Task 3: Working with LLM

Michal Piecuch (s32484)
Bioinformatics - Exercise 3

Part one: General experience

How often did you use AI during your work?

- ☐ At every step
- ☒ For generating the initial (skeleton) code, then only when you encountered a problem
- ☐ For generating the initial (skeleton) code, then worked completely independently
- ☐ Other: _____

I used AI mostly at the beginning to get a basic script working and check Biopython / matplotlib syntax, then mainly when I hit an error or wanted help with the extra features.

Part two: Specific applications

What exactly did you use AI for? (select all that apply)

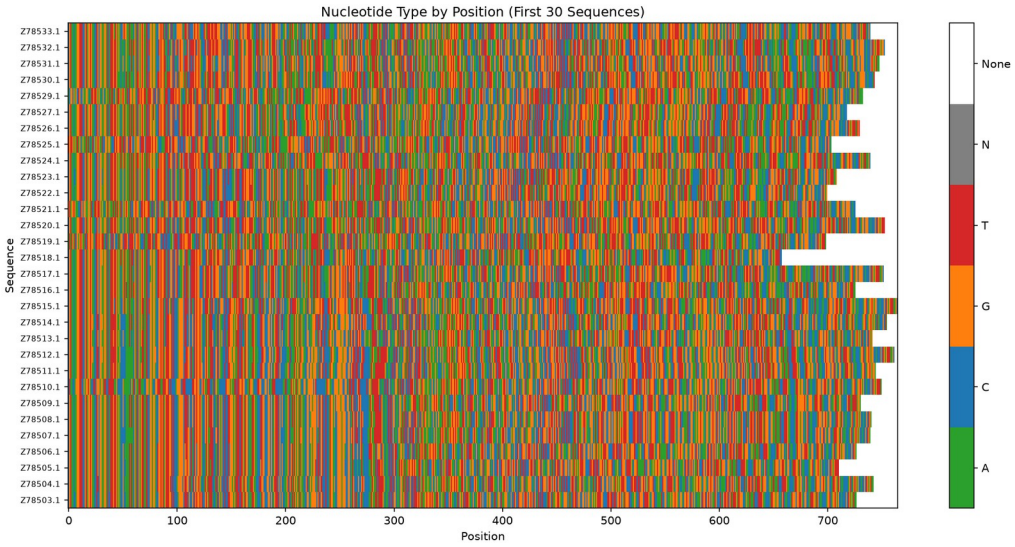
- ☐ Generating the entire code from scratch
- ☒ Explaining Biopython syntax
- ☒ Debugging errors
- ☒ Code optimization
- ☒ Creating visualizations
- ☒ Helping to understand biological concepts
- ☐ Other: _____

Provide 1-2 specific examples where AI was most helpful.

1. Fixing warnings in 6-frame translation (Task 2). When I ran translate() on the sequences from ls_orchid.fasta, Biopython showed a lot of partial codon warnings because the lengths were not divisible by 3 (especially after shifting by 1 or 2). AI reminded me that frames 4-6 go from reverse_complement() in 5' to 3' direction and suggested trimming each frame with len(seq) - len(seq) % 3 before translating. After that warnings were gone and all 6 frames matched the short example from the instruction.

```
(.venv) mp@omarchy:~/bioinformatics_ex3$ python3 -c "from Bio import SeqIO; s =
next(SeqIO.parse('ls_orchid.fasta', 'fasta')).seq; [s[i:].translate() for i in range(3)]"
/home/mp/bioinformatics_ex3/.venv/lib/python3.11/site-packages/Bio/Seq.py:2874: BiopythonWarning: Partial
codon, len(sequence) not a multiple of three. Explicitly trim the sequence or add trailing N before
translation. This may become an error in future.
  warnings.warn(
(.venv) mp@omarchy:~/bioinformatics_ex3$ python3 -c "from Bio.Seq import Seq; from s32484_task2 import
translate_6_frames; frames = translate_6_frames(Seq('ACATGCCTGAAACGAATTCTATATGGCCT')); [print(f'Frame {k}:
{frames[k]}') for k in range(1, 7)]"
Frame 1: TCLKRILYG
Frame 2: HA*NEFYMA
Frame 3: MPETNSIWP
Frame 4: RPYRIRFRH
Frame 5: GHIEFVSGM
Frame 6: AI*NSFQAC
(.venv) mp@omarchy:~/bioinformatics_ex3$
```

