Overlapy

A simple Protein Structure Superimposer based in Python

Álvaro Abella Bascarán
Josep Arus Pous
Samuel Miravet Verde

March 16, 2015

Introduction

The aim of this report is to explain the process and results obtained in the development of the *Overlapy* program, a protein structure superimposer coded in Python 3.4, for the subjects of Structural Bioinformatics and Python of the Master in Bioinformatics (Universitat Pompeu Fabra)

Biological interest and Application

The superimposition of structures (also called structural alignment) is a very useful way to establish homology between two or more polymers (e.g. proteins) based on their shape and three-dimensional conformation[1].

The most important features and advantages of the superimposition are:

- You do not need to have prior information about the structures to align[2].
- It can detect evolutionary relationships even between sequences with low sequence similarity due to the fact that the biological function (and consequently the structure) is harder to alter compared to the sequence.[3].
- It permits to define structural motifs related with determined functions if two proteins which share function align, despite being a priori unrelated. In addition, if you have previous evolutionary information, you can detect convergent evolution to a concrete function given a structure[4].
- Assess the quality of a predicted structure model[1].
- It may help in the detection of possible biopolymers interaction (for example if a protein structurally aligns with other protein able to bind RNA) or pharmacological targets (if two proteins are very similar in structure they should share binding sites)[5].

Our Contribution

Given that, having a good tool for structural alignment can be a turning point in the success of a research process. There are several programs to perform a superimposition of structures, like SuperPose[6] or STAMP[7]. However, we want to contribute our grain of sand with "*Overlapy*", a very easy way to do superimpositions of proteins, in a user-friendly web application[8] (coded in Python 3 using the Flask[9]) or in an installable program[10] (based on Python 3[11]).

Implementation

In this section the implementation is analysed. At the end of this part, you will find a general schema of the behaviour and performance of *Overlapy*.

Input Data

The input necessary to run the program consists in two PDBs files given by the user. The program takes and processes them using the Biopython module 'PDBParser' which is used to get the structure of both PDBs, later used by the principal class: Superimposer.

Selecting chains

In order to make *Overlapy* more flexible and competitive, there exists the possibility of selecting the chains to superimpose specifying them with the letter used in the PDB (for example H or L). You can set multiple selections in the same process. If the option is active, the program will only take into account those residues belonging to the chains selected.

In case of the user do not select chains, the program will take all the chains. In both situations, the implementation works in the same way, only changing the residues to superimpose.

Distances between Atoms

Once the PDBs are loaded the first step is performed, consisting on three actions:

1. The aminoacids of each protein are taken and stored (if chains are selected, it will only take those belonging to them).
2. The list containing the residues is processed in order to extract the alpha-carbons (C_α) and their coordinates.
3. Finally, the function `get_environment_distances` computes the distances between a concrete atom and the neighbor ones determined by a radius "r", so each atom will have associated that number of distances.

Generating a Score Matrix

In order to get the structural alignment, the program requires a score matrix (S). It will be used to obtain the best alignment by means of a dynamic programming algorithm. That matrix, of size m x n (m=number of C_α considered for the first PDB; n=for the second one), will include for each pair of atoms a score value computed as:

$$score = \sum_{i=1}^{n} \mid v_i - w_i \mid$$

Where v and w represent our two sets of atoms of n points each one.

Then, if the score is small, it means that the two atoms have a similar environment, so we expect a bigger probability of match between them.

Needleman & Wunsch algorithm

The matrix of scores is used in this step to get the most probable alignment. To achieve this, the dynamic programming algorithm of Needleman & Wunsch is used[12]. Basically it works applying the following steps:

1. A new empty matrix of *Accumulated Scores* (AS) is created, adding the gap state, so the result is a matrix (m+1 x n+1). The gap state has a penalty value 'g' (1, by default).

2. The AS matrix is filled taking the maximum value from the three results generated by the calculations (1), (2) and (3) for each position (i, j). Depending on the formula that gives the best score, a pointer for that position is stored. This step is run iteratively for i and j.

