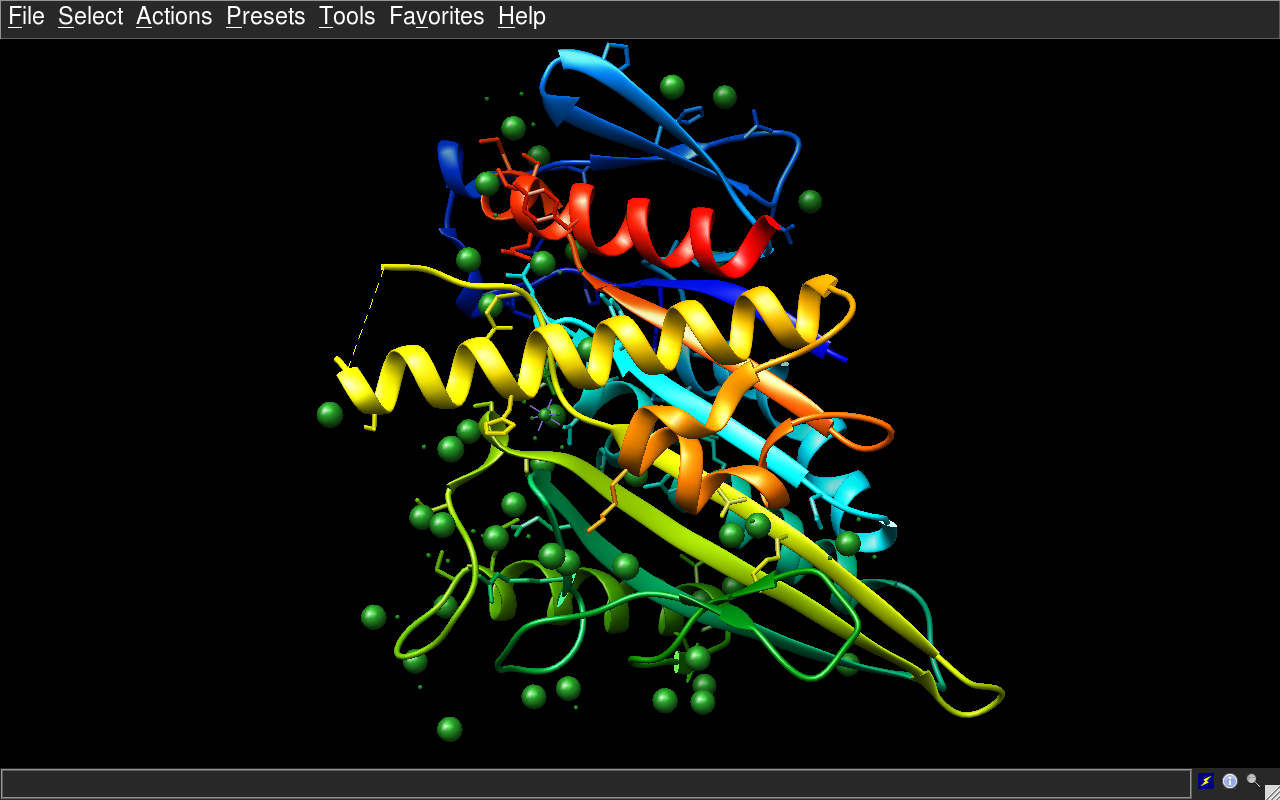
Step 1

1. X-ray diffraction
2. 1.85 A
3. Homo sapiens
4. 32% helical (14 helices; 107 residues)  
   26% beta sheet (18 strands; 89 residues)
5. >5WDE:A|PDBID|CHAIN|SEQUENCE  
   GSKGNIRVIARVRPVTKEDGEGPEATNAVTFDADDDSIIHLLHKGKPVSFELDKVFSPQASQQDVFQEVQALVTSCIDGFNVCIFAYGQTGAGKTYTMEGTAENPGINQRALQLLFSEVQEKASDWEYTITVSAAEIYNEVLRDLLGKEPQEKLEIRLCPDGSGQLYVPGLTEFQVQSVDDINKVFEFGHTNRTTEFTNLNEHSSRSHALLIVTVRGVDCSTGLRTTGKLNLVDLAGSERVGKSGAEGSRLREAQHINKSLSALGDVIAALRSRQGHVPFRNSKLTYLLQDSLSGDSKTLMVVQVSPVEKNTSETLYSLKFAERVRSVEL

