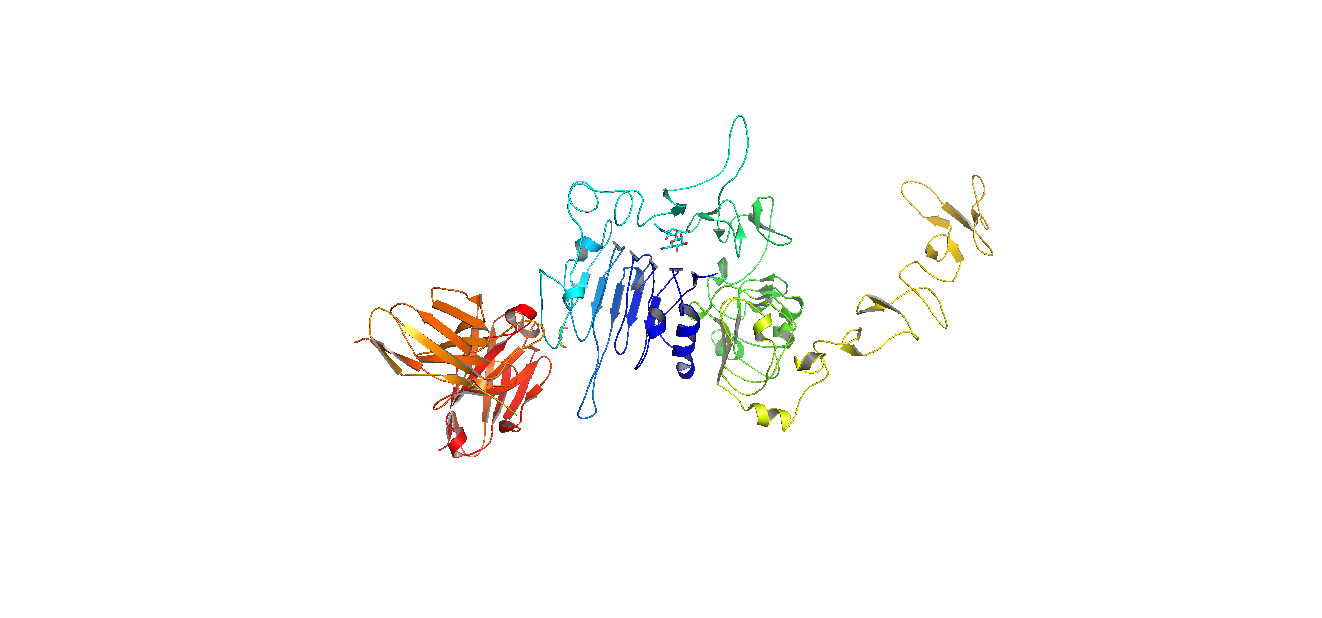
**Step #1:** Select a human disease and find the responsible protein.

Disease: Breast Cancer

Responsible Protein: HER2 (6J71)

**Step #2:** Present the protein’s structure associated with the disease. Give its three dimensional structure using PyMOL. You need to make use of Protein Data Bank (/www.rcsb.org/).

Figure 1: Protein structure of HER2 with pdb id: 6j71

**Step #3:** Find the gene coding this protein, give the accession number(s), length of the sequence(s), type of molecule (DNA, RNA) and the organism from which this sequence is derived from.

Corresponding gene: ERBB2

NCBI gene id: 2064

Accession number: NC\_000017.11:39688094-39728658

Length of the sequence: 40565

type of molecule: DNA

Organism: *Homo sapiens*

**Step #7:** Select your nucleotide sequences mentioned in the steps above and extract it using the command “getgenbank”. Then, display it using the command “seqdisp”

>> sequence = getgenbank('NC\_000017','PARTIALSEQ',[39688094,39728658], 'FileFormat', 'FASTA');

>> seqdisp(sequence)

ans =

678×72 char array

'>NC\_000017.11:39688094-39728658 Homo sapiens chromosome 17, GRCh38.p1...'

' 1 GTTCTTTATT CTACTCTCCG CTGAAGTCCA CACAGTTTAA ATTAAAGTTC CCGGATTTTT'

' 61 GTGGGCGCCT GCCCCGCCCC TCGTCCCCCT GCTGTGTCCA TATATCGAGG CGATAGGGTT'

' 121 AAGGGAAGGC GGACGCCTGA TGGGTTAATG AGCAAACTGA AGTGTTTTCC ATGATCTTTT'

' 181 TTGAGGTAGG GCTGTTTACT GTCACCACCC CTGTCGGATT TTACTTCCTA AACGTACCTG'

' 241 TAACTATCCA CTTCTCTCCA TCTCTTCTGG CACCACCCTG GTTAAAGACA CCATCATGTG'

' 301 TCGCCAAGAC AGCCGCAGTA GCTTCTTAAT GGCTCTCCCT GCCTCTACTT TTGCCTCTTC'

' 361 CAACCTGCGC TCCATTTTGA AAAATTAAAA TTTGCCCATA TCACTTTTTT TTTCTTAAAA'

' 421 TTATTTACTG GCTCCCAATT ACCTTGGGTA AAATACAGTC TCCACAAACC CTGCCTGATT'

' 481 TGGCCCCTGT CCACTGGTCT CCCTCACTCC CTTGCTCCAG ACCCGCTTCA GAGGGCTATG'

... (truncated output)

**Step #8:** Give the length of your sequence using “length” command.

>> length(sequence.Sequence)

ans =

40565

**Step #9:** Select five different features of your gene entry and display their values making use of the construction of a structure.

% Define the structure

geneEntry = struct();

% Assign values to each field

geneEntry.OfficialSymbol = 'ERBB2';

geneEntry.OfficialFullName = 'erb-b2 receptor tyrosine kinase 2';

geneEntry.GeneID = 2064;

geneEntry.GeneType = 'protein coding';

geneEntry.Organism = 'Homo sapiens';

% Display the geneEntry structure

disp('Gene Entry Details:');

disp(geneEntry);

**Output:**

Gene Entry Details:

OfficialSymbol: 'ERBB2'

OfficialFullName: 'erb-b2 receptor tyrosine kinase 2'

GeneID: 2064

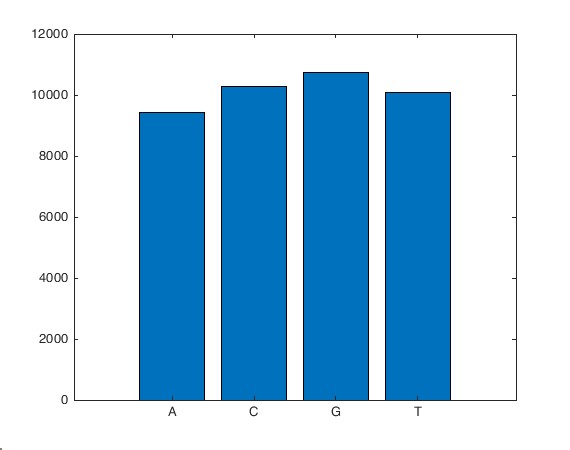
GeneType: 'protein coding'

Organism: 'Homo sapiens'

**Step #10:** Compute the sequence composition of nucleotide, dinucleotide and trinucleotides. Compare these frequency values to what is expected by pure chance. How close are sequence statistics to what you expect by pure chance?

Nucleotide Frequency:

>> basecount(sequence, 'Chart', 'bar')

Figure 2: Single nucleotide frequency plot

ans =

struct with fields:

A: 9427

C: 10287

G: 10753

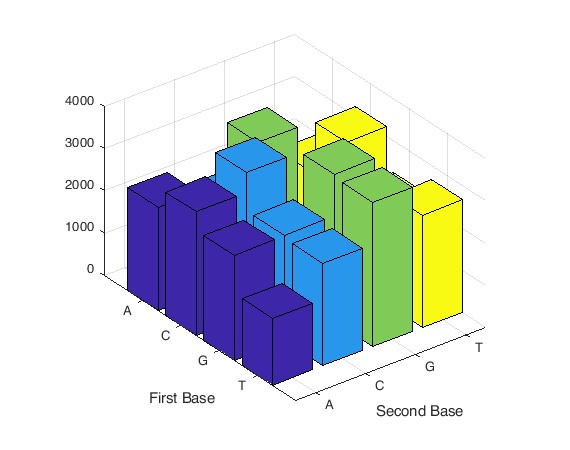
T: 10098

Di-nucleotide Frequency:

>> dimercount(sequence, 'Chart', 'bar')

ans =

struct with fields:

Figure 3: Di-nucleotide frequency plot

AA: 2442

AC: 1943

AG: 3075

AT: 1966

CA: 2924

CC: 3399

CG: 746

CT: 3218

GA: 2488

GC: 2512

GG: 3507

GT: 2246

TA: 1573

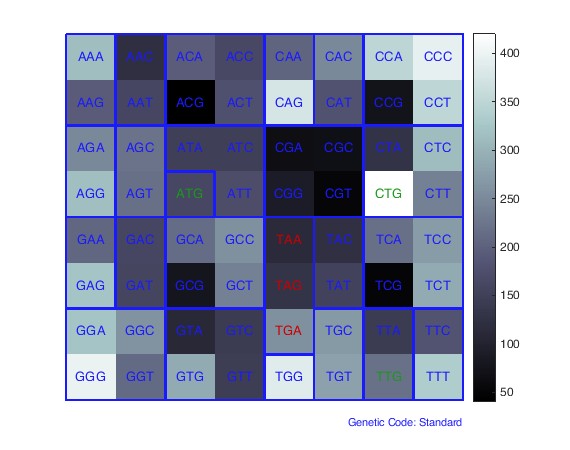
TC: 2433

TG: 3424

TT: 2668

Tri-nucleotide Frequency:

>> codoncount(sequence, 'Figure', 'true')

Figure 4: Tri-nucleotide frequency plot

AAA - 314 AAC - 122 AAG - 193 AAT - 162

ACA - 195 ACC - 165 ACG - 41 ACT - 177

AGA - 250 AGC - 223 AGG - 316 AGT - 218

ATA - 148 ATC - 150 ATG - 164 ATT - 173

CAA - 204 CAC - 249 CAG - 375 CAT - 180

CCA - 350 CCC - 395 CCG - 74 CCT - 352

CGA - 66 CGC - 69 CGG - 91 CGT - 52

CTA - 130 CTC - 312 CTG - 421 CTT - 239

GAA - 207 GAC - 162 GAG - 323 GAT - 159

GCA - 214 GCC - 258 GCG - 77 GCT - 239

GGA - 323 GGC - 262 GGG - 400 GGT - 211

GTA - 110 GTC - 143 GTG - 290 GTT - 146

TAA - 113 TAC - 124 TAG - 131 TAT - 155

TCA - 218 TCC - 273 TCG - 50 TCT - 290

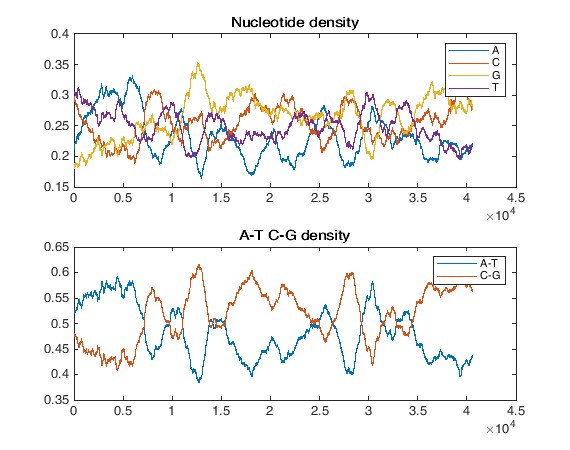
TGA - 258 TGC - 271 TGG - 387 TGT - 279

TTA - 142 TTC - 181 TTG - 220 TTT - 335

Composition of nucleotide, dinucleotide and trinucleotides by pure chance are respectively 25%, 6.25% and 1.56% i.e., for the sequence with length 40565 we should get count approximately 10141.25, 2535.31 and 633.83.

If we compare these expected values with the above counts, we can observe the deviation. Most of the entity deviates the expected values.

**Step #11:** Give the density plot for each nucleotide (Hint: ntdensity), and also for A-T and C-G density

Figure 5: Nucleotide density

For first graph:

x-axis represents the dna sequence position and y-axis represents density of each nucleotide.

**Blue line (A)**: This line represents the density of adenine (A) nucleotides along the DNA sequence.

**Orange line (C)**: This line represents the density of cytosine (C) nucleotides along the DNA sequence.

**Yellow line (G)**: This line represents the density of guanine (G) nucleotides along the DNA sequence.

**Purple line (T)**: This line represents the density of thymine (T) nucleotides along the DNA sequence.

If we consider position 0 in the graph:

density of A = 0.225

density of C = 0.3

density of G = 0.185

density of T = 0.3

For second graph:

x-axis represents the dna sequence position and combined density of adenine-thymine (A-T) pairs and cytosine-guanine (C-G) pairs.

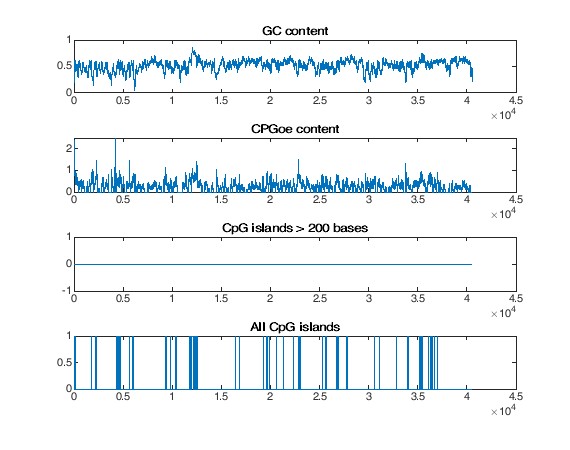
**Blue line (A-T)**: This line represents the combined density of adenine (A) and thymine (T) nucleotides.

**Orange line (C-G)**: This line represents the combined density of cytosine (C) and guanine (G) nucleotides.

If we consider position 0 in the graph:

density of A-T = 0.52

density of C-G = 0.48

Step #12: Give the CpG islands in your sequence. Show the plots as well as the start and end position of each existing island.

The graph analyzes the CpG content and distribution in a DNA sequence. CpG islands are regions with a high frequency of CpG sites, which are often associated with gene regulatory regions.

First Graph: GC Content

This graph displays the GC content across the DNA sequence's length.

The line fluctuating around 0.5 indicates varying GC-rich and GC-poor regions, potentially correlating with gene density or regulatory elements.

#### Second Graph: CpG Observed/Expected (CpGoe) Content

This graph illustrates the ratio of observed to expected CpG dinucleotides.

**In the graph, v**alues near 1 signify a match between observed and expected CpG dinucleotides, while values above 1 suggest an elevated frequency, hinting at potential CpG islands.

#### Third Graph: CpG Islands > 200 Bases

This graph identifies regions longer than 200 bases qualifying as CpG islands.

A lack of marked regions (indicated by "0") implies no CpG islands exceeding 200 bases in this segment.

#### Fourth Graph: All CpG Islands

This graph shows all identified CpG islands regardless of length.

Alternating 0s and 1s indicate numerous CpG islands scattered throughout, suggesting a high density of CpG-rich regions crucial for gene regulation.

**Step #13:** Create a cell array with the segments of sequences (exons) joined to form the coding region, CDS.

**Step #13p:** Use nt2aa to convert the sequence into amino acid in three forward and reverse frames.

Figure 6: Translation in six frames with truncated protein output

Step #14: Find the stop codons in all six open reading frames.

