The Neotia University

THE NEOTIA UNIVERSITY
ज्ञानम् आत्म प्रदीपाय

M.Sc Bioinformatics
Practical Manual
Course No:
2021

Mr. Ranojit Kumar Sarker

Department of Biotechnology

The Neotia University

CONTENTS

Experiment 1 & 2: Secondary Structure Analysis of Protein

Aim : To predict secondary structure of the give protein sequences

Introduction:

Protein secondary structure includes the regular polypeptide folding patterns such as helices, sheets, and turns. The backbone or main chain of a protein refers to the atoms that participate in peptide bonds, ignoring the side chains of the amino acid. The conformation of the backbone can therefore be described by the torsion angles (also called dihedral angles or rotation angles) around the Phi and the Psi of each residue. The helix structure looks like a spring. The most common shape is a right handed a-helix defined by the repeat length of 3.6 amino acid residues and a rise of 5.4 Å per turn.

Secondary structure in proteins consists of local inter-residue interactions mediated by hydrogen bonds, or not. The most common secondary structures are alpha helices and beta sheets. Other helices, such as the 310 helix and π helix, are calculated to have energetically favorable hydrogen-bonding patterns but are rarely if ever observed in natural proteins except at the ends of α helices due to unfavorable backbone packing in the center of the helix. Other extended structures such as the polyproline helix and alpha sheet are rare in native state proteins but are often hypothesized as important protein folding intermediates. Tight turns and loose, flexible loops link the more "regular" secondary structure elements. The random coil is not a true secondary structure, but is the class of conformations that indicate an absence of regular secondary structure.

Amino acids vary in their ability to form the various secondary structure elements. Proline and glycine are sometimes known as "helix breakers" because they disrupt the regularity of the α helical backbone conformation; however, both have unusual conformational abilities and are commonly found in turns. Amino acids that prefer to adopt helical conformations in proteins include methionine, alanine, leucine, glutamate and lysine ("MALEK" in amino-acid 1-letter codes); by contrast, the large aromatic residues (tryptophan, tyrosine and phenylalanine) and Cβ-branched amino acids (isoleucine, valine, and threonine) prefer to adopt β-strand conformations. However, these preferences are not strong enough to produce a reliable method of predicting secondary structure from sequence alone.

There are several methods for defining protein secondary structure (e.g. DEFINE, DSSP, STRIDE (protein)).

Structural features of the three major forms of protein helices

Geometry attribute	α-helix	3_{10} helix	π-helix
Residues per turn	3.6	3.0	4.4
Translation per residue	1.5 Å (0.15 nm)	2.0 Å (0.20 nm)	1.1 Å (0.11 nm)
Radius of helix	2.3 Å (0.23 nm)	1.9 Å (0.19 nm)	2.8 Å (0.28 nm)
Pitch	5.4 Å (0.54 nm)	6.0 Å (0.60 nm)	4.8 Å (0.48 nm)

1. To Compare the secondary structures of the following sequences and comment on the result.

>1
MGLSDGEWQLVLNVWGKVEADIPGHGQEVLIRLFKGHPETLEKFDKFKHLKSEDEMKA
SEDLKKHGATVLTALGGILKKKGHHEAEIKPLAQSHATKHKIPVKYLEFISECIIQVLQSK
HPGDFGADAQGAMNKALELFRKDMASNYKELGFQG
>2
MDPKQTTLLCLVLCLGQRIQAQEGDFPMPFISAKSSPVIPLDGSVKIQCQAIREAYLTQL
MIIKNSTYREIGRRLKFWNETDPEFVIDHMDANKAGRYQCQYRIGHYRFRYSDTLELVVT
GLYGKPFLSADRGLVLMPGENISLTCSSAHIPFDRFSLAKEGELSLPQHQSGEHPANFSL
GPVDLNVSGIYRCYGWYNRSPYLWSFPSNALELVVTDSIHQDYTTQNLIRMAVAGLVLV
ALLAILVENWHSHTALNKEASADVAEPSWSQQMCQPGLTFARTPSVCK

Methods:

1. Take the sequence from uniprot or copy the sequence if already given

2. Go to http://www.compbio.dundee.ac.uk/www-jpred/

3. Paste the sequence and click on make prediction

4. Wait for the software to predict the structure

5. Once Job is done. Save the output.

Results:

Output for seq 1 and 2:

Colour code for alignment:

Blue - Complete identity at a position

Shades of red - The more red a position is, the higher the level of conservation of chemical properties of the amino acids

Jnet - Final secondary structure prediction for query

jalign - Jnet alignment prediction

jhmm - Jnet hmm profile prediction

jpssm - Jnet PSIBLAST pssm profile prediction

Lupas - Lupas Coil prediction (window size of 14, 21 and 28)

Note on coiled coil predictions - = less than 50% probability
c = between 50% and 90% probability
C = greater than 90% probability
Jnet_25 - Jnet prediction of burial, less than 25% solvent accesibility
Jnet_5 - Jnet prediction of burial, less than 5% exposure
Jnet_0 - Jnet prediction of burial, 0% exposure
Jnet Rel - Jnet reliability of prediction accuracy, ranges from 0 to 9, bigger is better.

Sequence 1

2. Predict the secondary structure composition of O13837.

Methods:

1. Take the sequence from uniprot or copy the sequence if already given

2. Go to http://www.compbio.dundee.ac.uk/www-jpred/
3. Paste the sequence and click on make prediction
4. Wait for the software to predict the structure
5. Once Job is done . Save the output.

Results :

3. Find the secondary structure of the given sequence and compare with the output of 2.

Method:

1. Run Blastx to determine protein.

2. Predict Secondary Structure.

3. Go to http://www.compbio.dundee.ac.uk/www-jpred/

4. Paste the sequence and click on make prediction

5. Wait for the software to predict the structure

6. Once Job is done . Save the output.

>
ATGTCTTCTACTGCCACCGTTACTGAAAGCACTCATTTTTTTCCCAATGAGCCTCAAGGCCCTAGCATTA
AGACCGAAACTATTCCCGGTCCCAAAGGTAAGGCCGCTGCTGAAGAAATGTCCAAATACCACGACATC
AG
CGCTGTCAAGTTTCCTGTAGACTATGAAAAGTCCATTGGTAACTATCTTGTCGACTTGGATGGTAACGTT
CTCTTGGATGTTTACTCTCAAATCGCTACTATCCCCATTGGCTACAACAATCCTACTCTCCTCAAGGCTG
CCAAGTCGGACGAAGTCGCTACCATTTTAATGAACCGTCCTGCTTTGGGAAATTACCCTCCTAAGGAAT
G
GGCTCGTGTCGCTTATGAGGGTGCCATCAAATATGCCCCCAAGGGTCAAAAGTATGTTTACTTTCAAAT
G
AGTGGAAGTGATGCCAACGAGATTGCTTACAAGCTTGCTATGCTTCATCATTTCAACAACAAGCCTAGA
C CTACTGGTGATTACACTGCTGAAGAGAACGAGAGCTGCTTAAACAACGCTGCTCCTGGATCTCCCGAAG
T
TGCTGTTCTCTCTTTCCGTCACTCTTTCCACGGACGTCTCTTTGGTTCTCTTTCCACTACTCGCTCCAAG
CCTGTTCACAAGCTTGGTATGCCTGCTTTCCCATGGCCTCAAGCTGATTTCCCTGCTTTGAAGTATCCTT
TGGAAGAGCACGTCGAAGAGAATGCAAAGGAGGAGCAACGCTGCATTGACCAGGTCGAGCAAATTTTA
AC TAACCACCATTGCCCTGTCGTTGCCTGTATCATTGAGCCCATTCAATCTGAGGGTGGTGACAACCATGCC
TCTCCTGACTTTTTCCACAAGCTTCAAGCTACTTTGAAGAAGCATGATGTCAAGTTTATCGTCGATGAAG
TCCAAACTGGTGTCGGCTCTACCGGTACTTTATGGGCTCACGAGCAATGGAATTTACCCTATCCTCCTGA
CATGGTTACCTTTTCCAAGAAATTCCAGGCTGCCGGTATTTTCTATCATGATTTGGCTCTTCGTCCTCAT
GCTTATCAGCACTTCAATACTTGGATGGGTGACCCATTCCGTGCTGTTCAATCTAGATATATTCTTCAAG

AAATTCAAGACAAGGATCTCCTTAATAACGTCAAGTCTGTTGGCGATTTCTTGTATGCTGGACTTGAAG
A
GCTTGCTCGTAAGCACCCTGGCAAAATCAACAACCTCCGCGGTAAGGGAAAGGGTACTTTTATCGCTTG G
GATTGTGAGTCTCCTGCAGCCCGTGACAAATTCTGTGCTGACATGAGAATTAATGGTGTCAACATTGGT
G
GCTGTGGTGTAGCTGCTATTCGTCTTCGTCCTATGCTTGTATTCCAAAAGCACCATGCTCAAATCCTTCT
CAAGAAGATTGACGAATTGATTTA

Results:

Inference:
Schizosaccharomyces pombe chromosome I, complete replicon

Length=5579133

Features in this part of subject sequence:

4-aminobutyrate aminotransferase (GABA transaminase)

Score = 2632 bits (1425), Expect = 0.0

Identities = 1425/1425 (100%), Gaps = 0/1425 (0%)

Strand=Plus/Plus

Corresponding protein: O13837: 4-aminobutyrate aminotransferase

>gi|6016100|sp|O13837.1|GABAT_SCHPO RecName: Full=4-aminobutyrate aminotransferase; AltName: Full=GABA aminotransferase; Short=GABA-AT; AltName: Full=Gamma-amino-N-butyrate transaminase; Short=GABA transaminase

Secondary structures are the same.

