Exercise SBI 3P: Structural similarity. Protein superimposition and structural alignments using chimera and STAMP.

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 have different sequences. Remember that protein structure remains longer than sequence, across the evolutionary process. They way to assess structural similarity between proteins is to perform protein superimpositions.

Exercise 3.1: Structural similarity. Protein superimposition and structural alignments using chimera.

By Alberto Meseguer
Supervised by Baldo Oliva
Last update: January 1st, 2018

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 it is defined by the next formula:

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

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:

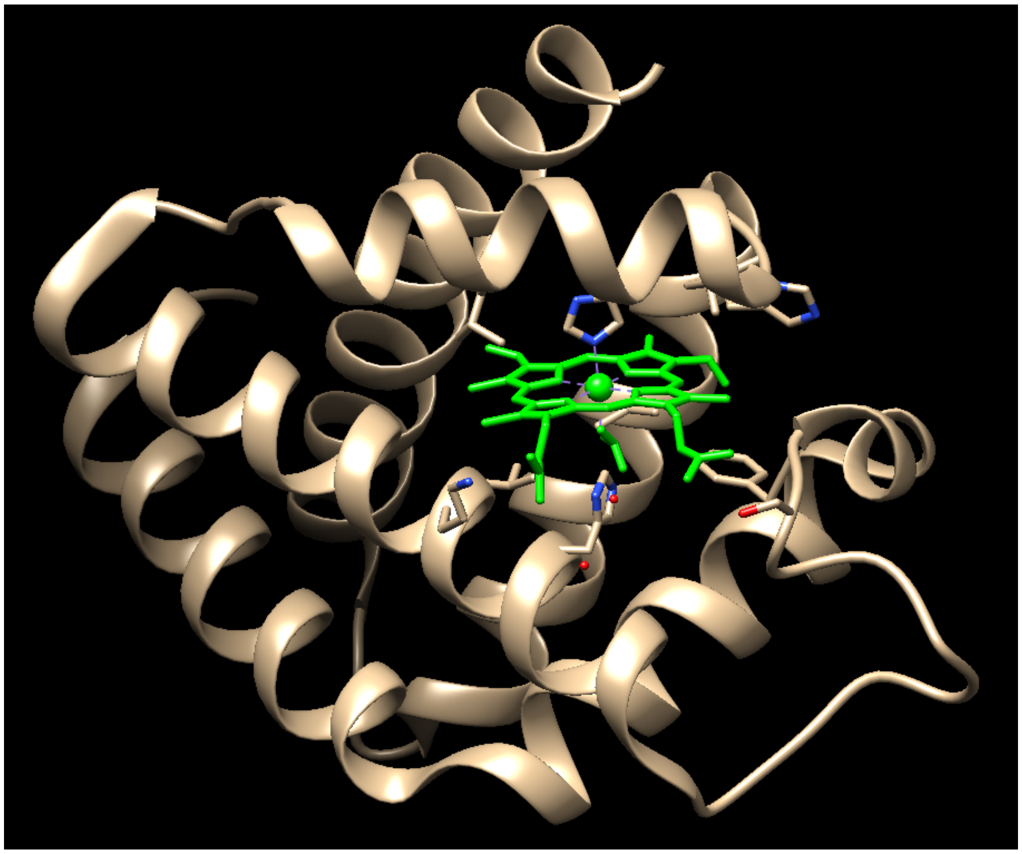
There are many programs that perform protein superimposition (XAM, STAMP, Tmalign,…). We are going to use chimera. Chimera is a program for protein structure visualization that can superimpose structures. Besides this, chimera has much more available tools, which makes it a very versatile program. I recommend you to explore the tutorial page of chimera: https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/frametut.html

Copy the files of "exercise_3" in your working directory, extract the folders and move to the "exercise_3" directory:

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 Chimera:

```
chimera 1lh1.pdb
```

This structure belongs to one globin monomer. In the following image you can see the protein with the hemo group highlighted in green:

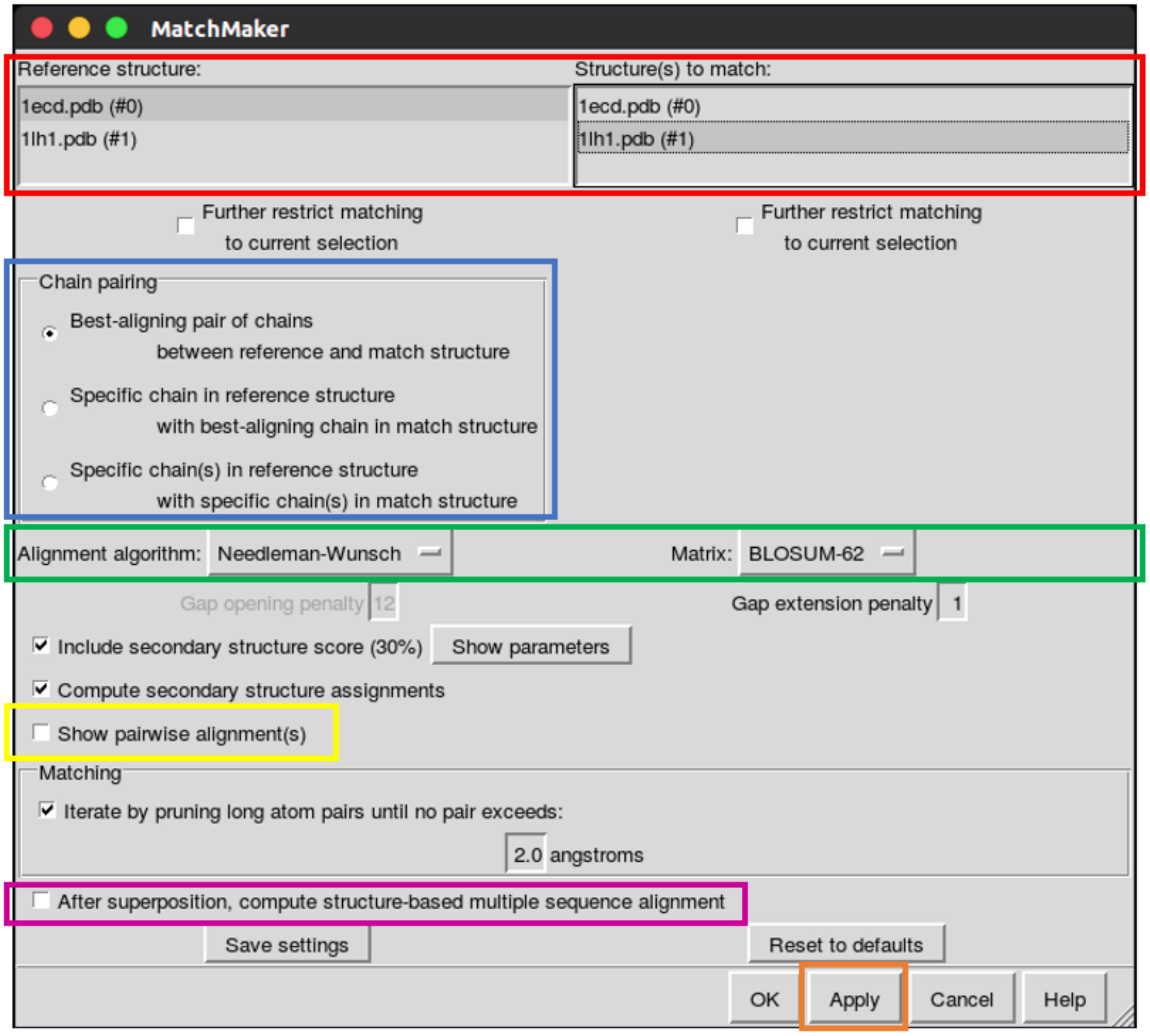


Step 1: Superimpose 2 structures

To superimpose 2 protein structures with chimera we will use the **"MatchMaker"** tool. We will start by opening two files with chimera:

```
chimera 1lh1.pdb 1ecd.pdb
```

We click on the **"Favorites"** and **"Reply Log"** buttons. This will make appear the reply log window. This window will display relevant data along the whole tutorial. Then, we click on the **"tools"**, **"Structure Comparison"** and **"Match Maker"** buttons. This will make appear the match maker window that looks as follows:



Reference structure:
1ecd.pdb (#0)
1lh1.pdb (#1)

Structure(s) to match:
1ecd.pdb (#0)
1lh1.pdb (#1)

☐ Further restrict matching to current selection

☐ Further restrict matching to current selection

Chain pairing
● Best-aligning pair of chains between reference and match structure
○ Specific chain in reference structure with best-aligning chain in match structure
○ Specific chain(s) in reference structure with specific chain(s) in match structure

Alignment algorithm: Needleman-Wunsch

Matrix: BLOSUM-62

Gap opening penalty 12

Gap extension penalty 1

☑ Include secondary structure score (30%)

☑ Compute secondary structure assignments

☐ Show pairwise alignment(s)

Matching
☑ Iterate by pruning long atom pairs until no pair exceeds: 2.0 angstroms

☐ After superposition, compute structure-based multiple sequence alignment

This window contains all the settings to perform the superimposition of the two structures. The most relevant ones are highlighted in colors:

- Reference structure and structures to match (red): you must select one reference structure and one or more structures to match. The latests will be rotated and translated to fit as best as possible the reference structure.

- Chain pairing (blue): if the proteins that we are superimposing have more than one chain we can select specific chains to be superimposed. We can select either chains from the reference or the match structures. In this example, the two proteins have one chain, so this option is not relevant.

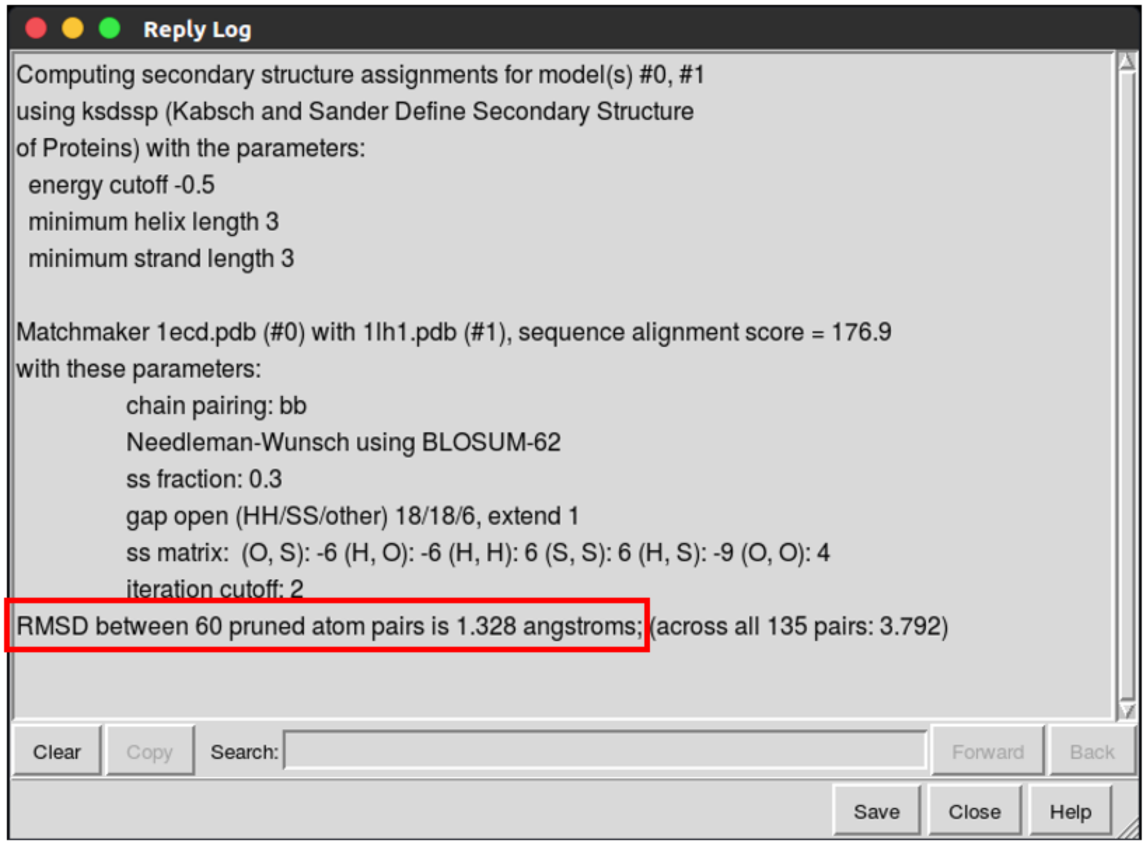
- Alignment algorithm and matrix (green): to perform protein superimposition, chimera takes into account both structural and sequence information. Therefore, it will try to match the structure of protein regions that have high sequence similarity. To assess sequence similarity it uses an alignment algorithm and a substitution matrix. Here you choose which alignment algorithm and substitution matrix you want to use. We will not change the default parameters during this tutorial.

- Show pairwise alignment (yellow): it shows the sequence based alignment that chimera has obtained using the alignment algorithm and the substitution matrix selected. We won't select this option yet.

- Compute structure-based MSA (purple): it shows the structure based alignment that chimera obtains after protein superimposition. We will explore this option in the following steps of the tutorial, by now we leave it unselected.

