COMPARATIVE ANALYSIS OF ACETYLCHOLINESTERASE (ACHE) USING BIOPYTHON: CONSERVED MOTIFS AND PHYLOGENETIC INSIGHTS

Muskan Kashyap (MSc Biotechnology)

WHAT IS THIS PROJECT ABOUT?

- A **bioinformatics analysis** of the enzyme Acetylcholinesterase (AChE).
- Used **Biopython** to study AChE sequences from different species.
- Identified **conserved motifs** (functionally important regions).
- Constructed a **phylogenetic tree** to understand evolutionary relationships.

WHY ACHE?

- AChE plays a critical role in **nerve signal transmission**.
- Inhibiting AChE is a key strategy in **Alzheimer's disease drug design**.
- Understanding conserved regions can help in **drug target identification**.

OBJECTIVES :

- To **collect acetylcholinesterase (AChE) protein sequences** from multiple species.
- To perform **multiple sequence alignment (MSA)** using Biopython.
- To **identify conserved motifs/regions** important for enzyme function.
- To **construct a phylogenetic tree** and explore evolutionary relationships.
- To demonstrate how **Biopython can be used** for real-world bioinformatics research.

Database → FASTA sequences → Alignment → Conserved motifs → Phylogenetic tree

BACKGROUND: ACETYLCHOLINESTERASE (ACHE) AND ALZHEIMER'S DISEASE

- **Acetylcholinesterase (AChE):**
  -Key enzyme that breaks down the neurotransmitter **acetylcholine** at nerve synapses.
  -Essential for proper **nerve signal transmission**.

- **Relevance to Alzheimer's disease (AD):**
  -In AD, acetylcholine levels drop, impairing memory and cognition.
  -**AChE inhibitors** are widely used as drugs (e.g., Donepezil, Rivastigmine).
  -Conserved regions in AChE may serve as **drug target sites**.

- **Why compare sequences across species?**
  -Conserved motifs → indicate **functionally important regions**.
  -Phylogenetic analysis → reveals **evolutionary relationships**.

TOOLS & TECHNOLOGIES :

FASTA Sequences → Biopython → Alignment (Clustal) → Tree (Matplotlib)

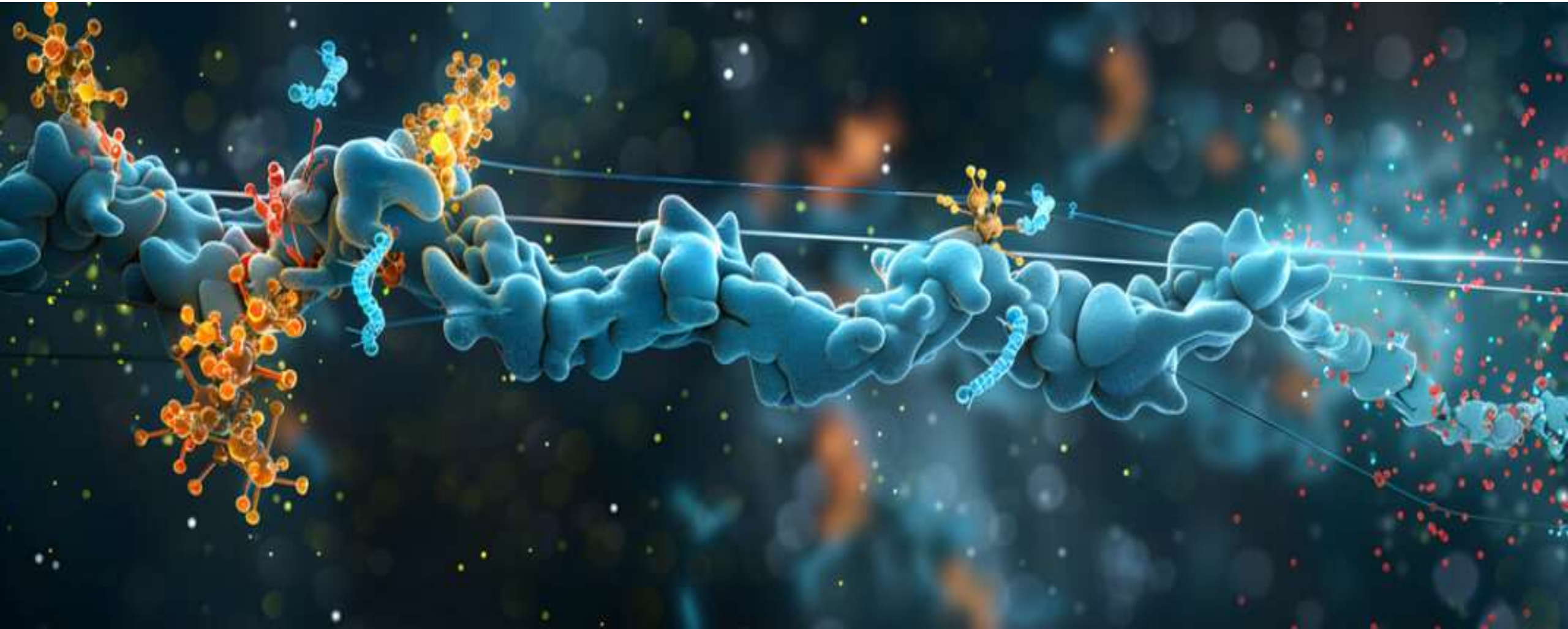
- **Programming Language**: Python

- **Libraries & Frameworks**:
  Biopython → Sequence handling, alignment, phylogenetic tree construction
  Matplotlib → Tree visualization

- **Alignment Tool**:
  Clustal Omega (via Biopython wrapper) → Multiple sequence alignment (MSA)

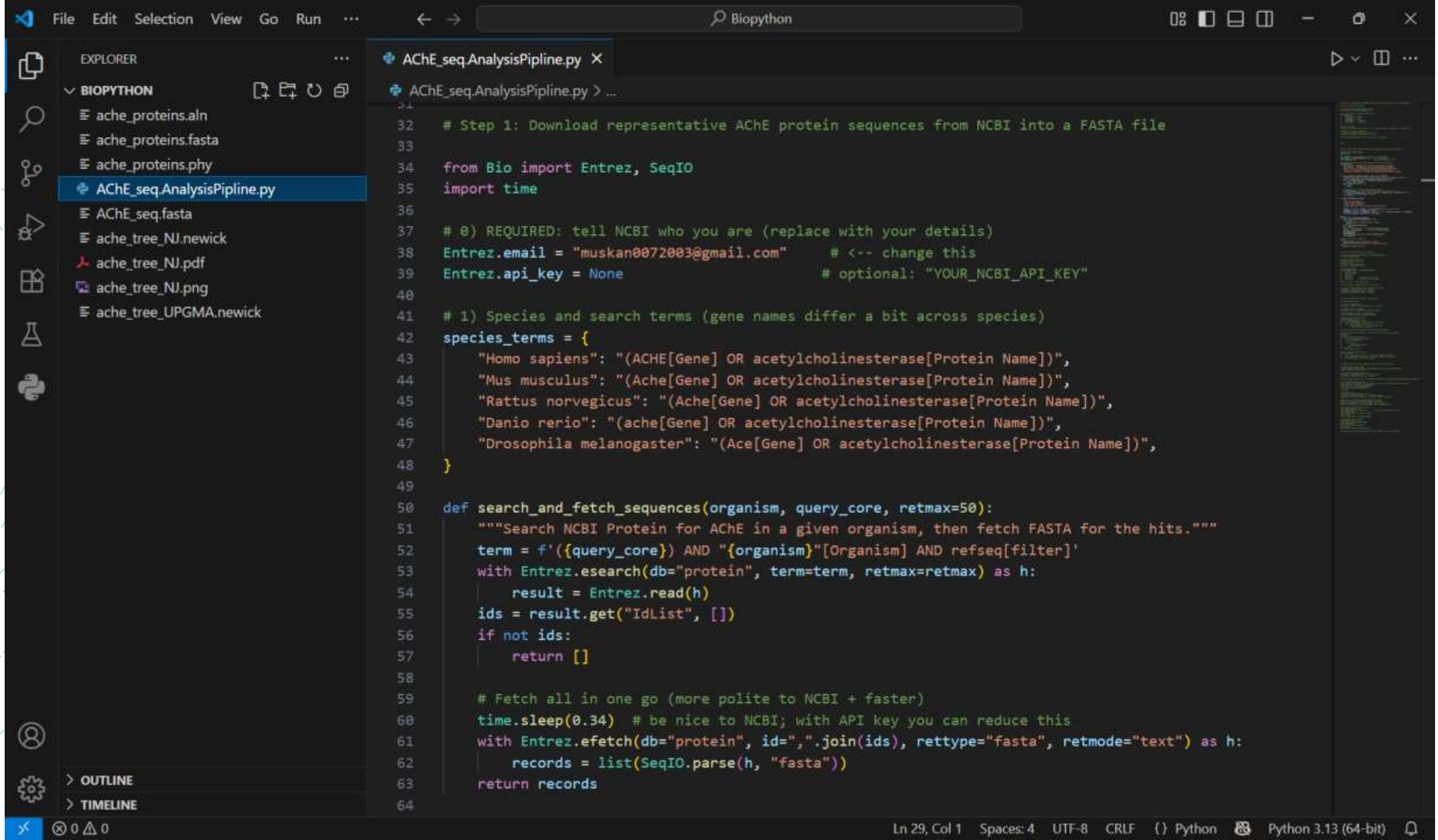
- **Data Source**:
  Protein sequences retrieved in **FASTA format** (from UniProt/NCBI)

