# November 9, 2012

The work in this folder is a continuation of the work in the "zdiffs" folder. The goal is still to evaluate the accuracy of homology models made from sequence alignments of HHOMP clusters. However, there will be some changes:

* No longer will I be assigning to each residue in the prediction target (the protein whose structure we are attempting to find via homology modeling) a number equal to the z component of the distance between the predicted and actual positions of its Cα. Instead, each target residue will get a True or False value, representing whether it was paired to the same position in the template as TopMatch paired it to with a structural alignment.
* The sequences of the target and template will be the full sequences taken from the PDB entry, not the partial sequences extracted from the PDB file. In the zdiffs folder, anything not resolved structurally was also missing from the sequence, which is silly. Ideally I would like to use the sequence of the natural protein, in case there are sections removed from the gene for which they solved the structure, but I'm probably not going to do that.
* Alignments will be produced with both ClustalW using the Gonnet series, and ClustalΩ, so that I can compare them.

There will also be a few changes of form with no change in content, small fixes:

* Alignment file's names will have file extensions
* Alignments will be in FASTA format rather than Clustal format, since this is what Daniel's backend takes.
  + Side note - this makes me happy that I use BioPython. If I wrote the backend, I would just change a single word (AlignIO.read(filename, "~~fasta~~ clustal")), or add a line of code that looks at the file extension and chooses an appropriate format! This would be *inconsequential*. It really makes me wonder what things I spend time on that I wouldn't have to pay attention to with the right piece of software...

# November 12, 2012

How severe are the differences between the PDB FASTA sequences, and the sequences of the structures in the structure files themselves?

I downloaded the FASTA format sequences from the PDB entries 2MPR, 1BY5, and 1T16.

I created a FASTA file that contains both the full sequence of 1T16 chain A and the structure-derived sequence, using the following commands in PyMOL (the session is abridged, but all the necessary steps are shown):

PyMOL>fetch 1t16

HEADER LIPID TRANSPORT 15-APR-04 1T16

TITLE CRYSTAL STRUCTURE OF THE BACTERIAL FATTY ACID TRANSPORTER

TITLE 2 FADL FROM ESCHERICHIA COLI

COMPND MOL\_ID: 1;

COMPND 2 MOLECULE: LONG-CHAIN FATTY ACID TRANSPORT PROTEIN;

COMPND 3 CHAIN: A, B;

COMPND 4 SYNONYM: OUTER MEMBRANE FADL PROTEIN, OUTER MEMBRANE FLP

COMPND 5 PROTEIN;

COMPND 6 ENGINEERED: YES

ObjectMolecule: Read secondary structure assignments.

ObjectMolecule: Read crystal symmetry information.

Symmetry: Found 2 symmetry operators.

CmdLoad: ".\1t16.pdb" loaded as "1t16".

PyMOL>stored.seq = ''

PyMOL>**from** sundries **import** one\_letter

PyMOL>iterate polymer & c. a & n. ca, stored.seq += one\_letter[resn]

Iterate: iterated over 427 atoms.

PyMOL>**import** Bio.SeqIO

PyMOL>**from** Bio.SeqIO **import** SeqRecord

PyMOL>**from** Bio.Seq **import** Seq

PyMOL>cd C:\cygwin\home\alex\beta-barrel-oligomerization\checking alignments against topmatch\comparing sequences

cd: now **in** C:\cygwin\home\alex\beta-barrel-oligomerization\checking alignments against topmatch\comparing sequences

PyMOL>full\_seqs = list(Bio.SeqIO.parse(open('1t16.fasta'), 'fasta'))

PyMOL>**print**(full\_seqs[0].id)

1T16:A|PDBID|CHAIN|SEQUENCE

PyMOL>PyMOL>stru\_seq = SeqRecord(Seq(stored.seq), 'seq\_from\_stru\_1t16')

PyMOL>Bio.SeqIO.write([stru\_seq, full\_seqs[0]], open('1t16 full and structure derived sequences.fasta', 'w'), 'fasta'

Opening these sequences in ClustalX, it is clear that they are exactly the same.

I did more or less the same procedure for 1by5:

PyMOL>**import** Bio.SeqIO

PyMOL>**from** Bio.SeqIO **import** SeqRecord

PyMOL>**from** Bio.Seq **import** Seq

PyMOL>fetch 1by5

HEADER METAL BINDING PROTEIN 23-OCT-98 1BY5

TITLE FHUA FROM E. COLI, WITH ITS LIGAND FERRICHROME

COMPND MOL\_ID: 1;

COMPND 2 MOLECULE: FERRIC HYDROXAMATE UPTAKE PROTEIN;

COMPND 3 CHAIN: A;

COMPND 4 SYNONYM: FHUA;

COMPND 5 MOL\_ID: 2;

COMPND 6 MOLECULE: FERRICHROME;

COMPND 7 CHAIN: B

ObjectMolecule: Read secondary structure assignments.