Step 2



Step 3

We decided that we wanted the process to happen more automatically. Here is our bash script which converted the file to the correct result specified:

|  |
| --- |
| #!/bin/bash  ENCOLUMNED=$(cat $1 | grep -v '\.$' | cut -c 14,17 | tail -n +2 | grep -v '!' | sed -e 's/ $/C/g')  **for** i **in** {1..2}; **do**  **for** j **in** $ENCOLUMNED; **do**   echo $j  **done** |   cut -c $i |   paste -s -d '';  **done** |

Step4:

Porter:

Subject: Porter response to

Query\_name:

Query\_length: 316

Prediction:

NIRVIARVRPVTKEDGEGPEATNAVTFDADDDSIIHLLHKGKPVSFELDKVFSPQASQQD

CEEEEEEECCCCHHHHCCCCCCECEEEECCCCEEEEEECCCCEEEEECCEEECCCCCHHH

VFQEVQALVTSCIDGFNVCIFAYGQTGAGKTYTMEGTAENPGINQRALQLLFSEVQEKAS

HHHHHHHHHHHHHCCCCEEEEEECCCCCCHHHHHCECCCCECHHHHHHHHHHHHHHHCCC

DWEYTITVSAAEIYNEVLRDLLGKEPQEKLEIRLCPDGSGQLYVPGLTEFQVQSVDDINK

CEEEEEEEEEEEEECCEEEECCCCCCCCCCCEEEECCCCCEEEECCCCCEEECCHHHHHH

VFEFGHTNRTTEFTNLNEHSSRSHALLIVTVRGVDCSTGLRTTGKLNLVDLAGSERVGSR

HHHHHHHHCCCCCCCCCCCHHHCEEEEEEEEEEEECCCCEEEEEEEEEEECCCCCCCCHH

LREAQHINKSLSALGDVIAALRSRQGHVPFRNSKLTYLLQDSLSGDSKTLMVVQVSPVEK

HHHHHHHCHHHHHHHHHHHHHHCCCCCCCHHHCHHHHHCHHHHCCCCEEEEEEEECCEHH

NTSETLYSLKFAERVR

HHHHHHHHHHHHHHHC

Predictions based on PDB templates (seq. similarity up to 100.0%)

Step 5

Results returned for best match:

|  |
| --- |
| > 1SDMA Length=344   Score = 247 bits (631), Expect = 4e-81, Method: Compositional matrix adjust.  Identities = 133/317 (42%), Positives = 194/317 (61%), Gaps = 7/317 (2%)  Query 2 IRVIARVRPVTKEDGEGPEATNAVTFDADDDSIIHLLHKGKPVSFELDKVFSPQASQQDV 61  IRV R+RP+ +++ E NA+ D+ ++ HL K D+VF A+Q DV Sbjct 2 IRVYCRLRPLCEKEIIAKE-RNAIR-SVDEFTVEHLWKDDKAKQHMYDRVFDGNATQDDV 59  Query 62 FQEVQALVTSCIDGFNVCIFAYGQTGAGKTYTMEGTAENPGINQRALQLLFSEVQEKASD 121  F++ + LV S +DG+NVCIFAYGQTG+GKT+T+ G NPG+ RA+ LF +++ ++  Sbjct 60 FEDTKYLVQSAVDGYNVCIFAYGQTGSGKTFTIYGADSNPGLTPRAMSELFRIMKKDSNK 119  Query 122 WEYTITVSAAEIYNEVLRDLLGKEPQEKLEIRLCPDGSGQLYVPGLTEFQVQSVDDINKV 181  + +++ E+Y + L DLL + ++L++ + D G + V +T + + +++ + Sbjct 120 FSFSLKAYMVELYQDTLVDLLLPKQAKRLKLDIKKDSKGMVSVENVTVVSISTYEELKTI 179  Query 182 FEFGHTNRTTEFTNLNEHSSRSHALLIVTVRGVDCSTGLRTTGKLNLVDLAGSERVGKGS 241  + G R T T +NE SSRSH ++ V + + T GKL+ VDLAGSERV K  Sbjct 180 IQRGSEQRHTTGTLMNEQSSRSHLIVSVIIESTNLQTQAIARGKLSFVDLAGSERVKK-- 237  Query 242 RLREAQHINKSLSALGDVIAALRSRQGHVPFRNSKLTYLLQDSLSGDSKTLMVVQVSPVE 301  EAQ INKSLSALGDVI+AL S H+P+RN KLT L+ DSL G++KTLM V +SP E Sbjct 238 ---EAQSINKSLSALGDVISALSSGNQHIPYRNHKLTMLMSDSLGGNAKTLMFVNISPAE 294  Query 302 KNTSETLYSLKFAERVR 318  N ET SL +A RVR Sbjct 295 SNLDETHNSLTYASRVR 311 |