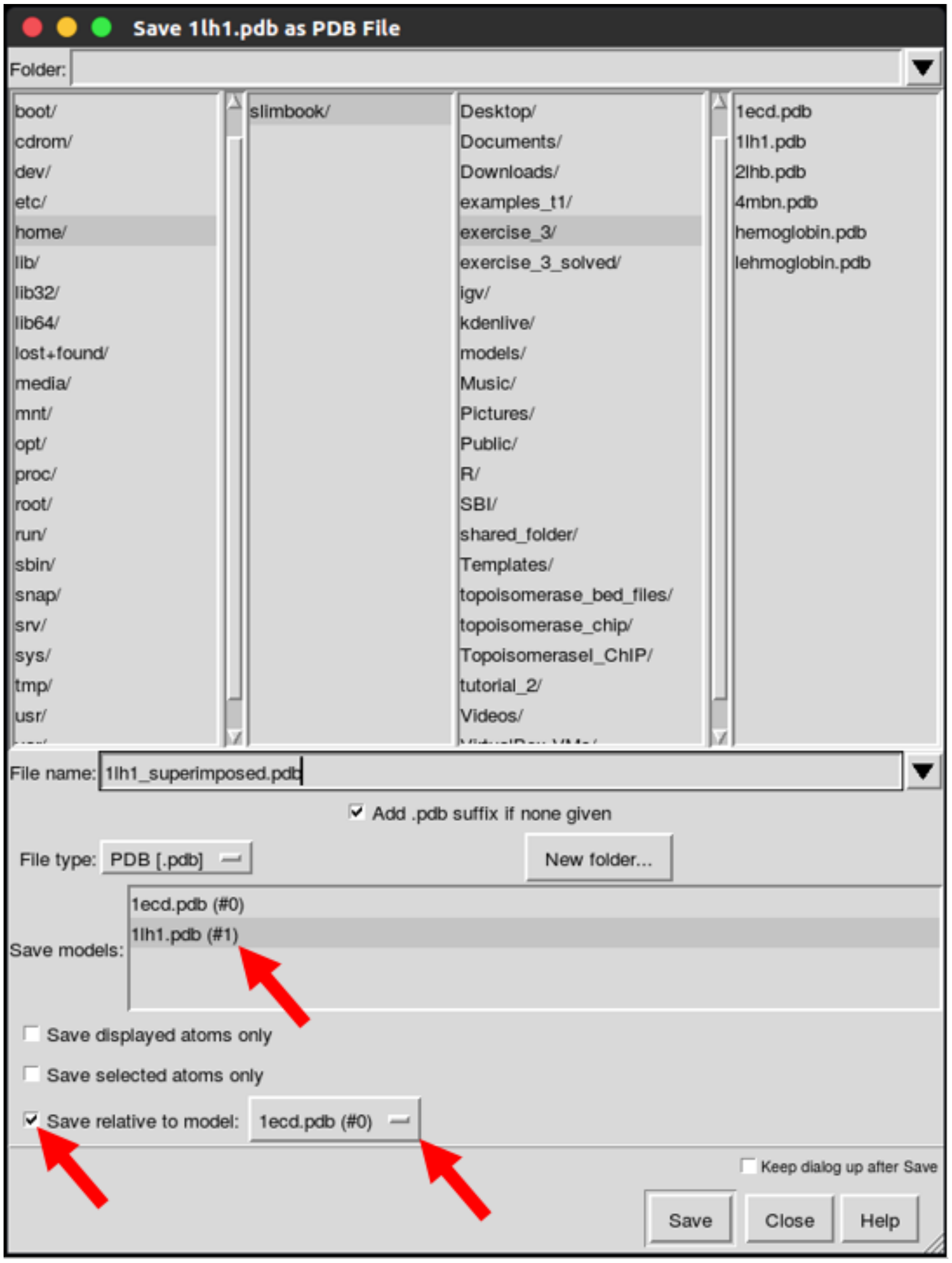
- Apply (orange): click this button to run the superimposition.

After applying the superimposition we see how the two structures fit quite well in space. To determine how different these two proteins are structurally, we will take a look to the **"Reply Log"** window we opened previously. We can see that it displays the information about the superimposition we have just performed:


```
Computing secondary structure assignments for model(s) #0, #1
using ksdssp (Kabsch and Sander Define Secondary Structure
of Proteins) with the parameters:
  energy cutoff -0.5
  minimum helix length 3
  minimum strand length 3

Matchmaker 1ecd.pdb (#0) with 1lh1.pdb (#1), sequence alignment score = 176.9
with these parameters:
        chain pairing: bb
        Needleman-Wunsch using BLOSUM-62
        ss fraction: 0.3
        gap open (HH/SS/other) 18/18/6, extend 1
        ss matrix:  (O, S): -6 (H, O): -6 (H, H): 6 (S, S): 6 (H, S): -9 (O, O): 4
        iteration cutoff: 2
RMSD between 60 pruned atom pairs is 1.328 angstroms; (across all 135 pairs: 3.792)
```

From all this information we are particularly interested in the RMSD (highlighted in red). The RMSD will tell us how different are these two proteins structurally.

Once the superimposition is over, we may want to obtain a new pdb that contains the superimposed proteins in their new set of coordinates. We can do that with chimera clicking in the **"File"** and **"Save PDB"** buttons. This displays the save PDB window. In order to save it is important to select the matched structure in the "save models" section and the "Save relative to model" option with the reference structure. These requirements are highlighted with red arrows in the next image:



Save 1lh1.pdb as PDB File

Folder:

boot/	slimbook/	Desktop/	1ecd.pdb
cdrom/		Documents/	1lh1.pdb
dev/		Downloads/	2lhb.pdb
etc/		examples_t1/	4mbn.pdb
home/		exercise_3/	hemoglobin.pdb
lib/		exercise_3_solved/	lehmoglobin.pdb
lib32/		igv/	
lib64/		kdenlive/	
lost+found/		models/	
media/		Music/	
mnt/		Pictures/	
opt/		Public/	
proc/		R/	
root/		SBI/	
run/		shared_folder/	
sbin/		Templates/	
snap/		topoisomerase_bed_files/	
srv/		topoisomerase_chip/	
sys/		TopoisomeraseI_ChIP/	
tmp/		tutorial_2/	
usr/		Videos/	

File name: 1lh1_superimposed.pdb

☑ Add .pdb suffix if none given

File type: PDB [.pdb]

Save models:
1ecd.pdb (#0)
1lh1.pdb (#1)

☐ Save displayed atoms only

☐ Save selected atoms only

☑ Save relative to model: 1ecd.pdb (#0)

☐ Keep dialog up after Save

Since we have superimposed only two proteins and one of them was the reference structure, only one protein has changed its coordinates during the superimposition. That is why we will only save one pdb, the one corresponding to the matching structure (1lh1.pdb).

Step 2: Superimpose many structures

To superimpose many structures we will use the same chimera tool as before with slight changes. Start by opening the pdb files with chimera:

```
chimera 1lh1.pdb 1ecd.pdb 2lhb.pdb 4mbn.pdb
```

Again, we click on the **"Favorites"** and **"Reply Log"** buttons to display the **"Reply Log"** window. We also click on the **"tools"**, **"Structure Comparison"** and **"Match Maker"** buttons.

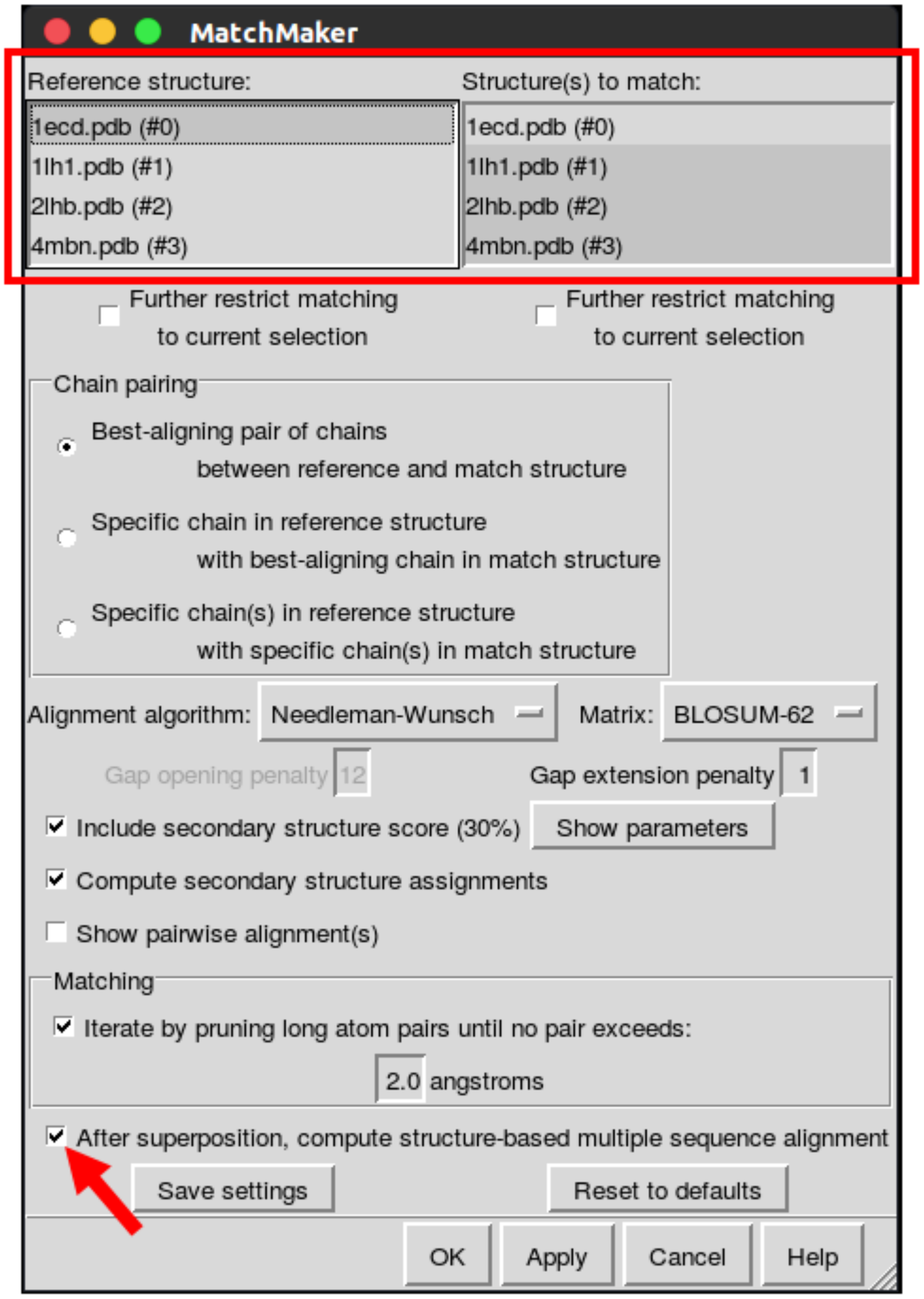
Now, we select only one protein as reference structure. Then, we select all the other pdbs as structures to match. Finally we click on apply to perform the superimposition.

Now, we can check the RMSD of each of the superimposed proteins to the reference one by looking at the "reply log" window. We can also save the superimposed PDBs by using the **"File"** and **"Save PDB"** buttons, as we saw previously.

Don't close the chimera session, the next step will start from this superimposition of several globins.

Step 3: Build a structure based HMM

We may want to represent the structural similarity between these superimposed globins as a HMM. To do so, we need to obtain a structure-based MSA. This can be done with chimera selecting the **"Compute structure-based MSA"** option in the **"Match Maker"** window. So, we open again the "Match Maker" window, we select this option and we click on apply:


MatchMaker

Reference structure:
1ecd.pdb (#0)
1lh1.pdb (#1)
2lhb.pdb (#2)
4mbn.pdb (#3)

Structure(s) to match:
1ecd.pdb (#0)
1lh1.pdb (#1)
2lhb.pdb (#2)
4mbn.pdb (#3)

☐ Further restrict matching to current selection
☐ Further restrict matching to current selection

