Danmarks Tekniske Universitet
DTU Bioinformatics

Read Trimmer for Next-Generation Sequencing Data

First Edition

22110 Python and Unix for Bioinformaticians Fall 22

Jiawen Wu & Oriade Simpson

November 2022

Technical University of Denmark

Project 8
Read Trimmer for Next-Generation Sequencing Data

Abstract

This report is focused around read trimmer software that can be used to process Next Generation Sequencing Data. Next Generation Sequencing machines produce a vast amount of high dimensional data. This data requires processing to ensure that the most accurate parts of this data are kept for downstream analysis. The read trimmer is used to identify and select the most accurate parts of the data. The read trimmer is the python program presented here.

Contribution

Oriade Latifah Simpson and Jiawen Wu are the contributors to this project. The contributions have been made via discussions about the algorithm, paired programming sessions in the library, WhatsApp chat, email, coffee talks , flow chart creation, report writing , Github repository creation, Jupyter notebook shared code and python program creation.

Introduction

Next Generation Sequencing (NGS) aims to generate DNA sequences based on Sequencing By Synthesis Technology. Data generated from Next Generating Sequencing machines are recorded in a text based file.This type of text based file format is called the FASTQ file format.

Millions of single stranded DNA sequences are recorded in a FASTQ file. Therefore this file may be compressed or remain uncompressed depending upon the amount of data recorded.
File compression is used to reduce the storage space on the computer and helps to accelerate computerised analysis of the sequencing data. Each Uncompressed FASTQ filename is suffixed with the abbreviation .fq or .fastq whereas compressed files have the .fastq.gz suffix.

FASTQ parleys to a data type called a double-ended queue or deque where elements can be added to or removed from the front or the back in a ‘fast’ or ‘quick’ manner.
FASTQ files are used as the input for the read trimmer program. Therefore, it is important to access and trim all of the reads inside of each FASTQ file.

The purpose of trimming the reads is to cleave adaptor sequences added during sequencing, and obtain reads that contain sections of nucleotides which have a high probability of being correctly sequenced.

The read trimmer computationally performs this process in the form of a python program. Downstream, these reads can then be aligned and assembled using other programs that use the Burrows-wheeler Transform and FM-index or De Bruijn graph methods, in order to obtain the final, genomic sequence.

Theory

In designing the program, it is important to understand the process of Next Generation Sequencing performed by Illumina Sequencing machines. Sequencing is the precursor to FASTQ file generation, therefore it is also important to understand the meaning of the different parts of the FASTQ file.

ILLUMINA Next Generation Sequencing

The NGS workflow can be divided into steps:

- library preparation
- cluster generation
- sequencing
- data analysis

Library preparation involves the fragmentation of DNA into smaller read fragments. A collection of smaller read fragments is called a DNA library. In paired-end sequencing, synthetic DNA sequences called adaptors are added to the end of the read fragments. The adaptor sequence is complementary to oligonucleotides on the glass plate. This glass plate is called a flow cell. Each adaptor on each DNA fragment binds to a complementary oligonucleotide on the flow cell.

Multiple libraries are sequenced at the same time. During sequencing, the libraries are loaded onto a flow cell and sequenced together. The clusters of DNA fragments are amplified on the glass flow cell in a process called Bridge polymerase chain reaction to create millions of copies of single stranded DNA. Cluster generation amplifies the amount of DNA strands from the library on the flow cell.

DNA polymerase incorporates free florescent-tagged nucleotides (and reversible terminator) to the single stranded DNA template via complementary base pairing. This process is called sequencing by synthesis. Sequencing is a process that obtains the nucleic acid sequences by the detection of characteristic fluorescent signals as each new nucleotide is added to the complementary strand. The sequencing by synthesis technology is employed during NGS. The nucleic acid sequence that comes out of each sequencing cycle is called a ‘Read’.

Laser imaging is used to detect the fluorescent signal that is emitted at the precise time when the free nucleotides are synthesised in the DNA Template strand. The fluorescent signal from only one sequence can hardly be detected. Therefore the fluorescent signal from a cluster of sequences is recorded by the computer.