2. Making the heatmap in matplotlib. I wasn't sure how to plot letters (A, C, G, T, N) as a heatmap, especially because the first 30 sequences have different lengths (658 to 765 bp). AI showed how to map letters to numbers 0-5 (where 5 is empty white space after a sequence ends) and use ListedColormap with custom colorbar labels. It worked pretty well, I just tweaked the figure size and shortened the NCBI IDs so the labels didn't overlap.



Provide 1-2 specific examples where AI failed or misled you.

1. Old Bio.SeqUtils.GC function. At the start AI imported GC from Bio.SeqUtils, which gave an ImportError in my Biopython version because it was removed in 1.80+. I checked the theory part in the instructions and changed it to gc_fraction() * 100.

2. Overlapping motifs and N bases in ls_orchid.fasta. For motif search AI first suggested re.finditer, which misses overlapping motifs (for example in TATATA it only found index 0 and skipped index 2). It also forgot that ls_orchid.fasta has N bases in several sequences. I replaced the regex with a simple str.find loop moving by idx + 1, and added N to the counts and charts.

```
(.venv) mp@omarchy:~/bioinformatics_ex3$ python3 -c "from Bio.SeqUtils import GC"
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ImportError: cannot import name 'GC' from 'Bio.SeqUtils'
(/home/mp/bioinformatics_ex3/.venv/lib/python3.11/site-packages/Bio/SeqUtils/__init__.py)
(.venv) mp@omarchy:~/bioinformatics_ex3$ python3 -c "from Bio.Seq import Seq; from Bio.SeqUtils import
gc_fraction; s = Seq('ACATGCCTGAAACGAATTCTATATGGCCT'); print(round(gc_fraction(s) * 100, 2))"
41.38
(.venv) mp@omarchy:~/bioinformatics_ex3$ python3 -c "import re; from s32484_task2 import
find_motif_positions; print('re.finditer:', [m.start() for m in re.finditer('TATA', 'TATATA')]);
print('find_motif_positions:', find_motif_positions('TATATA', 'TATA'))"
re.finditer: [0]
find_motif_positions: [0, 2]
(.venv) mp@omarchy:~/bioinformatics_ex3$ python3 -c "from Bio import SeqIO; recs =
list(SeqIO.parse('ls_orchid.fasta', 'fasta')); print([(r.id.split('|')[3], str(r.seq).count('N')) for r in
recs if 'N' in str(r.seq)])"
[('Z78524.1', 1), ('Z78482.1', 2), ('Z78475.1', 96), ('Z78462.1', 265), ('Z78454.1', 1), ('Z78448.1', 2),
('Z78443.1', 174)]
(.venv) mp@omarchy:~/bioinformatics_ex3$
```

Part three: Pros and cons

Main ADVANTAGES of working with AI:

- Saves a lot of time on repetitive stuff like matplotlib labels, colorbars, writing CSV/JSON files and input checks, instead of digging through docs for 20 minutes.
- Helpful for connecting biology theory with code, like how reverse_complement() relates to reading frames 4-6.
- Faster debugging, because you can paste a traceback or warning and quickly see what caused it.

Main DISADVANTAGES or LIMITATIONS of AI:

- Often gives outdated code from old tutorials (like Bio.SeqUtils.GC), so you always have to run and test it on your own Python setup.
- Usually assumes clean input data and skips edge cases, like different sequence lengths, N bases in FASTA files, or overlapping motifs.
- If you just copy-paste everything, you can end up with code that is too complicated and hard to explain later, so I still had to simplify some parts and check the calculations myself (for example making sure the inserted name is not counted in nucleotide stats).