(1)	$S_{(i,j)} + AS_{(i-1,j-1)}$	$\nwarrow$
(2)	$AS_{(i-1,j)} + g$	$\leftarrow$
(3)	$AS_{(i,j-1)} + g$	$\uparrow$

3. Once the matrix AS is completely filled, the score of the alignment can be found in the position AS_{nm}.

4. In order to obtain the most probable alignment the program takes the path determined by the pointers starting at AS_{nm} and ending at the position $AS_{1,1}$. The assignment of symbols follows:

$\nwarrow$	The atoms align
$\leftarrow$	Gap in the sequence i (vertical)
$\uparrow$	Gap in the sequence j (horizontal)

The alignment with gaps is part of the output generated, but in order to obtain the structural alignment we only take those pairs of atoms matching.

The Chain Order Problem

During the development of *Overlapy* we saw there were times when two very close multi-chain proteins gave wrong results, different than the expected. After several test, we found the problem was the chains order in the Needleman & Wunsch algorithm.

Let's suppose two related proteins with chains H and L the first and A and B the second one where we know H with A and L with B are structurally related. The structural alignment we would obtain from the scores matrices of the *figure 1* could be dramatically different.

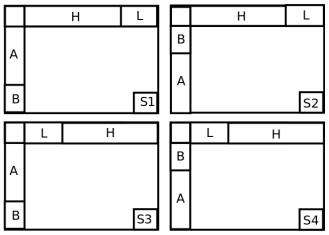


Figure 1: Possible score matrices depending on the disposition of chains.

If we were lucky and the residues in the PDBs were ordered by H and L, and A and B, we would obtain an alignment with score S1 and the result would be the expected and biologically significant. But what does it happen if they are not ordered?

To solve the problem, *Overlapy* test all the possible combinations of chains order, performs the score matrix step for them and it takes as scores reference matrix the one that minimizes the final score (right bottom cell). That matrix will be the one that minimizes the distances between the two structures, i.e. the one that would give the best result between all the possibilities.

Finally, the Needleman & Wunsch Associated Scores matrix is fulfilled using the best score matrix selected in the previous step.

That solution increase the computational cost of the algorithm only if the proteins given have more than one chain. However, even with bigger than two-chains proteins, the computational time is acceptable and it improves substantially the precision and applicability of the program.

Obtaining the Rotation Matrix that minimizes the RMSD

The Root-mean-square deviation (RMSD) is a measure of the average distance between the atoms of superimposed proteins. This is used as a value of the divergence from one set of atoms to another[13]. Given two sets of n atoms aligned v and w, the RMSD is computed as follows:

$$RMSD = \sqrt{\frac{1}{n}\sum_{i=1}^{n} || \, v_i - w_i \, ||^2}$$

Therefore, this step takes the alignment created before by the Needleman & Wunsch algorithm, but removing those residues pairing with a gap, in order to generate a rotation matrix R. This matrix, being applied to the set w, minimizes the RMSD (i.e. minimizes the divergence between the two sets):

$$RMSD = \sqrt{\frac{1}{n}\sum_{i=1}^{n} || \, v_i - Rw_i \, ||^2}$$

The process followed by *Overlapy* to generate the optimal rotation matrix applies the Kabsch & Sander algorithm[14], summarized as:

1. For v and w the program generates two coordinate matrices (n x 3), M_v and M_w, both including 3 columns (coordinates of the atom) and as many rows as the number of atoms in each set.

2. **Translation:** both sets of coordinates are displaced so that the protein centroid (geometrical center) coincides with the origin of coordinates. This is done by subtracting the coordinates of the respective centroid (vector of three elements, each value computed as the mean of the elements of its column) to each atom.

