

# BIO2502 生物计算编程语言课程项目开题报告

## 项目名称: bioseqkit —— 基础序列处理工具包

### 一、项目背景与生物学意义

生物序列 (DNA、RNA、蛋白质) 是生物信息学最基本的研究对象。无论是基因组组装、变异检测, 还是转录组分析、蛋白质功能预测, 所有下游分析都依赖于对序列文件 (FASTA/FASTQ) 的读取、统计与预处理。然而, 许多研究人员在处理大规模测序数据时, 往往直接调用 Biopython 等高层封装库, 缺乏对序列解析底层逻辑、流式处理 (streaming/iterator) 和索引机制的理解。

本项目 bioseqkit 旨在从零构建一个轻量级、可扩展的 Python 序列处理工具包。通过手工实现 FASTA/FASTQ 解析器、序列统计、k-mer 分析与命令行接口 (CLI), 深入理解生物信息学数据 I/O 的核心设计模式, 同时产出可用于真实数据分析的可复用软件包。

生物学应用场景包括:

- 对新测序的细菌基因组进行 GC 含量和 k-mer 组成分析, 辅助物种鉴定;
- 批量提取基因间区序列并计算六框翻译, 初步筛选候选 ORF;
- 为下游机器学习流程提供统一的序列特征提取接口。

### 二、项目意向

本项目选择「Part I: 经典生物信息学 Python Package 项目」中的 “**Project 1: bioseqkit**”。

选择理由:

1. 从 Python 软件工程角度看, 该项目涉及文件 I/O 设计、迭代器模式、面向对象封装与 CLI 构建, 是学习 package 开发的最佳入门点;
2. 从生物信息学角度看, 该项目覆盖了最基础也最重要的序列操作, 成果可复用于后续任何生信 workflow;
3. 项目难度属于 Beginner 级别, 但扩展空间充足 (多线程、minimizer、序列索引);

### 三、项目计划

#### 3.1 最低交付要求

##### 3.1.1 核心功能模块

1. FASTA 解析器: 支持普通文本与 gzip 压缩文件, 纯 Python 实现 (不使用 Biopython 的 SeqIO.parse), 采用 iterator/generator 模式处理大文件;
2. FASTQ 解析器: 同样支持 gzip, 实现 Phred 质量分数的正确解析;
3. 基础统计: 序列长度分布、GC 含量、N 碱基比例、碱基组成矩阵;

4. 反向互补: DNA 序列的 reverse complement 操作;
5. 序列翻译: 六框翻译 (3 个正向 reading frame + 3 个反向互补 frame), 支持标准遗传密码表;
6. k-mer 频率统计: k-mer 计数、top-k k-mer 输出, 支持 canonical k-mer;
7. CLI 命令行入口 (通过 click 或 argparse 实现):
  - `bioseqkit stats input.fa` — 序列统计
  - `bioseqkit revcomp input.fa` — 反向互补
  - `bioseqkit translate input.fa` — 六框翻译
  - `bioseqkit kmer input.fa -k 5` — k-mer 分析

### 3.1.2 单元测试

使用 `pytest` 编写至少 5 个单元测试, 覆盖:

1. 标准单序列 FASTA 解析
2. 多行序列 FASTA 解析
3. 空行及非法字符边界情况
4. 反向互补正确性 (与已知结果比对)
5. 六框翻译的 reading frame 坐标正确性

### 3.1.3 Jupyter Notebook 演示

使用真实小片段数据 (如 E. coli K-12 基因组或人线粒体 chrM 序列), 展示:

1. 数据导入与基本统计
2. GC 含量与序列长度分布图 (Matplotlib/Seaborn)
3. k-mer 频谱 (k-mer spectrum) 可视化
4. 六框翻译结果与已知 CDS 的一致性比较

### 3.1.4 软件工程规范

1. src-layout 项目结构, 含 `pyproject.toml`
2. 提供 `README.md` (背景、安装、示例、数据来源)
3. 提供 `environment.yml` 或 `requirements.txt`
4. 支持 `pip install -e .` 安装