ObjectMolecule: Read crystal symmetry information.

Symmetry: Found 4 symmetry operators.

CmdLoad: ".\1by5.pdb" loaded as "1by5".

PyMOL>stored.seq = ''

PyMOL>**from** sundries **import** one\_letter

PyMOL>cd C:\cygwin\home\alex\beta-barrel-oligomerization\checking alignments against topmatch\comparing sequences

cd: now **in** C:\cygwin\home\alex\beta-barrel-oligomerization\checking alignments against topmatch\comparing sequences

PyMOL>iterate polymer & c. a & n. ca, stored.seq += one\_letter[resn]

Iterate: iterated over 697 atoms.

PyMOL>full\_seqs = list(Bio.SeqIO.parse(open('1by5.fasta'), 'fasta'))

Here I take a look at the sequence for Chain B, and decide that it's inconsequential:

PyMOL>**print**(full\_seqs[1])

ID: 1BY5:B|PDBID|CHAIN|SEQUENCE

Name: 1BY5:B|PDBID|CHAIN|SEQUENCE

Description: 1BY5:B|PDBID|CHAIN|SEQUENCE

Number of features: 0

Seq('AAAGGG', SingleLetterAlphabet())

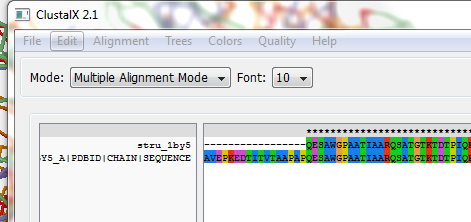
PyMOL>**print**(full\_seqs[0].id)

1BY5:A|PDBID|CHAIN|SEQUENCE

PyMOL>stru\_seq = SeqRecord(Seq(stored.seq), id='stru\_1by5',description = 'derived from .pdb file')

PyMOL>Bio.SeqIO.write([stru\_seq, full\_seqs[0]], open('1by5 structure-derived and full sequences.fasta', 'w'), 'fasta')

This time, looking at them in ClustalX, I found that they were *not* the same. I aligned them, using the identity matrix and with gap penalties set to 0, and found that the strcture-derived sequence is missing 15-20 residues from the beginning. However, no loops connecting the protein's strands are missing. Here's a piece of what I'm looking at:



I did the same thing for 2MPR:

PyMOL>rein

PyMOL>fetch 2mpr

HEADER OUTER MEMBRANE PROTEIN 07-FEB-97 2MPR

TITLE MALTOPORIN FROM SALMONELLA TYPHIMURIUM

COMPND MOL\_ID: 1;

COMPND 2 MOLECULE: MALTOPORIN;

COMPND 3 CHAIN: A, B, C;

COMPND 4 FRAGMENT: MATURE PROTEIN, SIGNAL SEQUENCE CLEAVED OFF;

COMPND 5 SYNONYM: LAM-B, MAL-L

ObjectMolecule: Read secondary structure assignments.

ObjectMolecule: Read crystal symmetry information.

Symmetry: Found 8 symmetry operators.

CmdLoad: PDB-string loaded into object "2mpr", state 1.

PyMOL>stored.seq = ''

PyMOL>iterate polymer & chain A & name CA, stored.seq += one\_letter[resn]

Iterate: iterated over 421 atoms.

PyMOL>full\_seqs = list(Bio.SeqIO.parse(open('2mpr.fasta'),'fasta'))

Here I demonstrate that each chain has the same sequence:

PyMOL>**print**(str(full\_seqs[0].seq) == str(full\_seqs[1].seq))

True

PyMOL>**print**(str(full\_seqs[1].seq) == str(full\_seqs[2].seq))

True

PyMOL>**for** i **in** full\_seqs: **print**(i.id)

2MPR:A|PDBID|CHAIN|SEQUENCE

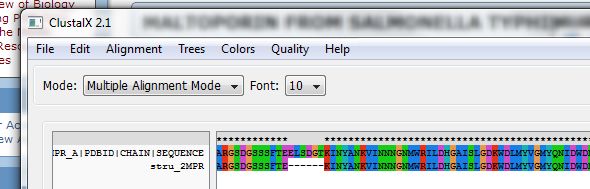
2MPR:B|PDBID|CHAIN|SEQUENCE

2MPR:C|PDBID|CHAIN|SEQUENCE

PyMOL>stru\_seq = SeqRecord(Seq(stored.seq), id='stru\_2MPR', description='2MPR sequence extracted from structure')