3. **Covariation Matrix:** the algorithm requires the computation of the covariation matrix A as the transpose of the first by the second:

$$A = M_v^T M_w$$

4. **Generating the optimal rotation matrix:** using Singular Value Decomposition (SVD) the program returns the matrix U, that is, the optimal rotation matrix[15]. This step is subdivided in three parts:

(a) Calculate the SVD of the covariance matrix A giving the matrices V, S and W^T.

$$A = VSW^T$$

(b) Decide if is needed to correct the rotation matrix to ensure a right-handed coordinate system taking the value of d, -1 or 1, as:

$$d = sign(determinant(WV^T))$$

(c) The U optimal matrix comes from the calculation:

$$U = W \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{pmatrix} V^T$$

The Kabsch & Sander is automatically computed by *Overlapy* and easily implemented thanks to the python package numpy[16], used to calculate the transposition, determinants and SVD of the intermediate matrices.

After running it, the program returns the optimal rotation matrix and the RMSD given by the one set of atoms against the second set rotated.

Rotating one of the PDBs

The last step performed by *Overlapy* takes the optimal rotation matrix and applies it to the atoms considered in one of the PDBs. With these modified coordinates along with the original coordinates from the other PDB, a new PDB is written containing the two superimposed structures, ready to be visualized in molecular visualization programs like *VMD*[17] or *RasMol*[18].

Output generated

In the web application as well as in the command line program, *Overlapy* allows to obtain:

- The best Needleman & Wunsch alignment pairing the residues structurally more similar.
- The minimal RMSD for the superimposition of the two given protein.
- The rotation matrix that gives the minimal RMSD.
- A PDB containing the coordinates of the two initial structures already superimposed and ready to be visualized in molecular visualization software.

Considerations about the Web Application

The web application[8] implements the *Overlapy* program in a web interface generated and controlled by Flask[9], a framework for web development in Python.

Overlapy web includes interesting features as queue control of jobs or mail advices when the job is completed, as well as all the features explained before (and in the section *Improvements*). In case of having further interested on the implementation, please visit the Git repository[19].

Schema of the Implementation

The *figure 2* shows in a general way, the steps implemented by *Overlapy*.

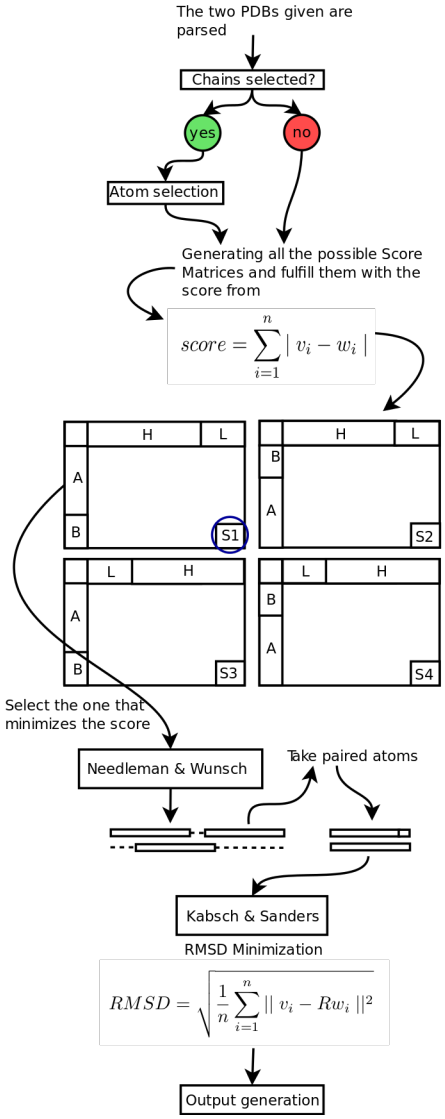


Figure 2: Workflow developed by *Overlapy* in order to generate the results.

Tutorial on Overlapy

Overlapy Web

In order to use the web application of *Overlapy* (http://overlapy.undeadpixels.net/), you just need to load the two PDBs you want to compare, set the chains to select (by default it takes all the chains present, please, be careful with chains that do not belong to the protein (e.g. I)), give your email and run it.

You will be redirected to a 'job' page that will show the form input and a message telling your task is not completed. In any case you will promptly receive an email telling you can go to the 'job' page and access your results. In general the program works fast so you could wait in the web too.

In case the process goes wrong, the page will show an error message. If this occurs, please contact the developers will receive and they will try their best in order to solve your problem.

