SOFTWARE TOOL ARTICLE

REVISED Falco: high-speed FastQC emulation for quality control of sequencing data [version 2; peer review: 2 approved]

Guilherme de Sena Brandine, Andrew D. Smith

Department of Quantitative and Computational Biology, University of Southern California, Los Angeles, California, 90089, USA

v2 **First published:** 07 Nov 2019, **8**:1874
https://doi.org/10.12688/f1000research.21142.1
Latest published: 27 Jan 2021, **8**:1874
https://doi.org/10.12688/f1000research.21142.2

Abstract

Quality control is an essential first step in sequencing data analysis, and software tools for quality control are deeply entrenched in standard pipelines at most sequencing centers. Although the associated computations are straightforward, in many settings the total computing effort required for quality control is appreciable and warrants optimization. We present Falco, an emulation of the popular FastQC tool that runs on average three times faster while generating equivalent results. Compared to FastQC, Falco also requires less memory to run and provides more flexible visualization of HTML reports.

Keywords

FastQC, high-throughput sequencing, quality control

Open Peer Review

Reviewer Status ✓ ✓

	Invited Reviewers	
	1	**2**
version 2 (revision) 27 Jan 2021	✓ report	
version 1 07 Nov 2019	? report	✓ report

1. **R. Henrik Nilsson**, University of Gothenburg, Gothenburg, Sweden

2. **Weihong Qi**, Functional Genomics Center Zurich, Zürich, Switzerland

Any reports and responses or comments on the article can be found at the end of the article.

Corresponding author: Andrew D. Smith (andrewds@usc.edu)

Author roles: de Sena Brandine G: Formal Analysis, Investigation, Methodology, Project Administration, Software, Validation, Visualization, Writing – Original Draft Preparation; **Smith AD**: Formal Analysis, Investigation, Methodology, Project Administration, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: The author(s) declared that no grants were involved in supporting this work.

How to cite this article: de Sena Brandine G and Smith AD. **Falco: high-speed FastQC emulation for quality control of sequencing data [version 2; peer review: 2 approved]** F1000Research 2021, **8**:1874 https://doi.org/10.12688/f1000research.21142.2

First published: 07 Nov 2019, **8**:1874 https://doi.org/10.12688/f1000research.21142.1

REVISED **Amendments from Version 1**

This article has been updated to address reviewer responses. Changes to the text were made in all sections. Table 3 and Table 4 were expanded to include time measurements for FastQC on long-read samples. No other table or figured was altered from the first version of the manuscript. The accompanying code for Falco has also undergone updates for this review. Falco version 0.2.4 was used ion this revised manuscript. The code changes that relate to the core computations were not altered since version 0.1.0 (used in the previous version of the manuscript), and we have verified that the times reported in Table 3 remain the same in both versions.

The main changes to the manuscript are listed below:

(1) The abstract was changed to highlight the memory comparison between QC software tools, and no longer mentions that FastQC does not run on long-read samples.

(2) The "Introduction" section includes more detail about quality control applications.

(3) The "Implementation choices" subsection under "Methods" now highlights that Falco does not contain a user interface, and that Falco was designed for UNIX systems.

(4) The "Methods" section now contains a "system requirements" subsection that describes the memory and disk requirements to run Falco.

(5) The subsection "Falco scales for larger nanopore reads" has been removed, and instead replaced with an additional paragraph on section "Falco is faster than popular QC tools", where the memory usage of each tool in each tested sample is discussed

(6) Instructions to report bugs and errors is reported in the "Software availability" section

(7) Formatting corrections were performed across the manuscript: "Falco" is now written in uppercase, superfulous line breaks were removed, reference formatting and the usage of the Oxford comma were standardized, links were separated from punctiation and two references were added.