PyMOL>Bio.SeqIO.write([full\_seqs[0], stru\_seq], open('2mpr structure derived and full sequences.fasta', 'w'), 'fasta')

I found that the structure of 2MPR is missing a few residues in the middle of the protein that are represented in the sequence:

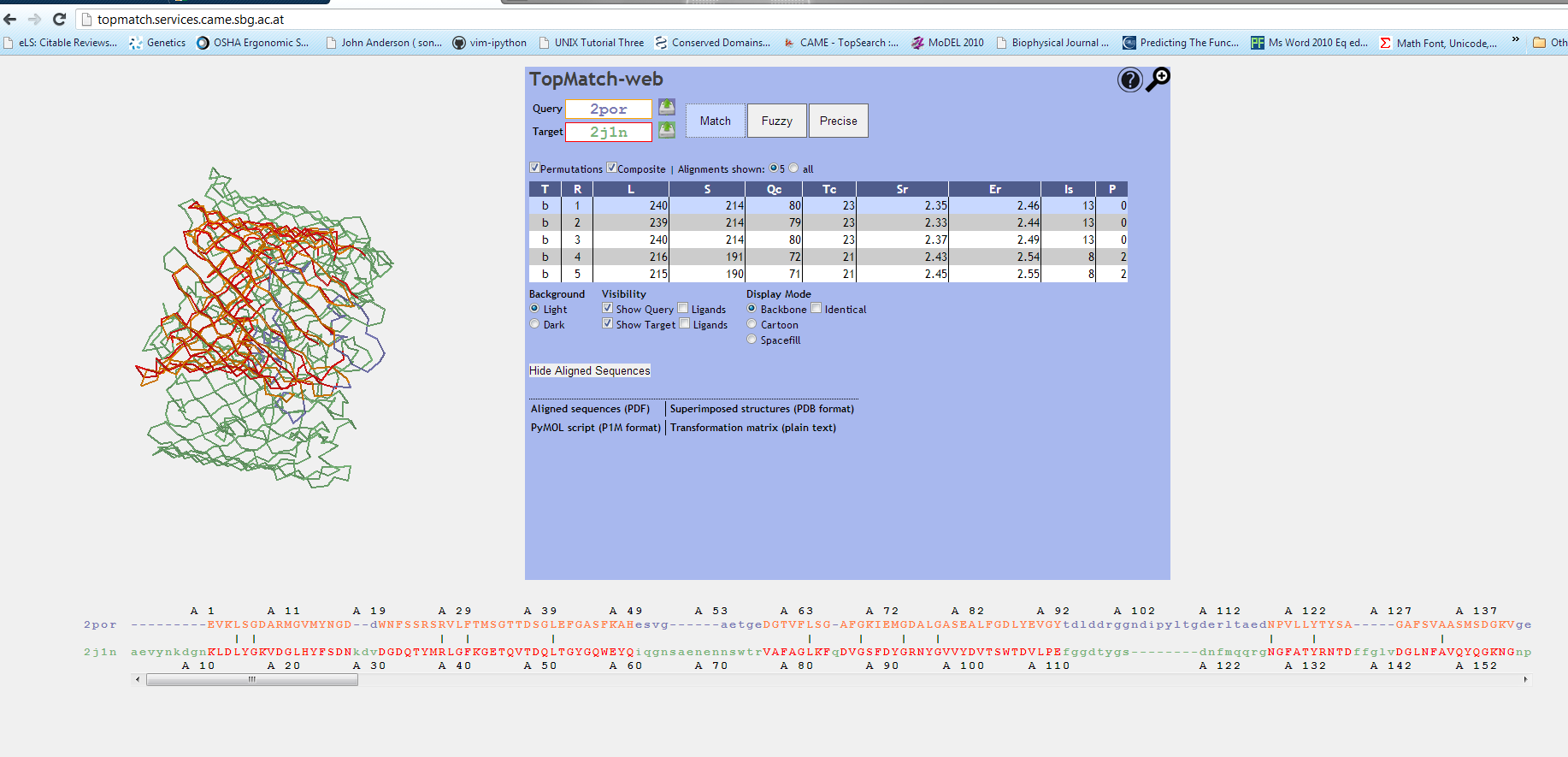


Again, the alignment was made in ClustalX using the identity matrix and no gap penalty.

