Advanced Programming 2024/2025

Project Specification

Andrea Giovanni Nuzzolese

Input

The input consists of a genomic dataset in FASTA format containing the mitochondrial DNA (mtDNA) sequences of multiple species. The dataset is derived from publicly available mitochondrial genomes, such as those in GenBank.

An example of a FASTA file:

```
NC_012920.1 Homo sapiens mitochondrion, complete genome
GATCACAGGTCTATCACCCTATTAACCACTCACGGGAGCTCTCCATGCATTTGGTATTTTCGTCTGGGGG ...

NC_002008.4 Pan troglodytes mitochondrion, complete genome
GAGCCCGTCTAAACTCCTCTATGTGTCTATGTCCTTGCTTTGGCGGTTTAGCCTTCACAGATCACCAACG ...
```

The full dataset is available in virtuale.

General Goal

Develop a software solution that models the data and functionality required for analyzing mitochondrial genomes in a structured and extensible way. The program should provide a foundation for handling genomic data while supporting various operations related to mitochondrial DNA analysis.

Key objectives:

1. Parse the FASTA file and organize the data systematically for efficient access and manipulation.
2. Represent mitochondrial DNA and other genomic elements as distinct entities within the system.
3. Enable operations such as:
    - Calculating GC content and sequence statistics.
    - Identifying conserved motifs across sequences.
    - Performing pairwise sequence alignments.
    - Generating and analyzing mitochondrial DNA-derived data, such as motifs and alignment results.

Detailed Instructions

Part 1: Parsing the Input

Design and implement a robust system for parsing and managing genomic data. The program should:

- Parse the FASTA file format and represent the parsed data in a structured format (e.g., a Pandas DataFrame).

- Capture key attributes, such as sequence identifiers, descriptions, and sequences, while ensuring the data can be efficiently queried and accessed.
- Abstract the parsing logic in a way that makes it reusable for other genomic data formats.

Part 2: Genomic Representation and Analysis

Model genomic elements as well-defined components of the system. The program should include:

1. **Mitochondrial DNA Representation**:
    - Capture attributes and behaviors of mitochondrial genomes.
    - Provide methods to:
        - Extract subsequences.
        - Calculate properties such as sequence length and GC content.
2. **Genomic Motifs**:
    - Represent sequence motifs as distinct elements of the system.
    - Provide methods to:
        - Search for motifs within mitochondrial DNA.
        - Count motif occurrences and analyze their distribution across sequences.
3. **Sequence Alignment**:
    - Model pairwise sequence alignments, ensuring the implementation can handle genomic data efficiently.
    - Provide results, including alignment scores and visual representations of alignments.

Part 3: Integration of Analytical Capabilities

Introduce interactions among genomic elements to enable high-level analytical objectives:

1. Compare the sequences of different species and visualize differences in their mitochondrial DNA.
2. Identify motifs conserved across species and display their occurrence patterns.
3. Allow alignment operations that accept sequences from multiple sources and produce comparative insights.

Part 4: User Interface

Implement a web-based interface for interacting with the system. Key functionalities include:

1. Uploading a FASTA file to populate the system with genomic data.
2. Viewing statistical summaries of mitochondrial DNA, such as sequence length distributions and GC content.
3. Exploring motif occurrences and alignment results through interactive visualizations.
4. Searching for specific motifs or sequence patterns and visualizing their locations within mitochondrial genomes.

Expectations from the Design

The software should be designed as a modular and extensible system, ensuring clear separation of responsibilities between components. The following aspects should be central to the design:

- **Scalability**: The system should allow the addition of new genomic operations without requiring significant changes to existing components.
- **Reusability**: Components such as sequence parsing, motif analysis, and alignment operations should be general-purpose and reusable.
- **Interoperability**: The design should allow genomic elements to interact seamlessly to support high-level analyses.

Project Documentation

The project document must include:

1. CRC Cards and UML diagrams that describe the system design and the relationships among its components.
2. Descriptions of how genomic elements are modeled and how they interact.
3. Examples of expected input/output for each functionality.

Project Delivery

The project must be delivered as a GitHub repository and include:

1. Complete source code organized into well-structured modules.
2. Instructions for running the program and accessing its functionalities.
3. A detailed project document describing the design and implementation.

By emphasizing modularity, interaction among components, and extensibility in the project specification, the use of object-oriented principles such as inheritance (e.g., shared behaviors for different genomic elements), encapsulation (e.g., data hiding within objects), data abstraction (e.g., treating genomic sequences as high-level entities), and polymorphism (e.g., generic methods for motif searches across different sequence types) becomes a natural part of the solution design.