>> count(forward\_frame, '\*')

forward\_count =

502

456

541

>> count(reverse\_frame, '\*')

reverse\_count =

511

568

539

Step #15: Produce a plot of the number of stop codons in each of the six frames

% Create a matrix with the counts for both forward and reverse frames

data = [forward\_count; reverse\_count]';

% Create a bar plot

figure;

bar(data);

% Customize the plot

xticks(1:6);

xticklabels({'1', '2', '3', '-1', '-2', '-3'});

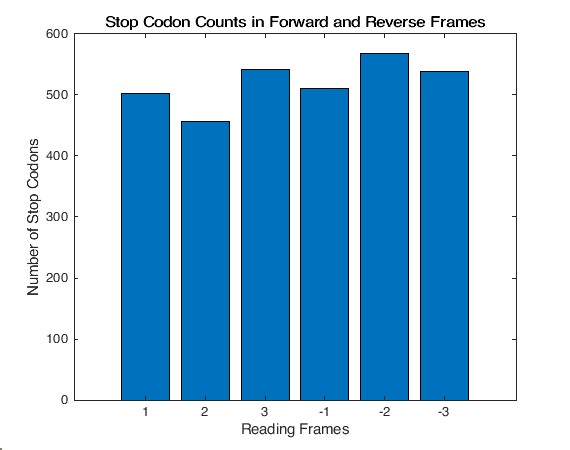
xlabel('Reading Frames');

ylabel('Number of Stop Codons');

title('Stop Codon Counts in Forward and Reverse Frames');

% Display the plot

shg;

Figure 7: Stop codon counts in each six frames

From the graph:

Number of stop codons are in descending order as:

Reverse frame 2 > forward frame 3 > reverse frame 3 > reverse frame 1 > forward frame 1 > forward frame 2

**Step #16:** Find the length of the longest uninterrupted span in all frames

% Function to find the longest uninterrupted span without stop codons

function [longest\_sequence, max\_length] = longest\_uninterrupted\_span(aa\_sequence)

spans = strsplit(aa\_sequence, '\*');

lengths = cellfun(@length, spans);

[max\_length, idx] = max(lengths);

longest\_sequence = spans{idx};

end

% Initialize arrays to store the results

max\_lengths = zeros(6, 1);

longest\_spans = cell(6, 1);

% Iterate over the forward\_frame

for i = 1:length(forward\_frame)

[longest\_spans{i}, max\_lengths(i)] = longest\_uninterrupted\_span(forward\_frame{i});

end

% Iterate over the reverse\_frame

for i = 1:length(reverse\_frame)

[longest\_spans{i+3}, max\_lengths(i+3)] = longest\_uninterrupted\_span(reverse\_frame{i});

end

% Display the results

disp('Longest uninterrupted span in each frame:');

for i = 1:3

disp(['Frame ', num2str(i), ': Length = ', num2str(max\_lengths(i)), ', Sequence = ', longest\_spans{i}]);

end

for i = 4:6

disp(['Frame ', num2str(-(i-3)), ': Length = ', num2str(max\_lengths(i)), ', Sequence = ', longest\_spans{i}]);

end

Longest uninterrupted span in each frame:

Frame 1: Length = 274, Sequence = PCHLPWSPHPRSVSDPHHDFLSCPQSGGGDLTLGLEPSEEEAPRSPLAPSEGAGSDVFDGDLGMGAAKGLQSLPTHDPSPLQRYSEDPTVPLPSETDGYVAPLTCSPQPGMESSLSRETDGQGRWDLQPRVHCGGRGSGRDTGVPSPNGSPSLDLSEYVNQPDVRPQPPSPREGPLPAARPAGATLERPKTLSPGKNGVVKDVFAFGGAVENPEYLTPQGGAAPQPHPPPAFSPAFDNLYYWDQDPPERGAPPSTFKGTPTAENPEYLGLDVPV

Frame 2: Length = 529, Sequence = VMGLSFRLHSLLTTKQHAQSRGEKEGESCGPHEHLDLAWTTHSSLALALFLLRVWWWWDSKTVKIAFSPPWGIWGLFKRPAPLLEGRRAPREAEGDRRGKGPLIIAHPTEILKGGVLIQRNPQLCYQDTILWKDIFHKNNQLALTLIDTNRSRACKPCPSLLPLLSDSLTPAANSQLTTQCLPATAPAAYTTHFLPLCPSCHLPVPLHLWGSLSCLPLLIGCASGLGASQPVWVPPLLCSWPRGLCCCLCLSLLLTRPLAVRHLSHCLSLVLSSGHPCSPMCKGSRCWGESSEDCQSRESQGGLESGKGRAGAGWNAGVIQVTWEGWDNRLGMSPLGQVVSLEGDADEGLVPRAPLSPHPALCPTVTRTVCAGGCARCKGPLPTDCCHEQCAAGCTGPKHSDCLVCASALCPMCSTPQDARGGHPAWYCPIAPGTPGQNSTVKASHLSPQACLHFNHSGICELHCPALVTYNTDTFESMPNPEGRYTFGASCVTACPCECQGETQFSHFGGEVCFCKWEHMGSTVCILL

Frame 3: Length = 246, Sequence = APWSWRPCAAGGSSSPSCPPEPRAPKVGLVWGGDGAAAGPCPVDAPPRSRGRRGQRGPDELSYPEVVDSRDAQGSRALGPSGGRGQLHGSGSRWPIQETGAFQAPRGSGNLSKKFSEIVQKVFPQRVYCVERARAFPPFLSPLKLSQSLSSWQPPPPDWPGLDSLGGSSALPLLQPLPAPLQTILVWLLLLLAGSGVCVCGGVEGGIATCPNQSRGGKGGPEGGLLLGSGLGAGETFALNRFLGPA

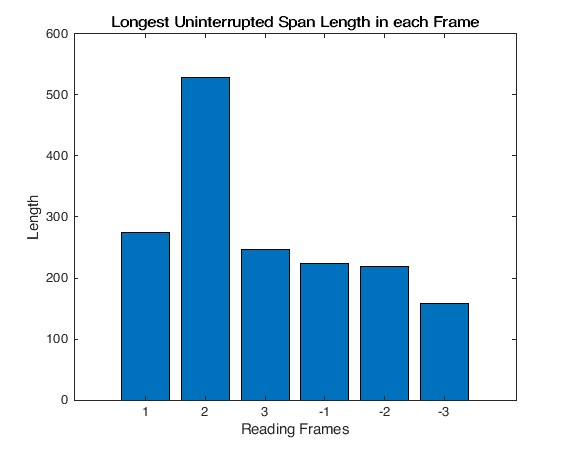
Frame -1: Length = 224, Sequence = GSSGSLCSLGRGSLPRSPLDPRRRSLPRSGSQSRGGGETPVLSFSTTPSECLDPTPEVPSTVPVSSFCLQWKPHTVSVAPLLPQCPQPSVILKKHPYYPKGTLHCQLPCSLFPWDQVAPTPSIRLSVCHTGSRDWDHLLDATDTTWYGGRGPRPRPVFFGTTPWSMRSRRWSRGYSSRRSSLAIFHDRFPDRPGFRSKTPPSSPSYHRGGVGSYVGRHSPDRMT

Frame -2: Length = 219, Sequence = RVVPSLGLSDSPGVWSPCWIFVSLCSVSWGLYFSVSLYRPPCLSLLYPPHPPTPSPPVRDPTFNPQTPTPVRQTLFSVVPPRTECPGSRQPQIVTLSVDQQEHLLQEGVLAQDHRLDPQGDLVLGRKVLETPHSLLVSWERGRQWKTKGTPGGRKIFSSSGKFVGGLRGPSSLSIEMAETQGGGGVGVVLVSLSVPPHSTGTIQVYEYTWRRERKEEWR

Frame -3: Length = 158, Sequence = GTLPFVGTSGRPSRTRTGVRVVLRGRIGEHLYPQTRPLWTFLGSTYGTVRRMWSSRPGPSGPQAPKGPVPDSPSVSRIRHSGRRGTYSPSTGLPPSVYPSGVVGSAYEGPLFPCDSPIPSGVWGFVVCRRVGSVWCRYRVSDTSEFHGRTCCTTLGQR

Step #17: Produce another plot to display the longest uninterrupted span in each frame

% Create a bar plot

Figure 8: Longest uninterrupted span in each span

figure;

bar(max\_lengths);

% Customize the plot

xticks(1:6);

xticklabels({'1', '2', '3', '-1', '-2', '-3'});

xlabel('Reading Frames');

ylabel('Length');

title('Longest Uninterrupted Span Length in each Frame');

% Display the plot

shg;

From the graph longest uninterrupted span is in forward frame 2, second longest is in forward frame 1, then forward frame 3, reverse frame 1, reverse frame 2 and reverse frame 3 consecutively.

Step #18: Find the corresponding protein sequence that has been coded by your gene. To which open reading frame does it correspond?

We found a protein with accession number NP\_001369735.1 which corresponded to open reading frame 1 and 2 in forward strand.

We took help of blastp. We only got match for forward frame 1 & 2 and for others no match found.

Step #19: Using the tool at https://molbiotools.com/restrictionanalyzer.php, find all possible restriction sites and the corresponding restriction enzymes.

| Enzyme | Recognition Sequence | Frequency | Cutting Positions |
| --- | --- | --- | --- |
| AgeI | ACCGGT | 1 | 28509 |
| EcoRV | GATATC | 1 | 20230 |
| SpeI | ACTAGT | 1 | 16716 |
| AatII | GACGTC | 2 | 25266, 35236 |
| BstBI | TTCGAA | 2 | 13321, 20428 |
| EagI | CGGCCG | 2 | 12290, 19623 |
| HpaI | GTTAAC | 2 | 26040, 29648 |
| SacII | CCGCGG | 2 | 12290, 36767 |
| SfiI | GGCC(N)5GGCC | 2 | 12298, 12307 |
| AflII | CTTAAG | 3 | 4721, 11424, 13661 |
| MfeI | CAATTG | 3 | 606, 14491, 30742 |
| PsiI | TTATAA | 3 | 1913, 22708, 26427 |
| FspI | TGCGCA | 4 | 7036, 15439, 24002, 27869 |
| PmlI | CACGTG | 4 | 3321, 14077, 22631, 34521 |
| AfeI | AGCGCT | 5 | 14904, 16065, 28638, 32659, 38549 |
| BspEI | TCCGGA | 5 | 3342, 8720, 12608, 14098, 36847 |
| EcoRI | GAATTC | 5 | 17773, 25755, 27431, 32074, 38519 |
| NaeI | GCCGGC | 5 | 4738, 12294, 19623, 22075, 37273 |
| XbaI | TCTAGA | 5 | 2539, 4223, 15269, 24582, 38380 |
| AseI | ATTAAT | 6 | 2301, 3681, 3722, 13106, 29551, 29657 |
| BspHI | TCATGA | 6 | 3769, 18511, 22494, 26187, 33568, 39177 |
| KpnI | GGTACC | 6 | 8446, 9481, 16637, 17633, 23958, 38827 |
| SacI | GAGCTC | 6 | 12322, 18625, 30751, 31425, 32535, 34057 |
| XhoI | CTCGAG | 6 | 2224, 8177, 12413, 26799, 34048, 34205 |
| BglII | AGATCT | 7 | 6448, 9663, 9964, 13677, 20142, 21225, 24313 |
| NdeI | CATATG | 7 | 2756, 20842, 22455, 29728, 31209, 31834, 36665 |
| ScaI | AGTACT | 7 | 14464, 20153, 27524, 28331, 37993, 39769, 40356 |
| BsrGI | TGTACA | 8 | 4127, 9202, 14039, 20696, 31297, 33616, 37947, 40405 |
| BssHII | GCGCGC | 8 | 11881, 12017, 12051, 12533, 12535, 12537, 12539, 26820 |
| DraI | TTTAAA | 8 | 38, 4329, 4638, 11367, 21127, 30844, 33023, 40455 |
| KasI | GGCGCC | 8 | 64, 17944, 21955, 22361, 22949, 25686, 27648, 31038 |
| NarI | GGCGCC | 8 | 65, 17945, 21956, 22362, 22950, 25687, 27649, 31039 |
| BsrBI | CCGCTC | 9 | 12675, 21348, 27217, 31489, 35703, 37004, 37705, 38768, 39870 |
| HindIII | AAGCTT | 9 | 547, 7220, 7350, 9941, 12570, 16945, 18723, 27504, 29034 |
| NsiI | ATGCAT | 9 | 4120, 7610, 11013, 13179, 13319, 26074, 30835, 33679, 34551 |
| StuI | AGGCCT | 10 | 1850, 7483, 10913, 14294, 15674, 16878, 17420, 20409, 21133, 40005 |
| AvrII | CCTAGG | 11 | 2415, 3534, 4891, 7323, 10523, 11662, 13001, 14295, 17646, 26645, 35715 |
| SmaI | CCCGGG | 11 | 5589, 11473, 12008, 16404, 20567, 26769, 31428, 32528, 34716, 37673, 38861 |
| SspI | AATATT | 11 | 837, 2299, 4064, 6853, 9135, 15520, 22728, 26447, 29594, 29661, 32513 |
| ApaLI | GTGCAC | 12 | 4083, 13342, 14710, 18898, 23787, 25695, 27762, 27953, 29620, 29686, 31129, 37452 |
| BamHI | GGATCC | 13 | 1441, 18408, 23858, 24408, 30879, 31213, 34185, 35294, 35497, 35771, 35826, 37662, 38016 |
| XmnI | GAANNNNTTC | 13 | 4251, 7409, 7423, 12493, 12504, 12861, 13408, 14397, 16560, 18585, 18687, 27100, 28711 |
| MscI | TGGCCA | 14 | 9098, 10463, 13713, 14886, 15298, 16692, 17554, 17561, 23013, 23305, 27128, 30515, 35861, 37183 |
| NcoI | CCATGG | 14 | 7287, 7687, 8490, 12118, 12144, 12915, 18066, 23252, 26033, 35961, 36543, 37163, 37348, 39139 |
| ApaI | GGGCCC | 16 | 12038, 12312, 12379, 13496, 14978, 15048, 21163, 22088, 24143, 28695, 29050, 33664, 35165, 38737, 39649, 40040 |
| SphI | GCATGC | 17 | 1685, 2139, 4460, 5966, 7608, 9411, 13177, 18221, 20948, 23246, 24007, 25531, 29357, 31498, 33328, 38447, 38676 |
| PvuII | CAGCTG | 25 | 4855, 11254, 14227, 15127, 15618, 15840, 17144, 17249, 17266, 18151, 18352, 18364, 19087, 21319, 22368, 22533, 23981, 27067, 27685, 28233, 31693, 32052, 34308, 35444, 36710 |
| Bsu36I | CCTNAGG | 32 | 1595, 3992, 4543, 4752, 5778, 5875, 7964, 9494, 10915, 11149, 14868, 14979, 15888, 16128, 17426, 17659, 17803, 18171, 21715, 24404, 27515, 28969, 29950, 30950, 33412, 36140, 37609, 38350, 39988, 40084, 40266, 40299 |
| PstI | CTGCAG | 38 | 2134, 5468, 8913, 11755, 13390, 13532, 13652, 13905, 15544, 17244, 18245, 18749, 19047, 19215, 20292, 20424, 20803, 23986, 26231, 27296, 27351, 27560, 28206, 28288, 28337, 29411, 29800, 31483, 31692, 32057, 32600, 32945, 32980, 34350, 35354, 35971, 38619, 39442 |

Step #20: Search all of them in your gene using the matlab function “regexpi”.

% Define the enzyme patterns

patterns = {'ACCGGT';'GATATC';'ACTAGT';'GACGTC';'TTCGAA';'CGGCCG';'GTTAAC';

'CCGCGG';'GGCC[ACGT]{5}GGCC';'CTTAAG';'CAATTG';'TTATAA';'TGCGCA';'CACGTG';

'AGCGCT';'TCCGGA';'GAATTC';'GCCGGC';'TCTAGA';'ATTAAT';'TCATGA';'GGTACC';

'GAGCTC';'CTCGAG';'AGATCT';'CATATG';'AGTACT';'TGTACA';'GCGCGC';'TTTAAA';

'GGCGCC';'GGCGCC';'CCGCTC';'AAGCTT';'ATGCAT';'AGGCCT';'CCTAGG';'CCCGGG';

'AATATT';'GTGCAC';'GGATCC';'GAA[ACGT]{4}TTC';'TGGCCA';'CCATGG';'GGGCCC';

'GCATGC';'CAGCTG';'CCT[ACGT]AGG';'CTGCAG'};

% Find start position of each pattern

indices = regexpi(sequence, patterns)

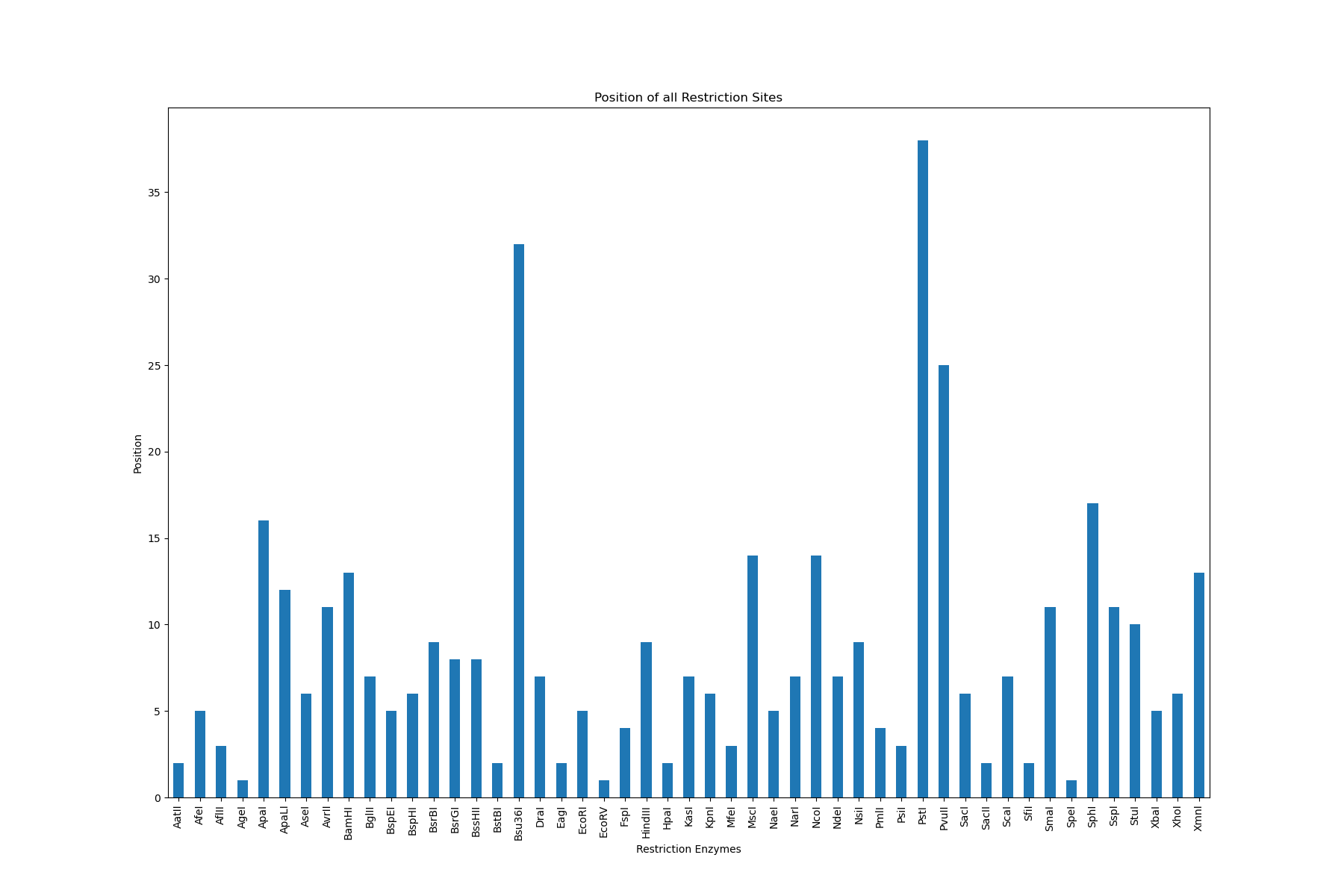
% Display the indices

disp(indices)

Step #21: Which substrings (pattern) had the highest amount of occurrence?

