In [1]:
```
%pip install biopython
```

```
Requirement already satisfied: biopython in ./.local/share/pipx/venvs/note
book/lib/python3.12/site-packages (1.86)
Requirement already satisfied: numpy in ./.local/share/pipx/venvs/noteboo
k/lib/python3.12/site-packages (from biopython) (2.3.5)

[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: /home/naushin_parveen/.local/share/pipx/venvs/not
ebook/bin/python -m pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```
from Bio import Entrez, SeqIO
print("Biopython import OK")
```

```
Biopython import OK
```

In [2]:
```
from Bio import Entrez, SeqIO
Entrez.email = "naushin.mansuri@iitgn.ac.in"
query = "globin[Protein Name] NOT partial[Title] NOT fragment[Title]"
handle = Entrez.esearch(db="protein", term=query, retmax=200)
record = Entrez.read(handle)
handle.close()
print("IDs Found:", len(record["IdList"]))
ids = record["IdList"]
handle = Entrez.efetch(db="protein", id=",".join(ids), rettype="fasta", r
sequence_data = handle.read()
handle.close()
print("\nSample Output (first 400 characters):\n")
print(sequence_data[:400])
with open("globin_sequences_raw.fasta", "w") as file:
    file.write(sequence_data)
print("\nSaved all sequences to 'globin_sequences_raw.fasta'")
```

```
IDs Found: 200

Sample Output (first 400 characters):

>WP_447037499.1 globin [Streptomyces sp. DSM 118878]
MDSVKEIPHGTVQEQTYYEQVGGEETFRRLVHLFYQGVAEDPLLRPMYPEGDLGPAEERFALFLMQYWGG
PRTYSDNRGHPRLRMRHAPFTVDRAAHDAWLKHMRAAVDQLGLSEEHERTLWNYLTYAAASMVNSEG

>WP_447029335.1 globin [Streptomyces hypolithicus]
MNEIPIGTLQEQTFYEQVGGEETFRRLVHRFYQGVAEDPLLKPMYPEEDLGPAEERLALFLMQYWGGPRT
YSDERGHPRLRMRHAPFTVDKAAHDAWLQHMRVAVDELGLSEDHERQLWNYLTYAAASMVNKTG

>WP_447006119.1 glo

Saved all sequences to 'globin_sequences_raw.fasta'
```

In [4]:
```
from Bio import SeqIO
print("Total sequences in file:", sum(1 for _ in SeqIO.parse("globin_sequ
rec = next(SeqIO.parse("globin_sequences_raw.fasta","fasta"))
print("First ID:", rec.id, "Length:", len(rec.seq))
```

```
Total sequences in file: 200
First ID: WP_447037499.1 Length: 137
```

In [4]:
```
from Bio import SeqIO
lengths = [len(r.seq) for r in SeqIO.parse("globin_sequences_raw.fasta","
print("Total sequences:", len(lengths))
print("Shortest:", min(lengths))
print("Longest:", max(lengths))
print("Example lengths:", lengths[:10])
```

```
Total sequences: 200
Shortest: 53
Longest: 166
Example lengths: [137, 134, 128, 154, 133, 139, 139, 139, 139, 130]
```

In [5]:
```
from Bio import SeqIO

input_file = "globin_sequences_raw.fasta"
output_file = "globin_sequences_filtered.fasta"

filtered_sequences = []

for record in SeqIO.parse(input_file, "fasta"):
    if len(record.seq) >= 100:
        filtered_sequences.append(record)

SeqIO.write(filtered_sequences, output_file, "fasta")

print("Total sequences after filtering:", len(filtered_sequences))
print("Saved filtered sequences to 'globin_sequences_filtered.fasta'")
```

```
Total sequences after filtering: 196
Saved filtered sequences to 'globin_sequences_filtered.fasta'
```

In [6]:
```
mv -f globin_sequences_filtered.fasta final_globin_sequences.fasta
```

In [7]:
```
from Bio import SeqIO
count = sum(1 for _ in SeqIO.parse("final_globin_sequences.fasta","fasta"
print("Total sequences in final file:", count)
```

```
Total sequences in final file: 196
```

In [8]:
```
%pip install matplotlib
```

```
Requirement already satisfied: matplotlib in ./.local/share/pipx/venvs/not
ebook/lib/python3.12/site-packages (3.10.7)
Requirement already satisfied: contourpy>=1.0.1 in ./.local/share/pipx/ven
vs/notebook/lib/python3.12/site-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in ./.local/share/pipx/venvs/n
otebook/lib/python3.12/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in ./.local/share/pipx/ve
nvs/notebook/lib/python3.12/site-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in ./.local/share/pipx/ve
nvs/notebook/lib/python3.12/site-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in ./.local/share/pipx/venvs/no
tebook/lib/python3.12/site-packages (from matplotlib) (2.3.5)
Requirement already satisfied: packaging>=20.0 in ./.local/share/pipx/venv
s/notebook/lib/python3.12/site-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in ./.local/share/pipx/venvs/note
book/lib/python3.12/site-packages (from matplotlib) (12.0.0)
Requirement already satisfied: pyparsing>=3 in ./.local/share/pipx/venvs/n
otebook/lib/python3.12/site-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in ./.local/share/pip
x/venvs/notebook/lib/python3.12/site-packages (from matplotlib) (2.9.0.pos
t0)
Requirement already satisfied: six>=1.5 in ./.local/share/pipx/venvs/noteb
ook/lib/python3.12/site-packages (from python-dateutil>=2.7->matplotlib)
(1.17.0)

[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: /home/naushin_parveen/.local/share/pipx/venvs/not
ebook/bin/python -m pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

In [10]:
```python
from Bio import SeqIO
import matplotlib.pyplot as plt

input_file = "final_globin_sequences.fasta"
```

In [11]:
```python
##Collect IDs and lengths
lengths = []
ids = []

for record in SeqIO.parse(input_file, "fasta"):
    ids.append(record.id)
    lengths.append(len(record.seq))

#Save ID + length
with open("sequence_id_length.txt", "w") as f:
    for seq_id, length in zip(ids, lengths):
        f.write(f"{seq_id}\t{length}\n")
```

In [12]:
```python
import os

print("File exists:", os.path.exists("sequence_id_length.txt"))
```

```
File exists: True
```

In [13]:
```python
with open("sequence_id_length.txt") as f:
    for i in range(10):
        print(f.readline().strip())
```

```
WP_447037499.1  137
WP_447029335.1  134
WP_447006119.1  128
WP_446888964.1  154
YBV26126.1      133
YBV21917.1      139
YBV18343.1      139
YBV09626.1      139
YBV14740.1      139
BGO70610.1      130
```

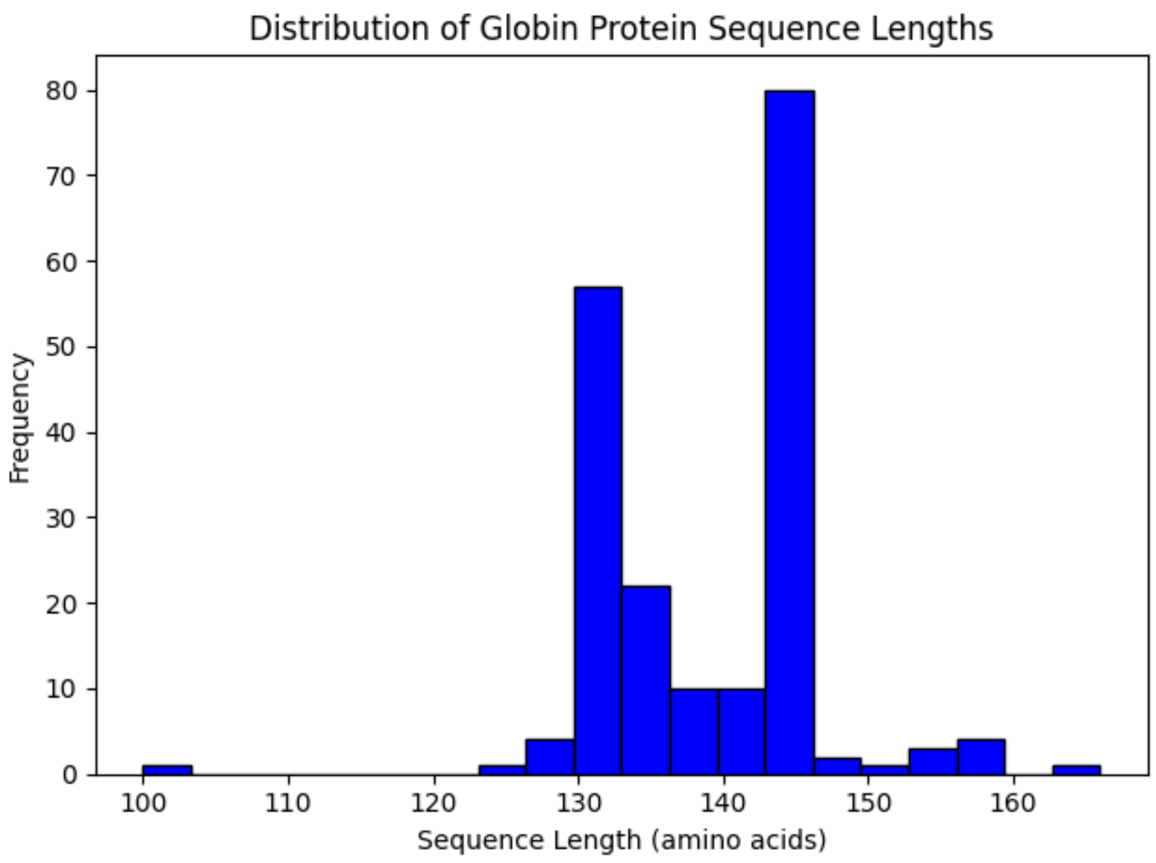
In [14]:
```
##summary
print("Total sequences:", len(lengths))
print("Minimum length:", min(lengths))
print("Maximum length:", max(lengths))
print("First 10 lengths (sorted):", sorted(lengths)[:10])
```

```
Total sequences: 196
Minimum length: 100
Maximum length: 166
First 10 lengths (sorted): [100, 126, 128, 129, 129, 129, 130, 130, 130, 1
30]
```

In [16]:
```
plt.savefig("sequence_length_histogram.png", dpi=300)
```

```
<Figure size 640x480 with 0 Axes>
```

In [25]:
```
##Plot histogram
plt.hist(lengths, bins=20, edgecolor='black', color='blue')
plt.title("Distribution of Globin Protein Sequence Lengths")
plt.xlabel("Sequence Length (amino acids)")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
outpng = "length_distribution.png"
fig.savefig(outpng, dpi=300)
print("Saved histogram to", outpng)
```



```
Saved histogram to length_distribution.png
```

In [26]:
```python
import os
print(os.path.exists("length_distribution.png"))
```

```
True
```

In [30]:
```python
for org in organisms:
    for r in records:
        if org.lower() in r.description.lower():
            outfile = org.replace(" ", "_") + ".fasta"
            SeqIO.write(r, outfile, "fasta")
            print("Saved:", outfile)
            break
```

```
Saved: Bacillus.fasta
```

In [37]:
```python
from Bio import SeqIO

records = list(SeqIO.parse("final_globin_sequences.fasta", "fasta"))

organisms = [
    "Streptomyces hypolithicus",
    "Saccharothrix isguenensis",
    "Pseudoalteromonas sp. SaAl2",
    "Sphingomonas sp. CJ20",
    "Leptospira interrogans",
    "Gordonia sp. J1A",
    "Azospira sp. I13",
    "Sphingopyxis sp.",
    "Streptomyces spiroverticillatus",
    "Pseudomonas aeruginosa"
]

counter = 1   # start numbering files
```

```
for org in organisms:
    for r in records:
        if org.lower() in r.description.lower():
            outfile = f"{counter}_" + org.replace(" ", "_") + ".fasta"
            SeqIO.write(r, outfile, "fasta")
            print("Saved:", outfile)
            counter += 1   # increase number after saving
            break
```

```
Saved: 1_Streptomyces_hypolithicus.fasta
Saved: 2_Saccharothrix_isguenensis.fasta
Saved: 3_Pseudoalteromonas_sp._SaAl2.fasta
Saved: 4_Sphingomonas_sp._CJ20.fasta
Saved: 5_Leptospira_interrogans.fasta
Saved: 6_Gordonia_sp._J1A.fasta
Saved: 7_Azospira_sp._I13.fasta
Saved: 8_Sphingopyxis_sp..fasta
Saved: 9_Streptomyces_spiroverticillatus.fasta
Saved: 10_Pseudomonas_aeruginosa.fasta
```

In [2]:

```
# Cell 1: checks and paths
import shutil, subprocess, sys
from pathlib import Path

# EDIT if your fasta has a different name or location
fasta_in = "combined_sequences.fasta"     # <-- your 10-sequence FASTA
align_out = "sequences_aligned.fasta"
clustal_out = "sequences_aligned.clustal"
tree_out = "sequences_tree.dnd"

print("Current working directory:", Path.cwd())
print("Project PDF (uploaded): /mnt/data/Project_assignment2.pdf\n")

# check fasta exists
if not Path(fasta_in).exists():
    print(f"ERROR: Input FASTA not found at: {fasta_in}")
    raise SystemExit("Place your combined FASTA in the notebook folder or

# check clustalo on PATH
clustalo_path = shutil.which("clustalo")
if clustalo_path:
    print("clustalo found at:", clustalo_path)
    try:
        out = subprocess.run(["clustalo", "--version"], capture_output=Tr
        ver = out.stdout.strip() or out.stderr.strip()
        print("clustalo version info:", ver)
    except subprocess.CalledProcessError:
        print("clustalo exists but version query failed; may still run.")
else:
    print("clustalo NOT found in PATH.")
    print("Install and start Jupyter from the same conda env, for example
    print("  conda activate MSA")
    print("  conda install -c bioconda clustalo")
    print("  jupyter notebook")
    raise SystemExit("Install clustalo in the env you run Jupyter from, t
Current working directory: /home/naushin_parveen
Project PDF (uploaded): /mnt/data/Project_assignment2.pdf

clustalo found at: /home/naushin_parveen/miniconda3/bin/clustalo
clustalo version info: 1.2.3
```

In [3]:
```python
# Cell 2: run clustalo to produce aligned FASTA and guide-tree
import subprocess
from pathlib import Path

cmd = [
    "clustalo",
    "-i", fasta_in,
    "-o", align_out,
    "--guidetree-out", tree_out,
    "--outfmt=fasta",
    "--force"
]

print("Running Clustal Omega:")
print(" ".join(cmd))
try:
    proc = subprocess.run(cmd, check=True, capture_output=True, text=True
    print("Clustal Omega finished (returncode=0).")
    if proc.stdout:
        print("STDOUT snippet:", proc.stdout[:600])
    if proc.stderr:
        # some versions print version/info to stderr
        print("STDERR snippet:", proc.stderr[:600])
    # sanity: confirm files created
    print("\nOutput files created:")
    for p in (align_out, tree_out):
        print(" -", Path(p).resolve(), "(exists)" if Path(p).exists() els
except FileNotFoundError:
    print("ERROR: clustalo executable not found. Please install and resta
    raise
except subprocess.CalledProcessError as e:
    print("clustalo failed. Return code:", e.returncode)
    print("stdout (head):", e.stdout[:1000])
    print("stderr (head):", e.stderr[:1000])
    raise
```

```
Running Clustal Omega:
clustalo -i combined_sequences.fasta -o sequences_aligned.fasta --guidetre
e-out sequences_tree.dnd --outfmt=fasta --force
Clustal Omega finished (returncode=0).