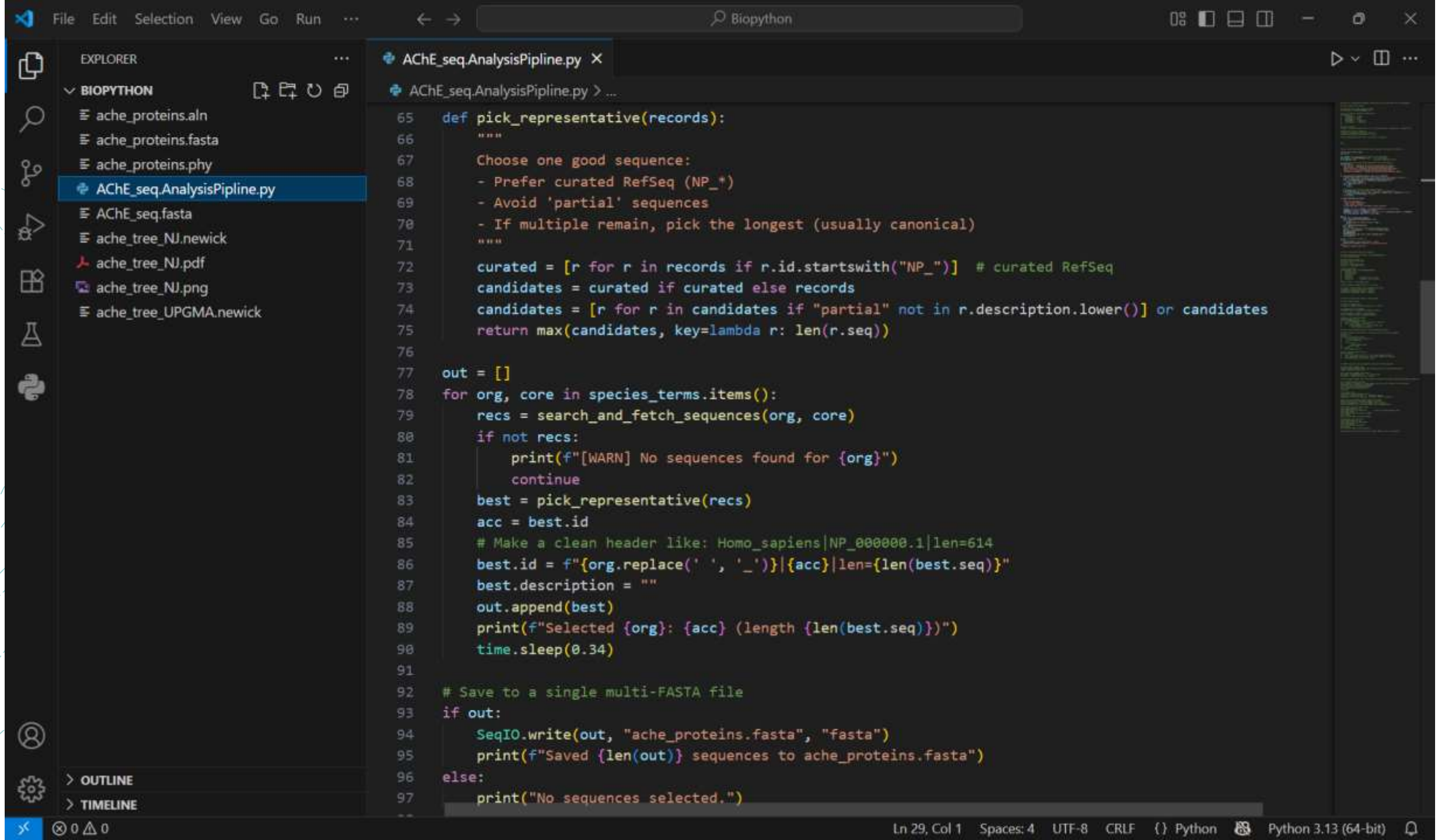
METHODOLOGY: STEP-BY-STEP PROCESS

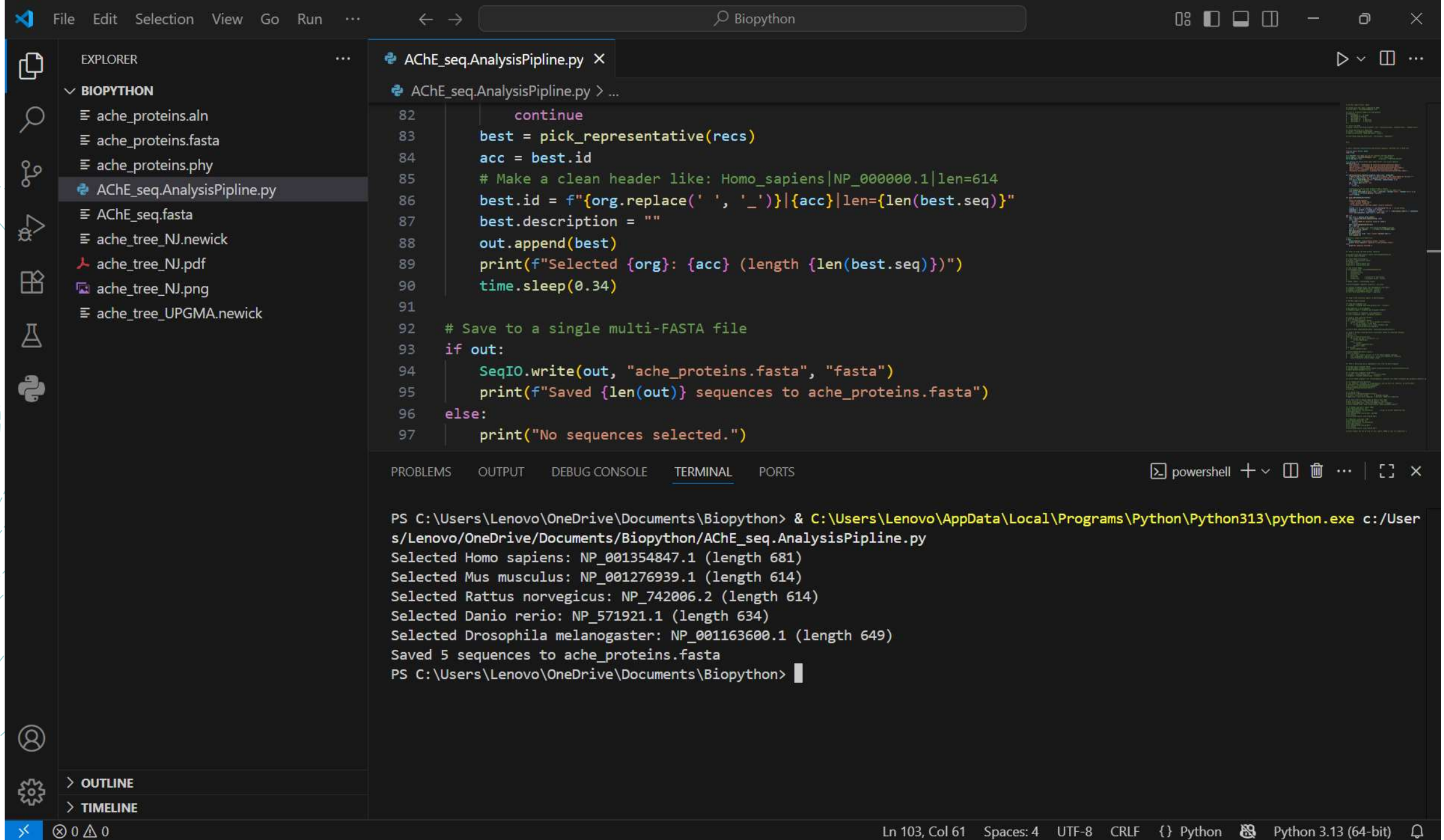


STEP 1: SEQUENCE COLLECTION

- **Collected Acetylcholinesterase (AChE) protein sequences from multiple species (Human, Mouse, Rat, Zebrafish, Fruit fly).**
- **Downloaded sequences in FASTA format from UniProt/NCBI databases.**
- **FASTA format = simple text file containing sequence name + amino acid sequence.**


File Edit Selection View Go Run ···
Biopython
EXPLORER
BIOPYTHON
ache_proteins.aln
ache_proteins.fasta
ache_proteins.phy
AChE_seq.AnalysisPipline.py
AChE_seq.fasta
ache_tree_NJ.newick
ache_tree_NJ.pdf
ache_tree_NJ.png
ache_tree_UPGMA.newick
OUTLINE
TIMELINE
AChE_seq.AnalysisPipline.py
AChE_seq.AnalysisPipline.py > ...
# Step 1: Download representative AChE protein sequences from NCBI into a FASTA file
from Bio import Entrez, SeqIO
import time
# 0) REQUIRED: tell NCBI who you are (replace with your details)
Entrez.email = "muskan0072003@gmail.com" # <-- change this
Entrez.api_key = None # optional: "YOUR_NCBI_API_KEY"
# 1) Species and search terms (gene names differ a bit across species)
species_terms = {
"Homo sapiens": "(ACHE[Gene] OR acetylcholinesterase[Protein Name])",
"Mus musculus": "(Ache[Gene] OR acetylcholinesterase[Protein Name])",
"Rattus norvegicus": "(Ache[Gene] OR acetylcholinesterase[Protein Name])",
"Danio rerio": "(ache[Gene] OR acetylcholinesterase[Protein Name])",
"Drosophila melanogaster": "(Ace[Gene] OR acetylcholinesterase[Protein Name])",
}
def search_and_fetch_sequences(organism, query_core, retmax=50):
"""Search NCBI Protein for AChE in a given organism, then fetch FASTA for the hits."""
term = f'({query_core}) AND "{organism}"[Organism] AND refseq[filter]'
with Entrez.esearch(db="protein", term=term, retmax=retmax) as h:
result = Entrez.read(h)
ids = result.get("IdList", [])
if not ids:
return []
Fetch all in one go (more polite to NCBI + faster)
time.sleep(0.34) # be nice to NCBI; with API key you can reduce this
with Entrez.efetch(db="protein", id=",".join(ids), rettype="fasta", retmode="text") as h:
records = list(SeqIO.parse(h, "fasta"))
return records
⊗ 0 ⚠ 0
Ln 29, Col 1 Spaces: 4 UTF-8 CRLF { } Python Python 3.13 (64-bit)

