

Exercise 3: Protein superimposition

We propose the following problem: we have two protein structures and we want to know how different they are structurally. In this situation we are not interested in sequence similarity, we only want to know if the spatial conformation of the two proteins is similar. Our hypothesis is that homologous proteins share a similar structure, although they may have different sequences. Remember that protein structure is more conserved across evolution than sequence. The way to assess structural similarity between proteins is to perform protein superimpositions.

By Alberto Meseguer

Supervised by Baldo Oliva and Altair C. Hernandez

Last update: January 25th, 2024

Theoretical concepts:

Protein superimposition: Protein superimposition is the procedure by which two protein structures are placed in space minimizing the distance between the backbone atoms of both structures. This involves that 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 how different two structures are is the Root-Mean-Square deviation.

RMSD (Root-Mean-Square deviation): It is a measurement of the average distance between two sets of atoms. When working with proteins, these are the atoms of superimposed proteins. This means that one of the two proteins has been rotated and translated to minimize the distance between the two proteins. In particular, we work with the atoms from the protein backbone. The RMSD is defined by the next formula:

$$\text{RMSD}(\mathbf{v}, \mathbf{w}) = \sqrt{\frac{1}{n} \sum_{i=1}^n ((v_{ix} - w_{ix})^2 + (v_{iy} - w_{iy})^2 + (v_{iz} - w_{iz})^2)}$$

Where V and W are the proteins that will be compared, n is the number of atom pairs that will be compared, (v_{ix}, v_{iy}, v_{iz}) are the coordinates of one atom of V and (w_{ix}, w_{iy}, w_{iz}) are the coordinates from one atom of W superposed with V using a linear application of rotation and translation. To compute the RMSD between proteins V and W we will use the coordinates of the backbone atoms of residues that are as close as possible in space after the superposition.

Structural alignment: in sequence alignments, equivalent residues are those that fill the same position in the alignment according to a sequence similarity criteria. In structural alignments, equivalent residues are those that are the closest in space. We can represent structural alignments by using the formats that we already know (clustal, fasta, stockholm).

Structural alignments contain information regarding the substitutions that take place among residues that are placed in specific locations of the protein. Therefore, they also contain evolutionary information. Since protein structure is more conserved than sequence

across evolution, structural alignments can be a reliable source of evolutionary data regarding distant homologous proteins.

We can transform structural alignments into position-specific substitution matrices (PSSM) or hidden markov models (HMM), just as we saw during the exercises 2.1 and 2.2, respectively. By doing so, we will obtain statistical models that contain the evolutionary relations between residues that play the same role in protein structure, for a set of homologous proteins.

Tutorial:

There are many programs that perform protein superimposition (XAM, STAMP, TMalign,...). Some of them, like chimera or pymol, are programs for protein visualization as well. In this practical we are going to use pymol. Pymol is a program for protein structure visualization that can superimpose structures. Besides this, pymol has much more available tools, which makes it a very versatile program.

If you want to download and/or install pymol, you can do it following the next link: <https://pymol.org/2/>

Pymol is used by a big community. This makes possible that we have an extense documentation and lots of tutorials for pymol. To solve any doubts you may have working with pymol, check the pymolwiki: https://pymolwiki.org/index.php/Main_Page. In this page you have access to the documentation of pymol as well as many other resources such as tutorials. Also, there are many pymol tutorials in youtube, in particular this list was very useful to me: https://youtube.com/playlist?list=PLUMhYZpMLtal_Z7to3by2ATHP-cl4ma5X

In this tutorial we are going to work with the structures of globins. Globins are proteins that are responsible for oxygen transport. To do so, they carry within them an hemo group, that allows oxygen binding. Globin monomers can associate creating homotetramers (protein complexes created by 4 identical subunits). We can take a look to these proteins by using pymol, just open pymol and open one of the structures you wave in the directory in this practical:

file > open > select files

Or you can just use the fetch command and download a structure from the PDB to your pymol session:

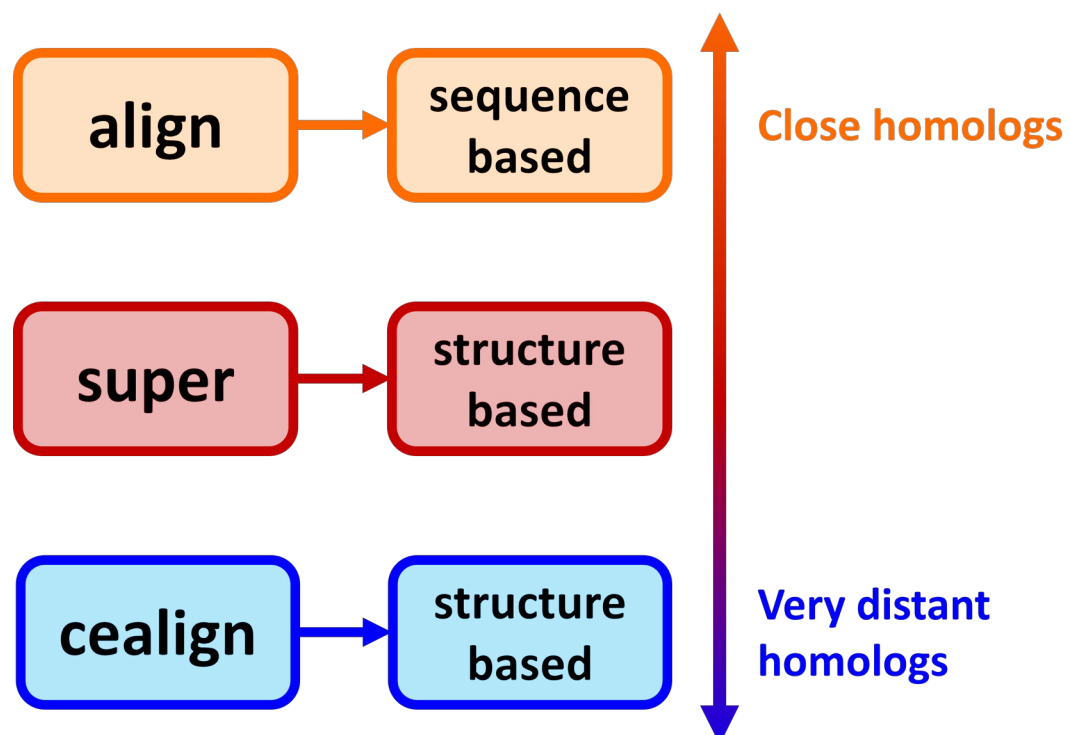
fetch lecd

Step 1: Superimpose 2 structures

Pymol has 3 different commands that can be used to perform protein superimposition: align, super and cealign. Each of these commands performs superimposition using a different algorithm and they perform better or worse depending on the similarity of the proteins you want to superimpose.

- **Align:** It performs a sequence alignment between the two proteins. Then, it matches the two structures by putting close in space the pairs of amino acids that were aligned based on sequence information. This method works well when the proteins that we are working with are close homologs, because align requires the two proteins to have similar sequences.
- **Super:** It superimposes two structures without using sequence information. It just tries to minimize the RMSD between the two proteins. This method works well with proteins that are distant homologs. Distant homologs have different sequences but similar structures, and since super only uses structural information, it just needs structural similarity to perform well.
- **Cealign:** It superimposes two structures without using sequence information. It just tries to minimize the RMSD between the two proteins, but using a different algorithm than super. Cealign works well with proteins that are very distant homologs. In such cases, sequence and structure can be quite different between the proteins we are working with.