Output files created:
 - /home/naushin_parveen/sequences_aligned.fasta (exists)
 - /home/naushin_parveen/sequences_tree.dnd (exists)
```

In [4]:
```python
# Cell 3: read alignment, print summary, write Clustal format, compute co
from Bio import AlignIO
from Bio.Align import AlignInfo
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
import numpy as np
from pathlib import Path

# Read alignment
```

```
alignment = AlignIO.read(align_out, "fasta")
nseq = len(alignment)
L = alignment.get_alignment_length()
print(f"Number of sequences in alignment: {nseq}")
print(f"Alignment length (columns): {L}\n")

# Preview first two sequences
for rec in alignment[:2]:
    print(f">{rec.id}\n{str(rec.seq)[:200]}...\n")

# Write Clustal format copy (useful for viewers)
AlignIO.write(alignment, clustal_out, "clustal")
print(f"Wrote Clustal-format alignment to: {Path(clustal_out).resolve()}"

# Consensus (majority rule) using Bio.Align.AlignInfo.SummaryInfo
summary = AlignInfo.SummaryInfo(alignment)
consensus = summary.dumb_consensus(threshold=0.5, ambiguous='X')  # thres
print("Consensus (dumb_consensus, 50% threshold):")
print(str(consensus)[:200] + ("..." if len(consensus) > 200 else ""))

# Pairwise percent identity matrix
def pairwise_pid(rec_a, rec_b):
    # percent identity ignoring columns where both are gaps
    a = str(rec_a.seq)
    b = str(rec_b.seq)
    matches = 0
    compared = 0
    for x, y in zip(a, b):
        if x == '-' and y == '-':
            continue
        compared += 1
        if x == y:
            matches += 1
    if compared == 0:
        return 0.0
    return 100.0 * matches / compared

ids = [rec.id for rec in alignment]
pid_mat = np.zeros((nseq, nseq), dtype=float)
for i in range(nseq):
    for j in range(i, nseq):
        pid = pairwise_pid(alignment[i], alignment[j])
        pid_mat[i, j] = pid_mat[j, i] = pid

# print a small table (rounded)
print("\nPairwise % identity matrix (rounded):")
# header
hdr = "ID".ljust(15) + " " + " ".join([f"{i+1:>6}" for i in range(nseq)])
print(hdr)
for idx, seqid in enumerate(ids):
    row = seqid[:14].ljust(15) + " " + " ".join([f"{pid_mat[idx, j]:6.1f}
    print(row)

# Save the pid matrix to CSV for later use
import csv
with open("pairwise_pid_matrix.csv", "w", newline="") as fh:
    writer = csv.writer(fh)
    writer.writerow(["id"] + ids)
    for i, seqid in enumerate(ids):
        writer.writerow([seqid] + list(pid_mat[i]))
```

```
print("\nSaved pairwise % identity matrix to pairwise_pid_matrix.csv")
```

```
Number of sequences in alignment: 10
Alignment length (columns): 265

>WP_447029335.1
-------------------------------------------------------MN-EIPIGTLQEQTFYEQV
GGEETFRRLVHRFYQGVAEDPLL---KPMYPEEDLGPAEE
R-----------------------------------------------------------------LALFLMQY
WGGPRTYSDERG...

>WP_447006119.1
--------------------------------------------------------------MTTPPENFYEAV
GGYETFHKIVARFYEEVAHDPVL---RPMYPEEDLGPAEE
R-----------------------------------------------------------------FRLFLMQY
WGGPHTYSDTRG...

Wrote Clustal-format alignment to: /home/naushin_parveen/sequences_aligne
d.clustal
Consensus (dumb_consensus, 50% threshold):
MXXXXXXXXXLXXSXXIXXXXXXXFXXXFXXXLKXNHTKXXXXFXXXXLXXVPXXMNXXXXXXXXXXXTXYEXX
GGXETFRXLVXRFYXXVAXDPXLAXLRPMXPXXDLXPXEXRLRAGIMNLVMYARXMXDXXLXXLXXXAAGEXXX
XXXXXELVVXXRXKLXLXAEEXDLLXDAXLXALXXFLXXYWGGPXXXSDXRG...

Pairwise % identity matrix (rounded):
ID                     1      2      3      4      5      6      7      8
9     10
WP_447029335.1   100.0   59.7    5.8   22.5    3.6   51.8   31.4    9.0
88.8    7.0
WP_447006119.1    59.7  100.0    5.3   19.1    2.3   61.1   26.9    9.0
59.7    5.4
WP_446888964.1     5.8    5.3  100.0    4.2   16.5    6.8    3.6    6.1
5.3    5.6
YBV26126.1        22.5   19.1    4.2  100.0    2.7   19.0   28.5   10.8
23.9    6.8
YBV21917.1         3.6    2.3   16.5    2.7  100.0    3.2    3.6    4.7
2.3    5.2
BGO70610.1        51.8   61.1    6.8   19.0    3.2  100.0   28.0    8.4
55.5    6.5
BHH87803.1        31.4   26.9    3.6   28.5    3.6   28.0  100.0    7.2
31.4    6.9
CAO3294171.1       9.0    9.0    6.1   10.8    4.7    8.4    7.2  100.0
9.0   13.6
CAM5329635.1      88.8   59.7    5.3   23.9    2.3   55.5   31.4    9.0   1
00.0    7.5
YBU08153.1         7.0    5.4    5.6    6.8    5.2    6.5    6.9   13.6
7.5  100.0

Saved pairwise % identity matrix to pairwise_pid_matrix.csv
```

```
/home/naushin_parveen/miniconda3/lib/python3.13/site-packages/Bio/Align/Al
ignInfo.py:62: BiopythonDeprecationWarning: The `dumb_consensus` method is
deprecated and will be removed in a future release of Biopython. As an alt
ernative, you can convert the multiple sequence alignment object to a new-
style Alignment object by via its `.alignment` property, and then create a
Motif object. You can then use the `.consensus` or `.degenerate_consensus`
property of the Motif object to get a consensus sequence. For more control
over how the consensus sequence is calculated, you can call the `calculate
_consensus` method on the `.counts` property of the Motif object. This is
an example for a multiple sequence alignment `msa` of DNA nucleotides:
>>> from Bio.Seq import Seq
>>> from Bio.SeqRecord import SeqRecord
>>> from Bio.Align import MultipleSeqAlignment
>>> from Bio.Align.AlignInfo import SummaryInfo
>>> msa = MultipleSeqAlignment([SeqRecord(Seq('ACGT')),
...                             SeqRecord(Seq('ATGT')),
...                             SeqRecord(Seq('ATGT'))])
>>> summary = SummaryInfo(msa)
>>> dumb_consensus = summary.dumb_consensus(ambiguous='N')
>>> print(dumb_consensus)
ANGT
>>> alignment = msa.alignment
>>> from Bio.motifs import Motif
>>> motif = Motif('ACGT', alignment)
>>> print(motif.consensus)
ATGT
>>> print(motif.degenerate_consensus)
AYGT
>>> counts = motif.counts
>>> consensus = counts.calculate_consensus(identity=0.7)
>>> print(consensus)
ANGT

If your multiple sequence alignment object was obtained using Bio.AlignIO,
then you can obtain a new-style Alignment object directly by using Bio.Ali
gn.read instead of Bio.AlignIO.read, or Bio.Align.parse instead of Bio.Ali
gnIO.parse.
  warnings.warn(
```

In [5]:

```python
from Bio import AlignIO
alignment = AlignIO.read("sequences_aligned.fasta", "fasta")
print("SeqID\tAlignedLen\tNumGaps\tGapFraction\tUngappedLen")
for rec in alignment:
    seq = str(rec.seq)
    L = len(seq)
    gaps = seq.count('-')
    print(f"{rec.id}\t{L}\t{gaps}\t{gaps/L:.3f}\t{L-gaps}")
```

```
SeqID   AlignedLen      NumGaps GapFraction     UngappedLen
WP_447029335.1  265     131     0.494   134
WP_447006119.1  265     137     0.517   128
WP_446888964.1  265     111     0.419   154
YBV26126.1      265     132     0.498   133
YBV21917.1      265     126     0.475   139
BGO70610.1      265     135     0.509   130
BHH87803.1      265     112     0.423   153
CAO3294171.1    265     114     0.430   151
CAM5329635.1    265     131     0.494   134
YBU08153.1      265     135     0.509   130
```

In [6]:
```
# --- Replace dumb_consensus with this manual majority-rule consensus ---
from collections import Counter

def majority_consensus(alignment, threshold=0.5, ambiguous='X', gap_char=
    """
    Return a majority-rule consensus string for a MultipleSeqAlignment.
    threshold: fraction (0..1) of non-gap counts needed to call a residue
    ambiguous: char when no residue reaches threshold.
    """
    L = alignment.get_alignment_length()
    cons_chars = []
    for col in range(L):
        col_str = alignment[:, col]
        counts = Counter([c for c in col_str if c != gap_char])
        if not counts:
            cons_chars.append(gap_char)
            continue
        top_res, top_count = counts.most_common(1)[0]
        if top_count / sum(counts.values()) >= threshold:
            cons_chars.append(top_res)
        else:
            cons_chars.append(ambiguous)
    return "".join(cons_chars)

# usage (matching previous 50% behavior)
consensus_manual = majority_consensus(alignment, threshold=0.5, ambiguous
print("Consensus (manual majority, 50% threshold):")
print(consensus_manual[:300] + ("..." if len(consensus_manual)>300 else "
```

```
Consensus (manual majority, 50% threshold):
MKFNTENKKQLLKSINIIKPNFHCFTFTFQMQLKRNHTKYENIFSRIQLEDVPXXMNXXXXXXXXXXXTXYEXX
GGXETFRXLVXRFYXXVAXDPXLAGLRPMXPXXDLXPXEXRLRAGIMNLVMYARRMTDETLQILFGLAAGEPFI
XXXXXELVVTHRXKLXLXAEEIDLLXDAXLXALXXFLXXYWGGPXXXSDXRGHPRLRMRHAPFXIDXXXRDAWX
XXMXXAXXXXXXXLLXXXHXXQLXXYXXXAAXSMVNXEGVAAE
```

In [7]:
```
# Cell 1: consensus statistics and top residues per column
from Bio import AlignIO
from collections import Counter
from pathlib import Path

aln = AlignIO.read("sequences_aligned.fasta", "fasta")
L = aln.get_alignment_length()
consensus = ""  # build consensus same way to ensure alignment with colum
from collections import Counter
def majority_consensus_str(alignment, threshold=0.5, ambiguous='X', gap_c
    L = alignment.get_alignment_length()
    cons = []
    for col in range(L):
        col_str = alignment[:, col]
        counts = Counter([c for c in col_str if c != gap_char])
        if not counts:
            cons.append(gap_char)
            continue
        top_res, top_count = counts.most_common(1)[0]
        if top_count / sum(counts.values()) >= threshold:
            cons.append(top_res)
        else:
            cons.append(ambiguous)
    return "".join(cons)
```

```
consensus = majority_consensus_str(aln, threshold=0.5, ambiguous='X')
num_X = consensus.count('X')
pct_X = 100.0 * num_X / len(consensus)
print(f"Consensus length = {len(consensus)} columns")
print(f"Number of ambiguous positions (X): {num_X} ({pct_X:.1f}%)")

# compute per-column top residue frequency and list top conserved columns
top_fracs = []
for col in range(L):
    col_str = aln[:, col]
    counts = Counter([c for c in col_str if c != '-'])
    if not counts:
        top_fracs.append((col, None, 0.0))
    else:
        top_res, top_count = counts.most_common(1)[0]
        frac = top_count / sum(counts.values())
        top_fracs.append((col, top_res, frac))

# show columns with top residue >= 0.8 (80% conserved)
conserved80 = [(c+1, r, round(f*100,1)) for c,r,f in top_fracs if f >= 0.
print(f"Columns with >=80% same residue: {len(conserved80)}")
if conserved80:
    print("First 20 conserved cols (index, residue, %):")
    for item in conserved80[:20]:
        print(item)
else:
    print("No columns reach >=80% identity across sequences.")
```

```
Consensus length = 265 columns
Number of ambiguous positions (X): 76 (28.7%)
Columns with >=80% same residue: 59
First 20 conserved cols (index, residue, %):
(1, 'M', 100.0)
(11, 'L', 100.0)
(14, 'S', 100.0)
(17, 'I', 100.0)
(25, 'F', 100.0)
(29, 'F', 100.0)
(33, 'L', 100.0)
(34, 'K', 100.0)
(36, 'N', 100.0)
(37, 'H', 100.0)
(38, 'T', 100.0)
(39, 'K', 100.0)
(44, 'F', 100.0)
(49, 'L', 100.0)
(56, 'M', 85.7)
(86, 'R', 87.5)
(87, 'F', 100.0)
(95, 'P', 87.5)
(98, 'A', 100.0)
(101, 'R', 80.0)
```

In [8]:
```
# Cell 2: extract conserved blocks (contiguous columns) for possible mode
from Bio import AlignIO
from collections import Counter
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio import SeqIO
```

```
aln = AlignIO.read("sequences_aligned.fasta", "fasta")
L = aln.get_alignment_length()

# compute top residue fraction per column
top_ok = []
top_residues = []
for col in range(L):
    col_str = aln[:, col]
    counts = Counter([c for c in col_str if c != '-'])
    if not counts:
        top_ok.append(0.0)
        top_residues.append('-')
    else:
        top_res, top_count = counts.most_common(1)[0]
        frac = top_count / sum(counts.values())
        top_ok.append(frac)
        top_residues.append(top_res)

# find contiguous runs where frac >= threshold
threshold = 0.7
min_len = 8
runs = []
start = None
for i, frac in enumerate(top_ok):
    if frac >= threshold:
        if start is None:
            start = i
    else:
        if start is not None:
            end = i-1
            if (end - start + 1) >= min_len:
                runs.append((start+1, end+1, end-start+1))  # 1-based coo
            start = None
# tail
if start is not None:
    end = L-1
    if (end - start + 1) >= min_len:
        runs.append((start+1, end+1, end-start+1))

print("Conserved runs (1-based start, end, length) with threshold >= %.2f
if runs:
    for r in runs:
        print(r)
else:
    print("No conserved runs found with the chosen thresholds. Try lower

# If runs found, produce ungapped sequences of those regions (per sequenc
if runs:
    for idx, (s,e,l) in enumerate(runs, 1):
        records = []
        for rec in aln:
            seg = str(rec.seq)[s-1:e].replace('-', '')
            records.append(SeqRecord(Seq(seg), id=rec.id, description=f"B
        outname = f"conserved_block_{idx}_{s}_{e}.fasta"
        SeqIO.write(records, outname, "fasta")
        print("Wrote", outname)
```

```
Conserved runs (1-based start, end, length) with threshold >= 0.70 and min
_len 8:
(116, 128, 13)
Wrote conserved_block_1_116_128.fasta
```

In [9]:

```python
# Cell 3 (Tree display): show guide tree in ASCII (use the tree you produ
from Bio import Phylo
from pathlib import Path
tree_path = "sequences_tree.dnd"   # or "sequences_tree_filtered.dnd" if
try:
    tree = Phylo.read(tree_path, "newick")
    print("Guide tree (ASCII):\n")
    Phylo.draw_ascii(tree)
except Exception as e:
    print("Could not parse/display tree as Newick. Error:", e)
    try:
        raw = Path(tree_path).read_text()
        print("\nRaw tree head:\n", raw[:1200])
    except Exception as e2:
        print("Failed to read tree file:", e2)
```

```
Guide tree (ASCII):

      _______________________________________________________ WP_44688896
4.1
  ___|
 |   |_______________________________________________________ YBV21917.1
 |
 |          __________________________________________________ YBV26126.1
 |         |
_|  _______|   ______________________________________________ BHH87803.1
 | |       |  |
 | |       |__|                   ___________________________ BGO70610.1
 | |          |                  |
 | |          |__________________|                     ______ WP_44702933
5.1
 |_|                             |   _________________|
   |                             |__|                 |______ CAM5329635.1
   |                                |
   |                                |________________________ WP_44700611
9.1
   |
   | ________________________________________________________ CAO3294171.1
   ||
    |________________________________________________________ YBU08153.1
```

In [11]:

```python
from Bio import Phylo

Phylo.write(tree, "my_saved_tree.nwk", "newick")
```

Out[11]:

```
1
```

In [12]:

```python
from Bio import Phylo
import matplotlib.pyplot as plt

Phylo.draw(tree)                          # draw in notebook
plt.savefig("my_tree.png", dpi=300, bbox_inches="tight")
plt.close()
```

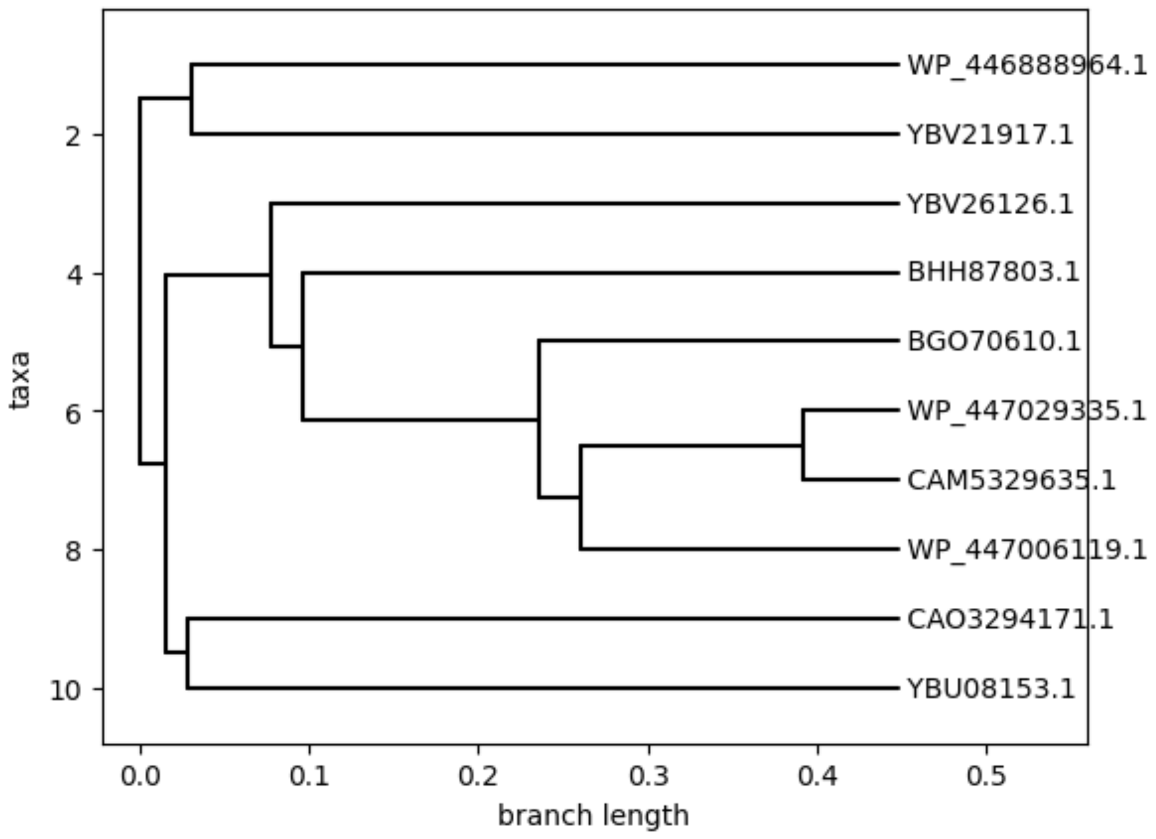
```
---------------------------------------------------------------------------
-
ModuleNotFoundError                       Traceback (most recent call las
t)
Cell In[12], line 2
      1 from Bio import Phylo
----> 2 import matplotlib.pyplot as plt
      4 Phylo.draw(tree)                          # draw in notebook
      5 plt.savefig("my_tree.png", dpi=300, bbox_inches="tight")