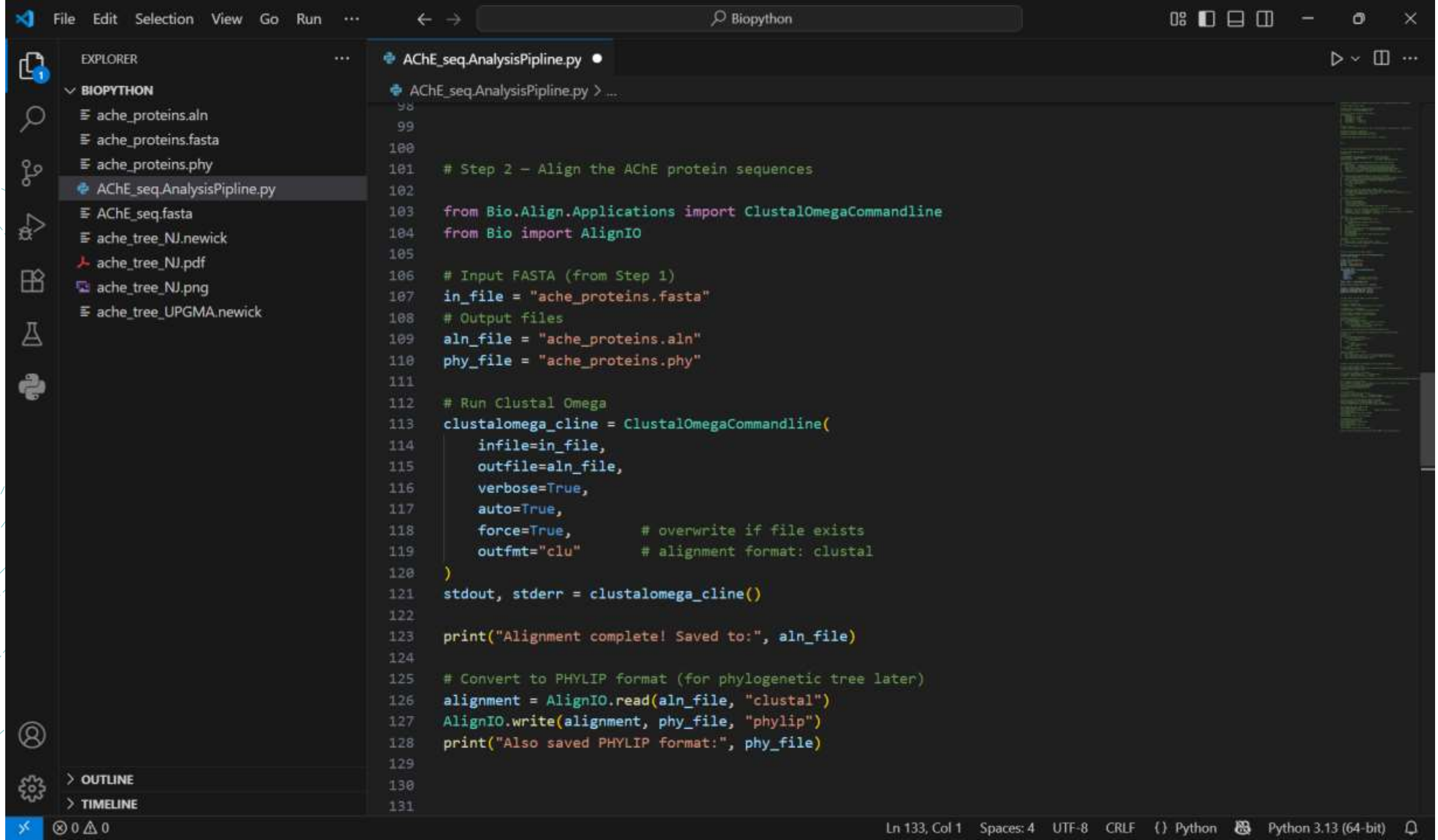

File Edit Selection View Go Run
Biopython
EXPLORER
BIOPYTHON
ache_proteins.aln
ache_proteins.fasta
ache_proteins.phy
AChE_seq.AnalysisPipline.py
AChE_seq.fasta
ache_tree_NJ.newick
ache_tree_NJ.pdf
ache_tree_NJ.png
ache_tree_UPGMA.newick
OUTLINE
TIMELINE
AChE_seq.AnalysisPipline.py
AChE_seq.AnalysisPipline.py > ...
def pick_representative(records):
"""
Choose one good sequence:
- Prefer curated RefSeq (NP_*)
- Avoid 'partial' sequences
- If multiple remain, pick the longest (usually canonical)
"""
curated = [r for r in records if r.id.startswith("NP_")] # curated RefSeq
candidates = curated if curated else records
candidates = [r for r in candidates if "partial" not in r.description.lower()] or candidates
return max(candidates, key=lambda r: len(r.seq))
out = []
for org, core in species_terms.items():
recs = search_and_fetch_sequences(org, core)
if not recs:
print(f"[WARN] No sequences found for {org}")
continue
best = pick_representative(recs)
acc = best.id
# Make a clean header like: Homo_sapiens|NP_000000.1|len=614
best.id = f"{org.replace(' ', '_')}|{acc}|len={len(best.seq)}"
best.description = ""
out.append(best)
print(f"Selected {org}: {acc} (length {len(best.seq)})")
time.sleep(0.34)
# Save to a single multi-FASTA file
if out:
SeqIO.write(out, "ache_proteins.fasta", "fasta")
print(f"Saved {len(out)} sequences to ache_proteins.fasta")
else:
print("No sequences selected.")
Ln 29, Col 1 Spaces: 4 UTF-8 CRLF {} Python Python 3.13 (64-bit)

