Analytical Biochemistry 548 (2018) 38–43

Contents lists available at ScienceDirect

Analytical Biochemistry

journal homepage: www.elsevier.com/locate/yabio

Quack: A quality assurance tool for high throughput sequence data

Adam Thrash, Mark Arick II, Daniel G. Peterson*

Institute for Genomics, Biocomputing & Biotechnology, Mississippi State University, Mississippi State, MS 39762, USA

ARTICLE INFO

Keywords:
Quality assurance
Visualization
FASTQ
Quality check
DNA

ABSTRACT

The quality of data generated by high-throughput DNA sequencing tools must be rapidly assessed in order to determine how useful the data may be in making biological discoveries; higher quality data leads to more confident results and conclusions. Due to the ever-increasing size of data sets and the importance of rapid quality assessment, tools that analyze sequencing data should quickly produce easily interpretable graphics. *Quack* addresses these issues by generating information-dense visualizations from FASTQ files at a speed far surpassing other publicly available quality assurance tools in a manner independent of sequencing technology.

Introduction

Twenty-first century sequencing techniques have increased biomolecular data production at a rate outpacing Moore's Law [9]. However, with the staggering proliferation of sequence data comes many issues including the need for faster, more intuitive quality control assessment prior to downstream data analysis [4]. Also, with the introduction of Pacific Biosciences and Oxford Nanopore Technologies sequencing platforms, the quality and length of sequence reads have become more varied, making platform specific tools designed for short read sequences [e.g. [3]] of limited utility.

Several principles underlie good quality assessment tools [2,13]. First, sequence quality data needs to be presented in a graphical format to allow rapid assessment. While numerical summary statistics can be extremely useful, simple graphs can expose fundamental problems that would not be detected using summary statistics alone (Fig. 1). Second, the visualizations need to be data-dense, meaning a large amount of information needs to be displayed in a relatively small area [13]. By displaying all of the information within the common eyespan, the data can be assessed more quickly and ergonomically. Third, the tool should be sequencing technology independent. Finally, the tool should be fast and should scale to increasingly larger data sets.

There are several available tools designed to assess the quality of sequence data. Ref [1] notes, "*The undisputed champion of quality control visualization is a tool named FastQC developed by Babraham Institute.*" However, Ref [1] adds, "*Even though it is a de-facto standard of visualization, its results are not always the simplest to interpret.*" While FastQC does follow most of the design principles outlined above, it is still relatively labor intensive, especially when examining the quality results for large datasets [1,5]. FastQC is written in Java [3].[1]

Fastqp [11] is another tool for quality assessment of FASTQ files. Though not as widely adopted as FastQC, Fastqp generates high quality graphics. Fastqp is written in Python.

To improve the speed and ease of quality assessment of FASTQ files, we developed the *Quack* algorithm. Quack is written in C which accounts for much of its speed. We demonstrate the value of Quack by comparing its speed and output to FastQC and Fastqp. Our findings indicate that Quack readily outperforms the other tools.

Methods

Implementation

We chose to write Quack in the programming language C for several reasons. First of all, C has been in use for > 40 years, yet, if anything, its relevance is increasing; e.g., TIOBE (www.tiobe.com), a company that accesses the quality and usage of software, named C its 2017 Programming Language of the Year [12]. Operating systems that are based, in full or part, on C include UNIX, Windows, Linux, GNU, and Macintosh, while mobile devices that use C kernels include iOS, Android, and Windows Phone. The most popular databases – Oracle, MySQL, MS SQL Server, and PostgreSQL – are primarily written in C, most embedded systems (e.g., programs that run appliances, vehicles, etc.) are written in C, and the majority of powerful supercomputers run the Linux (C-based) kernel. C has a relatively small, standardized vocabulary and a short runtime which have made it the language of choice

* Corresponding author.
E-mail address: dp127@msstate.edu (D.G. Peterson).