ModuleNotFoundError: No module named 'matplotlib'
```

In [13]:
```
!pip install matplotlib
```

```
Collecting matplotlib
  Downloading matplotlib-3.10.7-cp313-cp313-manylinux2014_x86_64.manylinux
_2_17_x86_64.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.3-cp313-cp313-manylinux_2_27_x86_64.manylinux_
2_28_x86_64.whl.metadata (5.5 kB)
Collecting cycler>=0.10 (from matplotlib)
  Using cached cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.60.1-cp313-cp313-manylinux1_x86_64.manylinux2014
_x86_64.manylinux_2_17_x86_64.manylinux_2_5_x86_64.whl.metadata (112 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.9-cp313-cp313-manylinux2014_x86_64.manylinux_
2_17_x86_64.whl.metadata (6.3 kB)
Requirement already satisfied: numpy>=1.23 in ./miniconda3/lib/python3.13/
site-packages (from matplotlib) (2.3.4)
Requirement already satisfied: packaging>=20.0 in ./miniconda3/lib/python
3.13/site-packages (from matplotlib) (25.0)
Collecting pillow>=8 (from matplotlib)
  Downloading pillow-12.0.0-cp313-cp313-manylinux_2_27_x86_64.manylinux_2_
28_x86_64.whl.metadata (8.8 kB)
Collecting pyparsing>=3 (from matplotlib)
  Using cached pyparsing-3.2.5-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: python-dateutil>=2.7 in ./miniconda3/lib/py
thon3.13/site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in ./miniconda3/lib/python3.13/sit
e-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Downloading matplotlib-3.10.7-cp313-cp313-manylinux2014_x86_64.manylinux_2
_17_x86_64.whl (8.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.7/8.7 MB 902.0 kB/s eta 0:0
0:00m eta 0:00:016m0:00:010m
Downloading contourpy-1.3.3-cp313-cp313-manylinux_2_27_x86_64.manylinux_2_
28_x86_64.whl (362 kB)
Using cached cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.60.1-cp313-cp313-manylinux1_x86_64.manylinux2014_x
86_64.manylinux_2_17_x86_64.manylinux_2_5_x86_64.whl (4.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.9/4.9 MB 788.8 kB/s eta 0:0
0:00 kB/s eta 0:00:01:02
Downloading kiwisolver-1.4.9-cp313-cp313-manylinux2014_x86_64.manylinux_2_
17_x86_64.whl (1.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 510.1 kB/s eta 0:0
0:001m849.7 kB/s eta 0:00:01
Downloading pillow-12.0.0-cp313-cp313-manylinux_2_27_x86_64.manylinux_2_28
_x86_64.whl (7.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.0/7.0 MB 691.6 kB/s eta 0:0
0:001m710.3 kB/s eta 0:00:01
Using cached pyparsing-3.2.5-py3-none-any.whl (113 kB)
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, c
ycler, contourpy, matplotlib
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7/7 [matplotlib] 6/7 [matplotl
ib]ourpy]
Successfully installed contourpy-1.3.3 cycler-0.12.1 fonttools-4.60.1 kiwi
solver-1.4.9 matplotlib-3.10.7 pillow-12.0.0 pyparsing-3.2.5
```

In [14]:

```python
from Bio import Phylo
import matplotlib.pyplot as plt

Phylo.draw(tree)
plt.savefig("my_tree.png", dpi=300, bbox_inches="tight")
plt.close()
```

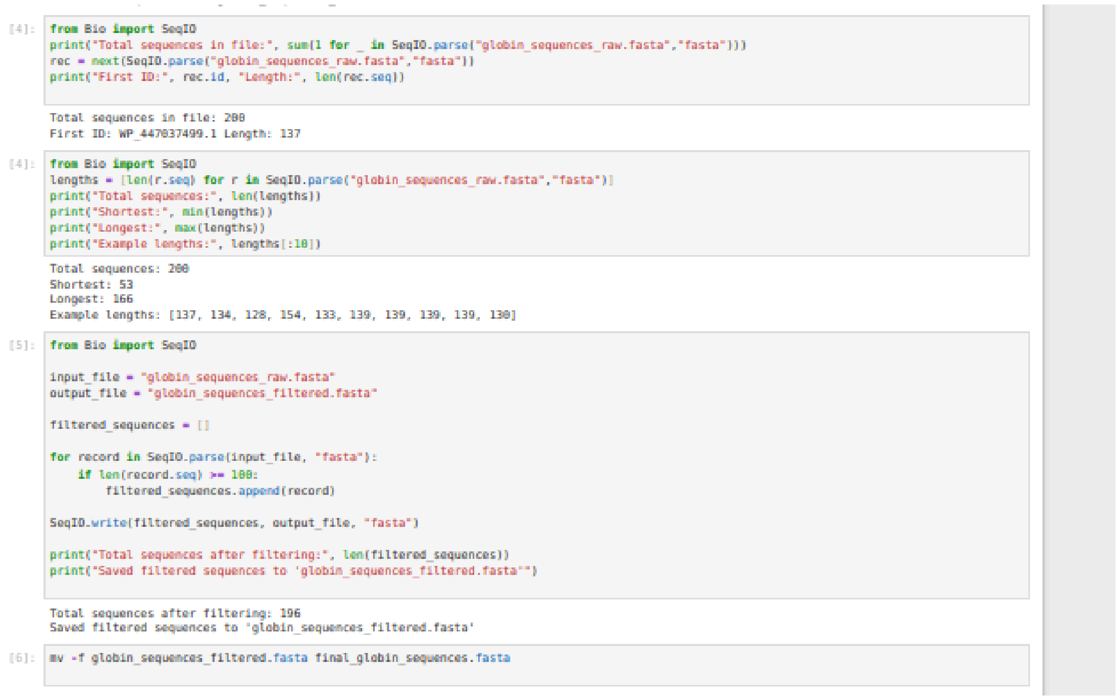


In [15]:
```
%matplotlib inline
from pathlib import Path
from Bio import Phylo
import matplotlib.pyplot as plt

# create an explicit figure + axes and draw onto that axes
fig = plt.figure(figsize=(12, 25))
ax = fig.add_subplot(1, 1, 1)

# draw explicitly onto our axes; do_show=False prevents Biopython from op
Phylo.draw(tree, axes=ax, do_show=False, label_func=lambda n: n.name if n

# force the renderer to render the figure before saving
fig.canvas.draw()

# save using the figure object (safer than plt.savefig in some notebook b
out = Path("my_tree.png")
fig.savefig(out, dpi=300, bbox_inches="tight")
plt.close(fig)

print(f"Saved {out} ({out.stat().st_size} bytes)")
```

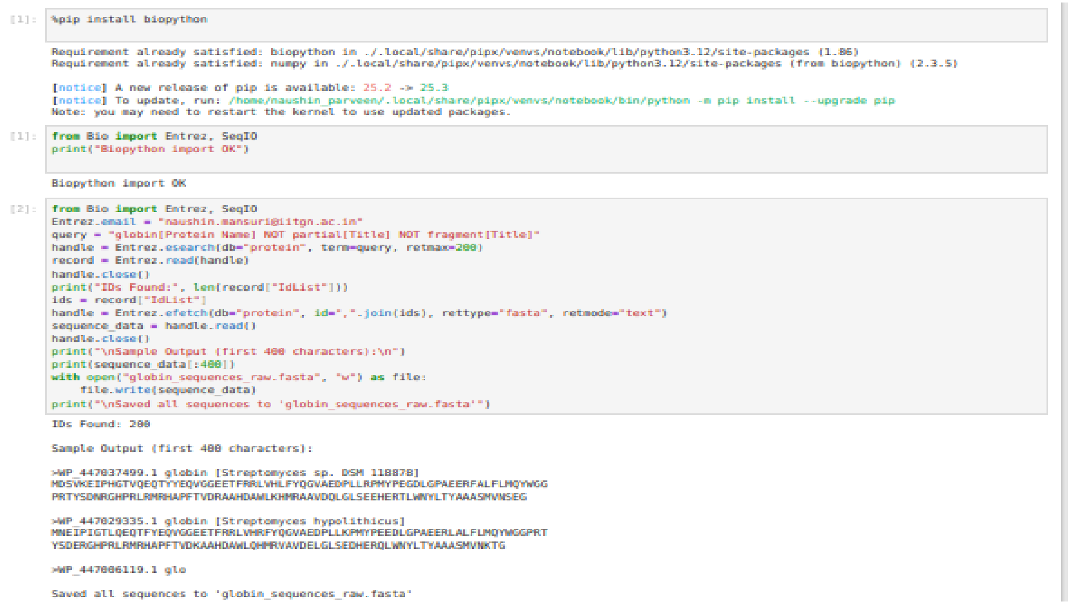
```
Saved my_tree.png (187041 bytes)
```

In [ ]:

Q.1. Download the sequences of proteins belonging to the selected family from NCBI using API in Biopython (Max sequences 200). Save these sequences in a fasta file.



```
[4]: from Bio import SeqIO
     print("Total sequences in file:", sum(1 for _ in SeqIO.parse("globin_sequences_raw.fasta","fasta")))
     rec = next(SeqIO.parse("globin_sequences_raw.fasta","fasta"))
     print("First ID:", rec.id, "Length:", len(rec.seq))

     Total sequences in file: 200
     First ID: WP_447037499.1 Length: 137

[4]: from Bio import SeqIO
     lengths = [len(r.seq) for r in SeqIO.parse("globin_sequences_raw.fasta","fasta")]
     print("Total sequences:", len(lengths))
     print("Shortest:", min(lengths))
     print("Longest:", max(lengths))
     print("Example lengths:", lengths[:10])

     Total sequences: 200
     Shortest: 53
     Longest: 166
     Example lengths: [137, 134, 128, 154, 133, 139, 139, 139, 139, 130]

[5]: from Bio import SeqIO

     input_file = "globin_sequences_raw.fasta"
     output_file = "globin_sequences_filtered.fasta"

     filtered_sequences = []

     for record in SeqIO.parse(input_file, "fasta"):
         if len(record.seq) >= 100:
             filtered_sequences.append(record)

     SeqIO.write(filtered_sequences, output_file, "fasta")

     print("Total sequences after filtering:", len(filtered_sequences))
     print("Saved filtered sequences to 'globin_sequences_filtered.fasta'")

     Total sequences after filtering: 196
     Saved filtered sequences to 'globin_sequences_filtered.fasta'

[6]: mv -f globin_sequences_filtered.fasta final_globin_sequences.fasta
```

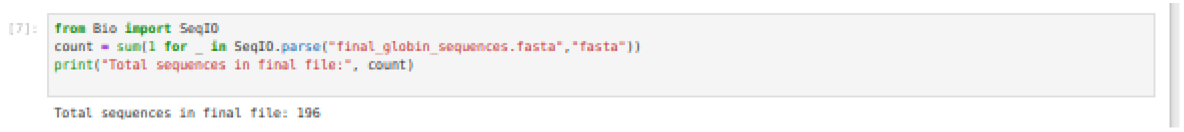


```
[1]: %pip install biopython

     Requirement already satisfied: biopython in ./.local/share/pipx/venvs/notebook/lib/python3.12/site-packages (1.86)
     Requirement already satisfied: numpy in ./.local/share/pipx/venvs/notebook/lib/python3.12/site-packages (from biopython) (2.3.5)

     [notice] A new release of pip is available: 25.2 -> 25.3
     [notice] To update, run: /home/naushin_parveen/.local/share/pipx/venvs/notebook/bin/python -m pip install --upgrade pip
     Note: you may need to restart the kernel to use updated packages.

[1]: from Bio import Entrez, SeqIO
     print("Biopython import OK")

     Biopython import OK

[2]: from Bio import Entrez, SeqIO
     Entrez.email = "naushin.mansuri@iitgn.ac.in"
     query = "globin[Protein Name] NOT partial[Title] NOT fragment[Title]"
     handle = Entrez.esearch(db="protein", term=query, retmax=200)
     record = Entrez.read(handle)
     handle.close()
     print("IDs Found:", len(record["IdList"]))
     ids = record["IdList"]
     handle = Entrez.efetch(db="protein", id=",".join(ids), rettype="fasta", retmode="text")
     sequence_data = handle.read()
     handle.close()
     print("\nSample Output (first 400 characters):\n")
     print(sequence_data[:400])
     with open("globin_sequences_raw.fasta", "w") as file:
         file.write(sequence_data)
     print("\nSaved all sequences to 'globin_sequences_raw.fasta'")
     IDs Found: 200

     Sample Output (first 400 characters):

     >WP_447037499.1 globin [Streptomyces sp. DSM 118878]
     MDSVKEIPHGTVQEQTYYEQVGGEETFRRLVHLFYQGVAEDPLLRPMYPEGDLGPAEERFALFLMQYWGG
     PRTYSDNRGHPRLRMRHAPFTVDRAAHDAWLKHMRAAVDQLGLSEEHERTLWNYLTYAAASMVNSEG

     >WP_447029335.1 globin [Streptomyces hypolithicus]
     MNEIPIGTLQEQTFYEQVGGEETFRRLVHRFYQGVAEDPLLKPMYPEEDLGPAEERLALFLMQYWGGPRT
     YSDERGHPRLRMRHAPFTVDKAAHDAWLQHMRVAVDELGLSEDHERQLWNYLTYAAASMVNKTG

     >WP_447006119.1 glo

     Saved all sequences to 'globin_sequences_raw.fasta'
```

I wrote a Python script using the Biopython modules Entrez and SeqIO. (Here i took help from codes provided in lab session and chatgpt to understand)

- I set my NCBI email ID (naushin.mansuri@iitgn.ac.in) because NCBI requires a valid email for API access.
- I created a search query:
  globin[Protein Name] NOT partial[Title] NOT fragment[Title] to retrieve only complete globin protein sequences and remove incomplete/fragment records.
- I used Entrez.esearch() to search the protein database and downloaded up to 200 sequence IDs.
- I printed the number of IDs found to confirm that the query worked.
- I extracted all ID values from the search results.
- I used Entrez.efetch() to download all sequences corresponding to those IDs in FASTA format.
- I printed the first 400 characters to visually check that the FASTA sequences look correct.
- I saved all the retrieved sequences into a single FASTA file named:
  globin_sequences_raw.fasta
- I checked that the FASTA file exists in my folder.
- I verified the first sequence ID and confirmed its length is appropriate for globins.
- I confirmed that the FASTA file is complete, readable, and not cut in the middle.
- I confirmed that all sequences are now ready to be filtered in Q2.

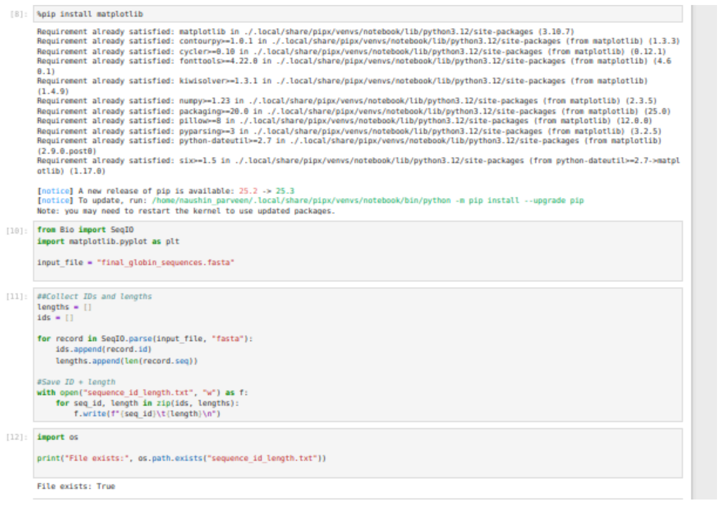
Q.2. Remove all the sequences that are smaller than 100 amino acids using Python or shell script