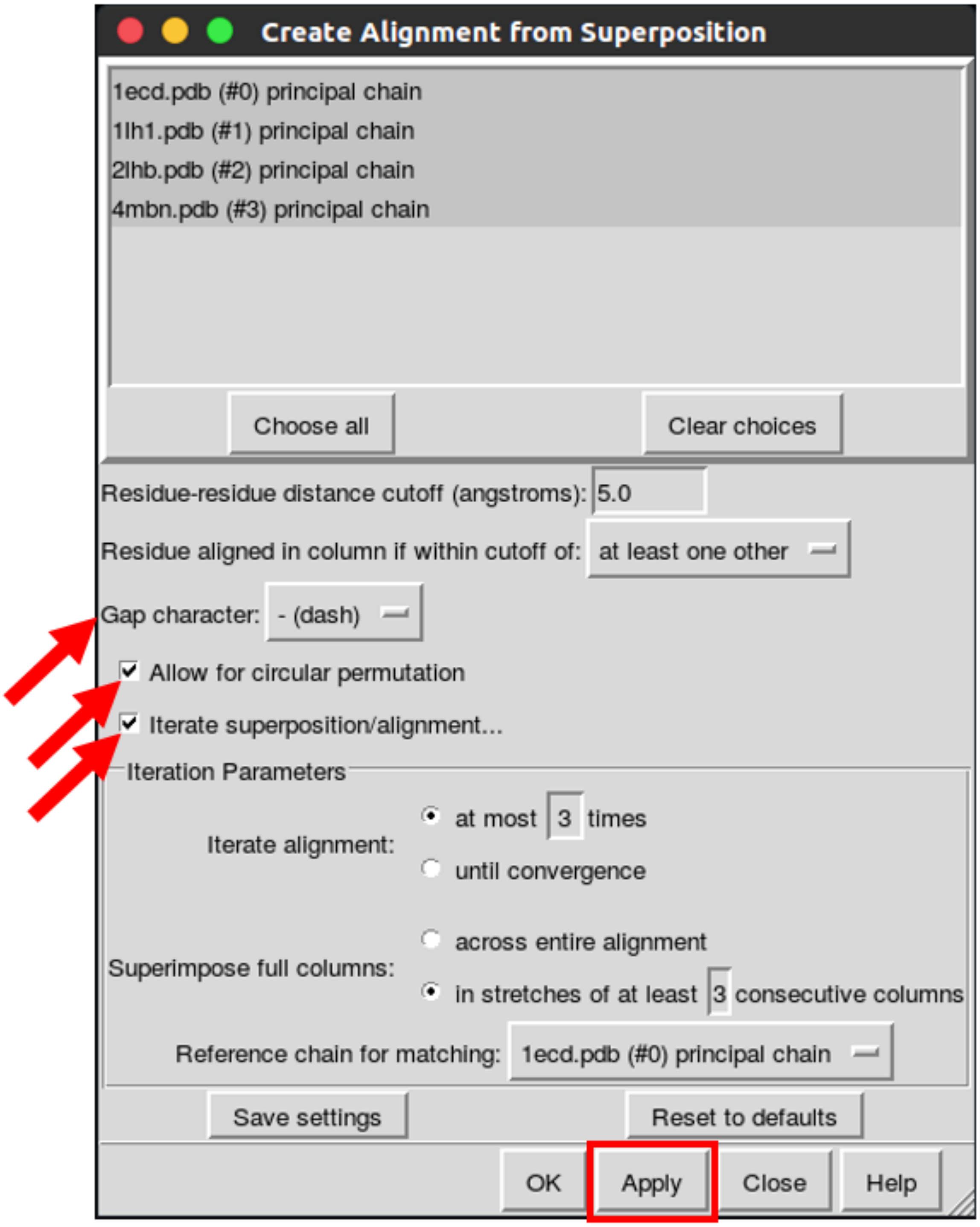
Chain pairing
● Best-aligning pair of chains between reference and match structure
○ Specific chain in reference structure with best-aligning chain in match structure
○ Specific chain(s) in reference structure with specific chain(s) in match structure

Alignment algorithm: Needleman-Wunsch
Matrix: BLOSUM-62
Gap opening penalty 12
Gap extension penalty 1
☑ Include secondary structure score (30%)
☑ Compute secondary structure assignments
☐ Show pairwise alignment(s)

Matching
☑ Iterate by pruning long atom pairs until no pair exceeds: 2.0 angstroms

☑ After superposition, compute structure-based multiple sequence alignment

This will display the **"Create alignment from superimposition"** window. On this window we set the gap character as a dash and we click in the **"allow circular permutation"** and **"iterate superimposition/alignment"** buttons (highlighted in red). Then we click on apply:



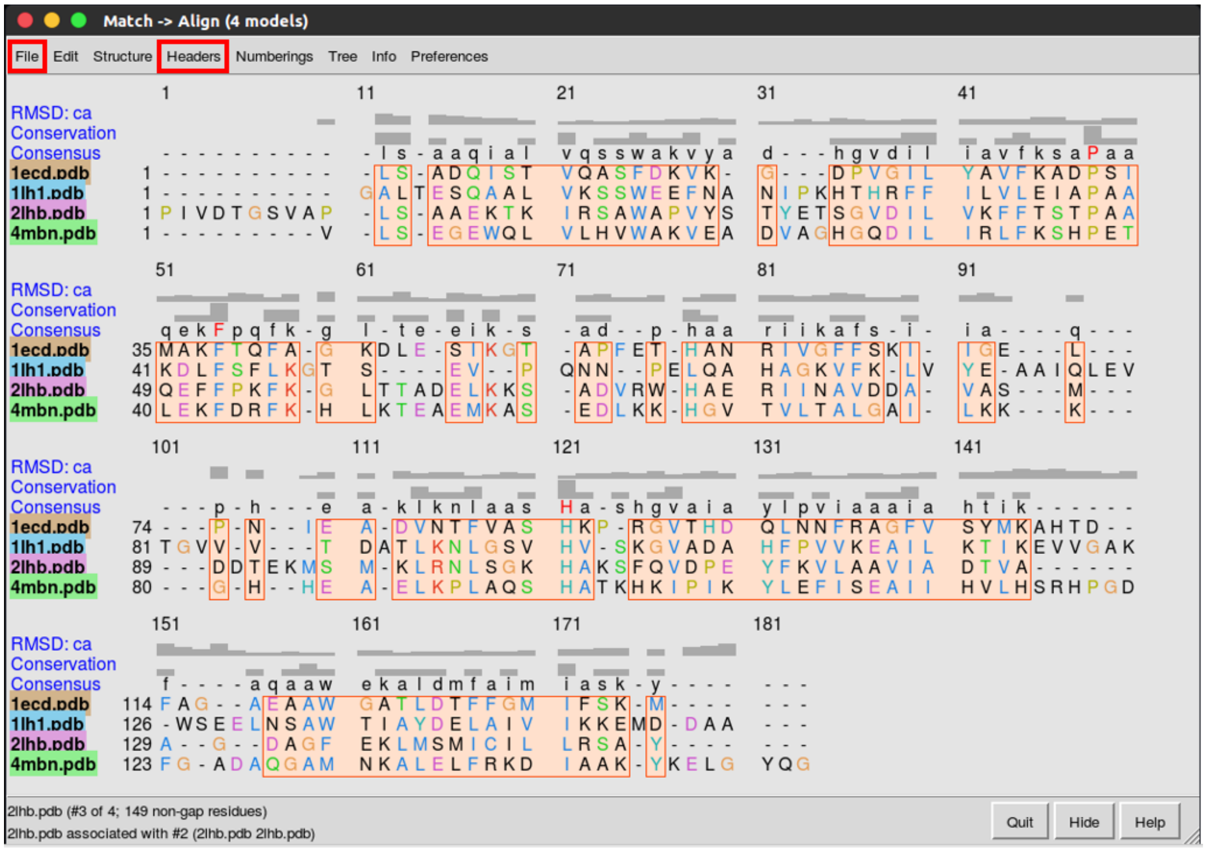
Create Alignment from Superposition

1ecd.pdb (#0) principal chain
1lh1.pdb (#1) principal chain
2lhb.pdb (#2) principal chain
4mbn.pdb (#3) principal chain

Residue-residue distance cutoff (angstroms): 5.0
Residue aligned in column if within cutoff of: at least one other
Gap character: - (dash)
[✓] Allow for circular permutation
[✓] Iterate superposition/alignment...

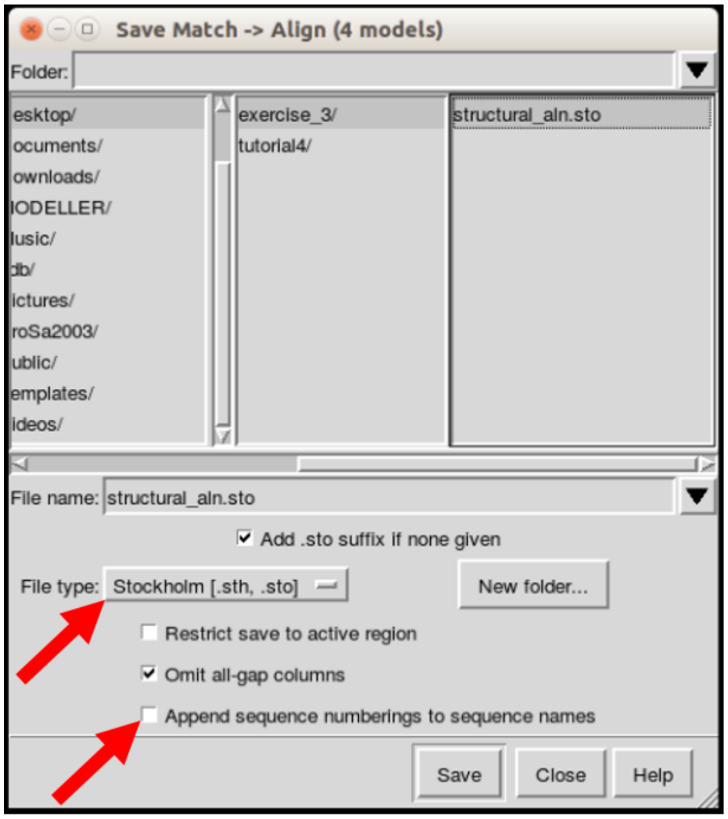
Iteration Parameters
Iterate alignment: [● at most 3 times] until convergence
Superimpose full columns: across entire alignment [● in stretches of at least 3 consecutive columns]
Reference chain for matching: 1ecd.pdb (#0) principal chain

Once the alignment has been computed, a window with the structural alignment will appear. Besides the aligned sequences, this alignment can display other variables such as the RMSD, the degree of conservation or the consensus sequence, among others. To show/hide these extra variables, click in the alignment window on the **"headers"** button and then select the variables you want.



```
                 1                     11                    21                    31                    41
RMSD: ca
Conservation
Consensus           - - - - - - - - - -   - l s - a a q i a l   v q s s w a k v y a   d - - - h g v d i l   i a v f k s a P a a
1ecd.pdb         1  - - - - - - - - - -   - L S - A D Q I S T   V Q A S F D K V K -   G - - - D P V G I L   Y A V F K A D P S I
1lh1.pdb         1  - - - - - - - - - -   G A L T E S Q A A L   V K S S W E E F N A   N I P K H T H R F F   I L V L E I A P A A
2lhb.pdb         1  P I V D T G S V A P   - L S - A A E K T K   I R S A W A P V Y S   T Y E T S G V D I L   V K F F T S T P A A
4mbn.pdb         1  - - - - - - - - - V   - L S - E G E W Q L   V L H V W A K V E A   D V A G H G Q D I L   I R L F K S H P E T

                 51                    61                    71                    81                    91
RMSD: ca
Conservation
Consensus           q e k F p q f k - g   l - t e - e i k - s   - a d - - p - h a a   r i i k a f s - i -   i a - - - - q - - -
1ecd.pdb        35  M A K F T Q F A - G   K D L E - S I K G T   - A P F E T - H A N   R I V G F F S K I -   I G E - - - L - - -
1lh1.pdb        41  K D L F S F L K G T   S - - - - - E V - -   Q N N - - P E L Q A   H A G K V F K - L V   Y E - A A I Q L E V
2lhb.pdb        49  Q E F F P K F K - G   L T T A D E L K K S   - A D V R W - H A E   R I I N A V D D A -   V A S - - - M - - -
4mbn.pdb        40  L E K F D R F K - H   L K T E A E M K A S   - E D L K K - H G V   T V L T A L G A I -   L K K - - - K - - -

                 101                   111                   121                   131                   141
RMSD: ca
Conservation
Consensus           - - - p - h - - - e   a - k l k n l a a s   H a - s h g v a i a   y l p v i a a a i a   h t i k - - - - - -
1ecd.pdb        74  - - - P - N - - I E   A - D V N T F V A S   H K P - R G V T H D   Q L N N F R A G F V   S Y M K A H T D - -
1lh1.pdb        81  T G V V - V - - - T   D A T L K N L G S V   H V - S K G V A D A   H F P V V K E A I L   K T I K E V V G A K
2lhb.pdb        89  - - - D D T E K M S   M - K L R N L S G K   H A K S F Q V D P E   Y F K V L A A V I A   D T V A - - - - - -
4mbn.pdb        80  - - - G - H - - H E   A - E L K P L A Q S   H A T K H K I P I K   Y L E F I S E A I I   H V L H S R H P G D

                 151                   161                   171                   181
RMSD: ca
Conservation
Consensus           f - - - - a q a a w   e k a l d m f a i m   i a s k - y - - - -   - - -
1ecd.pdb       114  F A G - - A E A A W   G A T L D T F F G M   I F S K - M - - - -   - - -
1lh1.pdb       126  - W S E E L N S A W   T I A Y D E L A I V   I K K E M D - D A A   - - -
2lhb.pdb       129  A - - G - - D A G F   E K L M S M I C I L   L R S A - Y - - - -   - - -
4mbn.pdb       123  F G - A D A Q G A M   N K A L E L F R K D   I A A K - Y K E L G   Y Q G

2lhb.pdb (#3 of 4; 149 non-gap residues)
2lhb.pdb associated with #2 (2lhb.pdb 2lhb.pdb)
```

However, we cannot use this alignment to build a HMM yet. To do so we will have to save this alignment using a stockholm format. This can be done clicking the “File” and “Save as” buttons

