Universitat Pompeu Fabra Barcelona

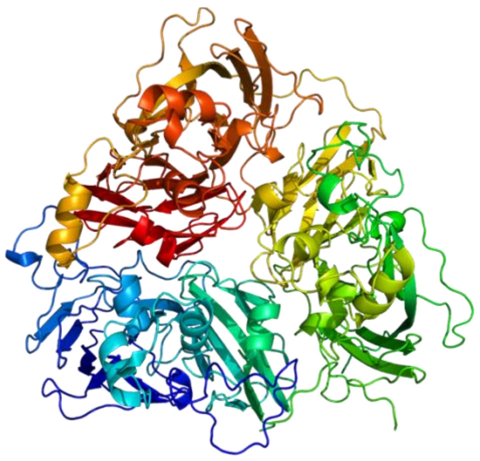
PYSBI CONSTRUCTOR

Standalone program that reconstructs protein macrocomplex

Master in Bioinformatics for Health Sciences

2018-19

Pompeu Fabra University



Laura González
Olfat Khannous
Marina Murillo

Universitat Pompeu Fabra Barcelona

TABLE OF CONTENTS

The following file contains a description of the standalone application (**PYSBI**) that solve the specific problem proposed in the Python and Structural Bioinformatics's subjects of the Master in Bioinformatics for Health Sciences at Pompeu Fabra University.

1. PROBLEM DEFINITION

Given a set of interacting pairs (prot-prot) reconstruct the complete macrocomplex. Using python as the main tool to solve the problem.

2. THEORETICAL BACKGROUND

In order to obtain the structure of a protein, different experimental procedures are needed. But nowadays, several bioinformatics tools have been developed to predict protein structures. The main problem of this project consisted to develop a standalone program that reconstructs and obtain the model of a protein macrocomplex using different pdb files that contain different parts of the protein that has to be rebuilt.

3.2 PROTEINS AND MACROCOMPLEXES

Proteins are biomolecules constituted of one or more chains of amino acid residues. They can interact with different biomolecules such as other proteins, DNA, ARN, lipids among others.

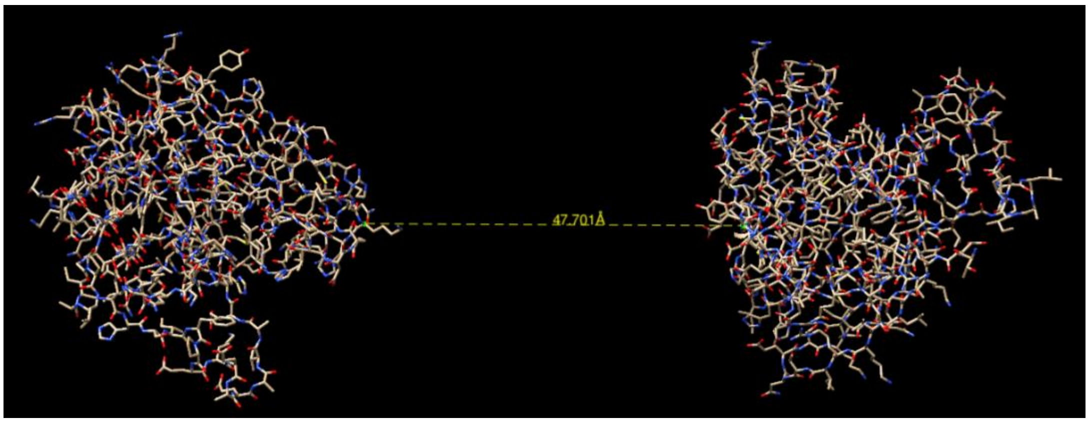
A macrocomplex (protein complex) is a group of two or more polypeptide chains that can be homomultimerics or heteromulimerics being the subunits identical or not respectively.

To be able to have information on the protein structure is one of the key points that allow us to understand the functionality of the protein, for this reason to be able to develop new methods that help in the reconstruction and prediction of protein structures has an important relevance in the field of bioinformatics.

