# 山东大学软件学院 2018 级

# 《项目实训》课程报告

| 导师签字 | |
|---|---|

同意参答辩

班级：18 级大数据

学号：201800301328

姓名：汪超

2021 年 6 月 3 日

# 目录

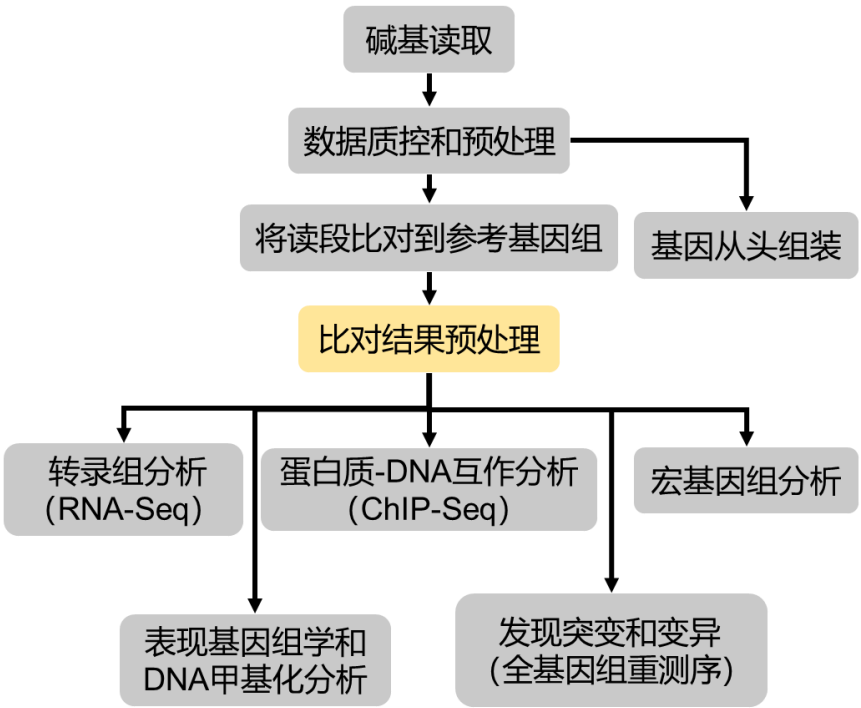
# 多核平台下测序数据比对结果质量控制软件的研究与优化

## 一、项目背景

BAM 格式是目前基因数据分析中最通用的比对结果存储格式，它既适合于短序列,也适合于长序列,也是最权威，使用最广的 NGS 数据比对软件 BWA 的标准输出文件。BAM 文件被广泛的用于生物信息学的相关领域。需要对测序数据比对结果中涉及到的生物相关知识了解，确保结果正确。一开始它的名字是 SAM（The Sequencing Alignment/Map Format 的英文简称），第一次出现的时候，它是 bwa 比对软件的标准输出文件，但原生的 SAM 是纯文本文件，十分巨大（比如：一个人 30x 全基因组测序的 SAM 大小超过 600G），非常容易导致存储空间爆满! 为了解决这个问题，bwa 的开发者李恒设计了一种比 gzip 更加高效的压缩算法，对其进行压缩，这就是当前 BAM 文件格式，它的文件大小差不多只有相应 SAM 格式的 1/6。

同时测序数据比对结果的质量控制是保证下游分析结果正确的保障，测序数据比对结果的质量高低决定着分析是否有意义。快速完成对测序数据比对结果的质量控制，能够保证整个分析流程的快速有效的进行[1]。

测序数据数据分析流程概览



# 二、项目完成过程

## 1. BAM 文件的信息提取与文件转化以及性能优化

### 目标简述

BAM 格式是目前基因数据分析中最通用的比对结果存储格式，但由于 BAM 文件高度压缩，故在进行质量控制的时候需要提取其有用的信息。参考现广泛使用的 Samtools 和 Fastqc 工具，完成 BAM 文件的信息提取及质量控制，并在多核平台进行性能优化。

### 完成过程

#### SAM/BAM 文件内容解析

当我们测序得到的 fastq 文件 map 到基因组之后，会得到一个以 SAM 或 BAM 为扩展名的文件。而 BAM 就是 SAM 的二进制文件，也就是压缩格式的 SAM 文件。参考[2],我们可以得到 SAM 文件的信息如下:

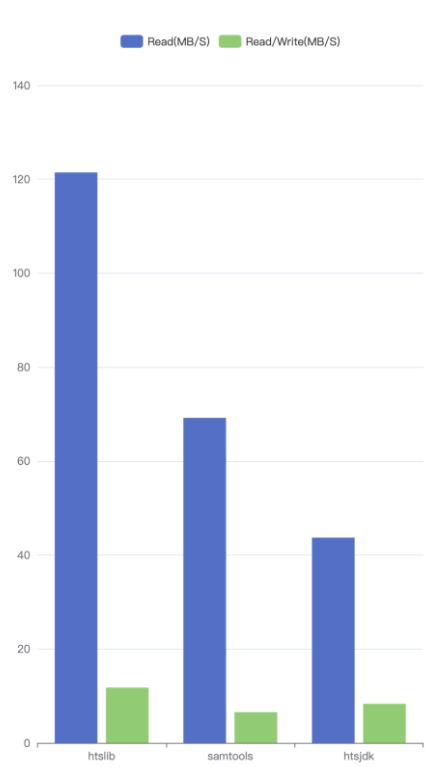
| Col | Field | Type | Regexp/Range | Brief description |
|---|---|---|---|---|
| 1 | QNAME | String | [!-?A-~]{1,254} | Query template NAME |
| 2 | FLAG | Int | [0,$2^{16}$-1] | bitwise FLAG |
| 3 | RNAME | String | *\|[!-()+-<>-~][!-~]* | Reference sequence NAME |
| 4 | POS | Int | [0,$2^{31}$-1] | 1-based leftmost mapping POSition |
| 5 | MAPQ | Int | [0,$2^{8}$-1] | MAPping Quality |
| 6 | CIGAR | String | *\|([0-9]+[MIDNSHPX=])+ | CIGAR string |
| 7 | RNEXT | String | *\|=\|[!-()+-<>-~][!-~]* | Ref. name of the mate/next read |
| 8 | PNEXT | Int | [0,$2^{31}$-1] | Position of the mate/next read |
| 9 | TLEN | Int | [-$2^{31}$+1,$2^{31}$-1] | observed Template LENgth |
| 10 | SEQ | String | *\|[A-Za-z=.]+ | segment SEQuence |
| 11 | QUAL | String | [!-~]+ | ASCII of Phred-scaled base QUALity+33 |

而将 SAM 文件压缩过后的 BAM 文件并非是一整个压缩块，结构如下:

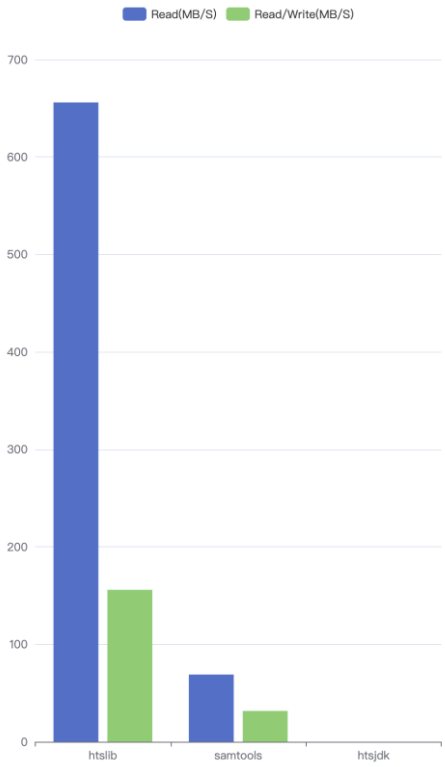

这就意味着 BAM 文件是分块读取的，且每块大小不大于 64KB，需对较小块进行频繁的解压缩。

#### 现有软件的效率

虽然已有开源软件，例如 samtool, htslib 等支持 BAM 文件的解析，但是其效率较差。相对于后续处理来说 BAM 预处理过程，BAM 解析较为耗时，是主要的性能瓶颈。其中 samtools 和 htslib 在不同线程下的运行情况如下:




单线程的解析



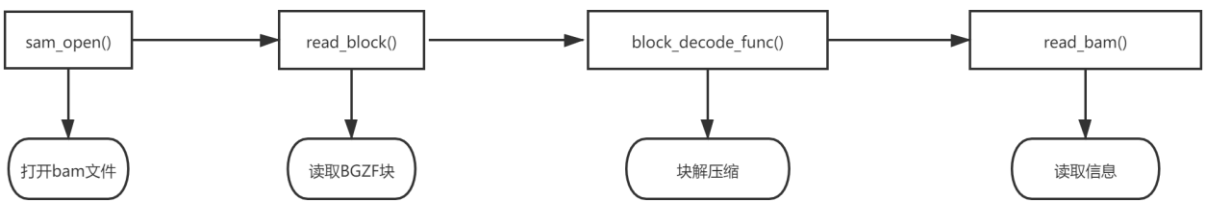

16 线程的解析

可以看见在多核平台上程序解析的加速比并不理想, 所以需要新设计文件的解析和多线程的优化算法，这也是该部分工作的主要内容。

## Htslib 库的使用（BAM 的信息提取）

首先我们要将 BAM 的信息提取部分分离出来。htslib 是一个处理高通量数据通用文件格式的库，是 samtools 软件和 bcftools 软件依赖的核心库。所以通过学习 htslib 的源码[3],从而编写能够多线程读取 BAM 文件的接口。

但是 htslib 的官网上并没有完善详尽的说明文档,所以这一步就走的比较困难。首先 BAM 文件的压缩格式是分块的，而 htslib 库对 BAM 文件的读取与解压缩也是以 BAM 的分块单元 BGZF 为单位进行的，所以在某个 BGZF 块上，进行读取与解压缩。下面是阅读完源码以后的信息提取的算法流程图。



尽管逻辑简单，但是 BAM 文件内容细节繁琐，从阅读源码一个个看参数到最后提取信息的解析与输出部分也挺艰难。

## 多线程算法设计(重点)

### 内存池

Htslib 库解析 BAM 文件时也是对单个 BGZF 分块进行操作，但是它是优先把所有的 BGZF 分块进行读取存储，然后依次进行解压缩与信息的提取，这样做会因为存在大量创建存储 BGZF 块数据结构的操作，内存碎片化严重，导致最后多线程的效率低下，于是我们引入内存池的概念。