Substring for PstI enzyme had highest amount of occurrence i.e., CTGCAG occurred 38 times.

Step #22: Create a bar plot that shows the position of all restriction sites.

Figure 9: Position of all restriction sites

Step #23: Find another protein which would be functionally related to your own protein.

We choose a protein from Mus musculus named “receptor tyrosine-protein kinase erbB-2 precursor [Mus musculus]” with accession id NP\_001003817.1. We took help of STRING and blastp to choose that functionally related protein.

**Step #24:** Via the link https://www.ebi.ac.uk/jdispatcher/psa (EMBL’s European Bioinformatics Institute), perform a pairwise sequence alignment using EMBOSS Needle. If your sequence is too long, you can use EMBOSS Stretcher instead. Perform the alignment using default parameters.

########################################

# Program: needle

# Rundate: Thu 30 May 2024 17:30:44

# Commandline: needle

# -auto

# -stdout

# -asequence emboss\_needle-I20240530-173042-0489-77864996-p1m.asequence

# -bsequence emboss\_needle-I20240530-173042-0489-77864996-p1m.bsequence

# -datafile EBLOSUM62

# -gapopen 10.0

# -gapextend 0.5

# -endopen 10.0

# -endextend 0.5

# -aformat3 pair

# -sprotein1

# -sprotein2

# Align\_format: pair

# Report\_file: stdout

########################################

#=======================================

#

# Aligned\_sequences: 2

# 1: NP\_001369735.1

# 2: NP\_001003817.1

# Matrix: EBLOSUM62

# Gap\_penalty: 10.0

# Extend\_penalty: 0.5

#

# Length: 1256

# Identity: 804/1256 (64.0%)

# Similarity: 837/1256 (66.6%)

# Gaps: 347/1256 (27.6%)

# Score: 4151.5

#

#

#=======================================

**Step #25:** State those default parameters (e.g., matrix, gap open, gap extend).

**Matrix**: A substitution matrix is a table that quantifies the similarity between amino acids in biological sequences. BLOSUM62 is a widely used substitution matrix for protein sequences.

**Gap Open Penalty**: This parameter determines the cost associated with opening a gap in sequence alignment. In EMBOSS Needle, the default gap open penalty is 10 for proteins.

**Gap Extension Penalty**: Gap extension penalty is the cost of extending an existing gap in sequence alignment. In EMBOSS Needle, the default gap extension penalty is 0.5 for proteins.

**Step #26:** Repeat the pairwise sequence alignment using three different gap “open” penalty values and compare the results (leave the gap extend at default).

|  |  |  |
| --- | --- | --- |
| # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  # Matrix: EBLOSUM62  **# Gap\_penalty: 1.0**  # Extend\_penalty: 0.5  # Length: 1267  # Identity: 812/1267 (64.1%)  # Similarity: 845/1267 (66.7%)  # Gaps: 369/1267 (29.1%)  # Score: 4221.5 | # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  # Matrix: EBLOSUM62  **# Gap\_penalty: 5.0**  # Extend\_penalty: 0.5  # Length: 1257  # Identity: 805/1257 (64.0%)  # Similarity: 839/1257 (66.7%)  # Gaps: 349/1257 (27.8%)  # Score: 4164.5 | # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  # Matrix: EBLOSUM62  **# Gap\_penalty: 100.0**  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 805/1256 (64.1%)  # Similarity: 838/1256 (66.7%)  # Gaps: 347/1256 (27.6%)  # Score: 4126.733 |

**Step #27:** Repeat the pairwise sequence alignment using three different gap “extend” penalty values and compare the results (leave the gap open at default).

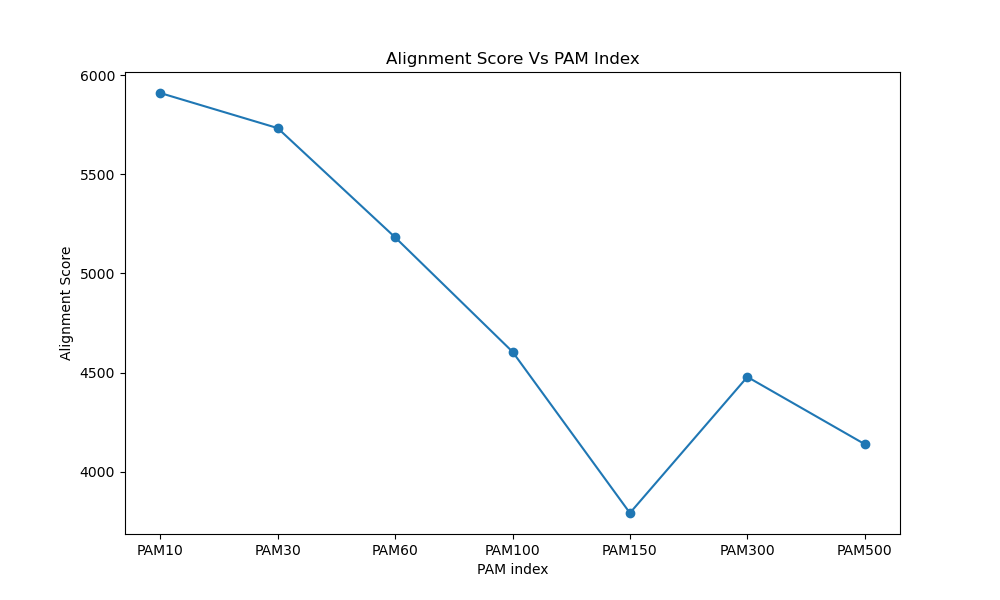
|  |  |  |
| --- | --- | --- |
| # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  # Matrix: EBLOSUM62  # Gap\_penalty: 10.0  **# Extend\_penalty: 0.0**  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 837/1256 (66.6%)  # Gaps: 347/1256 (27.6%)  # Score: 4324.0 | # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  # Matrix: EBLOSUM62  # Gap\_penalty: 10.0  **# Extend\_penalty: 0.05**  # Length: 1267  # Identity: 811/1267 (64.0%)  # Similarity: 845/1267 (66.7%)  # Gaps: 369/1267 (29.1%)  # Score: 4369.434 | # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  # Matrix: EBLOSUM62  # Gap\_penalty: 10.0  **# Extend\_penalty: 10.0**  # Length: 1264  # Identity: 548/1264 (43.4%)  # Similarity: 621/1264 (49.1%)  # Gaps: 363/1264 (28.7%)  # Score: 2301.0 |

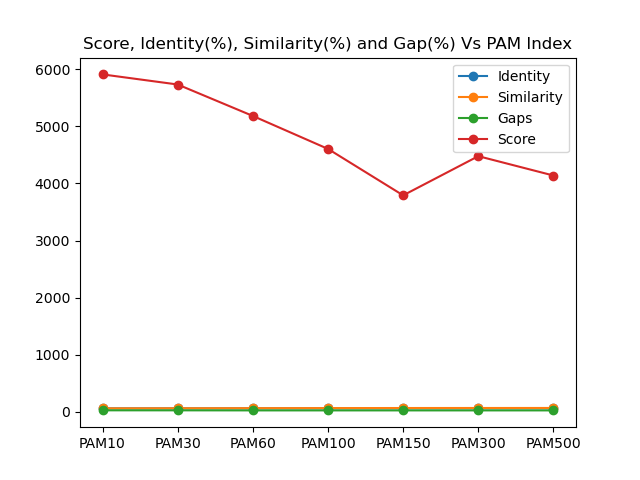
**Step #28:** Repeat the pairwise sequence alignment using all possible PAM matrices (10, 20, ...500). Use default parameters for gap open and gap extend.

|  |  |  |  |
| --- | --- | --- | --- |
| # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  **# Matrix: EPAM10**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1265  # Identity: 807/1265 (63.8%)  # Similarity: 807/1265 (63.8%)  # Gaps: 365/1265 (28.9%)  # Score: 5910.0 | # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  **# Matrix: EPAM30**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1263  # Identity: 806/1263 (63.8%)  # Similarity: 823/1263 (65.2%)  # Gaps: 361/1263 (28.6%)  # Score: 5733.0 | # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  **# Matrix: EPAM60**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 838/1256 (66.7%)  # Gaps: 347/1256 (27.6%)  # Score: 5182.5 | # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  **# Matrix: EPAM100**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 855/1256 (68.1%)  # Gaps: 347/1256 (27.6%)  # Score: 4605.5 |

|  |  |  |
| --- | --- | --- |
| # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  **# Matrix: EPAM150**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 870/1256 (69.3%)  # Gaps: 347/1256 (27.6%)  # Score: 3792.5 | # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  **# Matrix: EPAM300**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 876/1256 (69.7%)  # Gaps: 347/1256 (27.6%)  # Score: 4478.5 | # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  **# Matrix: EPAM500**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 880/1256 (70.1%)  # Gaps: 347/1256 (27.6%)  # Score: 4139.5 |

Step #29: Create a plot (linepoint) for score values, identity %, similarity %, and gap % versus PAM index values.

Figure 10: Alignment Score Vs PAM index

Figure 11: Linepoint for score values, identity %, similarity %, and gap % versus PAM index

PAM matrices are designed to reflect evolutionary distances. Lower PAM values (e.g., PAM10, PAM30 etc.) are suitable for closely related sequences, while higher PAM values (e.g., PAM250, PAM300 etc.) are used for more distantly related sequences.

The choice of different PAM matrix affects how matches, mismatches, and gaps are scored. Different matrices may assign different scores to the same alignment, leading to variations in the optimal alignment and its overall score.

In our case, we have approximately 64% identical residues. Typically, protein sequences with >30-40% identity are considered closely related. This level of identity usually indicates that the sequences have a similar structure and function.

As we have 64% identity, we can say both species are closely related. And for closely related sequence, PAM matrices with lower values are provides biologically significant alignment.

In the graph, we get a line with downward slopping for different PAM matrix i.e., high score for PAM10 and lower score for towards PAM500.

**Step #30:** Which PAM matrix best represent the alignment of these two results?

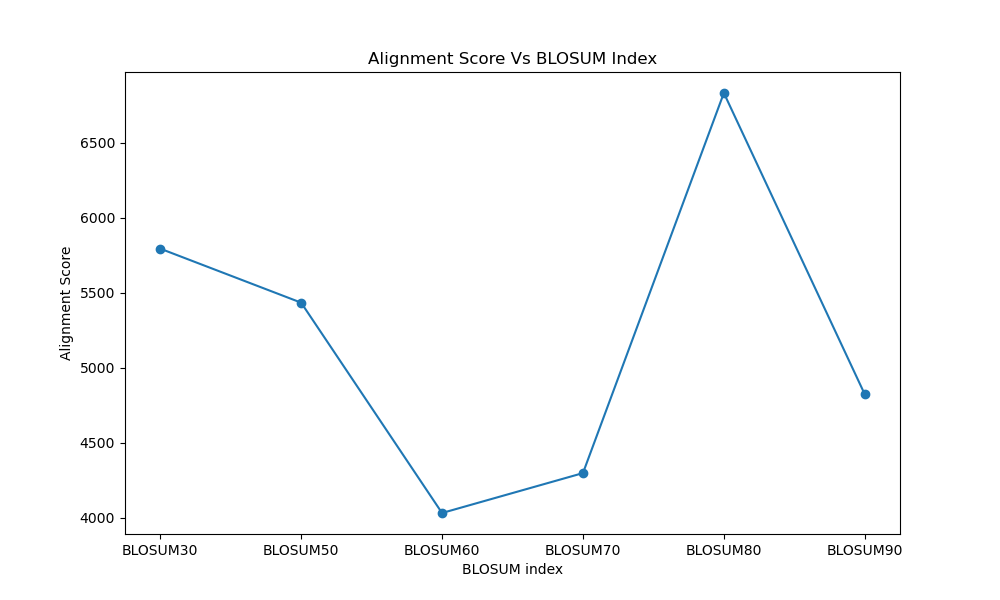
As PAM10 is used for closely related protein sequence and we get high score for PAM10 substitution matrix, we can consider the alignment as biologically plausible. So, PAM10 best represent the above alignment.

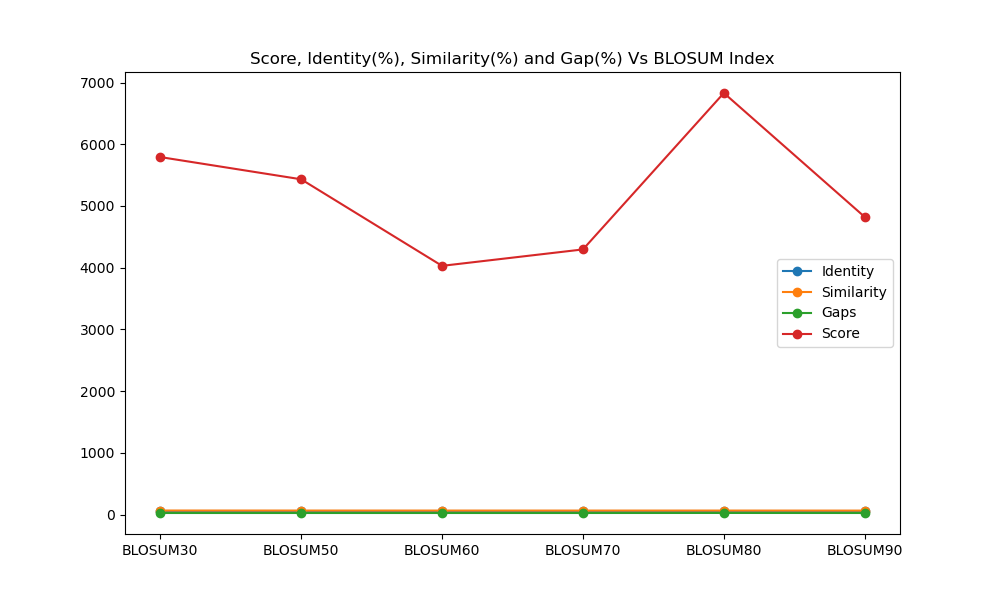
**Step #31:** Repeat the pairwise sequence alignment using all possible BLOSUM matrices (30, 35, ...90).

|  |  |  |
| --- | --- | --- |
| # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  **# Matrix: EBLOSUM30**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 848/1256 (67.5%)  # Gaps: 347/1256 (27.6%)  # Score: 5792.5 | # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  **# Matrix: EBLOSUM50**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 841/1256 (67.0%)  # Gaps: 347/1256 (27.6%)  # Score: 5432.5 | # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  **# Matrix: EBLOSUM60**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 839/1256 (66.8%)  # Gaps: 347/1256 (27.6%)  # Score: 4030.5 |

|  |  |  |
| --- | --- | --- |
| # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  **# Matrix: EBLOSUM70**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 836/1256 (66.6%)  # Gaps: 347/1256 (27.6%)  # Score: 4296.5 | # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  **# Matrix: EBLOSUM80**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1260  # Identity: 806/1260 (64.0%)  # Similarity: 842/1260 (66.8%)  # Gaps: 355/1260 (28.2%)  # Score: 6830.0 | # Aligned\_sequences: 2  # 1: NP\_001369735.1  # 2: NP\_001003817.1  **# Matrix: EBLOSUM90**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 831/1256 (66.2%)  # Gaps: 347/1256 (27.6%)  # Score: 4821.5 |

**Step #32:** Create a plot (linepoint) for score values, identity %, similarity %, and gap % versus BLOSUM index values.

Figure 12: Alignment Score Vs BLOSUM index

Figure 13: Linepoint for score values, identity %, similarity %, and gap % versus BLOSUM index

BLOSUM matrices are designed to align protein sequences based on observed substitutions in conserved blocks of sequences. Unlike PAM matrices, BLOSUM matrices are numbered inversely to the degree of evolutionary divergence they represent (i.e., higher numbers for closely related sequences).

#### Suitable BLOSUM Matrices for Different Identity Percentages:

**BLOSUM80 to BLOSUM90:** Suitable for aligning closely related protein sequences with high identity percentages (e.g., >60% identity). Designed for less divergent sequences.

**BLOSUM60 to BLOSUM70:** Suitable for sequences with moderate identity (e.g., 30-60%). Accommodate more evolutionary changes.

**BLOSUM30 to BLOSUM50:** Suitable for distantly related sequences with lower identity percentages (e.g., <30-40%). Handle significant evolutionary divergence.

With an identity percentage of 64%, the sequences are quite closely related. This level of identity indicates substantial similarity in structure and function, and a high degree of evolutionary conservation.

### High Score with BLOSUM80

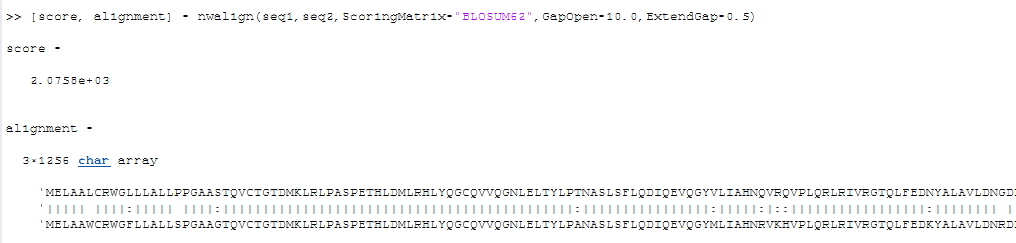
Since BLOSUM80 is designed for aligning sequences with around 80% identity, its use for sequences with 64% identity is reasonable.

It implies that BLOSUM80 can effectively capture the substitutions and similarities typical of closely related sequences.

**Step #33:** Which BLOSUM matrix best represent the alignment of these two results?

The sequences have 64% identity and BLOSUM80 yielded the highest score among the tested matrices, it indicates that BLOSUM80 is well-suited for aligning the sequences. The high score with BLOSUM80 suggests that it provides an accurate and meaningful alignment for closely related sequences, reflecting the observed substitutions effectively. So, BLOSUM80 matrix best represent the alignment.

Step #34: Repeat the pairwise sequence alignment using nwalign function of MATLAB and the default alignment parameters of EMBOSS Needle. Is there any difference between the two results?