Any further responses from the reviewers can be found at the end of the article

Introduction

High-throughput sequencing is routinely used to profile copy number variations in cancers[1], assemble genomes of microbial organisms[2,3], quantify gene expression[4], identify cell populations from single-cell transcriptomes in a variety of tissues[5], and track epigenetic changes in developing organisms and diseases[6], among numerous other applications. New sequencing protocols are constantly being introduced[7,8], and as the cost of sequencing per base decreases, sequencing data is growing in abundance, dataset size, and read length[9].

Quality control (QC) is often the first step in high-throughput sequencing data analysis pipelines. The QC step measures a set of statistics in a file of sequenced reads to assess if its content matches the experiment expectations and if the data is suitable for downstream analysis. Common QC tests include counting relative frequency of nucleotides in each position of a set of reads to detect potential deviations from expected frequencies, summarizing the distribution of Phred[10] quality scores to identify base positions with globally low quality (suggesting degeneration in the sequencing process), and measuring the frequency of sequencing adapters and contaminants that are not expected to be biological DNA from the sample.

Data that passes specific QC tests then undergoes downstream analysis steps, which may include adapter trimming, filtering contaminants and low-quality reads, and mapping the resulting reads to a reference genome or transcriptome. With the exception of sequence assembly applications, read mapping should be the most computationally expensive step early in analysis pipelines. In comparison, the time and computation required for QC should be negligible. However, the efficiency of mapping algorithms has improved substantially over the past decade, while software for QC has received far less attention. As a consequence, the computation required for QC is appreciable, and can no longer be ignored when considering the total cost of sequencing.

The most commonly used tool for quality control of sequencing data is `FastQC`[11], which, since its release, has incorporated a wide range of QC tests covering multiple use cases. Its analysis reports have become the standard for several QC tools, and automated analysis pipelines often rely on its result as a criterion to proceed with downstream steps or, alternatively, to filter, trim, or ultimately discard the data[12,13]. `FastQC` reports ten analysis modules that summarize the content of a sequencing file (Table 1). An input file may pass or fail the tests run in each module, and high-quality sequencing data from most protocols is expected to pass all tests.

In `FastQC`'s implementation, each module computation is executed sequentially after an input sequence is read. This design allows new modules to be incorporated easily, but it implies that the time required to process each read is the sum of the processing times for each module. If multiple modules

Table 1. Comparison of analysis modules provided by `fastp` and `HTQC`, two commonly used QC software tools.

`FastQC` module	`fastp`	`HTQC`
Per base sequence quality	No	Yes
Per base N content	Yes	Yes
Per tile sequence quality	No	Yes
Per sequence quality scores	No	Yes
Per sequence GC content	No	No
Sequence length distribution	No	Yes
Sequence duplication levels	Yes	No
Overrepresented sequences	Yes	No
Adapter content	No	No
Kmer content	Yes	No

compute similar measurements, such as nucleotide content or Phred quality scores, the same calculation will be performed multiple times, causing the total analysis run time to increase.

Several QC software tools have been introduced since FastQC, many focusing on speed improvements, more flexible module visualization, incorporation of paired-end reads, and filtering sequences that failed QC tests. Despite proposing different alternatives to calculate and present QC results, the modules available in these tools are largely similar to FastQC's (Table 1).

At the same time, FastQC's analysis results are already part of many standard initial analysis pipelines. If a new QC software tool is incorporated in these pipelines, it is desirable that its results, and its output formats, remain consistent with those generated by FastQC.

To address potential speed limitations in FastQC's implementation while retaining its behavior, we developed *FastQC Alternative Code* (Falco)[14], an emulation of the FastQC software tool. We show that Falco generates the same results as FastQC across a wide variety of datasets of different read lengths, sizes, file formats, and library preparation protocols at significantly shorter running times and using less memory. While the text outputs are comparable to FastQC, Falco also provides more flexible interaction with graphical plots in its HTML report using the same visualization standards set by FastQC.