Experiment 3: Tertiary Structure Analysis

Aim: Determine the 3d structure of human gaba transaminase using homology modeling

Introduction:

The tertiary structure of a protein or any other macromolecule is its three-dimensional structure, as defined by the atomic coordinates. Tertiary structure is considered to be largely determined by the protein's primary structure - the sequence of amino acids of which it is composed. Efforts to predict tertiary structure from the primary structure are known generally as protein structure prediction. However, the environment in which a protein is synthesized and allowed to fold are significant determinants of its final shape and are usually not directly taken into account by current prediction methods. Most such methods do rely on comparisons between the sequence to be predicted and sequences of known structure in the Protein Data Bank and thus account for environment indirectly, assuming the target and template sequences share similar cellular contexts. In globular proteins, tertiary interactions are frequently stabilized by the sequestration of hydrophobic amino acid residues in the protein core, from which water is excluded, and by the consequent enrichment of charged or hydrophilic residues on the protein's water-exposed surface. In secreted proteins that do not spend time in the cytoplasm, disulfide bonds between cysteine residues help to maintain the protein's tertiary structure. A variety of common and stable tertiary structures appear in a large number of proteins that are unrelated in both function and evolution - for example, many proteins are shaped like a TIM barrel, named for the enzyme triosephosphateisomerase. Another common structure is a highly stable dimeric coiled coil structure composed of 2-7 alpha helices. Proteins are classified by the folds they represent in databases like SCOP and CATH.

Homology modeling, also known as comparative modeling of protein refers to constructing an atomic-resolution model of the "target" protein from its amino acid sequence and an experimental three-dimensional structure of a related homologous protein (the "template"). Homology modeling relies on the identification of one or more known protein structures likely to resemble the structure of the query sequence, and on the production of an alignment that maps residues in the query sequence to residues in the template sequence. It has been shown that protein structures are more

conserved than protein sequences amongst homologues, but sequences falling below a 20% sequence identity can have very different structure. Homology modeling aims to build three-dimensional protein structure models using experimentally determined structures of related family members as templates. SWISS-MODEL workspace is an integrated Web-based modeling expert system. For a given target protein, a library of experimental protein structures is searched to identify suitable templates. On the basis of a sequence alignment between the target protein and the template structure, a three-dimensional model for the target protein is generated. Model quality assessment tools are used to estimate the reliability of the resulting models. Homology modeling is currently the most accurate computational method to generate reliable structural models and is routinely used in many biological applications. Typically, the computational effort for a modeling project is less than 2 h. However, this does not include the time required for visualization and interpretation of the model, which may vary depending on personal experience working with protein structures.

Swiss PDB viewer and swiss modeler are used as homology modeling software and workspace.

Swiss-Pdb Viewer provides a user friendly interface allowing to analyze several proteins at the same time.

1. Superimposition - structural alignments and compare their active sites or any other relevant parts

2. . Make amino acid mutations

3. Generate Hydrogen bonds

4. Calculate angles and distances between atoms

5. Tightly linked to Swiss-Model, an automated homology modeling server

6. Thread a protein primary sequence onto a 3D template

7. Build missing loops and refine sidechain packing

8. Read electron density maps and build into the density

9. Perform energy minimization

10. POV-Ray scenes can be generated for stunning ray-traced quality images

Swiss Modeller

The SWISS-MODEL Workspace is a web-based integrated service dedicated to protein structure homology modelling. It assists and guides the user in building protein homology models at different levels of complexity.

Successful model building requires at least one experimentally determined 3D structure (template) that shows significant amino acid sequence similarity with the target sequence. Building a homology model comprises four main steps: identification of structural template(s), alignment of target sequence and template structure(s), model building, and model quality evaluation. These steps can be repeated until a satisfying modelling result is achieved. Each of the four steps requires specialized software and access to up-to-date protein sequence and structure databases.

Protein sequence and structure databases necessary for modelling are accessible from the workspace and are updated in regular intervals. Software tools for template selection, model building, and structure quality evaluation can be invoked from within the workspace. A personal working environment (workspace), where several modelling projects can be carried out in parallel, is provided for each user.

Methods:

1. load the 1OHV protein

2. select the chain A, in control panel and in the menu bar click the bulid option and select the inverse selection and then click on the remove selected residues.

3. save it separately as 1OHVA.pdb

4. open the empty window again, and click the swissmodel to load the raw sequence.

5. open the pdb file through the import structures in the "File" menubar.

6. Click the magic fit, iterative magic fit from Fit option in the menubar.

7. Open the alignment window from the wind and select the residues which are not aligned.

8. Delete the residues which are not aligned using the Build option in the menubar and click the remove residues and save it.

9. Now submit this to the swiss modelling request for the raw

10. Download the modelled protein and open in the swiss viewer.

11. In Bulid option, click the energy minimization.

12. open the seq-structure aligned protein (step 8) and energy minimized protein in the viewer and click the improve fit

13. Calculate the RMS value from Fit option

14. Render the model in 3D view.

15. Use Protein Structure & Model Assessment Tools for analyzing the protein.

Result and Inference:

Query sequence (gabat.txt)

>sp|P80404|GABT_HUMAN 4-aminobutyrate aminotransferase, mitochondrial OS=Homo sapiens GN=ABAT PE=1 SV=3

MASMLLAQRLACSFQHSYRLLVPGSRHISQAAAKVDVEFDYDGPLMKTEVPGPRSQELM
KQLNIIQNAEAVHFFCNYEESRGNYLVDVDGNRMLDLYSQISSVPIGYSHPALLKLIQQPQ
NASMFVNRPALGILPPENFVEKLRQSLLSVAPKGMSQLITMACGSCSNENALKTIFMWYR
SKERGQRGFSQEELETCMINQAPGCPDYSILSFMGAFHGRTMGCLATTHSKAIHKIDIPSF
DWPIAPFPRLKYPLEEFVKENQQEEARCLEEVEDLIVKYRKKKKTVAGIIVEPIQSEGGDN
HASDDFFRKLRDIARKHGCAFLVDEVQTGGGCTGKFWAHEHWGLDDPADVMTFSKKM
MTGGFFHKEEFRPNAPYRIFNTWLGDPSKNLLLAEVINIIKREDLLNNAAHAGKALLTGL
LDLQARYPQFISRVRGRGTFCSFDTPDDSIRNKLILIARNKGVVLGGCGDKSIRFRPTLVFR
DHHAHLFLNIFSDILADFK

Sequences producing significant alignments:

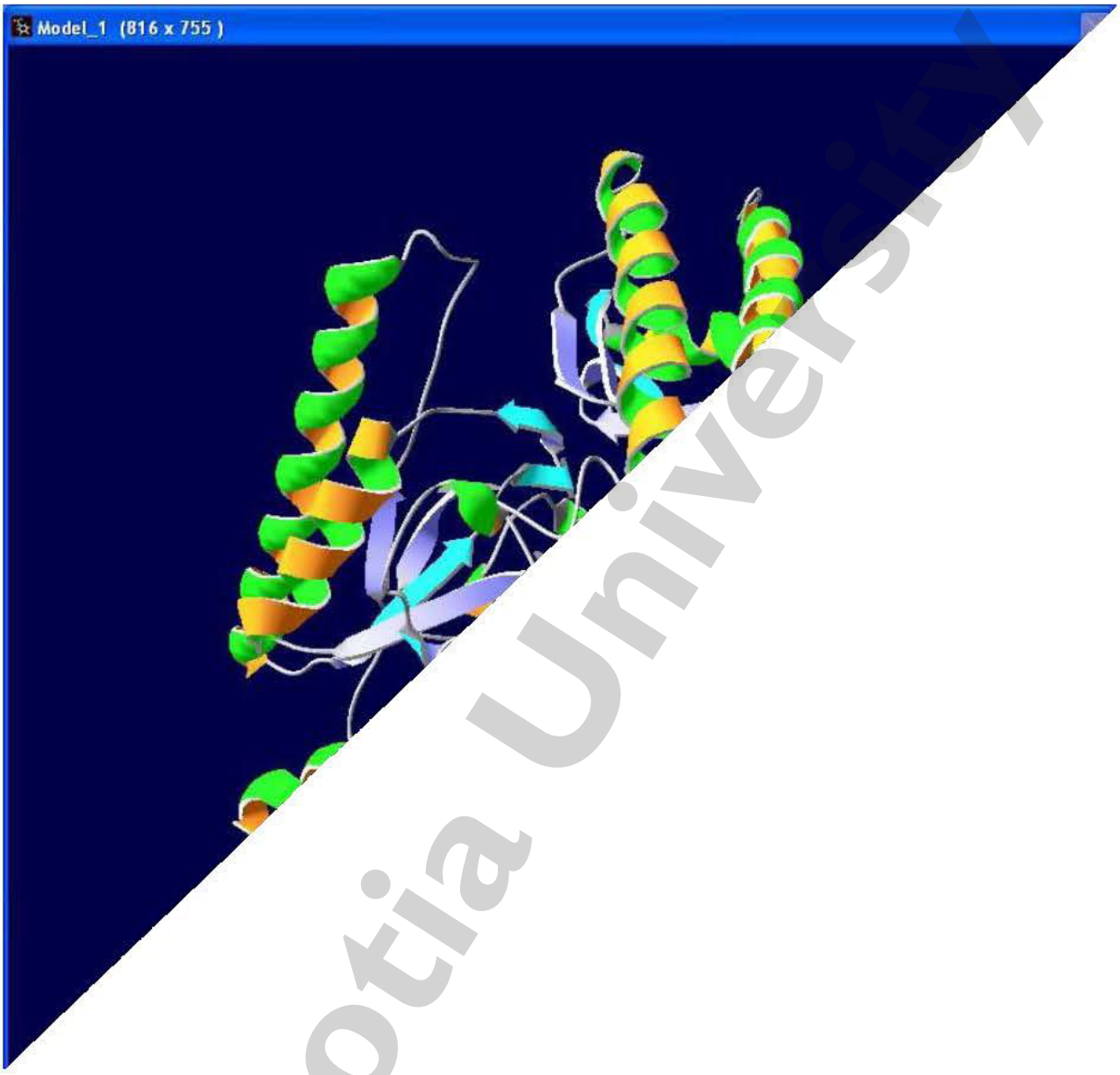
Accession	Description	Max score
1OHV A	Chain A, 4-Aminobutyrate-Aminotransferase From Pig	
2JJF A	Chain A, N328a Mutant Of M	
2CIN A		