The overlapping signals that are recorded indicate which nucleotides have been added to the strand. This is not a precise method, therefore there is a probability of error for each nucleotide in each individual read sequence. The error rates are recorded by the machine for miscalling the nucleotides or giving false positive detections of insertions and deletions of nucleotides into the DNA strand.

Fluorescent tags are washed away and the reversible terminator is cleaved, so that the next nucleotide can be added to the single stranded DNA and detected. Afterwards the forward DNA strand is read and the sequence is denatured to yield single stranded DNA again so that the reverse DNA strand can be read.

The sequencing is performed multiple times to get a higher coverage of the DNA.
Millions of reads are sequenced in parallel during next generation sequencing.

FASTQ FILE

The FASTQ filename often contains information about a sample. There are four distinct lines that make up a set within the FASTQ file.

The first element in the set is called the read identification line. The second element of the set contains the DNA sequence. At the start of the third line of the set there is the '+' symbol.
The fourth element of the set is the quality score. The entire set of four elements is called a Read.
For clarity, this second element of the set, which is the single DNA sequence, is also called a Read.

This header line starts with @ symbol which prefixes the name of the sequencing machine.

Each part of the header line is separated by a ':' symbol. The next parts of the header line are :

- The machine identification number.
- The run number for how many times the read is sequenced.
- The flow-cell ID, the lane on the flow-cell where the reads were generated.
- The tile number
- The X coordinate for the read on the flow cell
- The Y coordinate for read on the flow cell
- The Read direction, where _1 represents forward and _2 represents the reverse direction.
- The answer in Yes or No Format as to whether the read has been filtered according to the quality score.
- The control bits and index sequence

The third line is present to act as a separator between the read and the quality line.

The quality score itself has a numeric decimal value e.g. (10, 20, 30, 40, 50) which can be represented algebraically by the letter Q. This numeric decimal value is converted into a single ASCII character so that it can align vertically to the single letters above it in the DNA sequence. Therefore, a single letter in the DNA sequence matches to a single character in the quality score.

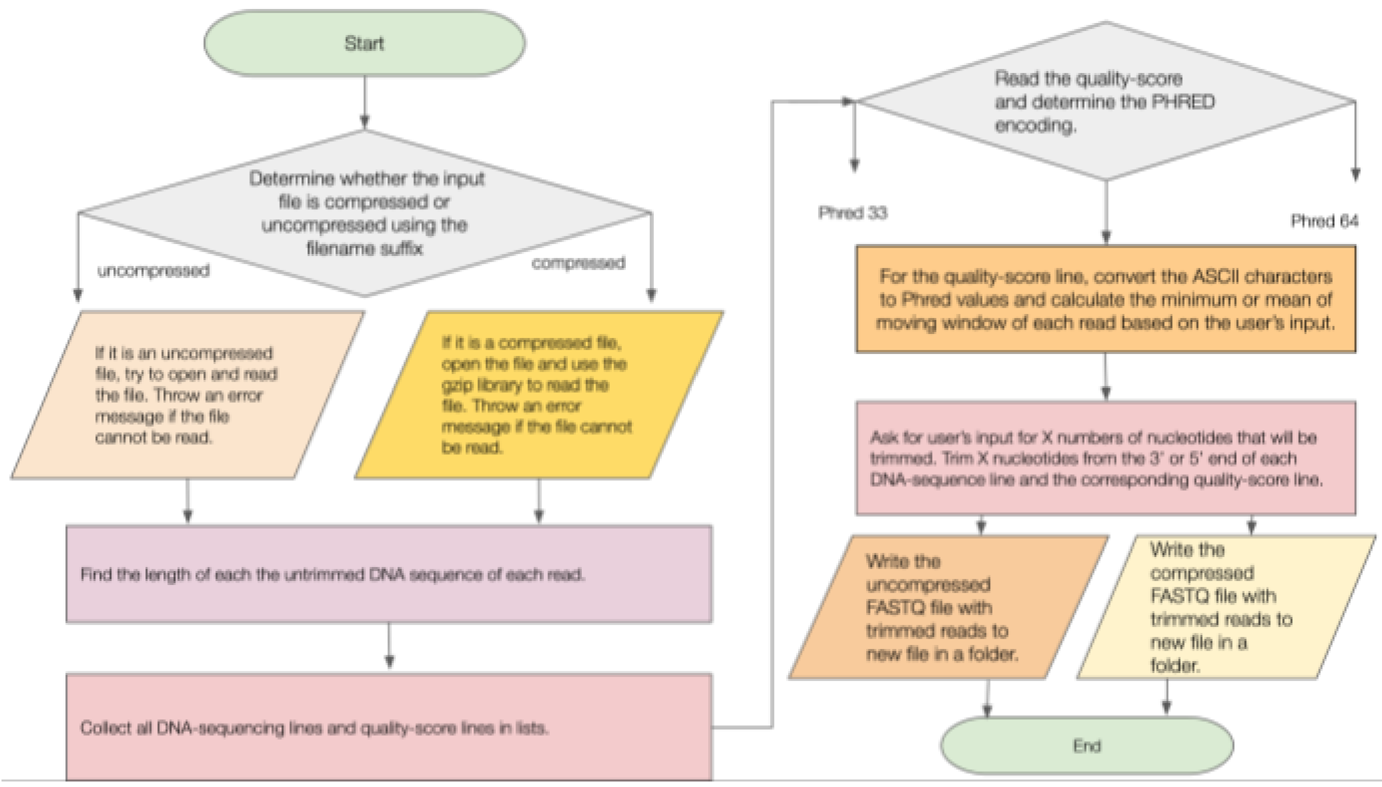
The quality score, or Phred quality score, represents the probability of a nucleotide in the DNA sequence being recorded incorrectly by the machine during Next Generation Sequencing. There are one of two encoding patterns for the quality scores called Phred 64 and Phred 33.