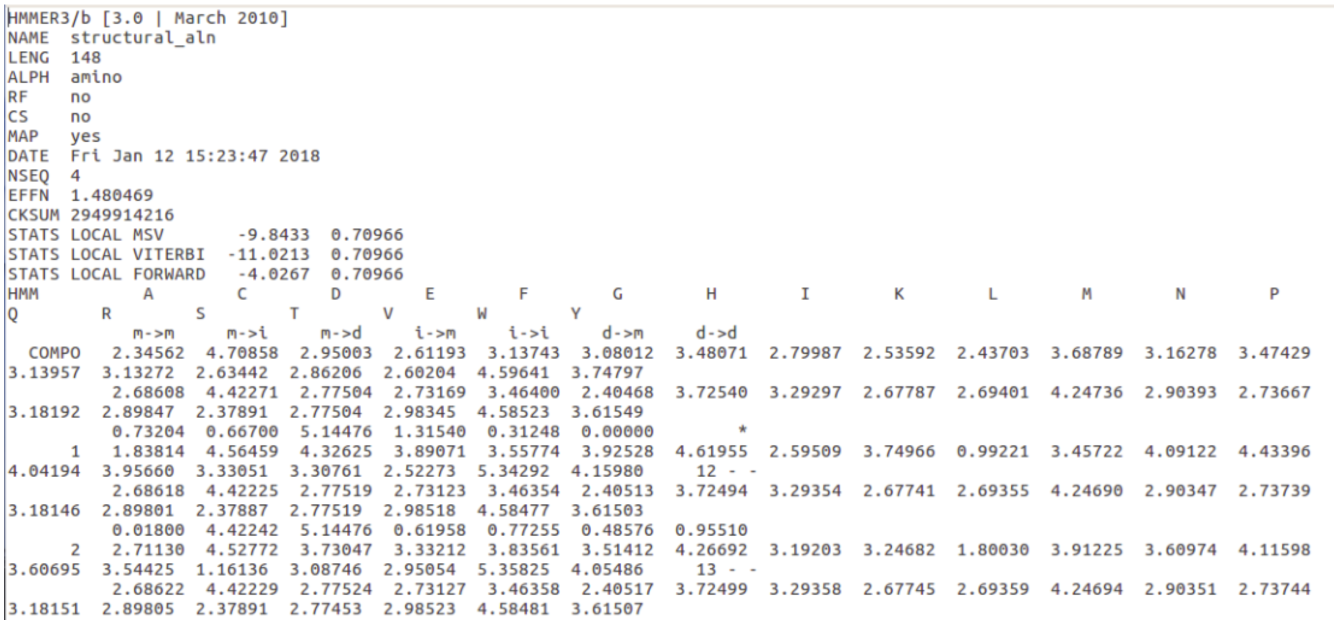


```
Folder:
esktop/          exercise_3/     structural_aln.sto
ocuments/        tutorial4/
ownloads/
IODELLER/
lusic/
db/
ictures/
roSa2003/
ublic/
emplates/
ideos/

File name: structural_aln.sto
[✓] Add .sto suffix if none given
File type: Stockholm [.sth, .sto]
[ ] Restrict save to active region
[✓] Omit all-gap columns
[ ] Append sequence numberings to sequence names
```

in the alignment window. Be sure that you select the “Stockholm” option in the file type field and disable the “Append sequence numberings to sequence names”.
Once we have an alignment in stockholm format, we can build a HMM with hmmbuild using the alignment as input (as we saw in exercise 2.2).

```
hmmbuild structural_aln.hmm structural_aln.sto
```

Once we have this HMM we can use it to identify homologous proteins or align sequences, according to the structural data of this set of globins. Here you can see the first lines of our structure-based HMM:



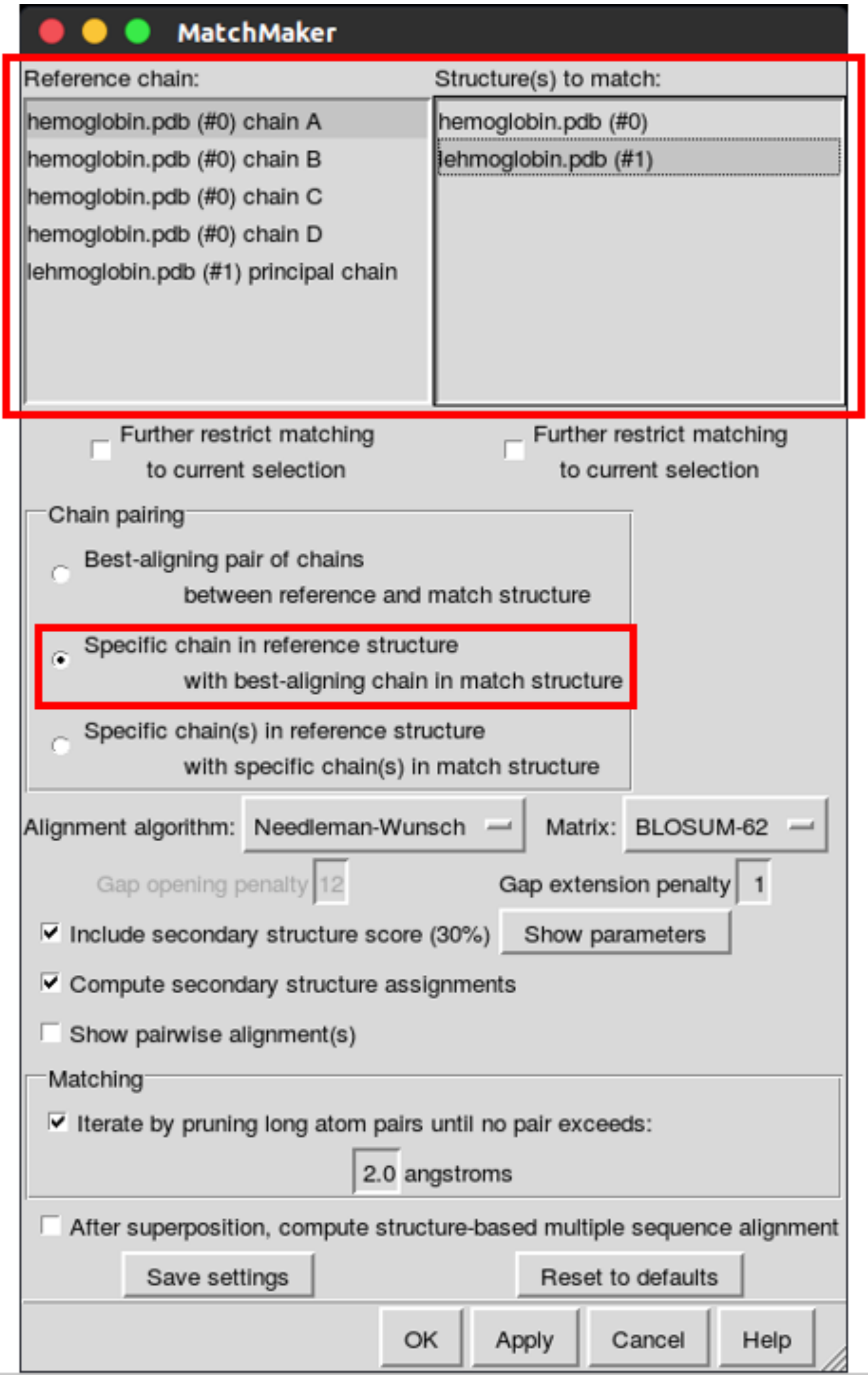
```
HMMER3/b [3.0 | March 2010]
NAME  structural_aln
LENG  148
ALPH  amino
RF    no
CS    no
MAP   yes
DATE  Fri Jan 12 15:23:47 2018
NSEQ  4
EFFN  1.480469
CKSUM 2949914216
STATS LOCAL MSV       -9.8433  0.70966
STATS LOCAL VITERBI  -11.0213  0.70966
STATS LOCAL FORWARD   -4.0267  0.70966
HMM          A        C        D        E        F        G        H        I        K        L        M        N        P
Q        R        S        T        V        W        Y
            m->m     m->i     m->d     i->m     i->i     d->m     d->d
  COMPO   2.34562  4.70858  2.95003  2.61193  3.13743  3.08012  3.48071  2.79987  2.53592  2.43703  3.68789  3.16278  3.47429
3.13957  3.13272  2.63442  2.86206  2.60204  4.59641  3.74797
          2.68608  4.42271  2.77504  2.73169  3.46400  2.40468  3.72540  3.29297  2.67787  2.69401  4.24736  2.90393  2.73667
3.18192  2.89847  2.37891  2.77504  2.98345  4.58523  3.61549
          0.73204  0.66700  5.14476  1.31540  0.31248  0.00000        *
      1   1.83814  4.56459  4.32625  3.89071  3.55774  3.92528  4.61955  2.59509  3.74966  0.99221  3.45722  4.09122  4.43396
4.04194  3.95660  3.33051  3.30761  2.52273  5.34292  4.15980      12 - -
          2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494  3.29354  2.67741  2.69355  4.24690  2.90347  2.73739
3.18146  2.89801  2.37887  2.77519  2.98518  4.58477  3.61503
          0.01800  4.42242  5.14476  0.61958  0.77255  0.48576  0.95510
      2   2.71130  4.52772  3.73047  3.33212  3.83561  3.51412  4.26692  3.19203  3.24682  1.80030  3.91225  3.60974  4.11598
3.60695  3.54425  1.16136  3.08746  2.95054  5.35825  4.05486      13 - -
          2.68622  4.42229  2.77524  2.73127  3.46358  2.40517  3.72499  3.29358  2.67745  2.69359  4.24694  2.90351  2.73744
3.18151  2.89805  2.37891  2.77453  2.98523  4.58481  3.61507
```

Step 4: Use protein superimposition to reconstruct protein-protein interactions

We propose the following problem: we are studying the structure of the leghemoglobin, a vegetal globin. We know that human hemoglobin, a remote homolog, is able to build homotetramers. Since homologous proteins tend to share the same protein-protein interactions, we expect the leghemoglobin to build homotetramers too. We want to study the tetrameric form of the leghemoglobin, but there is no available structure of the complex, only of the monomer. Nevertheless, we have the structure of the tetrameric complex of the human hemoglobin. Now we will learn how to use the tetrameric structure of human hemoglobin to reconstruct the tetrameric structure of vegetal leghemoglobin. The structure of tetrameric hemoglobin from human is **2hhb.pdb**, and for leghemoglobin is **1bin.pdb**. They are in the folder named hemoglobin.pdb and leghemoglobin.pdb. We start by opening both structures with chimera:

```
chimera hemoglobin.pdb leghemoglobin.pdb
```

Then, we open the **"Match Maker"** window by clicking on the **"tools"**, **"Structure Comparison"** and **"Match Maker"** buttons. On the **"Match Maker"** window we select the option **"Specific chain in reference structure with best aligning chain in match structure"** in the **"chain pairing"** section:


MatchMaker

Reference chain:
- hemoglobin.pdb (#0) chain A
- hemoglobin.pdb (#0) chain B
- hemoglobin.pdb (#0) chain C
- hemoglobin.pdb (#0) chain D
- lehmoglobin.pdb (#1) principal chain

Structure(s) to match:
- hemoglobin.pdb (#0)
- lehmoglobin.pdb (#1)

☐ Further restrict matching to current selection
☐ Further restrict matching to current selection

Chain pairing
- ○ Best-aligning pair of chains between reference and match structure
- ● Specific chain in reference structure with best-aligning chain in match structure
- ○ Specific chain(s) in reference structure with specific chain(s) in match structure

Alignment algorithm: Needleman-Wunsch   Matrix: BLOSUM-62

Gap opening penalty 12   Gap extension penalty 1

☑ Include secondary structure score (30%)   Show parameters

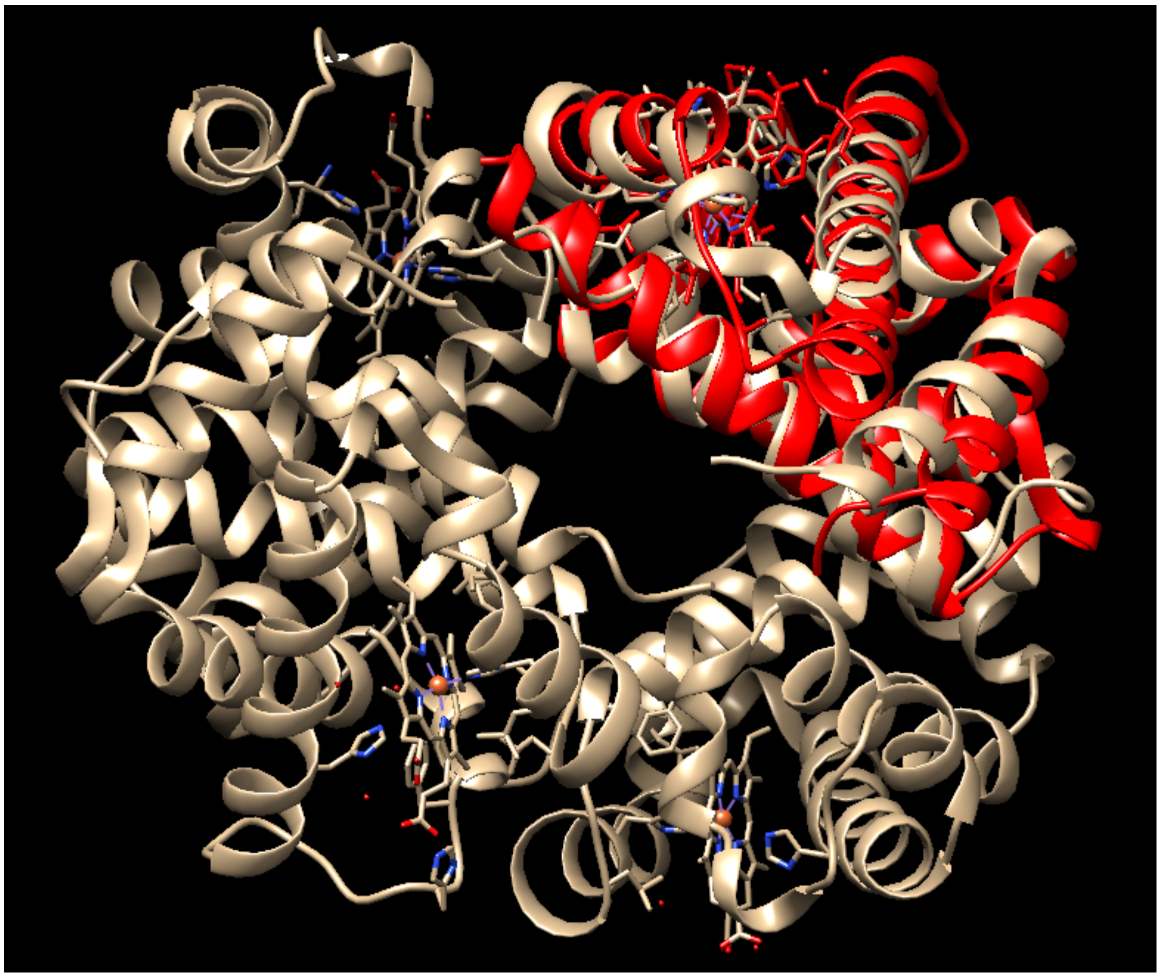
☑ Compute secondary structure assignments