>pdb|1OHV|A Chain A, 4-Aminobutyrate-Aminotransferase From Pig
pdb|1OHV|B Chain B, 4-Aminobutyrate-Aminotransferase From Pig
pdb|1OHV|C Chain C, 4-Aminobutyrate-Aminotransferase From Pig
▶9 more sequence titles
Length=472

Score = 959 bits (2479), Expect = 0.0, Method: Compositional matrix adjust.
Identities = 453/472 (96%), Positives = 464/472 (98%), Gaps = 0/472 (0%)

```
Query  29   SQAAAKVDVEFDYDGPLMKTEVPGPRSQELMKQLNIIQNAEAVHFFCNYEESRGNYLVDV  88
            SQAAAKVDVEFDYDGPLMKTEVPGPRS+ELMKQLNIIQNAEAVHFFCNYEESRGNYLVDV
Sbjct  1    SQAAAKVDVEFDYDGPLMKTEVPGPRSRELMKQLNIIQNAEAVHFFCNYEESRGNYLVDV  60

Query  89   DGNRMLDLYSQISSVPIGYSHPALLKLIQQPQNASMFVNRPALGILPPENFVEKLRQSLL  148
            DGNRMLDLYSQISS+PIGYSHPAL+KL+QQPQN S F+NRPALGILPPENFVEKLR+SLL
Sbjct  61   DGNRMLDLYSQISSIPIGYSHPALVKLVQQPQNVSTFINRPALGILPPENFVEKLRESLL  120

Query  149  SVAPKGMSQLITMACGSCSNENALKTIFMWYRSKERGQRGFSQEELETCMINQAPGCPDY  208
            SVAPKGMSQLITMACGSCSNENA KTIFMWYRSKERGQ  FS+EELETCMINQAPGCPDY
Sbjct  121  SVAPKGMSQLITMACGSCSNENAFKTIFMWYRSKERGQSAFSKEELETCMINQAPGCPDY  180

Query  209  SILSFMGAFHGRTMGCLATTHSKAIHKIDIPSFDWPIAPFPRLKYPLEEFVKENQQEEAR  268
            SILSFMGAFHGRTMGCLATTHSKAIHKIDIPSFDWPIAPFPRLKYPLEEFVKENQQEEAR
Sbjct  181  SILSFMGAFHGRTMGCLATTHSKAIHKIDIPSFDWPIAPFPRLKYPLEEFVKENQQEEAR  240

Query  269  CLEEVEDLIVKYRKKKKTVAGIIVEPIQSEGGDNHASDDFFRKLRDIARKHGCAFLVDEV  328
            CLEEVEDLIVKYRKKKKTVAGIIVEPIQSEGGDNHASDDFFRKLRDI+RKHGCAFLVDEV
Sbjct  241  CLEEVEDLIVKYRKKKKTVAGIIVEPIQSEGGDNHASDDFFRKLRDISRKHGCAFLVDEV  300

Query  329  QTGGGCTGKFWAHEHWGLDDPADVMTFSKKMMTGGFFHKEEFRPNAPYRIFNTWLGDPSK  388
            QTGGG TGKFWAHEHWGLDDPADVMTFSKKMMTGGFFHKEEFRPNAPYRIFNTWLGDPSK
Sbjct  301  QTGGGSTGKFWAHEHWGLDDPADVMTFSKKMMTGGFFHKEEFRPNAPYRIFNTWLGDPSK  360

Query  389  NLLLAEVINIIKREDLLNNAAHAGKALLTGLLDLQARYPQFISRVRGRGTFCSFDTPDDS  448
            NLLLAEVINIIKREDLL+NAAHAGK LLTGLLDLQARYPQFISRVRGRGTFCSFDTPD+S
Sbjct  361  NLLLAEVINIIKREDLLSNAAHAGKVLLTGLLDLQARYPQFISRVRGRGTFCSFDTPDES  420

Query  449  IRNKLILIARNKGVVLGGCGDKSIRFRPTLVFRDHHAHLFLNIFSDILADFK  500
            IRNKLI IARNKGV+LGGCGDKSIRFRPTLVFRDHHAHLFLNIFSDILADFK
Sbjct  421  IRNKLISIARNKGVMLGGCGDKSIRFRPTLVFRDHHAHLFLNIFSDILADFK  472
```

Gabat.txt and 1OHVA.pdb Modeled Structure at swisspdb viewer and swiss modeler



Energy minimization score: -26789.707

RMSD: 0.07A

Quality information:

QMEAN Z-Score: -1.129

Ligand information:

Global Model Quality Estimation:

QMEAN4 global scores:

Local scores:

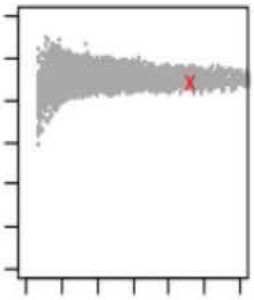
QMEANscore4:

Estimated absolute model quality:

Score components:

Coloring by resid error:

0.695



Z-Score: -1.129

QMEAN4 global scores:

The QMEAN4 score is a composit potential terms (estimated mo contributing terms are give for high-resolution exp

Scoring function f

C_beta inter

All-ator

Sc

Procheck: [+/-]

```
 +----------<<<  P  R  O  C  H  E  C  K     S  U  M  M  A  R  Y  >>>----------+
 |                                                                             |
 | input_atom_only.pdb   2.5                                      461 residues |
 |                                                                             |
*| Ramachandran plot:   88.8% core   10.2% allow    0.5% gener    0.5% disall |
 |                                                                             |
+| All Ramachandrans:   10 labelled residues (out of 459)                      |
+| Chi1-chi2 plots:      2 labelled residues (out of 296)                      |
 | Main-chain params:    6 better     0 inside     0 worse                     |
 | Side-chain params:    5 better     0 inside     0 worse                     |
 |                                                                             |
+| Residue properties: Max.deviation:    11.0            Bad contacts:    1    |
+|                     Bond len/angle:    4.4    Morris et al class:  1  1  2  |
+|     3 cis-peptides                                                          |
 | G-factors           Dihedrals:  -0.02  Covalent:   0.42    Overall:   0.15 |
 |                                                                             |
 | M/c bond lengths:100.0% within limits   0.0% highlighted                    |
 | M/c bond angles:  99.6% within limits   0.4% highlighted                    |
*| Planar groups:    89.5% within limits  10.5% highlighted       1 off graph |
 |                                                                             |
 +-----------------------------------------------------------------------------+
   + May be worth investigating further.  * Worth investigating further.
```

The qmean score(-1.129) and procheck (rc plot : 99.5% in allowed region)score were within ranges proving protein structure as stable.

Homology Modeling Using Modeller

AIM: To do homology modeling for human gaba transaminase using MODELLER.

Introduction:

MODELLER is used for homology or comparative modeling of protein three-dimensional structures. The user provides an alignment of a sequence to be modeled with known related

structures and MODELLER automatically calculates a model containing all non-hydrogen atoms. MODELLER implements comparative protein structure modeling by satisfaction of spatial restraints and can perform many additional tasks, including de novo modeling of loops in protein structures, optimization of various models of protein structure with respect to a flexibly defined objective function, multiple alignment of protein sequences and/or structures, clustering, searching of sequence databases, comparison of protein structures, etc. MODELLER is available for download for most Unix/Linux systems, Windows, and Mac.