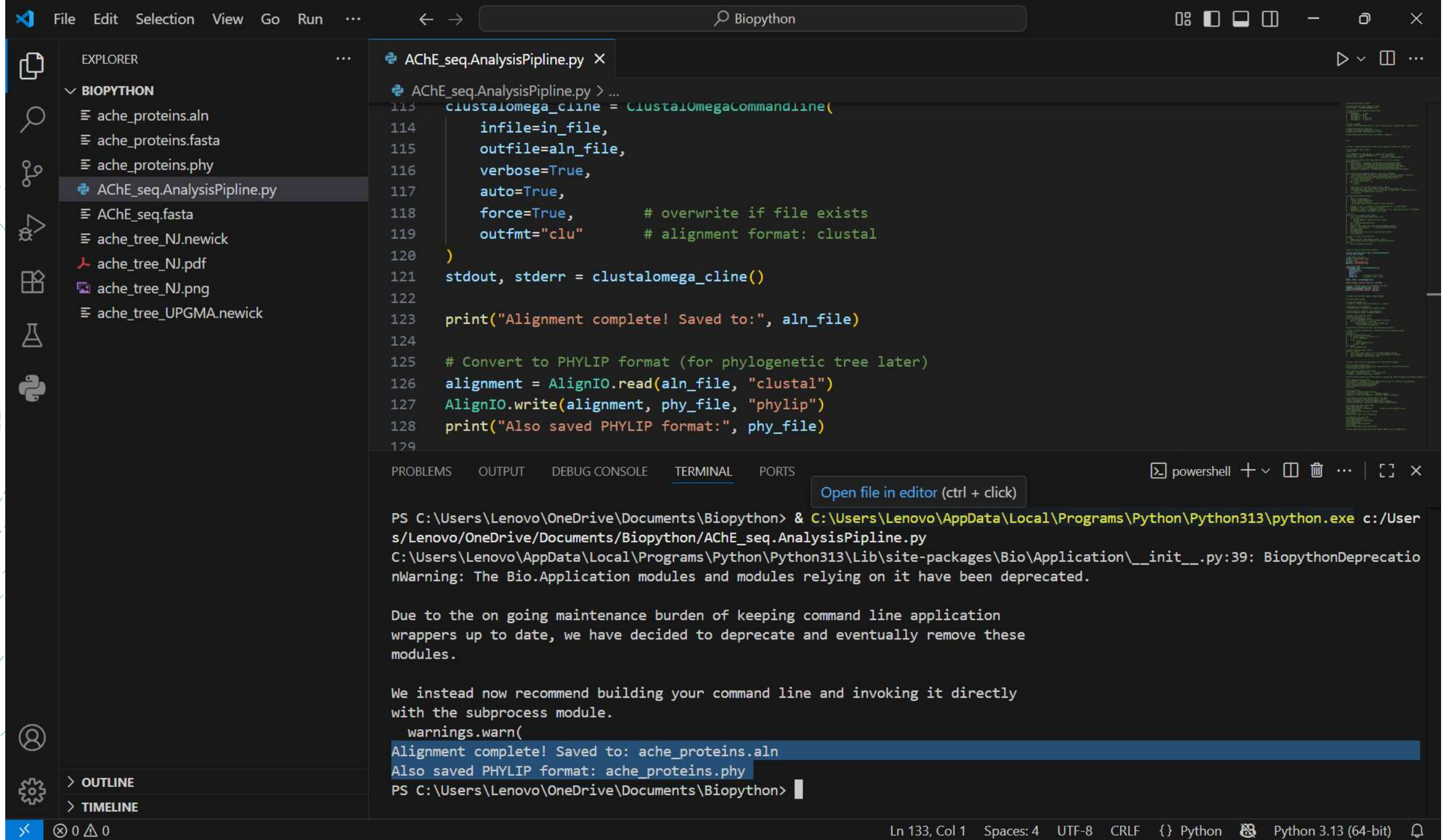

File Edit Selection View Go Run
Biopython
EXPLORER
BIOPYTHON
ache_proteins.aln
ache_proteins.fasta
ache_proteins.phy
AChE_seq.AnalysisPipline.py
AChE_seq.fasta
ache_tree_NJ.newick
ache_tree_NJ.pdf
ache_tree_NJ.png
ache_tree_UPGMA.newick
OUTLINE
TIMELINE
AChE_seq.AnalysisPipline.py
AChE_seq.AnalysisPipline.py > ...
82 continue
83 best = pick_representative(recs)
84 acc = best.id
85 # Make a clean header like: Homo_sapiens|NP_000000.1|len=614
86 best.id = f"{org.replace(' ', '_')}|{acc}|len={len(best.seq)}"
87 best.description = ""
88 out.append(best)
89 print(f"Selected {org}: {acc} (length {len(best.seq)})")
90 time.sleep(0.34)
91
92 # Save to a single multi-FASTA file
93 if out:
94 SeqIO.write(out, "ache_proteins.fasta", "fasta")
95 print(f"Saved {len(out)} sequences to ache_proteins.fasta")
96 else:
97 print("No sequences selected.")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
PS C:\Users\Lenovo\OneDrive\Documents\Biopython> & C:\Users\Lenovo\AppData\Local\Programs\Python\Python313\python.exe c:/Users/Lenovo/OneDrive/Documents/Biopython/AChE_seq.AnalysisPipline.py
Selected Homo sapiens: NP_001354847.1 (length 681)
Selected Mus musculus: NP_001276939.1 (length 614)
Selected Rattus norvegicus: NP_742006.2 (length 614)
Selected Danio rerio: NP_571921.1 (length 634)
Selected Drosophila melanogaster: NP_001163600.1 (length 649)
Saved 5 sequences to ache_proteins.fasta
PS C:\Users\Lenovo\OneDrive\Documents\Biopython>
Ln 103, Col 61 Spaces: 4 UTF-8 CRLF {} Python Python 3.13 (64-bit)

STEP 2: MULTIPLE SEQUENCE ALIGNMENT (MSA)

- Used Clustal Omega (through Biopython) to align sequences.
- MSA arranges sequences in a way that similar regions are aligned vertically.
- Purpose: To identify conserved motifs that remain unchanged across species (indicating functional importance).

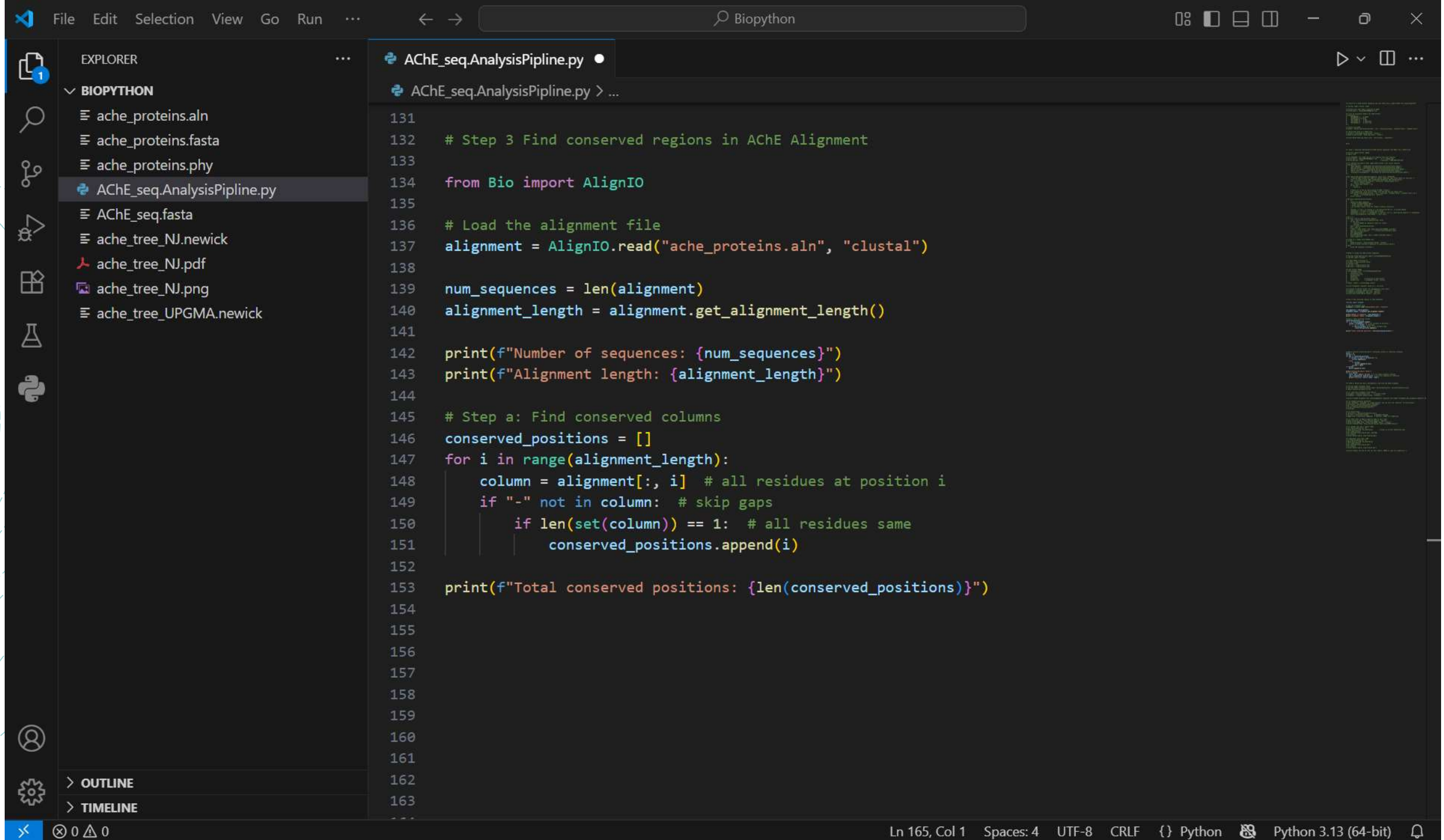
- Output files generated:
  .aln → alignment file (can be opened in Jalview/Clustal viewer).
  .phy → PHYLIP format (used in phylogenetic tree construction).