Step #35: Perform a pairwise sequence alignment using EMBOSS Water. Perform the alignment using default parameters.

# Aligned\_sequences: 2

# 1: NP\_001003817.1

# 2: NP\_001369735.1

# Matrix: EBLOSUM62

# Gap\_penalty: 10.0

# Extend\_penalty: 0.5

# Length: 1256

# Identity: 804/1256 (64.0%)

# Similarity: 837/1256 (66.6%)

# Gaps: 347/1256 (27.6%)

# Score: 4151.5

**Step #36:** State those default parameters.

#### Substitution Matrix

* **PAM (Point Accepted Mutation) Matrices:** Used for proteins, with different versions (e.g., PAM30, PAM250 etc.) indicating the evolutionary distance.
* **BLOSUM (Blocks Substitution Matrix) Matrices:** Also used for proteins, with versions like BLOSUM62, BLOSUM80, BLOSUM90 etc., representing different levels of sequence similarity.
* **Nucleotide Substitution Matrices:** For DNA sequences, simpler matrices like the identity matrix can be used.

#### Gap Penalty

* **Gap Opening Penalty (gap\_open):** The score deducted when a gap is first introduced into the alignment. It is typically a larger penalty to discourage opening new gaps.
* **Gap Extension Penalty (gap\_extend):** The score deducted for extending an existing gap. It is usually smaller than the gap opening penalty and is applied for each additional gap position.

**Step #37:** Repeat the pairwise sequence alignment using three different gap “open” penalty values and compare the results.

|  |  |  |
| --- | --- | --- |
| # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  # Matrix: EBLOSUM62  **# Gap\_penalty: 1.0**  # Extend\_penalty: 0.5  # Length: 1267  # Identity: 812/1267 (64.1%)  # Similarity: 845/1267 (66.7%)  # Gaps: 369/1267 (29.1%)  # Score: 4221.5 | # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  # Matrix: EBLOSUM62  **# Gap\_penalty: 25.0**  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 837/1256 (66.6%)  # Gaps: 347/1256 (27.6%)  # Score: 4121.5 | # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  # Matrix: EBLOSUM62  **# Gap\_penalty: 100.0**  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 837/1256 (66.6%)  # Gaps: 347/1256 (27.6%)  # Score: 3971.5 |

Step #38: Repeat the pairwise sequence alignment using three different gap “extend” penalty values and compare the results.

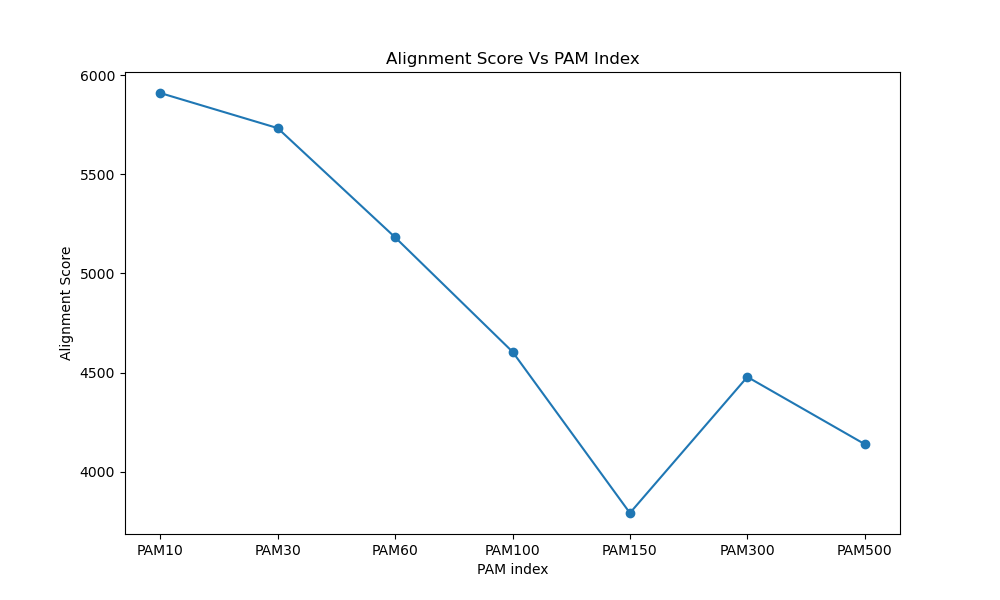
|  |  |  |
| --- | --- | --- |
| # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  # Matrix: EBLOSUM62  # Gap\_penalty: 10.0  **# Extend\_penalty: 0.1**  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 837/1256 (66.6%)  # Gaps: 347/1256 (27.6%)  # Score: 4289.5 | # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  # Matrix: EBLOSUM62  # Gap\_penalty: 10.0  **# Extend\_penalty: 5.0**  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 837/1256 (66.6%)  # Gaps: 347/1256 (27.6%)  # Score: 2599.0 | # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  # Matrix: EBLOSUM62  # Gap\_penalty: 10.0  **# Extend\_penalty: 10.0**  # Length: 504  # Identity: 457/504 (90.7%)  # Similarity: 471/504 (93.5%)  # Gaps: 0/504 ( 0.0%)  # Score: 2441.0 |

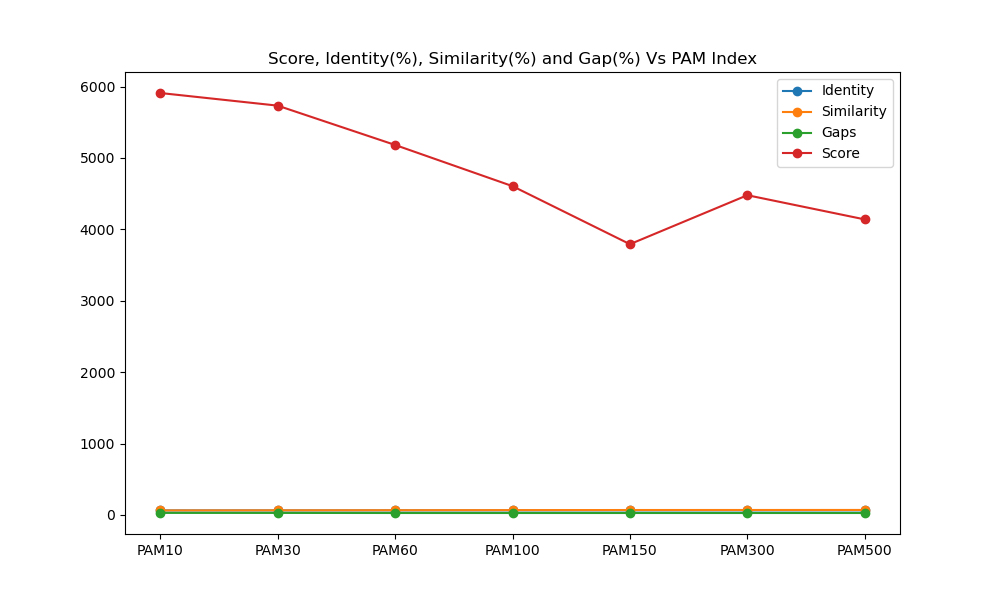
Step #39 Repeat the pairwise sequence alignment using all possible PAM matrices (10, 20, ...500).

|  |  |  |  |
| --- | --- | --- | --- |
| # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  **# Matrix: EPAM10**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1265  # Identity: 807/1265 (63.8%)  # Similarity: 807/1265 (63.8%)  # Gaps: 365/1265 (28.9%)  # Score: 5910.0 | # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  **# Matrix: EPAM30**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1263  # Identity: 806/1263 (63.8%)  # Similarity: 823/1263 (65.2%)  # Gaps: 361/1263 (28.6%)  # Score: 5733.0 | # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  **# Matrix: EPAM60**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 838/1256 (66.7%)  # Gaps: 347/1256 (27.6%)  # Score: 5182.5 | # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  **# Matrix: EPAM100**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 855/1256 (68.1%)  # Gaps: 347/1256 (27.6%)  # Score: 4605.5 |

|  |  |  |
| --- | --- | --- |
| # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  **# Matrix: EPAM150**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 870/1256 (69.3%)  # Gaps: 347/1256 (27.6%)  # Score: 3792.5 | # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  **# Matrix: EPAM300**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 876/1256 (69.7%)  # Gaps: 347/1256 (27.6%)  # Score: 4478.5 | # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  **# Matrix: EPAM500**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 880/1256 (70.1%)  # Gaps: 347/1256 (27.6%)  # Score: 4139.5 |

Step #40: Create a plot (linepoint) for score values, identity %, similarity %, and gap % versus PAM index values.

Figure 14: Alignment Score Vs PAM index

Figure 15: Linepoint for score values, identity %, similarity %, and gap % versus PAM index

Needleman-Wunsch algorithm and Smith-Waterman algorithm has similarities except alignment matrix initialization and backtracking.

PAM matrix behaviour is same for SW algorithm also. i.e., Lower PAM values (e.g., PAM10, PAM30 etc.) are suitable for closely related sequences, while higher PAM values (e.g., PAM250, PAM300 etc.) are used for more distantly related sequences.

The choice of different PAM matrix affects how matches, mismatches, and gaps are scored. Different matrices may assign different scores to the same alignment, leading to variations in the optimal alignment and its overall score.

In our case, we have approximately 64% identical residues. Typically, protein sequences with >30-40% identity are considered closely related. This level of identity usually indicates that the sequences have a similar structure and function.

As we have 64% identity, we can say both species are closely related. And for closely related sequence, PAM matrices with lower values are provides biologically significant alignment.

In the graph, we get a line with downward slopping for different PAM matrix i.e., high score for PAM10 and lower score for towards PAM500.

**Step #41:** Which PAM matrix best represent the alignment of these two results?

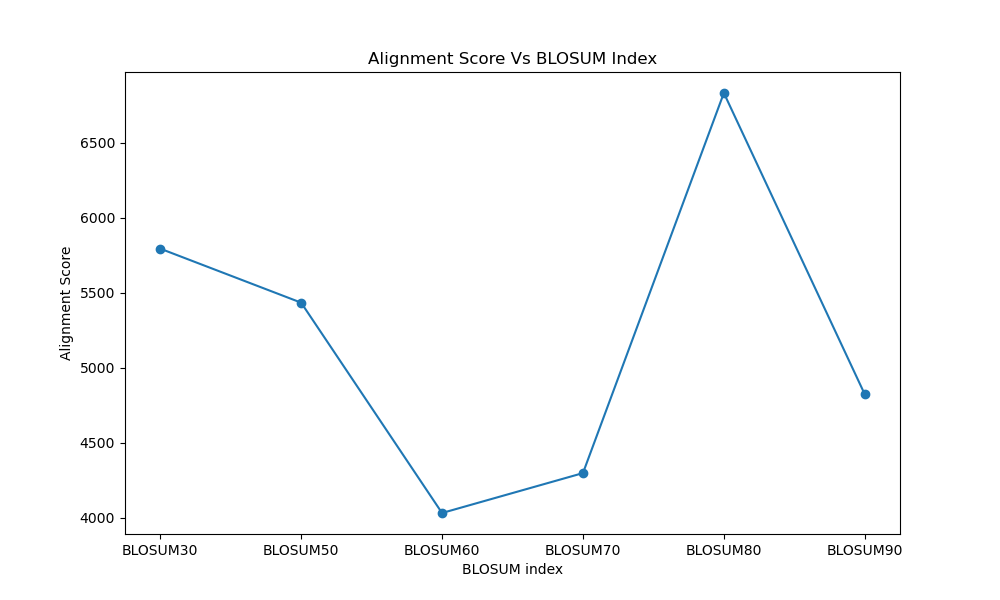
As PAM10 is used for closely related protein sequence and we get high score for PAM10 substitution matrix, we can consider the alignment as biologically plausible. So, PAM10 best represent the above alignment.

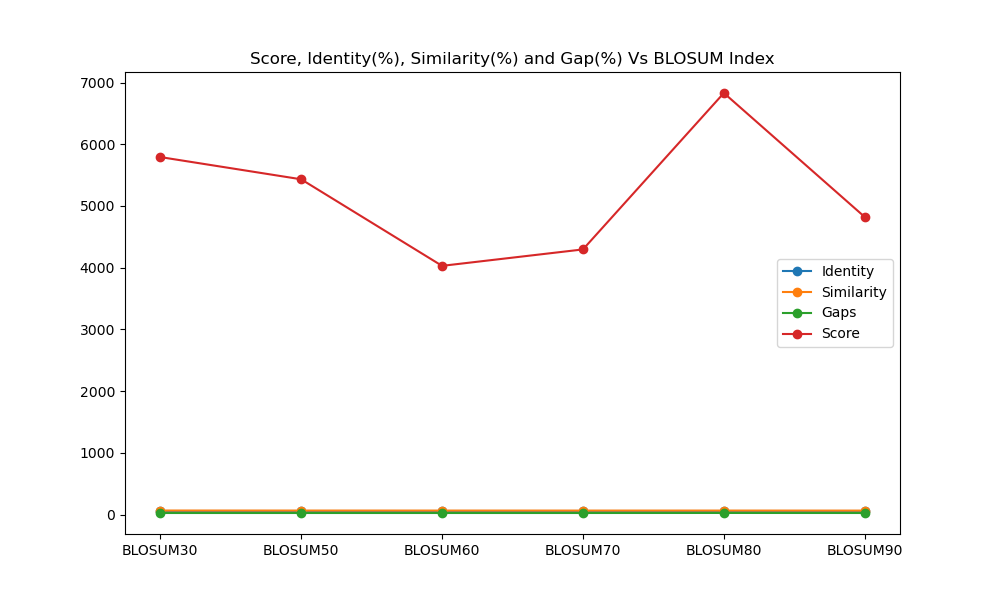
**Step #42:** Repeat the pairwise sequence alignment using all possible BLOSUM matrices (30, 35, ...90).

|  |  |  |
| --- | --- | --- |
| # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  **# Matrix: EBLOSUM30**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 848/1256 (67.5%)  # Gaps: 347/1256 (27.6%)  # Score: 5792.5 | # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  **# Matrix: EBLOSUM50**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 841/1256 (67.0%)  # Gaps: 347/1256 (27.6%)  # Score: 5432.5 | # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  **# Matrix: EBLOSUM60**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 839/1256 (66.8%)  # Gaps: 347/1256 (27.6%)  # Score: 4030.5 |

|  |  |  |
| --- | --- | --- |
| # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  **# Matrix: EBLOSUM70**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 836/1256 (66.6%)  # Gaps: 347/1256 (27.6%)  # Score: 4296.5 | # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  **# Matrix: EBLOSUM80**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1260  # Identity: 806/1260 (64.0%)  # Similarity: 842/1260 (66.8%)  # Gaps: 355/1260 (28.2%)  # Score: 6830.0 | # Aligned\_sequences: 2  # 1: NP\_001003817.1  # 2: NP\_001369735.1  **# Matrix: EBLOSUM90**  # Gap\_penalty: 10.0  # Extend\_penalty: 0.5  # Length: 1256  # Identity: 804/1256 (64.0%)  # Similarity: 831/1256 (66.2%)  # Gaps: 347/1256 (27.6%)  # Score: 4821.5 |

Step #43: Create a plot (linepoint) for score values, identity %, similarity %, and gap % versus BLOSUM index values.

Figure 16: Alignment Score Vs BLOSUM index

Figure 17: Linepoint for score values, identity %, similarity %, and gap % versus BLOSUM index

BLOSUM matrix shows similar characteristics as like NW algorithm in SW algorithm also. Unlike PAM matrices, BLOSUM matrices are numbered inversely to the degree of evolutionary divergence they represent (i.e., higher numbers for closely related sequences).

#### Suitable BLOSUM Matrices for Different Identity Percentages:

**BLOSUM80 to BLOSUM90:** Suitable for aligning closely related protein sequences with high identity percentages (e.g., >60% identity). Designed for less divergent sequences.

**BLOSUM60 to BLOSUM70:** Suitable for sequences with moderate identity (e.g., 30-60%). Accommodate more evolutionary changes.

**BLOSUM30 to BLOSUM50:** Suitable for distantly related sequences with lower identity percentages (e.g., <30-40%). Handle significant evolutionary divergence.

With an identity percentage of 64%, the sequences are quite closely related. This level of identity indicates substantial similarity in structure and function, and a high degree of evolutionary conservation.

### High Score with BLOSUM80

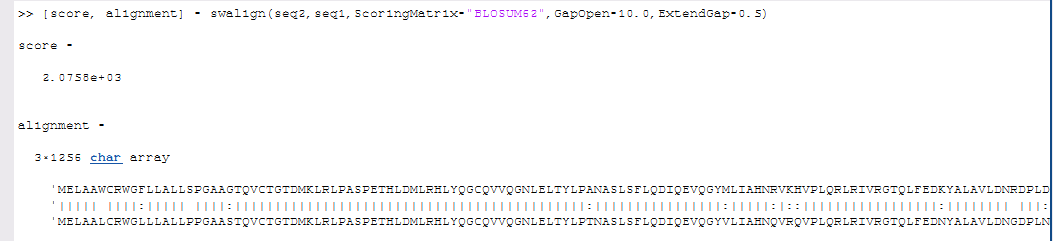
Since BLOSUM80 is designed for aligning sequences with around 80% identity, its use for sequences with 64% identity is reasonable.

It implies that BLOSUM80 can effectively capture the substitutions and similarities typical of closely related sequences.