The quality score is converted to the base calling error probability, which is represented algebraically by the letter P. At the root of the project the program will need to handle Phred 64 and Phred 33 high dimensional read data.

Algorithm Design

The central algorithm to describe the idea or plan of how the program should work is displayed in the flowchart below.

The read trimmer is to be designed to trim DNA sequences and the accompanying quality score sequence. The sequences can be trimmed from either end in order to remove remnants of adaptor sequences or to keep sections of reads which contain high quality scores. The design idea is that the header line will have a Y that indicates reads are filtered according to the quality score.



Start
Determine whether the input file is compressed or uncompressed using the filename suffix
uncompressed
compressed
If it is an uncompressed file, try to open and read the file. Throw an error message if the file cannot be read.
If it is a compressed file, open the file and use the gzip library to read the file. Throw an error message if the file cannot be read.
Find the length of each the untrimmed DNA sequence of each read.
Collect all DNA-sequencing lines and quality-score lines in lists.
Read the quality-score and determine the PHRED encoding.
Phred 33
Phred 64
For the quality-score line, convert the ASCII characters to Phred values and calculate the minimum or mean of moving window of each read based on the user's input.
Ask for user's input for X numbers of nucleotides that will be trimmed. Trim X nucleotides from the 3' or 5' end of each DNA-sequence line and the corresponding quality-score line.
Write the uncompressed FASTQ file with trimmed reads to new file in a folder.
Write the compressed FASTQ file with trimmed reads to new file in a folder.
End

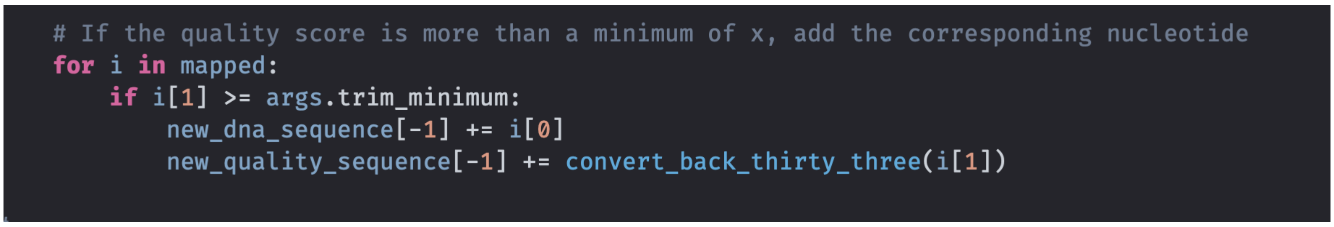
Program Design

The program is designed based on three other pre-existing software programs. The first program layout comes from bcftools version 1.9. The second program layout idea comes from Kallisto version 0.45.0. The third program layout idea comes from BWA alignment via Burrows Wheeler Transformation version 0.7.