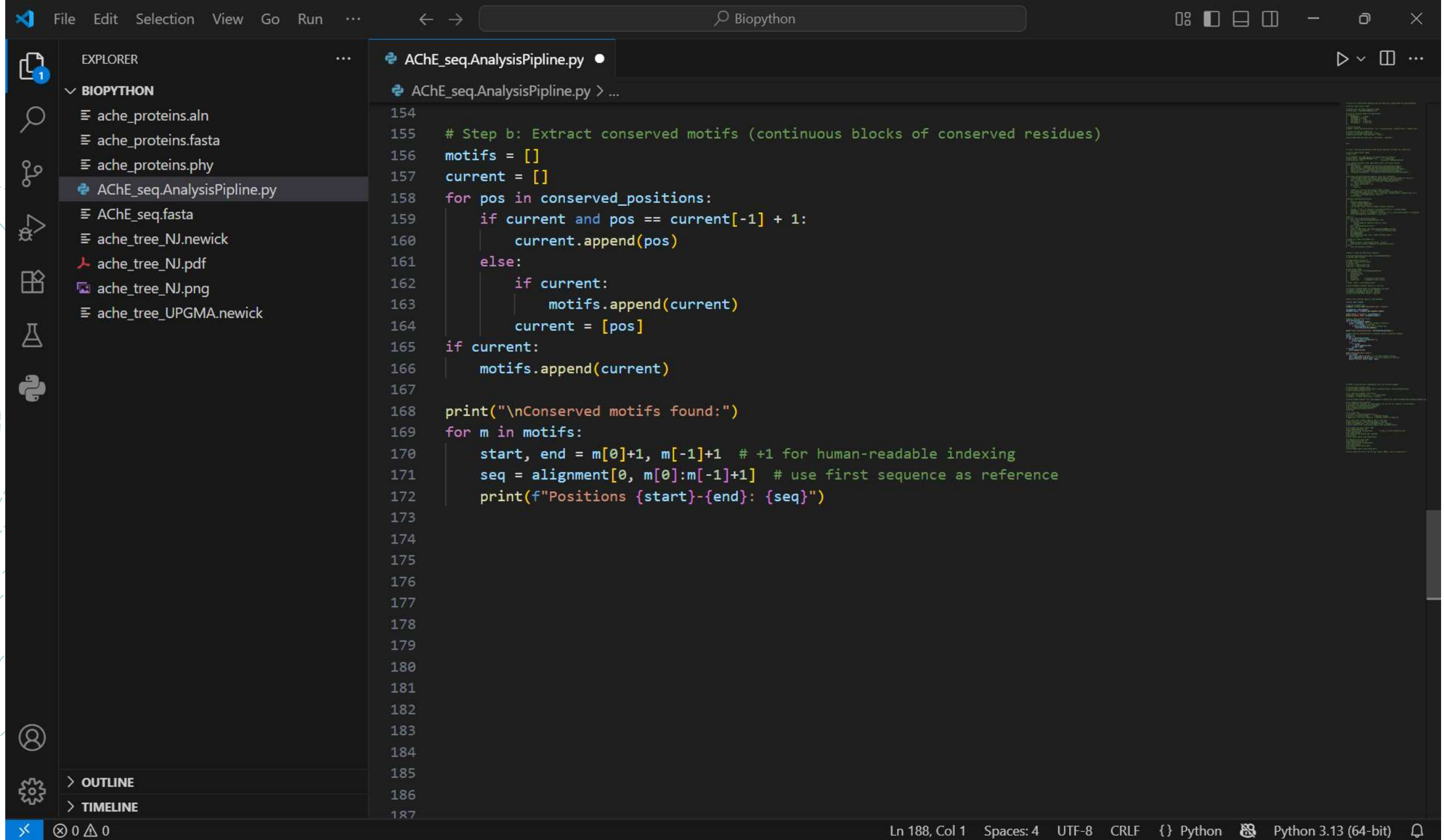

File Edit Selection View Go Run
Biopython
EXPLORER
BIOPYTHON
ache_proteins.aln
ache_proteins.fasta
ache_proteins.phy
AChE_seq.AnalysisPipline.py
AChE_seq.fasta
ache_tree_NJ.newick
ache_tree_NJ.pdf
ache_tree_NJ.png
ache_tree_UPGMA.newick
OUTLINE
TIMELINE
AChE_seq.AnalysisPipline.py
AChE_seq.AnalysisPipline.py > ...
# Step 2 – Align the AChE protein sequences
from Bio.Align.Applications import ClustalOmegaCommandline
from Bio import AlignIO
# Input FASTA (from Step 1)
in_file = "ache_proteins.fasta"
# Output files
aln_file = "ache_proteins.aln"
phy_file = "ache_proteins.phy"
# Run Clustal Omega
clustalomega_cline = ClustalOmegaCommandline(
infile=in_file,
outfile=aln_file,
verbose=True,
auto=True,
force=True, # overwrite if file exists
outfmt="clu" # alignment format: clustal
)
stdout, stderr = clustalomega_cline()
print("Alignment complete! Saved to:", aln_file)
# Convert to PHYLIP format (for phylogenetic tree later)
alignment = AlignIO.read(aln_file, "clustal")
AlignIO.write(alignment, phy_file, "phylip")
print("Also saved PHYLIP format:", phy_file)
Ln 133, Col 1 Spaces: 4 UTF-8 CRLF {} Python Python 3.13 (64-bit)

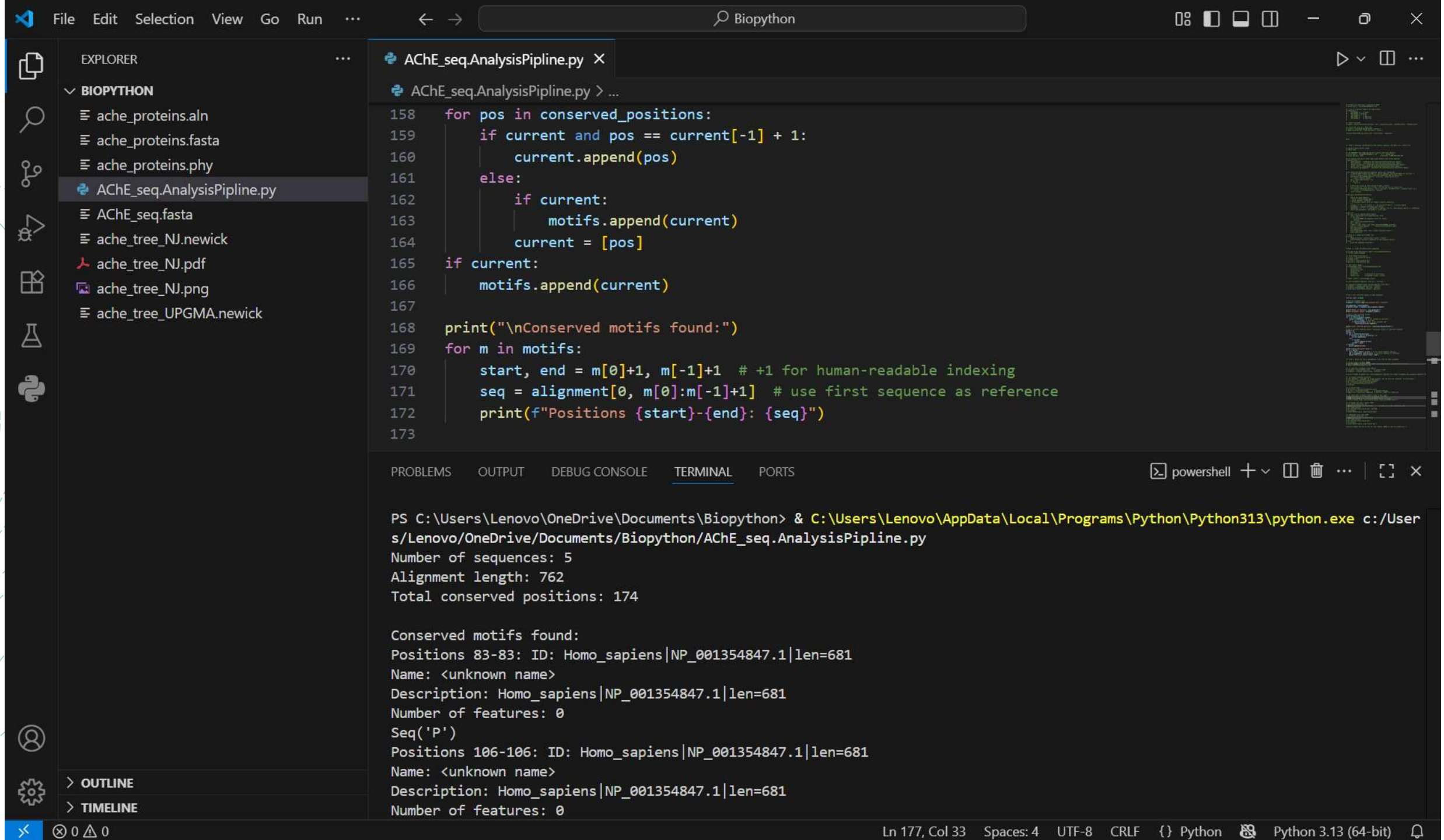

File Edit Selection View Go Run
Biopython
EXPLORER
BIOPYTHON
ache_proteins.aln
ache_proteins.fasta
ache_proteins.phy
AChE_seq.AnalysisPipline.py
AChE_seq.fasta
ache_tree_NJ.newick
ache_tree_NJ.pdf
ache_tree_NJ.png
ache_tree_UPGMA.newick
OUTLINE
TIMELINE
AChE_seq.AnalysisPipline.py
AChE_seq.AnalysisPipline.py > ...
clustalomega_cline = ClustalOmegaCommandline(
infile=in_file,
outfile=aln_file,
verbose=True,
auto=True,
force=True, # overwrite if file exists
outfmt="clu" # alignment format: clustal
)
stdout, stderr = clustalomega_cline()
print("Alignment complete! Saved to:", aln_file)
# Convert to PHYLIP format (for phylogenetic tree later)
alignment = AlignIO.read(aln_file, "clustal")
AlignIO.write(alignment, phy_file, "phylip")
print("Also saved PHYLIP format:", phy_file)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Open file in editor (ctrl + click)
PS C:\Users\Lenovo\OneDrive\Documents\Biopython> & C:\Users\Lenovo\AppData\Local\Programs\Python\Python313\python.exe c:/Users/Lenovo/OneDrive/Documents/Biopython/AChE_seq.AnalysisPipline.py
C:\Users\Lenovo\AppData\Local\Programs\Python\Python313\Lib\site-packages\Bio\Application__init__.py:39: BiopythonDeprecationWarning: The Bio.Application modules and modules relying on it have been deprecated.
Due to the on going maintenance burden of keeping command line application
wrappers up to date, we have decided to deprecate and eventually remove these
modules.
We instead now recommend building your command line and invoking it directly
with the subprocess module.
warnings.warn(
Alignment complete! Saved to: ache_proteins.aln
Also saved PHYLIP format: ache_proteins.phy
PS C:\Users\Lenovo\OneDrive\Documents\Biopython>
0 0
Ln 133, Col 1 Spaces: 4 UTF-8 CRLF {} Python Python 3.13 (64-bit)

STEP 3: CONSERVED MOTIF IDENTIFICATION

- **Analyzed the aligned sequences to find blocks of amino acids conserved across species.**
- **These conserved motifs likely represent active sites or binding regions of AChE.**
- **Such regions are important in drug design (because drugs bind to conserved functional sites).**


File Edit Selection View Go Run ···
Biopython
EXPLORER
BIOPYTHON
ache_proteins.aln
ache_proteins.fasta
ache_proteins.phy
AChE_seq.AnalysisPipline.py
AChE_seq.fasta
ache_tree_NJ.newick
ache_tree_NJ.pdf
ache_tree_NJ.png
ache_tree_UPGMA.newick
OUTLINE
TIMELINE
AChE_seq.AnalysisPipline.py
AChE_seq.AnalysisPipline.py > ...
131
132 # Step 3 Find conserved regions in AChE Alignment
133
134 from Bio import AlignIO
135
136 # Load the alignment file
137 alignment = AlignIO.read("ache_proteins.aln", "clustal")
138
139 num_sequences = len(alignment)
140 alignment_length = alignment.get_alignment_length()
141
142 print(f"Number of sequences: {num_sequences}")
143 print(f"Alignment length: {alignment_length}")
144
145 # Step a: Find conserved columns
146 conserved_positions = []
147 for i in range(alignment_length):
148 column = alignment[:, i] # all residues at position i
149 if "-" not in column: # skip gaps
150 if len(set(column)) == 1: # all residues same
151 conserved_positions.append(i)
152
153 print(f"Total conserved positions: {len(conserved_positions)}")
154
155
156
157
158
159
160
161
162
163
⊗ 0 ⚠ 0
Ln 165, Col 1 Spaces: 4 UTF-8 CRLF { } Python Python 3.13 (64-bit)