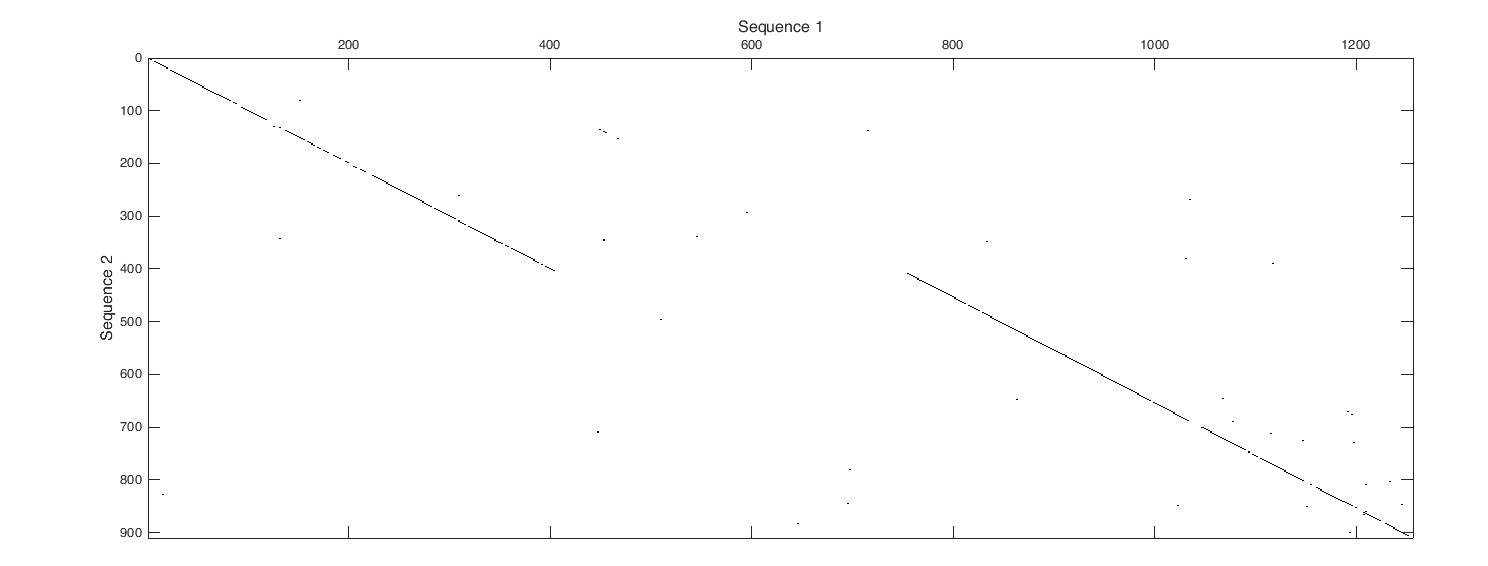
**Step #44:** Which BLOSUM matrix best represent the alignment of these two results?

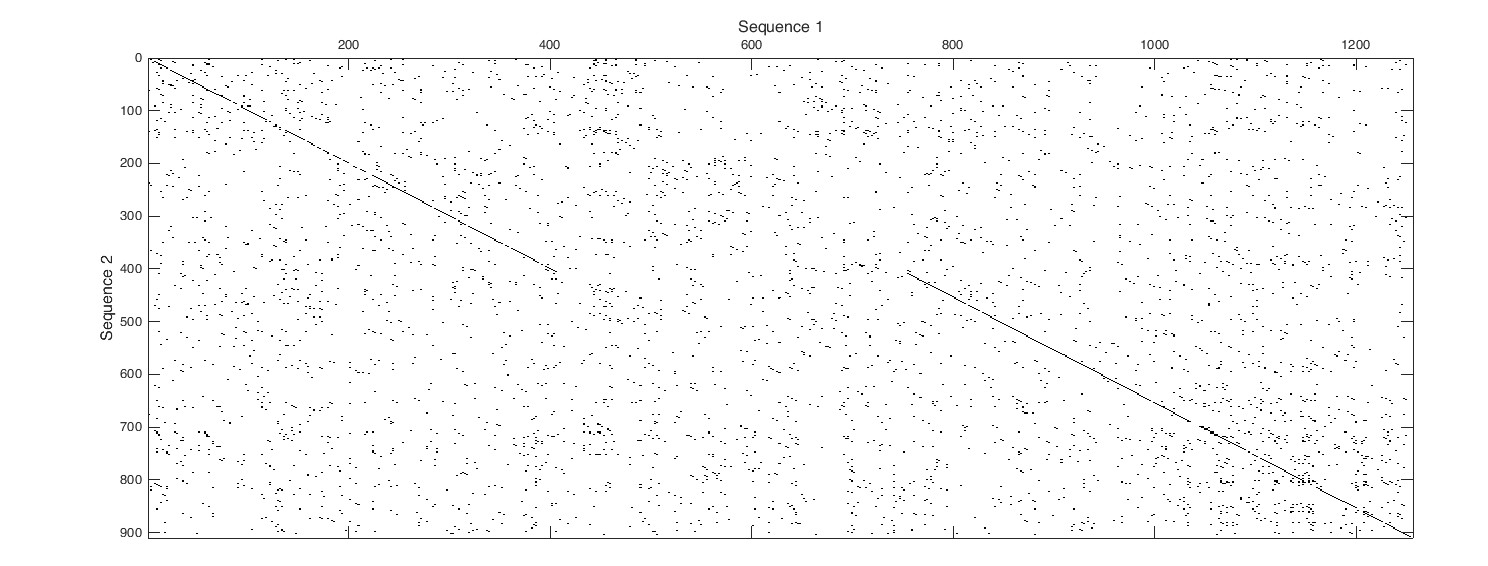
The sequences have 64% identity and BLOSUM80 yielded the highest score among the tested matrices, it indicates that BLOSUM80 is well-suited for aligning the sequences. The high score with BLOSUM80 suggests that it provides an accurate and meaningful alignment for closely related sequences, reflecting the observed substitutions effectively. So, BLOSUM80 matrix best represent the alignment.

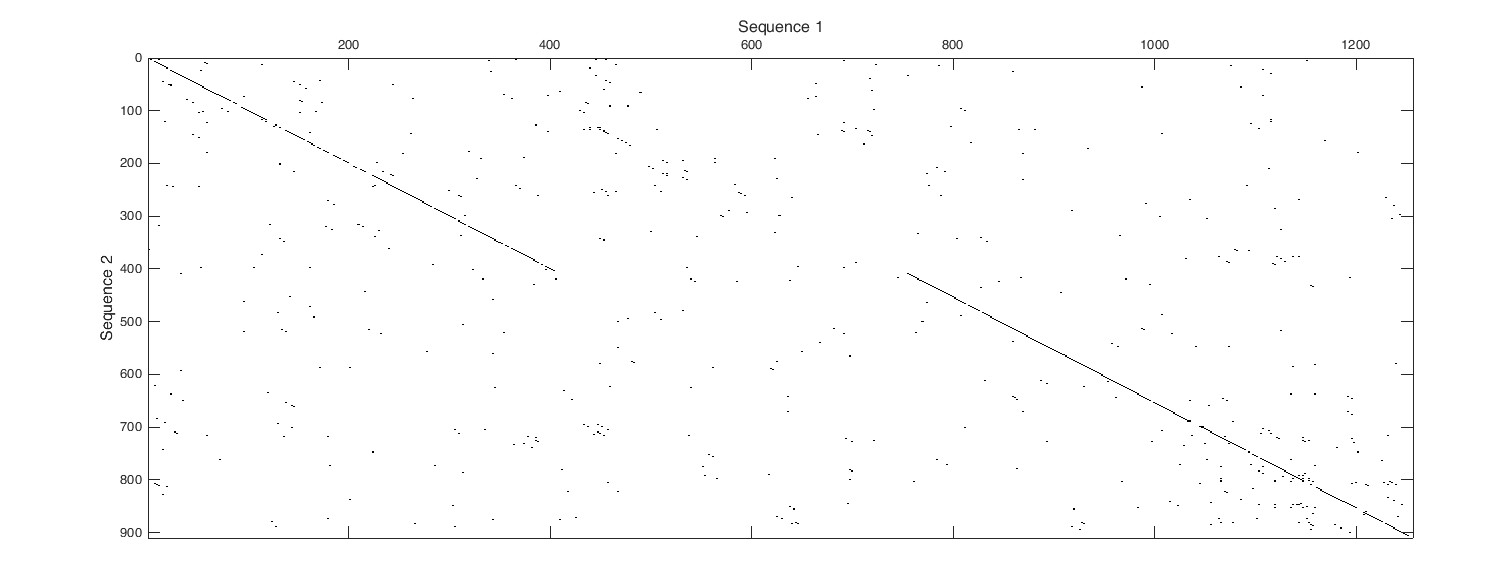
Step #45: Repeat the pairwise sequence alignment using swalign function of MATLAB and the default alignment parameters of EMBOSS Needle. Is there any difference between the two results?

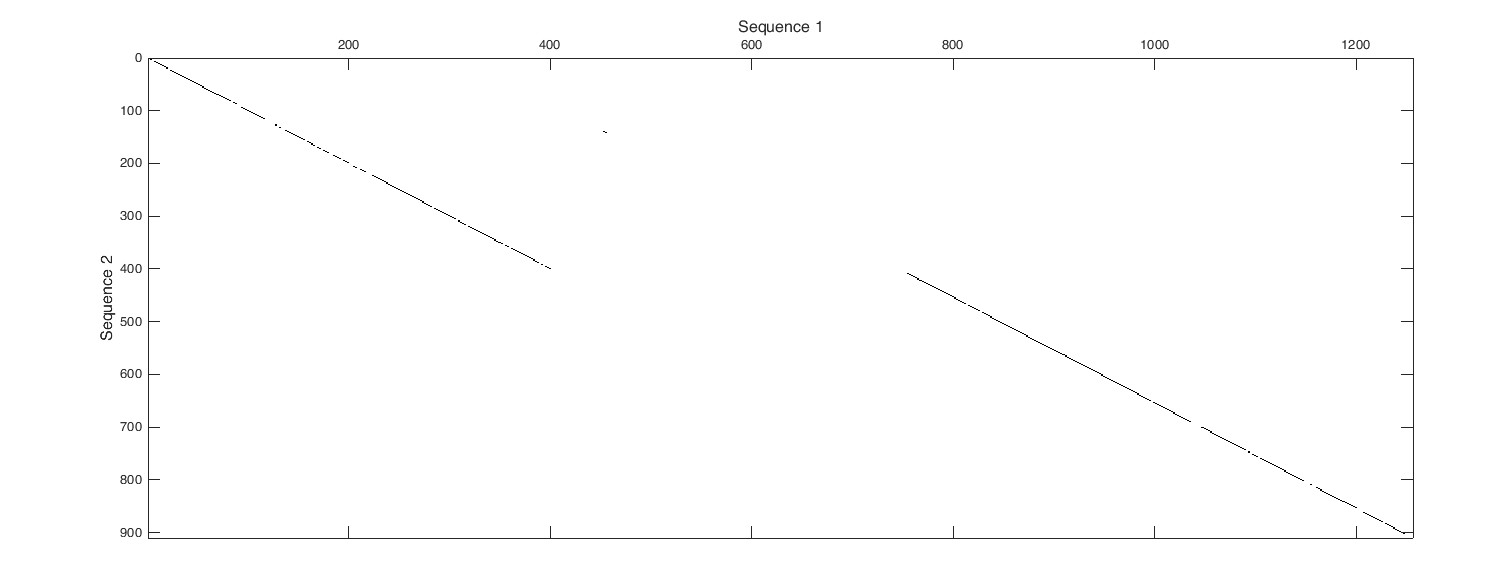


Step #46: Using seqdotplot function of MATLAB, perform a dot plot analysis between your protein sequence and the related sequence for four different sets of window size and stringency values. Which set bets represent the correspondence between two sequences.

Figure 18: Dot plot for window size 9 and stringency value 5

Figure 19: Dot plot for window size 11 and stringency value 3

Figure 20: Dot plot for window size 17 and stringency value 5

Figure 21: Dot plot for window size 17 and stringency value 9

### Seqdotplot Parameters:

#### Window Size

* The window size determines the length of the sequence segments being compared.
* A larger window size results in a smoother plot but may overlook smaller regions of similarity.
* A smaller window size provides more detail but may also include more noise.

#### Integer Value (Stringency)

* The integer value sets the stringency level for what is considered a match within the window.
* A higher value requires more matches within the window for a dot to be plotted, resulting in a cleaner plot with fewer dots, but potentially missing weaker similarities.

In seqdotplot:

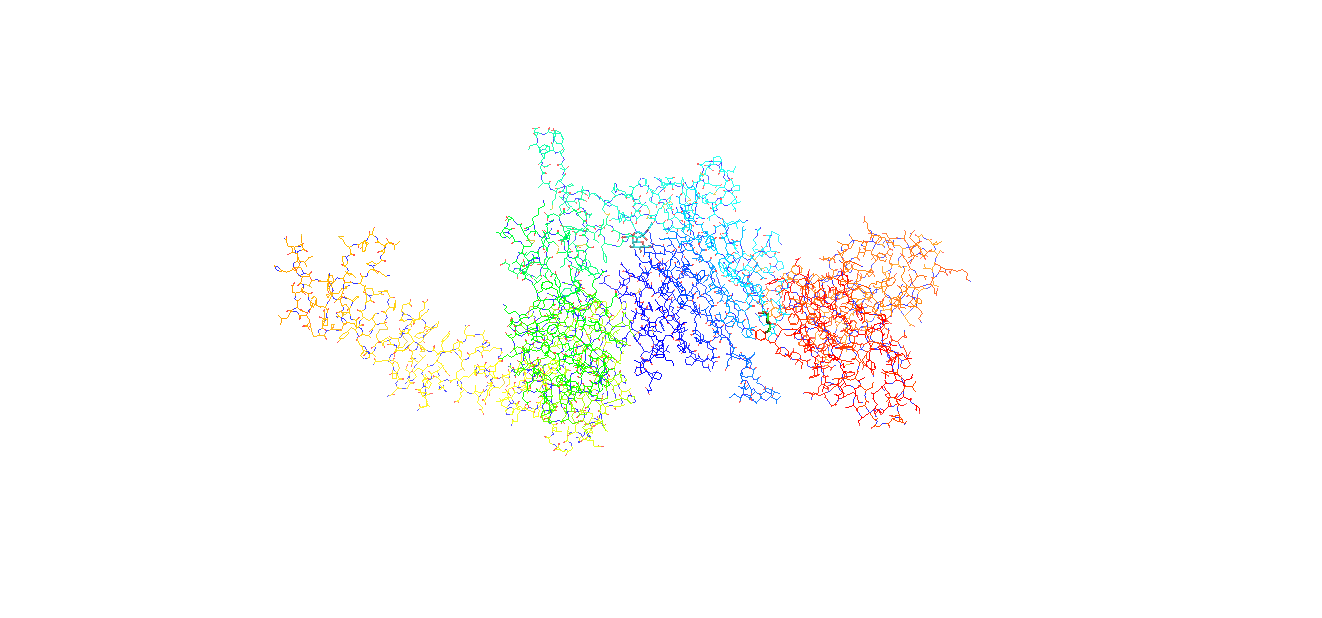
x-axis: Sequence 1

y-axis: Sequence 2

For identical sequences, a solid diagonal line from the top left to the bottom right of the plot is expected. Highly similar sequences are anticipated to produce a prominent diagonal line with minor breaks or gaps representing small differences. In the case of partially similar sequences, several shorter diagonal lines would indicate regions of local similarity. Lastly, sequences with repeats are likely to result in multiple parallel diagonal lines indicating repeated regions within the sequences.

From the above figures, window size 17 and stringency value 9 represents the best dot plot.

Step #47: Visualize your protein using PyMOL. Provide lines, cartoon and spheres representions.