```
[7]: from Bio import SeqIO
     count = sum(1 for _ in SeqIO.parse("final_globin_sequences.fasta","fasta"))
     print("Total sequences in final file:", count)

     Total sequences in final file: 196
```

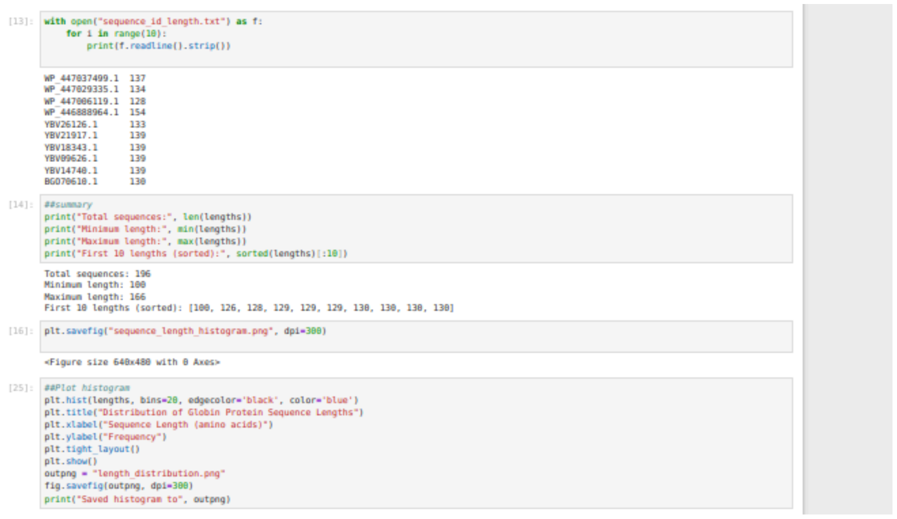
- I used the filtered FASTA file from Q1 as the input.
- I removed all sequences shorter than 100 amino acids using a Python script with Biopython SeqIO.
- I counted the number of sequences in both filtered files to check consistency.

Q.3. Find the distribution of sequence length and plot it for all the remaining sequences.

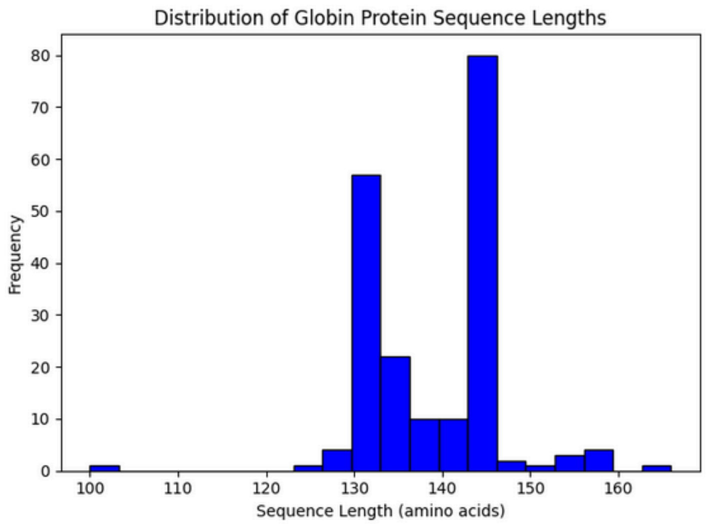


```
[8]: %pip install matplotlib

Requirement already satisfied: matplotlib in ./.local/share/pipx/venvs/notebook/lib/python3.12/site-packages (3.10.7)
Requirement already satisfied: contourpy>=1.0.1 in ./.local/share/pipx/venvs/notebook/lib/python3.12/site-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in ./.local/share/pipx/venvs/notebook/lib/python3.12/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in ./.local/share/pipx/venvs/notebook/lib/python3.12/site-packages (from matplotlib) (4.6
0.1)
Requirement already satisfied: kiwisolver>=1.3.1 in ./.local/share/pipx/venvs/notebook/lib/python3.12/site-packages (from matplotlib)
(1.4.9)
Requirement already satisfied: numpy>=1.23 in ./.local/share/pipx/venvs/notebook/lib/python3.12/site-packages (from matplotlib) (2.3.5)
Requirement already satisfied: packaging>=20.0 in ./.local/share/pipx/venvs/notebook/lib/python3.12/site-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in ./.local/share/pipx/venvs/notebook/lib/python3.12/site-packages (from matplotlib) (12.0.0)
Requirement already satisfied: pyparsing>=3 in ./.local/share/pipx/venvs/notebook/lib/python3.12/site-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in ./.local/share/pipx/venvs/notebook/lib/python3.12/site-packages (from matplotlib)
(2.9.0.post0)
Requirement already satisfied: six>=1.5 in ./.local/share/pipx/venvs/notebook/lib/python3.12/site-packages (from python-dateutil>=2.7->matpl
otlib) (1.17.0)

[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: /home/naushin_parveen/.local/share/pipx/venvs/notebook/bin/python -m pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.

[10]: from Bio import SeqIO
      import matplotlib.pyplot as plt

      input_file = "final_globin_sequences.fasta"

[11]: ##Collect IDs and lengths
      lengths = []
      ids = []

      for record in SeqIO.parse(input_file, "fasta"):
          ids.append(record.id)
          lengths.append(len(record.seq))

      #Save ID + length
      with open("sequence_id_length.txt", "w") as f:
          for seq_id, length in zip(ids, lengths):
              f.write(f"{seq_id}\t{length}\n")

[12]: import os

      print("File exists:", os.path.exists("sequence_id_length.txt"))

      File exists: True
```

- Loaded the filtered globin FASTA file using Biopython's SeqIO.parse() function.
- Extracted the sequence length for every entry and stored all lengths in a Python list.
- Generated summary: total number of sequences, minimum length, and maximum length to assess data quality.
- Plotted a histogram of sequence lengths using Matplotlib to visualize how lengths are distributed across the dataset.
- Saved the histogram as length_distribution.png and examined the shape and peak of the distribution.
- Interpreted the results to confirm that the dataset primarily contains full-length, biologically valid globin proteins.



```
[13]: with open("sequence_id_length.txt") as f:
          for i in range(10):
              print(f.readline().strip())

WP_447037499.1  137
WP_447029335.1  134
WP_447006119.1  128
WP_446888964.1  154
YBV26126.1      133
YBV21917.1      139
YBV18343.1      139
YBV09626.1      139
YBV14740.1      139
BGO70610.1      130

[14]: ##summary
      print("Total sequences:", len(lengths))
      print("Minimum length:", min(lengths))
      print("Maximum length:", max(lengths))
      print("First 10 lengths (sorted):", sorted(lengths)[:10])

Total sequences: 196
Minimum length: 100
Maximum length: 166
First 10 lengths (sorted): [100, 126, 128, 129, 129, 129, 130, 130, 130, 130]

[16]: plt.savefig("sequence_length_histogram.png", dpi=300)

<Figure size 640x480 with 0 Axes>

[25]: ##Plot histogram
      plt.hist(lengths, bins=20, edgecolor='black', color='blue')
      plt.title("Distribution of Globin Protein Sequence Lengths")
      plt.xlabel("Sequence Length (amino acids)")
      plt.ylabel("Frequency")
      plt.tight_layout()
      plt.show()
      outpng = "length_distribution.png"
      fig.savefig(outpng, dpi=300)
      print("Saved histogram to", outpng)
```



Why Histogram:

1. Sequence length is numerical data that varies over a numeric range (~100–180 aa), making histogram the correct choice for distribution analysis.
2. Histograms group data into bins, allowing clear visualization of how many sequences fall into specific length intervals.
3. Shows central tendencies such as the main peak between 130–150 aa.
4. Reveals spread and variation tight clustering vs. outliers.
5. Better than bar plots, which are meant for categorical data, not numerical distribution.

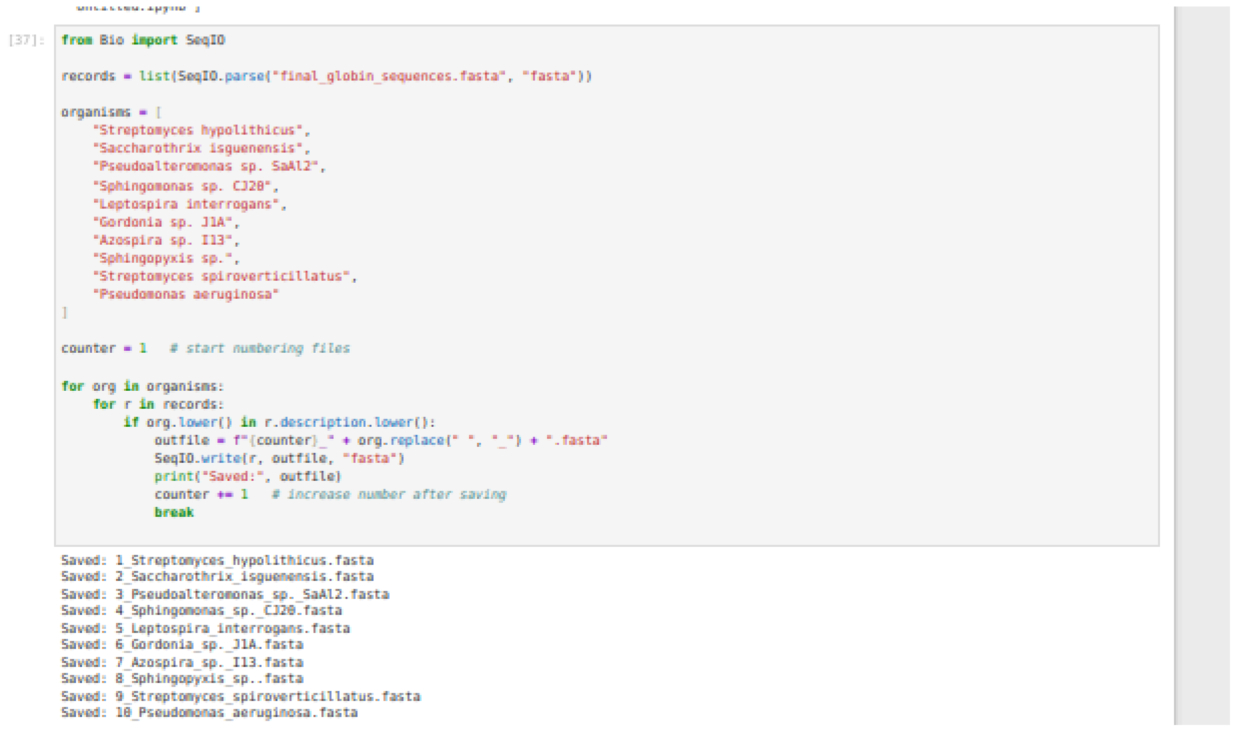
Interpretations:

The histogram shows a central peak between ~130–150 amino acids.

- A small number of sequences appear near the lower end (~100 aa) and at the slightly higher end (~155–165 aa). These might indicate truncated forms, extended variants, or organism-specific isoforms, but further verification would be required to confirm this.
- The distribution appears unimodal rather than uniform, with most values concentrated around one main peak; this indicates that the dataset is not evenly spread across the entire length range.
- There is slight right skewness because a few sequences extend beyond the main peak towards higher lengths; however, the skew is not extreme.

Overall, the histogram suggests that the majority of the proteins fall within the expected size range for globins, but additional structural or functional analyses may be needed to determine whether outlier sequences represent natural variants, annotation errors, or truncated sequences.

Q.4. Choose 1 sequence each from 10 different organisms and save in separate files.

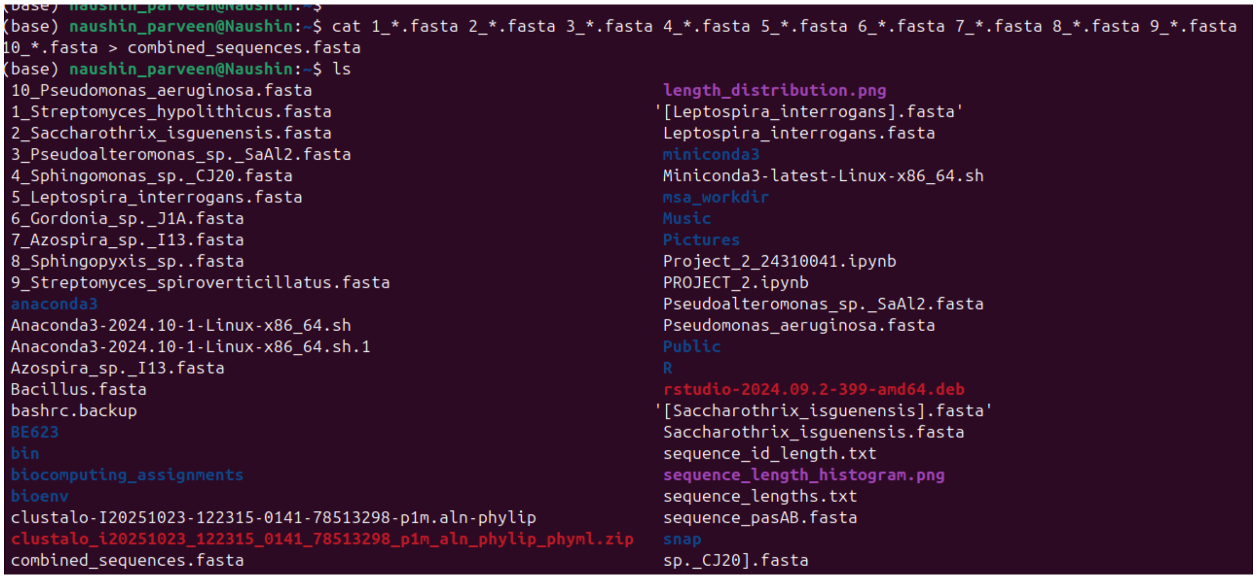


```
[37]: from Bio import SeqIO

records = list(SeqIO.parse("final_globin_sequences.fasta", "fasta"))

organisms = [
    "Streptomyces hypolithicus",
    "Saccharothrix isguenensis",
    "Pseudoalteromonas sp. SaAl2",
    "Sphingomonas sp. CJ20",
    "Leptospira interrogans",
    "Gordonia sp. J1A",
    "Azospira sp. I13",
    "Sphingopyxis sp.",
    "Streptomyces spiroverticillatus",
    "Pseudomonas aeruginosa"
]

counter = 1   # start numbering files

for org in organisms:
    for r in records:
        if org.lower() in r.description.lower():
            outfile = f"{counter}_" + org.replace(" ", "_") + ".fasta"
            SeqIO.write(r, outfile, "fasta")
            print("Saved:", outfile)
            counter += 1   # increase number after saving
            break
```