File Edit Selection View Go Run ···
Biopython
EXPLORER
BIOPYTHON
ache_proteins.aln
ache_proteins.fasta
ache_proteins.phy
AChE_seq.AnalysisPipline.py
AChE_seq.fasta
ache_tree_NJ.newick
ache_tree_NJ.pdf
ache_tree_NJ.png
ache_tree_UPGMA.newick
OUTLINE
TIMELINE
AChE_seq.AnalysisPipline.py
AChE_seq.AnalysisPipline.py > ...
154
155 # Step b: Extract conserved motifs (continuous blocks of conserved residues)
156 motifs = []
157 current = []
158 for pos in conserved_positions:
159 if current and pos == current[-1] + 1:
160 current.append(pos)
161 else:
162 if current:
163 motifs.append(current)
164 current = [pos]
165 if current:
166 motifs.append(current)
167
168 print("\nConserved motifs found:")
169 for m in motifs:
170 start, end = m[0]+1, m[-1]+1 # +1 for human-readable indexing
171 seq = alignment[0, m[0]:m[-1]+1] # use first sequence as reference
172 print(f"Positions {start}-{end}: {seq}")
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
0 0
Ln 188, Col 1 Spaces: 4 UTF-8 CRLF {} Python Python 3.13 (64-bit)


File Edit Selection View Go Run
Biopython
EXPLORER
BIOPYTHON
ache_proteins.aln
ache_proteins.fasta
ache_proteins.phy
AChE_seq.AnalysisPipline.py
AChE_seq.fasta
ache_tree_NJ.newick
ache_tree_NJ.pdf
ache_tree_NJ.png
ache_tree_UPGMA.newick
OUTLINE
TIMELINE
AChE_seq.AnalysisPipline.py
AChE_seq.AnalysisPipline.py > ...
158 for pos in conserved_positions:
159 if current and pos == current[-1] + 1:
160 current.append(pos)
161 else:
162 if current:
163 motifs.append(current)
164 current = [pos]
165 if current:
166 motifs.append(current)
167
168 print("\nConserved motifs found:")
169 for m in motifs:
170 start, end = m[0]+1, m[-1]+1 # +1 for human-readable indexing
171 seq = alignment[0, m[0]:m[-1]+1] # use first sequence as reference
172 print(f"Positions {start}-{end}: {seq}")
173
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
PS C:\Users\Lenovo\OneDrive\Documents\Biopython> & C:\Users\Lenovo\AppData\Local\Programs\Python\Python313\python.exe c:/Users/Lenovo/OneDrive/Documents/Biopython/AChE_seq.AnalysisPipline.py
Number of sequences: 5
Alignment length: 762
Total conserved positions: 174
Conserved motifs found:
Positions 83-83: ID: Homo_sapiens|NP_001354847.1|len=681
Name: <unknown name>
Description: Homo_sapiens|NP_001354847.1|len=681
Number of features: 0
Seq('P')
Positions 106-106: ID: Homo_sapiens|NP_001354847.1|len=681
Name: <unknown name>
Description: Homo_sapiens|NP_001354847.1|len=681
Number of features: 0
Ln 177, Col 33 Spaces: 4 UTF-8 CRLF {} Python Python 3.13 (64-bit)

STEP 4: PHYLOGENETIC TREE CONSTRUCTION

- **Used Biopython’s Phylo module to construct a tree from the alignment.**
- **The tree shows evolutionary relationships among species based on AChE similarity.**
- **Example: Human AChE is closer to Mouse/Rat than to Zebrafish or Fruit fly.**
- **Exported and visualized tree as .png for presentation.**

File Edit Selection View Go Run ··· Biopython

EXPLORER

BIOPYTHON
- ache_proteins.aln
- ache_proteins.fasta
- ache_proteins.phy
- AChE_seq.AnalysisPipline.py
- AChE_seq.fasta
- ache_tree_NJ.newick
- ache_tree_NJ.pdf
- ache_tree_NJ.png
- ache_tree_UPGMA.newick

OUTLINE

TIMELINE

AChE_seq.AnalysisPipline.py ●

AChE_seq.AnalysisPipline.py > ...

```python
# STEP 4: Build and save a phylogenetic tree from the AChE alignment

from Bio import AlignIO, Phylo
from Bio.Phylo.TreeConstruction import DistanceCalculator, DistanceTreeConstructor
import matplotlib.pyplot as plt

# 1) Load your alignment (from Step 2)
aln_path = "ache_proteins.aln"     # clustal format
alignment = AlignIO.read(aln_path, "clustal")

print(f"Loaded alignment with {len(alignment)} sequences and length {alignment.get_alignment_length()} a

# 2) Compute pairwise distances
# For proteins, 'blosum62' is a good default; you can also try 'identity' to sanity-check.
calculator = DistanceCalculator('blosum62')
dm = calculator.get_distance(alignment)
print("\nPairwise distance matrix:")
print(dm)

# 3) Build trees
constructor = DistanceTreeConstructor()
nj_tree = constructor.nj(dm)        # Neighbor-Joining
upgma_tree = constructor.upgma(dm)  # Optional: UPGMA for comparison

# 4) Save trees to Newick (good to keep in your repo)
Phylo.write(nj_tree, "ache_tree_NJ.newick", "newick")
Phylo.write(upgma_tree, "ache_tree_UPGMA.newick", "newick")
print("\nSaved trees: ache_tree_NJ.newick, ache_tree_UPGMA.newick")
```

⊗ 0 ⚠ 0 Ln 208, Col 1 Spaces: 4 UTF-8 CRLF {} Python Python 3.13 (64-bit)

File Edit Selection View Go Run ···

Biopython

EXPLORER

BIOPYTHON

- ache_proteins.aln
- ache_proteins.fasta
- ache_proteins.phy
- AChE_seq.AnalysisPipline.py
- AChE_seq.fasta
- ache_tree_NJ.newick
- ache_tree_NJ.pdf
- ache_tree_NJ.png
- ache_tree_UPGMA.newick

OUTLINE

TIMELINE

AChE_seq.AnalysisPipline.py ●

AChE_seq.AnalysisPipline.py > ...

```
209
210 # 5) Render and save a figure (PNG)
211 plt.figure(figsize=(8, 5))
212 Phylo.draw(nj_tree, do_show=False)        # draws on current matplotlib axes
213 plt.tight_layout()
214 plt.savefig("ache_tree_NJ.png", dpi=300)
215 plt.close()
216 print("Saved figure: ache_tree_NJ.png")
217
218 # (Optional) also save a PDF
219 plt.figure(figsize=(8, 5))
220 Phylo.draw(nj_tree, do_show=False)
221 plt.tight_layout()
222 plt.savefig("ache_tree_NJ.pdf")
223 plt.close()
224 print("Saved figure: ache_tree_NJ.pdf")
225
226 print("\nDone! Use the NJ tree for your report; UPGMA is just for comparison.")
227
228
229
```

⊗ 0 ⚠ 0 Ln 201, Col 60 Spaces: 4 UTF-8 CRLF {} Python Python 3.13 (64-bit)

File Edit Selection View Go Run ··· Biopython

EXPLORER

BIOPYTHON
- ache_proteins.aln
- ache_proteins.fasta
- ache_proteins.phy
- AChE_seq.AnalysisPipline.py
- AChE_seq.fasta
- ache_tree_NJ.newick
- ache_tree_NJ.pdf
- ache_tree_NJ.png
- ache_tree_UPGMA.newick