Figure 22: Line representation of protein

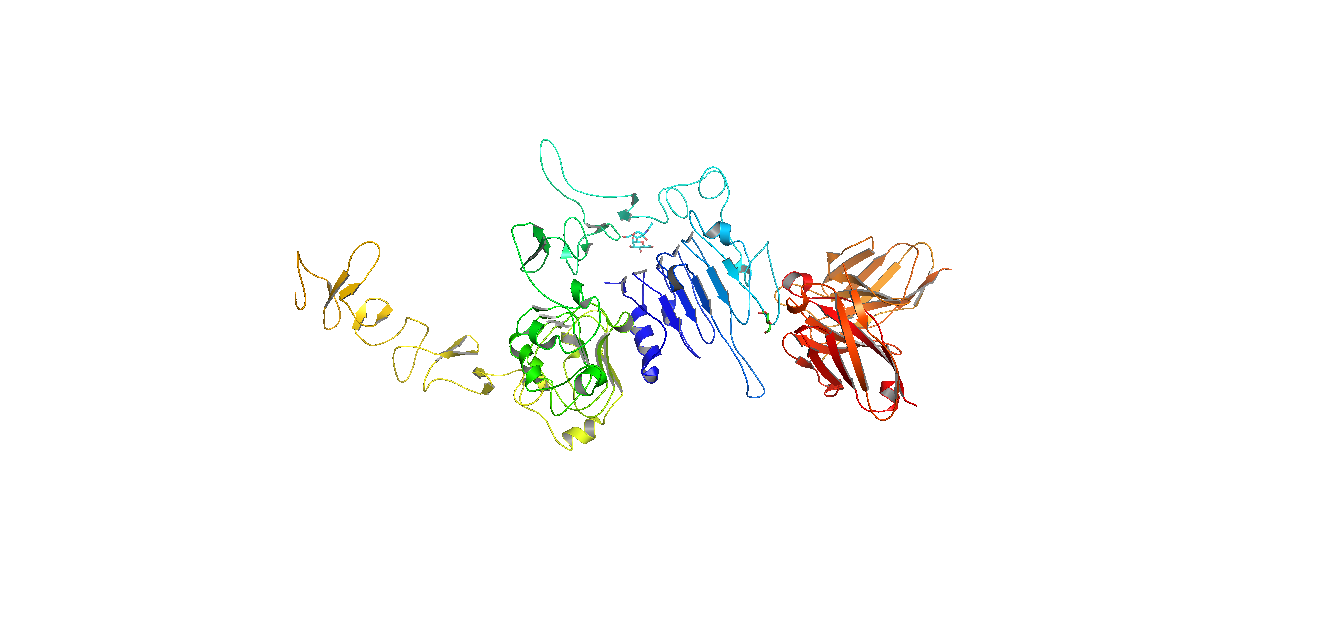
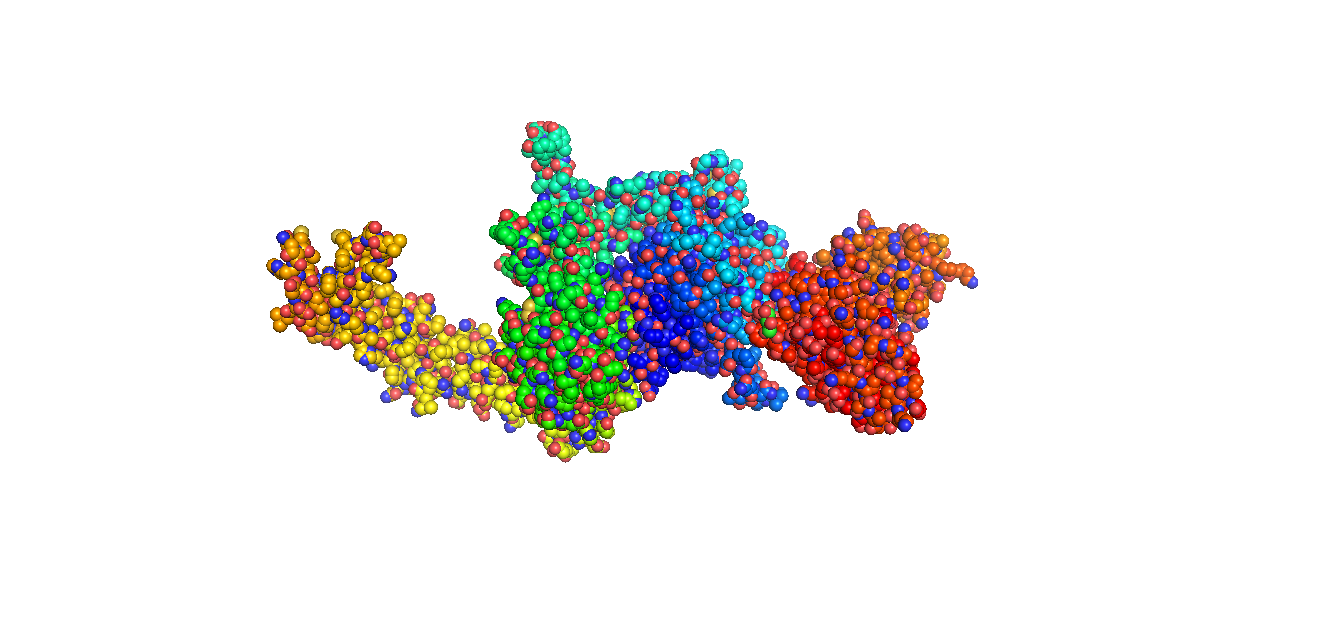
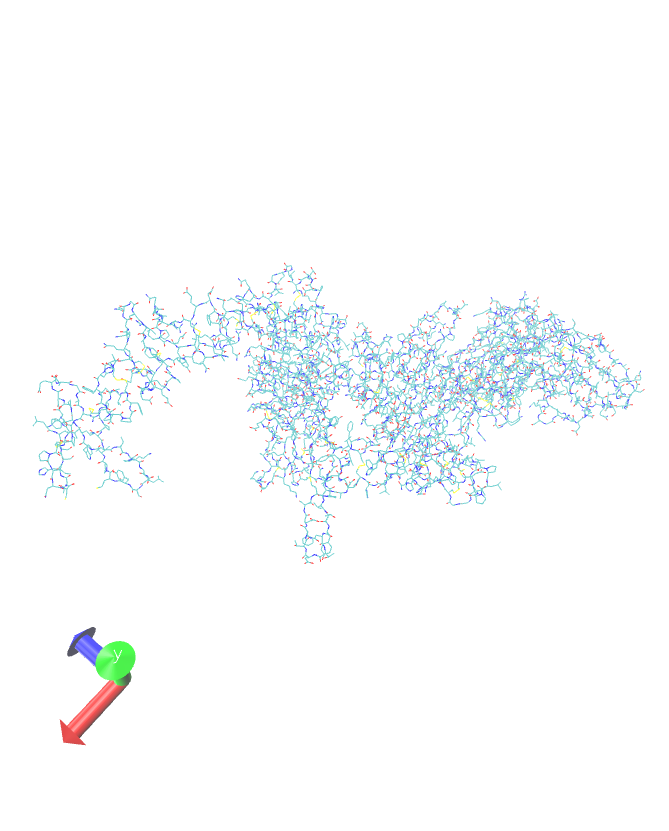
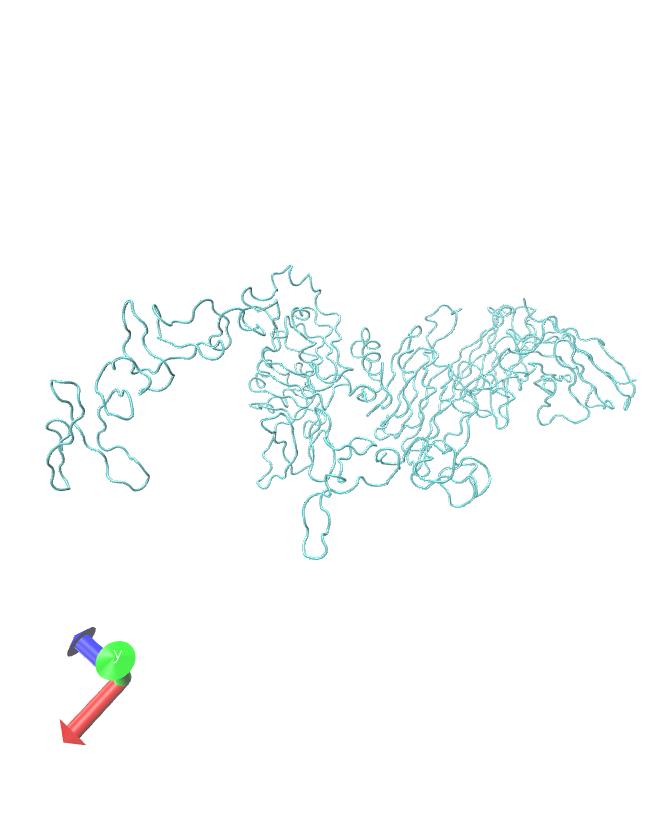


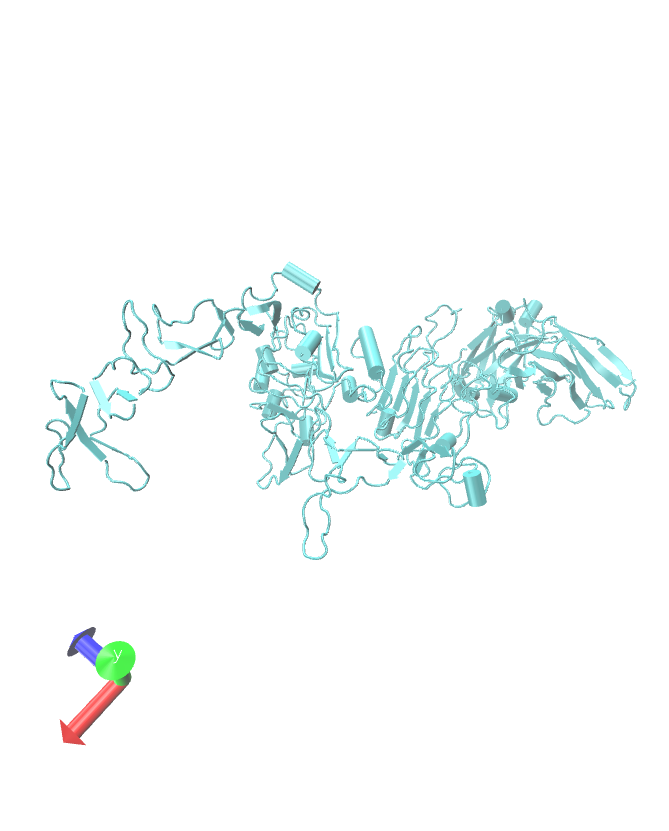
Figure 23: Cartoon representation of protein

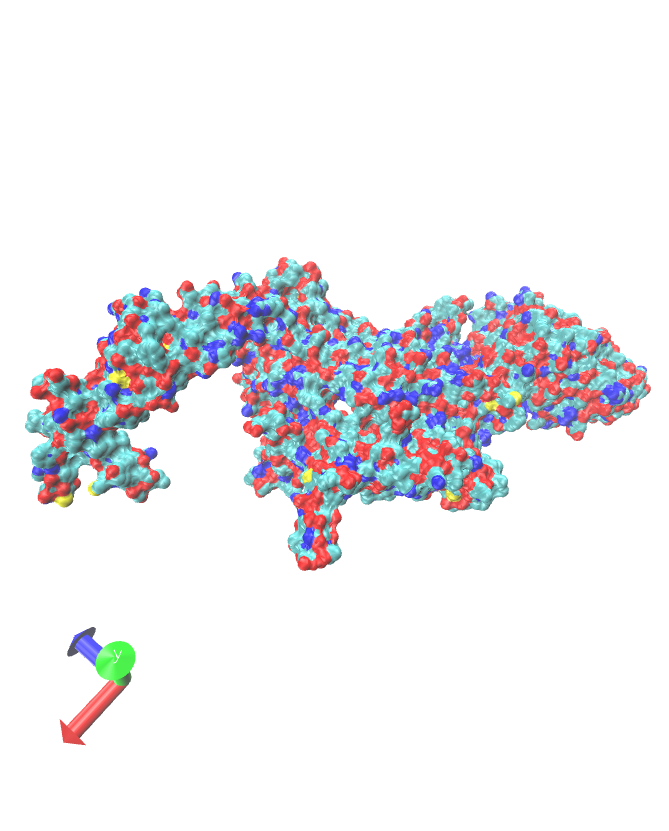
Figure 24: Sphere representation of protein

Step #48: Visualize your protein using VMD. Provide lines, tube, cartoon and surface representions

Figure 25: Line representation in VMD

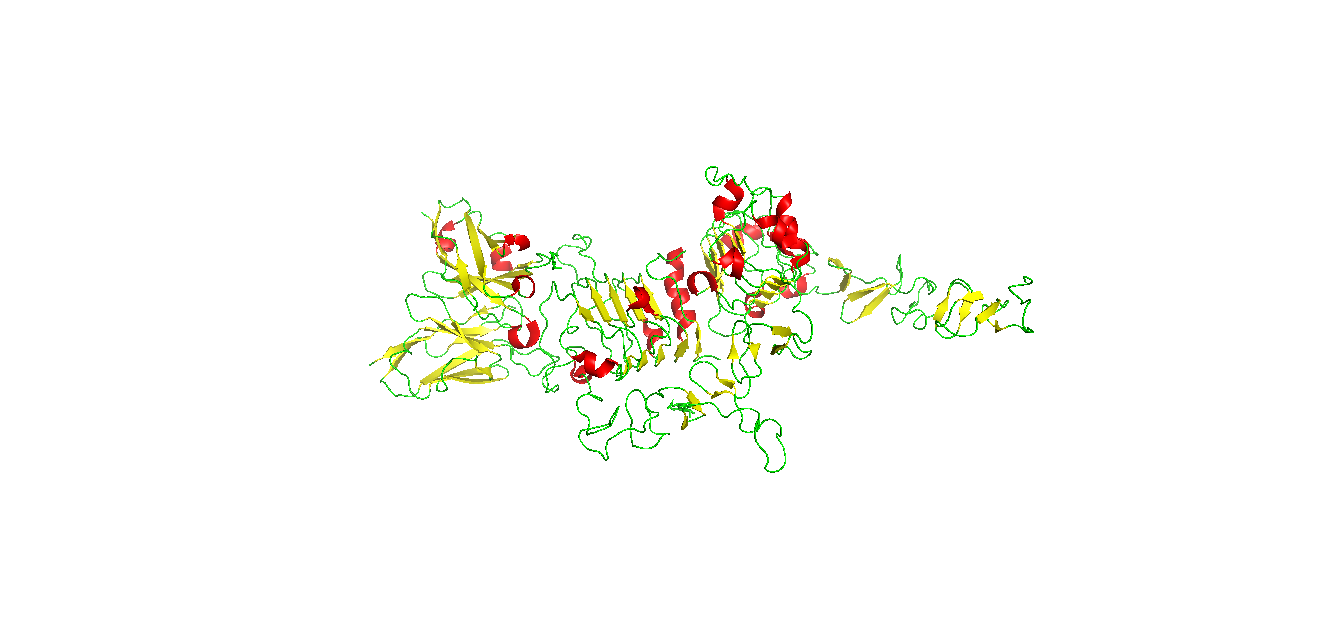
Figure 26: Tube representation in VMD

Figure 27: Cartoon representation in VMD

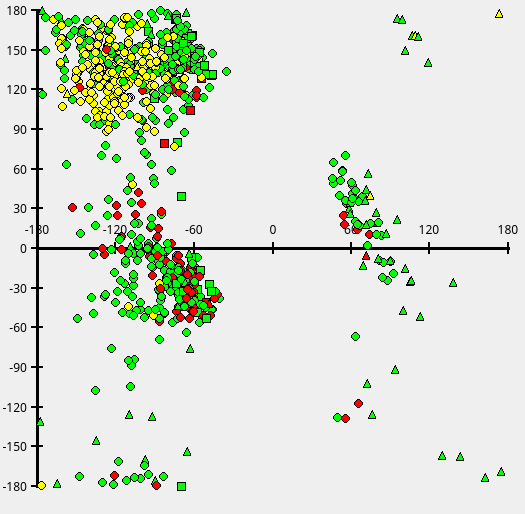
Figure 28: Surface representation in VMD

Step #49: How many secondary structures do you observe in your protein? Which representation type is suitable for observing it?

I observed 71 secondary structures; 18 helices and 53 beta sheets. Cartoon representation is suitable for observing it.

Figure 29: Cartoon representation with secondary structure

Step #50: Display Ramachandran plot using PyMod plugin of PyMOL and VMD tool.



A Ramachandran plot is a visualization tool representing the dihedral angles ψ (psi) and φ (phi) of amino acid residues in a protein structure.

**X-axis (φ):** Represents the phi (φ) dihedral angle, typically ranging from -180° to 180°.

**Y-axis (ψ):** Represents the psi (ψ) dihedral angle, also ranging from -180° to 180°.

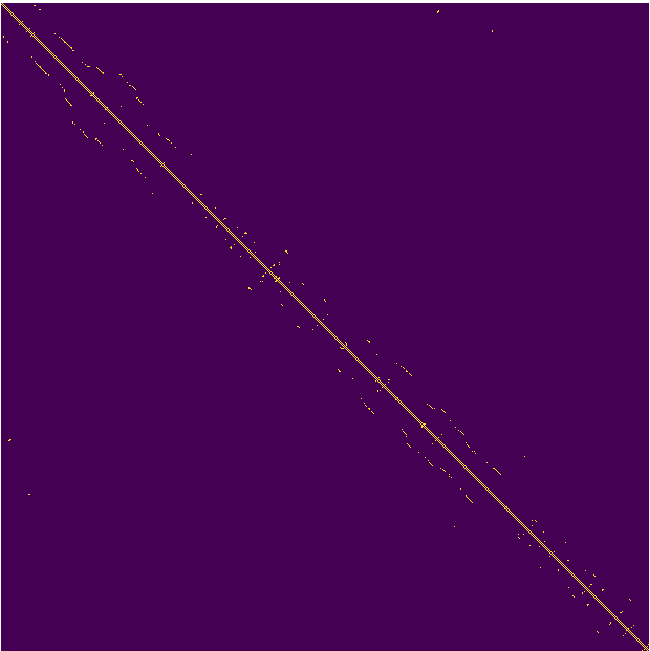
Each point on the plot represents a specific pair of φ and ψ angles for a residue in a protein structure. The colors and shapes often signify different types of amino acids or specific regions, although without a legend, we can only infer general patterns.

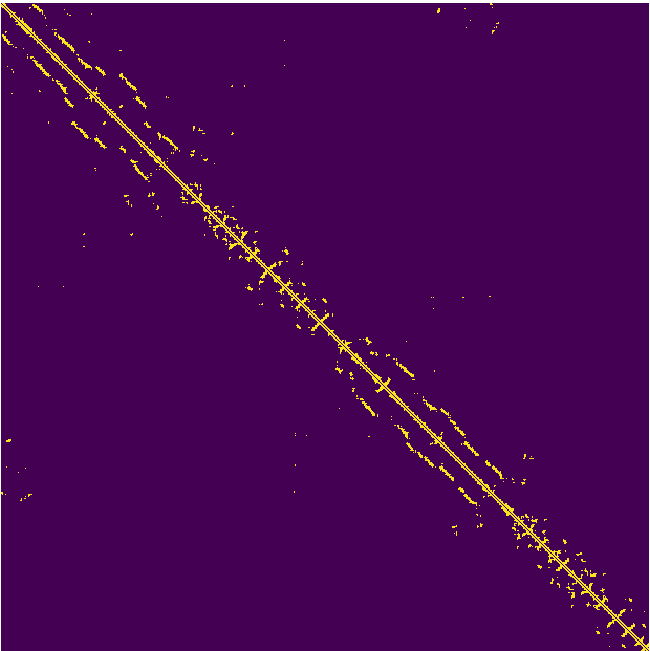
**In the graph:**

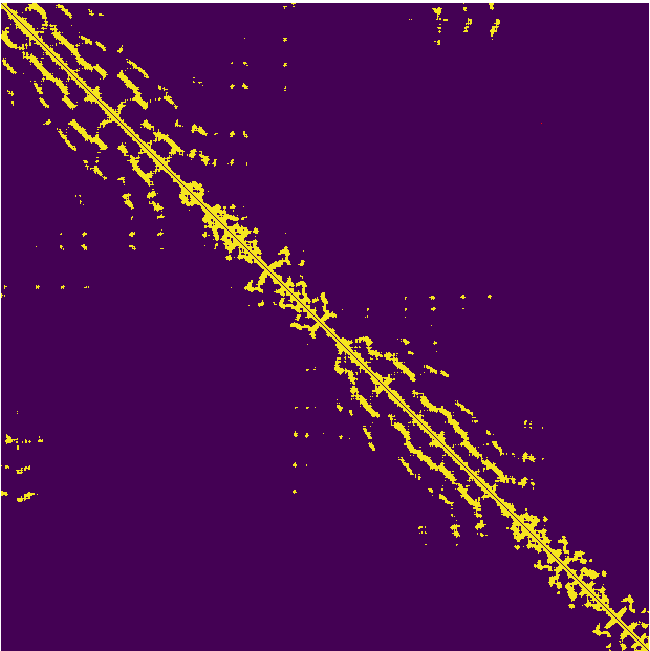
Top left quadrant (α-helix conformation) and Bottom left quadrant (β-sheet and beta turns conformations) are most populated regions.Right side (less populated) indicates less common conformations.