## 3.2 进阶交付要求

### 3.2.1 多线程 k-mer 计数

使用 Python `concurrent.futures` 或 `multiprocessing` 模块实现并行 k-mer counting。对输入序列进行分块, 每个线程独立统计其分块的 k-mer, 最终合并计数表。通过对比单线程与多线程在大基因组 (如人类 chr1) 上的运行时间, 展示加速比与 scalability 曲线。

### 3.2.2 Minimizer 序列抽样

实现 minimizer 算法: 对给定序列, 通过滑动窗口选择每个窗口内字典序最小的 m-mer 作为 minimizer。该技术广泛用于序列比对 (minimap2) 与基因组

sketching (Mash)。提供参数可调的窗口大小 (w) 与 k-mer 大小 (k)，可视化 minimizer 在序列上的分布。

### 3.2.3 FAI-like 序列索引

仿照 samtools faidx 的索引格式 (\*.fai)，实现 FASTA 文件的随机访问索引。索引文件记录每条序列的名称、长度、在原始文件中的起始偏移量和每行的碱基数/字节数。通过索引，支持 bioseqkit fetch input.fa chr1:1000-2000 子序列快速提取，无需遍历整个文件。

### 3.2.4 GitHub Actions 自动测试

在 GitHub 仓库中配置 CI workflow，每次 push 自动运行 pytest 测试套件与 ruff 代码风格检查，确保代码质量持续可控。

### 3.2.5 Sphinx 文档

使用 Sphinx + reStructuredText 或 Markdown 构建 API 文档，部署至 GitHub Pages，包含模块文档字符串与快速入门教程。

### 3.2.6 与真实数据库或公开数据集对接

通过 bioseqkit 内置的数据下载模块，直接从 NCBI Entrez API 获取参考序列，自动完成解析与统计，形成「数据获取 — 处理 — 分析」一键式 workflow。

### 3.2.7 Benchmark 与复杂度分析

对核心功能进行时间复杂度分析（如 k-mer 计数为  $O(nk)$ 、序列索引构建为  $O(n)$  等），并使用 timeit 或 pytest-benchmark 在多个基因组规模上测量实际运行时间与内存占用，绘制 scalability 曲线。

## 四、数据来源

- NCBI Nucleotide: <https://www.ncbi.nlm.nih.gov/nucleotide/>
- UCSC Genome Browser: <https://genome.ucsc.edu/>

## 五、技术路线

软件架构采用 src-layout 模块化设计：

```
bioseqkit/  
  pyproject.toml  
  README.md  
  LICENSE  
  src/  
    bioseqkit/  
      __init__.py  
      io.py  
      stats.py  
      transform.py  
      kmer.py  
      index.py
```

```
cli.py
tests/
  test_io.py
  test_stats.py
  test_transform.py
  test_kmer.py
  test_cli.py
examples/
  demo.ipynb
  example_data/
docs/
  index.rst
environment.yml
```

核心数据结构:

1. FastaRecord: namedtuple 或 dataclass, 包含 id, description, sequence
2. FastqRecord: namedtuple 或 dataclass, 包含字段同上, 外加 quality

核心设计原则:

1. I/O 层使用 generator 避免全量加载内存;
2. 统计算法使用标准库, 不引入 Biopython 等生信专用依赖;
3. CLI 层使用 click 库提供友好的参数解析与帮助信息。

## 六、实施时间线

1. 前期: 环境搭建、数据下载、核心 I/O 模块开发;
2. 中期: 算法模块完善、CLI 构建、进阶功能开发与 notebook 演示;
3. 后期: 测试与性能评估、文档撰写与可复现环境整理;
4. 末期: 结果解释、报告撰写。

## 七、预期成果

1. 一个遵循 Python 软件工程规范的 bioseqkit package, 可通过 PyPI 安装;
2. 至少 5 个覆盖核心模块的 pytest 单元测试;
3. 一份完整的 Jupyter Notebook 演示, 使用真实细菌基因组数据展示序列统计、k-mer 分析与翻译结果;
4. CLI 工具可独立运行, 支持 stats、revcomp、translate、kmer 等子命令;
5. 多线程 k-mer 计数与单线程的 benchmark 对比曲线;
6. minimizer 算法的完整实现及其在序列长度/sketching 中的应用演示;
7. FAI-like 索引实现, 支持子序列快速随机提取;
8. GitHub Actions CI 自动化测试通过;
9. Sphinx 生成的 API 文档。