I wrote a python script to automatically convert the output to PIR format:

|  |
| --- |
| **import** sys **import** os **import** re  inp = open(sys.argv[1]) out = open(sys.argv[2], 'w')  name1 = '' name2 = '' method = 'structureX' pfxname = '' s1starti = 1000000000 s2starti = 1000000000 s1endi = 0 s2endi = 0 shift = 0  seq1 = '' seq2 = ''  # find the nth first match. It will start with a > **for** line **in** inp.readlines():  **if** name2 == '':  r = re.match(r'^Query= (.+)$', line)  **if** r:  name2 = r.group(1)    **elif** name1 == '':  r = re.match(r'^> (.+)$', line)  **if** r:  name1 = r.group(1)    **else**:  r = re.match(r'^>.\*$', line)  **if** r:  **break** # we have found all the data    r = re.match(r'^(\S+)\s+(\d+)\s+(\S+)\s+(\d+)$', line)  **if** r:  t = r.group(1)  s = int(r.group(2))  p = r.group(3)  e = int(r.group(4))   **if** t == 'Query':  seq2 += p   s2starti = min(s2starti, s)  s2endi = max(s2endi, e)      **elif** t == 'Sbjct':  seq1 += p   s1starti = min(s1starti, s)  s1endi = max(s2endi, e)  out.write(">P1;{}\n".format(name1)) out.write("{}:{}: {}: : {}: : : : :\n".format(method, name1, s1starti, s1endi)) out.write(seq1 + "\*\n\n")  out.write(">P1;{}\n".format(name2)) out.write(" : : : : : : : : :\n") out.write(seq2 + "\*\n\n")  out.close() |

Sample output:

|  |
| --- |
| >P1;1SDMA structureX:1SDMA: 2: : 318: : : : : IRVYCRLRPLCEKEIIAKE-RNAIR-SVDEFTVEHLWKDDKAKQHMYDRVFDGNATQDDVFEDTKYLVQSAVDGYNVCIFAYGQTGSGKTFTIYGADSNPGLTPRAMSELFRIMKKDSNKFSFSLKAYMVELYQDTLVDLLLPKQAKRLKLDIKKDSKGMVSVENVTVVSISTYEELKTIIQRGSEQRHTTGTLMNEQSSRSHLIVSVIIESTNLQTQAIARGKLSFVDLAGSERVKK-----EAQSINKSLSALGDVISALSSGNQHIPYRNHKLTMLMSDSLGGNAKTLMFVNISPAESNLDETHNSLTYASRVR\*  >P1;bioinfo  : : : : : : : : : IRVIARVRPVTKEDGEGPEATNAVTFDADDDSIIHLLHKGKPVSFELDKVFSPQASQQDVFQEVQALVTSCIDGFNVCIFAYGQTGAGKTYTMEGTAENPGINQRALQLLFSEVQEKASDWEYTITVSAAEIYNEVLRDLLGKEPQEKLEIRLCPDGSGQLYVPGLTEFQVQSVDDINKVFEFGHTNRTTEFTNLNEHSSRSHALLIVTVRGVDCSTGLRTTGKLNLVDLAGSERVGKGSRLREAQHINKSLSALGDVIAALRSRQGHVPFRNSKLTYLLQDSLSGDSKTLMVVQVSPVEKNTSETLYSLKFAERVR\* |