This Ramachandran plot indicates typical protein secondary structure elements, with dense populations in regions corresponding to α-helices and β-sheets. The spread of points in less populated areas suggests the presence of less common or flexible conformations. Overall, the plot provides a visual confirmation of the structural characteristics and conformational preferences of the protein.

**Step #51:** Display Contact Map with three different threshold values (5, 8 and 12 Å). Discuss their differences.

Figure 30: Contact map with 5 Å threshold

Figure 31: Contact map with 8 Å threshold

Figure 32: Contact map with 12 Å threshold

A contact map is a two-dimensional representation of the interactions between amino acids in a protein, indicating whether a pair of residues is in contact within a specified distance threshold.

### Components of a Contact Map:

**Axes:** The x-axis and y-axis represent the sequence of amino acids in the protein, listing the residues in the same order.

**Elements:** Each element (i, j) in the matrix indicates whether the residue at position i is in contact with the residue at position j.

**Threshold:** A distance threshold is used to define a contact, typically ranging from 4 to 8 Å (angstroms).

### In a Contact Map:

* Elements with values (usually 1 or a dot) indicate contacts. In a binary contact map, '1' means contact and '0' means no contact.
* The main diagonal represents self-contacts and is usually ignored.
* **Off-diagonal Elements** represent interactions between different residues. Clusters or lines off the diagonal indicate secondary structure elements like helices or beta sheets.
* Helices appear as thick bands parallel to the diagonal, while beta sheets appear as parallel or anti-parallel lines off the diagonal.
* Long-range contacts (far from the diagonal) provide information about the protein’s three-dimensional fold, indicating interactions between residues far apart in the sequence but close in space.

In the above contact maps, the map with threshold 12 Å shows higher contacts.

**Step #52:** Find the corresponding protein in another species, mostly a bacteria or a virus.

We have chosen a corresponding protein in Avian leukosis virus with accession id P00534.2. We took help of NCBI protein database for this.

**Step #53:** Using SALIGN Structure Alignment in PyMOL (PyMod:Tools:Alignment Tools:Structural Alignment:SALIGN), perform a structural alignment and display their structural divergence.

>6j71\_chain\_A

------------------------------------------------------------

-----------------------------------------------------------S

TQVCTGTDMKLRLPASPETHLDMLRHLYQGCQVVQGNLELTYLPTNASLSFLQDIQEVQG

YVLIAHNQVRQVPLQRLRIVRGTQLFEDNYALAVLDNGDPLNNTTPVTGASPGGLRELQL

RSLTEILKGGVLIQRNPQLCYQDTILWKDIFHKNNQLALTLIDTNRSRACHPCSPMCKGS

RCWGESSEDCQSLTRTVCAGGCARCKGPLPTDCCHEQCAAGCTGPKHSDCLACLHFNHSG

ICELHCPALVTYNTDTFESMPNPEGRYTFGASCVTACPYN----YLSTDVGSC-------

------------------TLVCPLH-------NQEVTAEDGTQRCEKCSKPCARVCYGLG

MEHLR----------------EV-------------------------------------

-------------------------------------RAVTSANIQEFAGCKK-------

-------IFGS-------------------------------------------------

------------------------------------------------LAFLPESFDGDP

ASNTAPLQPEQLQVFETLEEITGYLYISAWPDSLPDLSVFQNLQ----------------

------------------------------------------------------------

-------------------------VIRGRILHNGAYSLTLQGLGISWLGLRSLRELGSG

LALIHHNTHLCFVHTVPWDQLFRNPHQALLHTANRPEDECVGEGLACHQLCARGHCWGPG

PTQCVNCSQ---------------------------------------------------

------------------------------------------------------------

---------------------------------------FLRGQECVEECRVLQGLPREY

VNARHCLPCHPECQPQNGSVTCFGPEADQCVACAHYKDPPFCVARCPSGVKPDLSYMPIW

KFPDEEGACQPCPINCTHSCVDLDD-----------------------------------

-------------------------------------

>model\_001\_processed\_chain\_A

MKCAHFIDGPHCVKACPAGVLGENDTLVWKYADANAVCQLCHPNCTRGCKGPGLEGCPNG

SKTPSIAAGVVGGLLCLVVVGLGIGLYLRRRHIVRKRTLRRLLQERELVEPLTPSGEAP-

--------NQAH------------------------------------------------

---------------------LRIL-----------------------------------

------------------------------------------------------------

------------------------------------------------------------

---------------------------KETEFKKVKVLGSGAFGTVY----KG-------

----LWIPEGEKVKIPVAIKELREATSPKANK---------------------------E

ILDEA----------------YVMASVD--NPHVCRLLGICLTSTVQLITQL-------M

PYGCLLDYIREHKDNIGSQYLLNWCVQIAKGMNYLEERRL-------VHRDLAARNVLVK

TPQHVKITDFGLAKLLGADEKEYHAEGGKVP-----------------------------

----------------------------IKWMALESILHR------IY------------

----------THQSDVWS---------------------YGVTVWELMTFGSKPYDGIPA

SEISSVLEKGERLPQPPICTIDVYMIMVKCWMIDADS-----------------------

------------------RPKFREL--------------------IAEFS----------

----------KMA-------------------------------RDPPRYL---------

------VIQGDERMHLPSPTDSKFYRTLMEEEDMEDIVDADEYLVPHQGFFNSPSTSRTP

LLSSLSATSNNSATNCIDRNGQGHPVREDSFVQRYSSDPTGNFLEESIDDGFLPAPEYVN

QLMPKKPSTAMVQNQIYNNISLTAISKLPMDSRYQNSHS---------------------

------------------------------------------------------------

-------TAVDNP----------EYLNTNQSPLAKTVFESSPYWIQSGNHQINLDNPDYQ

QDFLPNETKPNGLLKVPAAENPEYLRVAAPKSEYIEA

>6j71\_chain\_B

------------------------------------------------------------

------------------------------------------------------------

------------------------------------------------------------

------------------------------------------------------------

------------------------------------------------------------

------------------------------------------------------------

-----------------------------------------ADIVLT----QSPDSLAVS

LGER----------VTINCKSSQPL----EYS---------------------------N

NQWNYLAWYQQKPGQSPKLLISWASTRKSGVPDRFSGSGSG-----TDFTLTISSVQAE-

------------------------------------------------------------

-DVAVYYCGQYSD---------------YPNTFGAGTKLEIKQVQLVQSGAEVVKPGASV

KISCKASGYPFTQYFIHWVKQNPGQRLEWIGQISSSYATVDYNQKF--------------

---------------------------------------------------------KGK

ATLT--------------------------VDTSASIAYMELSSLRSEDTAVYYCVRSGN

YEEYAMDYWGQGTLVTVS------------------------------------------

------------------------------------------------------------

------------------------------------------------------------

------------------------------------------------------------

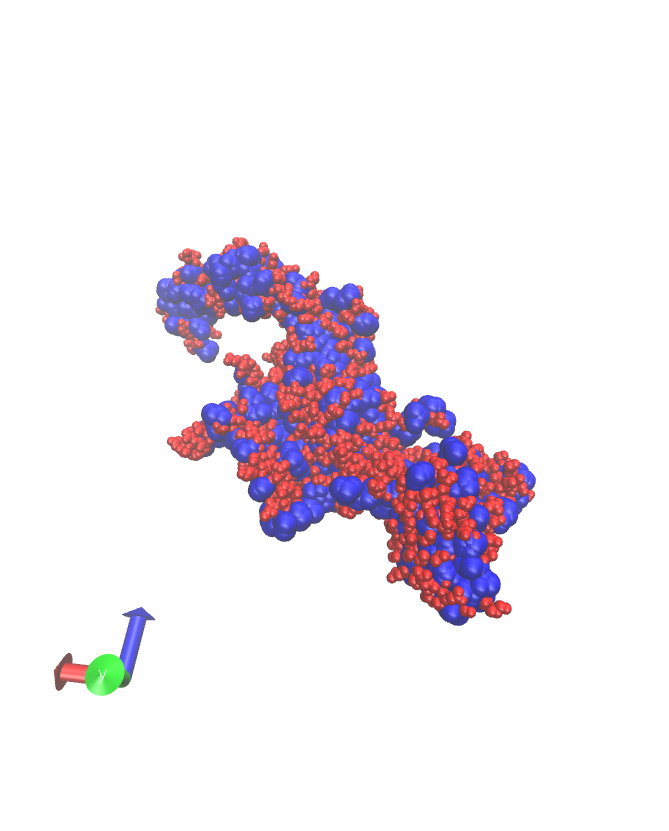
------------------------------------------------------------

------------------------------------------------------------

------------------------------------------------------------

-------------------------------------

Step #54: Using VMD, color your hydrophobic residues in blue and polar residues in red. Then, represent your protein as VDW spheres. Then, double the size of the hydrophobic residues and show how the representation changes afterwards.

Figure 33: Double sized hydrophobic residues in blue and polar in red

Step #55: Print out the residue names of your alpha Carbons.

GLN(362), GLU(363), PHE(364), ALA(365), GLY(366), CYS(367), LYS(368), LYS(369), ILE(370), PHE(371), GLY(372), SER(373), LEU(374), ALA(375), PHE(376), LEU(377), PRO(378), GLU(379), SER(380), PHE(381), ASP(382), GLY(383), ASP(384), PRO(385), ALA(386), SER(387), ASN(388), THR(389), ALA(390), PRO(391), LEU(392), GLN(393), PRO(394), GLU(395), GLN(396), LEU(397), GLN(398), VAL(399), PHE(400), GLU(401), THR(402), LEU(403), GLU(404), GLU(405), ILE(406), THR(407), GLY(408), TYR(409), LEU(410), TYR(411), ILE(412), SER(413), ALA(414), TRP(415), PRO(416), ASP(417), SER(418), LEU(419), PRO(420), ASP(421), LEU(422), SER(423), VAL(424), PHE(425), GLN(426), ASN(427), LEU(428), GLN(429), VAL(430), ILE(431), ARG(432), GLY(433), ARG(434), ILE(435), LEU(436), HIS(437), ASN(438), GLY(439), ALA(440), TYR(441), SER(442), LEU(443), THR(444), LEU(445), GLN(446), GLY(447), LEU(448), GLY(449), ILE(450), SER(451), TRP(452), LEU(453), GLY(454), LEU(455), ARG(456), SER(457), LEU(458), ARG(459), GLU(460), LEU(461), GLY(462), SER(463), GLY(464), LEU(465), ALA(466), LEU(467), ILE(468), HIS(469), HIS(470), ASN(471), THR(472), HIS(473), LEU(474), CYS(475), PHE(476), VAL(477), HIS(478), THR(479), VAL(480), PRO(481), TRP(482), ASP(483), GLN(484), LEU(485), PHE(486), ARG(487), ASN(488), PRO(489), HIS(490), GLN(491), ALA(492), LEU(493), LEU(494), HIS(495), THR(496), ALA(497), ASN(498), ARG(499), PRO(500), GLU(501), ASP(502), GLU(503), CYS(504), VAL(505), GLY(506), GLU(507), GLY(508), LEU(509), ALA(510), CYS(511), HIS(512), GLN(513), LEU(514), CYS(515), ALA(516), ARG(517), GLY(518), HIS(519), CYS(520), TRP(521), GLY(522), PRO(523), GLY(524), PRO(525), THR(526), GLN(527), CYS(528), VAL(529), ASN(530), CYS(531), SER(532), GLN(533), PHE(534), LEU(535), ARG(536), GLY(537), GLN(538), GLU(539), CYS(540), VAL(541), GLU(542), GLU(543), CYS(544), ARG(545), VAL(546), LEU(547), GLN(548), GLY(549), LEU(550), PRO(551), ARG(552), GLU(553), TYR(554), VAL(555), ASN(556), ALA(557), ARG(558), HIS(559), CYS(560), LEU(561), PRO(562), CYS(563), HIS(564), PRO(565), GLU(566), CYS(567), GLN(568), PRO(569), GLN(570), ASN(571), GLY(572), SER(573), VAL(574), THR(575), CYS(576), PHE(577), GLY(578), PRO(579), GLU(580), ALA(581), ASP(582), GLN(583), CYS(584), VAL(585), ALA(586), CYS(587), ALA(588), HIS(589), TYR(590), LYS(591), ASP(592), PRO(593), PRO(594), PHE(595), CYS(596), VAL(597), ALA(598), ARG(599), CYS(600), PRO(601), SER(602), GLY(603), VAL(604), LYS(605), PRO(606), ASP(607), LEU(608), SER(609), TYR(610), MET(611), PRO(612), ILE(613), TRP(614), LYS(615), PHE(616), PRO(617), ASP(618), GLU(619), GLU(620), GLY(621), ALA(622), CYS(623), GLN(624), PRO(625), CYS(626), PRO(627), ILE(628), ASN(629), CYS(630), THR(631), HIS(632), SER(633), CYS(634), VAL(635), ASP(636), LEU(637), ASP(638), ASP(639), ALA(5), ASP(6), ILE(7), VAL(8), LEU(9), THR(10), GLN(11), SER(12), PRO(13), ASP(14), SER(15), LEU(16), ALA(17), VAL(18), SER(19), LEU(20), GLY(21), GLU(22), ARG(23), VAL(24), THR(25), ILE(26), ASN(27), CYS(28), LYS(29), SER(30), SER(31), GLN(32), PRO(33), LEU(34), GLU(35), TYR(36), SER(37), ASN(38), ASN(39), GLN(40), TRP(41), ASN(42), TYR(43), LEU(44), ALA(45), TRP(46), TYR(47), GLN(48), GLN(49), LYS(50), PRO(51), GLY(52), GLN(53), SER(54), PRO(55), LYS(56), LEU(57), LEU(58), ILE(59), SER(60), TRP(61), ALA(62), SER(63), THR(64), ARG(65), LYS(66), SER(67), GLY(68), VAL(69), PRO(70), ASP(71), ARG(72), PHE(73), SER(74), GLY(75), SER(76), GLY(77), SER(78), GLY(79), THR(80), ASP(81), PHE(82), THR(83), LEU(84), THR(85), ILE(86), SER(87), SER(88), VAL(89), GLN(90), ALA(91), GLU(92), ASP(93), VAL(94), ALA(95), VAL(96), TYR(97), TYR(98), CYS(99), GLY(100), GLN(101), TYR(102), SER(103), ASP(104), TYR(105), PRO(106), ASN(107), THR(108), PHE(109), GLY(110), ALA(111), GLY(112), THR(113), LYS(114), LEU(115), GLU(116), ILE(117), LYS(118), GLN(140), VAL(141), GLN(142), LEU(143), VAL(144), GLN(145), SER(146), GLY(147), ALA(148), GLU(149), VAL(150), VAL(151), LYS(152), PRO(153), GLY(154), ALA(155), SER(156), VAL(157), LYS(158), ILE(159), SER(160), CYS(161), LYS(162), ALA(163), SER(164), GLY(165), TYR(166), PRO(167), PHE(168), THR(169), GLN(170), TYR(171), PHE(172), ILE(173), HIS(174), TRP(175), VAL(176), LYS(177), GLN(178), ASN(179), PRO(180), GLY(181), GLN(182), ARG(183), LEU(184), GLU(185), TRP(186), ILE(187), GLY(188), GLN(189), ILE(190), SER(191), SER(192), SER(193), TYR(194), ALA(195), THR(196), VAL(197), ASP(198), TYR(199), ASN(200), GLN(201), LYS(202), PHE(203), LYS(204), GLY(205), LYS(206), ALA(207), THR(208), LEU(209), THR(210), VAL(211), ASP(212), THR(213), SER(214), ALA(215), SER(216), ILE(217), ALA(218), TYR(219), MET(220), GLU(221), LEU(222), SER(223), SER(224), LEU(225), ARG(226), SER(227), GLU(228), ASP(229), THR(230), ALA(231), VAL(232), TYR(233), TYR(234), CYS(235), VAL(236), ARG(237), SER(238), GLY(239), ASN(240), TYR(241), GLU(242), GLU(243), TYR(244), ALA(245), MET(246), ASP(247), TYR(248), TRP(249), GLY(250), GLN(251), GLY(252), THR(253), LEU(254), VAL(255), THR(256), VAL(257), SER(258)

