SEQC - A fastq quality reporting app

Antje Oldenburg

Department of Computer Science, Free University of Berlin, Takustr. 9, 14195 Berlin, Germany
E-mail: *antje.oldenburg@fu-berlin.de*

ABSTRACT

This paper reports on the application created during the course "Software Project Management" for the project "Mini-FASTQC: NGS Quality Control". Using seqan C++ library, an application was created to report statistics on NGS data. We show which measures the application can report on and how it compares to existing NGS quality control applications.

1 INTRODUCTION

One of the first steps in analyzing any next generation sequencing (NGS) datasets is to take a quality measure of the data. Each technology has their own quality issues. To provide confidence in the reliability of results of any following step in the analytic pipeline, output quality reporting must be able to inform about quality criteria that where met and more importantly, failed to meet.

This paper introduces the seqc, a NGS quality reporting application that was implemented using the seqan library. It was modelled after existing tools, most notably fastqc (Andrews, 2010) and the fastx toolkit (Gordon and Hannon, 2010).

In a next step, data manipulation may follow to prevent undesirable outcomes in assembly or mapping. Using insight gained from the report, low-quality reads may be discarded or trimmed, controlled for contamination and corrected for adapter sequences. Ideally, all preliminary quality control procedures should be applied to raw reads before further analysis and – if possible – in a single run.

2 METHODS

Established basic quality measures for the analysis of fastq files are base call quality and nucleotide distribution. In addition to those, checks for overrepresented and duplicated sequences, k-mer distribution.

While some users may want to perform the base calling step themselves, we we will assume that the machine manufacturers proprietary software is used to extract a FASTQ file that will be our input. After the application has been run on the dataset, tab separated text files will be created that contain the information gathered. Those can then be used in a second step to generate graphical representations.

2.1 FASTQ quality encoding

The application can guess the encoding scheme of phred scores based on the range of occuring encoded scores. If the user is certain about the offset used to encode the quality score, an encoding can be used instead of the guess.

2.2 Quality distribution statistics

2.2.1 Per Position To reduce the distribution description to a small set of most useful indicators and promote the creation of graphical representations, the following data is output per position in read: Mean, Median, Lower and Upper Quartile, 10th and 90th Percentile. For that purpose, a 2-dimensional matrix [position in read] $\times$ [quality score] is filled with the number of encountered scores and can thus be used to recreate complete distribution data.

2.2.2 Per Read Per read quality thresholds are set to be < 27 (equates to a 0.2% error rate) to warn and < 20 (equates to a 1% error rate) to fail.

2.3 Nucleotide distribution statistics

Distribution of called bases are presented as a percentage of each nucleotide per read, as well as per position in read. They are collected as counts in a 2-dimensional matrix [position in read] $\times$ [base] so that individual (ACGTN) as well as aggregate (CG content) ... can be displayed.

2.3.1 Distribution of bases called Given a random sequencing job, one can assume that the distribution of bases called between positions in read should be more or less consistent. The portion of the base per read should reflect an overall portion of that base in the sequenced library. Therefor, a diversion from an established pattern in a particular position is likely a technical problem rather than a particularity of the sequenced data.

A warning will be issued if the difference between the rations of `A/T`, or `G/C` is greater than 10% in any position. If this difference is greater than 20%, the result will indicate a fail.

2.3.2 Distribution GC content A deviation from the average GC content bias in one position is most likely a symptom of a systematic error. A common cause can be an overrepresented sequence. This file indicates a warn state if GC content of any base differs more than 5% from the mean GC content. and a error for any that differ by more than 10%.

2.3.3 Distribution N content In a case that the case calling program cannot with any certainty decide for a specific nucleotide, an N will be called. A certain noise (very low portion) can be expected, especially near the end of a sequence. If this portion rises above 5% in any position, a warning will be indicated. This test fails when those are more than 20%.

2.4 Read length distribution

The number of encountered read lengths will be given for each read length.

2.5 Sequence duplication statistics

A low level of duplication spread through the dataset may indicate a high level of coverage of the target sequence, but a high level of duplication in a few sequences is more likely to be a symptom of enrichment bias.

The implementation uses a Suffix Array (Seqan Index-ESA) to optimize search for reoccuring sequences. In solving the issue of memory demand for this feature, we followed the example of fastqc and limited the size of the haystack to the first 200.000 reads. Each sequence in the haystack is then tracked to the end of the file. While this method may not show the exact duplication levels in a sequencing run, it can give a representative count of the overall duplication level.

2.6 Application input

Sequence streaming was used as provided by the seqan library. It provided us with the ability to read FASTQ input raw or compressed.

2.7 Text file output

To create textfiles in a consistent manner throughout different application modules, a TsvWriter class was created. It implements text formatting and streaming of different views, whereas a ReadStats object provides the statistical data itself.

3 RESULTS AND DISCUSSION

3.1 Input and Output

The application was successfully tested (without usage of the kmer statistics) on datasets from Sanger, Illumina HighSeq 2000, Solexa, Ion Torrent PGM and 454 GS FLX. On all datasets, fastqc output differs from our output. Those differences where investigated and found to be (slight) deviations from the exact values, which are provided by this application.

The application's cpu time requirements are vastly exceeding those of fastqc. A run of a file of 7Gb (ENA:SRR066417_1) used 13.2 min CPU time, this applications was cancelled after 70min. Memory requirements seemed to be stable at about 420MB per run whereas fastqc used merely 190 MB.

Further investigation will likely show that the library's efficiency was not fully taken advantage of.

3.2 Performance

4 CONCLUSION

The application presented implements the basic quality control reporting functionality needed in a NGS workflow. The software was implemented in C++ using seqan library and can therefore benefit from efficient algorithms and data structures optimized for sequence analysis. Even though,the application will not be applicable in a production environment before strong improvement in effective usage of data structures.

REFERENCES

Andrews, S. (2010). Fastqc. *A quality control tool for high throughput sequence data [http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc/]*.

Gordon, A. and Hannon, G. (2010). Fastx-toolkit. *FASTQ/A short-reads pre-processing tools (unpublished) http://hannonlab. cshl. edu/fastx_toolkit*.