Step 6:

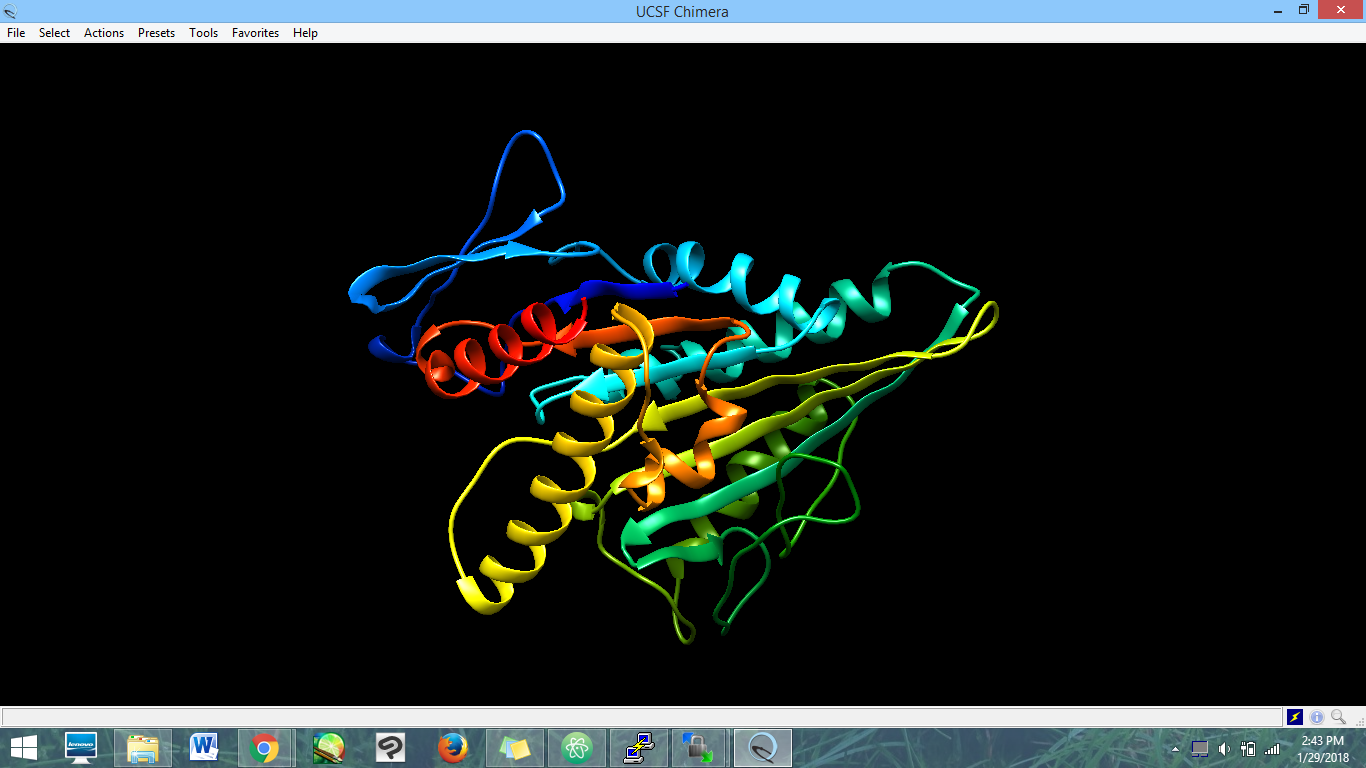
Use modeller the generate the bioinfo structure:



Picture above is structure from given example, but errors start to show up when we start solving the real problem. During the process we already generate all resources the modeller needs: the .atm file from code provide by Dr.Cao in simon and Format the .pir file from the PSI-BLAST result. But modeller still not able to generate the .pdb file cause some of the input format is not correct. One of our group member spend all the time try to solve this problem but it still could work. But we all agree that if our group got more time to finish this project we should able to solve this problem.