Methods

Implementation choices

Falco[14] is an Open Source C++ implementation of the FastQC software tool built for UNIX-based operating systems. We designed it to faithfully emulate FastQC's calculations, results and text reports. The goal of Falco is to minimize the effort required to replace the command-line behavior of FastQC in the context of larger automated analysis pipelines. We use the same set of command-line arguments, configuration file names, and input file formats as FastQC. We also produce the same plain text format output, and the same report structure, allowing users to take advantage of improved speed without adjusting to different program behaviors. Falco is intended to be used in a command-line environment. Unlike FastQC, Falco cannot be run through a graphical user interface.

There are major differences between the implementations of Falco and FastQC. While FastQC's code emphasizes modularity, which allows new QC metrics to be added easily and uniformly, Falco's design centralizes the function to read sequences from the input file and collects the minimum data necessary to subsequently create all modules after file processing. To ensure consistency with FastQC, we wrote each module's source code based on FastQC's implementation, adapting the portions that relate to sequence processing and maintaining the postprocessing functions that define how the collected data is used to generate summaries and reports.

Operation

Compilation of Falco requires a GNU GCC compiler version 5.0.0 (July 16, 2015; full support for the C++11 standard) or greater. Once compiled, Falco can be run on uncompressed files (FASTQ and SAM) without any additional dependencies. In order to process files in gzip compressed FASTQ and BAM formats, Falco must be compiled with the ZLib[15] and HTSLib[16] libraries, respectively. The full documentation on how to compile, install dependencies, and run the program is available in the README file in the Falco repository.

Use cases

Like FastQC, Falco[14] can be applied to any file of sequenced reads in the formats accepted by FastQC. The only required command-line argument is the path to the input file. Also like FastQC, a wide range of options can be provided if users only require a given subset of its analysis modules or outputs. The letters and symbols used for command-line arguments were chosen to maintain consistency with FastQC's options. As mentioned above, this choice is to facilitate integration with larger pipelines that already employ FastQC and depend on its behaviors.

Falco can be run on a FASTQ format file named example.fq with the following simple command:

```
$ falco example.fq
```

This will generate three files:

1. fastqc_data.txt: The complete numerical values generated in each module's individual analysis.
2. fastqc_report.html: A visual page display of the text report's data and plots generated in modules.
3. summary.txt: A short summary indicating whether the input file passed or failed each module, and whether any warnings were raised.

Default configuration files are contained in a Configuration directory that is included with the program, but Falco also allows users to manually define the thresholds to pass or fail each module, the list of adapters to search for in reads, and the list of contaminants to compare with overrepresented sequences by using configuration files in the same format used by FastQC.

System requirements

Falco requires little memory and disk space to run, and there are no constraints on the minimum or maximum FASTQ input size or number of reads. Reads are analyzed sequentially, with one read stored in memory at a time, so the amount of memory necessary to run depends on the largest read length in a dataset, but not on the size of the input file. For instance, processing a short-read sample, with reads of length at most 1000 bases, requires 100 MB of available RAM, whereas processing a long-read sample containing at least one read with 1 million bases require 500 MB of RAM. The total disk space necessary

to store the three output files generated by Falco is no more than 1 MB.

Results

Falco matches FastQC's output

We compared the output of Falco[14] to its FastQC counterpart using 11 datasets (Table 2). The tests consist of Illumina files originating from a range of different library preparation protocols for DNA, RNA, and epigenetic experiments, as well as reads from the nanopore[17] technology. For simplicity, Illumina paired-end datasets were only tested on the first read end.

FASTQ files available in the Sequence Read Archive (SRA)[18] were downloaded using the fastq-dump command from the SRA toolkit. We used the following flags when running fastq-dump: -skip-technical, -readids, -read-filter pass, -dumpbase, -split-3 and -clip. One dataset was downloaded from the Whole Human Genome Sequencing Project[19].

We directly compared the text summary for each output of Falco to FastQC's output summary files, obtaining the same outputs (pass, warning, or fail) for all tested criteria in all datasets.