It is clear to see that each method is intended for proteins within an homology range. So, although cealign can superimpose very distant proteins, for close homologs the command performing the best will be align. You can see the properties of these different commands in the following image:



In this tutorial we are going to work with the super command, since the proteins that we will superimpose have similar structures, but quite different sequences. The syntax to use these three commands is the same, so if you know how to use one of them you can use the three of them. Here you have the links to the documentation for each of these commands:

Align: <https://pymolwiki.org/index.php/Align>

Super: <https://pymolwiki.org/index.php/Super>

Cealign: <https://pymolwiki.org/index.php/Cealign>

Start by loading in your pymol session the pdb structures 4mbn and 1ecd (you can load them from the directory of the practical or fetch them from the PDB)

```
fetch 4mbn
```

```
fetch 1ecd
```

Remove all the water molecules, since they could interfere with the superimposition:

```
remove resn hoh
```

Use the super command to superimpose the two structures:

```
super 4mbn, 1ecd, object=aln2
```

See that the syntax involves putting first the pdb object that will move, then the pdb object that will remain fixed, and finally the name that we want to give to the superimposition object.

After executing this command you will have the RMSD value for this superimposition in your pymol console:

```
File Edit Build Movie Display Setting Scene Mouse Wizard Plugin Help
ExecutiveAlign: 775 atoms aligned.
ExecutiveRMS: 23 atoms rejected during cycle 1 (RMSD=2.77).
ExecutiveRMS: 12 atoms rejected during cycle 2 (RMSD=2.58).
ExecutiveRMS: 5 atoms rejected during cycle 3 (RMSD=2.51).
ExecutiveRMS: 1 atoms rejected during cycle 4 (RMSD=2.49).
ExecutiveRMS: 1 atoms rejected during cycle 5 (RMSD=2.48).
Executive: RMSD = 2.476 (733 to 733 atoms)
Executive: object "aln2" created.
```

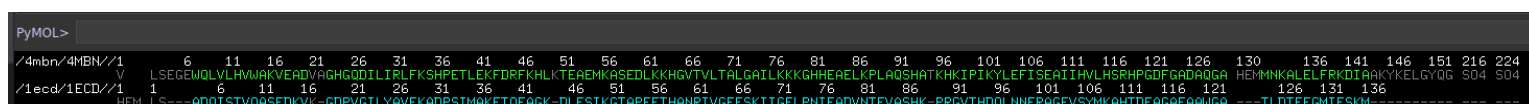
Besides, we are giving a name to the superimposition object that we are creating in this execution. Giving a name to this superimposition has the advantage that it enables us to

represent it as an alignment. This alignment is based in structural similarity, remember that we are using the super command. Here the criteria that we use to determine what amino acids go in the same column is to minimize the RMSD between the two structures. Therefore, this alignment that we get is based on structural similarity and not sequence similarity (like the ones we saw on previous practicals).

You can visualize this alignment if you click on the S button on the bottom right corner of your pymol screen:



This will display the alignment in the section of the screen where we can visualize protein sequences:



See that you can select groups of amino acids in this sequence/alignment display, which is very useful. Finally, we can store this alignment as a separate file in clustalw format using the save command:

save aln2.aln, aln2

See that the syntax of the command is save, then the name of the output file, and finally the name of the object that contains the alignment (aln2 in this case). This command will save the alignment in your pymol directory.

Step 2: Superimpose as many structures as you want

In this next step we are going to superimpose 4 globin proteins. These 4 proteins are in the P3_directory or you can get them from the PDB using the fetch command. They are 1ecd.pdb, 1lh1.pdb, 2lhb.pdb and 4mbn.pdb.

Remember to remove the water molecules:

remove resn hoh

To superimpose many structures together you need to set one structure as reference (1ecd) and superimpose the rest of proteins on top. To do this you will have to use the

super command several times. See that, since we want all the 4 proteins to be included into the same superimposition, we will force pymol to put all the superimpositions into the same object:

super 4mbn, 1ecd, object=aln4

super 2lhb, 1ecd, object=aln4

super 1lh1, 1ecd, object=aln4

After doing these commands you should see that the 4 proteins are properly superimposed. Now take a look at the alignment you are getting from this superimposition, you should see something like this:

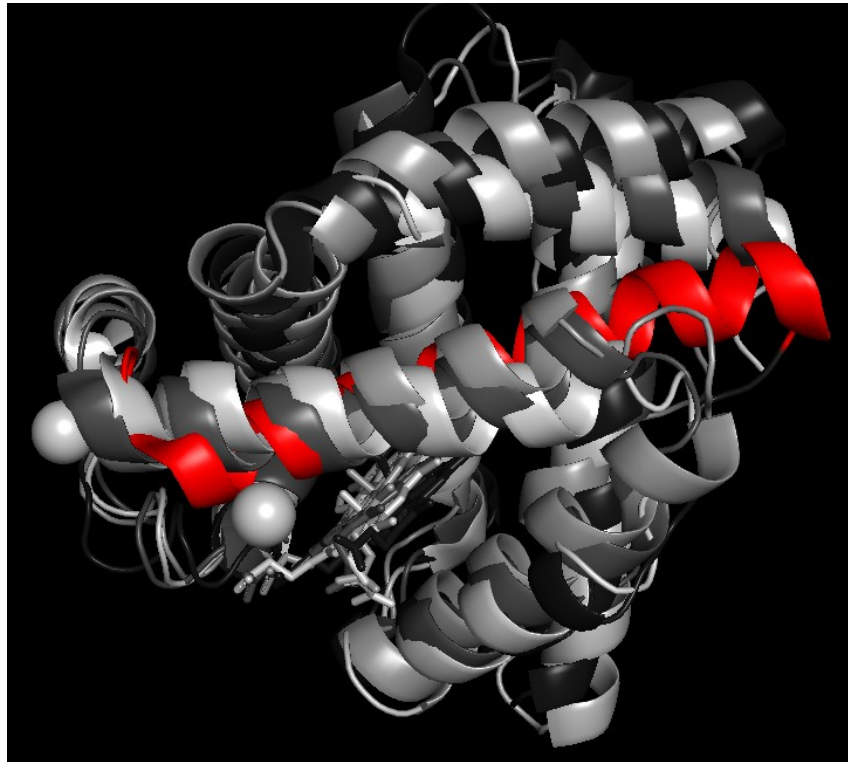
```
PyMOL>
/4mbn/4MBN/1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126
V L S EGE-----M--QLVLAHIAKVEADVA-G---HGQDILIRLFKSHPETLEKFDRF---KHLK-----TEADMK-A--SEDLKKHGVTLTALGAILK-KK-GHHEAELKPLAQSHATKHKITPKYLEFISEAIIHVLHSHR---PG-D-FGADADG
/2lhb/2LHB/1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136
P I V DTGSVAPLS-A-REKTKIRSAWAPYST-YETSGVDILVKFFSTPAQGEFFPKF---KGLT---TADLKKSHDVHHAERIINAVDDAVASMD-DTEKMSKRLNLSGKHAK-SFQVDPE--YFKVL-AAVIADTVR---A-G-DAGFEKL-
/1lh1/1LH1/1 1 2 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131
G HEM ACE AL-----T-ESQARLVKSSUEEFNA-N-IPKHTHRFFILVLEIAPAKDLFSFLKGTSEVPNNPELQAHAGKVFKLVYERAIQLEVTG---VV---VTDATLKNLGSVHV-SKGVADAHFPVVKAILKTIKEVVGAKW-S-EELNSAUT
/1ecd/1ECD/1 1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116
HEM --- --- ---LS-A-DQISTVQHSFDKV-K-G--DPVGILYAVFKADPSIMAKTQF---AGK-----DLESIK-G--TAPFETHANRIVGFFSKIIG-EL-PHIEADVNTFVASHK-PRGVTHDQLNNFRAGFVSYMKHT---D-F-AGAEARAG
```

Do you think that this alignment is good? Do you think that this alignment corresponds with the superimposition that you are seeing in the screen? Can you identify a region of this alignment that is wrong? Take five minutes to think about these questions.