通过创建一系列用于存储 BGZF 块的数据结构，减少创建操作和内存碎片化，同时线程利用指针和内存池进行通信和获取内存，减少内存拷贝，降低线程通信代价。下面介绍最主要的数据结构 BamBlock 类。

```
class BamBlock
{
public:
    BamBlock();
    BamBlock(BamBlockConfig *config);
    pair<bam_block *, int> getEmpty(); //提取一个空的内存块
    void inputblock(int id); // 导入未解压的数据
    pair<bam_block *, int> getCompressdata(); //提取一个填入了压缩数据的内存块
    void backempty(int id); //归还一个已经解析完的内存块，
    bool isComplete(); //是否读取结束了
    void ReadComplete(); //读取结束

public:
    BamBlockConfig *config;
    mutex mtx_read;
    mutex mtx_compress;
    bam_block **buffer; //内存块
    int *compress; //压缩数据内存块id数组 (循环数组)
    int compress_bg; //头指针
    int compress_ed;//尾指针
    int *read; //空内存块的id数组,其它同上
    int read_bg;
    int read_ed;
};
```

首先创建 buffer 内存池，存放着用于存储 bgzf 分块的内存，再使用两个 int 数组分别保存着内存池中不同状态（是否使用）的内存 id，最后创建线程锁，保证 int 数组的更新操作具有原子性。

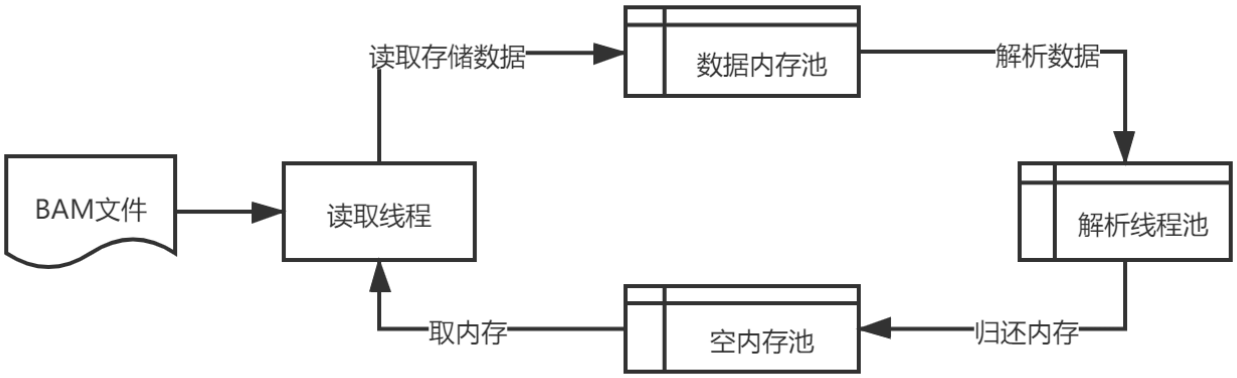
生产者-消费者模型

生产者消费者问题，也称有限缓冲问题，是一个多线程同步问题的经典案例。该问题描述了共享固定大小缓冲区的两个线程——即所谓的“生产者”和“消费者”——在实际运行时会发生的问题。生产者的主要作用是生成一定量的数据放到缓冲区中，然后重复此过程。与此同时，消费者也在缓冲区消耗这些数据。该问题的关键就是要保证生产者不会在缓冲区满时加入数据，消费者也不会在缓冲区中空时消耗数据。

而我们的这个问题恰好可以用上述生产者消费者模型来解决，考虑到 BAM 文件的 BGZF 块读取远快于文件的解析，所以我们安排一个线程来进行读取操作，另外的线程进行消费者的工作，即文件的解析