☐ Show pairwise alignment(s)

Matching
☑ Iterate by pruning long atom pairs until no pair exceeds: 2.0 angstroms

☐ After superposition, compute structure-based multiple sequence alignment

Now, we select as reference structure the chain A of the human hemoglobin, and as match structure the monomeric leghemoglobin. After doing so, we will see how the leghemoglobin (in red) is superimposed with only one of the four chains of the human hemoglobin (in beige):



Afterwards, we click the buttons **"File"** and **"Save PDB"**; and we save the superimposed leghemoglobin as leghemoglobin_A.pdb (remember that we saw how to save superimposed structures in the step 1). We repeat this same operation for the other chains in the hemoglobin complex. So, we superimpose the leghemoglobin setting the chain B of hemoglobin as reference structure and save that PDB as leghemoglobin_B.pdb. And we do the same for chains C and D.

At this point of the tutorial we have created four files: leghemoglobin_A, B, C, and D.pdb. Now we will concatenate all of them into the same file by using the following commands:

```
cat leghemoglobin_A.pdb > tetrameric_leghemoglobin.pdb
```

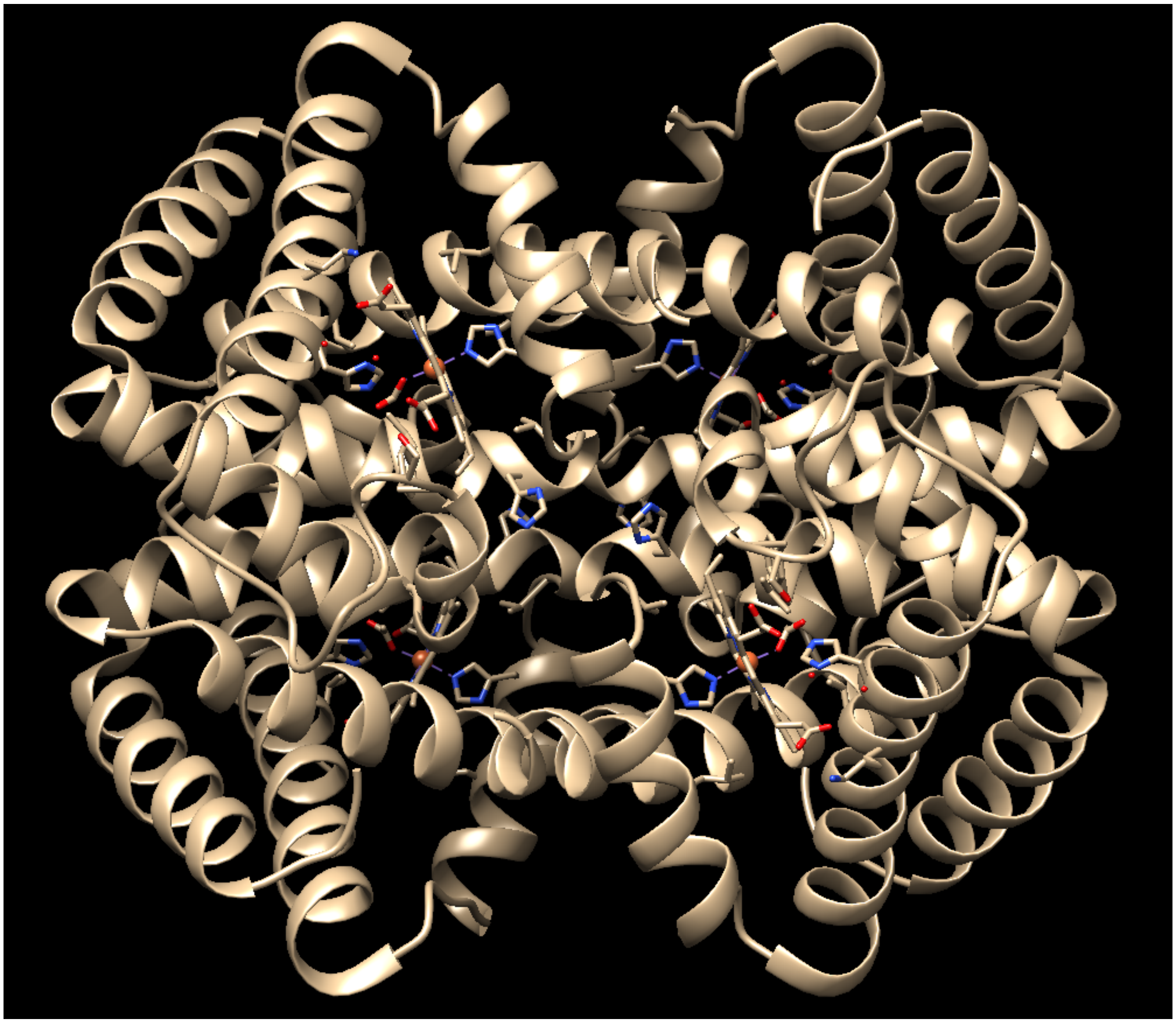
```
cat leghemoglobin_B.pdb >> tetrameric_leghemoglobin.pdb
```

```
cat leghemoglobin_C.pdb >> tetrameric_leghemoglobin.pdb
cat leghemoglobin_D.pdb >> tetrameric_leghemoglobin.pdb
```

Now, the tetrameric_leghemoglobin.pdb file will contain leghemoglobin monomers fitting the positions of each one of the subunits in the tetrameric hemoglobin. We can study the obtained tetramer by using chimera:

```
chimera tetrameric_leghemoglobin.pdb
```

The examination of this tetramer can help to assess its stability or to identify relevant residues for its assembly. Here you can see an image of the leghemoglobin tetramer:



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 leghemoglobin tetramer.

5) Try to build the tetrameric complex for other globins in the tutorial (1ecd.pdb, 2lhb.pdb and 4mbn.pdb).

Exercise 3.2: Structural similarity. Protein superimposition and structural alignments using STAMP.

By Nuria B. Centeno.
Modified by Baldo Oliva
Last update: January 1st, 2018

Overview

STAMP is a package for the alignment of protein sequence based on three-dimensional structure. It automatically provides multiple alignments and the corresponding best-fit superimpositions.

To this end, the program build a matrix containing a numerical measure of the pairwise similarity of each position in one sequence to each position in another sequence, and then it uses an algorithm to find the path that maximizes the similarity.

Without going into details, to measure the pairwise similarity a probability function is used, consisting of Ca distance and a local main chain conformation terms. STAMP performs multiple structural alignments by following a systematically derived hierarchy of structural similarity. The program first compares all possible pairs of structures, and then this is used to produces a dendogram, or tree, from which multiple alignments and superimposition may be generated. Besides RMS deviation, STAMP provides a structural similarity score (Sc) to measure the confidence on the quality alignment:

- Alignments having a Sc between 5.5 and 9.8 imply a high degree of structural similarity and almost always suggest a functional and/or evolutionary relationship.
- Values between 2.5 and 5.5 correspond to more distantly related structures, and do not always imply a functional or evolutionary relationship.
- Values less than 2.0 generally indicate little overall structural similarity.

Getting started

In this part of the practice we will use an example of 6 globins to
learn how to use STAMP

1. Create a working directory and get the pdb files of molecules to be compared

```
$ mkdir working-directory
$ cd working-directory
Copy globin.tar in the working directory
$ tar -xvf globin.tar
```

These commands will create a subdirectory in our working directory called globin, which contains 5 pdb files plus a file called globin.domains (the basic STAMP input file).

In this exercise we will analyze the structural similarity of a set of globins: hemoglobins from human (two chains), lamprey and chironomus fly; a myoglobin from sperm whale, and a leghemoglobin from yellow bush lupine.

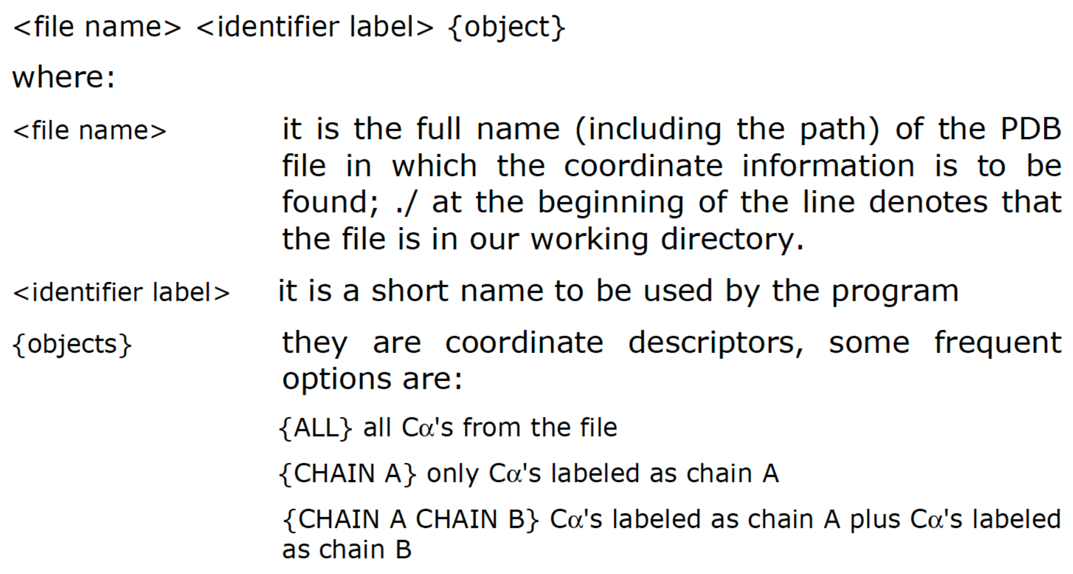
Using STAMP

Input file generation

The input file contains the information needed by STAMP about the proteins to be aligned. It is advisable to use the extension *.domains*

The file has as many lines as proteins we want to align, plus a blank line at the end. The format of each line is:

<file name> <identifier label> {object}



<file name> <identifier label> {object}

where:

<file name>	it is the full name (including the path) of the PDB file in which the coordinate information is to be found; ./ at the beginning of the line denotes that the file is in our working directory.
<identifier label>	it is a short name to be used by the program
{objects}	they are coordinate descriptors, some frequent options are: {ALL} all Cα's from the file; {CHAIN A} only Cα's labeled as chain A; {CHAIN A CHAIN B} Cα's labeled as chain A plus Cα's labeled as chain B

Our file globin.domains looks like:

```
./pdb2hhb.ent 2hhba {CHAIN A}
./pdb2hhb.ent 2hhbb {CHAIN B}
./pdb1lh1.ent 1lh1 {ALL}
./pdb2lhb.ent 2lhb {ALL}
./pdb4mbn.ent 4mbn {ALL}
./pdb1ecd.ent 1ecd {ALL}
```