OUTLINE
TIMELINE

AChE_seq.AnalysisPipline.py

AChE_seq.AnalysisPipline.py > ...

```
212  Phylo.draw(nj_tree, do_show=False)        # draws on current matplotlib axes
213  plt.tight_layout()
214  plt.savefig("ache_tree_NJ.png", dpi=300)
215  plt.close()
216  print("Saved figure: ache_tree_NJ.png")
217
218  # (Optional) also save a PDF
219  plt.figure(figsize=(8, 5))
220  Phylo.draw(nj_tree, do_show=False)
221  plt.tight_layout()
222  plt.savefig("ache_tree_NJ.pdf")
223  plt.close()
224  print("Saved figure: ache_tree_NJ.pdf")
225
226  print("\nDone! Use the NJ tree for your report; UPGMA is just for comparison.")
227
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

powershell

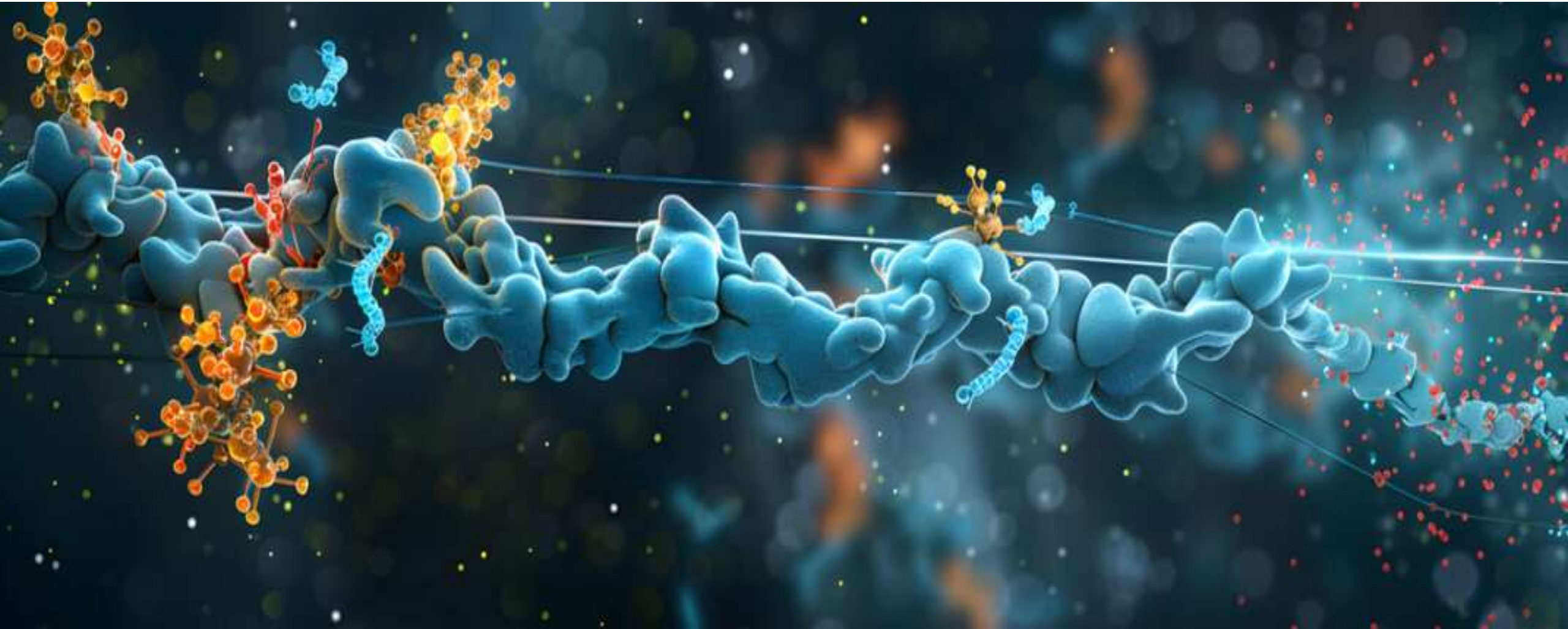
```
Pairwise distance matrix:
Homo_sapiens|NP_001354847.1|len=681 0.000000
Mus_musculus|NP_001276939.1|len=614 0.101726    0.000000
Rattus_norvegicus|NP_742006.2|len=614   0.098093    0.012129    0.000000
Danio_rerio|NP_571921.1|len=634 0.381098    0.380792    0.377723    0.000000
Drosophila_melanogaster|NP_001163600.1|len=649  0.667700    0.661848    0.662999    0.653087    0.000000
    Homo_sapiens|NP_001354847.1|len=681 Mus_musculus|NP_001276939.1|len=614 Rattus_norvegicus|NP_742006.2|len=614   Danio_rer
io|NP_571921.1|len=634 Drosophila_melanogaster|NP_001163600.1|len=649

Saved trees: ache_tree_NJ.newick, ache_tree_UPGMA.newick
Saved figure: ache_tree_NJ.png
Saved figure: ache_tree_NJ.pdf