Installation

If you want to use *Overlapy* locally, first of all, you have to install the program. To do this, go to the `setup.py` directory and type:

```
sudo python3 setup.py install
```

Once you have done this, you will have *Overlapy* installed in your computer and executable with the name `overlapy`.

Basic Usage

This section shows a generic usage example. If you want to see *Overlapy* in action, see the next section: 'Examples of usage' The basic help can be displayed in the terminal by typing

```
overlapy -h
```

or

```
overlapy --help
```

If you type this, you will be able to see there exists several arguments you can use; between all of them, only three are necessary by default:

-i1	first input filename, in PDB format
-i2	second input filename, in PDB format
-o	output filename, in PDB format

So we could run a basic *Overlapy* just typing:

```
overlapy -i1 protein1.pdb -i2 protein2.pdb -o protein1protein2.pdb
```

This will generate a PDB file containing the two initial structures superimposed (stored the input 1 as chain A and chain B for the second) and will show in your terminal:

- The best Needleman & Wunsch alignment pairing the residues structurally more similar.

- The minimal RMSD for the superimposition of the two given protein.
- The rotation matrix that gives the minimal RMSD.

If you want to store all this information in files, you can use three additional arguments:

-m	store the rotation matrix, in `tsv` format
-r	generate a text file containing the minimized RMSD
-a	save the alignment in clustal format

Selecting Chains

Overlapy also allows you to perform the superimposition only over selected chains of your proteins. To do this, it provides to additional arguments:

-c1	Chains selected from the first PDB
-c2	Chains selected from the second PDB

For this function take into account two considerations:

- Provide the chains to use with one letter code (H, L, A, B...).
- If you want to take more than one chain in a PDB, write the letters without spaces (e.g. HL).
- In case you only select chains in one PDB, the other will be taken completely.
- In general, be careful with proteins containing non-proteic elements like inhibitors. They are considered 'chains' in the PDB so if you use them the alignment will be wrong.

Then, we could run *Overlapy* selecting chains using:

```
overlapy -i1 protein1.pdb -c1 HL -i2 protein2.pdb -c2 A -o protein1protein2.pdb -a -m
```

With that command we will find the best superimposition of the chains H and L of the first PDB and the chain A of the second one. The PDB output will be stored in the `protein1protein2.pdb` file. Two extra files, containing the matrix and the alignment, will be generated as we have activate the '-a' and '-m' options; the RMSD will be displayed in the terminal.

Specific Examples

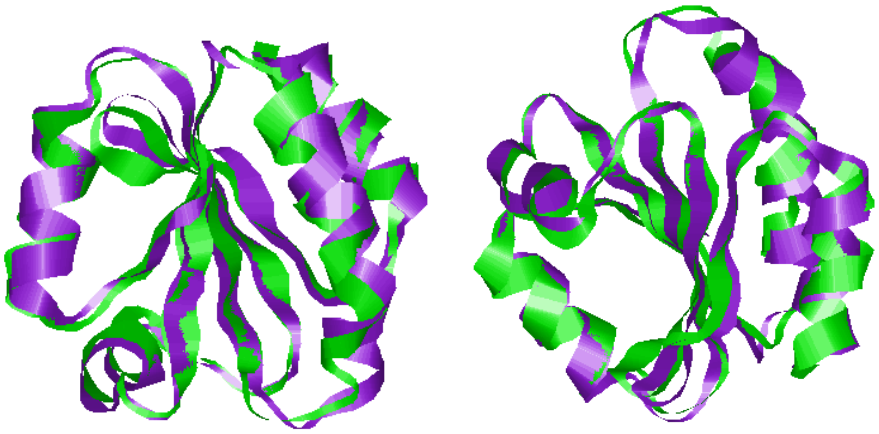
An easy one: Thioredoxins

To see *Overlapy* in action, we are going to start superimposing the thioredoxin of human and the one of *Drosophila melanogaster*. That protein contains only a chain and participates in essential antioxidant and redox-regulatory processes via a pair of conserved cysteine residues[20].