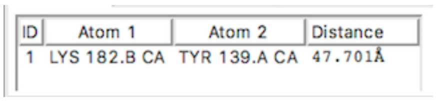
3.2 INTERACTING CHAINS

One of the parameters that must be taken into account during the reconstruction is the interaction of the chains. It is considered that two chains interact if there is contact or spatial proximity between them[1]. The input files that we work contain at least two chains, therefore, identifying if these chains were interacting is an important factor to be able to later reconstruct the complex. In order to identify if they are in contact and therefore, they interact, the distance between the Cα of the residues of the chains is calculated. A distance cutoff of 6Å is used, so a distance lower than 6Å was considered as interaction between residues[2]. (*Fig 1 and Fig 2*)

A



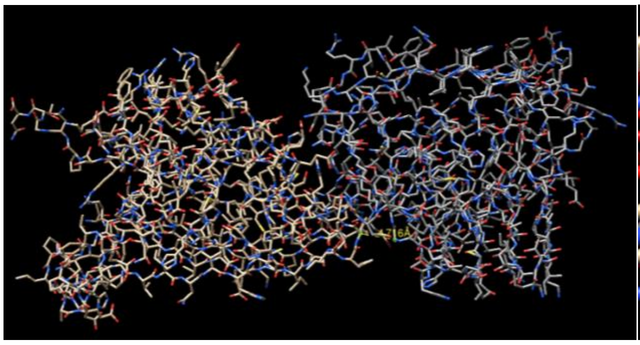
B



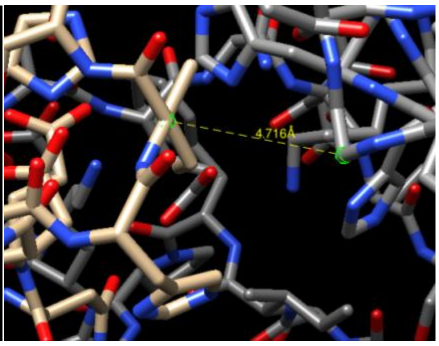
ID	Atom 1	Atom 2	Distance
1	LYS 182.B CA	TYR 139.A CA	47.701Å

Fig1. *A) Example of a file that have non interacting chains. B) Distance of 47.701Å between the Cα of LYS 182 from chain B and the Cα of TYR 139 from chain A. Image obtained from Chimera.*

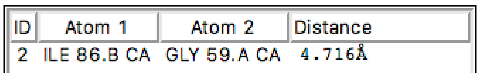
A



B



C



ID	Atom 1	Atom 2	Distance
2	ILE 86.B CA	GLY 59.A CA	4.716Å

Fig2. *A) Example of a file that have interacting chains. B) Close up where we can see the two residues distance C) Distance of 4.716Å between the Cα of ILE 86 of chain B and the Cα of GLY 59 of chain A. Image obtained from Chimera.*

3.2 PAIRWISE ALIGNMENT

Pairwise Sequence Alignment is used to identify regions of similarity that may indicate functional, structural and/or evolutionary relationships between two biological sequences (protein or nucleic acid). In our approach we use it to score the identity of pairs of chains in order to decide if we want to superpose them or not. With an identity score higher than 0.95 we can consider that two chains are equal.

3.2 SUPERIMPOSITION

The protein superimposition is a procedure by which two protein structures are placed in space minimizing the distance between backbone atoms of both structures. One of the two structures is rotated and translated in space, so that fits as good as possible the coordinates of the other structure.

Once two proteins are superimposed we can quantify how different the two structures are. The measurement we use to quantify it is RMSD (Root-Mean-Square deviation) which is the measurement of the average distance between two sets of atoms (atoms of the superimposed proteins).

To be able to perform the superimposition, first a fixed and a moved chain are needed. The chains have to be of the same length and need to have a similarity in sequence greater than 95%. Taking as reference the fixed chain the atoms of the moving chain will be rotated and translated until they completely match (superimpose), the atomic structure of the fixed chain. The translational and rotational coordinates applied are stored in a matrix (rotran). Next, the coordinates stored in the matrix are applied to the partner of the moved chain, in this way, when the partner chain is added to the model, it acquires the correct spatial location. Therefore, from two pairs of chains after the superimposition and addition of the partner chain to the model (the starting model is the fixed chain and it's partner), a structure with 3 chains is obtained (*Fig3*).