In this alignment you can see that there is a big region that is not aligned with the rest of the sequences:

```
PyMOL>
/4mbn/4MBN/1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126
V L S EGE-----M--QLVLAHIAKVEADVA-G---HGQDILIRLFKSHPETLEKFDRF---KHLK-----TEADMK-A--SEDLKKHGVTLTALGAILK-KK-GHHEAELKPLAQSHATKHKITPKYLEFISEAIIHVLHSHR---PG-D-FGADADG
/2lhb/2LHB/1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136
P I V DTGSVAPLS-A-REKTKIRSAWAPYST-YETSGVDILVKFFSTPAQGEFFPKF---KGLT---TADLKKSHDVHHAERIINAVDDAVASMD-DTEKMSKRLNLSGKHAK-SFQVDPE--YFKVL-AAVIADTVR---A-G-DAGFEKL-
/1lh1/1LH1/1 1 2 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131
G HEM ACE AL-----T-ESQARLVKSSUEEFNA-N-IPKHTHRFFILVLEIAPAKDLFSFLKGTSEVPNNPELQAHAGKVFKLVYERAIQLEVTG---VV---VTDATLKNLGSVHV-SKGVADAHFPVVKAILKTIKEVVGAKW-S-EELNSAUT
/1ecd/1ECD/1 1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116
HEM --- --- ---LS-A-DQISTVQHSFDKV-K-G--DPVGILYAVFKADPSIMAKTQF---AGK-----DLESIK-G--TAPFETHANRIVGFFSKIIG-EL-PHIEADVNTFVASHK-PRGVTHDQLNNFRAGFVSYMKHT---D-F-AGAEARAG
```

If you look for this region in the superimposition you will see that is properly superimposed, but is slightly different to the other superimposed structures. Here you can see this problematic regions highlighted in red:



As you can see, this helix is not fitting with the other super well. When pymol builds the alignment based on structural information is putting a distance cutoff between the amino acids that can be paired. Since this helix is different to the others, the amino acids of the helix don't pass the distance cutoff and they become unaligned. This is an artifact that we can correct using properly the command super.

Now, we are going to force that the distance cutoff for two amino acids to be aligned is 5.0 Amstrongs, instead of 2.0 Amstrongs which is the default. To do this, restart your pymol session, load again all the proteins and use the following commands:

super 4mbn, 1ecd, object=aln4_corrected, cutoff=5.0

super 2lhb, 1ecd, object=aln4_corrected, cutoff=5.0

super 1lh1, 1ecd, object=aln4_corrected, cutoff=5.0

After doing this you should get an alignment that looks way better:

```
PyMOL>
/4mbn/4MBN//1
V L S EGE-----M-QLVLHVMKVEADY-AG--HGQDILRLFKSHPETLEKFDRF-----KHLK-TEAEHK-A--SEDLKKGVTYLTALGAILKKK-GHHEBELKPLAOSHATKHKIKPIKYLEFTSEATIHVLHSPH---PGDFGADADGA HEMMNAKLELFRKD---IARKYKELGYC
/2lhb/2LHB//1
P I V DTGSAVPLSA-AEKTIRSAWAPVYSTYETSGVDILVKFFTSTPADEFFPKF---KGLT-TADELKKSDVRWAERIINAVDDAVASMDDEKMSMKLRNLGSKHAK-SFOVDPE--YFKVL-AAVIADTVR---AGDAGFEKL-M---SMICILLRSAY HEM
/1lh1/1LH1//1
I 2 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136 141 146 151
HEM ACE AL-----TESQAALVKSSWEFNA-NIPKHTRRFFILVLEIAPAKDLFSFKGTSEVPCHNPQLQA-H--AGKVEKLYEARIOLEVTVGV---VTRATLKNLGSVHV-SKGVADARFPVKEAILKTIKEVGAKUSEELNSAWTI---AYDELAIVIKK---EMDDRA
/1ecd/1ECD//1
HEM ---L-1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136
-----LSA-DQISTVQASFDKV--KG--DPVGILYAVFKADPSIMAKFTQF-----AGK--DLESIK-G--TAPFETHANRIVGFFSKIIGEL-PNIEADVNTFVASHK-PRGVTHDQLNFRAGFVSVMKAHT---DFAGAEAWGA---TLDTFFGMIFS---KM
```

This alignment looks way better. However, we still get gaps in it. Can you find where are these gaps happening? Can you find any pattern? Does it make sense to you?

