The Academic
International Journal of Multidisciplinary Research

An Online Peer Reviewed / Refereed Journal
Volume 4 | Issue 9 | September 2026
ISSN: 2583-973X (Online), SJIF IF(2026): 5.6
Website: www.theacademic.in

Application of Biopython for Analysis of Plant DNA, RNA and Protein Sequences

Harisharanappa S. Padsalgi

Gulbarga University, Kalaburagi Department of Computer Science, Sharnbasveshwar College of Science

Email: spharish1974@gmail.com

DOI : https://doi.org/10.5281/zenodo.23773088

ARTICLE DETAILS

Research Paper

Received: 28-08-2026

Accepted: 22-09-2026

Published: 20-10-2026

Keywords:

Biopython, plant bioinformatics, DNA sequence, RNA analysis,protein sequence, sequence alignment

ABSTRACT

Bioinformatics supports the analysis of biological data by combining biology, computer science and statistics. Biopython is an open-source Python library that provides tools for representing biological sequences, processing sequence files, transcribing and translating sequences, performing pairwise alignment, and accessing biological databases. This paper reviews the application of Biopython to plant DNA, RNA and protein sequence analysis and presents an illustrative computational workflow. Python examples demonstrate nucleotide counting, GC-content calculation, DNA transcription, translation of a coding sequence, protein composition analysis, pairwise alignment and FASTA-file processing. Potential applications in plant identification, genetic diversity studies, crop improvement, transcriptomics and comparative genomics are discussed alongside limitations and requirements for reproducibility. The examples use teaching sequences and are not presented as experimental observations from actual plants. Biopython can make routine sequence-processing tasks more accessible and repeatable, but biological conclusions depend on the quality and provenance of input data, appropriate analytical settings and independent validation where necessary.

Disclaimer: The opinions, interpretations, conclusions, recommendations, analyses, and statements expressed in this research article are solely those of the author(s) and do not reflect the views, policies, or positions of the Publisher, Chief Editor, Editors, Editorial Board, Reviewers, or their affiliated institutions. The author(s) are solely responsible for ensuring the originality, authenticity, and integrity of the manuscript and for ensuring that it is free from plagiarism, AI-generated content, copyright infringement, data falsification, and any other form of academic misconduct.

1. Introduction

Plants are fundamental to food security, medicine, ecosystem stability and the economy. Research into plant genetics and molecular biology helps explain species relationships, inherited traits, adaptation to environmental stress and the basis of crop improvement. DNA, RNA and proteins provide complementary views of biological systems: DNA stores hereditary information, RNA participates in gene expression and protein synthesis, and proteins perform structural, catalytic, transport and regulatory functions.

The growth of sequencing technologies has made large quantities of biological sequence data available. Manual processing is inefficient when researchers need to compare many sequences, count nucleotides, inspect coding regions or organize records from public databases. Bioinformatics addresses these challenges by applying computational methods to biological information. Python is useful in this area because it has readable syntax, a large scientific ecosystem and supports automation.

Biopython is an open-source collection of Python tools for computational biology. Its modules support biological sequence objects, sequence file input/output, alignment, phylogenetic data and access to resources such as the NCBI Entrez system. For plant research, Biopython can help process DNA barcodes, examine candidate genes, prepare RNA or protein sequences for downstream analysis and compare homologous sequences. It does not replace experimental methods; rather, it provides a reproducible way to handle and inspect data generated experimentally or retrieved from public resources.

This paper presents a methodological overview of Biopython applications for plant DNA, RNA and protein sequences. It combines explanations with short illustrative programs and discusses practical uses, limitations and good reporting practice. The example sequences are computational demonstrations, not sequences claimed to have been collected from a named plant species.

2. Objectives

The objectives of this paper are: (i) to introduce Biopython and its relevance to plant bioinformatics; (ii) to demonstrate basic DNA sequence characterization, including length, nucleotide counts and GC content; (iii) to explain DNA-to-RNA transcription and coding-sequence translation; (iv) to illustrate protein composition analysis and pairwise alignment; (v) to describe potential applications in taxonomy, genetic diversity, crop research and transcriptomics; and (vi) to identify limitations and reproducibility practices needed for responsible interpretation.

3. Review of Literature

Computational analysis has become a standard component of molecular biology as biological databases and sequence technologies have expanded. Chapman and Chang (2000) described Biopython as a set of Python tools for computational biology, helping make routine molecular-data operations reusable rather than requiring each researcher to write new code. Cock et al. (2009) later documented the Biopython project, its freely available tools and its role in computational molecular biology and bioinformatics.

Sequence comparison is another foundational method. The Basic Local Alignment Search Tool (BLAST), introduced by Altschul et al. (1990), enabled researchers to identify local similarities between nucleotide or protein sequences. Similarity searches can help identify candidate homologues, but they do not alone establish biological function. Sequence databases such as GenBank and the NCBI Protein database provide records that may include sequence accessions, organism names and annotations. Researchers should inspect these details and cite accession numbers when using public data.

