 2.1**: *http://www.clustal.org/clustal2/**
-***BLAST 2.2.31+:*** ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.2.31/*
-**Biopython 1.7**: *http://biopython.org/wiki/Download**
-***PDB database:*** https://www.rcsb.org/pages/download/ftp*
We assume Python is already installed in the computer (since it is usually this way), but if not it should also be installed.

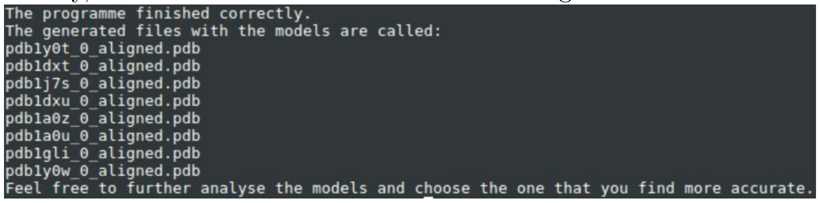
Execution tutorial

To execute our program we need to open the terminal of our computer and go to the folder with our input data (the pdbs the researcher gave us).
Then, we would simply need to run the command:
python3 ModelStructure.py -i infile1.pdb infile2.pdb -d ./databases/pdbseqres.txt -v True
Which, in the case of our example would be:
python3 ModelStructure.py -i AB.pdb BC.pdb -d ./databases/pdbseqres.txt -v True
The program will then start parsing the input files, searching for templates and downloading them:


```
Parsing input files...
Searching templates...
Downloading PDB structure '1y0t'...
Downloading PDB structure '1dxt'...
Downloading PDB structure '1o1l'...
Downloading PDB structure '1j7s'...
Downloading PDB structure '1o1n'...
Downloading PDB structure '1dxu'...
Downloading PDB structure '1a0z'...
Downloading PDB structure '1a0u'...
Downloading PDB structure '1gli'...
Downloading PDB structure '1y0w'...
```

Then it would run ClustalW, superimpose chain and remove models with clashes:


```
Running ClustalW...
Assigning target-template chain relations...
Superimposing chains...
Omitting models that contain clashes...
```

Finally, we would obtain a list with the resulting models' names:


```
The programme finished correctly.
The generated files with the models are called:
pdb1y0t_0_aligned.pdb
pdb1dxt_0_aligned.pdb
pdb1j7s_0_aligned.pdb
pdb1dxu_0_aligned.pdb
pdb1a0z_0_aligned.pdb
pdb1a0u_0_aligned.pdb
pdb1gli_0_aligned.pdb
pdb1y0w_0_aligned.pdb
Feel free to further analyse the models and choose the one that you find more accurate.
```

Analysing the results

One of the things we could do to further analyse our final model is to calculate its energy. Knowing the energy of our model would tell us about its quality (the lower the energy, the better the model). There are many programs to do this, like Prosa, Anolea or DOPE. We have not specified any so everyone can use the one they are more used to.
Also, in the example we have opened the model with Chimera to show the final result, but this is also not in the program because we thought everyone should be able to use the program they prefer, like Chimera or PyMol, for example.
Taking the Hemoglobin model again as an example, we could use Chimera to see the final structure of our protein:

Applicability

We designed the program using hemoglobin as input, where each chain had the same name every time we had it interacting with another but maybe was not the same name in the template.
We assumed that the researcher who gave us the data has labelled each chain with a different name, so if we have the chain A in three interacting pairs, the sequence and structure will always be the same. For this reason, the program would not work if the researcher who provided us with the data has labelled the chains of the different interactions with equal names.

Possible improvements

As said above, our program assumes that the researched who provided us with the information has labelled each chain of their query protein differently. One improvement would be to change this so it could be run even if all chains are named equally or differently every time they appear. To do so, we could first run a ClustalW to see which chains are equal and which are different, to change their names.
Another improvement, if we make the previous CustalW, would be to use only one chain if all chains in the complex are equal. This way the program should be much faster.
Finally, another improvement would be to use more restrictive distances, for both clashes and interactions.