[1] Recently, a tool called FQC was developed to parse FastQC's HTML output into a more interactive format [5]. However, while FQC increases the utility of FastQC, it is ultimately dependent upon (and ostensibly limited by) the FastQC algorithm, and thus it is not considered separately from FastQC.

https://doi.org/10.1016/j.ab.2018.01.028
Received 3 October 2017; Received in revised form 25 January 2018; Accepted 30 January 2018
Available online 01 February 2018

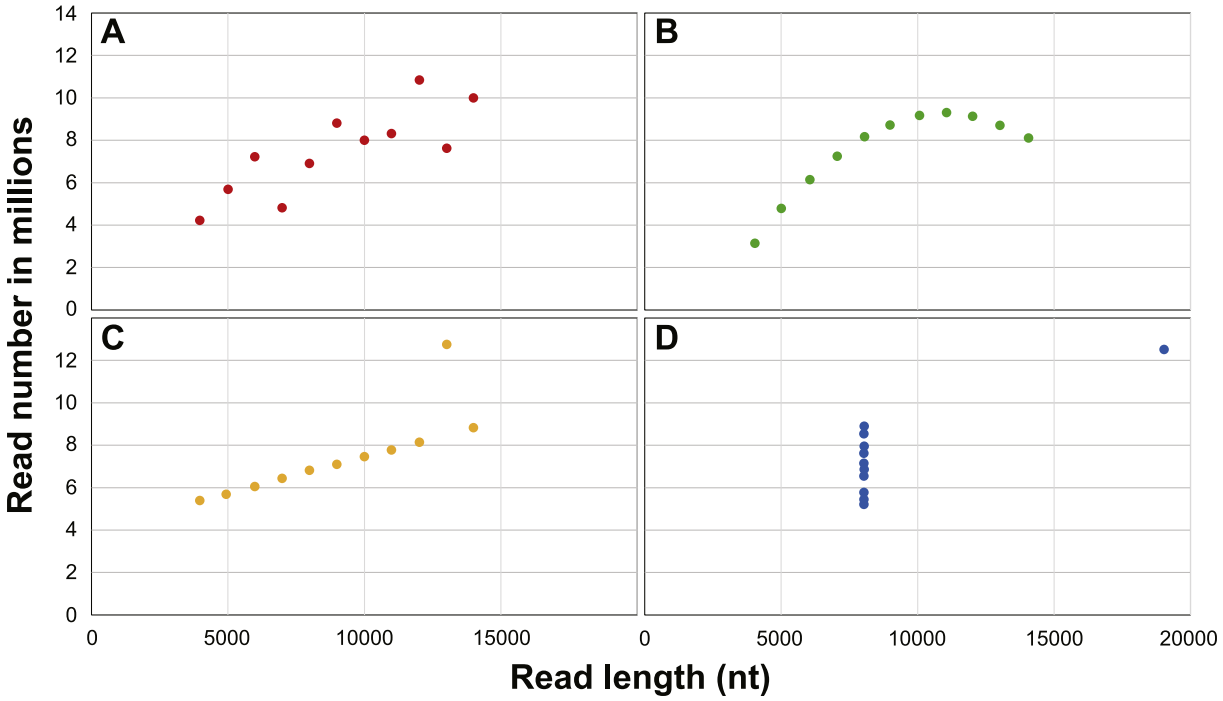

Fig. 1. Value of graphing data as illustrated using Anscombe's quartet [2]. The quartet consists of four data sets that have nearly identical descriptive statistics (e.g., mean of *x*, mean of *y*, sample variance of *x*, sample variance of *y*) yet bear no resemblance to each other when graphed. Anscombe developed the quartet – shown in A-D in a slightly modified format to resemble DNA sequence read data – to demonstrate the how reliance on numerical statistical values alone can be perilous.