In plant science, DNA barcoding, gene comparisons and protein analysis contribute to species identification, studies of genetic variation and investigation of molecular evolution. Biopython can support parts of these workflows by handling records, transforming sequences and performing pairwise alignment. More advanced studies may require dedicated tools for multiple sequence alignment, phylogenetic inference, read mapping, gene-expression analysis or population genetics. Thus, Biopython is best viewed as a flexible toolkit within a broader analytical workflow.

4. Materials and Methods

4.1 Study design. This manuscript is a methodological review with illustrative computational examples. No laboratory samples were collected and no new empirical dataset was analysed for this paper. The sequences in the code examples are short teaching sequences chosen to demonstrate how Biopython functions. Their output must not be interpreted as measurements from real plant samples.

4.2 Software. The examples require Python and Biopython. Installation can be performed from a command prompt using `python -m pip install biopython`. The installed version can be checked with `import Bio; print(Bio.__version__)`. A researcher should record the operating system, Python version, Biopython version and relevant settings when preparing a reproducible study.

4.3 Sequence sources and formats. Real research can use sequences generated in a laboratory or downloaded from curated public databases. Each sequence should have an identifiable source, accession number where

available, organism name and description of the region represented. FASTA is a widely used format for storing nucleotide and protein sequences. Before analysis, researchers should check sequence length, ambiguous symbols, orientation, completeness and whether the sequence is genomic DNA, a coding region, RNA or protein.

4.4 DNA sequence characterization. Basic characterization includes sequence length, nucleotide frequencies and GC content. For a sequence containing only A, T, G and C, GC content is calculated as (G + C) divided by total sequence length, multiplied by 100. Ambiguous bases require an explicit policy; otherwise, the denominator can distort the percentage. GC content is descriptive and should not be treated as a stand-alone indicator of species identity or gene function.

4.5 RNA and protein processing. Biopython provides methods for transcription and translation. Transcription converts a DNA coding-strand representation into the corresponding RNA representation, replacing T with U. Translation reads nucleotide triplets (codons) according to a genetic code. The correct strand, reading frame, coding region and genetic code must be selected. Plant organelles can use genetic codes that differ from the standard code. Internal stop codons or incomplete codons can indicate that the input is not a complete coding sequence or that the frame needs checking.

4.6 Sequence alignment. Pairwise alignment compares two sequences according to a selected algorithm and scoring scheme. Global alignment compares sequences over their full lengths; local alignment identifies the best matching region. Match, mismatch and gap scores affect the output and must be reported. Alignment scores are not automatically equivalent to percentage identity. For broader evolutionary questions, researchers may need multiple sequence alignment and suitable phylogenetic methods.

5. Illustrative Biopython Programs

The following examples demonstrate common operations. They should be run with Biopython installed. They are intentionally small for teaching and do not constitute a completed plant-species study.

5.1 DNA length, nucleotide counts and GC content

```
from Bio.Seq import Seq
dna = Seq("ATGCGTACG")
sequence = str(dna).upper()
print("DNA:", dna)
print("Length:", len(dna))
```

```
for base in "ATGC":
    print(base, ":", sequence.count(base))
gc = (sequence.count("G") + sequence.count("C")) / len(sequence) * 100
print("GC content: {:.2f}%".format(gc))
```

OUT-PUT(Result from Python)

DNA: ATGCGTACG

Length: 9

A : 2

T : 2

G : 3

C : 2

GC content: 55.56%

The sequence contains nine nucleotides: A = 2, T = 2, G = 3 and C = 2. GC content is therefore (3 + 2)/9 × 100 = 55.56%. This calculation is valid because all characters are standard DNA bases. For real records, ambiguous symbols should be checked before calculating the percentage.

5.2 Complement and reverse complement

```
from Bio.Seq import Seq
dna = Seq("ATGCGTACG")
print("Complement:", dna.complement())
print("Reverse complement:", dna.reverse_complement())
```

OUT-PUT

Complement: TACGCATGC

Reverse complement: CGTACGCAT

The complement pairs A with T and G with C. The reverse complement also reverses the complementary sequence and is useful when a sequence may be represented in the opposite strand orientation. Strand orientation must still be interpreted in the context of the gene or marker being studied.

5.3 Transcription and translation

```
from Bio.Seq import Seq
dna = Seq("ATGGCTGCTGCTTAA")
```

```
rna = dna.transcribe()
protein = dna.translate()
print("DNA:", dna)
print("RNA:", rna)
print("Protein:", protein)
```

OUT-PUT

DNA: ATGGCTGCTGCTTAA

RNA: AUGGCUGCUGCUUAA

Protein: MAAA*

For this example, the RNA sequence is `AUGGCUGCUGCUUAA`, and translation under the standard genetic code produces `MAAA*`. The asterisk represents a stop codon. The example is a deliberately selected coding sequence. In real analysis, a predicted translation must be checked against the biological source, reading frame and coding-sequence boundaries.