We download the two proteins from `www.rcsb.org/pdb/home/` using their protein codes: 3TRX for the human thioredoxin and 1XWC for the fly one (the files are also available in the 'pdbs' folder of the repository). In order to obtain the superimposition, we run:

```
overlapy -i1 1XWC.pdb -i2 3TRX.pdb -o thioredoxins.pdb -a -m -r
```

This generates a pdb called `thioredoxins.pdb` containing the superimposition. The result can be visualized in the *figure 3* using *RasMol*[18], displaying by ribbons and coloring by chain: chain A (first file, fly thioredoxin) in purple and chain B (human one) in green.



(a) view 1 (b) view 2

Figure 3: Two views in RasMol for the superimposition of fly and human thioredoxins using *Overlapy*.

In addition three extra files are generated: 1) structural alignment in clustal format, 2) rotation matrix in tsv format and 3) a plain text file containing the RMSD (*figure 4*).

```
CLUSTAL X (1.81) multiple sequence alignment

First                           VKDKA-D-L-D-G-QL-T---KAS--G-KLV-V-LD-F--F-AT--W-C-
Second                          IESKTA-F-Q-E-A--L-DAA---GD-K-LVV-V--D-FS-A--TW-C-G

First                           GP--CK-----------------MISPKLVELSTQFADNVV-V-L-KV-D
Second                          --PC-KMIKPFFHSLSEKYSNVI-----------------F-L-E-VD-

First                           -V-D-E-C-EDIA-ME-Y-N-ISS--MPTFV-F-LK--NGVK-VE-E-FA
Second                          V-D-D-C-Q-DVAS-EC-E-V-K-CT-PTFQF-F--KK-GQ-KVGE-F-S

First                           -G-AN-A-K-RL-E-DV
Second                          G-A-NK-E-K-LE-A-T
```

(a) Structural Alignment

```
-1.745171479626019817e-01 -6.518426187085006873e-01 -7.380007166059547874e-01
4.111075404382766862e-01  6.328148409559595233e-01  -6.561524189642207183e-01
8.947258536847790822e-01  -4.179075140009873568e-01 1.575403673606965427e-01
```

(b) Rotation Matrix

```
PDB1:1XWC.pdb
PDB2:3TRX.pdb

RMSD:
1.1600220464442352
```

(c) RMSD

Figure 4: Structural alignment, rotation matrix and RMSD for fly and human thioredoxins superimposition using *Overlapy*.

As we can see, thanks to the superimposition we are able to assert that the 3TRX and 1WXC are really structurally similar, thing that makes sense with the fact they have the same function.

One step further: Subtilisins

Subtilisins belong to subtilases, a group of serine proteases, initially described in *Bacillus subtilis*. In addition to be a key element in several bacterial pathogenic processes, they are a common tool in molecular biology[21].

Let's consider we are interested in study the structural homology of several known-structure subtilisines. For example, 1SBH and 1SCJ, both from *Bacillus subtilis*. We obtain a very bad result (RMSD=6.23) if we run a basic *Overlapy* (Note as we do not activate -r, -m and -a options the alignment, rotation matrix and RMSD will be displayed by the terminal):

```
overlapy -i1 1SBH.pdb -i2 1SCJ.pdb -o subtilisins.pdb
```

And the superimposed PDB obtained shows how it seems the 1SCJ have two chains and only one could be superimposed to 1SBH (*figure 5*) and the second chain of 1SCJ is generating some noise displacing the whole protein to an incorrect result.