Done! Use the NJ tree for your report; UPGMA is just for comparison.
PS C:\Users\Lenovo\OneDrive\Documents\Biopython>
```

⊗ 0 ⚠ 0 Ln 201, Col 60 Spaces: 4 UTF-8 CRLF {} Python Python 3.13 (64-bit)

VISUALIZATION & INTERPRETATION



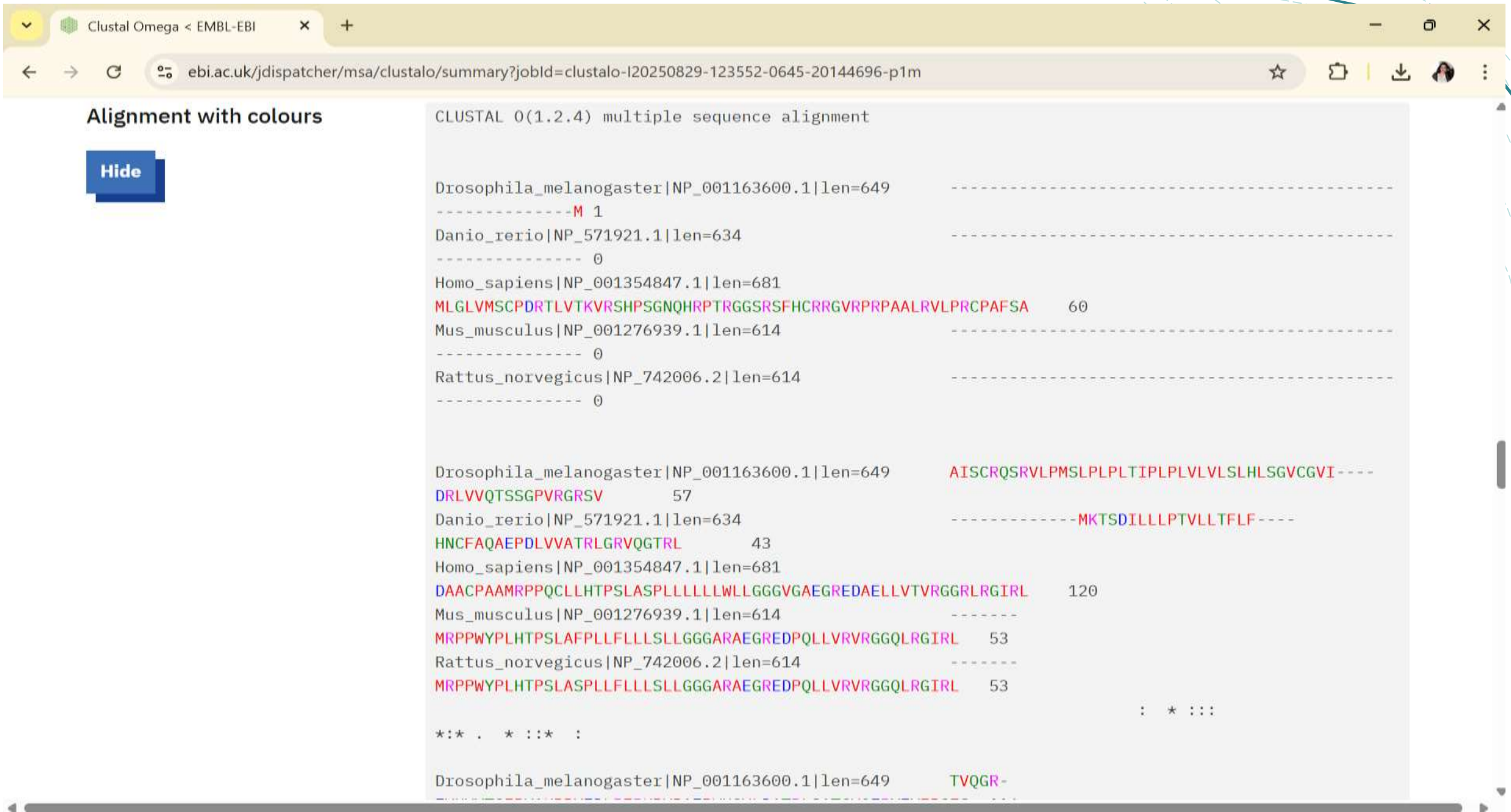
RESULTS – MULTIPLE SEQUENCE ALIGNMENT (MSA) & CONSERVED MOTIFS

- **Alignment Process**
  Performed using **Clustal Omega** via Biopython.
  Input: ache_proteins.fasta (AChE sequences from human, mouse, rat, zebrafish, fruit fly).

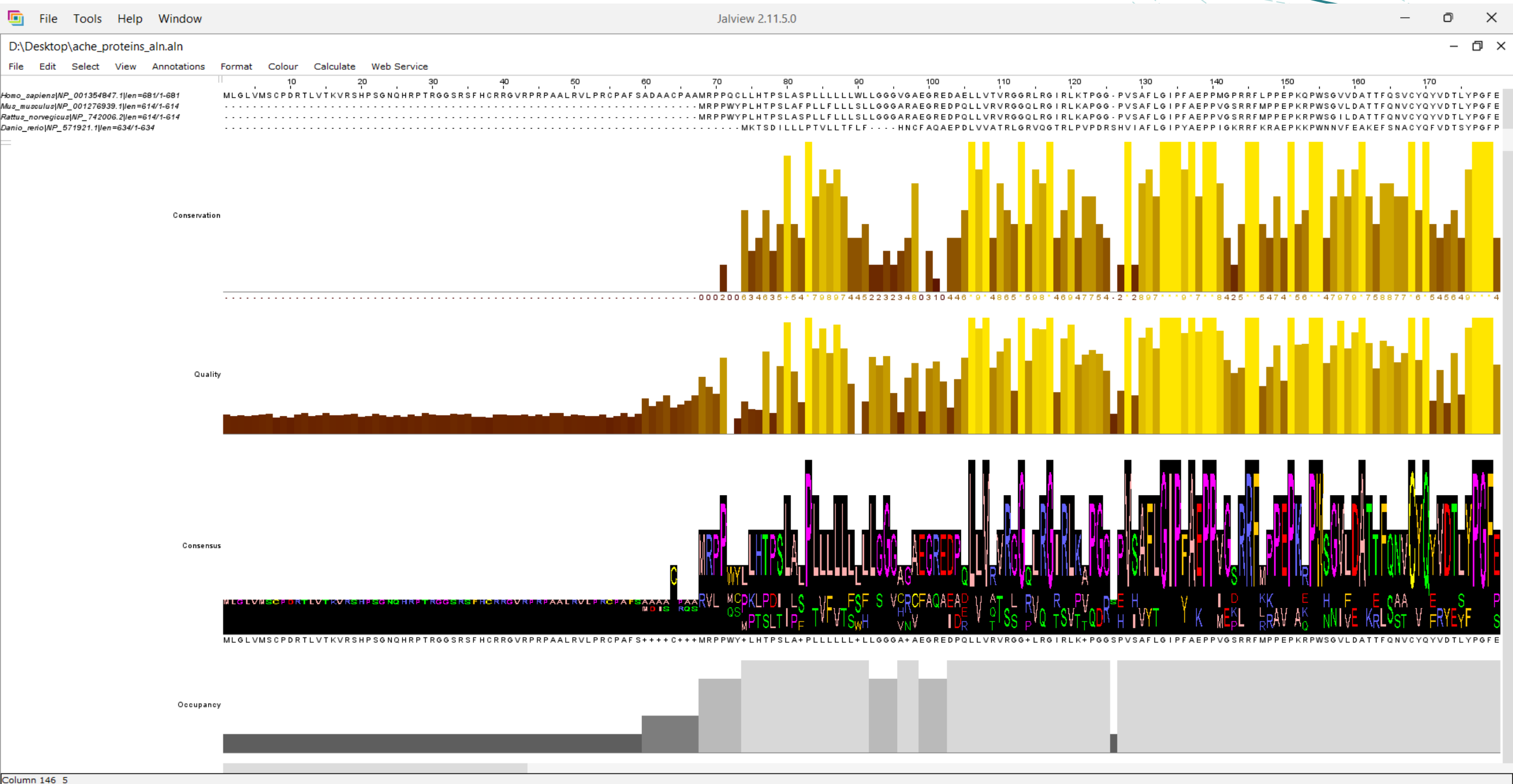
- **Output files:**
  .aln file → for alignment visualization.
  .phy file → for tree construction.

- **Key Observations**
  Fully conserved motifs detected → identical residues across all species.
  Semi-conserved regions → substitutions with chemically similar amino acids.
  Variable regions → species-specific differences.
  Conserved motifs suggest **functional importance** (likely active or binding sites).

ALIGNMENT VIEWED IN CLUSTAL OMEGA ONLINE VIEWER



ALIGNMENT VIEWED IN JALVIEW



RESULTS – PHYLOGENETIC TREE (BIOPYTHON + MATPLOTLIB)

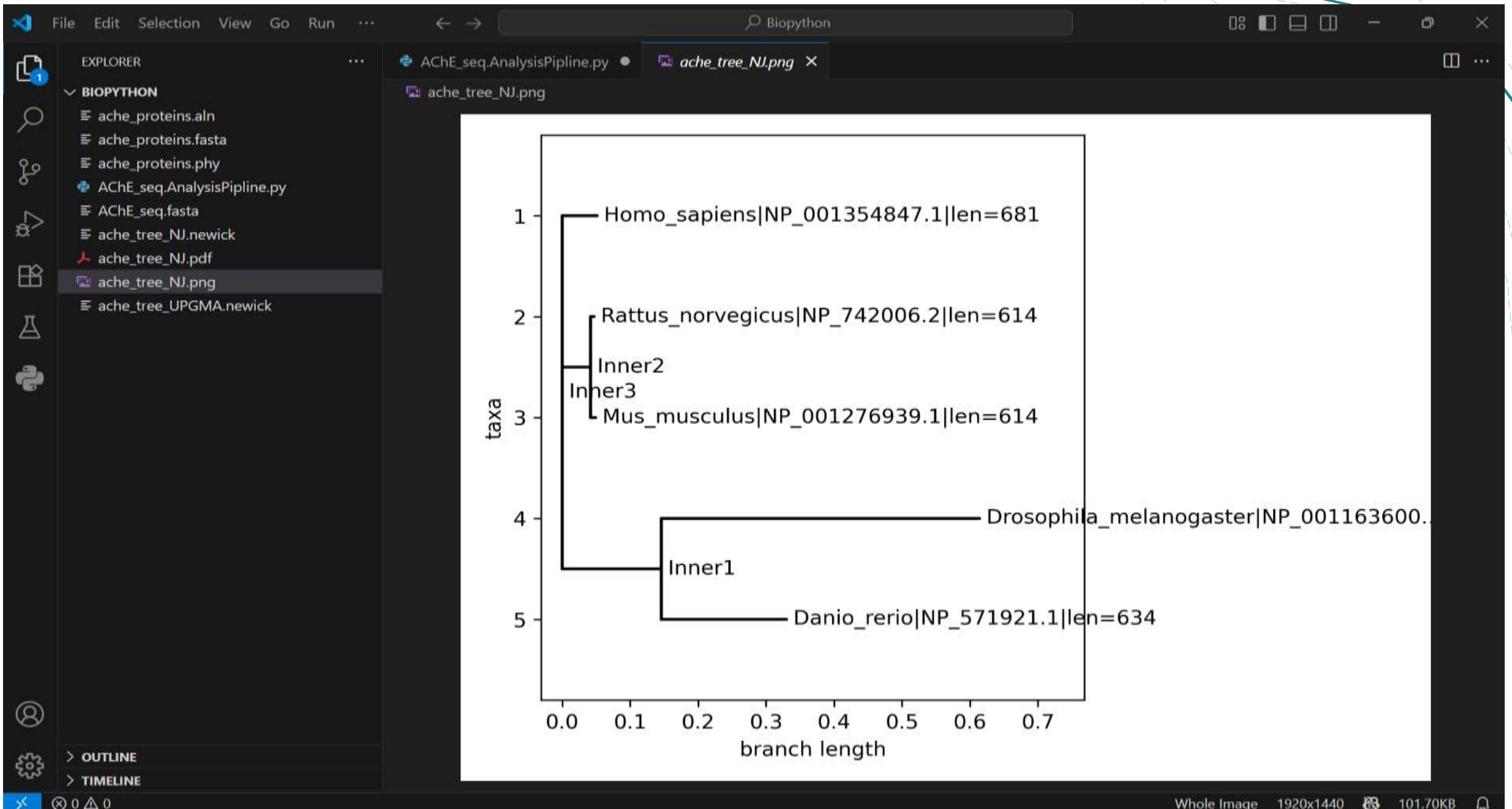
- **Tree Construction**
  Used **Biopython (Phylo module)** with the .phy alignment file.
  Tree visualization performed using **Matplotlib**.
  Exported tree as an image (tree.png).

- **Key Observations**
  -Human AChE clustered closely with **mouse and rat**, reflecting mammalian evolutionary relationship.

  -Zebrafish formed a separate branch → closer to mammals than insects but still distinct.

  -Fruit fly (insect) appeared as the **most distant branch**, confirming evolutionary divergence.

PHYLOGENETIC TREE VISUALIZED WITH MATPLOTLIB + BIOPYTHON



DISCUSSION & BIOLOGICAL SIGNIFICANCE :

✓ Key Findings

- **Multiple Sequence Alignment (MSA):**
  -Conserved motifs identified across all 5 species.
  -Suggest these residues are **functionally essential** (active/binding sites).

- **Phylogenetic Tree:**
  -Mammals (human, mouse, rat) grouped together.
  -Zebrafish formed a separate branch but still closer to mammals than insects.
  -Fruit fly diverged earliest, showing **largest evolutionary distance**.

✓ Biological Significance

Conserved motifs → potential **drug target regions** for Alzheimer's therapies.
Mammalian clustering → validates mouse/rat as suitable models for AChE research.
Evolutionary insights → highlights structural conservation of AChE across species.

CONCLUSION :

- ✓ Successfully analyzed **Acetylcholinesterase (AChE) sequences** from multiple species using Biopython.
- ✓ **Multiple Sequence Alignment (MSA):** revealed **conserved motifs**, indicating essential functional regions.
- ✓ **Phylogenetic Tree:** showed evolutionary relationships — mammals grouped closely, insect (fruit fly) most distant.
- ✓ Demonstrated how **bioinformatics tools (Biopython, Jalview, Matplotlib)** can provide insights into protein conservation and evolution.

FUTURE WORK :

- **Expand dataset** → include more species for broader evolutionary analysis.
- **Structural modeling** → map conserved motifs onto 3D AChE protein structure.
- **Drug docking studies** → test interactions of conserved sites with potential inhibitors (e.g., curcumin, donepezil).
- **Automated pipeline** → build a Biopython script to handle retrieval, alignment, tree building, and visualization in one go.

ACKNOWLEDGEMENTS

- **Biopython developers** for providing open-source bioinformatics tools.
- **Clustal Omega & Jalview** for multiple sequence alignment visualization.
- **Matplotlib** for phylogenetic tree plotting.
- **NCBI** for sequence data retrieval.

REFERENCES

1) Cock PJA et al. (2009). *Biopython: freely available Python tools for computational molecular biology and bioinformatics*. **Bioinformatics, 25(11):1422–1423.**

2) Sievers F & Higgins DG. (2018). *Clustal Omega for making accurate alignments of many protein sequences*. **Protein Science, 27(1):135–145.**

3) Waterhouse AM et al. (2009). *Jalview Version 2—a multiple sequence alignment editor and analysis workbench*. **Bioinformatics, 25(9):1189–1191.**

4) Hunter JD. (2007). *Matplotlib: A 2D graphics environment*. **Computing in Science & Engineering, 9(3):90–95.**

THANK YOU!