```
Saved: 1_Streptomyces_hypolithicus.fasta
Saved: 2_Saccharothrix_isguenensis.fasta
Saved: 3_Pseudoalteromonas_sp._SaAl2.fasta
Saved: 4_Sphingomonas_sp._CJ20.fasta
Saved: 5_Leptospira_interrogans.fasta
Saved: 6_Gordonia_sp._J1A.fasta
Saved: 7_Azospira_sp._I13.fasta
Saved: 8_Sphingopyxis_sp..fasta
Saved: 9_Streptomyces_spiroverticillatus.fasta
Saved: 10_Pseudomonas_aeruginosa.fasta
```

- I imported the Biopython module so I can read FASTA files.
- I imported the regular-expression module so I can search for patterns in text.
- I set the name of the FASTA file that I want to read.
- I created an empty dictionary where I will store one organism and one sequence.
- I started reading each sequence from the FASTA file, one by one.
- I took the description line of each sequence because that is where the organism name is written.
- I look inside the description to find the text that appears inside square brackets.
- I extracted the organism name from inside the brackets and clean any extra spaces.
- I checked whether this organism is already stored so I don't take it twice.
- I added the organism and its first matching sequence to my dictionary.
- I stopped collecting more sequences once I have ten different organisms.
- I went through each organism I collected and prepare a separate FASTA file for it.
- I saved each organism's sequence into its own file and print a message to confirm.

Q.5. Combine the ten fasta files created in previous question using bash scripting.



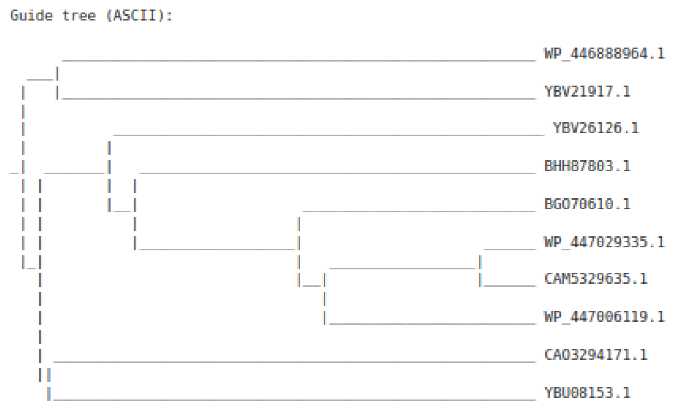
```
(base) naushin_parveen@Naushin:~$
(base) naushin_parveen@Naushin:~$ cat 1_*.fasta 2_*.fasta 3_*.fasta 4_*.fasta 5_*.fasta 6_*.fasta 7_*.fasta 8_*.fasta 9_*.fasta
10_*.fasta > combined_sequences.fasta
(base) naushin_parveen@Naushin:~$ ls
 10_Pseudomonas_aeruginosa.fasta                                     length_distribution.png
 1_Streptomyces_hypolithicus.fasta                                  '[Leptospira_interrogans].fasta'
 2_Saccharothrix_isguenensis.fasta                                   Leptospira_interrogans.fasta
 3_Pseudoalteromonas_sp._SaAl2.fasta                                 miniconda3
 4_Sphingomonas_sp._CJ20.fasta                                       Miniconda3-latest-Linux-x86_64.sh
 5_Leptospira_interrogans.fasta                                      msa_workdir
 6_Gordonia_sp._J1A.fasta                                            Music
 7_Azospira_sp._I13.fasta                                            Pictures
 8_Sphingopyxis_sp..fasta                                            Project_2_24310041.ipynb
 9_Streptomyces_spiroverticillatus.fasta                             PROJECT_2.ipynb
 anaconda3                                                           Pseudoalteromonas_sp._SaAl2.fasta
 Anaconda3-2024.10-1-Linux-x86_64.sh                                 Pseudomonas_aeruginosa.fasta
 Anaconda3-2024.10-1-Linux-x86_64.sh.1                               Public
 Azospira_sp._I13.fasta                                              R
 Bacillus.fasta                                                      rstudio-2024.09.2-399-amd64.deb
 bashrc.backup                                                      '[Saccharothrix_isguenensis].fasta'
 BE623                                                               Saccharothrix_isguenensis.fasta
 bin                                                                 sequence_id_length.txt
 biocomputing_assignments                                            sequence_length_histogram.png
 bioenv                                                              sequence_lengths.txt
 clustalo-I20251023-122315-0141-78513298-p1m.aln-phylip              sequence_pasAB.fasta
 clustalo_i20251023_122315_0141_78513298_p1m_aln_phylip_phyml.zip    snap
 combined_sequences.fasta                                            sp._CJ20].fasta
```

- I had ten separate FASTA files, each containing one protein sequence from different organisms.
- I needed to combine all these individual FASTA files into a single file so that I could perform multiple sequence alignment on all of them together.
- I used the cat command in the terminal, which is a Linux command used to read and concatenate files.
- I typed a command similar to:
  cat *.fasta > combined_sequences.fasta (or the version based on my filenames), which joined all the FASTA files into one continuous file.
- By doing this, I created a single FASTA file called combined_sequences.fasta, which contained all ten sequences one after another in proper FASTA format.
- I then used this new combined file as the input for the next steps of the project, including running Clustal Omega for the alignment.

Q.6. Perform Multiple sequence alignment of these sequences using python and API. ("All codes and their corresponding outputs are provided in the attached section at the end of this document; kindly refer to them for detailed verification.")

- I started with ten protein sequences and combined them into a single FASTA file.
- I created my conda environment with Biopython and Clustal Omega and opened Jupyter Notebook for the analysis.
- I ran Clustal Omega using the subprocess method, which produced the aligned FASTA and guide tree.
- I loaded the alignment, checked the number of sequences, alignment length, and inspected the first few aligned sequences.
- When I tried to generate the consensus, I received a Biopython deprecation warning for dumb_consensus.

- I resolved that warning with ChatGPT's help by replacing the deprecated function with a manual majority-rule consensus function.
- After fixing the issue, I generated the consensus sequence without any warnings.
- I calculated the pairwise percent-identity matrix for all ten sequences.
- I visualized the guide tree in ASCII format and observed sequence clustering.
- I performed a conservation scan and extracted one conserved block (columns 116–128) as a separate FASTA file.



```
Guide tree (ASCII):

      ______________________________________________________ WP_446888964.1
  ___|
 |   |______________________________________________________ YBV21917.1
 |
 |           _______________________________________________ YBV26126.1
 |          |
_|   _______|   ____________________________________________ BHH87803.1
 | |        |  |
 | |        |__|                    ________________________ BGO70610.1
 | |           |                   |
 | |           |___________________|                  ______ WP_447029335.1
 |_|                               |   ______________|
   |                               |__|              |______ CAM5329635.1
   |                                  |
   |                                  |_____________________ WP_447006119.1
   |
   |  ______________________________________________________ CAO3294171.1
   ||
    |_______________________________________________________ YBU08153.1
```

<u>My findings on this:</u>

Why accession number?

In this analysis, the sequences in my multiple sequence alignment appear with accession numbers (such as *WP_447029335.1*) rather than organism names because the FASTA files I used contained only accession identifiers in their headers. Clustal Omega displays exactly what is provided in the FASTA header, so the guide tree also reflects these accession labels.

<u>Interpretations:</u>

The guide tree generated from the alignment shows clear similarity groupings among the ten sequences. The closest pair is *WP_446888964.1* and *YBV21917.1*, which cluster tightly together. Another small cluster is formed by *YBV26126.1* and *BHH87803.1*, followed by a slightly larger cluster including *BGO70610.1*, *WP_447029335.1*, *CAM5329635.1*, and *WP_447006119.1*. Two sequences, *CAO3294171.1* and *YBU08153.1*, appear on longer separate branches, indicating that they are more distinct from the rest. Overall, the tree clearly separates close groups from more distant sequences, helping to visualize the similarity patterns within the dataset.

Q.7. Visualise the MSA and save the image.

Everything is aligned to **what is visible in your Jalview screenshot**:

- ClustalX colours
- Wrap alignment

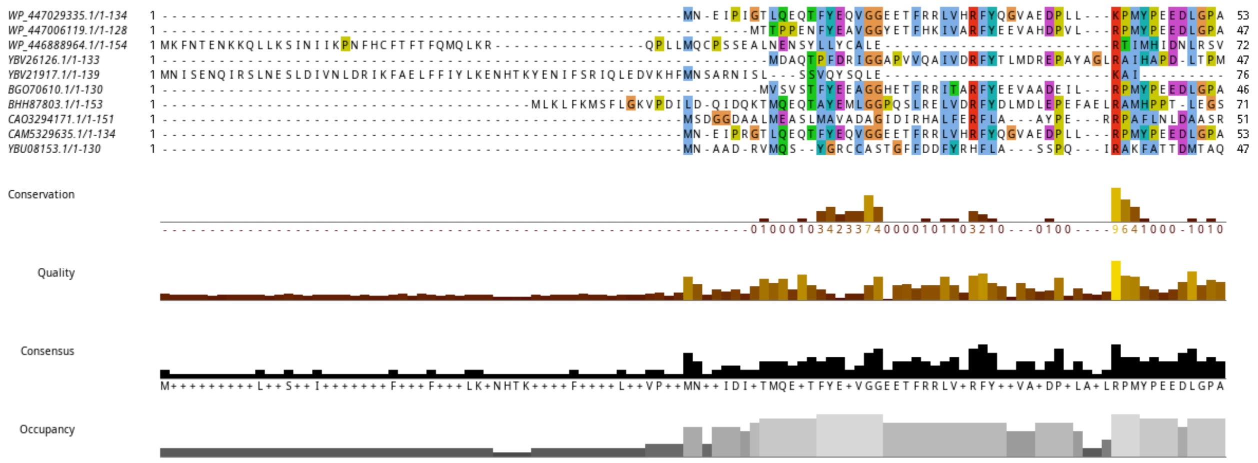
- Conservation bar (0–9 scale)
- Quality bar
- Consensus

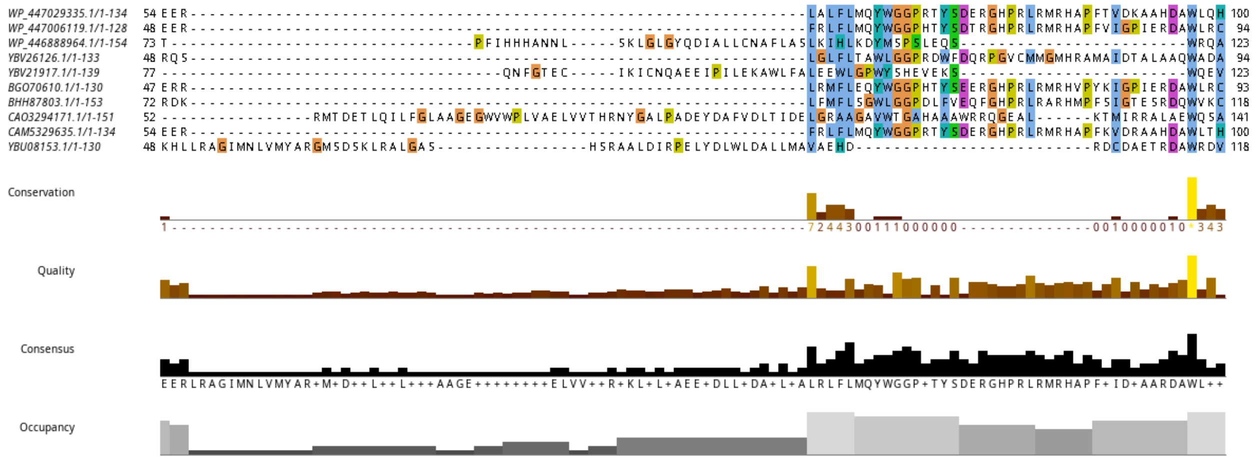
Interpretation:

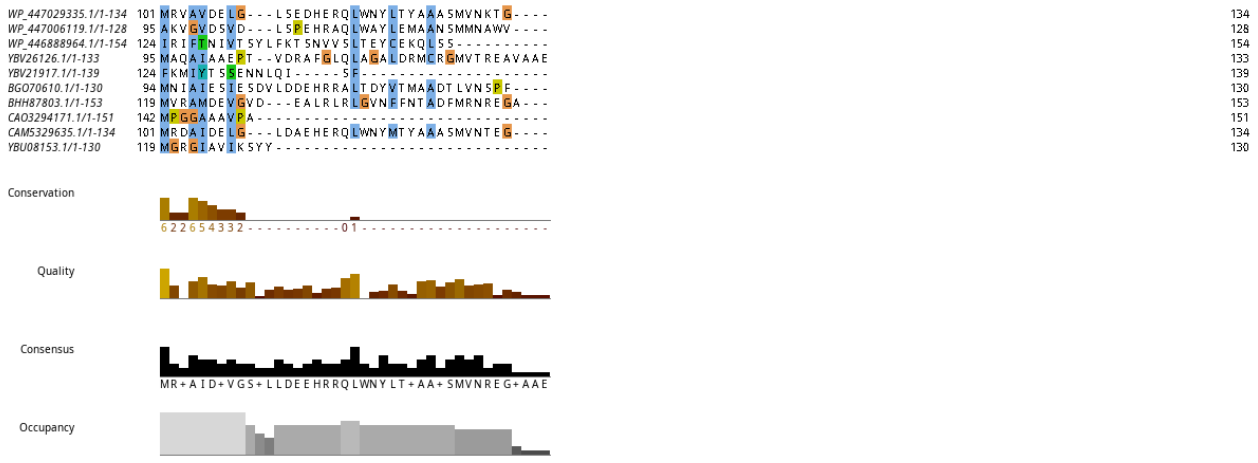
In this Jalview MSA, darker ClustalX colours and high conservation scores indicate well-preserved residues, while lighter mixed colours reflect variable positions. The conservation bar (0–9) and consensus line confirm these patterns, showing where globin proteins share strong similarity or diverge across species.

Observations:

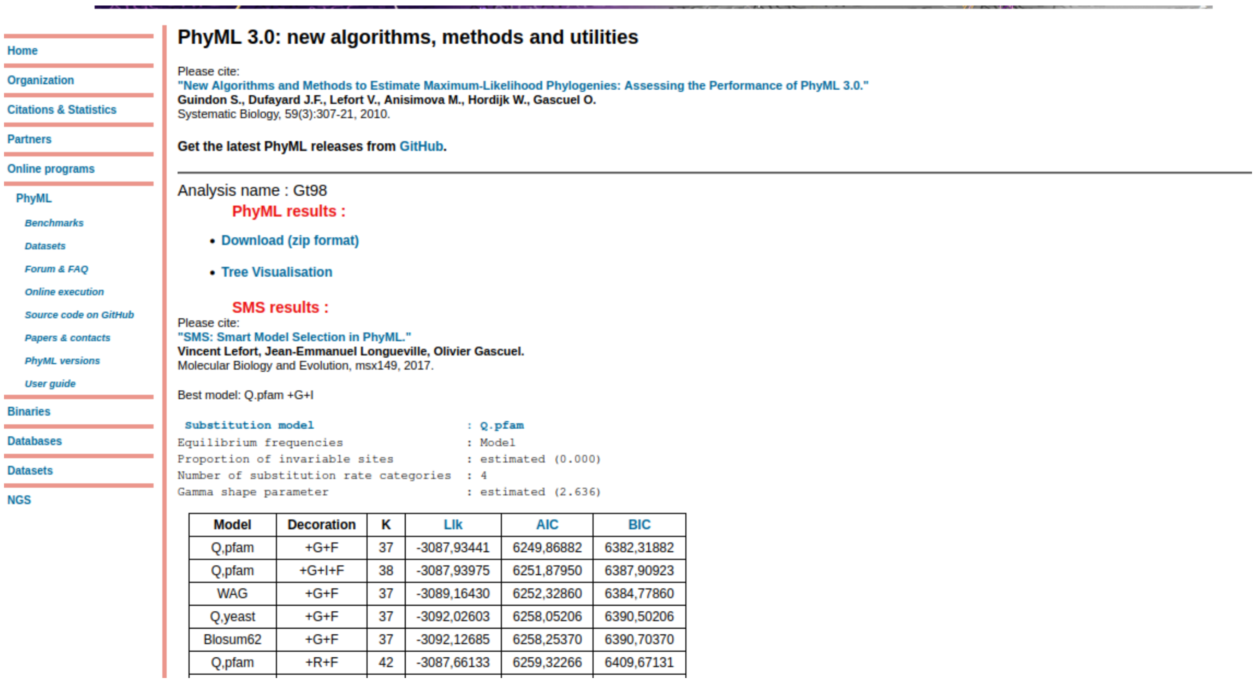
1. Highly conserved residues (scores 7–9) are represented as continuous dark blocks appear around columns ~40, ~72, and ~85, where most sequences show identical hydrophobic residues like L, A, F, and G. These positions correspond to structurally important regions common to all globins.
2. Moderately conserved positions (scores 4–6) were mixed but chemically similar residues (V, I, M) occur around columns ~25 and ~63, producing medium-height conservation bars and softer colour uniformity. These allow substitution without major structural disruption.
3. Low conservation regions (scores 0–3) were the N-terminal (~1–15) and C-terminal (~110–130) areas show frequent gaps, diverse colours, and short conservation bars. Columns such as ~7 and ~118 contain variable residues like S, N, and Q, indicating flexible or species-specific segments.


WP_447029335.1/1-134
WP_447006119.1/1-128
WP_446888964.1/1-154
YBV26126.1/1-133
YBV21917.1/1-139
BGO70610.1/1-130
BHH87803.1/1-153
CAO3294171.1/1-151
CAM5329635.1/1-134
YBU08153.1/1-130
Conservation
Quality
Consensus
Occupancy


WP_447029335.1/1-134
WP_447006119.1/1-128
WP_446888964.1/1-154
YBV26126.1/1-133
YBV21917.1/1-139
BGO70610.1/1-130
BHH87803.1/1-153
CAO3294171.1/1-151
CAM5329635.1/1-134
YBU08153.1/1-130
Conservation
Quality
Consensus
Occupancy


WP_447029335.1/1-134
WP_447006119.1/1-128
WP_446888964.1/1-154
YBV26126.1/1-133
YBV21917.1/1-139
BGO70610.1/1-130
BHH87803.1/1-153
CAO3294171.1/1-151
CAM5329635.1/1-134
YBU08153.1/1-130
Conservation
Quality
Consensus
Occupancy

Q.8. Use the same MSA to build a Maximum Likelihood tree using PhyML and 100 bootstraps. Use Smart Model Selection to determine the best fitting evolutionary model. Mention the model and its details such as model free parameters etc. Interpret this tree based on your understanding.



PhyML 3.0: new algorithms, methods and utilities

Please cite:
"New Algorithms and Methods to Estimate Maximum-Likelihood Phylogenies: Assessing the Performance of PhyML 3.0."
Guindon S., Dufayard J.F., Lefort V., Anisimova M., Hordijk W., Gascuel O.
Systematic Biology, 59(3):307-21, 2010.

Get the latest PhyML releases from GitHub.

Analysis name : Gt98

PhyML results :

- Download (zip format)
- Tree Visualisation

SMS results :

Please cite:
"SMS: Smart Model Selection in PhyML."
Vincent Lefort, Jean-Emmanuel Longueville, Olivier Gascuel.
Molecular Biology and Evolution, msx149, 2017.

Best model: Q.pfam +G+I