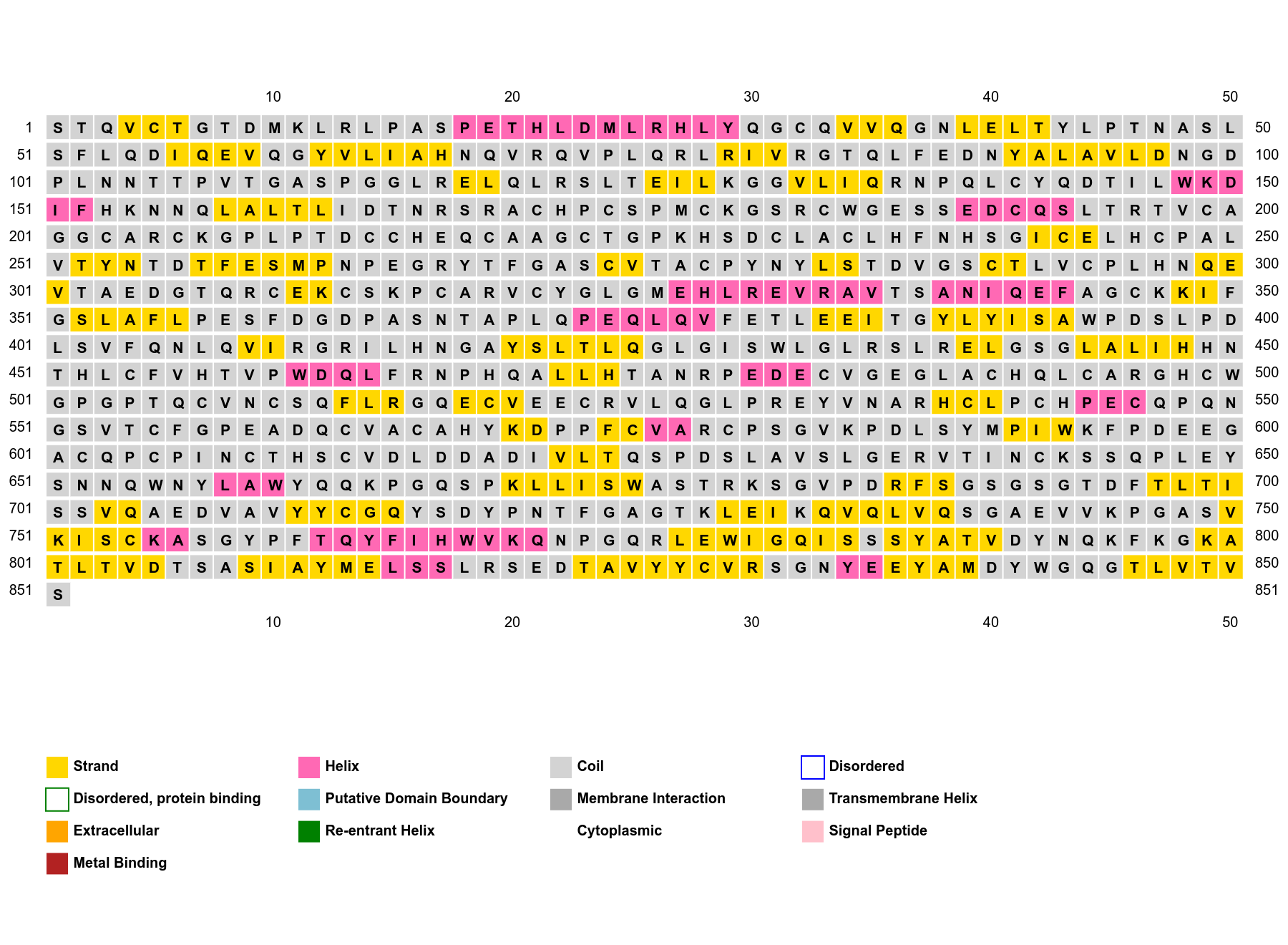
Step #56: Measure the size of your protein using VMD. See lecture slides for more details.

Minimum coordinates in the protein: (569.078, 117.845, 348.791)

Maximum coordinates in the protein: (702.971, 172.165, 468.667)

Coordinates are measured in 3D (x,y,z) direction.

Step #57: Predict the secondary structure using PSIPRED server



The predicted secondary structure using PSIPRED server is given below:

CCCEEECCCCCCCCCCCHHHHHHHHHHHHCCCCEEECCEEEECCCCCCCCCCCCCEEEECCEEEEEECCCCCCCCCCCEEECCCCCCCCCEEEEEEECCCCCCCCCCCCCCCCCCCCEECCCCCCEEECCCEEEECCCCCCCCCCCCHHHHHCCCCCEEEEECCCCCCCCCCCCCCCCCCCCCCCCCCHHHHHCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCEEECCCCCCCEEECCEEEEEECCCCCCCCCCCEECCCCCCCEECCCCCEECCCCCCCEEECCCCCCCCCEECCCCCCCCCCCCCCHHHHHHHHHCCHHHHHHCCCCEECCEEEEECCCCCCCCCCCCCCCCHHHHHHCCCCEEECCEEEEEECCCCCCCCCCCCCCCEECCCCCCCCCEEEEEECCCCCCCCCCCCCEECCCEEEEECCCCCCCCCCCCHHHHCCCCCCCEEECCCCCHHHCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCEEECCEEECCCCCCCCCCCCCCCCCEEECCCHHHCCCCCCCCCCCCCCCCCCCCCCCEECCEEHHCCCCCCCCCCCCCEEECCCCCCCCCCCCCCCCCCCCCCCCCCCCEEECCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCHHHCCCCCCCCCEEEEEECCCCCCCCCCEEECCCCCCCCEEEECCEECCCCCCEEEEECCCCCCCCCCCCCEEECEEEEEECCCCCCCCCCCEEEEEHHCCCCCHHHHHHHHHHCCCCCEEEEEEEECEEEEECCCCCCCCEEEEEEECCCEEEEEEHHHCCCCCEEEEEEEECCCHHEEEECCCCCCEEEEEC

Step #58: Calculate the observed secondary structure using ss.tcl.

|  |  |  |
| --- | --- | --- |
|  | PSIPRED | ss.tcl |
| Number of Helices | 75 | 69 |
| Number of Betasheets | 197 | 281 |
| Number of Coils | 579 | 501 |

Step #59: What is the accuracy of the prediction tool? (Hint: # of predicted equal to observed / total # of residues). Make use of slides 18 to 36 in Lecture “May31st\_Prediction\_PartI”.

Number of observed helices: 69

Number of predicted helices: 75

Number of observed beta sheets: 281

Number of predicted beta sheets: 197

Number of observed coils: 501

Number of predicted coils: 579

Total correctly predicted = 69 + 197 + 501 = 767

Total Number of residues = 851

So, accuracy = 767 / 851 = 90.13%

Step #60: Extract the sequence of the same protein from a different species for which there is no tertiary structure in Protein Data Bank.

Extracted sequence is given below with accession id P00534.2:

>sp|P00534.2|ERBB\_ALV RecName: Full=Tyrosine-protein kinase transforming protein erbB

MKCAHFIDGPHCVKACPAGVLGENDTLVWKYADANAVCQLCHPNCTRGCKGPGLEGCPNGSKTPSIAAGV

VGGLLCLVVVGLGIGLYLRRRHIVRKRTLRRLLQERELVEPLTPSGEAPNQAHLRILKETEFKKVKVLGS

GAFGTVYKGLWIPEGEKVKIPVAIKELREATSPKANKEILDEAYVMASVDNPHVCRLLGICLTSTVQLIT

QLMPYGCLLDYIREHKDNIGSQYLLNWCVQIAKGMNYLEERRLVHRDLAARNVLVKTPQHVKITDFGLAK

LLGADEKEYHAEGGKVPIKWMALESILHRIYTHQSDVWSYGVTVWELMTFGSKPYDGIPASEISSVLEKG

ERLPQPPICTIDVYMIMVKCWMIDADSRPKFRELIAEFSKMARDPPRYLVIQGDERMHLPSPTDSKFYRT

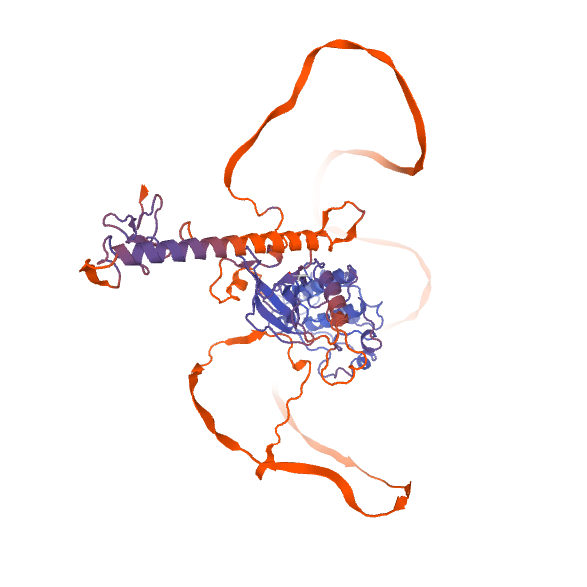
LMEEEDMEDIVDADEYLVPHQGFFNSPSTSRTPLLSSLSATSNNSATNCIDRNGQGHPVREDSFVQRYSS

DPTGNFLEESIDDGFLPAPEYVNQLMPKKPSTAMVQNQIYNNISLTAISKLPMDSRYQNSHSTAVDNPEY

LNTNQSPLAKTVFESSPYWIQSGNHQINLDNPDYQQDFLPNETKPNGLLKVPAAENPEYLRVAAPKSEYI

EASA

Step #61: Perform a homology modeling to predict the structure of the protein in Step #59.

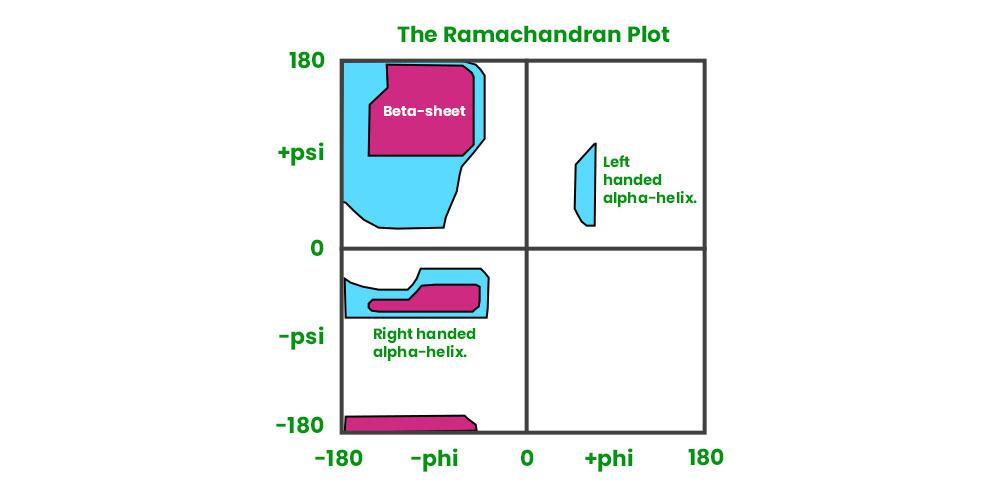
Figure 34: Predicted protein structure by homology modelling

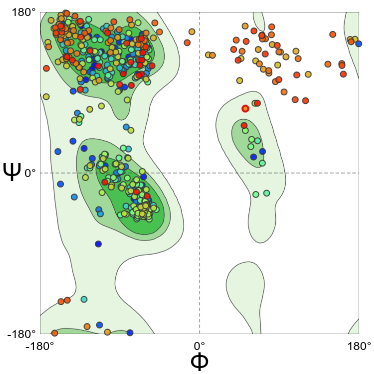
MolProbity Results:

MolProbity score: 1.87

Clash score: 0.30

Ramachandran Favoured: 82.22%

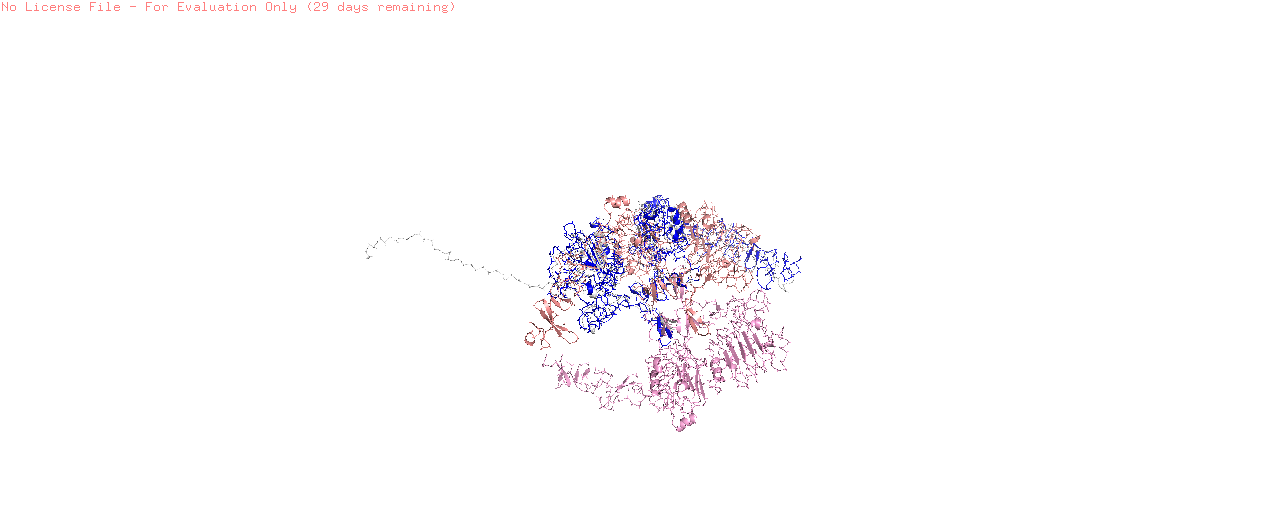


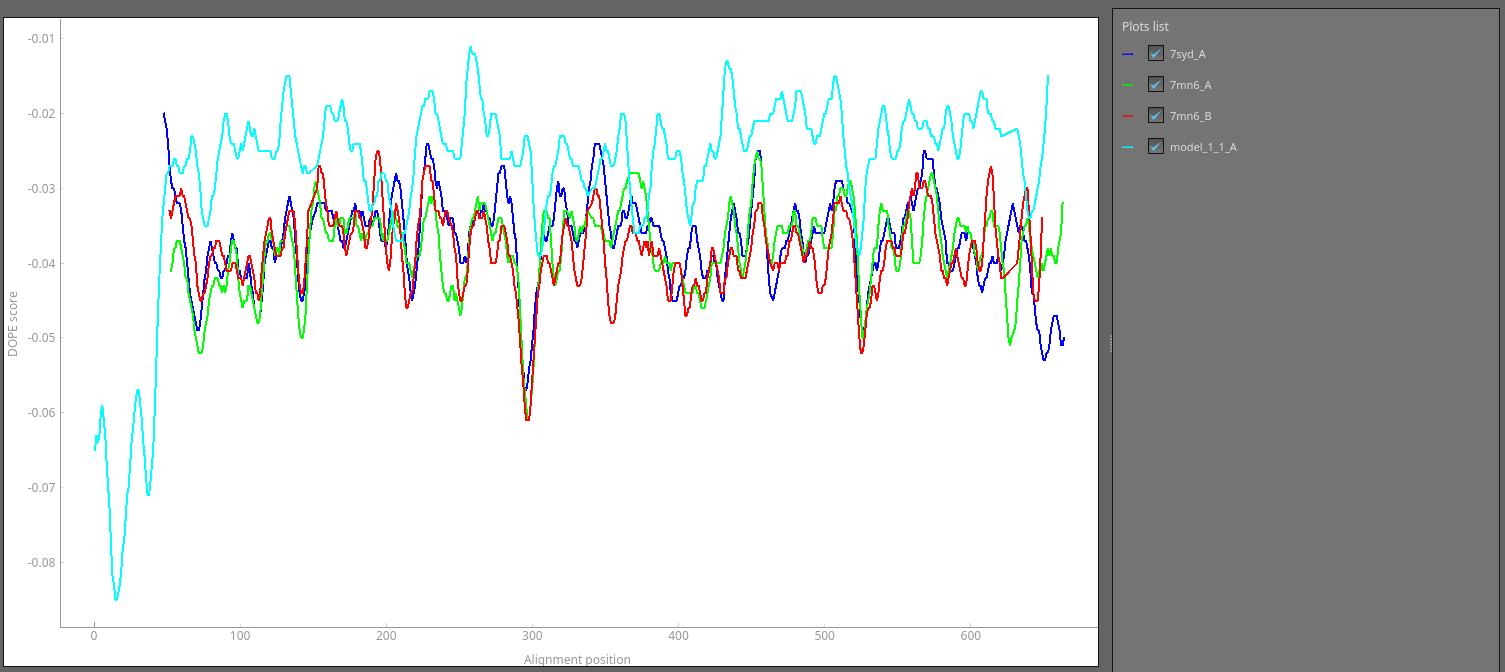


In the graph, we can see most residues are in ramachandran favoured regions. Top-left region are abundant with beta sheets residues. It shows the abundance in helical region too.

Step #62: Using PRISM, predict the complex structure between your protein and the related protein you used in Part III.

Step #63: If the protein structure of the related structure is not resolved, predict the structure using MODELLER in PyMOL.

Figure 35: Predicted structure using Pymol MODELLER

Figure 36: DOPE graph for protein model assessment

The Discrete Optimized Protein Energy (DOPE) is a statistical potential used in protein structure prediction to assess the quality of protein models. The DOPE score indicates the likelihood of a protein's structure being accurate based on energy calculations. A DOPE figure plots the DOPE score for each residue in the protein model.

Lower DOPE scores indicate more favorable (likely accurate) residue environments.

Higher DOPE scores suggest less favorable (likely inaccurate) environments.

**X-axis:** Represents the residue number or sequence position.

**Y-axis:** Represents the DOPE score for each residue.

### In the DOPE Figure**:**

* Looking for residues with significantly higher DOPE scores, indicates problematic regions in the model such as incorrect folds, misaligned residues, or regions with poor stereochemistry.
* A smooth, consistently low curve indicates a well-optimized model. Sharp peaks suggest areas that might need refinement.
* Loop regions often have higher scores due to their flexibility, but extreme outliers should be inspected.
* While assessing the overall distribution of scores, if a model with many residues having low scores is generally of higher quality.

The figure shows four lines:

Blue line: 7syd model with chain A

green line: 7mn6 with chain A

red line: 7mn6 with chain B

Cyan line: model\_1\_1 with chain A

model\_1\_1\_A shows high fluctuations while others maintain a fair consistency.