。

整个算法的流程图如下：

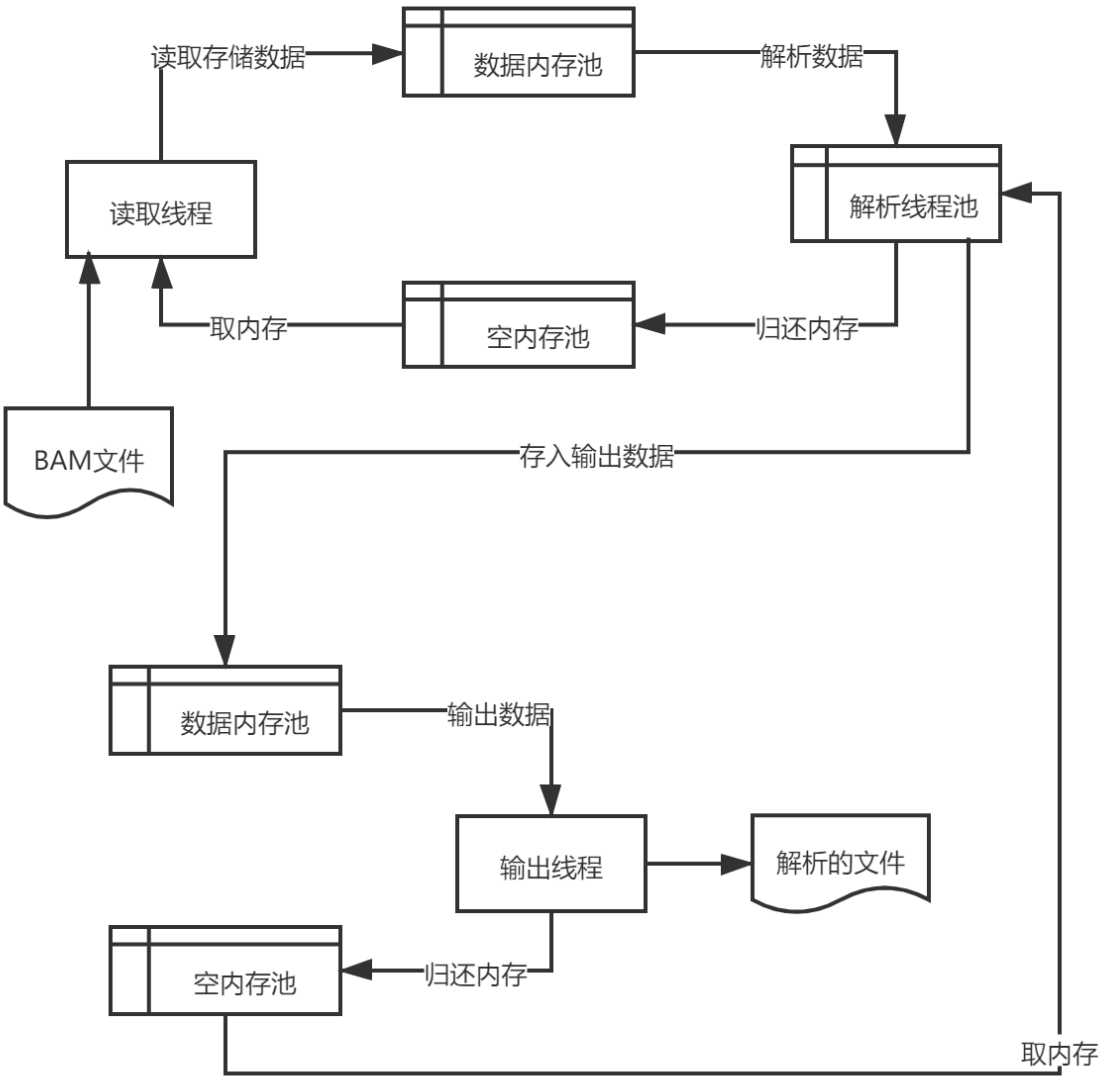



缓冲区输出

本来可以在解析数据完以后一起输出信息，但是一联想到生产者消费者模型，就发现其实可以在输出的部分利用这个思想来做到更高效的表现。

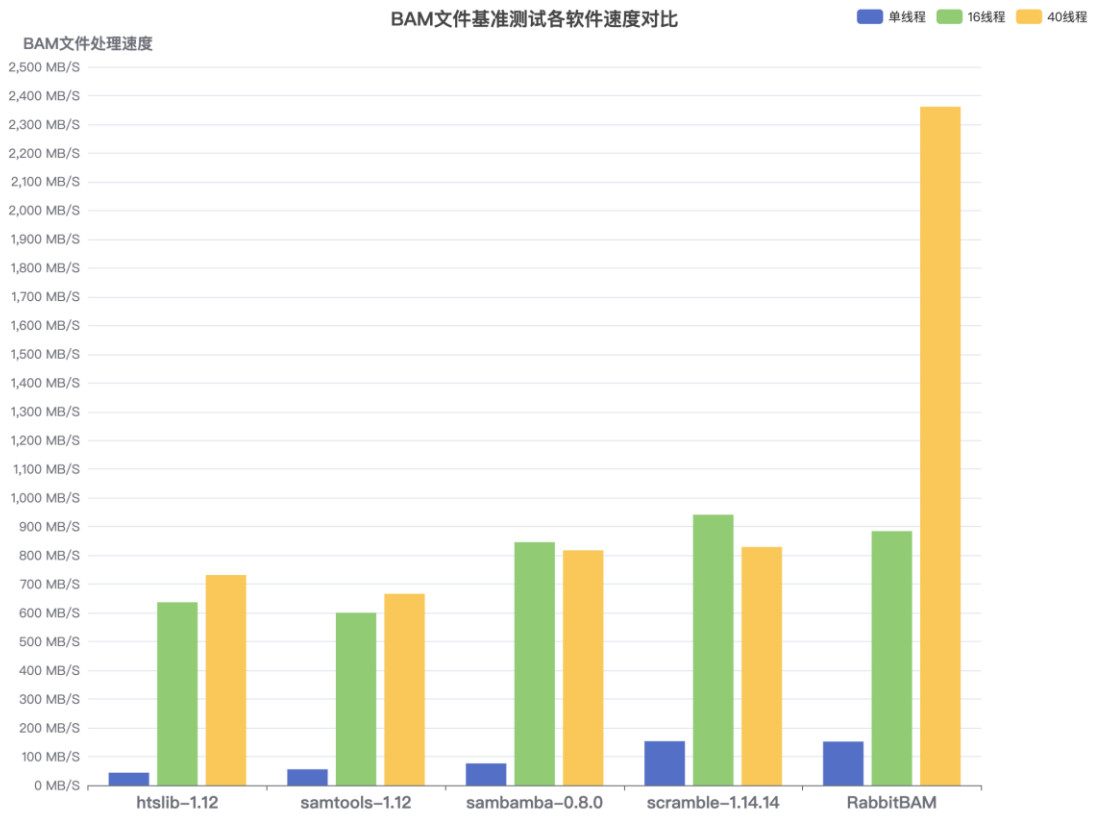
我们定义输出内存池 Buffer, 这次上面消费者变成了生产者，生产待输出的数据，同时考虑到输出的效率相比于解析也很高，我们再安排一个线程用于输出，来作为消费者。实现输出上内存的操作与存储的优化。