5.4 Protein composition

```
from Bio.Seq import Seq
from collections import Counter
protein = Seq("MAAAGGTT")
print("Length:", len(protein))
print("Composition:", Counter(str(protein)))
```

OUT-PUT

Length: 8

Composition: Counter({'A': 3, 'G': 2, 'T': 2, 'M': 1})

This program reports a sequence length of eight amino acids and counts each amino-acid symbol. Composition alone does not reveal a protein's three-dimensional structure, activity or cellular role; those questions require additional evidence.

5.5 Reading a FASTA file

```
from Bio import SeqIO
for record in SeqIO.parse("plant_sequences.fasta", "fasta"):
```

```
print("ID:", record.id)
print("Description:", record.description)
print("Length:", len(record.seq))
```

The file `plant_sequences.fasta` must exist in the working directory and contain one or more FASTA records. `SeqIO.parse` is suitable for reading multiple records. For a file expected to contain exactly one record, `SeqIO.read` can be used and will report an error if the file does not contain exactly one record.

5.6 Pairwise alignment

```
from Bio import Align
seq1 = "ATGGCTGCTGCT"
seq2 = "ATGGCCGCTGCT"
aligner = Align.PairwiseAligner()
aligner.mode = "global"
aligner.match_score = 1
aligner.mismatch_score = -1
aligner.open_gap_score = -2
aligner.extend_gap_score = -0.5
alignments = aligner.align(seq1, seq2)
print("Alignment score:", alignments.score)
print(alignments[0])
```

OUT-PUT

```
Alignment score: 10.0
target          0 ATGGCTGCTGCT 12
                0 |||||.|||||| 12
query           0 ATGGCCGCTGCT 12
```

The sequences differ at one nucleotide position. The score depends on the specified scoring scheme. For a scientific comparison, researchers should report the alignment method and parameters and calculate percentage identity separately if it is required.

6. Discussion

The examples show how a small number of reusable commands can reduce repetitive manual work and make sequence-processing steps easier to reproduce. DNA length and nucleotide counts provide basic quality and composition information; transcription and translation help inspect candidate coding sequences; FASTA processing supports batch analysis; and alignment identifies differences between sequences under defined parameters.

These operations may contribute to plant identification and molecular taxonomy when applied to suitable barcode regions such as `rbcL` or `matK` and compared with reliable reference records. Comparing documented accessions can support genetic-diversity and conservation studies. Candidate-gene comparisons may contribute to crop-improvement research, while RNA and protein sequence handling can form part of larger transcriptomic or protein-characterization workflows.

The results must be interpreted cautiously. GC content alone does not identify a species, a high similarity score does not prove identical function, and computational transcription does not demonstrate gene expression in a particular tissue. Sequence errors, contamination, incorrect labels, wrong strand orientation, unsuitable reading frames and inappropriate genetic codes can produce misleading outputs. Larger studies may need specialized tools for quality control, mapping, multiple sequence alignment, statistical analysis and phylogenetic inference.

For reproducibility, researchers should report data sources and accession numbers, retrieval dates, sequence-selection criteria, software versions, quality-control decisions, ambiguous-base treatment, alignment methods and scoring parameters. Code and sufficient instructions should be shared where possible. Any study making biological claims should use actual, traceable data and validate interpretations with appropriate evidence.

7. Results

The illustrative workflow demonstrates that Biopython can automate basic sequence characterization, transcription, translation, protein-symbol counting, FASTA reading and pairwise comparison. These are demonstrations of software functions only. No conclusions about a plant species, population, gene expression or protein function can be drawn from the teaching sequences.

8. Conclusion

Biopython offers an accessible and flexible framework for processing plant DNA, RNA and protein sequences. Its tools for sequence manipulation, file handling, transcription, translation and pairwise

alignment can support preliminary molecular characterization, comparative analysis and bioinformatics education. The illustrative programs in this paper show how common operations can be automated and documented. Applications include plant identification, genetic diversity research, crop improvement, transcriptomics and protein comparison. However, meaningful conclusions require reliable sequence data, suitable analytical choices and careful biological interpretation. Future plant-specific studies using documented accessions and actual computational results can demonstrate the value of Biopython more directly for a particular research question.

9. References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. Journal of Molecular Biology, 215(3), 403–410. https://doi.org/10.1016/S0022-2836(05)80360-2

Chapman, B. A., & Chang, J. T. (2000). Biopython: Python tools for computational biology. ACM SIGBIO Newsletter, 20(2), 15–19.

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., & de Hoon, M. J. L. (2009). Biopython: Freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics, 25(11), 1422–1423. https://doi.org/10.1093/bioinformatics/btp163