MODELLER is used for homology or comparative modeling of protein three-dimensional structures. The user provides an alignment of a sequence to be modeled with known related structures and MODELLER automatically calculates a model containing all non-hydrogen atoms.There are 5 modeling examples that the user can follow:

Basic Modeling. Model a sequence with high identity to a template.This exercise introduces the use of MODELLER in a simple case where the template selection and target-template alignments are not a problem.

Advanced Modeling. Model a sequence based on multiple templates and bound to a ligand.This exercise introduces the use of multiple templates, ligands and loop refinement in the process of model building with MODELLER.

Iterative Modeling. Increase the accuracy of the modeling exercise by iterating the 4 step process.This exercise introduces the concept of MOULDING to improve the accuracy of comparative models.

Difficult Modeling. Model a sequence based on a low identity to a template.This exercise uses resources external to MODELLER in order to select a template for a difficult case of protein structure prediction.

Modeling with cryo-EM. Model a sequence using both template and cryo-EM data.This exercise assesses the quality of generated models and loops by rigid fitting into cryo-EM maps, and improves them with flexible EM fitting.

Method:

1. Take query sequence whose structure needs to be modelled (e.g gabat) in PIR format.

2. Save the file with .ali extension in the bin folder of modeller.

3. Open build_profile.py file. Change the append filename to the query sequence(gabat.ali).

4. Open the command line by clicking the 'Modeller' link from the Start Menu in Windows.

5. Run the build_profile.py.This will search for potentially related sequences of known structure. Two files are created build_profile_gabat.ali file and build_profile_gabat.prf file.

6. Open the build_profile.prf file and select the sequences which has an e value 0.0 .

7. Download the structures of the selected protein from the PDB and save it in bin folder of modeller.

8. Open the compare.py file.Write the the name of the selected proteins.

9. Run compare.py command in command line. A compare.log output file is created.

10. Choose the sequence with high resolution and moderate identity.

11. Align the query sequence with the template by using align2d command.

12. Two output files are created .pap file and .ali file.

13. Open model_single.py file .Use the above created .ali file .Run the model_single.py command in the command line.

14. 5 possible models are generated .Select the best model which has the lowest dope score.

15. Run evaluate_model.py command for evaluating the selected model.Note the Dope score.

16. Run evaluate_template.py command for evaluating the template. Note the Dope score.

17. Compare the dope score of both model and template.

Results and Inference:

Build_profile_gabat.ali (output for build_profile.py)

```
>P1;gabat
sequence:gabat: 0: : 0: :::-1.00:-1.00
MASMLLAQRLACSFQHSYRLLVPGSRHISQAAAKVDVEFDYDGPLMKTEVPGPRSQELMKQLNIIQNAEAV
HFFC
```

```
NYEESRGNYLVDVDGNRMLDLYSQISSVPIGYSHPALLKLIQQPQNASMFVNRPALGILPPENFVEKLRQS
LLSV
APKGMSQLITMACGSCSNENALKTIFMWYRSKERGQRGFSQEELETCMINQAPGCPDYSILSFMGAFHGRT
MGCL
ATTHSKAIHKIDIPSFDWPIAPFPRLKYPLEEFVKENQQEEARCLEEVEDLIVKYRKKKKTVAGIIVEPIQ
SEGG
DNHASDDFFRKLRDIARKHGCAFLVDEVQTGGGCTGKFWAHEHWGLDDPADVMTFSKKMMTGGFFHKEEFR
PNAP
YRIFNTWLGDPSKNLLLAEVINIIKREDLLNNAAHAGKALLTGLLDLQARYPQFISRVRGRGTFCSFDTPD
DSIR
NKLILIARNKGVVLGGCGDKSIRFRPTLVFRDHHAHLFLNIFSDILADFK*


>P1;2oatA
structure:2oatA: 28: : 404: :::-1.00:-1.00
-------------------------------------------------------------------------------
-----
--ERGKGIYLWDVEGRKYFDFLSSYSAVNQGHCHPKIVNALKSQVDKLTLTSRAVLG--EYEEYITKL---
-----
--FNYHKVLPMNTGVEAGETACKLARKW---------GYTVKGIQKYKA---------
KIVFAAGNFWGRTLSAI
SSS-------TDPTSYD-GFGPF----MPGFDIIPYND------LPALERAL-----
QDPNVAAFMVEPIQGEAG
VVVPDPGYLMGVRELCTRHQVLFIADEIQTGLARTGRWLAVDYENV--RPDIVLLG-
KALSGGLYDDDIMLTIKP
GEHGSTYGGNPLGCRVAIAALEVLEEENLAENADKLGIILRNELMKLPS---
DVVTAVRGKGLLNAIVIKEDWDA
WKVCLRLRDNGLLAKPTHGDIIRFAPPLVIKEDELRESIEIINKTILSF-*

>P1;1d7uA
structure:1d7uA: 28: : 427: ::-1.00:-1.00
-------------------------------------------------------------------------------
-----
--ERAKGSFVYDADGRAILDFTSGQMSAVLGHCHPEIVSVIGEYAGKSGMLSRP----------
VVDLATRLANI
TPPGLDRALLLSTGAESNEAAIR----------------------MAKLVTG--
KYEIVGFAQSWHGMTGAAA
SATYSKGVGPAAVGSFAIP-APFPR-------FERNGAYDYLAELDYAFDLI--
DRQSSGNLAAFIAEPILSSGG
IIELPDGYMAALKRKCEARGMLLILDEAQTGVGRTGTMFACQRDGV—
TPDILTLSKTLGAGTSAAIEERAHELG

YLFYTTHVSDPLPAAVGLRVLDVVQRDGLVARANVMGDRLRRGLLDLMERF-
DCIGDVRGRGLLLGVEEPADGLG
AKITRECMNLGVQLPGMGG-VFRIAPPLTVSEDEIDLGLSLLGQAI--- *


>P1;1s0aA
structure:1s0aA: 32: : 261: :::-1.00:-1.00
-------------------------------------------------------------------------------
-----
----AEGCELILSDGRRLVDGMSSWWAAIHGYNHPQLNAAMKSQIDAMSHVMFGGITHAP----
AIELCRKLVAM
```

```
TPQPLECVFLADSGSVAVEVAMKMALQYWQAKGEARQRF----------------------
LTFRNGYHGDTFGAM
SVCDDNSMHSL------WKFAPAPQSR--MGEWDERDMVGFAR-------LMAAHRHE---
IAAVIIEPIQGAGG
MRMYHPEWLKRIRKICDREGILLIADEIATGFGRTGKLFACEH--------------------------
------
------------------------------------------------------------------------------------------
------
-------------------------------------------------------------------*

>P1;2gsaA
structure:2gsaA: 38: : 338: :::-1.00:-1.00
------------------------------------------------------------------------------------------
------
-FDRVKDAYAWDVDGNRYIDYVGTWGPAICGHAHPEVIEALKVAMEKGTSFGAPC----
ALENLAEMVNDAVPSI
E----MVRFVNSGTEACM----AVLRLMRAYTGRDK-----------------------------
IIKFEGCYHGHADMFL
VKAGS-GVATLGLPSS--PGVP-----------
KKTTANTLTTPYNDLEAVKALFAENPGEIAGVILEPIVGNSG
FIVPDAGFLEGLREITLEHDALLVFDEVMTGGGVQEKFGV--------
TPDLTTLGKGLPVGAYGGKREIAPAGP
MYQAGTLSGNPLAMTAGIKTLELLRQPGTYEYLDQITKRLSDGLL--------------------------
------
-------------------------------------------------------------*

>P1;1ohvA
structure:1ohvA: 1: : 461: :::-1.00:-1.00
-----------------------------------------------
FDYDGPLMKTEVPGPRSRELMKQLNIIQNAEAVHFFC
NYEESRGNYLVDVDGNRMLDLYSQISSIPIGYSHPALVKLVQQPQNVSTFINRPALGILPPENFVEKLRES
LLSV
APKGMSQLITMACGSCSNENAFKTIFMWYRSKERGQSAFSKEELETCMINQAPGCPDYSILSFMGAFHGRT
MGCL
ATTHSKAIHKIDIPSFDWPIAPFPRLKYPLEEFVKENQQEEARCLEEVEDLIVKYRKKKKTVAGIIVEPIQ
SEGG
DNHASDDFFRKLRDISRKHGCAFLVDEVQTGGGSTGKFWAHEHWGLDDPADVMTFSKKMMTGGFFHKEEFR
PNAP
YRIFNTWLGDPSKNLLLAEVINIIKREDLLSNAAHAGKVLLTGLLDLQARYPQFISRVRGRGTFCSFDTPD
ESIR
NKLISIARNKGVMLGGCGDKSIRFRPTLVFRDHHAHLFLNIFSDILADF-*

>P1;1sffA
structure:1sffA: 36: : 424: :::-1.00:-1.00
------------------------------------------------------------------------------------------
------
----------DVEGREYLDFAGGIAVLNTGHLHPKVVAAVEAQLKK---
LSHTCFQVLAYEPYLELCEIMNQKV
PGDFAKKTLLVTTGSEAVENAVKI------ARAATKRS--------------------
GTIAFSGAYHGRTHYTL
ALT-----GKVNPYSAGMGLMPVYRALYPCP--LHGISEDDA--IASIH-
RIFKNDAAPEDIAAIVIEPVQGEGG
FYASSPAFMQRLRALCDEHGIMLIADEVQSGAGRTGTLFAMEQMGV--APDLTTFAKS-
IAGGFGRAEVMDAVAP
GGLGGTYAGNPIACVAALEVLKVFEQENLLQKANDLGQKLKDGLLAIAEKHPE-
IGDVRGLGAMIAIELFEDGDH
```