Interestingly, most of the gaps are happening in loops. Loops usually are the most variable regions in a protein family, both in terms of structure and also sequence. Structure variability in loops is usually high because they are the most flexible parts of a protein.

Finally, you can store this alignment in clustalw format in your pymol directory by using the save command:

```
save aln4_corrected.aln, aln4_corrected
```

Step 3: Create a structure based HMM

The MSA we just created is different from any other MSA we created before. This new MSA is based on structural similarity. Also, since the proteins that we aligned are distant homologs (different sequence, similar structure). Therefore, our MSA contains structural information for distant homologs of the globin protein family. We can use this MSA to create a HMM, and then this HMM will contain structural information for distant homologs of the globin family.

To create a HMM out of this MSA the first thing you have to do is to upload the MSA file to the citrix. We recommend you to do that by using google drive.

Next, you have to transform the format of the alignment, from clustalw to stockholm, which is the only format that programs from the HMMer package can take as input:

```
perl ~/Documents/perl_scripts/convertMod2.pl -in c -out f  
<aln4_corrected.aln>aln4_corrected.fa
```

```
perl ~/Documents/perl_scripts/fasta2sto.pl aln4_corrected.fa >  
aln4_corrected.sto
```

Finally, you can execute hmmbuild to create a HMM using as input your MSA in stockholm format.

```
hmmbuild hmm_structural.hmm aln4_corrected.sto
```

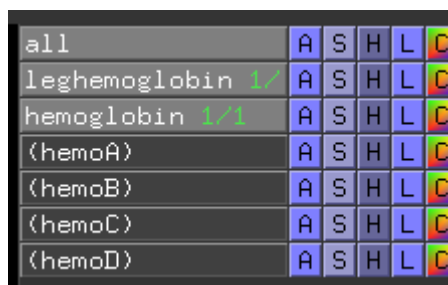

Step 4: Use superimposition to model protein-protein interactions

Homologous tend to share their protein-protein interactions as well as the conformation in which these interactions take place. This principle is at the basis of modeling protein-protein interactions using superimposition and the template of the interaction we want to model.

For this step we will work with legemoglobin, a vegetal globin, and the tetrameric form of human hemoglobin. Leghemoglobin and hemoglobin are homologous proteins, therefore we expect that leghemoglobin can also create tetramers. The problem is that we don't know how this leghemoglobin looks like. To solve this we can use superimposition.

We know that the conformation of tetrameric leghemoglobin should be similar to the tetrameric conformation of hemoglobin. Therefore, we can superimpose the leghemoglobin monomer on top of each of the hemoglobin monomers. This will create four different sets of coordinates for the leghemoglobin monomer. If we save as a new pdb each one of these new set of coordinates we will be able to reconstruct the tetrameric structure of leghemoglobin.

Start by opening a new session of pymol and loading the pdb structures for hemoglobin leghemoglobin that you have in the P3_directory. Then, create new pymol objects for each of the chains in the hemoglobin tetramer. We will call them hemo_A, hemo_B, hemo_C and hemo_D. Using the sequence display and the chain identifiers is a good way to select and create objects for each one of the chains.



all	A	S	H	L	C
leghemoglobin 1/	A	S	H	L	C
hemoglobin 1/1	A	S	H	L	C
(hemoA)	A	S	H	L	C
(hemoB)	A	S	H	L	C
(hemoC)	A	S	H	L	C
(hemoD)	A	S	H	L	C