To assess if Falco's output is consistent with FastQC's format, we used the fastqcr[20] R package version 0.1.2 and MultiQC[12] version 1.9. Both tools can successfully parse the text reports generated by Falco for the tested files. Differences in the fastqc_data.txt files between the two programs result from choices for numerical precision output, or as a result of Falco calculating certain averages based on more of the data within each file.

Falco is faster than popular QC tools

Some alternative software tools exist for quality control of sequencing data, and users may opt for them due to their efficiency in cases where not all FastQC analysis modules are necessary. Among these, fastp[21] has gained popularity for its speed and versatile set of options for trimming. fastp has demonstrated superior runtime to FastQC even when generating FASTQ format output files corrected by trimming adapters and filtering (which requires both input and output). HTQC[22] is another tool that was developed with the intent to both improve speed performance and incorporate trimming functions after quality control. The two programs were used as benchmarks to compare Falco with.

Although most fastp modules are both calculated and displayed equivalently to FastQC, one major difference between these tools is how overrepresented sequences are estimated. While fastp counts the sequences at every *P* reads (which users may specify), FastQC stores the first 100,000 reads encountered for the first time, and subsequently checks if the following sequences match any of the stored candidates. This choice of implementation causes fastp's runtime to greatly differ when overrepresentation is enabled. Conversely, FastQC's runtime does not seem to be affected by disabling the overrepresented sequences module. For a comprehensive comparison between programs, we have measured the run times for our test datasets both with and without the overrepresented sequences module enabled. Programs were compared both in compressed (gzip FASTQ) and uncompressed (plain FASTQ) file formats.

Files used to assess Falco's output comparison to FastQC (Table 2) were also used for speed and memory comparison. Tests were executed in an Intel Xeon CPU E5-2640 v3 2.60GHz processor with a CentOS Linux 7 operating system. All file I/O was done using local disk to reduce variability in execution runtime. Both fastp and FastQC were instructed to run using a single thread.

FastQC version 0.11.8 was run with default parameters and the configuration limits, adapters and contaminants provided with the software. fastp version 0.20.0 was run with the -A, -G, -Q and -L flags to disable adapter trimming, poly-G

Table 2. Datasets used for comparison with FastQC's output and run time speed benchmarking between QC tools.

test	accession	reference	file size (FASTQ)	reads	length (bp)	protocol
1	SRR10124060	unpublished	7.3GB	25,172,255	130	RNA-Seq
2	SRR10143153	unpublished	11.0GB	15,949,900	150	miRNA-Seq
3	SRR3897196	23	4.2GB	15,570,348	100	BS-Seq
4	SRR9624732	24	1.6GB	18,807,797	150	ChIP-Seq
5	SRR1853178	25	130.0GB	510,210,716	60	Drop-Seq
6	SRR6387347	26	20.0GB	305,434,830	100	10x genomics
7	SRR6954584	5	56.0GB	152,853,013	150	Microwell-Seq
8	SRR891268	27	46.0GB	192,904,649	50	ATAC-Seq
9	SRR9878537	unpublished	38.0MB	3,284	64,000	Nanopore
10	wgs-FAB49164	19	8.4GB	746,333	180,000	Nanopore
11	SRR6059706	unpublished	1.4GB	892,313	150,000	Nanopore

trimming, quality filtering and length filtering, thus requiring the program to only perform QC tests without generating a new FASTQ file. When testing for overrepresented sequences, we set the `-p` flag to enable this module, and set the frequency of counts to the program's default value of $P = 20$. We ran the `ht-stat` program on the tested files using the `-S` flag for single-ended reads. `HTQC` was not tested on gzip FASTQ files as this file format is not accepted by the program. We used the GNU `time` command to measure the total running times for each program, using the total elapsed wall time as measurement. The benchmarking results (Table 3 and Table 4) show that `Falco` performs faster than `fastp` and `FastQC` in all datasets, with an average 3 times faster runtime than `FastQC`, both with the overrepresented sequences module on and off. Despite `HTQC` failing to process most test datasets due to unaccepted header formats, the two tests that ran to completion demonstrate that `Falco`'s analysis times are also significantly smaller in comparison.