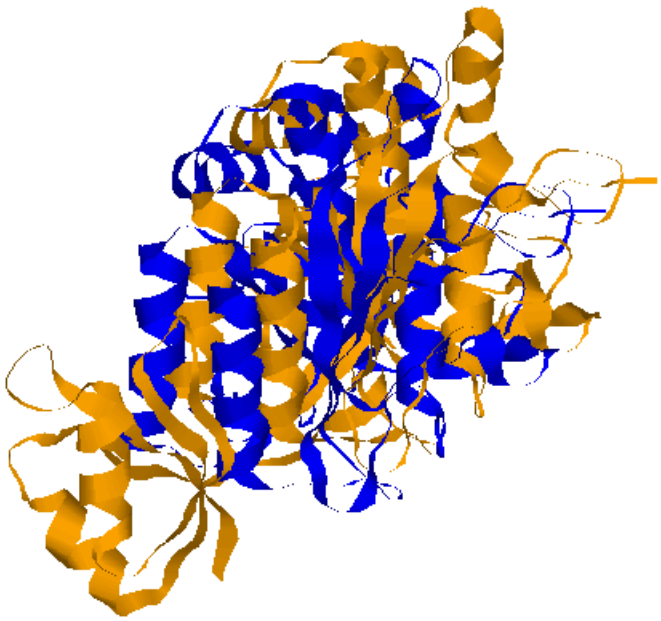


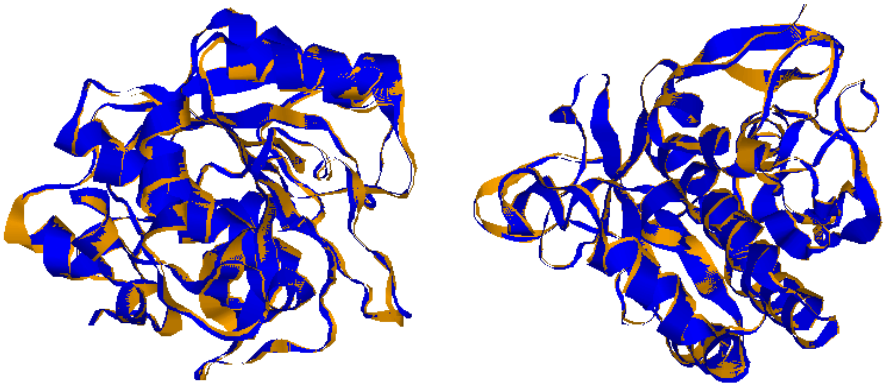
Figure 5: 1SBH(blue) superimposed with the 1SCJ(orange)

We can use the chain selection option of *Overlapy* to see if our assumptions are correct. As 1SBH only have one chain, we only have to select the chain A for the 1SCJ.

```
overlapy -i1 1SBH.pdb -i2 1SCJ.pdb -c2 A -o subtilisins.pdb
```

And this time we obtain a very good structural alignment with a RMSD of 0.44 (*figure 6*).

In reality, 1SBH and 1SCJ are the same protein, but 1SBH is the processed subtilisin, we use this structures as an example of the correct performance of the option 'chains' as we have obtained the expected result.



(a) view 1 (b) view 2

Figure 6: Two views in RasMol for the superimposition of two different subtilisins using *Overlapy*.

Reliability of the results

In order to see if our results are reliable, we compare them with the ones obtained replicating the superimposition of thioredoxins (1WXWC and 3TRX) by SuperPose web version[22].

For this concrete case, we obtain a RMSD of 1.51, very similar to our 1.16 given by *Overlapy* for the same case. In addition, the superimposed PDB result (*figure 7*) is almost exactly the same compared with the *Overlapy* result.

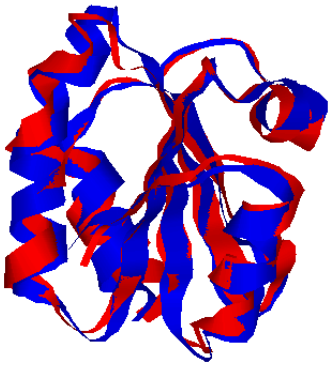


Figure 7: 3TRX(red) superimposed with the 1XWC(blue) using SuperPose.

Regrettably, *SuperPose* web do not give the structural alignment and the rotation matrix so we cannot compare them. However, it seems that at least *Overlapy* works returning similar results to those obtained by *SuperPose*.