Structural alignment using an initial rough superimposition: ROUGHFIT

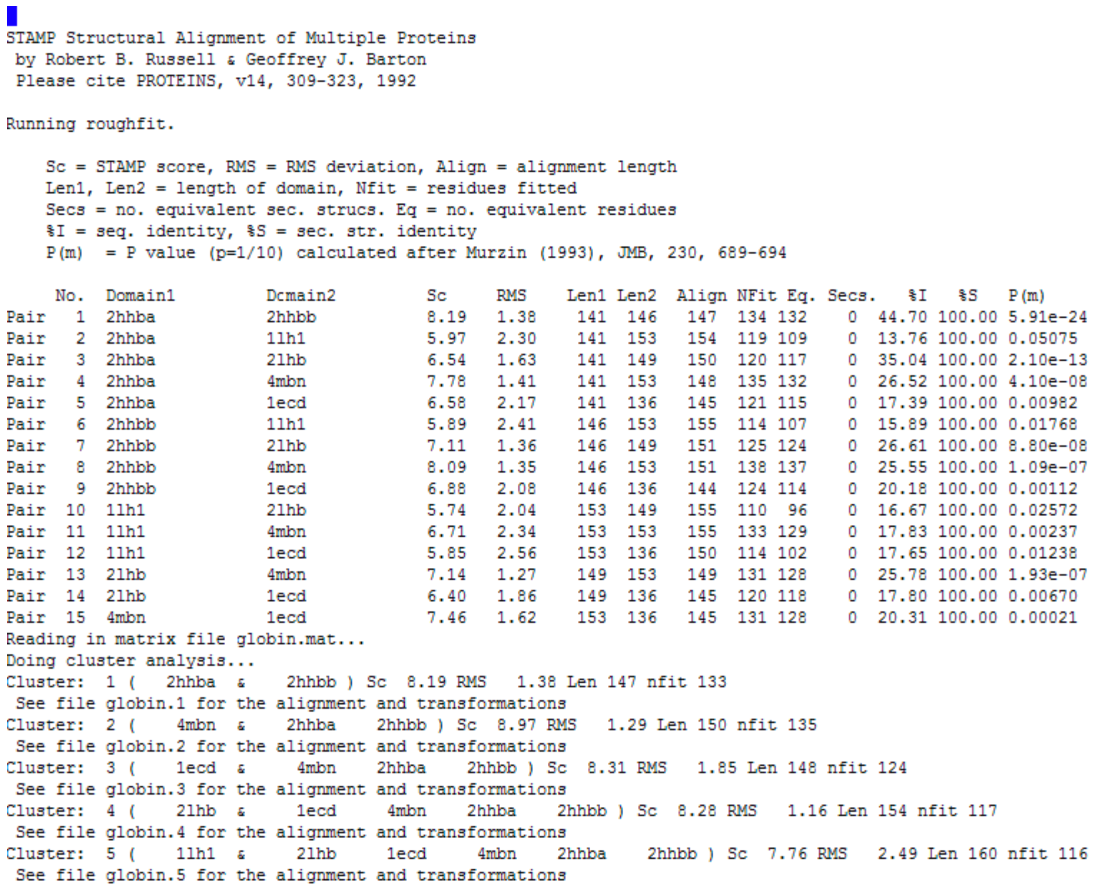
This is the simplest way to use STAMP, without any initial information but the coordinates of the structures to be compared. It tends to work for homologous proteins, or those having very similar lengths despite no sequence similarity.

To run STAMP in this example, type

```
$ stamp –l globin.domains –rough –n 2 –prefix globin > globin.out
```

-rough performs the initial superimposition and -n 2 means that an initial conformation based fit is performed in order to correct any inaccuracies in the initial superimposition.

The command should produce the following standard output:



```
STAMP Structural Alignment of Multiple Proteins
 by Robert B. Russell & Geoffrey J. Barton
 Please cite PROTEINS, v14, 309-323, 1992

Running roughfit.

    Sc = STAMP score, RMS = RMS deviation, Align = alignment length
    Len1, Len2 = length of domain, Nfit = residues fitted
    Secs = no. equivalent sec. strucs. Eq = no. equivalent residues
    %I = seq. identity, %S = sec. str. identity
    P(m)  = P value (p=1/10) calculated after Murzin (1993), JMB, 230, 689-694

     No.  Domain1         Dcmain2         Sc      RMS     Len1 Len2  Align NFit Eq. Secs.   %I    %S    P(m)
Pair   1  2hhba           2hhbb           8.19    1.38     141  146   147  134 132    0  44.70 100.00 5.91e-24
Pair   2  2hhba           1lh1            5.97    2.30     141  153   154  119 109    0  13.76 100.00 0.05075
Pair   3  2hhba           2lhb            6.54    1.63     141  149   150  120 117    0  35.04 100.00 2.10e-13
Pair   4  2hhba           4mbn            7.78    1.41     141  153   148  135 132    0  26.52 100.00 4.10e-08
Pair   5  2hhba           1ecd            6.58    2.17     141  136   145  121 115    0  17.39 100.00 0.00982
Pair   6  2hhbb           1lh1            5.89    2.41     146  153   155  114 107    0  15.89 100.00 0.01768
Pair   7  2hhbb           2lhb            7.11    1.36     146  149   151  125 124    0  26.61 100.00 8.80e-08
Pair   8  2hhbb           4mbn            8.09    1.35     146  153   151  138 137    0  25.55 100.00 1.09e-07
Pair   9  2hhbb           1ecd            6.88    2.08     146  136   144  124 114    0  20.18 100.00 0.00112
Pair  10  1lh1            2lhb            5.74    2.04     153  149   155  110  96    0  16.67 100.00 0.02572
Pair  11  1lh1            4mbn            6.71    2.34     153  153   155  133 129    0  17.83 100.00 0.00237
Pair  12  1lh1            1ecd            5.85    2.56     153  136   150  114 102    0  17.65 100.00 0.01238
Pair  13  2lhb            4mbn            7.14    1.27     149  153   149  131 128    0  25.78 100.00 1.93e-07
Pair  14  2lhb            1ecd            6.40    1.86     149  136   145  120 118    0  17.80 100.00 0.00670
Pair  15  4mbn            1ecd            7.46    1.62     153  136   145  131 128    0  20.31 100.00 0.00021
Reading in matrix file globin.mat...
Doing cluster analysis...
Cluster:  1 (   2hhba  &    2hhbb ) Sc  8.19 RMS   1.38 Len 147 nfit 133
 See file globin.1 for the alignment and transformations
Cluster:  2 (    4mbn  &    2hhba    2hhbb ) Sc  8.97 RMS   1.29 Len 150 nfit 135
 See file globin.2 for the alignment and transformations
Cluster:  3 (    1ecd  &     4mbn    2hhba    2hhbb ) Sc  8.31 RMS   1.85 Len 148 nfit 124
 See file globin.3 for the alignment and transformations
Cluster:  4 (    2lhb  &     1ecd     4mbn    2hhba    2hhbb ) Sc  8.28 RMS   1.16 Len 154 nfit 117
 See file globin.4 for the alignment and transformations
Cluster:  5 (    1lh1  &     2lhb     1ecd     4mbn    2hhba    2hhbb ) Sc  7.76 RMS   2.49 Len 160 nfit 116
 See file globin.5 for the alignment and transformations
```

The various fields describe details of the pairwise and threewise comparisons: Sc, RMS deviation, the alignment length (Align), the length of each structure in residues (Len1, Len2), the number of atoms used in the RMSfit (Nfit), and the number of equivalent residues (Eq), among others.

STAMP will also produce other several files.

1. globin.mat - a file containing the information used to derive the structural similarity tree. This is an upper diagonal matrix containing the pairwise Sc values.

2. stamp_rough.trans - a file containing the information needed to generate superimposed coordinates.

3. globin.N - a series of files containing transformations and alignments created during the cluster analysis performed among all the pairwise comparisons.

Each file corresponds to a node in the similarity tree, where two groups of one or more structures have been combined to form an alignment (and a superimposition). The higher the value of N, the more structurally dissimilar the proteins contained in the file are. Highly similar structures are clustered at an early stage in the program, with more distantly structures being clustered towards the end.

The number of nodes is unknown before running STAMP. It depends on the similarity relationships of the studied proteins. The node with the highest N value contains all the proteins. We will focus our analysis in these files, since they allow us to establish the similarity relationships among the studied proteins, as well as obtain their structural alignments and superimpositions.

The information contained in each file can be divided in three parts:

- The top of each file contains the information needed to generate the superimposed coordinates

- Then various details of the similarity are given (Sc, RMS deviation, Align and Nfit)

- The bottom portion of the file contains the structural alignment in STAMP format.

This can be seen in the file globin.5, which contains the information about how similar are the six studied globins:

TRANSFORMATION MATRICES

```
/pdb1lh1.ent 1lh1 { ALL
  0.73917   -0.67330   -0.01701        14.29311
  0.37652    0.43403   -0.81845         8.89115
  0.55844    0.59857    0.57433       -59.22414  }
/pdb2lhb.ent 2lhb { ALL
 -0.15959    0.83999   -0.51861       -20.14010
  0.89511    0.34467    0.28282       -15.34162
  0.41631   -0.41907   -0.80688        -7.73481  }
/pdb1ecd.ent 1ecd { ALL
  0.21024   -0.33103   -0.91990        36.37023
 -0.43737   -0.87337    0.21433        17.41861
 -0.87436    0.35728   -0.32840        -5.28921  }
/pdb4mbn.ent 4mbn { ALL
  0.90777   -0.40212    0.11934         6.06018
 -0.06945    0.13649    0.98820         1.13498
 -0.41367   -0.90535    0.09597        15.95217  }
/pdb2hhb.ent 2hhba { CHAIN A
  0.93187   -0.35603   -0.06973         2.39600
  0.36258    0.92058    0.14512        -3.13810
  0.01253   -0.16052    0.98695         0.94267  }
/pdb2hhb.ent 2hhbb { CHAIN B
  0.92112    0.38665    0.04518         2.54219
  0.38865   -0.90675   -0.16355        -4.34164
 -0.02227    0.16821   -0.98550         2.95246  }
```

SIMILARITY ALIGNMENT

```
Alignment score  Sc = 7.763868
Alignment length Lp = 160
RMS deviation after fitting on 116 atoms =  2.493845
Secondary structures are from DSSP
```

(block format)

```
(...)
* iteration 1
 P       ?
 I       ?
 V       ?
 D       ?
 T       ?
 G       ?
 S       ?
 V       ?
GA   V ??   ?
AP VVH ?? ???
LLLLLL ?????? 1  0.46697   1.88313   6.76700
TSSSST ?????? 1  0.50702   1.98539   7.33000
EAAEPP ?????? 1  0.56600   1.88225   8.15900
SADGAE ?????? 1  0.62200   1.78431   8.94600
QEQEDE ?????? 1  0.71598   1.61692  10.26700
AKIWKK ?????? 1  0.76700   1.50272  10.98400
ATSQTS ?????? 1  0.76899   1.49092  11.01200
LKTLNA ?????? 1  0.82398   1.33536  11.78500
VIVVVV ?????? 1  0.86802   1.19291
(...)
```

Generating transformed coordinates: TRANSFORM

The program transform can be used to output a pdb file of the superimposed structures belonging to a particular node of the cluster analysis performed by STAMP. For example, to get the pdb file of the superimposition for all the studied globins, type

```
$ transform –f globin.5 –g –o globin5_stamp.pdb
```

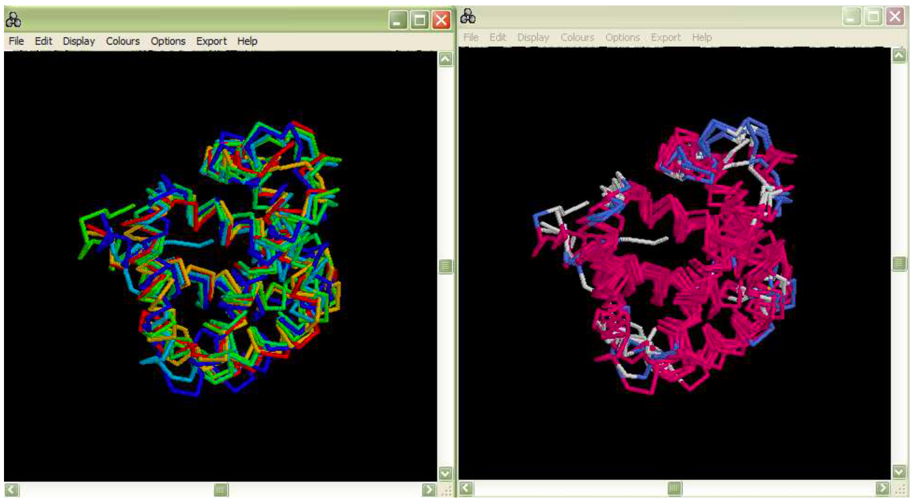
Note that the name of the output file is just a suggestion. It's advisable that the file has the pdb extension, and the name must be as self-explanatory as possible.

The -g -o options put transformed coordinates for each structure into one file. Each structure will be labeled sequentially with a different chain identifier (i.e. A, B, C) This will be the header of the resulting pdb file:

```
REMARK Output from transform
REMARK STAMP Package (Russell and Barton Proteins, 14, 309-323, 1992)
REMARK Domains were read from the file globin.5
REMARK Chains are labelled sequentially starting with 'A' and
REMARK after the order given in the file globin.5
REMARK The domains in this file are:
REMARK 1lh1 chain A
REMARK 2lhb chain B
REMARK 1ecd chain C
REMARK 4mbn chain D
REMARK 2hhba chain E
REMARK 2hhbb chain F
REMARK Does not include heteroatoms
REMARK Does not include DNA/RNA
REMARK Does not include waters
```

By default, TRANSFORM does not include heteroatoms in the output. If you wish heteroatoms to be included, then add -het to the transform command. If you wish waters to be included in the file add -hoh.

Then, we can display this file to further analysis the similarity:


File Edit Display Colours Options Export Help
File Edit Display Colours Options Export Help

Notice that if you want to emphasize dissimilarity is useful to colour by chain, while colouring by structure emphasizes similarity.