The memory required to run `Falco` differs between short-read samples (tests 1-8; Table 2) and long-read samples (tests 9-11). All programs demonstrated similar behavior in memory usage, with all short-read samples having similar memory requirements, and test 10 requiring the most memory (as it contains the longest read). The total memory usage was also measured by GNU `time` command. For `Falco`, short-read samples required 92 MB of RAM, whereas long-read samples used at most 342 MB of RAM. In short-read samples, `FastQC` and `fastp`

Table 3. Real run times for `Falco`, `fastp` and `FastQC` on uncompressed `FASTQ` format, with the overrepresented sequences module on and off. Asterisks (*) indicate tests in which tools did not run to completion.

test	Falco	fastp	FastQC	Falco	fastp	FastQC	HTQC
	overrep off	overrep off	overrep off	overrep on	overrep on	overrep on	
1	48s	1m54s	3m30s	55s	5m57s	3m23s	12m09s
2	36s	1m20s	2m08s	37s	4m32s	2m10s	*
3	27s	1m04s	1m25s	30s	2m16s	1m24s	*
4	44s	1m48s	2m40s	51s	3m37s	2m38s	*
5	15m49s	35m14s	41m27s	15m58s	44m30s	37m43s	*
6	7m59s	18m42s	22m59s	8m33s	42m50s	22m53s	134m42s
7	6m05	13m50s	19m42s	6m49s	41m55s	19m52s	*
8	5m12s	11m47s	13m59s	5m20s	15m25s	14m08s	*
9	1s	1s	6s	1s	0m26s	6s	*
10	1m37s	1m50s	3m11s	1m32s	4m37s	3m16s	*
11	13s	24s	43s	13s	1m07s	46s	*

Table 4. Real run times for `Falco`, `fastp` and `FastQC` on gzip compressed FASTQ format.

test	Falco	fastp	FastQC	Falco	fastp	FastQC
	overrep off	overrep off	overrep off	overrep on	overrep on	overrep on
1	1m19s	2m19s	3m49s	1m25s	6m23s	3m50s
2	45s	1m31s	2m21s	51s	5m23s	2m24s
3	33s	1m10s	1m35s	35s	2m26s	1m36s
4	1m01s	2m06s	3m01s	1m03s	3m59s	3m00s
5	16m05s	42m40s	44m57s	18m17s	53m09s	44m59s
6	12m26s	23m18s	26m39s	12m29s	47m32s	26m38s
7	8m40s	17m34s	22m31s	8m34s	44m41s	22m31s
8	7m08s	14m37s	16m06s	6m31s	18m19s	16m11s
9	2s	1s	7s	1s	27s	7s
10	2m23s	2m32s	4m01s	2m34s	5m22s	4m09s
11	22s	31s	48s	23s	1m14s	51s

used 319 MB and 568 MB of RAM, respectively. In long-read samples, FastQC and fastp used at most 4.88 GB and 1.28 GB of RAM, respectively. This comparison suggests that Falco's memory requirement is also the lowest across all tests.

Falco allows dynamic visualization of results

Despite FastQC's clarity in its HTML reports, graphs are displayed as static images and have limited visualization flexibility, such as tile heatmaps not displaying raw deviations from average Phred scores in base positions, or raw values in line plots not being visible. We have opted to display Falco's analysis results using the Plotly JavaScript library[28], which allows interactive changes of axis labels, hovering on data points to visualize raw values, and screenshots from specific positions on the plot (Figure 1). This choice of presentation provides greater options to explore and interpret QC results while maintaining the visualization standards set by FastQC.