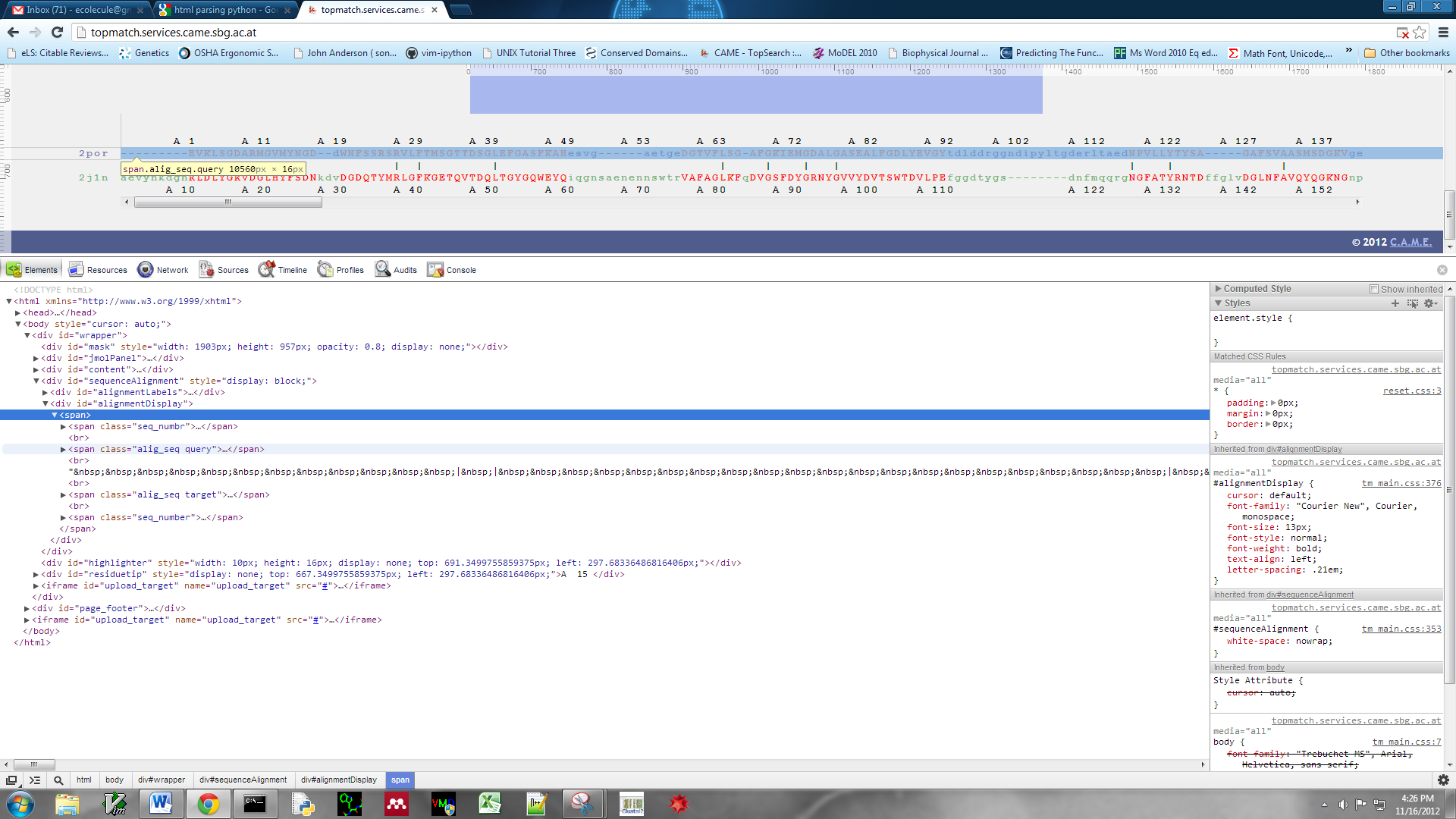
# November 16, 2012

TopMatch returns results in JSON format. I am interested in the sequence alignment that is part of a TopMatch pairing.

In the JSON results, the sequence alignment is under the pattern of keys algbox\_html-->htmlstring. These keys lead to the HTML code for the sequence box in the online TopMatch GUI, at http://topmatch.services.came.sbg.ac.at/.



Using Google Chrome's "Inspect Element" tool, some details of the organization of this html code are revealed. The tool looks like this:



It shows that the query sequence is contained within a tag <span class="alig\_seq query">, and the target sequence within a tag <span class="alig\_seq target">. Within these tags, the sequence is represented as a simple sequence of letters. The only tags are <span class="aligEquivQuery" id=...>, which surround sections of the sequence that were successfully structurally aligned.