Displaying alignments: ACONVERT

The program aconvert converts various alignment formats back and forth, and can be used to change the STAMP block file format into another friendlier format, such CLUSTAL. The format is

aconvert –in format1 –out format2 <input file> output file

where:

format1	format of the input file
format2	format of the output file
input file	the original alignment file with format1
output file	the new file with the desired format2

Available formats are:

b block; c clustalw; h hmmer; p pir; m msf; f fasta; s pfam;y phylip.

To convert the globin.5 STAMP alignment into CLUSTAL format, type

```
$ aconvert –in b –out c <globin.5 > globin5_stamp.aln
```

Note that, again, the name of output file is just a suggestion.

The output file looks like:

```
1lh1       --------GALTESQAALVKSSWEEFNANIPKHTHRFFILVLEIAPAAKDLFSFLKGTSE
2lhb       PIVDTGSVAPLSAAEKTKIRSAWAPVYSTYETSGVDILVKFFTSTPAAQEFFPKFKGLTT
1ecd       ----------LSADQISTVQASFDKV--K-G-DPVGILYAVFKADPSIMAKFTQFAG-KD
4mbn       ---------VLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFKSHPETLEKFDRFKHLKT
2hhba      ---------VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-D---
2hhbb      --------VHLTPEEKSAVTALWG-KV-NVDEVGGEALGRLLVVYPWTQRFFESFGDLST
space      ------------------------------------------------------------
1lh1_dssp  --------????????????????????????????????????????????????????
2lhb_dssp  ????????????????????????????????????????????????????????????
1ecd_dssp  ----------????????????????--?-?-?????????????????????????-??
4mbn_dssp  ---------???????????????????????????????????????????????????
2hhba_dssp ---------??????????????????????????????????????????????-?---
2hhbb_dssp --------????????????????-??-????????????????????????????????
(…)
```

As you can see, there are additional lines for the DSSP secondary structures. Since this information is not available in our case, we have ? characters instead. If the folder contains the

DSSP files it will show the secondary structure. It is very important to remove these lines from the alignment to get a MSA for HMMER.

Structural alignment using an initial sequence alignment: ALIGNFIT

ROUGHFIT will not always work. If when using the ROUGHFIT option in the standard output warnings of LOW SCORES appear, this usually means that ROUGHFIT hasn't managed to generating a good enough starting superimposition. Then, one possible solution is to provide an initial sequence alignment to guide the starting superimposition. This can be done using the ALIGNFIT option.

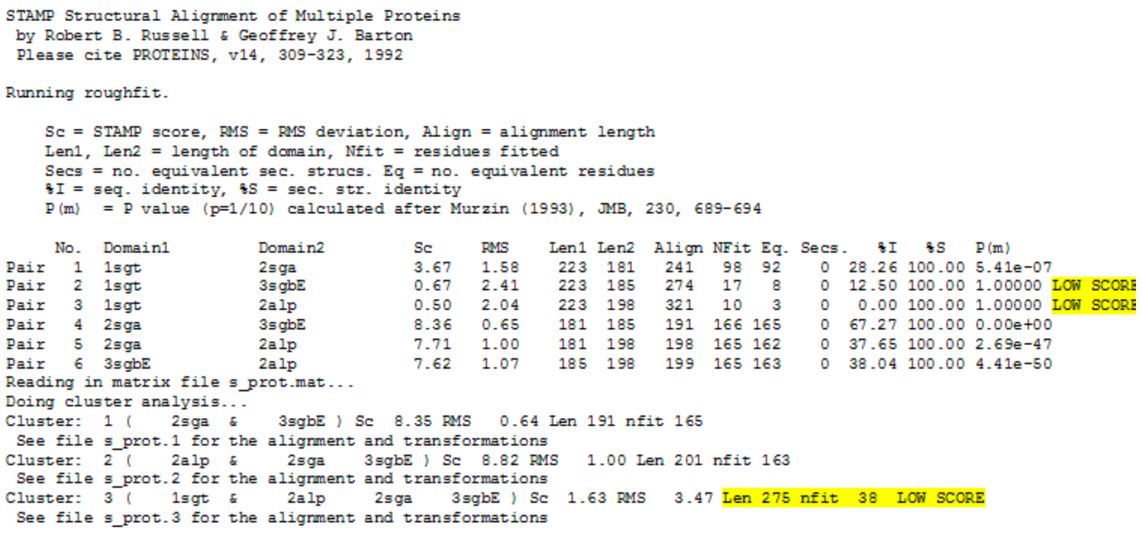
We will use a set of four bacterial serine proteinases as an example.

The files are in the directory:
/cursos/BE/exercises/[year]/practice3/STAMP/s_prot.

We can try the ROUGHFIT option by typing

```
$ stamp -l s_prot.domains -rough -n 2 -prefix s_prot > s_prot.out
```

and we will obtain a standard output, which will look like:



```
STAMP Structural Alignment of Multiple Proteins
 by Robert B. Russell & Geoffrey J. Barton
 Please cite PROTEINS, v14, 309-323, 1992

Running roughfit.

    Sc = STAMP score, RMS = RMS deviation, Align = alignment length
    Len1, Len2 = length of domain, Nfit = residues fitted
    Secs = no. equivalent sec. strucs. Eq = no. equivalent residues
    %I = seq. identity, %S = sec. str. identity
    P(m)  = P value (p=1/10) calculated after Murzin (1993), JMB, 230, 689-694

     No.  Domain1         Domain2         Sc     RMS    Len1 Len2  Align NFit Eq. Secs.   %I    %S    P(m)
Pair   1  1sgt            2sga            3.67   1.58    223  181   241   98  92    0  28.26 100.00 5.41e-07
Pair   2  1sgt            3sgbE           0.67   2.41    223  185   274   17   8    0  12.50 100.00 1.00000 LOW SCORE
Pair   3  1sgt            2alp            0.50   2.04    223  198   321   10   3    0   0.00 100.00 1.00000 LOW SCORE
Pair   4  2sga            3sgbE           8.36   0.65    181  185   191  166 165    0  67.27 100.00 0.00e+00
Pair   5  2sga            2alp            7.71   1.00    181  198   198  165 162    0  37.65 100.00 2.69e-47
Pair   6  3sgbE           2alp            7.62   1.07    185  198   199  165 163    0  38.04 100.00 4.41e-50
Reading in matrix file s_prot.mat...
Doing cluster analysis...
Cluster:  1 (    2sga  &     3sgbE ) Sc  8.35 RMS   0.64 Len 191 nfit 165
 See file s_prot.1 for the alignment and transformations
Cluster:  2 (    2alp  &     2sga    3sgbE ) Sc  8.82 RMS   1.00 Len 201 nfit 163
 See file s_prot.2 for the alignment and transformations
Cluster:  3 (    1sgt  &     2alp     2sga    3sgbE ) Sc  1.63 RMS   3.47 Len 275 nfit  38  LOW SCORE
 See file s_prot.3 for the alignment and transformations
```

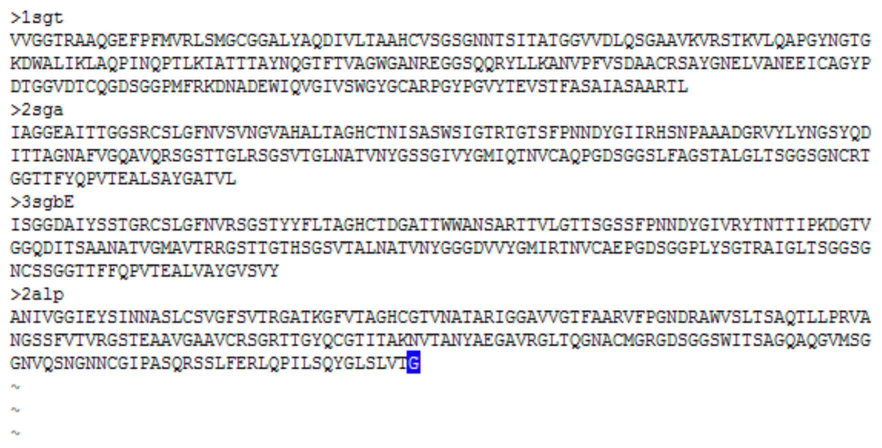
The low scores warning indicates that STAMP is not able to correctly superimpose some of the proteins. Notice that in cluster 3 when protein 1sgt is added to the previous cluster the number of fitted residues is almost negligible (38 vs 275).

To perform STAMP with an initial sequence alignment we can follow these five steps:

1. Run the program pdbseq to extract the protein sequences in fasta format from the information stored in the domains file:

```
$ pdbseq –f s_prot.domains > s_prot.fa
```

Edit s_prot.fa and clean the title line of each sequence in such a way that it must only contain the symbol ">" plus the same label for the protein as in s_prot.domains. So the "cleaned" file would look like:

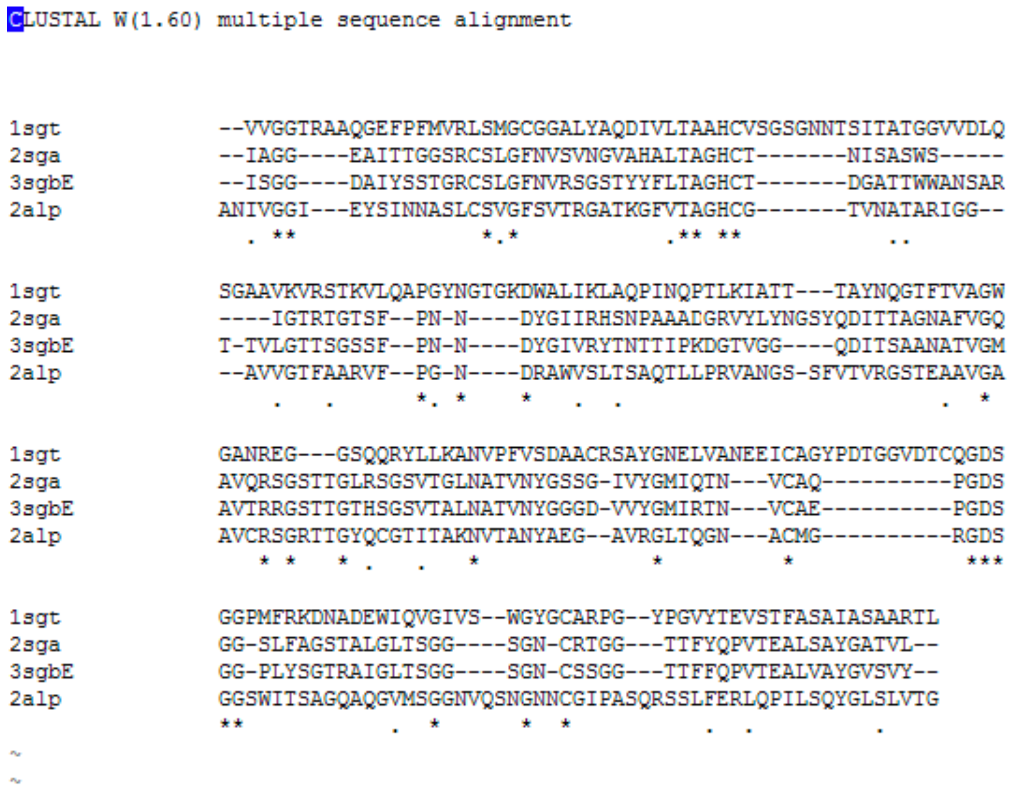


```
>1sgt
VVGGTRAAQGEFPFMVRLSMGCGGALYAQDIVLTAAHCVSGSGNNTSITATGGVVDLQSGAAVKVRSTKVLQAPGYNGTG
KDWALIKLAQPINQPTLKIATTTAYNQGTFTVAGWGANREGGSQQRYLLKANVPFVSDAACRSAYGNELVANEEICAGYP
DTGGVDTCQGDSGGPMFRKDNADEWIQVGIVSWGYGCARPGYPGVYTEVSTFASAIASAARTL
>2sga
IAGGEAITTGGSRCSLGFNVSVNGVAHALTAGHCTNISASWSIGTRTGTSFPNNDYGIIRHSNPAAADGRVYLYNGSYQD
ITTAGNAFVGQAVQRSGSTTGLRSGSVTGLNATVNYGSSGIVYGMIQTNVCAQPGDSGGSLFAGSTALGLTSGGSGNCRT
GGTTFYQPVTEALSAYGATVL
>3sgbE
ISGGDAIYSSTGRCSLGFNVRSGSTYYFLTAGHCTDGATTWWANSARTTVLGTTSGSSFPNNDYGIVRYTNTTIPKDGTV
GGQDITSAANATVGMAVTRRGSTTGTHSGSVTALNATVNYGGGDVVYGMIRTNVCAEPGDSGGPLYSGTRAIGLTSGGSG
NCSSGGTTFFQPVTEALVAYGVSVY
>2alp
ANIVGGIEYSINNASLCSVGFSVTRGATKGFVTAGHCGTVNATARIGGAVVGTFAARVFPGNDRAWVSLTSAQTLLPRVA
NGSSFVTVRGSTEAAVGAAVCRSGRTTGYQCGTITAKNVTANYAEGAVRGLTQGNACMGRGDSGGSWITSAGQAQGVMSG
GNVQSNGNNCGIPASQRSSLFERLQPILSQYGLSLVTG
~
~
~
```