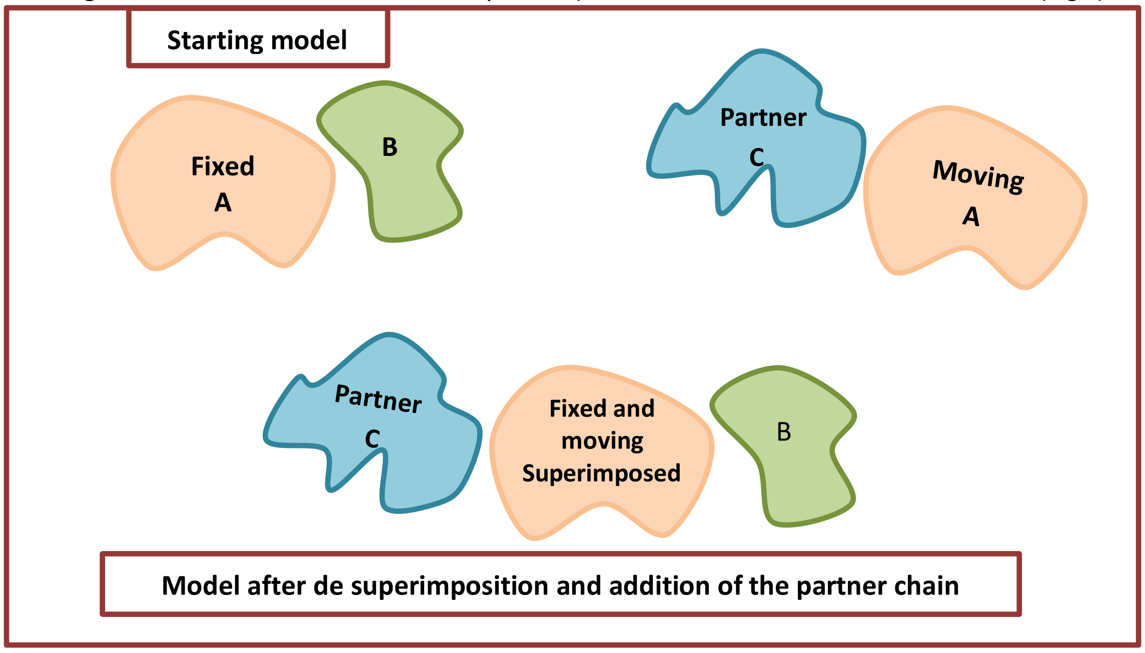

Starting model
Fixed A
B
Partner C
Moving A
Partner C
Fixed and moving Superimposed
B
Model after de superimposition and addition of the partner chain

Fig3. Representation of what happens after applying the superimposition between the moved and fixing chain and applying the coordinates of the matrix rotran to the partner chain.

3.2 CLASHES

Clash is one of the main problems that we can have once that the superimposition and addition of chains to the model is performed. Clashes appear due to the unnatural overlap of any two non-bonding atoms in a protein structure. In order to know if clashes between different chain residues are produced, the distance between them can be calculated, the cutoff that tells us if there are clashes or not is 0.4 Å between the Cα of two residues of different chains, so a calculated distance of less than 0.4Å is indicating us that we are obtaining a wrong model[3].

3. PROGRAM WORKFLOW

3.1 FILTERING

Before starting to treat the files and generate the macrocomplex, the program first detects if the arguments entered by the user are correct, it also checks that the input folder contains files in PDB format. Once this filtering is done the files that have close/interacting chains are selected. (distance of 6Å)

The second step of filtration is to analyze the PDB files, line by line, to see if they have information only for protein or also have information of nucleic acids, for this reason new PDB files are created. Those new PDB files contain lines of atomic information only for proteins. The files will be saved in the filtered_pdb_files folder inside the output folder (specific name entered by arguments). It also checks if the chains a

Then the pdb files are treated in order to create a dictionary that will contain all the chains of all the filtered files. This dictionary will be used during the superimposition step.

-Functions used in the steps:

- **filterinput:** Makes sure that the arguments are the correct ones and that only works with .pdb extension files.
- **parser_filter:** Parses the pdb files and apply the chains_interaction and nuc_prot_filter functions.
- **chains_interaction:** Select the files that contain interacting chains.
- **nuc_prot_filter:** Discards nucleotide information

3.2 BUILDING THE MACROCOMPLEX

Our macrocomplex building approach is based in a recursive function based on the module superimpose of the Biopython[4] package. The recursive function performs the following steps.