Future improvements

Some interesting complementary options we are interested in trying to implement are:

- Include the option to select the C_β instead of the C_α. That approach presents a special advantage: it takes into account the rotameric state of each residue as well as its location along the backbone. In order to solve the problem the glycine does not have C_β we could use the C_α or virtualize the C_β for it.
- Include the possibility of adding from which amino acid to which other the superimposition is going to be restrained.
- The implementation based on Needleman & Wunsch will always return an alignment, even if the structures are absolutely distant. It would be interesting to add a kind of "check" in order to give a confidence or reliability value to the user, for instance, set a minimum sequence similarity.
- Try to adapt the implementation in order to be able to perform superimposition of RNA structures, as made by some projects[23]. This could give us information about the importance of the three-dimensional structure of RNA in some fields like expression, cell response or mutation, as well as finding possible homology relations between structure and function of RNAs.

Conclusion

We have developed from scratch a fully usable program: *Overlapy*, capable of superimposing protein structures. It is able to generate coherent and reliable results from two PDBs, giving an unique PDB with the two structures superimposed in addition to files containing the structural alignment, the rotation matrix and the RMSD in case you need it.

Despite having used Biopython and Numpy to deal with PDB parsing and linear algebra operations, we have fully implemented the algorithm needed to do the superimposition (Needleman & Wunsch and Kabsch & Sander).

Finally, *Overlapy* is presented in three different flavours:

1. Web application: user-friendly web [8] that wants to simplify the life of those researchers that only want a direct and easy to use structural aligner application.
2. Command line tool: an installable program you can download and use locally in your terminal. Simple and useful to integrate in bigger pipelines and scripts.
3. Python library: ready to be included as part of bigger programs.

References

[1] Zhang, Y.; Skolnick, J. (2005). "The protein structure prediction problem could be solved using the current PDB library". Proc Natl Acad Sci USA 102 (4): 102934.

[2] Sippl, M.; Wiederstein, M. (2012). "Detection of spatial correlations in protein structures and molecular complexes". Structure 20 (4): 718728.

[3] Godzik, A. (1996). "The structural alignment between two proteins: Is there a unique answer?". Protein science : a publication of the Protein Society 5 (7): 132538.

[4] Jana, B.; Morcos, F.; Onuchic, JN.(2014). "From structure to function: the convergence of structure based models and co-evolutionary information". Phys Chem. 16 14):6496-507.

[5] Petras, J.; Ilya, A. (2013). "Global and local structural similarity in proteinprotein complexes: Implications for template-based docking". Proteins: Structure, Function, and Bioinformatics. 81 12):21372142.

[6] http://wishart.biology.ualberta.ca/SuperPose/

[7] http://www.compbio.dundee.ac.uk/manuals/stamp.4.2/

[8] http://overlapy.undeadpixels.net/

[9] http://flask.pocoo.org/

[10] https://github.com/SMV818VMS/overlapy

[11] https://www.python.org/download/releases/3.0/

[12] Needleman, S.B.; Wunsch, C.D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of molecular biology (Elsevier) 48 (3): 443453.

[13] Maiorov VN, Crippen GM (1994). "Significance of root-mean-square deviation in comparing three-dimensional structures of globular proteins". J Mol Biol 235 (2): 625634.

[14] Kabsch, W. (1976) "A solution for the best rotation to relate two sets of vectors", Acta Crystallographica 32:922.

[15] Golub, G. H.; Kahan, W. (1965). "Calculating the singular values and pseudo-inverse of a matrix". Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis 2 (2): 205224.

[16] http://www.numpy.org/

[17] http://www.ks.uiuc.edu/Research/vmd/

[18] http://rasmol.org/

[19] https://github.com/undeadpixel/overlapy_web

[20] Wollman E, d'Auriol L, Rimsky L, Shaw A, Jacquot J, Wingfield P et al. (October 1988). "Cloning and expression of a cDNA for human thioredoxin". J. Biol. Chem. 263 (30): 1550612.

[21] Ottesen Mm, Svendsen I (1970). "The subtilisins". Methods Enzymol. 19: 199215.

[22] http://wishart.biology.ualberta.ca/SuperPose/

[23] Hoksza, D.; Svozil, D. (2012). "Efficient RNA pairwise structure comparison by SETTER method". Bioinformatics 28 (14): 18581864.