2. Make a multiple sequence alignment, using clustalw:

$ clustalw s_prot.fa

as a result, we will obtain, among others, the file s_prot.aln, which contains the multiple sequence alignment:



```
CLUSTAL W(1.60) multiple sequence alignment



1sgt            --VVGGTRAAQGEFPFMVRLSMGCGGALYAQDIVLTAAHCVSGSGNNTSITATGGVVDLQ
2sga            --IAGG----EAITTGGSRCSLGFNVSVNGVAHALTAGHCT-------NISASWS-----
3sgbE           --ISGG----DAIYSSTGRCSLGFNVRSGSTYYFLTAGHCT-------DGATTWWANSAR
2alp            ANIVGGI---EYSINNASLCSVGFSVTRGATKGFVTAGHCG-------TVNATARIGG--
                  . **              *.*           .** **           ..

1sgt            SGAAVKVRSTKVLQAPGYNGTGKDWALIKLAQPINQPTLKIATT---TAYNQGTFTVAGW
2sga            ----IGTRTGTSF--PN-N----DYGIIRHSNPAAADGRVYLYNGSYQDITTAGNAFVGQ
3sgbE           T-TVLGTTSGSSF--PN-N----DYGIVRYTNTTIPKDGTVGG----QDITSAANATVGM
2alp            --AVVGTFAARVF--PG-N----DRAWVSLTSAQTLLPRVANGS-SFVTVRGSTEAAVGA
                    .   .      *. *    *   .  .                        .  *

1sgt            GANREG---GSQQRYLLKANVPFVSDAACRSAYGNELVANEEICAGYPDTGGVDTCQGDS
2sga            AVQRSGSTTGLRSGSVTGLNATVNYGSSG-IVYGMIQTN---VCAQ----------PGDS
3sgbE           AVTRRGSTTGTHSGSVTALNATVNYGGGD-VVYGMIRTN---VCAE----------PGDS
2alp            AVCRSGRTTGYQCGTITAKNVTANYAEG--AVRGLTQGN---ACMG----------RGDS
                   * *   * .   .   *             *         *             ***

1sgt            GGPMFRKDNADEWIQVGIVS--WGYGCARPG--YPGVYTEVSTFASAIASAARTL
2sga            GG-SLFAGSTALGLTSGG----SGN-CRTGG---TTFYQPVTEALSAYGATVL--
3sgbE           GG-PLYSGTRAIGLTSGG----SGN-CSSGG---TTFFQPVTEALVAYGVSVY--
2alp            GGSWITSAGQAQGVMSGGNVQSNGNNCGIPASQRSSLFERLQPILSQYGLSLVTG
                **           .  *      *  *          .  .         .
~
~
```

3. Run the program aconvert to convert the clustal alignment into block format:

```
$ aconvert -in c -out b <s_prot.aln> s_prot.block
```

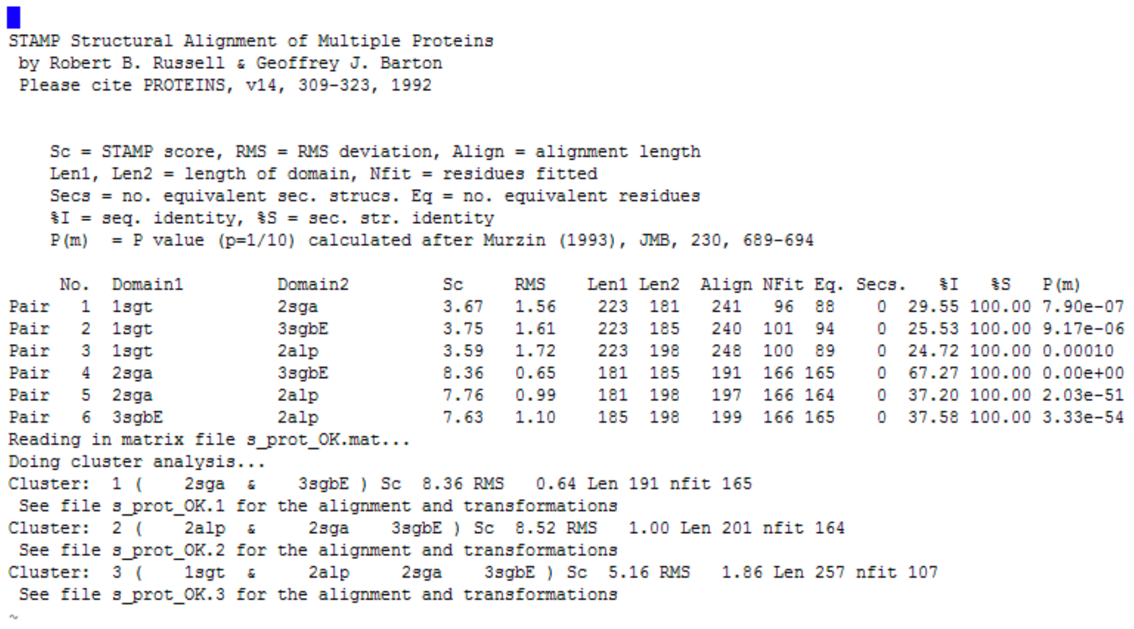
4. Run the program alignfit to incorporate the sequence alignment information

```
$ alignfit -f s_prot.block -d s_prot.domains -out s_prot.trans
```

5. Run stamp

```
$ stamp -l s_prot.trans -prefix s_prot_OK > s_prot_OK.out
```

and now the standard output file will look like:

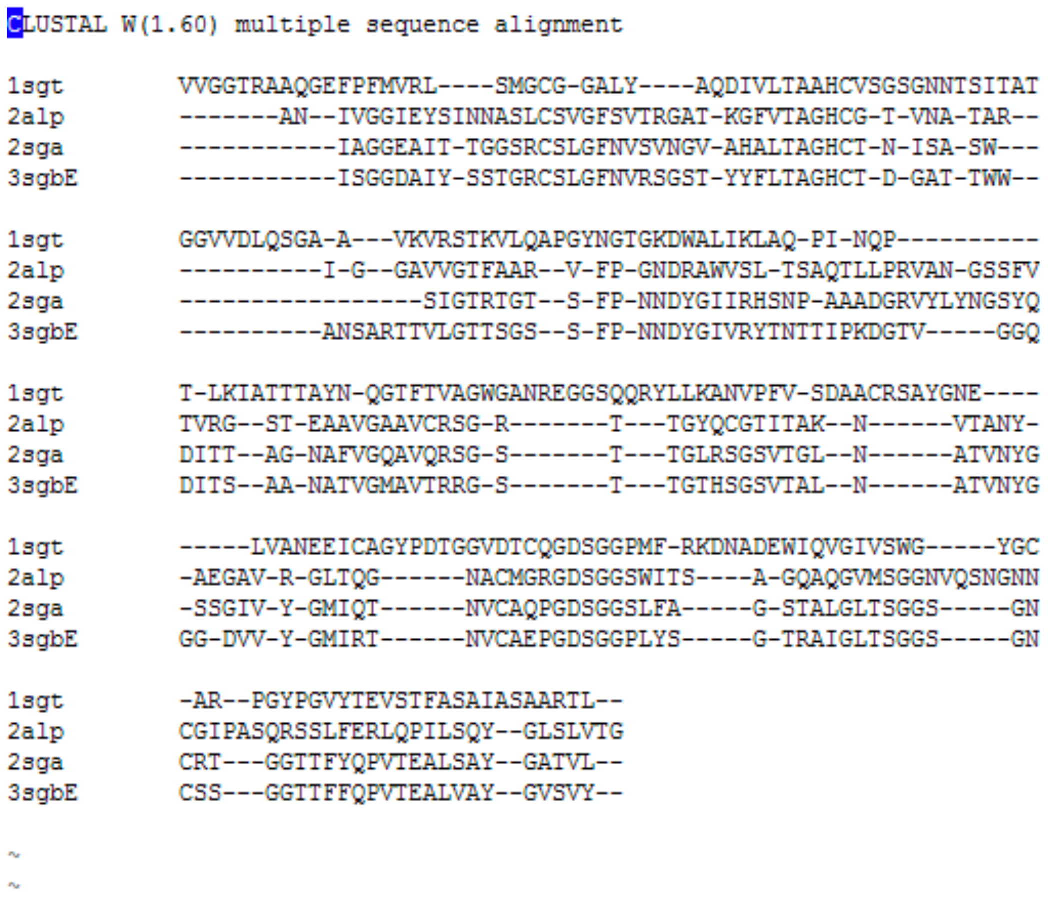


```
STAMP Structural Alignment of Multiple Proteins
 by Robert B. Russell & Geoffrey J. Barton
 Please cite PROTEINS, v14, 309-323, 1992


    Sc = STAMP score, RMS = RMS deviation, Align = alignment length
    Len1, Len2 = length of domain, Nfit = residues fitted
    Secs = no. equivalent sec. strucs. Eq = no. equivalent residues
    %I = seq. identity, %S = sec. str. identity
    P(m)  = P value (p=1/10) calculated after Murzin (1993), JMB, 230, 689-694

     No.  Domain1          Domain2          Sc     RMS     Len1 Len2  Align NFit Eq. Secs.   %I    %S    P(m)
Pair   1  1sgt             2sga             3.67   1.56     223  181   241   96  88     0  29.55 100.00 7.90e-07
Pair   2  1sgt             3sgbE            3.75   1.61     223  185   240  101  94     0  25.53 100.00 9.17e-06
Pair   3  1sgt             2alp             3.59   1.72     223  198   248  100  89     0  24.72 100.00 0.00010
Pair   4  2sga             3sgbE            8.36   0.65     181  185   191  166 165     0  67.27 100.00 0.00e+00
Pair   5  2sga             2alp             7.76   0.99     181  198   197  166 164     0  37.20 100.00 2.03e-51
Pair   6  3sgbE            2alp             7.63   1.10     185  198   199  166 165     0  37.58 100.00 3.33e-54
Reading in matrix file s_prot_OK.mat...
Doing cluster analysis...
Cluster:  1 (    2sga  &    3sgbE ) Sc  8.36 RMS   0.64 Len 191 nfit 165
 See file s_prot_OK.1 for the alignment and transformations
Cluster:  2 (    2alp  &     2sga    3sgbE ) Sc  8.52 RMS   1.00 Len 201 nfit 164
 See file s_prot_OK.2 for the alignment and transformations
Cluster:  3 (    1sgt  &     2alp     2sga    3sgbE ) Sc  5.16 RMS   1.86 Len 257 nfit 107
 See file s_prot_OK.3 for the alignment and transformations
~
```

6. We can now obtain the superimposition using transform, and the structural alignment using aconvert:

```
$ transform - f s_prot_OK.3 -g -o s_prot_OK3_stamp.pdb
$ aconvert -in b -out c <s_prot_OK.3> s_prot_OK3_stamp.aln
```

Notice that there are differences between this alignment and the CLUSTAL one:



```
CLUSTAL W(1.60) multiple sequence alignment

1sgt        VVGGTRAAQGEFPFMVRL----SMGCG-GALY----AQDIVLTAAHCVSGSGNNTSITAT
2alp        -------AN--IVGGIEYSINNASLCSVGFSVTRGAT-KGFVTAGHCG-T-VNA-TAR--
2sga        -----------IAGGEAIT-TGGSRCSLGFNVSVNGV-AHALTAGHCT-N-ISA-SW---
3sgbE       -----------ISGGDAIY-SSTGRCSLGFNVRSGST-YYFLTAGHCT-D-GAT-TWW--

1sgt        GGVVDLQSGA-A---VKVRSTKVLQAPGYNGTGKDWALIKLAQ-PI-NQP----------
2alp        ----------I-G--GAVVGTFAAR--V-FP-GNDRAWVSL-TSAQTLLPRVAN-GSSFV
2sga        -----------------SIGTRTGT--S-FP-NNDYGIIRHSNP-AAADGRVYLYNGSYQ
3sgbE       ----------ANSARTTVLGTTSGS--S-FP-NNDYGIVRYTNTTIPKDGTV-----GGQ

1sgt        T-LKIATTTAYN-QGTFTVAGWGANREGGSQQRYLLKANVPFV-SDAACRSAYGNE----
2alp        TVRG--ST-EAAVGAAVCRSG-R-------T---TGYQCGTITAK--N------VTANY-
2sga        DITT--AG-NAFVGQAVQRSG-S-------T---TGLRSGSVTGL--N------ATVNYG
3sgbE       DITS--AA-NATVGMAVTRRG-S-------T---TGTHSGSVTAL--N------ATVNYG

1sgt        -----LVANEEICAGYPDTGGVDTCQGDSGGPMF-RKDNADEWIQVGIVSWG-----YGC
2alp        -AEGAV-R-GLTQG------NACMGRGDSGGSWITS----A-GQAQGVMSGGNVQSNGNN
2sga        -SSGIV-Y-GMIQT------NVCAQPGDSGGSLFA-----G-STALGLTSGGS-----GN
3sgbE       GG-DVV-Y-GMIRT------NVCAEPGDSGGPLYS-----G-TRAIGLTSGGS-----GN

1sgt        -AR--PGYPGVYTEVSTFASAIASAARTL--
2alp        CGIPASQRSSLFERLQPILSQY--GLSLVTG
2sga        CRT---GGTTFYQPVTEALSAY--GATVL--
3sgbE       CSS---GGTTFFQPVTEALVAY--GVSVY--

~
~
```

It is also interesting to display the superimposition and analyze the similarities and dissimilarities among these proteins.