Ege University /Computer Engineering

Yunus Can Duman

19.02.2026

COMPUTATIONAL BIOLOGY

FINDING THE MOST FREQUENT CODON USAGE IN THE HUMAN GENOME

Task:

Which codons are used more frequently in the human genome?

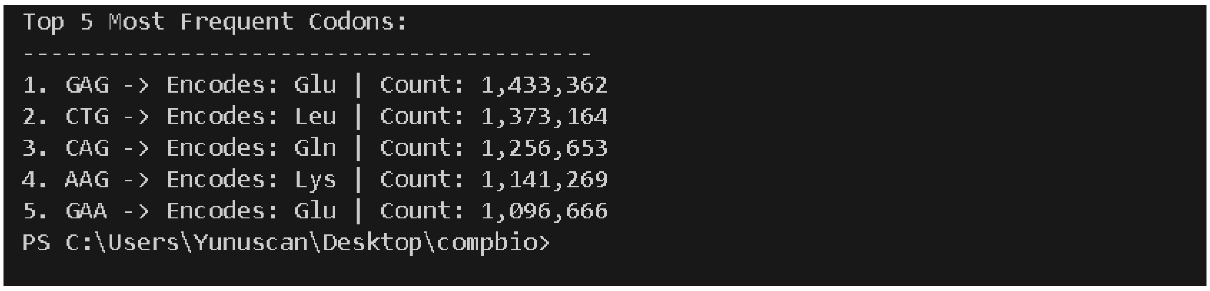
Report the top 5 codons and their amino-acids that they encode for.

Show your code and explain your approach.

Source Code(.py File):

```python
import collections

def get_top_codons(fasta_file, top_n=5):
    # Standard genetic code mapping
    amino_acids = {
        'GCT': 'Ala', 'GCC': 'Ala', 'GCA': 'Ala', 'GCG': 'Ala',
        'CGT': 'Arg', 'CGC': 'Arg', 'CGA': 'Arg', 'CGG': 'Arg', 'AGA': 'Arg', 'AGG': 'Arg',
        'AAT': 'Asn', 'AAC': 'Asn',
        'GAT': 'Asp', 'GAC': 'Asp',
        'TGT': 'Cys', 'TGC': 'Cys',
        'CAA': 'Gln', 'CAG': 'Gln',
        'GAA': 'Glu', 'GAG': 'Glu',
        'GGT': 'Gly', 'GGC': 'Gly', 'GGA': 'Gly', 'GGG': 'Gly',
        'CAT': 'His', 'CAC': 'His',
        'ATT': 'Ile', 'ATC': 'Ile', 'ATA': 'Ile',
        'TTA': 'Leu', 'TTG': 'Leu', 'CTT': 'Leu', 'CTC': 'Leu', 'CTA': 'Leu', 'CTG': 'Leu',
        'AAA': 'Lys', 'AAG': 'Lys',
        'ATG': 'Met',
        'TTT': 'Phe', 'TTC': 'Phe',
        'CCT': 'Pro', 'CCC': 'Pro', 'CCA': 'Pro', 'CCG': 'Pro',
        'TCT': 'Ser', 'TCC': 'Ser', 'TCA': 'Ser', 'TCG': 'Ser', 'AGT': 'Ser', 'AGC': 'Ser',
        'ACT': 'Thr', 'ACC': 'Thr', 'ACA': 'Thr', 'ACG': 'Thr',
        'TGG': 'Trp',
        'TAT': 'Tyr', 'TAC': 'Tyr',
        'GTT': 'Val', 'GTC': 'Val', 'GTA': 'Val', 'GTG': 'Val',
        'TAA': 'Stop', 'TGA': 'Stop', 'TAG': 'Stop'
    }
    codon_counts = collections.Counter()

    # Parsing the FASTA file
    with open(fasta_file, 'r') as file:
        sequence = []
        for line in file:
            line = line.strip()
            if line.startswith(">"):
                # Processing the accumulated sequence before starting the new one
                if sequence:
                    full_seq = "".join(sequence).upper()
                    # Extract codons (step by 3)
                    for i in range(0, len(full_seq) - 2, 3):
                        codon = full_seq[i:i+3]
                        # Only counting clean, valid length codons
                        if len(codon) == 3 and 'N' not in codon:
                            codon_counts[codon] += 1
                sequence = [] # Reseting for the next sequence
            else:
                sequence.append(line)

        # Processing the very last sequence in the file
        if sequence:
            full_seq = "".join(sequence).upper()
            for i in range(0, len(full_seq) - 2, 3):
                codon = full_seq[i:i+3]
                if len(codon) == 3 and 'N' not in codon:
                    codon_counts[codon] += 1

    # Getting the top N codons
    top_codons = codon_counts.most_common(top_n)

    # Printing the results
    print(f"Top {top_n} Most Frequent Codons:")
    print("-" * 40)
    for rank, (codon, count) in enumerate(top_codons, 1):
        aa = amino_acids.get(codon, "Unknown")
        print(f"{rank}. {codon} -> Encodes: {aa} | Count: {count:,}")

# file path
fasta_filename = "Homo_sapiens.GRCh38.cds.all.fa"
get_top_codons(fasta_filename)
```

Output:



```
Top 5 Most Frequent Codons:
----------------------------------------
1. GAG -> Encodes: Glu | Count: 1,433,362
2. CTG -> Encodes: Leu | Count: 1,373,164
3. CAG -> Encodes: Gln | Count: 1,256,653
4. AAG -> Encodes: Lys | Count: 1,141,269
5. GAA -> Encodes: Glu | Count: 1,096,666
PS C:\Users\Yunuscan\Desktop\compbio>
```

The Overall Strategy (The Algorithm)

Because human genome files are enormous (often gigabytes in size), we cannot load the entire file into the computer's memory (RAM) all at once. If we did, the computer would freeze or crash.

To solve this, my approach uses a "Line-by-Line Buffering" strategy.

Here is the logic in 5 simple steps:

1. Open the file safely and read it one single line at a time.
2. Clean the line by removing invisible characters (like "Enter" key strokes) using .strip().
3. Check for a Header (>): * If the line is a header, it means a new gene is starting.
    a. Before looking at this new gene, we must stop and process the sequence of the previous gene we were collecting.
    b. After processing, we empty our collection basket to prepare for the new gene.
4. Collect the Sequence: If the line is not a header, it must be DNA. We glue it to the pieces we already have in our basket.
5. The Final Cleanup: When the file completely ends, the last gene is still in the basket. We process it one final time outside the loop.