Principal function that is used to obtain the model is **Model**.

3.2.1 Selection of the superimposed chains.

In order to perform the superimposition, it is necessary to superimpose the chains that are similar. To check the identity between the chains, a global alignment was made, it was considered that an identity higher than 0.95 indicated that the chains were practically the same. Therefore, pairs of similar chains are stored in a tuple. Chain pairs will be used to perform the superimposition.

Before creating the tuple, a function that allows changing the id since a model cannot contain chains with repeated ids, therefore they are changed by random letters taking in to account that the id cannot be repeated.

Functions used in the steps:

- **tuple_superposition:** Generates the tuple with pairs of similar chains.
- **Align:** Performs the global alignment
- **format_alignment**
- **New_chain_id:** changes the chain id.

3.2.2 Select mov and fix chains.

In this step we use a function that allows to choose the moved chain and the fixed chain.

-**Function used in that step:**

- **find_common_atoms**

3.2.3 Superimposition of the chains.

As we have explained in the theoretical part of the project, the moved and fixed chains are the ones that are superimposed, then, we must apply the rotation matrix to the partner of the moved chain. Then the partner is going to be added to the model.

-**Function used in that step:**

- **get_chain_to_move:** apply the rotran matrix to the partner chain.

3.2.4 Check clashes.

Each time two chains are superimposed, the partner chain must be added to the model, but before adding the string it is necessary to make sure that clashes are not produced between the new chain that is added and the rest of the chains that are already part of the model .

In order to observe the clashes, atomic distances were calculated, if a distance of less than 0.4 A was detected between two atoms of different chains, it was considered that clashes were produced so that the chain was not added to the model.

-**Function used in that step:**

- **clashes**

Since we have a recursive function, the program will perform the steps described above until the desired model is obtained. The different models obtained in pdb format are saved (they are in the output folder).

4. INSTALLATION AND TUTORIAL

This section will help you to install (installation) our program and to understand how to use it properly (tutorial).

INSTALLATION:

I. Download the ***macrocomplex_rec.tar.gz*** folder.

II. Unzip the folder: **tar -xvzf *macrocomplex_rec.tar.gz***

The program uses the Biopython[4] module as the main resource to deal with the information of the pdb files.

FOLDER CONTENT:

- *modules_arg.py*
- *maincompl.py*
- *functions.py*

Several folders in order to try the program:

- *His3*
- *1FN3*

Brief explanation of each of the files.py:

-modules_arg.py: contains the modules that have to be imported and the argparser that makes the program easy to write user-friendly command-line interfaces.

-maincompl.py: contains the main program.

-functions.py: all the functions that are implemented to solve the problem.

TUTORIAL:

COMMAND-LINE ARGUMENTS

--input (-i): Name of the directory that contains the files

--output (-o): Directory that contains the set of outputs that are obtained (pdb files with the resulting macrocomplexes)

--verbose (-v): Using this option the user can follow step by step what the program is doing (It's printed in the Standard Error).

--distance (-d): Interaction distance of the chains of the input pdb files. Used to filter the input. The default value is 6.

COMMAND LINE APPLICATION

The following command line is an example of the simplest way to run our program correctly:

```
python3 maincompl.py –i 1FN3 –o OUTPUT
```

If the user wants to see brief explanations of the steps that the program follows to obtain the output can add the verbose argument (-v) and these descriptions will appear in the standard error:

```
python3 maincompl.py –i 1FN3 –o OUTPUT –v
```

Another optional argument is the distance (-d). Our program filters and uses the input pdbs that have 2 interacting chains using a default distance of 6. If the user wants to use another one can use this argument to change it:

```
python3 maincompl.py –i 1FN3 –o OUTPUT –v –d 8
```

5. EXAMPLES OF APPLICABILITY

- **HEMOGLOBIN (PDB code: 1FN3)**

This is an example where we see the reconstruction of hemoglobin, this is formed by 4 chains. The chains are equal two to two (A2B2). We can see how the reconstruction is almost identical to the real protein obtained with PDB.