NKIVARARDKGLILLSCGPNVLRILVPLTIEDAQIRQGLEIISQCFDEAK*

Compare.log (output for compare.py)

```
Sequence identity comparison (ID_TABLE):

   Diagonal       ... number of residues;
   Upper triangle ... number of identical residues;
   Lower triangle ... % sequence identity, id/min(length).

         2oatA @21d7uA @21s0aA @12gsaA @21ohvA @21sffA @1
2oatA @2      404     108      93      84      76     112
1d7uA @2       27     431      86      76      86     117
1s0aA @1       23      20     427      79      72     107
2gsaA @2       21      18      19     427      63      97
1ohvA @2       19      20      17      15     461     102
1sffA @1       28      28      25      23      24     425

Weighted pair-group average clustering based on a distance matrix:

                                                    .--- 2oatA @2.0    72.0000
                                                    |
                                                .----- 1sffA @1.9      72.5000
                                                |
                                        .---------------------------- 1d7uA @2.0    78.0000
                                        |
                                .------------------------------------ 1s0aA @1.7    80.5000
                                |
                        .---------------------------------------------- 2gsaA @2.4    83.0625
                        |
                .---------------------------------------------------- 1ohvA @2.3

              +----+----+----+----+----+----+----+----+----+----+----+----+
          83.5050   81.5137   79.5225   77.5312   75.5400   73.5488   71.5575
               82.5094   80.5181   78.5269   76.5356   74.5444   72.5531
```

Align2d.ali

```
>P1;1ohvA
structureX:1ohv.pdb: 11 :A:+461 :A:MOL_ID 1; MOLECULE 4-AMINOBUTYRATE
AMINOTRANSFERASE; CHAIN A, B, C, D; FRAGMENT RESIDUES 29-500; SYNONYM
GAMMA-AMINO-N-BUTYRATE TRANSAMINASE, GABA TRANSAMI GABA
AMINOTRANSFERASE, GABA-AT, GABA-T; EC 2.6.1.19:MOL_ID 1;
ORGANISM_SCIENTIFIC SUS SCROFA; ORGANISM_COMMON PIG; ORGANISM_TAXID
9823; ORGAN LIVER: 2.30:-1.00
-------------------------------------------------
FDYDGPLMKTEVPGPRSRELMKQLNIIQNAEAVHFFCNYEESRGNYLVDVDGNRMLDLYSQISSIPIGYS

HPALVKLVQQPQNVSTFINRPALGILPPENFVEKLRESLLSVAPKGMSQLITMACGSCSNENAFKTIFMWYRSKERGQSA
FSKEELETCMINQAPGCPDYSILSFMGAFHGRTMGCLATTHSKAIHKIDIPSFDWPIAPFPRLKYPLEEFVKENQQEEARCL
```

EEVEDLIVKYRKKKKTVAGIIVEPIQSEGGDNHASDDFFRKLRDISRKHGCAFLVDEVQTGGGSTGKFWAHEHWGLDDPA
DVMTFSKKMMTGGFFHKEEFRPNAPYRIFNTWLGDPSKNLLLAEVINIIKREDLLSNAAHAGKVLLTGLLDLQARYPQFIS
RVRGRGTFCSFDTPDESIRNKLISIARNKGVMLGGCGDKSIRFRPTLVFRDHHAHLFLNIFSDILADF-*

```
>P1;gabat
sequence:gabat: : : : ::: 0.00: 0.00
MASMLLAQRLACSFQHSYRLLVPGSRHISQAAAKVDVEFDYDGPLMKTEVPGPRSQELMKQLNIIQNAEAV
HFFCNYEESRGNYLVDVDGNRMLDLYSQISSVPIGYSHPALLKLIQQPQNASMFVNRPALGILPPENFVEK
LRQSLLSVAPKGMSQLITMACGSCSNENALKTIFMWYRSKERGQRGFSQEELETCMINQAPGCPDYSILSF
MGAFHGRTMGCLATTHSKAIHKIDIPSFDWPIAPFPRLKYPLEEFVKENQQEEARCLEEVEDLIVKYRKKK
KTVAGIIVEPIQSEGGDNHASDDFFRKLRDIARKHGCAFLVDEVQTGGGCTGKFWAHEHWGLDDPADVMTF
SKKMMTGGFFHKEEFRPNAPYRIFNTWLGDPSKNLLLAEVINIIKREDLLNNAAHAGKALLTGLLDLQARY
PQFISRVRGRGTFCSFDTPDDSIRNKLILIARNKGVVLGGCGDKSIRFRPTLVFRDHHAHLFLNIFSDILA
DFK*
```

Model-single.py (model generated gabat-1ohvA with dope score)

```
<< end of ENERGY.
DOPE score               : -55550.527344
>> Model assessment by GA341 potential

Surface library          : C:\Program Files\Modeller9v7/modlib/surf5.de
Pair library             : C:\Program Files\Modeller9v7/modlib/pair9.de
Chain identifier         : _
% sequence identity      :    95.878998
Sequence length          :          500
Compactness              :     0.092349
Native energy (pair)     :  -563.688055
Native energy (surface)  :    -3.234556
Native energy (combined) :    -8.943275
Z score (pair)           :   -10.823216
Z score (surface)        :    -6.227564
Z score (combined)       :   -11.747523
GA341 score              :     1.000000

>> Summary of successfully produced models:
Filename                          molpdf     DOPE score    GA341 score
----------------------------------------------------------------------
gabat.B99990001.pdb           2768.29199   -55550.52734        1.00000
```

Evaluate_template.py

```
openf___224_> Open           1ohvA.profile
# Energy of each residue is written to: 1ohvA.profile
# The profile IS normalized by the number of restraints.
# The profiles are smoothed over a window of residues:   13
# The sum of all numbers in the file:      -17.5030

<< end of ENERGY.
DOPE score               : -56652.394531

Dynamically allocated memory at          finish [B,KiB,MiB]:     21326537   20826.695    20.339
Starting time                                                : 2011/11/19 23:00:14
Closing time                                                 : 2011/11/19 23:00:29
Total CPU time [seconds]                                     :      15.56
```

Evaluate_model.py

Using **gabat** as query sequence and **1ohvA** as a template **"gabat.B99990001.pdb(gabat-1ohvA)"** structure was modeled using modeler with **dope score as -55550.52734.**

Experiment 4 : Understanding the metabolic network: Kyto Encyclopedia of Genes and Genome (KEGG) database

Aim: To understand the network of metabolic pathways among the living cells

Background :

The KEGG database project was initiated in 1995 by Minoru Kanehisa, Professor at the Institute for Chemical Research, Kyoto University, under the then ongoing Japanese Human Genome Program. Foreseeing the need for a computerized resource that can be used for biological interpretation of genome sequence data, he started developing the KEGG PATHWAY database. It is a collection of manually drawn KEGG pathway maps representing experimental knowledge on metabolism and various other functions of the cell and the organism. Each pathway map contains a network of molecular interactions and reactions and is designed to link genes in the genome to gene products (mostly proteins) in the pathway. This has enabled the analysis called KEGG pathway mapping, whereby the gene content in the genome is compared with the KEGG PATHWAY database to examine which pathways and associated functions are likely to be encoded in the genome.

According to the developers, KEGG is a "computer representation" of the biological system. It integrates building blocks and wiring diagrams of the system — more specifically, genetic building blocks of genes and proteins, chemical building blocks of small molecules and reactions, and wiring diagrams of molecular interaction and reaction networks. This concept is realized in the following databases of KEGG, which are categorized into systems, genomic, chemical, and health information.

Systems information

The KEGG PATHWAY database, the wiring diagram database, is the core of the KEGG resource. It is a collection of pathway maps integrating many entities including genes, proteins, RNAs, chemical compounds, glycans, and chemical reactions, as well as disease genes and drug targets, which are stored as individual entries in the other databases of KEGG. The pathway maps are classified into the following sections:

- Metabolism
- Genetic information processing (transcription, translation, replication and repair, etc.)
- Environmental information processing (membrane transport, signal transduction, etc.)
- Cellular processes (cell growth, cell death, cell membrane functions, etc.)
- Organismal systems (immune system, endocrine system, nervous system, etc.)
- Human diseases
- Drug development

The metabolism section contains aesthetically drawn global maps showing an overall picture of metabolism, in addition to regular metabolic pathway maps. The low-resolution global maps can be used, for example, to compare metabolic capacities of different organisms in genomics studies and different environmental samples in metagenomics studies. In contrast, KEGG modules in the KEGG MODULE database are higher-resolution, localized wiring diagrams, representing tighter functional units within a pathway map, such as subpathways conserved among specific organism groups and molecular complexes. KEGG modules are defined as characteristic gene sets that can be linked to specific metabolic capacities and other phenotypic features, so that they can be used for automatic interpretation of genome and metagenome data.