The help command is designed to display the instructions for the program.

```
python3 program.py  --help
```

The - - trim_minimum command takes a DNA sequence and its corresponding quality score and converts the quality scores into the corresponding decimal number. An argument is entered into the program specifying the minimum quality score that a base should have. Each nucleotide base is extracted by index depending upon whether or not the decimal number is higher than the minimum value. A snippet of the code for this is shown below.



```
# If the quality score is more than a minimum of x, add the corresponding nucleotide
for i in mapped:
    if i[1] >= args.trim_minimum:
        new_dna_sequence[-1] += i[0]
        new_quality_sequence[-1] += convert_back_thirty_three(i[1])
```

The - - trim_mininimum command is designed to work only with uncompressed, phred33 FASTQ files.

```
python3 program.py --filename fastqfile.fq --trim_minimum  20
```

The reads can be trimmed solely based on the quality score using the trim_minimum function.
In the example above, only nucleotides with a matching quality score of 20 or higher will be included in the output FASTQ file.

The - - shear and - - snip commands work as a pair. In order to trim the reads, the program takes input in the form of a filename from a FASTQ file as well as an indication of how many nucleotides to trim from the 3' end (right side) and 5' end (left side) of each read.

```
python3 program.py --filename fastqfile.fq --shear 5 --snip 5
```

These commands will work for compressed files, uncompressed files, Phred 33 encoded files and Phred 64 encoded files.

Program Manual

Genvej

Read Trimmer for Next Generation Sequencing Data

lifecycle experimental

Overview

`Genvej` is Read Trimmer for Next Generation Sequencing Data program written in Python 3.

`Genvej` provides a consistent set of verbs that help you trim reads from FASTQ files.

These files may be compressed, uncompressed, encoded in with PHRED 64 or PHRED 33 quality scores.

```
python3 project_test_.py
```

Method 1

- `--filename` is used before the name of the file is entered
- `--shear` Trims 3' prime end of a read by (x) amount of nucleotides
- `--snip` Trims 5' prime end of a read by (x) amount of nucleotides

```
python3 project_test_.py --filename PHRED.gz --shear 5 --snip 5
```

Method 2

- `--filename` is used before the name of the file is entered
- `--trim_minimum` Trims reads based on a minimum quality score using PHRED 33 only.

```
python3 project_test_.py --filename BRISCOE.fastq --trim_minimum 20
```

- `--name` Receive a friendly welcome message. This is used before your name is entered.

```
python3 project_test_.py --name
```

- `--version` Display the version of the program

```
python3 project_test_.py --version
```

