Genomic sequence analyser and annotator

Team 1

Shalu.S Nivetha.E Vanisha.R Hamsini.R Subraja.s.k Karen.J

INTRODUCTION

A genomic sequence analyzer and annotator is a tool used to analyze and interpret genomic data. It takes a DNA or protein sequence as input and provides various types of analysis and annotations, such as:

- *Sequence Analysis:* Calculating GC content, molecular weight, and other sequence properties.
- *Gene Prediction:* Identifying potential genes and their locations within the sequence.
- *Functional Annotation:* Assigning functional roles to predicted genes and proteins.
- *Comparative Genomics:* Comparing the sequence to other known sequences to identify similarities and differences.

```
pip install biopython
```

```
Collecting biopython
  Downloading biopython-1.85-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
Downloading biopython-1.85-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_6
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.3/3.3 MB 23.0 MB/s eta 0:00
Installing collected packages: biopython
Successfully installed biopython-1.85
```

```
from Bio.Seq import Seq

my_seq = Seq("/content/sample_data/rcsb_pdb_2HHB (1).fasta")
motif = Seq("GAT")
print(my_seq.count(motif))
```

```
0
```

```
import os
print(os.getcwd())
```

```
/content
```

```
!pip install biopython
from Bio.Seq import Seq

coding_dna = Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG")
print(coding_dna.translate())
```

```
Collecting biopython
  Downloading biopython-1.85-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
Downloading biopython-1.85-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_6
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.3/3.3 MB 27.3 MB/s eta 0:00
Installing collected packages: biopython
Successfully installed biopython-1.85
MAIVMGR*KGAR*
```

```
from Bio import SeqFeature

# Create a sequence feature
feature = SeqFeature.SeqFeature(
    location=SeqFeature.FeatureLocation(10, 20),
    type="gene",
    qualifiers={"protein": "2hhb"}
)

print(feature)
```

```
type: gene
location: [10:20]
qualifiers:
    Key: protein, Value: 2hhb
```

```
from Bio.Seq import Seq

def calculate_gc_content(sequence):
    gc_count = sequence.count('G') + sequence.count('C')
    return gc_count / len(sequence) * 100

def calculate_molecular_weight(sequence):
    # Molecular weights of nucleotides (in g/mol)
    molecular_weights = {
        'A': 331.2,
        'T': 323.2,
        'C': 307.2,
        'G': 347.2
    }
    return sum(molecular_weights[base] for base in sequence)

def analyze_sequence(sequence):
    gc_content = calculate_gc_content(sequence)
    molecular_weight = calculate_molecular_weight(sequence)
    return gc_content, molecular_weight

sequence = Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC")
gc_content, molecular_weight = analyze_sequence(sequence)
```

```
print(f"Sequence: {sequence}")
print(f"GC Content: {gc_content:.2f}%")
print(f"Molecular Weight: {molecular_weight:.2f} g/mol")
```

```
Sequence: GATCGATGGGCCTATATAGGATCGAAAATCGC
GC Content: 46.88%
Molecular Weight: 10542.40 g/mol
```

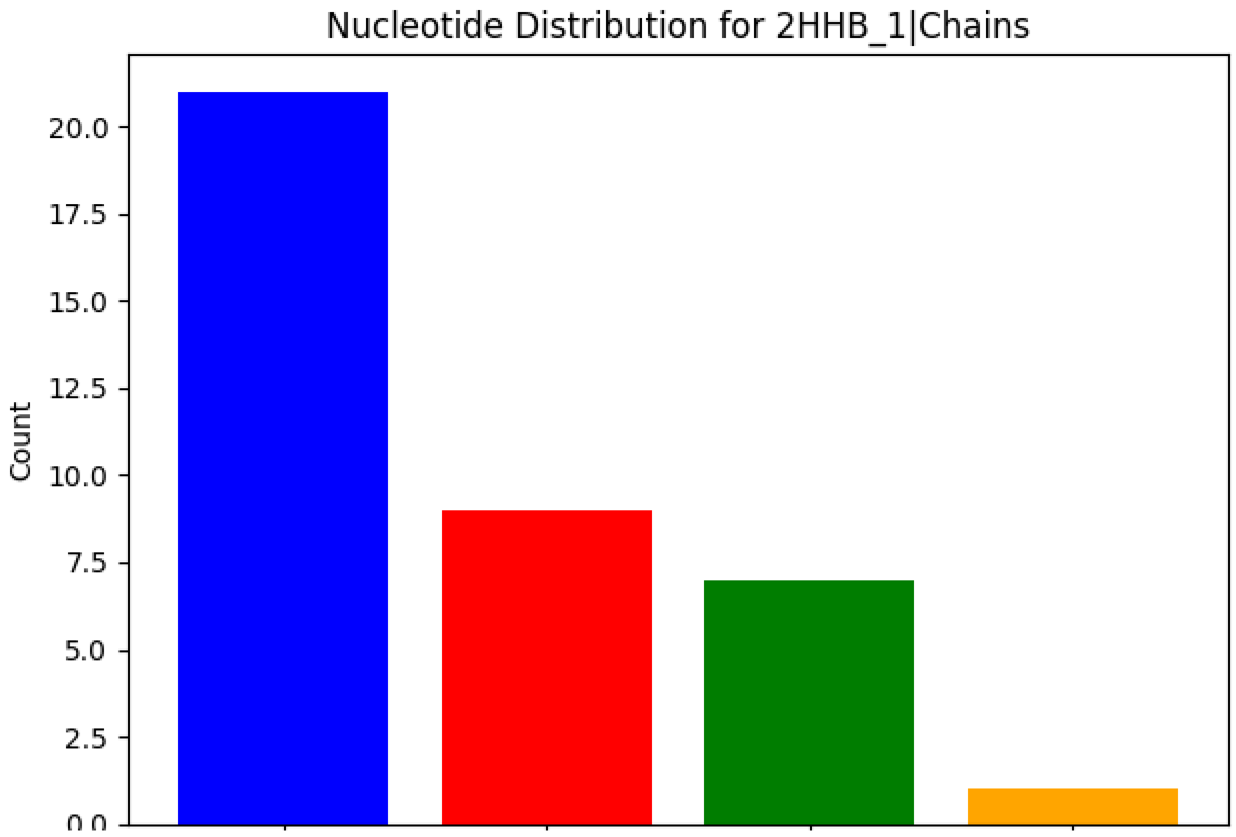
```
from Bio import SeqIO

for record in SeqIO.parse("/content/sample_data/rcsb_pdb_2HHB (1).fasta", "fasta"):
    print(record.id)
    print(record.features)
```

```
2HHB_1|Chains
[]
2HHB_2|Chains
[]
```

```
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")
```

```python
    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/rcsb_pdb_2HHB (1).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: 2HHB_1|Chains
Description: 2HHB_1|Chains A, C|HEMOGLOBIN (DEOXY) (ALPHA CHAIN)|Homo sapiens (9
Length: 141 bp
Nucleotide Counts: {'A': 21, 'T': 9, 'G': 7, 'C': 1}
Reverse Complement (first 50 bases): YRMSALBASBSTLFMHLSTDBTPAFETPLDTTLABLLGDSLLM
mRNA Sequence (first 50 bases): VLSPADKUNVKAAWGKVGAHAGEYGAEALERMFLSFPUUKUYFPHFDL
```


Nucleotide Distribution for 2HHB_1|Chains
Count
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0