Another database that supplements KEGG PATHWAY is the KEGG BRITE database. It is an ontology database containing hierarchical classifications of various entities including genes, proteins, organisms, diseases, drugs, and chemical compounds. While KEGG PATHWAY is limited to molecular interactions and reactions of these entities, KEGG BRITE incorporates many different types of relationships.

Genomic information

Several months after the KEGG project was initiated in 1995, the first report of the completely sequenced bacterial genome was published. Since then all published complete genomes are accumulated in KEGG for both eukaryotes and prokaryotes. The KEGG GENES database contains gene/protein-level information and the KEGG GENOME database contains organism-level information for these genomes. The KEGG GENES database consists of gene sets for the complete

genomes, and genes in each set are given annotations in the form of establishing correspondences to the wiring diagrams of KEGG pathway maps, KEGG modules, and BRITE hierarchies.

These correspondences are made using the concept of orthologs. The KEGG pathway maps are drawn based on experimental evidence in specific organisms but they are designed to be applicable to other organisms as well, because different organisms, such as human and mouse, often share identical pathways consisting of functionally identical genes, called orthologous genes or orthologs. All the genes in the KEGG GENES database are being grouped into such orthologs in the KEGG ORTHOLOGY (KO) database. Because the nodes (gene products) of KEGG pathway maps, as well as KEGG modules and BRITE hierarchies, are given KO identifiers, the correspondences are established once genes in the genome are annotated with KO identifiers by the genome annotation procedure in KEGG.

Chemical information

The KEGG metabolic pathway maps are drawn to represent the dual aspects of the metabolic network: the genomic network of how genome-encoded enzymes are connected to catalyze consecutive reactions and the chemical network of how chemical structures of substrates and products are transformed by these reactions. A set of enzyme genes in the genome will identify enzyme relation networks when superimposed on the KEGG pathway maps, which in turn characterize chemical structure transformation networks allowing interpretation of biosynthetic and biodegradation potentials of the organism. Alternatively, a set of metabolites identified in the metabolome will lead to the understanding of enzymatic pathways and enzyme genes involved.

The databases in the chemical information category, which are collectively called KEGG LIGAND, are organized by capturing knowledge of the chemical network. In the beginning of the KEGG project, KEGG LIGAND consisted of three databases: KEGG COMPOUND for chemical compounds, KEGG REACTION for chemical reactions, and KEGG ENZYME for reactions in the enzyme nomenclature. Currently, there are additional databases: KEGG GLYCAN for glycans and two auxiliary reaction databases called RPAIR (reactant pair alignments) and RCLASS (reaction class). KEGG COMPOUND has also been expanded to contain various compounds such as xenobiotics, in addition to metabolites.

Health information

In KEGG, diseases are viewed as perturbed states of the biological system caused by perturbates of genetic factors and environmental factors, and drugs are viewed as different types of perturbates. The KEGG PATHWAY database includes not only the normal states but also the perturbed states of the biological systems. However, disease pathway maps cannot be drawn for most diseases because molecular mechanisms are not well understood. An alternative approach is taken in the KEGG DISEASE database, which simply catalogues known genetic factors and environmental factors of diseases. These catalogues may eventually lead to more complete wiring diagrams of diseases.

The KEGG DRUG database contains active ingredients of approved drugs in Japan, the US, and Europe. They are distinguished by chemical structures and/or chemical components and associated with target molecules, metabolizing enzymes, and other molecular interaction network information in the KEGG pathway maps and the BRITE hierarchies. This enables an integrated analysis of drug interactions with genomic information. Crude drugs and other health-related substances, which are outside the category of approved drugs, are stored in the KEGG ENVIRON database. The databases in the health information category are collectively called KEGG MEDICUS, which also includes package inserts of all marketed drugs in Japan.

Procedure:

1. Open the KEGG website
2. https://www.genome.jp/kegg/?sess=ebfe2ad23e021e38540f798c803dd061

2. Select a particular protein (enzyme) name in the text box

3. On pressing search button the result page is displayed

4. Study the classification of the KEGG

5. Explore PATHWAY — pathway maps for cellular and organismal functions

6. Explore MODULE — modules or functional units of genes

7. Explore BRITE — hierarchical classifications of biological entities

8. Explore GENOME — complete genomes

9. Explore GENES — genes and proteins in the complete genomes

10. Explore ORTHOLOGY — ortholog groups of genes in the complete genomes

11. Explore COMPOUND, GLYCAN — chemical compounds and glycans

12. Explore REACTION, RPAIR, RCLASS — chemical reactions

13. Explore ENZYME — enzyme nomenclature

14. Explore DISEASE — human diseases

15. Explore DRUG — approved drugs

16. Explore ENVIRON — crude drugs and health-related substances

Interpretation:

Experiment V & VI : PROTEIN DATA BANK (PDB)

Aim: To retrieve the structure of a protein and viewing it in RASMOL viewer.

Description:

The Protein Data Bank (PDB) is a repository for the 3-D structural data of large biological molecules, such as proteins and nucleic acids. The data, typically obtained by X-ray crystallography or NMR spectroscopy and submitted by biologists and biochemists from around the world, can be accessed at no charge on the internet. The PDB is overseen by an organization called the Worldwide Protein Data Bank. The PDB is a key resource in areas of structural biology, such as structural genomics. Most major scientific journals, and some funding agencies, such as the NIH in the USA, now require scientists to submit their structure data to the PDB. If the contents of the PDB are thought of as primary data, then there are hundreds of derived (i.e., secondary) databases that categorize the data differently. For example, both SCOP and CATH categorize structures according to type of structure and assumed evolutionary relations; GO categorize structures based on genes.

Procedure:

1. Open the PDB website
2. Type the protein name in the text box titled enter keyword or type the PDB ID
3. On pressing search button the result page is displayed
4. Choose the appropriate structure by double clicking the PDB ID
5. A web page is displayed with details about the structure
6. Download the structure file from the right hand corner of the webpage
7. Save the file as PDB file.
8. Open the RASMOL viewer to view the downloaded structure.