Step 7:

Structure for bioinfo from I-TASSER



MULTICOM, Robetta, and QUARK did not produce results.

For the 10 sequence prediction, we first tried using I-TASSER, but quickly realized that we would not get the results we needed in time. Next on the list was MULTICOM, since it allowed us to submit more than one sequence at a time. However, the results from MULTICOM are not public, so we cannot view them without an emailed link, and we found that there was either a problem with MULTICOM’s emails or the service simply takes more than two weeks, as we still have not received any results.

We eventually settled on Rosetta, the software used by the Robetta web service, since it could be easily downloaded onto Simon and process multiple jobs at once. The unforeseen difficulty with Rosetta was that it had very sparse instructions and required a lot of file manipulation, as well as additional files that had to be generated before the structure prediction. These “fragment files” were very difficult to generate using the Rosetta software because they required results from PSIPRED and the original PDB files for the sequences. So we instead chose to use the Robetta service to generate the fragment files, and found that it was actually faster than attempting to generate the files using Rosetta on Simon. This worked for all but three of the sequences: 5VOS, 5YUD, and 6F6I. The constraints for the Robetta fragment generator were that the fasta sequences had to be between 27 and 1000 residues; 5VOS was too short and 5YUD was too long, so for these we used Rosetta’s fragment picker with a full NR database import (354GB in size) to get the necessary files. This is the command used to generate fragment files:

|  |
| --- |
| screen -L ../../rosetta/tools/fragment\_tools/make\_fragments.pl fasta |

...where “fasta” was the file containing the FASTA sequence for the protien we are trying to fold. The “screen” command was used to allow for the application to run in the background for many hours while I was off the server.

6F6I was an interesting case, because the Robetta server crashed the day we had planned to run the sequence through it, and it did not recover until the monday night before the presentation. We also found, however, that for unknown reasons the Rosetta fragment picker was unable to generate the fragment files for this particular sequence. We were unable to get a structure prediction for 6F6I in time. The command we used to generate the final PDB files with Rosetta was:

|  |
| --- |
| screen -L ../../bin/AbinitioRelax.linuxgccrelease @../../options.txt |

… where options.txt was an argument file for Rosetta which looked like the following (see comments for info on how this works):

|  |
| --- |
| # Make sure all variable names have been replaced with absolute path and that no line begins with a $ or ~s -**in**  -file # -**native** 3lpt.pdb # **native** PDB file (optional)  -fasta ../fasta # protein sequence **in** fasta format  -frag3 ../t001\_.200.3mers # protein 3-residue fragments file  -frag9 ../t001\_.200.9mers # protein 9-residue fragments file -abinitio  -relax  -increase\_cycles 10 # Increase the number of cycles at each stage **in** AbinitioRelax **by** **this** factor  -rg\_reweight 0.5 # Reweight contribution of radius of gyration to total score **by** **this** scale factor  -rsd\_wt\_helix 0.5 # Reweight env, pair, and cb scores **for** helix residues **by** **this** factor  -rsd\_wt\_loop 0.5 # Reweight env, pair, and cb scores **for** loop residues **by** **this** factor -relax  -fast # At the end of the de novo protein\_folding, **do** a relax step of type "FastRelax". This has been shown to be the best deal **for** speed and robustness. -**out**  -pdb  -nstruct 1000 # how many structures **do** you want to generate? Usually want to fold at least 1,000.  -file  -scorefile ../score.sc -overwrite # overwrite any existing output with the same name you may have generated  -nstruct 1 |

Our final RMSD scores are:

5ubw: 4.62

5uzl: 2.94

5vos: 1.56

5w7b: 4.4

5w7c: 4.92

5wb3: 5.37

5wlj: 1.6

5yud: 9.63

5yw2: 4.55

6f6i: N/A

Average: 4.398

Normed Average: 8.959

**Our presentation slides are** [**here.**](https://docs.google.com/presentation/d/11EbmF4z8Y53U2kzw6nwGk8iFn8q91aJmNBRJ46oQ9Fk/edit?usp=sharing)