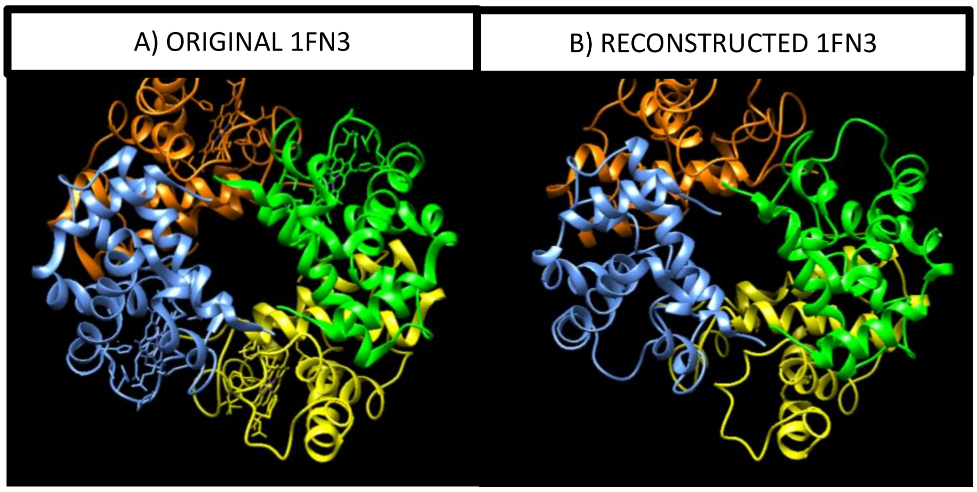


Fig4. A) It is an image of the original hemoglobin visualized with chimera. B) It is the reconstruction of the macrocomplex.

- **HIS3 (EXAMPLE PROVIDED BY THE PROFESSOR)**

In the folder used as input there were 27 files of which only 2 passed the different filters applied. All the strings contained in the files were the same, so we get a homodimer. As we can see we get 4 equal chains.

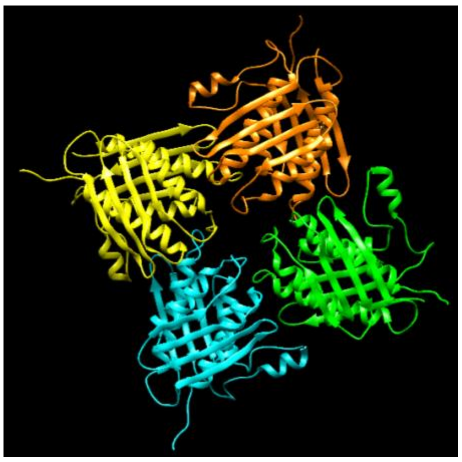


Fig 5. A) Reconstruction of his3 (input given by the teacher)

6. LIMITATIONS

- The program is unable to reconstruct protein-DNA/RNA complexes.

- By convention the chains have a uppercase ids but we during the performance of the program we obtained a run error because of the identification of the chains. To avoid this problem the final approach uses a lowercase dictionary.

- Our program indexes chains with the string "ascii_letters", which contains all the alphabet first in lower and then in uppercase, so a maximum of 54 chains could be indexed without porblems, but higher chain ranks were not considered as input

- The maximum number of chains that our program can add to a model is the total of chains contained in the filtered pdb files, because chains not in contact are considered not to give information and no algorithm to duplicate chains was developed. To duplicate existing chains, a copy of filtered files should be done, with a new maximum of 54, because of the alphabet.

- Some models are the same, because they are the result of different interaction of identical chains, but no filter of similar final models was applied. (that could waste memory resources but for the size of the trial input it was not considered a problem).

- With default distance of chain interactions of 5Ä, no 1fn3 complexes were built, so the 6 Ä was chosen as default instead.

7. BIBLIOGRAPHY

1. Cohen, M., Potapov, V. & Schreiber, G. Four Distances between Pairs of Amino Acids Provide a Precise Description of their Interaction. **5,** (2009).

2. Reddy, B. V. B. & Kaznessis, Y. Use of secondary structural information and C α -C α distance restraints to model protein structures with MODELLER Use of secondary structural information and C α -C α distance restraints to model protein structures with MODELLER. (2007).

3. Srinivas Ramachandran, Pradeep Kota, Feng Ding, and N. V. D. Automated Minimization of Steric Clashes in Protein Structures. **79,** 261–270 (2012).

4. Cock, P. J. A. *et al.* Biopython : freely available Python tools for computational molecular biology and bioinformatics. **25,** 1422–1423 (2009).