9. Interpret the results.

Interpretation:

```
HEADER    CELL CYCLE                              21-DEC-04   1YC1
TITLE     CRYSTAL STRUCTURES OF HUMAN HSP90ALPHA COMPLEXED WITH
TITLE    2 DIHYDROXYPHENYLPYRAZOLES
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: HEAT SHOCK PROTEIN HSP 90-ALPHA;
COMPND   3 CHAIN: A;
COMPND   4 FRAGMENT: N-TERMINAL ATP BINDING DOMAIN RESIDUES 9-223;
COMPND   5 SYNONYM: HSP 86;
COMPND   6 ENGINEERED: YES;
COMPND   7 MUTATION: YES
SOURCE    MOL_ID: 1;
SOURCE   2  ORGANISM_SCIENTIFIC: HOMO SAPIENS;
SOURCE   3 ORGANISM_COMMON: HUMAN;
AUTHOR    A.KREUSCH,S.HAN,A.BRINKER,V.ZHOU,H.CHOI,Y.HE,S.A.LESLEY,
AUTHOR   2 J.CALDWELL,X.GU
REVDAT   1   22-FEB-05 1YC1    0
JRNL        AUTH   A.KREUSCH,S.HAN,A.BRINKER,V.ZHOU,H.CHOI,Y.HE,
JRNL        AUTH 2 S.A.LESLEY,J.CALDWELL,X.GU
```

```
REMARK   3   RESOLUTION RANGE HIGH (ANGSTROMS) : 1.70
REMARK   3   RESOLUTION RANGE LOW  (ANGSTROMS) : 47.18
REMARK   3   DATA CUTOFF            (SIGMA(F)) : 0.000
REMARK   3   DATA CUTOFF HIGH         (ABS(F)) : 1654342.920
REMARK   3   R VALUE            (WORKING SET) : 0.190
DBREF  1YC1 A    9   236  SWS    P07900   HS9A_HUMAN       8    235
SEQADV 1YC1 MET A  -27  SWS  P07900              CLONING ARTIFACT
SEQADV 1YC1 ARG A  -26  SWS  P07900              CLONING ARTIFACT
SEQADV 1YC1 GLY A  -25  SWS  P07900              CLONING ARTIFACT
SEQADV 1YC1 SER A  -24  SWS  P07900              CLONING ARTIFACT
SEQADV 1YC1 HIS A  -23  SWS  P07900              CLONING ARTIFACT
SEQADV 1YC1 HIS A  -22  SWS  P07900              CLONING ARTIFACT
SEQADV 1YC1 HIS A  -21  SWS  P07900              CLONING ARTIFACT
SEQADV 1YC1 HIS A  -20  SWS  P07900              CLONING ARTIFACT
SEQADV 1YC1 HIS A  -19  SWS  P07900              CLONING ARTIFACT
SEQRES   1 A  264  MET ARG GLY SER HIS HIS HIS HIS HIS HIS GLY MET ALA
SEQRES   2 A  264  SER MET THR GLY GLY GLN GLN MET GLY ARG ASP LEU TYR
SEQRES   3 A  264  ASP ASP ASP ASP LYS ASP ARG TRP GLY SER ASP GLN PRO
SEQRES   4 A  264  MET GLU GLU GLU GLU VAL GLU THR PHE ALA PHE GLN ALA
FORMUL   3  HOH    *307(H2 O1)
HELIX    1   1 GLN A   23  THR A   36  1                                  14
HELIX    2   2 GLU A   42  ASP A   66  1                                  25
HELIX    3   3 PRO A   67  GLY A   73  5                                   7
HELIX    4   4 THR A   99  THR A  109  1                                  11
HELIX    5   5 GLY A  114  ALA A  124  1                                  11
HELIX    6   6 ASP A  127  GLY A  135  5                                   9
```

```
HELIX    7   7 VAL A  136  TYR A  139  5                                    4
HELIX    8   8 SER A  140  VAL A  144  1                                    5
HELIX    9   9 GLU A  192  LEU A  198  5                                    7
HELIX   10  10 GLU A  199  SER A  211  1                                   13
SHEET    1   A 8 VAL A  17  ALA A  21  0
SHEET    2   A 8 SER A 169  THR A 174 -1  O  PHE A 170   N  PHE A  20
SHEET    3   A 8 TYR A 160  SER A 164 -1  N  ALA A 161   O  ARG A 173
SHEET    4   A 8 ALA A 145  LYS A 153 -1  N  VAL A 150   O  TRP A 162
SHEET    5   A 8 GLY A 183  LEU A 190 -1  O  ILE A 187   N  THR A 149
SHEET    6   A 8 THR A  88  ASP A  93 -1  N  ILE A  91   O  VAL A 186
SHEET    7   A 8 ILE A  78  ASN A  83 -1  N  ILE A  81   O  THR A  90
SHEET    8   A 8 ILE A 218  LEU A 220  1  O  THR A 219   N  LEU A  80
CRYST1   71.611   89.000   88.386  90.00  90.00  90.00 C 2 2 21      8
ATOM      1  N   GLU A  15       7.459   9.464  -9.942  1.00 49.96           N
ATOM      2  CA  GLU A  15       7.254  10.912 -10.271  1.00 49.72           C
ATOM      3  C   GLU A  15       8.598  11.661 -10.244  1.00 47.42           C
ATOM      4  O   GLU A  15       9.539  11.265  -9.539  1.00 47.68           O
ATOM      5  CB  GLU A  15       6.287  11.565  -9.257  1.00 52.36           C
ATOM      6  CG  GLU A  15       5.135  10.641  -8.729  1.00 56.31           C
ATOM      7  CD  GLU A  15       3.816  10.753  -9.522  1.00 58.31           C
ATOM      8  OE1 GLU A  15       3.231  11.871  -9.553  1.00 58.81           O
HETATM 1698  O   HOH    16      14.685  44.698   2.576  1.00 17.46           O
HETATM 1699  O   HOH    17       6.337  38.090  -5.540  1.00 21.43           O
HETATM 1700  O   HOH    18      15.629  27.133   7.448  1.00 22.34           O
HETATM 1701  O   HOH    19     -19.432  26.889   7.633  1.00 16.80           O
HETATM 1702  O   HOH    20       9.843  35.417  -0.097  1.00 19.80           O
```

```
HETATM 1703  O   HOH    21      -12.567  20.212  12.271  1.00 20.66           O
HETATM 1704  O   HOH    22      -15.662  30.713  -0.155  1.00 19.67           O
HETATM 1705  O   HOH    23      -14.464  18.892   5.131  1.00 27.82           O
HETATM 1706  O   HOH    24        0.786  38.004   3.837  1.00 22.20           O
HETATM 1707  O   HOH    25       -4.108  32.222 -13.997  1.00 23.42           O
HETATM 1708  O   HOH    26        0.675  26.905 -12.899  1.00 18.53           O
CONECT 1657 1656 1658 1661
CONECT 1658 1657 1659
CONECT 1659 1658 1660
CONECT 1660 1659 1661
CONECT 1661 1657 1660 1662
CONECT 1662 1661 1663
CONECT 1663 1662 1664
CONECT 1664 1656 1663 1665
END
```

Result analysis :

Exp 7-A: Basic mathematical operations

Aim: To perform Basic mathematical operations using PERL

Program: (Example)

```
#!/usr/bin/perl -w
$x=10;
print"\tThe value of first variable,x is : $x\n";
$y=5;
print"\tThe value of second variable,y is : $y\n";
$sum=$x+$y;
print"\tThe sum of Two variables is : $sum\n";
$diff=$x-$y;
print"\tThe difference of Two variables is : $diff\n";
exit;
```

Output :

```
The value of first variable,x is : 10
The value of second variable,y is : 5
The sum of Two variables is : 15
The difference of Two variables is : 5
```

Exercise:

1. Print different types of numbers on the Screen.
2. Print Binary & Hexadecimal numbers using perl script.
3. Write perl script to swap values.

Exp 7-B : Basic mathematical operations

Aim: To perform Basic mathematical operations using PERL

Program :

```
#!/usr/bin/perl -w
$x=10;
print"\tThe value of first variable,x is : $x\n";
$y=5;
print"\tThe value of second variable,y is : $y\n";
$sum=$x+$y;
print"\tThe sum of Two variables is : $sum\n";
$diff=$x-$y;
print"\tThe difference of Two variables is : $diff\n";
exit;
```

Output :

```
The value of first variable,x is : 10
The value of second variable,y is : 5
The sum of Two variables is : 15
The difference of Two variables is : 5
```

Exp 7-C: Concatenating DNA

Aim: To Concatenating DNA sequences using PERL

Program:

```
#!/usr/bin/perl -w
# Concatenating DNA
# Store two DNA fragments into two variables called $DNA1 and $DNA2
$DNA1 = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';
$DNA2 = 'ATAGTGCCGTGAGAGTGATGTAGTA';
# Print the DNA onto the screen
print "Here are the original two DNA fragments:\n\n";
print $DNA1, "\n";
print $DNA2, "\n\n";
# Concatenate the DNA fragments into a third variable and print them
# Using "string interpolation"
$DNA3 = "$DNA1$DNA2";
print "Here is the concatenation of the first two fragments (version 1):\n\n";
print "$DNA3\n\n";
# An alternative way using the "dot operator":
# Concatenate the DNA fragments into a third variable and print them
$DNA3 = $DNA1 . $DNA2;
print "Here is the concatenation of the first two fragments (version 2):\n\n";
print "$DNA3\n\n";
print "Here is the concatenation of the first two fragments (version 3):\n\n";
print $DNA1, $DNA2, "\n";
exit;
```

Output :

Here are the original two DNA fragments:

ACGGGAGGACGGGAAAATTACTACGGCATTAGCATAGTGCCGTGAGAGTGATGTAGT
A

Here is the concatenation of the first two fragments

(version 1):

ACGGGAGGACGGGAAAATTACTACGGCATTAGCATAGTGCCGTGAGAGTGATGTAGT
A

Here is the concatenation of the first two fragments

(version 2):

ACGGGAGGACGGGAAAATTACTACGGCATTAGCATAGTGCCGTGAGAGTGATGTAGT
A

Here is the concatenation of the first two fragments

(version 3):

ACGGGAGGACGGGAAAATTACTACGGCATTAGCATAGTGCCGTGAGAGTGATGTAGT
A

Exp 7-D: Transcribing DNA into RNA

Aim: To Transcribe DNA sequence into RNA sequence using PERL

Program :

```
#!/usr/bin/perl -w

# Transcribing DNA into RNA

# The DNA

$DNA = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';

# Print the DNA onto the screen

print "Here is the starting DNA:\n\n";
```

```
print "$DNA\n\n";

# Transcribe the DNA to RNA by substituting all T's with U's.

$RNA = $DNA;

$RNA =~ s/T/U/g;

# Print the RNA onto the screen

print "Here is the result of transcribing the DNA to

RNA:\n\n";

print "$RNA\n";

# Exit the program.

exit;
```

OutPut :

```
Here is the starting DNA:

ACGGGAGGACGGGAAAATTACTACGGCATTAGC

Here is the result of transcribing the DNA to RNA:

ACGGGAGGACGGGAAAAUUACUACGGCAUUAGC
```

Exp 8-A: Calculating the reverse complement of a strand of DNA

Aim: To calculate the reverse complement of a strand of DNA using PERL

Program :

```
#!/usr/bin/perl -w
# Calculating the reverse complement of a strand of DNA
$DNA = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';
# Print the DNA onto the screen
print "Here is the starting DNA:\n\n";
print "$DNA\n\n";
# Calculate the reverse complement
# Warning: this attempt will fail!
# First, copy the DNA into new variable $revcom
# (short for REVerse COMplement)
# Notice that variable names can use lowercase letters like
# "revcom" as well as uppercase like "DNA". In fact,
# lowercase is more common.
#
# It doesn't matter if we first reverse the string and then
# do the complementation; or if we first do the
complementation
# and then reverse the string. Same result each time.
# So when we make the copy we'll do the reverse in the same
statement.
#
$revcom = reverse $DNA;
#
```

```
# Next substitute all bases by their complements,
# A->T, T->A, G->C, C->G
#
$revcom =~ s/A/T/g;
$revcom =~ s/T/A/g;
$revcom =~ s/G/C/g;
$revcom =~ s/C/G/g;
# Print the reverse complement DNA onto the screen
print "Here is the reverse complement DNA:\n\n";
print "$revcom\n";
#
# Oh-oh, that didn't work right!
# Our reverse complement should have all the bases in it,
since the
# original DNA had all the bases--but ours only has A and G!
#
# Do you see why?
#
# The problem is that the first two substitute commands
above change
# all the A's to T's (so there are no A's) and then all the
# T's to A's (so all the original A's and T's are all now
A's).
# Same thing happens to the G's and C's all turning into G's.
#
print "\nThat was a bad algorithm, and the reverse complement was wrong!\n";
print "Try again ... \n\n";
```

```
# Make a new copy of the DNA (see why we saved the original?)
$revcom = reverse $DNA;
# See the text for a discussion of tr///
$revcom =~ tr/ACGTacgt/TGCAtgca/;
# Print the reverse complement DNA onto the screen
print "Here is the reverse complement DNA:\n\n";
print "$revcom\n";
print "\nThis time it worked!\n\n";
exit;
```