完整的算法流程图如下:




性能测试

在性能测试上，使用的测序比对文件为 EA_WGS_T_1.bam,大小约为 9.5 个 G,分别在单线程，16 线程以及 40 线程下运行，最后效果得到了显著的提升，并且加速比也比较合理。




## 2．对测序比对数据文件结果进行质量控制

### 目标简述

参考 FastQC 软件对于 BAM 文件的分析，集成上面编写了以下功能[4]，分别是：基础分析，碱基位置质量分析，碱基序列质量分析，测序序列 GC 比例分析，测序序列 N 比例分析，测序序列长度分析，测序序列重复度分析，并且选择合适的图表展示。

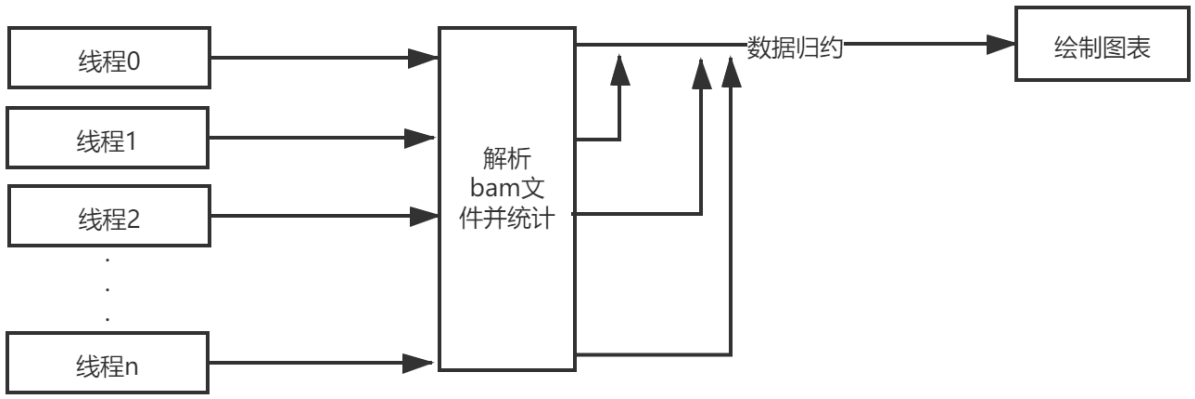
### 完成过程

#### 图表可视化方法

这里我们参考了 FastQC 软件可视化的方法,用 c++字符串填充数据的方式生成 html 文件, 绘制图表则是使用了 echarts 图表, 尽管最后过程繁琐但是不是很难，并且这么做对于软件的集成性会更好。