Conclusions

Falco[14] is a faster alternative to calculate the wide range of QC metrics reported by FastQC. It is entirely based on emulating the analysis modules FastQC provides while running faster than popular QC tools and generating dynamic visual summaries of analysis results. Falco's text output provides the same

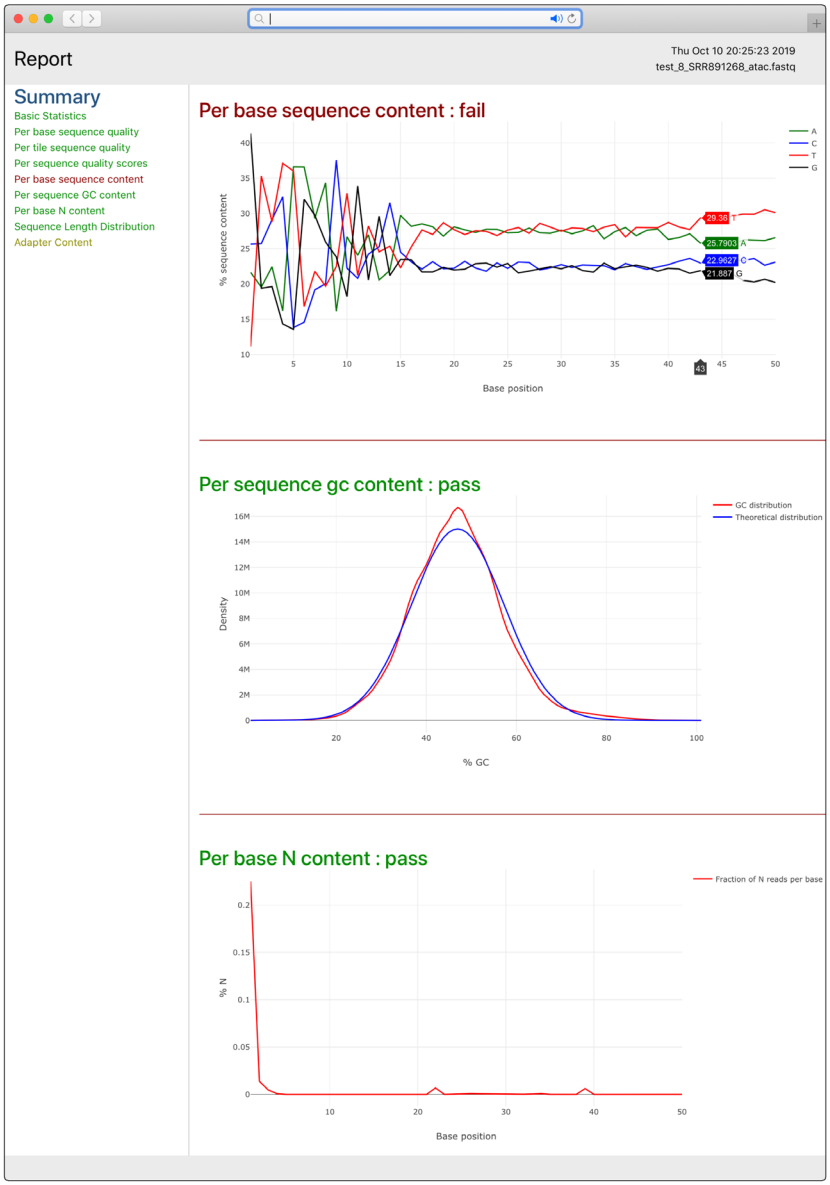

Figure 1. Sample HTML report for test 8 (accession SRR891268). Layout and plots are based on FastQC's HTML report.

information generated by FastQC, so tools that parse this file for custom visualization and downstream analysis can seamlessly incorporate Falco into their pipeline.

Data availability

Datasets used to compare Falco and FastQC are shown in Table 2. Guidance for how to accept accession wgs-FAB49164 is available from the Benchmark directory of the Falco GitHub page.