Output :

```
Here is the starting DNA:
ACGGGAGGACGGGAAAATTACTACGGCATTAGC
Here is the reverse complement DNA:
GGAAAAGGGGAAGAAAAAAAGGGGAGGAGGGGA
That was a bad algorithm, and the reverse complement was wrong!
Try again ...
Here is the reverse complement DNA:
GCTAATGCCGTAGTAATTTTCCCGTCCTCCCGT
This time it worked!
```

Exp 8-B: Reading protein sequence data from a file

Aim: To Read a protein sequence data from a file using PERL

Program :

```
#!/usr/bin/perl -w
```

```
# Reading protein sequence data from a file, take 2
# The filename of the file containing the protein sequence
data
$proteinfilename = 'NM_021964fragment.pep';
# First we have to "open" the file, and associate
# a "filehandle" with it. We choose the filehandle
# PROTEINFILE for readability.
open(PROTEINFILE, $proteinfilename);
# Now we do the actual reading of the protein sequence data
from the file,
# by using the angle brackets < and > to get the input from
the
# filehandle. We store the data into our variable $protein.
#
# Since the file has three lines, and since the read only
Is
# returning one line, we'll read a line and print it, three
times.
# First line
$protein = <PROTEINFILE>;
# Print the protein onto the screen
print "\nHere is the first line of the protein file:\n\n";
print $protein;
# Second line
```

```
$protein = <PROTEINFILE>;
# Print the protein onto the screen
print "\nHere is the second line of the protein file:\n\n";
print $protein;
# Third line
$protein = <PROTEINFILE>;
# Print the protein onto the screen
print "\nHere is the third line of the protein file:\n\n";
print $protein;
# Now that we've got our data, we can close the file.
close PROTEINFILE;
exit;
```

Output:

```
Here is the first line of the protein file:
MNIDDKLEGLFLKCGGIDEMQSSRTMVVMGGVSGQSTVSGELQD
Here is the second line of the protein file:
SVLQDRSMPHQEILAADEVLQESEMRQQDMISHDELMVHEETVKNDEEQM
ETHERLPQ
Here is the third line of the protein file:
GLQYALNVPISVKQEITFTDVSEQLMRDKKQIR
```

Exp 9-A: Conditional Statement

Aim: To use the conditional statements in PERL

Program

```
#!/usr/bin/perl -w
# if-elsif-else
$word = 'MNIDDKL';
# if-elsif-else conditionals
if($word eq 'QSTVSGE') {
print "QSTVSGE\n";
} elsif($word eq 'MRQQDMISHDEL') {
print "MRQQDMISHDEL\n";
} elsif ( $word eq 'MNIDDKL' ) {
print "MNIDDKL--the magic word!\n";
} else {
print "Is \"$word\" a peptide? This program is not
sure.\n";
}
exit;
```

Output :

MNIDDKL--the magic word!

Exp 9-B: REGULAR EXPRESSIONS

Program 1

```
#!/usr/bin/perl
print “Enter your DNA Sequence : ”;
$DNA = <>;
chomp $DNA;
print “EcoRI site found!” if $DNA =~/GAATTC/;
print$DNA;
```

Program 2

```
#!/usr/bin/perl
$string = "do the words heaven and eleven match?";
if ( $string =~ /even/ )
{
print "A match was found.\n";
}
else
{
print "No match was found.\n";
}
```

Program 3 : Array operations

Aim: To perform various array operations using PERL

1: Pop operation using arrays

```
#!/usr/bin/perl –w
@bases = ('A', 'C', 'G', 'T');
$base1 = pop @bases;
print "Here's the element removed from the end: ";
print $base1, "\n\n";
print "Here's the remaining array of bases: ";
print "@bases";
```

OutPut :

Here's the element removed from the end: T

Here's the remaining array of bases: A C G

2 :Shift operation on arrays

```
#!/usr/bin/perl –w
@bases = ('A', 'C', 'G', 'T');
$base2 = shift @bases;
print "Here's an element removed from the beginning: ";
print $base2, "\n\n";
print "Here's the remaining array of bases: ";
print "@bases";
```

Output :

Here's an element removed from the beginning: A

Here's the remaining array of bases: C G T

3: Unshift operations on arrays

```
#!/usr/bin/perl –w
@bases = ('A', 'C', 'G', 'T');
$base1 = pop @bases;
unshift (@bases, $base1);
print "Here's the element from the end put on the beginning:";
print "@bases\n\n";
```

Output:

Here's the element from the end put on the beginning: T A C G

4: Push operation on arrays

```
#!/usr/bin/perl –w
@bases = ('A', 'C', 'G', 'T');
$base2 = shift @bases;
push (@bases, $base2);
print "Here's the element from the beginning put on the end:";
print "@bases\n\n";
```

Output:

Here's the element from the beginning put on the end: C G T A

5: Reverse of an array

```
#!/usr/bin/perl –w
@bases = ('A', 'C', 'G', 'T');
@reverse = reverse @bases;
print "Here's the array in reverse: ";
print "@reverse\n\n"
```

Output:

Here's the array in reverse: T G C A

6: Length of an array

```
#!/usr/bin/perl –w
@bases = ('A', 'C', 'G', 'T');
print "Here's the length of the array: ";
print scalar @bases, "\n";
```

Output:

Here's the length of the array: 4

7: Splicing of an array

```
#!/usr/bin/perl –w
@bases = ('A', 'C', 'G', 'T');
splice ( @bases, 2, 0, 'X');
print "Here's the array with an element inserted after the
2nd element: ";
print "@bases\n";
```

Output:

Here's the array with an element inserted after the 2nd element: A C X G T

Exp 10-A: Searching for motifs

Aim: To search for a motif in a DNA sequence using PERL

Program :

```
#!/usr/bin/perl -w
# Searching for motifs
# Ask the user for the filename of the file containing
# the protein sequence data, and collect it from the
keyboard
print "Please type the filename of the protein sequence
data: ";
$proteinfilename = <STDIN>;
# Remove the newline from the protein filename
chomp $proteinfilename;
# open the file, or exit
unless ( open(PROTEINFILE, $proteinfilename) ) {
print "Cannot open file \"$proteinfilename\"\n\n";
exit;
}
# Read the protein sequence data from the file, and store
it
# into the array variable @protein
@protein = <PROTEINFILE>;
# Close the file - we've read all the data into @protein
now.
close PROTEINFILE;
# Put the protein sequence data into a single string, as
it's easier
```

```
# to search for a motif in a string than in an array of
# lines (what if the motif occurs over a line break?)
$protein = join( '', @protein);
# Remove whitespace
$protein =~ s/\s//g;
# In a loop, ask the user for a motif, search for the motif,
# and report if it was found.
# Exit if no motif is entered.
do {
print "Enter a motif to search for: ";
$motif = <STDIN>;
# Remove the newline at the end of $motif
chomp $motif;
# Look for the motif
if ( $protein =~ /$motif/ ) {
print "I found it!\n\n";
} else {
print "I couldn\'t find it.\n\n";
}
# exit on an empty user input
} until ( $motif =~ /^\s*$/ );
# exit the program
exit;
```

Output :

Please type the filename of the protein sequence data:

NM_021964fragment.pep

Enter a motif to search for: SVLQ

I found it!

Enter a motif to search for: jkl

I couldn't find it.

Enter a motif to search for: QDSV

I found it!

Enter a motif to search for: HERLPQGLQ

I found it!

Enter a motif to search for:

I couldn't find it.

Exp 10-B: A subroutine to append ACGT to DNA

Aim: To append ACGT to DNA using subroutine

Program :

```
#!/usr/bin/perl -w
# A program with a subroutine to append ACGT to DNA
# The original DNA
$dna = 'CGACGTCTTCTCAGGCGA';
# The call to the subroutine "addACGT".
# The argument being passed in is $dna; the result is saved
```

```
in $longer_dna
$longer_dna = addACGT($dna);
print "I added ACGT to $dna and got $longer_dna\n\n";
exit;
# Here is the definition for subroutine "addACGT"
sub addACGT {
my($dna) = @_;
$dna .= 'ACGT';
return $dna;
}
```

Output:

```
I added ACGT to CGACGTCTTCTCAGGCGA and got
CGACGTCTTCTCAGGCGAACGT
```