- `--cite` Display the citation for the paper associated with the program

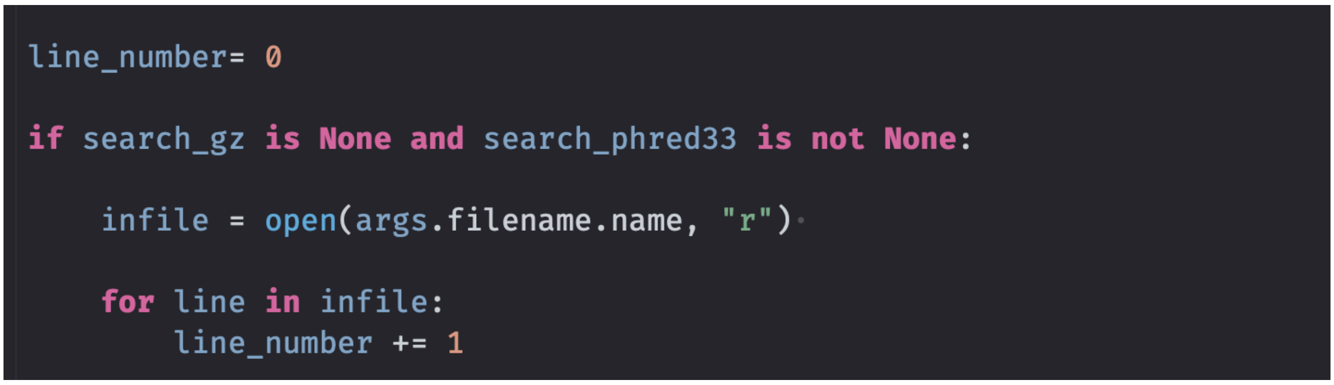
```
python3 project_test_.py --cite
```

- Learn about the functions in the help section of the program `--help` .

Runtime Analysis

The running time of a program or algorithm can be written in Big O notation. The running time relates to the complexity of the algorithm. It can be used to analyse how a function scales with input of increasing size.

The algorithm to estimate the number of lines in the FASTQ file is O(N), where the performance grows linearly with the amount of lines in the input file.



```
line_number= 0

if search_gz is None and search_phred33 is not None:

    infile = open(args.filename.name, "r")

    for line in infile:
        line_number += 1
```

Conclusion

The designed python program is able to trim DNA sequences of interest from both compressed and uncompressed FASTQ files. A custom number of nucleotides input by the user from the command line dictates how many reads will be trimmed from each end. The program creates new output files containing all of the trimmed reads in a folder called Genvej.

The - - trim_minimum function only works for Phred33 , uncompressed files. However this part of the program could be expanded to work with compressed files and Phred 64 encoded files.

The logging module was used at the end of making the program, however it could have been used from the beginning instead of using print statements throughout the program.

A deeper understanding of Big O notation could be fruitful in designing a better program.

In future, the algorithm could be adapted to have the - - trim_minimum function work for all types of compression and encoding.

Bibliography

- ☐ NGS Workflow Steps | Illumina Sequencing Workflow . https://emea.illumina.com/science/technology/next-generation-sequencing/beginners/ngs-workflow.html Accessed 10 Nov 2022

- ☐ Chauhan, Dr. Tushar “What is Sequencing Read in NGS?”, Genetic Education 24 August 2022, https://geneticeducation.co.in/what-is-sequencing-read-in-ngs/.

- ☐ Eisenhaber Frank, “Introduction to Bioinformatics. By Arthur M. Lesk” Biotechnology Journal , vol 3, no. 11, Nov. 2008, pp. 1452-53, DOI.org, https://doi.org/10.1002/biot.200800277

- ☐ Sætre, Glenn Peter & Mark Ravinet, Evolutionary Genetics, Concepts, Analysis and Practice. First Edition, Oxford University Press , 2019

- ☐ ZoteroBib: Fast , Free Bibliography Generator, MLA, APA, Chicago, Harvard Citations, https://zbib.org/ . Accessed 11 November 2022

- ☐ Bcftools by Samtools. http://samtools.github.io.bcftools Accessed 11 November 2022

- ☐ Bray NL, Pimentel H, Melsted P, Pachter L "Near-optimal probabilistic RNA-seq quantification with Kallisto." Nat Biotechnol. 2016 May;34(5):525-7. https://doi.org/10.1038/nbt.3519

- ☐ Li H. and Durbin R. (2010) BWA Fast and accurate long-read alignment with Burrows-Wheeler Transform. Bioinformatics, Epub. [PMID: 20080505]

- ☐ Argparse Help Messages, (2022) https://stackoverflow.com/questions/35847084/customize-argparse-help-message

- ☐ *A Beginner’s Guide to Big O Notation - Rob Bell*. https://robbell.io/2009/06/a-beginners-guide-to-big-o-notation. Accessed 28 Nov. 2022.

- ☐ “Big O Notation | Interview Cake.” *Interview Cake: Programming Interview Questions and Tips*, https://www.interviewcake.com/article/java/big-o-notation-time-and-space-complexity. Accessed 28 Nov. 2022.