```
 Substitution model                       : Q.pfam
Equilibrium frequencies                   : Model
Proportion of invariable sites            : estimated (0.000)
Number of substitution rate categories    : 4
Gamma shape parameter                     : estimated (2.636)
```

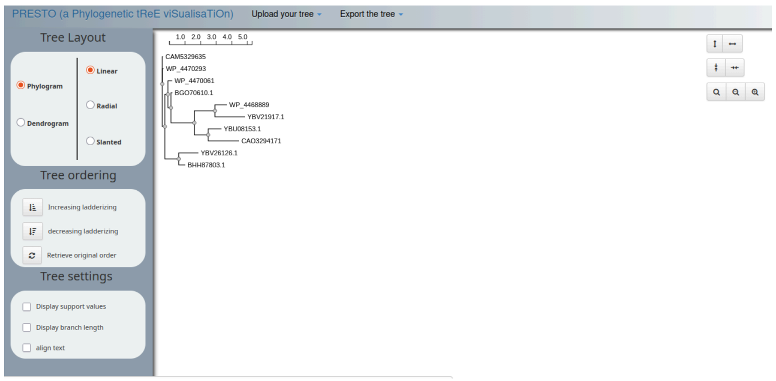
Model	Decoration	K	Llk	AIC	BIC
Q,pfam	+G+F	37	-3087,93441	6249,86882	6382,31882
Q,pfam	+G+I+F	38	-3087,93975	6251,87950	6387,90923
WAG	+G+F	37	-3089,16430	6252,32860	6384,77860
Q,yeast	+G+F	37	-3092,02603	6258,05206	6390,50206
Blosum62	+G+F	37	-3092,12685	6258,25370	6390,70370
Q,pfam	+R+F	42	-3087,66133	6259,32266	6409,67131

In this study, I used Smart Model Selection (SMS) to identify the most appropriate evolutionary model for my protein alignment. SMS selected Q.pfam + G + I because this empirical matrix is derived from thousands of Pfam protein families and therefore matches the substitution patterns expected in my dataset. The +G component indicates that sites evolve at different rates, and the estimated gamma value (2.636) reflects moderate rate variation across positions. Although +I was included, the model estimated the proportion of invariable sites as 0.000, showing that almost all alignment sites show evolutionary change. Overall, this model best fit the data by balancing high likelihood with appropriate complexity.

The model free parameters are the numerical values that PhyML must estimate directly from my alignment because they are not fixed in the substitution matrix. In this model, the free parameters include the gamma shape parameter (α = 2.636), the proportion of invariable sites (I = 0.000), and all branch lengths in the final tree. The gamma value indicates moderate variability across sites, while the zero invariable-site estimate means the alignment does not contain strongly conserved positions. These parameter values emerge naturally from the data and explain how much rate heterogeneity and evolutionary change are present in the sequences used in this study.

<u>Interpretation based on tree</u>

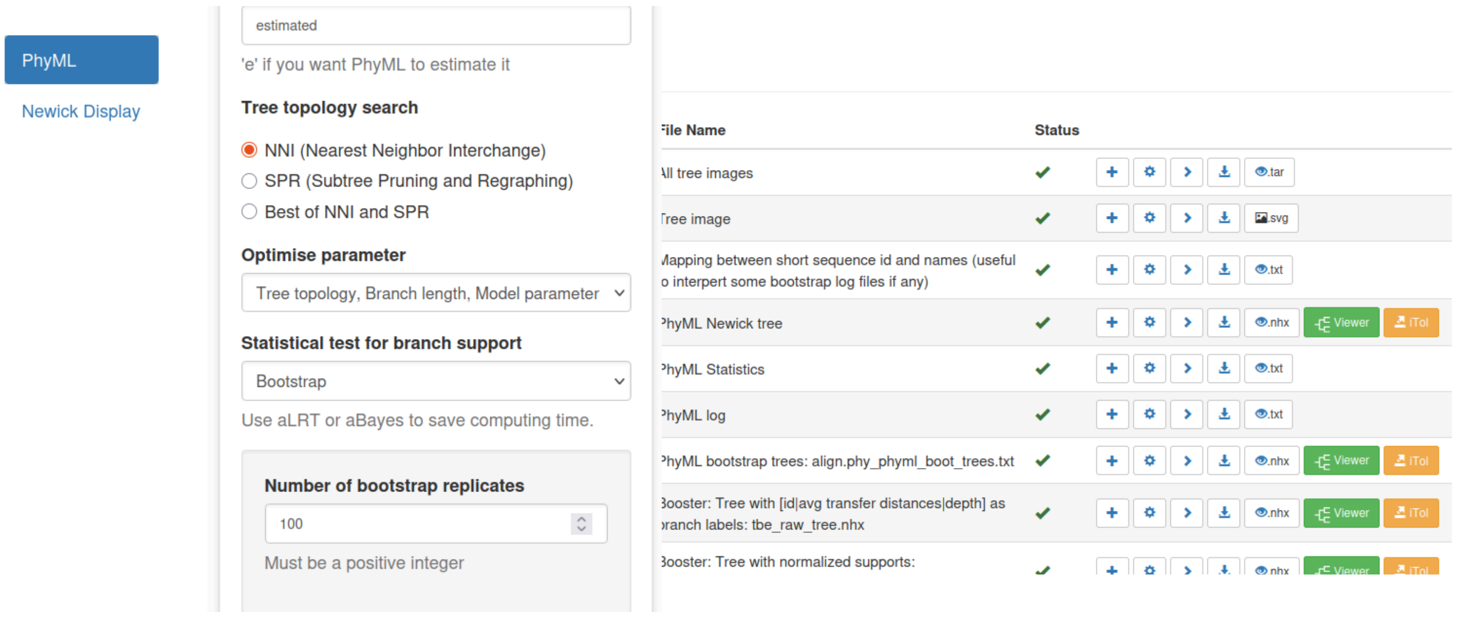
The phylogenetic tree shows how similar or different each of my protein sequences is from the others. The names on the right represent each sequence, and those that sit close together on the same branch are more alike, while those that sit farther apart have more differences. In this tree, the sequences naturally form two main groups: The first group includes CAM5329635, WP_4470293, WP_4470061, and BGO70610.1, which cluster very closely, indicating they share a strong evolutionary relationship and may perform similar functions. The second large group contains WP_4468889, YBV21917.1, YBU08153.1, CAO3294171, YBV26126.1, and BHH87803.1, forming a tight cluster that suggests a common origin and conserved roles within this subfamily. Inside this lower group, WP_4468889, YBV21917.1, YBU08153.1, and CAO3294171 appear especially close to each other, showing an even stronger level of similarity. YBV26126.1 and BHH87803.1 form another small pair within the same group, indicating they are more similar to each other compared to the rest. The large distance separating the two main clusters shows that the sequences belong to two distinct families, while the short branches within each cluster reflect strong relatedness.



[✓ Phylogram] Dendrogram

[✓ Linear] Radial Slanted

[ ] Display support values
[ ] Display branch length
[ ] align text



estimated

'e' if you want PhyML to estimate it

Tree topology search

[✓ NNI (Nearest Neighbor Interchange)]
SPR (Subtree Pruning and Regraphing)
Best of NNI and SPR

Optimise parameter

Tree topology, Branch length, Model parameter

Statistical test for branch support

Bootstrap

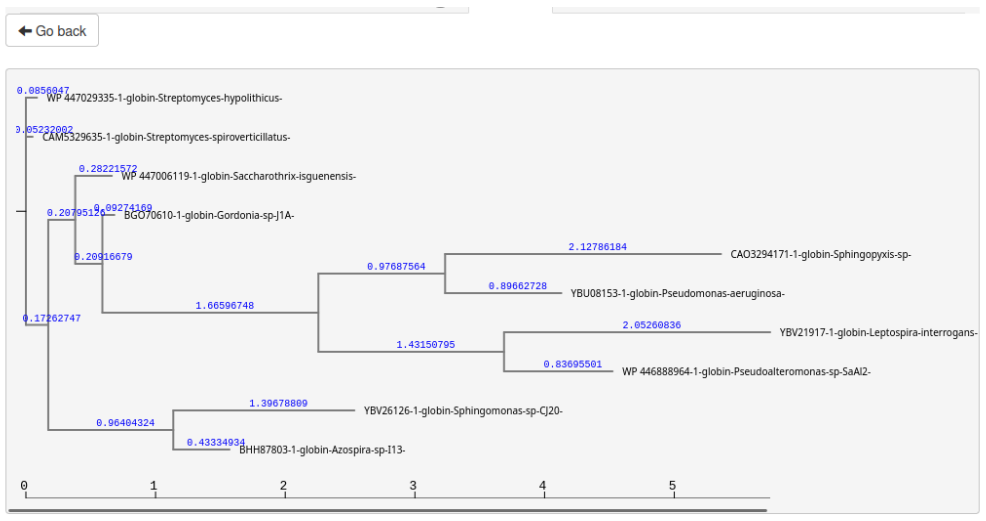
Use aLRT or aBayes to save computing time.

Number of bootstrap replicates

100

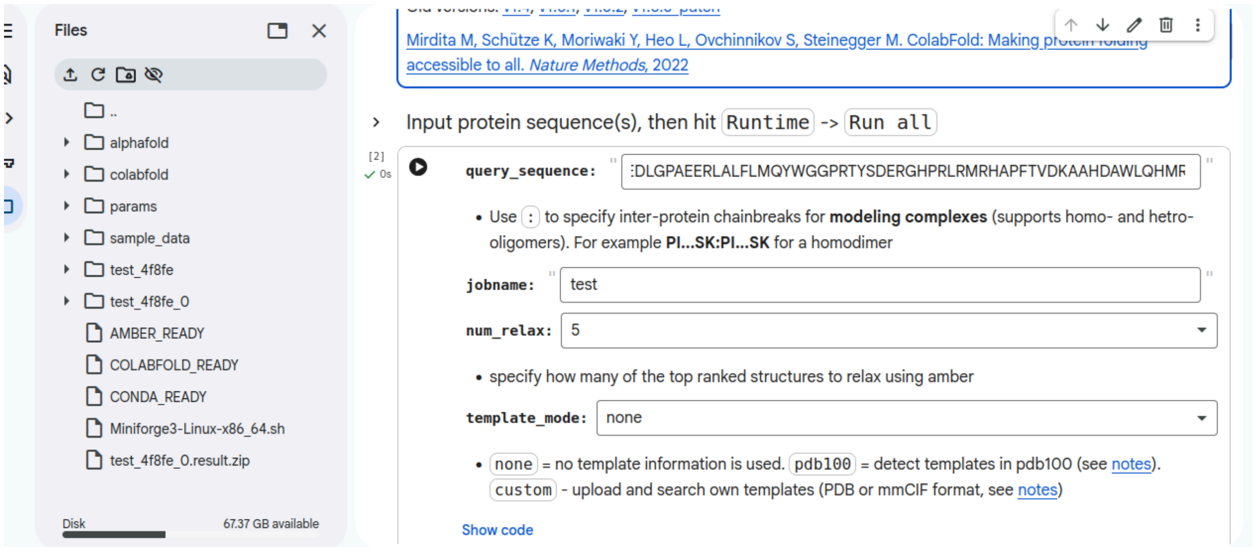
Must be a positive integer

ile Name	Status	
ll tree images	✓	.tar
ree image	✓	.svg
lapping between short sequence id and names (useful o interpret some bootstrap log files if any)	✓	.txt
hyML Newick tree	✓	.nhx Viewer iTol
hyML Statistics	✓	.txt
hyML log	✓	.txt
hyML bootstrap trees: align.phy_phyml_boot_trees.txt	✓	.nhx Viewer iTol
ooster: Tree with [id|avg transfer distances|depth] as ranch labels: tbe_raw_tree.nhx	✓	.nhx Viewer iTol
ooster: Tree with normalized supports:	✓	.nhx Viewer iTol


Go back
0.0856047
WP 447029335-1-globin-Streptomyces-hypolithicus-
0.05232002
CAM5329635-1-globin-Streptomyces-spiroverticillatus-
0.28221572
WP 447006119-1-globin-Saccharothrix-isguenensis-
0.20795126
0.09274169
BGO70610-1-globin-Gordonia-sp-J1A-
0.20916679
2.12786184
CAO3294171-1-globin-Sphingopyxis-sp-
0.97687564
0.89662728
YBU08153-1-globin-Pseudomonas-aeruginosa-
1.66596748
0.17262747
2.05260836
YBV21917-1-globin-Leptospira-interrogans-
1.43150795
0.83695501
WP 446888964-1-globin-Pseudoalteromonas-sp-SaAl2-
1.39678809
YBV26126-1-globin-Sphingomonas-sp-CJ20-
0.96404324
0.43334934
BHH87803-1-globin-Azospira-sp-I13-
0
1
2
3
4
5

In this study, I performed a Maximum Likelihood analysis with 100 bootstrap replicates to check how reliable each branch of my tree is. Bootstrapping resamples the alignment many times, and high values indicate strong support for the grouping of sequences. The PhyML statistics showed that the LG model was used, with a gamma shape parameter of 2.354 and no invariable sites, meaning my sequences show moderate rate variation and no fully conserved positions. Although SMS selected Q.pfam +G+I as the best model, it was not available in NGPhylogeny's PhyML model list, so I used the closest available empirical model (LG), which performs well for protein datasets. Both the PhyML output and the NGPhylogeny tree viewer display the same topology, but NGPhylogeny provides clearer branch lengths and full labels, making it easier to interpret the evolutionary relationships.

Q.9. Predict the tertiary structure of any two out of the above ten protein sequences using AlphaFold 2 or Alpha Fold 3. Visualise the structure and report the pLDDT as well as PAE of each structure. Write interpretation of the pLDDT and PAE plots.



Files

..
alphafold
colabfold
params
sample_data
test_4f8fe
test_4f8fe_0
AMBER_READY
COLABFOLD_READY
CONDA_READY
Miniforge3-Linux-x86_64.sh
test_4f8fe_0.result.zip

Disk 67.37 GB available

Mirdita M, Schütze K, Moriwaki Y, Heo L, Ovchinnikov S, Steinegger M. ColabFold: Making protein folding accessible to all. *Nature Methods*, 2022

Input protein sequence(s), then hit `Runtime` -> `Run all`

query_sequence: :DLGPAEERLALFLMQYWGGPRTYSDERGHPRLRMRHAPFTVDKAAHDAWLQHMR

- Use `:` to specify inter-protein chainbreaks for **modeling complexes** (supports homo- and hetro-oligomers). For example **PI...SK:PI...SK** for a homodimer

jobname: test

num_relax: 5

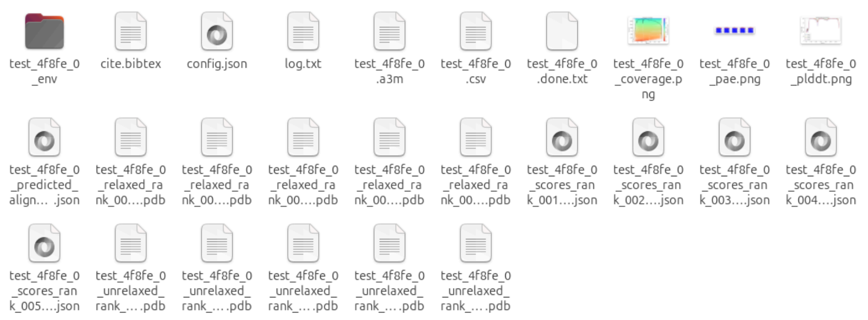
- specify how many of the top ranked structures to relax using amber

template_mode: none

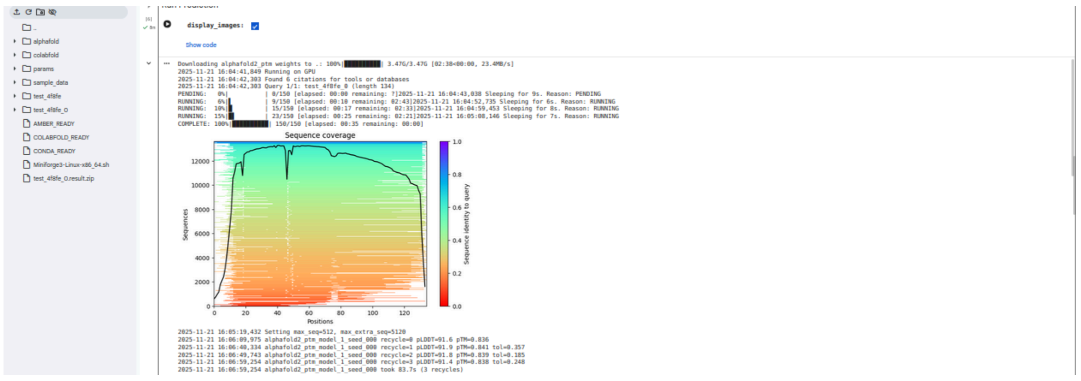
- `none` = no template information is used. `pdb100` = detect templates in pdb100 (see notes). `custom` - upload and search own templates (PDB or mmCIF format, see notes)

Show code

bfd.mgnify30.metaeuk30.sm... .a3m
msa.sh
out.tar.gz
pdb70.m8
uniref.a3m



test_4f8fe_0_env
cite.bibtex
config.json
log.txt
test_4f8fe_0.a3m
test_4f8fe_0.csv
test_4f8fe_0.done.txt
test_4f8fe_0_coverage.png
test_4f8fe_0_pae.png
test_4f8fe_0_plddt.png
test_4f8fe_0_predicted_align... .json
test_4f8fe_0_relaxed_rank_00....pdb
test_4f8fe_0_relaxed_rank_00....pdb
test_4f8fe_0_relaxed_rank_00....pdb
test_4f8fe_0_relaxed_rank_00....pdb
test_4f8fe_0_relaxed_rank_00....pdb
test_4f8fe_0_scores_rank_001....json
test_4f8fe_0_scores_rank_002....json
test_4f8fe_0_scores_rank_003....json
test_4f8fe_0_scores_rank_004....json
test_4f8fe_0_scores_rank_005....json
test_4f8fe_0_unrelaxed_rank_....pdb
test_4f8fe_0_unrelaxed_rank_....pdb
test_4f8fe_0_unrelaxed_rank_....pdb
test_4f8fe_0_unrelaxed_rank_....pdb
test_4f8fe_0_unrelaxed_rank_....pdb

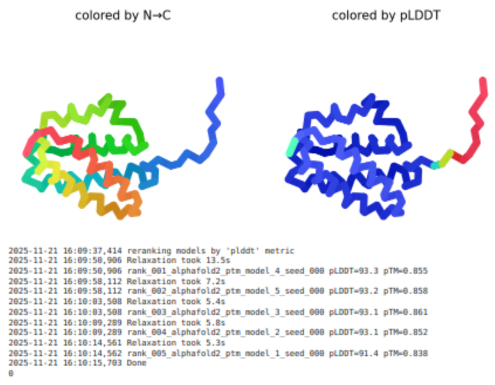


display_images: ✓

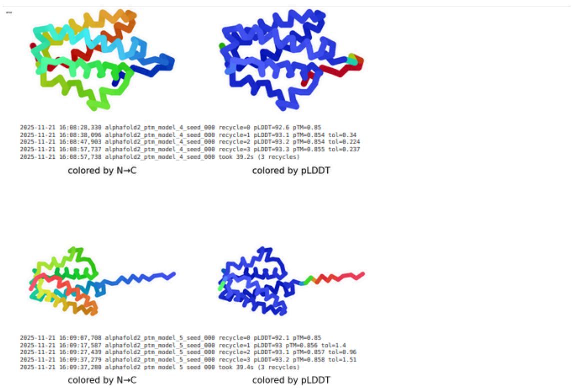
Show code

