Basic Processing of Biological Sequences: an Object Oriented Implementation

Antonio Almeida, UP2015058365

March 6^{th}, 2019

1 Introduction

Biological sequence analysis consists of subjecting DNA (deoxyribonucleic acid), RNA (ribonucleic acid), or peptide sequences to various analytical methods in order to understand its features. This project's goal was to implement a Python package with a reusable set of features focused on basic biological processing of DNA, RNA, and Protein sequences, with an Object Oriented architecture in mind.

2 Architecture and Implementation Decisions

As requested by the project's specification, the package's architecture has an Object Oriented approach in mind. Initially, the mains concerns were how to cover the requested feature set while minimizing code duplication, and still considering that some features should only be available to a particular sequence type, e.g, *transcription* only makes sense on DNA sequences (Transcription (biology), 2019). This quickly led to the main class, `BioSeq`, becoming *abstract*, leaving most implementations to its subclasses, as described below.

For RNA and DNA sequences, an intermediate class that encapsulates the common features was introduced - `DNRACommon`. This helped increase code re-usability and, alongside the final class implementations - `DNASeq` and `RNASeq`, permitted a clear separation of what is exclusive (or not) to both DNA and RNA sequences.

For amino acid and protein sequences, considering the same goal of code re-usability, two classes were developed: `AminoacidSeq`, encapsulates features related to generic amino acid sequences, e.g., computing its possible proteins; and `ProteinSeq`, to encapsulate features specific to possible protein sequences (in the packages' context), i.e., *open reading frames*. This class was introduced with future extensibility in mind, as it does not implement any extra feature. However, `ProteinSeq` is a subclass of `AminoacidSeq`, thus inheriting all its features.

It's worth noting that the tool attempts to replicate the behavior of operations that biologically generate a new type of sequence, e.g., DNA *transcription* creates a RNA sequece. Following this example, the result of `dna.transcription()`, with `dna` being an instance of `DNASeq`, results in an instance of `RNASeq`. This format is repeated on similar operations.

3 Results

The following features have been implemented:

- Support for DNA, RNA, and Protein sequences

- Generic sequence features:
  - Displaying a summary of a sequence's information, i.e., "pretty printing"
  - Sequence validation, according to its type
  - Computing *symbol frequency*
  - Save/Load sequence state from/to a file
  - Load sequences from a FASTA formatted file (Yangzhag Lab, 2010)
- Features for DNA and RNA sequences:
  - Computing *GC-content*
  - Computing *reverse complement*
  - Computing *translation*
  - Computing *codon usage*
  - Computing *reading frames*
  - Computing *open reading frames*, i.e., all putative proteins of a DNA or RNA sequence
- Features specific to DNA sequences:
  - Computing *transcription*
- Features specific to RNA sequences:
  - Computing *reverse transcription*
- Features specific to Protein/Amino acid sequences:
  - Computing possible proteins

4 Conclusions

An Object Oriented approach to this set of features permits an intuitive relationship between the biological concepts and their counterparts on the tool, allowing for an easy user adaptation, provided enough knowledge of molecular biology.

Additionally, even though Python isn't notorious for its Object Oriented features, it allowed for a good enough development experience, and I believe this architecture is scalable and extensible enough for a more serious tool.

Finally, I believe my knowledge on the subject of molecular biology and sequence processing has significantly increased throughout the development of this project, and I'm happy with the results.

References

Transcription (biology). (2019). *Transcription (biology).* Retrieved from `https://en.wikipedia.org/wiki/Transcription_(biology)` ([Online; accessed 6-March-2019])

Yangzhag Lab. (2010). *What is fasta format?* Retrieved from `https://zhanglab.ccmb.med.umich.edu/FASTA/` ([Online; accessed 6 March 2019])