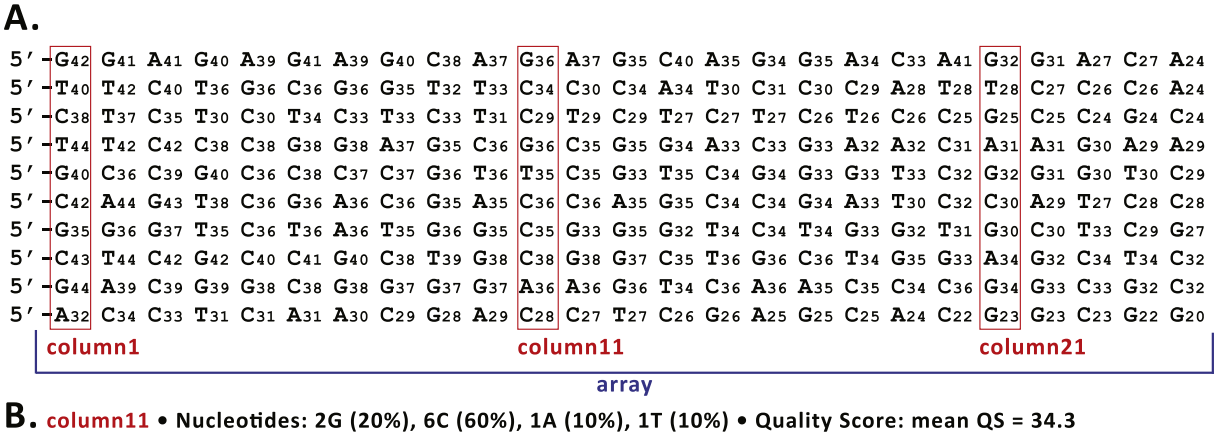

Fig. 2. Nomenclature used to discuss FASTQ data. (A) In this simplified example, there are 10 sequence reads (rows) each 25 nt in length. The nucleotide and quality score information at each position along the stacked reads is called a *column*, while the set of contiguous columns representing the rows of stacked sequence reads is called an *array*. In a real FASTQ file, there may be thousands to millions of stacked reads in an array. Additionally, each nucleotide has a quality score represented by a single ASCII symbol versus a two digit number. We show the quality scores as numerical values to facilitate understanding. (B) Summary data from column 11 including number and percentage of each nucleotide, and mean quality score for the column.

for those seeking long-term reliability/stability and high speed [10]. Moreover, as noted by Ref. [8]; "There are so many C compilers, you can write stuff in C and have it run pretty much anywhere."

Quack's operation can be described by three major steps; reading the adapters file, reading the FASTQ file, and drawing the data.

(1) *Reading the adapters file.* Quack reads the adapters file by parsing each adapter into *k*-mers of length 10. Every possible 10-mer of contiguous bases is generated for each adapter (e.g., the first 10-mer for an adapter consists of nucleotides 1–9, the second 10-mer consists of nucleotides 2–10, etc.). A hashing function is used to create a database from all of the 10-mers, and this database is used to check adapter content in the next step.

(2) *Reading the FASTQ file.* FASTQ files contain thousands to millions of single or paired-end reads. Quack stacks all the reads in a FASTQ file in a sequence start to sequence finish orientation (for paired end sequences, the reads are separated into forward and reverse read stacks). As shown in Fig. 2, each read serves as a row while each position along a set of stacked reads is referred to as a *column*. The columns are in a fixed linear order (e.g., column1, column2, column3 …) with each column containing the numerous stacked nucleotides (one nucleotide from each stacked read) and their respective quality scores. A set of contiguous columns is deemed an *array*. While the FASTQ file is read, Quack uses a two-dimensional array to determine the relative frequency of each of the four nucleotides in a column and records the quality score information for the column (Fig. 2B). Using a 10-mer sliding window, each read is checked against the adapter database one 10-mer at a time. If a read is found to contain an adapter 10-mer, the start location of the match along the read is stored, and further adapter searching along