Genomic Sequence Analyzer and Annotator

Team 2

RITHIHA SHREE.S , RESHMA.R , ANUSHA.K, FATHIMA SHAMEEHA.S, LEKSHMI.S.S , SAAJINA.M , SHERIN FATHIMA.S.H , SAFA HANIA.S , ISHANA FATHIMA.A , SHALIHA.S

INTRODUCTION

This Python code is a basic Genomic Sequence Analyzer that reads DNA sequences from FASTA or GenBank files and performs simple biological operations. It counts the number of each nucleotide (A, T, G, C), generates the reverse complement of the sequence, and transcribes DNA into mRNA. It also prints a summary report and visualizes the nucleotide composition using a bar chart. This tool helps beginners explore and understand essential sequence analysis concepts in bioinformatics using Biopython and matplotlib.

To install program

```
pip install biopython
```

```
Collecting biopython
  Downloading biopython-1.85-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (13 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from biopython) (2.0.2)
Downloading biopython-1.85-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.3/3.3 MB 29.2 MB/s eta 0:00:00
Installing collected packages: biopython
Successfully installed biopython-1.85
```

```
from Bio.Seq import Seq
```

Parsing fasta and genbank files

```
# Genomic Sequence Analyzer and Annotator

# Parsing FASTA File
from Bio import SeqIO

print("FASTA File Output:")
for record in SeqIO.parse("/content/sample_data/sequence.fasta", "fasta"):
    print("ID:", record.id)
    print("Sequence:", record.seq)
    print()

# Parsing GenBank File
print("GenBank File Output:")
for record in SeqIO.parse("/content/sample_data/sequence.gb", "genbank"):
    print("ID:", record.id)
    print("Sequence Length:", len(record.seq))
    print("Total Features:", len(record.features))
    print()
```

```
FASTA File Output:
ID: AH002844.2
Sequence: CTCGAGGGGCCTAGACATTGCCCTCCAGAGAGAGCACCCAACACCCTCCAGGCTTGACCGGCCAGGGTGTCCCCTTCCTACCTTGGAGAGAGCAGCCCCAGGGCATCCTGCAGGGGGTGCTGGGAC

GenBank File Output:
ID: NC_000913.3
Sequence Length: 4641652
Total Features: 1
```

calculate gc content

```
# gc content
def gc_content(seq):
  g=seq.count("G")
  c=seq.count("C")
  return 100 * (g+c) / len(seq)

print(gc_content("ATGCGTAACG"))

from Bio.SeqUtils import gc_fraction
seq=Seq("ATGCGTAACGTACGTATGCGT")
gc=gc_fraction(seq)*100
print(gc)
```

```
50.0
47.61904761904761
```

Calculate molecular weight

```
# molecular weight
from Bio.SeqUtils import molecular_weight
dna_seq=Seq("ATGCGTAACGTACGTATGCGT")
molecular_weight(dna_seq)
```

```
6541.169599999999
```

Generate comprehensive sequence reports

Output: 📄 Multi-format reports with tables & 📊 visual graphs

```
from Bio import SeqIO
import matplotlib.pyplot as plt

def parse_file(filename):
    """Parse FASTA or GenBank file and return sequence records"""
    if filename.endswith(".gb") or filename.endswith(".gbk"):
        return list(SeqIO.parse(filename, "genbank"))
    elif filename.endswith(".fasta") or filename.endswith(".fa"):
        return list(SeqIO.parse(filename, "fasta"))
    else:
        raise ValueError("Unsupported file format.")

def analyze_sequence_basic(record):
    """Perform basic sequence analysis"""
    seq = record.seq
    nucleotide_counts = {
        'A': seq.count('A'),
        'T': seq.count('T'),
        'G': seq.count('G'),
        'C': seq.count('C')
    }
    reverse_complement = str(seq.reverse_complement())
    mRNA = str(seq.transcribe())

    return {
        "ID": record.id,
        "Description": record.description,
        "Length": len(seq),
        "Nucleotide Counts": nucleotide_counts,
        "Reverse Complement": reverse_complement[:50] + "...",
        "mRNA Sequence": mRNA[:50] + "..."
    }

def generate_report(data):
    print("\n📄 Basic Sequence Report")
    print(f"ID: {data['ID']}")
    print(f"Description: {data['Description']}")
    print(f"Length: {data['Length']} bp")
    print(f"Nucleotide Counts: {data['Nucleotide Counts']}")
    print(f"Reverse Complement (first 50 bases): {data['Reverse Complement']}")
    print(f"mRNA Sequence (first 50 bases): {data['mRNA Sequence']}")

def plot_nucleotide_counts(nuc_counts, seq_id):
```

```
    bases = list(nuc_counts.keys())
    counts = list(nuc_counts.values())

    plt.bar(bases, counts, color=["blue", "red", "green", "orange"])
    plt.title(f"Nucleotide Distribution for {seq_id}")
    plt.xlabel("Nucleotide")
    plt.ylabel("Count")
    plt.tight_layout()
    plt.savefig("nucleotide_distribution.png")
    plt.show()

# --- 🔬 Main ---
filename = "/content/sample_data/sequence.fasta"  # Change to your file
records = parse_file(filename)

for record in records:
    data = analyze_sequence_basic(record)
    generate_report(data)
    plot_nucleotide_counts(data["Nucleotide Counts"], data["ID"])
```

```
📄 Basic Sequence Report
ID: AH002844.2
Description: AH002844.2 Homo sapiens insulin (INS) gene, complete cds
Length: 4969 bp
Nucleotide Counts: {'A': 891, 'T': 869, 'G': 1657, 'C': 1452}
Reverse Complement (first 50 bases): TGACGAGCAGACTCTCAAAAAACAAACAAGCAAACAAACAAAAAACAAAA...
mRNA Sequence (first 50 bases): CUCGAGGGGCCUAGACAUUGCCCUCCAGAGAGAGCACCCAACACCCUCCA...
```