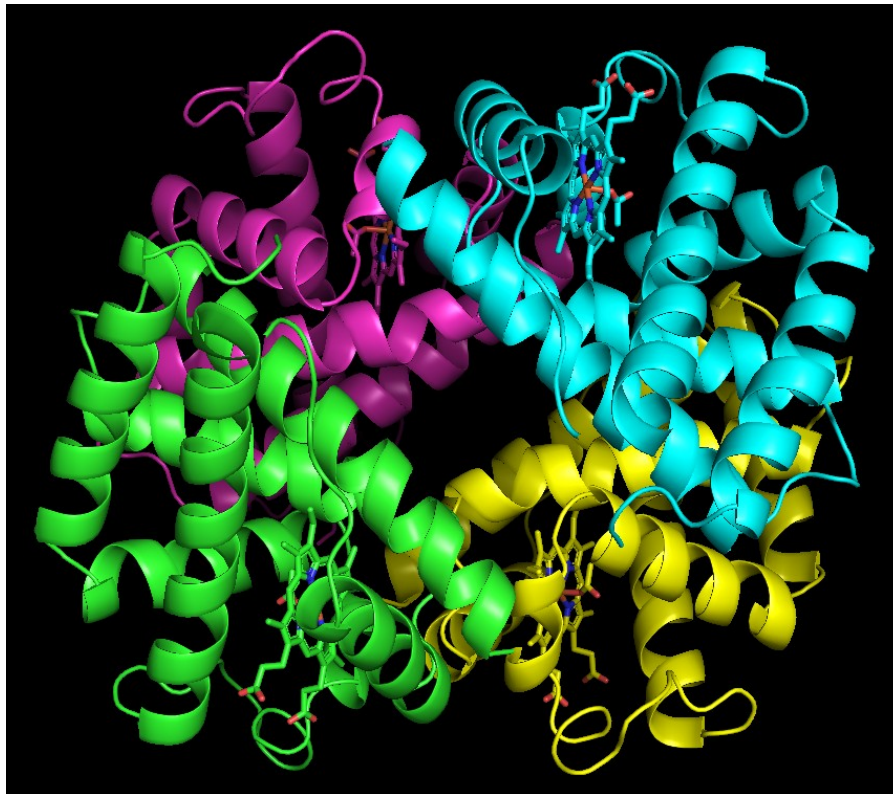
Now we will superimpose the leghemoglobin monomer on the chain A of the tetrameric hemoglobin:

super leghemoglobin, hemoA

Once done this, save the leghemoglobin monomer in this new set of coordinates as a new pdb file:

save leghemoglobin_A.pdb, leghemoglobin

Now, repeat this same procedure for hemoglobin chains B, C and D. Finally, if you open all the leghemoglobin pdb files you have generated you will be able to see the tetrameric form of leghemoglobin:



Step 5: Superimpose specific regions of two proteins

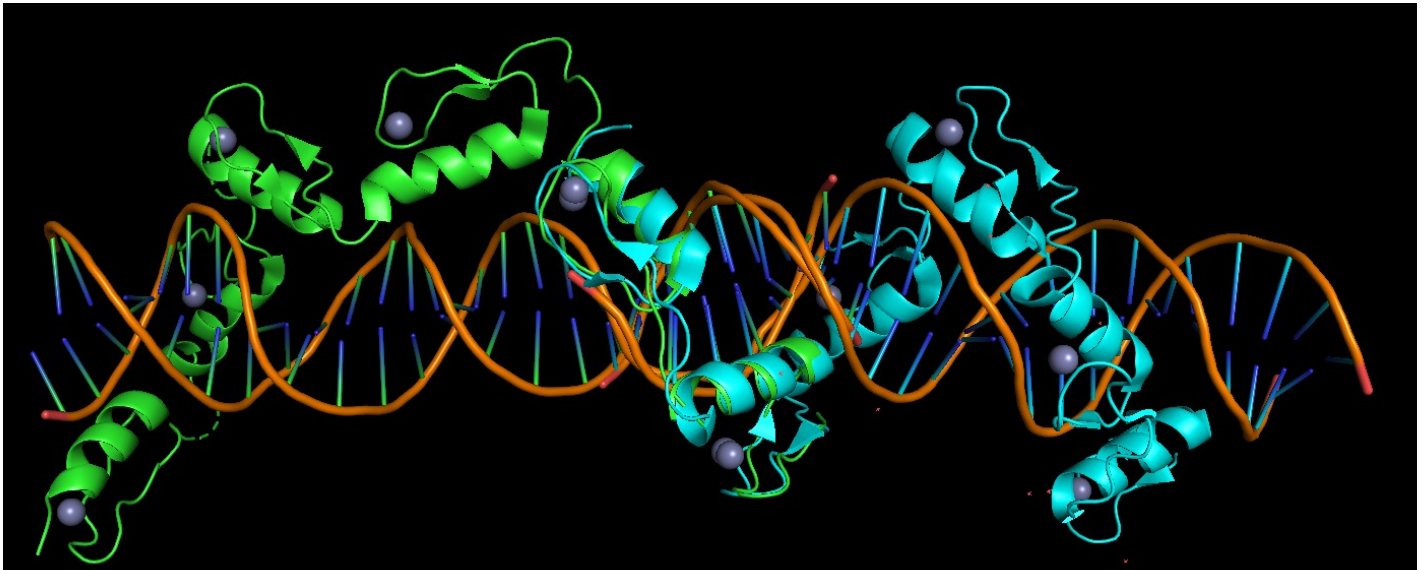
CTCF is a zinc finger transcription factor that plays a fundamental role in the regulation of gene expression. The DNA binding domain of CTCF is made of 11 zinc finger domains. Since this is a long DNA binding domain, there are no available structures for the entire DNA binding domain. However, there are structures that contain part of these DNA binding domains. By superimposing these structures on the parts that they overlap we may be able to reconstruct a model of the DNA binding site of CTCF.

Open the two structures in the CTCF directory inside the P3_directory (5t0u.pdb and 5yel.pdb). These files contain different parts of the DNA binding domain of CTCF. If you visualize the sequence of these proteins you will see that 5t0u goes from amino acid 293 to amino acid 461, while 5yel goes from amino acid 405 to amino acid 576.

We want to superimpose these two structures, but we have to be sure that we only superimpose the regions that are overlapping. To do this we have to specify what residues we want to include in the superimposition. Also, we will use the align command because the two structures that we want to superimpose belong to the same protein. Therefore, the amino acid sequences will be identical. Use the next command:

```
align 5t0u and resi 405-460, 5yel and resi 405-460
```

See that we are forcing the superimposition between amino acids 405 and 460. Since the two structures belong to the same protein, they match perfectly. See that we are superimposing also the DNA molecules that are in the structures.



Questions from the tutorial:

- 1) Get the sequences of the four compared monomeric globins (1ecd.pdb, 1lh1.pdb, 2lhb.pdb and 4mbn.pdb) and make a sequence alignment using clustalw. Then compare this alignment to the structure-based one that we obtained in the step 3 of this tutorial. Are they similar? Why is this happening?
- 2) Build a structure-based HMM using 4 templates for the sequence contained in the file "question2_target.fa". Can you use this HMM to search for homologous proteins of the target we are working with?
- 3) Build a structure-based HMM using 4 templates belonging to the globin protein family. Use the HMM from the PFAM database. Which are the differences between the obtained HMM and the ones in PFAM?
- 4) Try to identify the residues that are more relevant for the assembly of the lehmoglobin tetramer.
- 5) Try to build the tetrameric complex for some of the other globins in the tutorial (1ecd.pdb, 2lhb.pdb or 4mbn.pdb).
- 6) The same procedure we use to reconstruct protein-protein interactions via superimposition can be used to model protein-ligand interactions. In the protein-ligand directory you have two structures: A1_receptor.pdb and caffeine_receptor.pdb. The first structure contains an A1 adenosine receptor and the second contains an A2A adenosine receptor binding a caffeine molecule. Can you use superimposition to reconstruct the interaction between caffeine and the A1 adenosine receptor?
- 7) Use the commands align, super and cealign to superimpose the PDB structure 1bco with the chain B of structure 1c0m. What of the 3 commands is giving you better results? Why do you think this is happening?