通过创建一个 BamStatus 类来进行信息的存储与图形的绘制，并在每个线程上创建私有实例，分别单独存储 BAM 文件的相关信息，最后数据再汇总到

一个 BamStatus 实例中，用作最后图形的绘制，算法流程图如下：




质量控制功能展示

基础分析

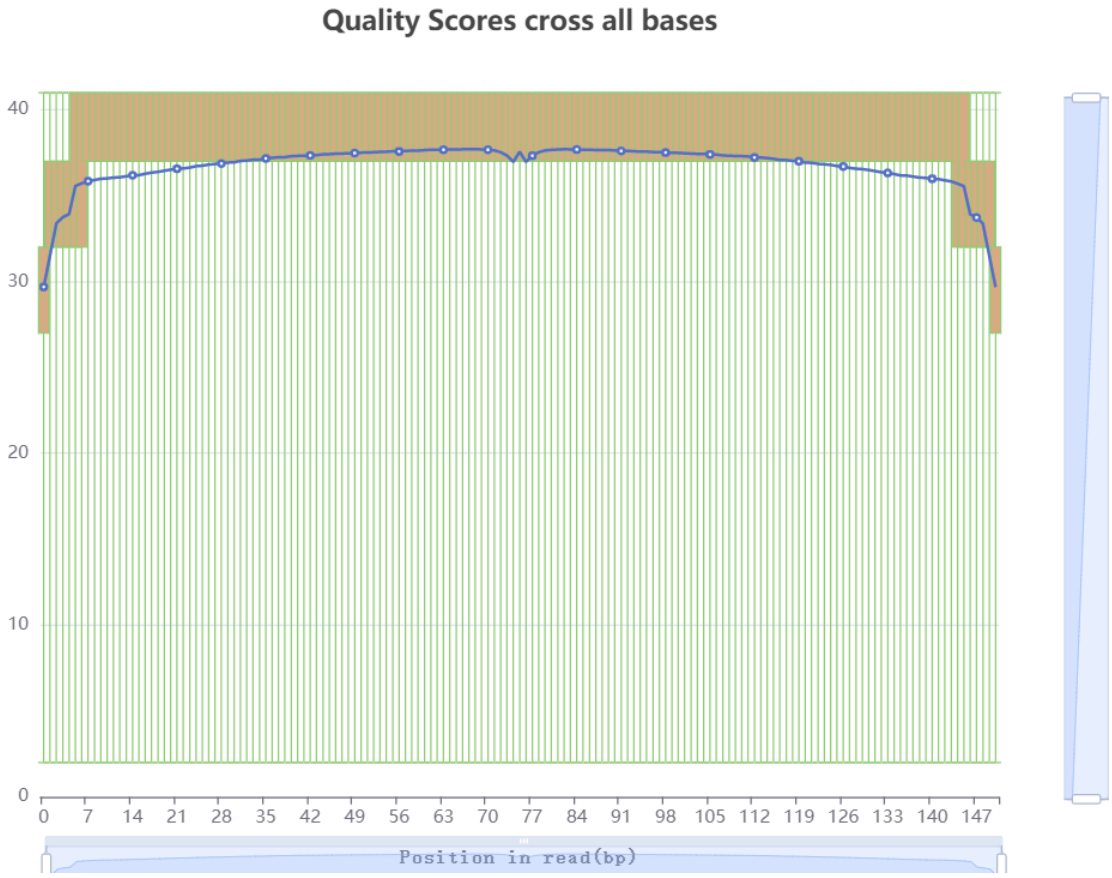
基础分析包括 文件的名称，类型，编码方式以及总 read 数量，read 长度，以及碱基 gc 含量等，比较简单。

| Measure | Value |
| --- | --- |
| FileName | ../data/NC_T_1.sorted.bam |
| File Type | Conventional base calls |
| Encoding | Sanger / Illumina 1.9 |
| Total Sequence | 1212177023 |
| Sequence flagged as poor quality | 0 |
| Sequence Length | 30-151 |
| %GC | 40 |
| Over Represent Date | 0.070000% |

碱基位置质量分析

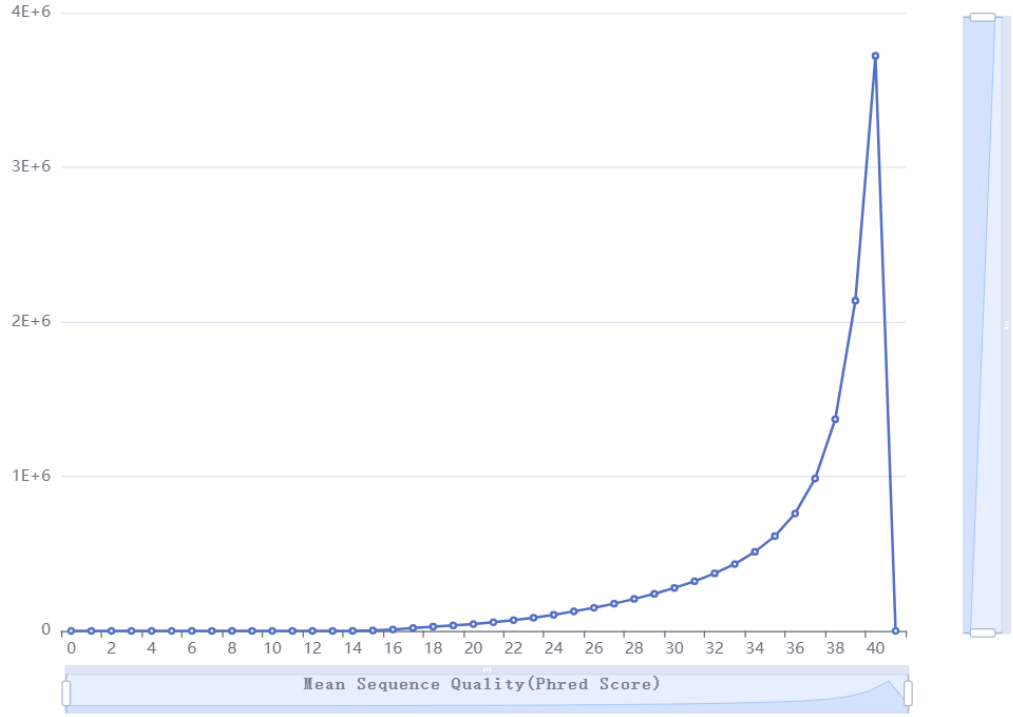
在做 read 质量值分析的时候，FastQC 并不单独查看具体某一条 read 中碱基的质量值，而是将 Fastq 文件中所有的 read 数据都综合起来一起分析。我们参考

该方法绘制了每个位置上所有碱基的质量分布图（箱线图）。横坐标代表每个每个碱基的位置，反映了读长信息，比如测序的读长为 150bp,横坐标就是 1 到 150；纵坐标代表碱基质量值。