```
Downloading alphafold2_ptm weights to .: 100%|██████████| 3.47G/3.47G [02:38<00:00, 23.4MB/s]
2025-11-21 16:04:41,849 Running on GPU
2025-11-21 16:04:42,303 Found 6 citations for tools or databases
2025-11-21 16:04:42,303 Query 1/1: test_4f8fe_0 (length 134)
PENDING:   0%|          | 0/150 [elapsed: 00:00 remaining: ?]2025-11-21 16:04:43,038 Sleeping for 9s. Reason: PENDING
RUNNING:   6%|▌         | 9/150 [elapsed: 00:10 remaining: 02:43]2025-11-21 16:04:52,735 Sleeping for 6s. Reason: RUNNING
RUNNING:  10%|█         | 15/150 [elapsed: 00:17 remaining: 02:33]2025-11-21 16:04:59,453 Sleeping for 8s. Reason: RUNNING
RUNNING:  15%|█▌        | 23/150 [elapsed: 00:25 remaining: 02:21]2025-11-21 16:05:08,146 Sleeping for 7s. Reason: RUNNING
COMPLETE: 100%|██████████| 150/150 [elapsed: 00:35 remaining: 00:00]
```

```
2025-11-21 16:05:19,432 Setting max_seq=512, max_extra_seq=5120
2025-11-21 16:06:09,975 alphafold2_ptm_model_1_seed_000 recycle=0 pLDDT=91.6 pTM=0.836
2025-11-21 16:06:40,334 alphafold2_ptm_model_1_seed_000 recycle=1 pLDDT=91.9 pTM=0.841 tol=0.357
2025-11-21 16:06:49,743 alphafold2_ptm_model_1_seed_000 recycle=2 pLDDT=91.8 pTM=0.839 tol=0.185
2025-11-21 16:06:59,254 alphafold2_ptm_model_1_seed_000 recycle=3 pLDDT=91.4 pTM=0.838 tol=0.248
2025-11-21 16:06:59,254 alphafold2_ptm_model_1_seed_000 took 83.7s (3 recycles)
```

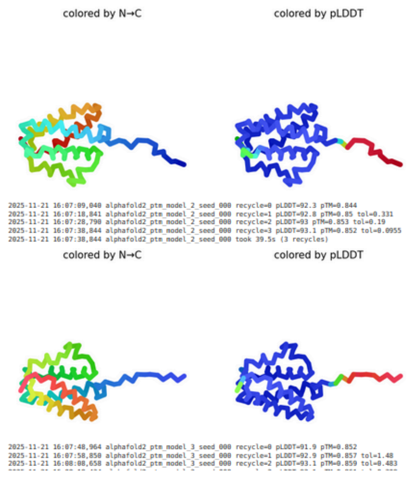


```
2025-11-21 16:09:37,414 reranking models by 'plddt' metric
2025-11-21 16:09:50,906 Relaxation took 13.5s
2025-11-21 16:09:50,906 rank_001_alphafold2_ptm_model_4_seed_000 pLDDT=93.3 pTM=0.855
2025-11-21 16:09:58,112 Relaxation took 7.2s
2025-11-21 16:09:58,112 rank_002_alphafold2_ptm_model_5_seed_000 pLDDT=93.2 pTM=0.858
2025-11-21 16:10:03,508 Relaxation took 5.4s
2025-11-21 16:10:03,508 rank_003_alphafold2_ptm_model_3_seed_000 pLDDT=93.1 pTM=0.861
2025-11-21 16:10:09,289 Relaxation took 5.8s
2025-11-21 16:10:09,289 rank_004_alphafold2_ptm_model_2_seed_000 pLDDT=93.1 pTM=0.852
2025-11-21 16:10:14,561 Relaxation took 5.3s
2025-11-21 16:10:14,562 rank_005_alphafold2_ptm_model_1_seed_000 pLDDT=91.4 pTM=0.838
2025-11-21 16:10:15,703 Done
0
```



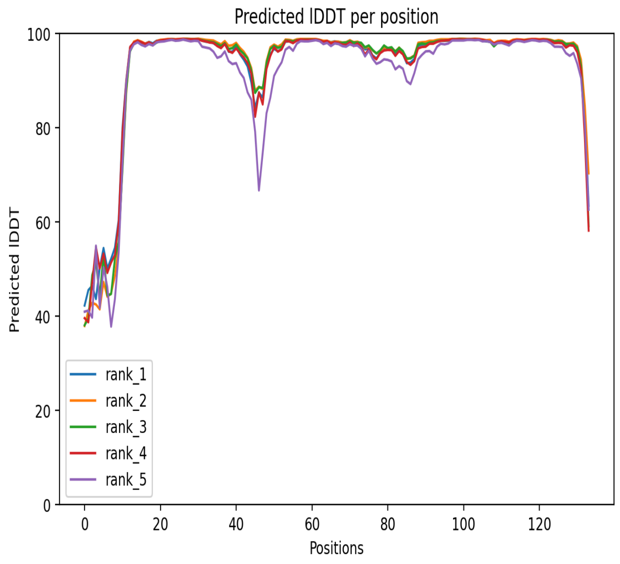
```
2025-11-21 16:08:28,330 alphafold2_ptm_model_4_seed_000 recycle=0 pLDDT=92.6 pTM=0.85
2025-11-21 16:08:38,096 alphafold2_ptm_model_4_seed_000 recycle=1 pLDDT=93.1 pTM=0.854 tol=0.34
2025-11-21 16:08:47,903 alphafold2_ptm_model_4_seed_000 recycle=2 pLDDT=93.2 pTM=0.854 tol=0.224
2025-11-21 16:08:57,737 alphafold2_ptm_model_4_seed_000 recycle=3 pLDDT=93.3 pTM=0.855 tol=0.237
2025-11-21 16:08:57,738 alphafold2_ptm_model_4_seed_000 took 39.2s (3 recycles)
```

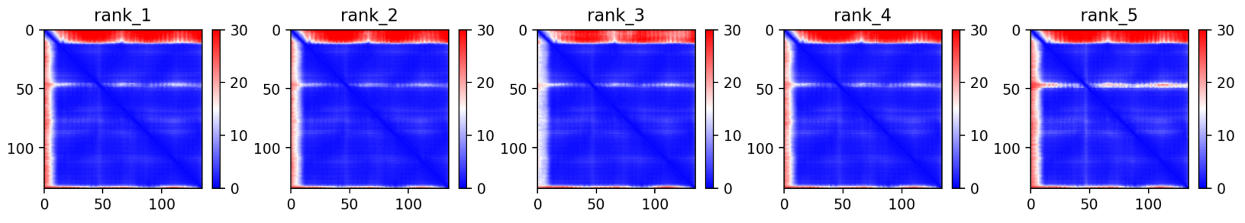
```
2025-11-21 16:09:07,708 alphafold2_ptm_model_5_seed_000 recycle=0 pLDDT=92.1 pTM=0.85
2025-11-21 16:09:17,587 alphafold2_ptm_model_5_seed_000 recycle=1 pLDDT=93 pTM=0.856 tol=1.4
2025-11-21 16:09:27,439 alphafold2_ptm_model_5_seed_000 recycle=2 pLDDT=93.1 pTM=0.857 tol=0.96
2025-11-21 16:09:37,279 alphafold2_ptm_model_5_seed_000 recycle=3 pLDDT=93.2 pTM=0.858 tol=1.51
2025-11-21 16:09:37,280 alphafold2 ptm model 5 seed 000 took 39.4s (3 recycles)
```



```
2025-11-21 16:07:09,040 alphafold2_ptm_model_2_seed_000 recycle=0 pLDDT=92.3 pTM=0.844
2025-11-21 16:07:18,841 alphafold2_ptm_model_2_seed_000 recycle=1 pLDDT=92.8 pTM=0.85 tol=0.331
2025-11-21 16:07:28,790 alphafold2_ptm_model_2_seed_000 recycle=2 pLDDT=93 pTM=0.853 tol=0.19
2025-11-21 16:07:38,844 alphafold2_ptm_model_2_seed_000 recycle=3 pLDDT=93.1 pTM=0.852 tol=0.0955
2025-11-21 16:07:38,844 alphafold2_ptm_model_2_seed_000 took 39.5s (3 recycles)
```

```
2025-11-21 16:07:48,964 alphafold2_ptm_model_3_seed_000 recycle=0 pLDDT=91.9 pTM=0.852
2025-11-21 16:07:58,850 alphafold2_ptm_model_3_seed_000 recycle=1 pLDDT=92.9 pTM=0.857 tol=1.48
2025-11-21 16:08:08,658 alphafold2_ptm_model_3_seed_000 recycle=2 pLDDT=93.1 pTM=0.859 tol=0.483
```



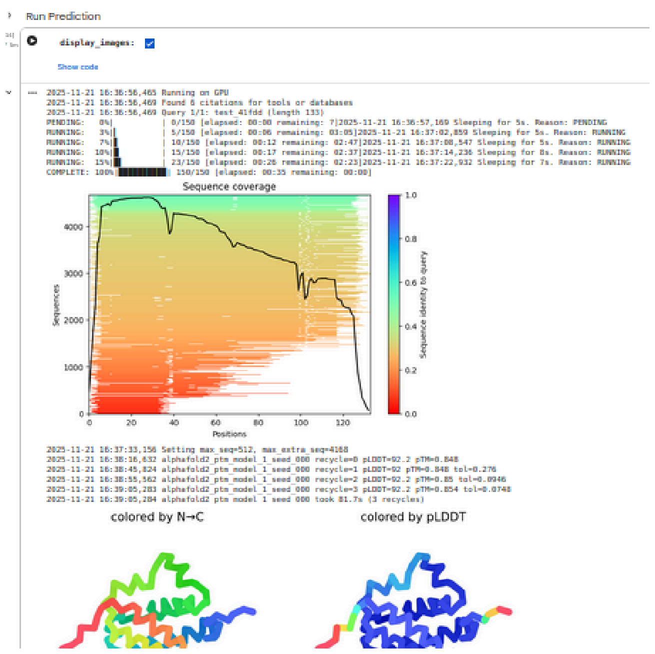


For 1st sequence

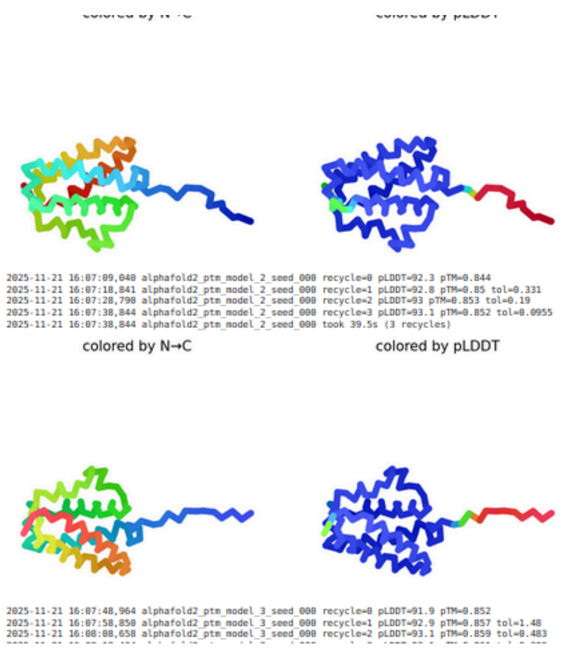
- I selected two sequences from the ten organisms used in my MSA and phylogenetic tree.
- I chose WP_447029335.1 (Streptomyces hypolithicus) and YBV26126.1 (Sphingomonas sp. CJ20).
- I selected these because they belonged to different bacterial groups and were complete, good-quality sequences.
- I extracted both sequences from the aligned MSA file.
- I removed alignment gaps ("–") to obtain the original ungapped amino acid sequences.
- I prepared both sequences in clean FASTA format.
- I used AlphaFold2 (ColabFold) to predict the 3D structures.
- I pasted one sequence at a time into the query_sequence input box.
- I set num_relax = 5 and kept template mode = none.
- I executed the notebook using Runtime → Run all.
- I downloaded all AlphaFold-generated outputs (PDB, pLDDT plot, PAE plot, confidence images).
- I manually saved the N→C colored structure images from the notebook.
- I visualized each predicted structure using Mol* online

Same I did with second sequence. Interpretation is at last

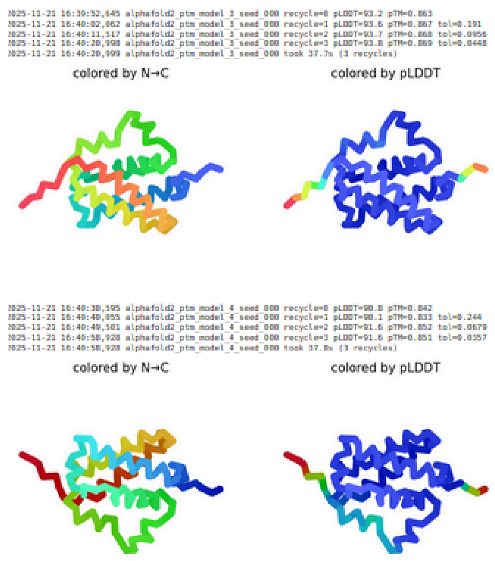
For 2nd sequence:



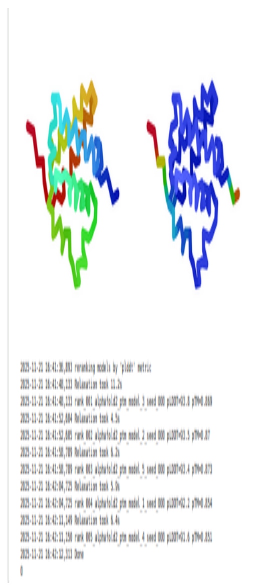
```
2025-11-21 16:36:56,465 Running on GPU
2025-11-21 16:36:56,469 Found 6 citations for tools or databases
2025-11-21 16:36:56,469 Query 1/1: test_41fdd (length 133)
PENDING:   0%|          | 0/150 [elapsed: 00:00 remaining: ?]2025-11-21 16:36:57,169 Sleeping for 5s. Reason: PENDING
RUNNING:   3%|▎         | 5/150 [elapsed: 00:06 remaining: 03:05]2025-11-21 16:37:02,859 Sleeping for 5s. Reason: RUNNING
RUNNING:   7%|▋         | 10/150 [elapsed: 00:12 remaining: 02:47]2025-11-21 16:37:08,547 Sleeping for 5s. Reason: RUNNING
RUNNING:  10%|█         | 15/150 [elapsed: 00:17 remaining: 02:37]2025-11-21 16:37:14,236 Sleeping for 8s. Reason: RUNNING
RUNNING:  15%|█▌        | 23/150 [elapsed: 00:26 remaining: 02:23]2025-11-21 16:37:22,932 Sleeping for 7s. Reason: RUNNING
COMPLETE: 100%|██████████| 150/150 [elapsed: 00:35 remaining: 00:00]