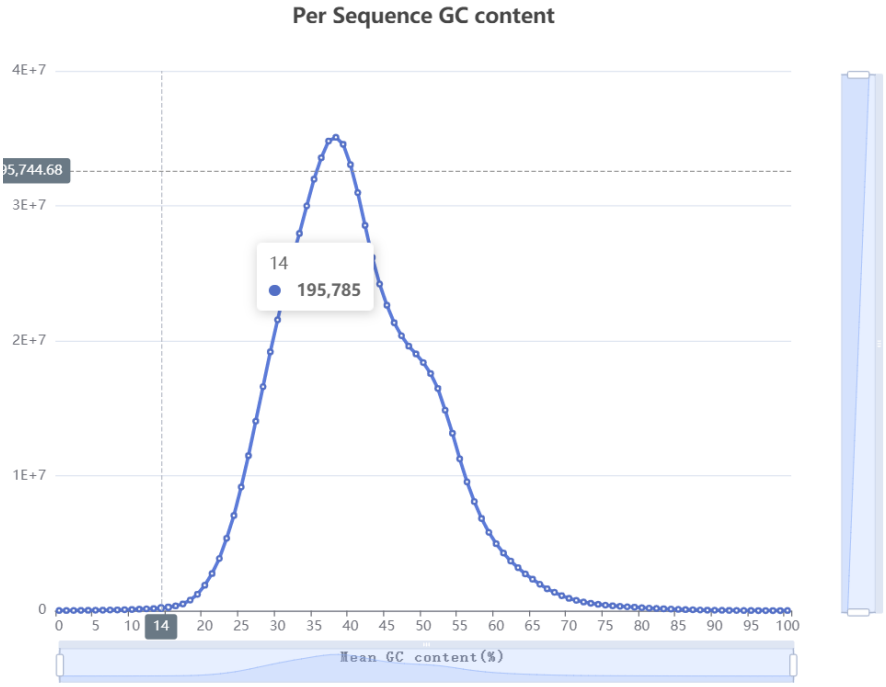

碱基序列质量分析

这个部分是统计所有序列的质量值并进行统计，只要碱基总体质量值分布，只要大部分都高于 20，那么就比较正常。图像如下:



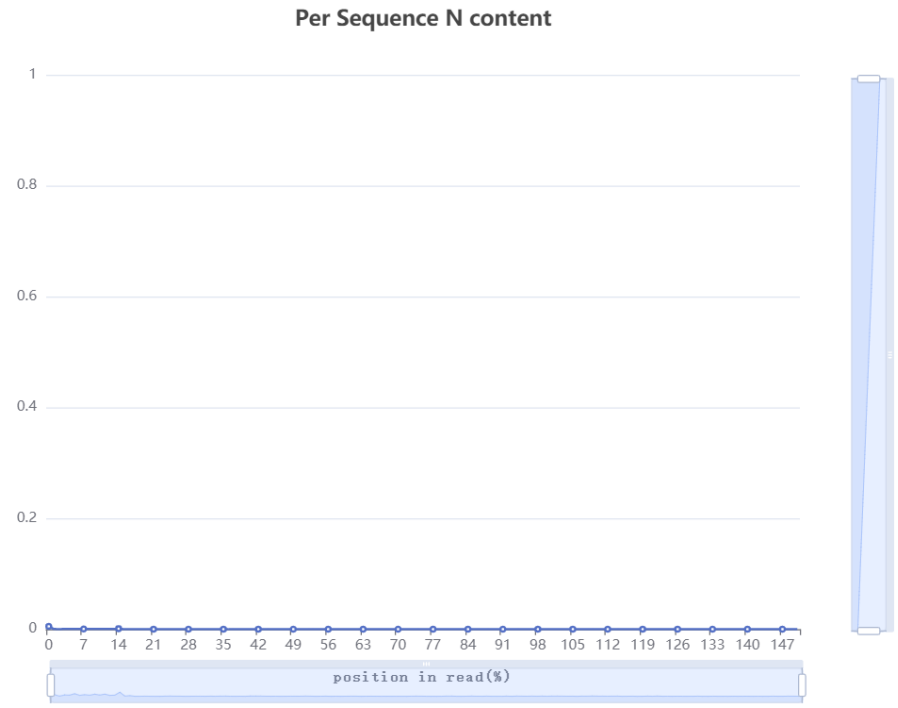

测序序列 GC 比例分析

GC 含量指的是 G 和 C 这两种碱基占总碱基的比例。二代测序平台或多或少都存在一定的测序偏向性，我们可以通过查看这个值来协助判断测序过程是否足够随机。



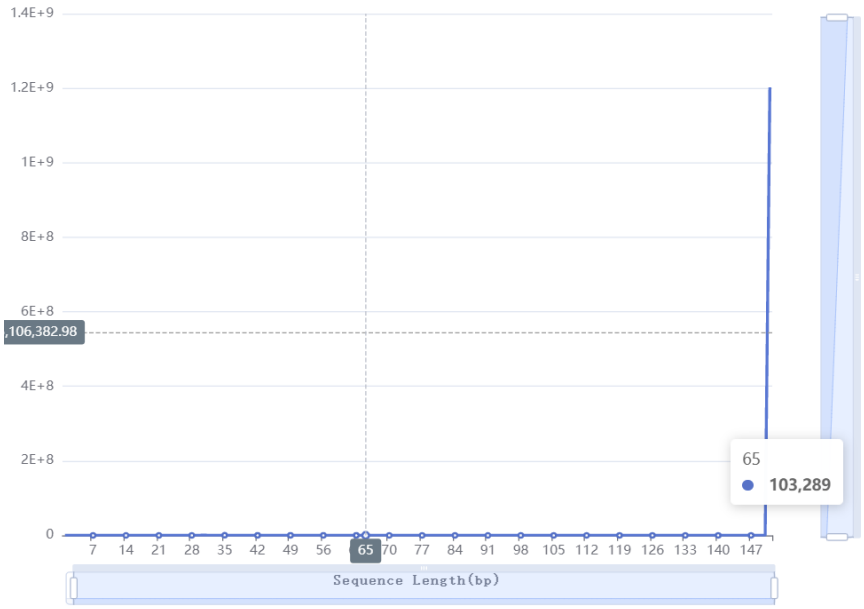
测序序列 N 比例分析

N 在测序数据中一般是不应该出现的，如果出现则意味着，测序的光学信号无法被清晰分辨，如果这种情况多的话，往往意味着测序系统或者测序试剂的错误。



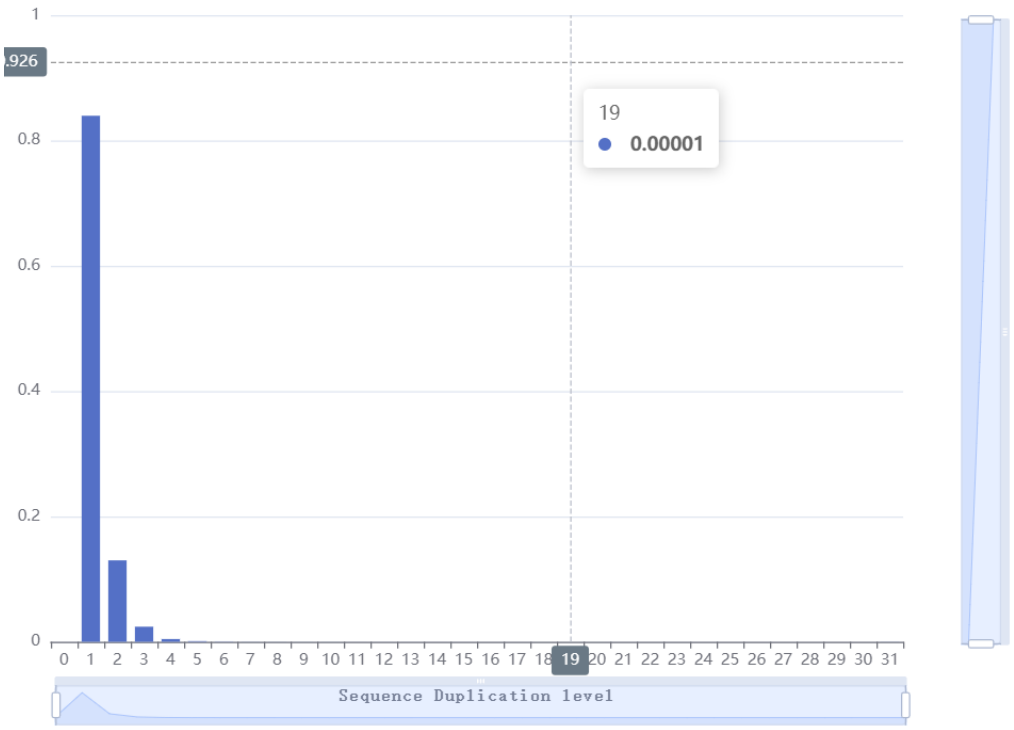

测序序列长度分析

长度分析就是统计序列的长度分布。



测序序列重复度分析

序列重复度分布统计




## 三、项目总结与 Github 地址

参考开题报告的计划，这次项目实训较好的完成了 BAM 文件的信息提取与文件转化与性能优化以及对测序序列比对结果的质量控制，从分析现有软件的不足到参考 BAM 文件解析库自行编写文件解析的步骤，从内存池的性能优化方法到参考生产者消费者模型设计多线程算法与文件 i/o 算法，整个过程下来，看了很多的资料，学到了很多与高性能计算相关的知识，同时该项目非常有意义的一点就是应用层面了，将高性能计算和生物测序技术相结合解决了一个交叉学科的问题，从测序文件的比对结果文件的理解到质量控制的过程与意义，也获得了很多平时无法接触到的知识，但唯一遗憾的是没有能够按照开题计划那样参与到对质量控制的的高性能优化上。

在此特别感谢本次项目实训的指导老师刘卫国教授给予了一个这么有意义的课题，也要感谢高性能计算实验室的赵展学长和闫立峰学长，在项目进行的过程中遇到困难时给予宝贵的指导与建议 。

Github 地址

https://github.com/Antidotec/kyss2021

# 参考文献

[1] https://book.sciencereading.cn/shop/book/Booksimple/show.do?id=B63F62C512748B76AE053020B0A0A1438000

[2] Li H， Handsaker B， Wysoker A , et al. The Sequence Alignment/Map format and SAMtools[J]. Bioinformatics, 2009, 25(16):2078-2079.

[3] https://github.com/samtools/htslib

[4] https://book.sciencereading.cn/shop/book/Booksimple/show.do?id=B63F62C512748B76AE053020B0A0A1438000