2025-11-21 16:37:33,156 Setting max_seq=512, max_extra_seq=4168
2025-11-21 16:38:16,632 alphafold2_ptm_model_1_seed_000 recycle=0 pLDDT=92.2 pTM=0.848
2025-11-21 16:38:45,824 alphafold2_ptm_model_1_seed_000 recycle=1 pLDDT=92 pTM=0.848 tol=0.276
2025-11-21 16:38:55,562 alphafold2_ptm_model_1_seed_000 recycle=2 pLDDT=92.2 pTM=0.85 tol=0.0946
2025-11-21 16:39:05,283 alphafold2_ptm_model_1_seed_000 recycle=3 pLDDT=92.2 pTM=0.854 tol=0.0748
2025-11-21 16:39:05,284 alphafold2_ptm_model_1_seed_000 took 81.7s (3 recycles)
```



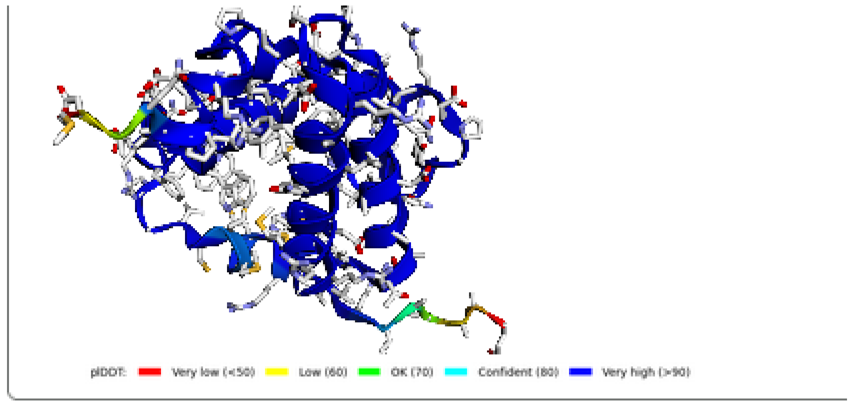
```
2025-11-21 16:07:09,040 alphafold2_ptm_model_2_seed_000 recycle=0 pLDDT=92.3 pTM=0.844
2025-11-21 16:07:18,841 alphafold2_ptm_model_2_seed_000 recycle=1 pLDDT=92.8 pTM=0.85 tol=0.331
2025-11-21 16:07:28,790 alphafold2_ptm_model_2_seed_000 recycle=2 pLDDT=93 pTM=0.853 tol=0.19
2025-11-21 16:07:38,844 alphafold2_ptm_model_2_seed_000 recycle=3 pLDDT=93.1 pTM=0.852 tol=0.0955
2025-11-21 16:07:38,844 alphafold2_ptm_model_2_seed_000 took 39.5s (3 recycles)

2025-11-21 16:07:48,964 alphafold2_ptm_model_3_seed_000 recycle=0 pLDDT=91.9 pTM=0.852
2025-11-21 16:07:58,850 alphafold2_ptm_model_3_seed_000 recycle=1 pLDDT=92.9 pTM=0.857 tol=1.48
2025-11-21 16:08:08,658 alphafold2_ptm_model_3_seed_000 recycle=2 pLDDT=93.1 pTM=0.859 tol=0.483
```

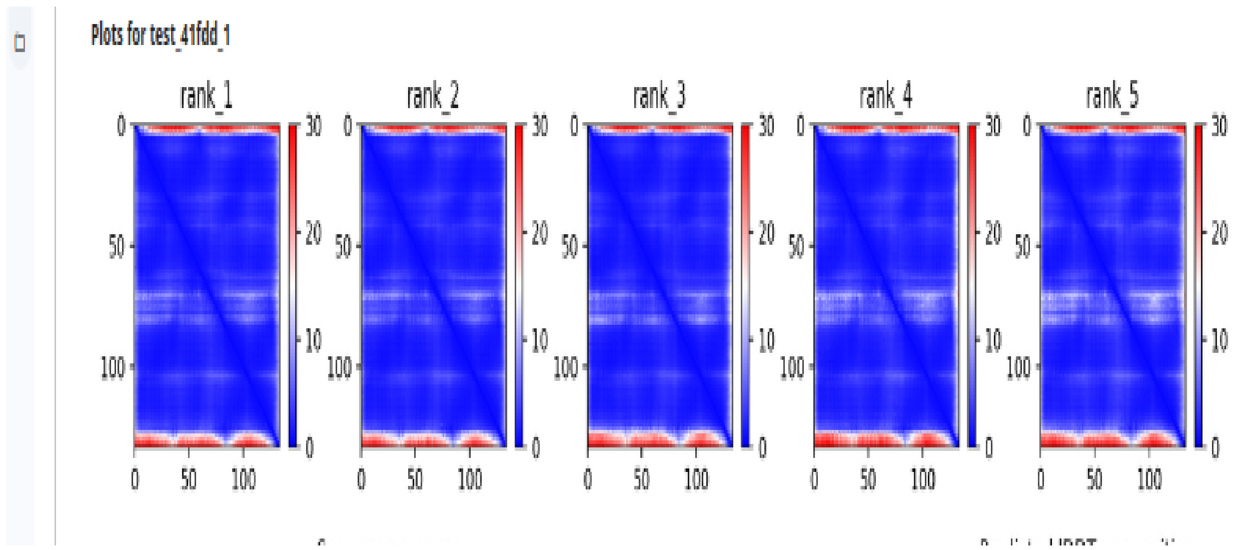


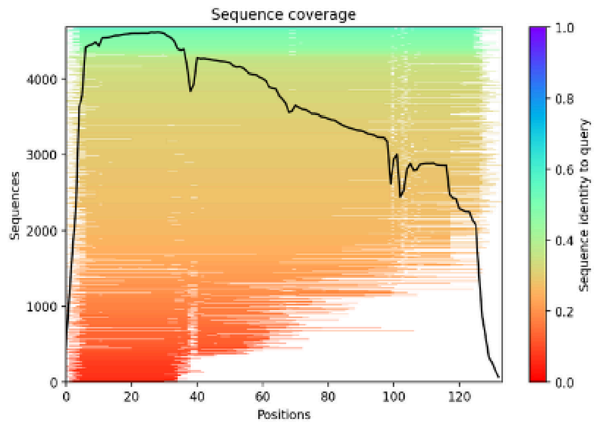
```
2025-11-21 16:39:52,645 alphafold2_ptm_model_3_seed_000 recycle=0 pLDDT=93.2 pTM=0.863
2025-11-21 16:40:02,062 alphafold2_ptm_model_3_seed_000 recycle=1 pLDDT=93.6 pTM=0.867 tol=0.191
2025-11-21 16:40:11,517 alphafold2_ptm_model_3_seed_000 recycle=2 pLDDT=93.7 pTM=0.868 tol=0.0956
2025-11-21 16:40:20,998 alphafold2_ptm_model_3_seed_000 recycle=3 pLDDT=93.8 pTM=0.869 tol=0.0448
2025-11-21 16:40:20,999 alphafold2_ptm_model_3_seed_000 took 37.7s (3 recycles)

2025-11-21 16:40:30,595 alphafold2_ptm_model_4_seed_000 recycle=0 pLDDT=90.8 pTM=0.842
2025-11-21 16:40:40,055 alphafold2_ptm_model_4_seed_000 recycle=1 pLDDT=90.1 pTM=0.833 tol=0.244
2025-11-21 16:40:49,501 alphafold2_ptm_model_4_seed_000 recycle=2 pLDDT=91.6 pTM=0.852 tol=0.0679
2025-11-21 16:40:58,928 alphafold2_ptm_model_4_seed_000 recycle=3 pLDDT=91.6 pTM=0.851 tol=0.0357
2025-11-21 16:40:58,928 alphafold2_ptm_model_4_seed_000 took 37.8s (3 recycles)
```

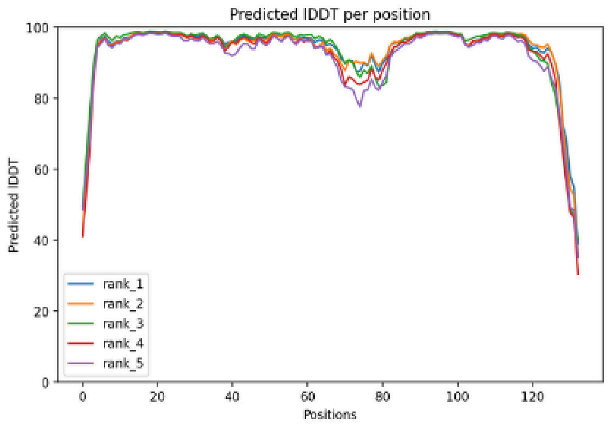


```
2025-11-21 16:41:36,803 reranking models by 'plddt' metric
2025-11-21 16:41:48,133 Relaxation took 11.2s
2025-11-21 16:41:48,133 rank_001_alphafold2_ptm_model_3_seed_000 pLDDT=93.8 pTM=0.869
2025-11-21 16:41:52,604 Relaxation took 4.5s
2025-11-21 16:41:52,605 rank_002_alphafold2_ptm_model_2_seed_000 pLDDT=93.5 pTM=0.87
2025-11-21 16:41:58,709 Relaxation took 6.2s
2025-11-21 16:41:58,709 rank_003_alphafold2_ptm_model_5_seed_000 pLDDT=93.4 pTM=0.873
2025-11-21 16:42:04,725 Relaxation took 5.9s
2025-11-21 16:42:04,725 rank_004_alphafold2_ptm_model_1_seed_000 pLDDT=92.2 pTM=0.854
2025-11-21 16:42:11,149 Relaxation took 6.4s
2025-11-21 16:42:11,150 rank_005_alphafold2_ptm_model_4_seed_000 pLDDT=91.6 pTM=0.851
2025-11-21 16:42:12,313 Done
0
```


pIDDT: 🟥 Very low (<50) 🟨 Low (60) 🟩 OK (70) 🟦 Confident (80) 🟦 Very high (>90)


Plots for test_41fdd_1
rank_1
rank_2
rank_3
rank_4
rank_5
0
50
100
0
10
20
30


Sequence coverage
Sequences
Positions
Sequence identity to query
0
1000
2000
3000
4000
20
40
60
80
100
120
0.0
0.2
0.4
0.6
0.8
1.0


Predicted lDDT per position
Predicted lDDT
Positions
rank_1
rank_2
rank_3
rank_4
rank_5
0
20
40
60
80
100
120

For 1st sequence's Interpretation

1. PAE (Predicted Aligned Error) Observation

Across all five ranked models, the PAE heatmaps are mostly dark blue, showing that AlphaFold is confident about the overall packing of the protein. The top horizontal red band corresponds to the N-terminal region, indicating flexible or poorly oriented ends. A thin light-colored band in the middle is seen in every model, marking one internal region whose orientation is less certain. Apart from these two flexible stretches, the rest of the protein behaves like a single compact and well-defined domain.

2. pLDDT / IDDT Observation

The pLDDT/IDDT line plot shows very high confidence (~90–100) across the central structured region. Both termini display low confidence (~40–60), consistent with flexible ends. A clear confidence drop in the middle appears across all five models, showing a loop or variable segment. The remaining residues are predicted with strong reliability, indicating well-formed helices/sheets in the core.

3. Sequence Coverage Observation

The coverage plot shows very high MSA depth across most positions, explaining the overall high model confidence. Two sharp dips in coverage align exactly with the central low-confidence region seen in pLDDT/IDDT, indicating evolutionary variability at that segment. The ends also have lower coverage, consistent with the terminal flexibility observed in both PAE and pLDDT plots.

For 2nd seq:

pLDDT (Predicted IDDT per position)

- Most residues show high pLDDT (~90–100), suggesting the model may be reliable in these regions.
- A small dip around the middle positions may indicate a flexible loop or locally uncertain region.
- Both termini show lower confidence, which may be typical for flexible ends.

PAE (Predicted Aligned Error)

- The PAE maps are mostly dark blue, suggesting low predicted error and a stable single-domain fold.
- A thin red strip at the top and bottom may correspond to flexible terminal regions.
- No large red blocks, indicating no major uncertainty between structural regions.

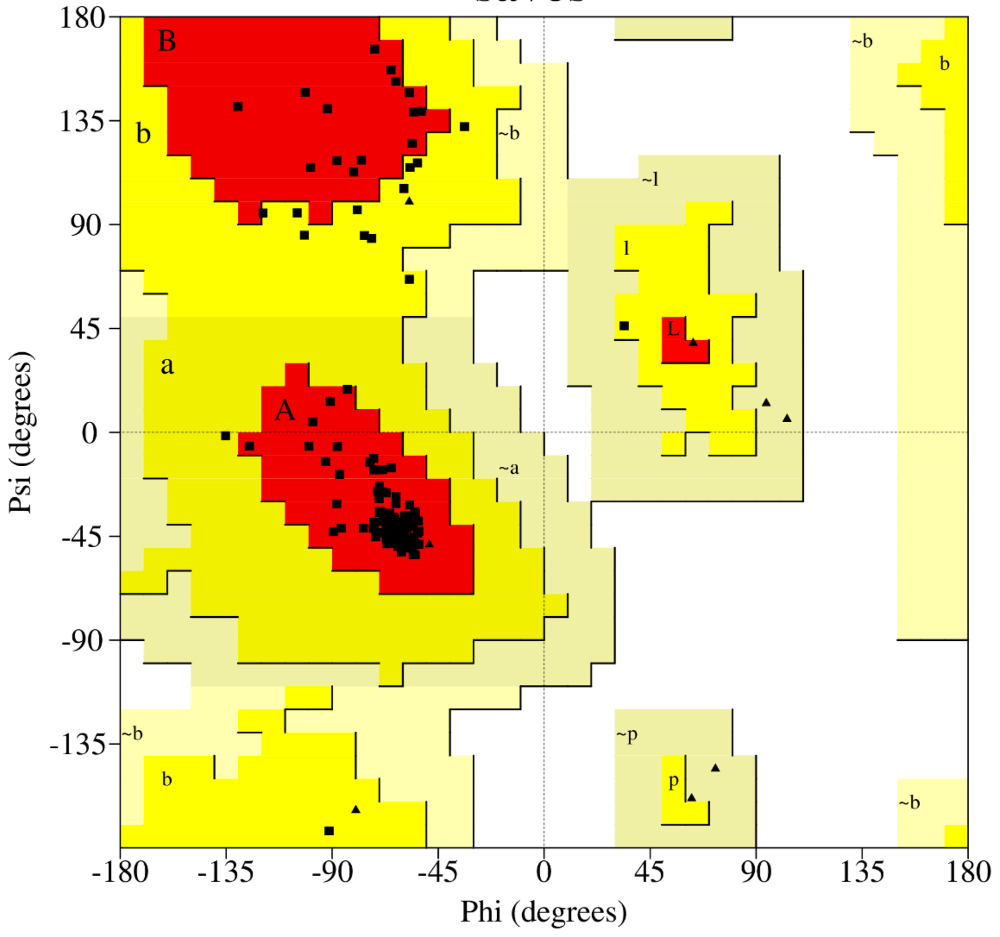
Sequence Coverage

- Coverage is high for most of the sequence, suggesting strong MSA depth and good evolutionary support.
- Small dips indicate positions where fewer homologs align, which may correspond to the same flexible/low-confidence regions seen in pLDDT.

PROCHECK

Ramachandran Plot

saves



Plot statistics

```
Residues in most favoured regions  [A,B,L]                102     88.7%
Residues in additional allowed regions  [a,b,l,p]          13     11.3%
Residues in generously allowed regions  [~a,~b,~l,~p]       0      0.0%
Residues in disallowed regions                              0      0.0%
                                                          ----   ------
Number of non-glycine and non-proline residues            115    100.0%

Number of end-residues (excl. Gly and Pro)                  1

Number of glycine residues (shown as triangles)            10
Number of proline residues                                  8
                                                          ----
Total number of residues                                  134
```

Based on an analysis of 118 structures of resolution of at least 2.0 Angstroms and R-factor no greater than 20%, a good quality model would be expected to have over 90% in the most favoured regions.

saves_01.ps

PROCHECK Ramachandran plot shows that 88.7% of the residues fall in the most-favoured regions (A, B, L), while 11.3% lie in the additionally allowed regions, and 0% appear in generously allowed or disallowed regions. This distribution indicates that the backbone φ–ψ angles of the model are highly acceptable, with no problematic outliers. The clustering of residues in the red α-helix and β-sheet regions reflects good stereochemical geometry, while the absence of disallowed residues confirms that the model does not contain strained or energetically unfavourable conformations. Overall, these PROCHECK statistics demonstrate that the predicted AlphaFold model has good structural quality and reliable backbone conformations.