Software availability

Source code for Falco available at: https://github.com/smithlabcode/falco.

Users may report errors, bugs, installation problems, and improvement suggestions in the same page provided to download the source code under the "issues" section.

The scripts used to download files and reproduce the benchmarking steps described are also available in the same repository within the "benchmark" directory.

Archived source code at time of publication: http://doi.org/10.5281/zenodo.4429381[14].

License: GNU General Public License version 3.0.

References

1. Alkan C, Kidd JM, Marques-Bonet T, *et al.*: **Personalized copy number and segmental duplication maps using next-generation sequencing.** *Nature Genetics.* 2009; **41**(10): 1061–1068.
PubMed Abstract | Publisher Full Text | Free Full Text
2. Loman NJ, Quick J, Simpson JT: **A complete bacterial genome assembled *de novo* using only nanopore sequencing data.** *Nature Methods.* 2015; **12**(8): 733–738.
PubMed Abstract | Publisher Full Text
3. Masella AP, Bartram AK, Truszkowski JM, *et al.*: **PANDAseq: paired-end assembler for illumina sequences.** *BMC Bioinformatics.* 2012; **13**(1): 31.
PubMed Abstract | Publisher Full Text | Free Full Text
4. Ozsolak F, Milos PM: **RNA sequencing: advances, challenges and opportunities.** *Nature Reviews Genetics.* 2011; **12**(2): 87–98.
PubMed Abstract | Publisher Full Text | Free Full Text
5. Han X, Wang R, Zhou Y, *et al.*: **Mapping the mouse cell atlas by Microwell-Seq.** *Cell.* 2018; **172**(5): 1091–1107.e17.
PubMed Abstract | Publisher Full Text
6. Buenrostro JD, Wu B, Chang HY, *et al.*: **ATAC-seq: A method for assaying chromatin accessibility genome-wide.** *Current Protocols in Molecular Biology.* 2015; **109**(1): 21.29.1–9.
PubMed Abstract | Publisher Full Text | Free Full Text
7. Datlinger P, Rendeiro AF, Schmidl C, *et al.*: **Pooled CRISPR screening with single-cell transcriptome readout.** *Nature Methods.* 2017; **14**(3): 297–301.
PubMed Abstract | Publisher Full Text | Free Full Text
8. Spanjaard B, Hu B, Mitic N, *et al.*: **Simultaneous lineage tracing and cell-type identification using CRISPR–Cas9-induced genetic scars.** *Nature Biotechnology.* 2018; **36**(5): 469–473.
PubMed Abstract | Publisher Full Text | Free Full Text
9. Svensson V, Vento-Tormo R, Teichmann SA: **Exponential scaling of single-cell RNA-seq in the past decade.** *Nature Protocols.* 2018; **13**(4): 599–604.
PubMed Abstract | Publisher Full Text
10. Ewing B, Hillier L, Wendl MC, *et al.*: **Base-calling of automated sequencer traces using Phred.** *Genome Res.* 1998; **8**(3): 175–185.
PubMed Abstract | Publisher Full Text
11. Andrews S: **FastQC: a quality control tool for high throughput sequence data**. 2010.
Reference Source
12. Ewels P, Magnusson M, Lundin S, *et al.*: **MultiQC: summarize analysis results for multiple tools and samples in a single report.** *Bioinformatics.* 2016; **32**(19): 3047–3048.
Publisher Full Text
13. Brown J, Pirrung M, McCue LA: **FQC Dashboard: integrates FastQC results into a web-based, interactive, and extensible FASTQ quality control tool.** *Bioinformatics.* 2017; **33**(19): 3137–3139.
PubMed Abstract | Publisher Full Text | Free Full Text
14. De Sena Brandine G, Smith A: **smithlabcode/falco: 0.2.4 - 2019/10/28**. 2019.
http://www.doi.org/10.5281/zenodo.4429381
15. Deutsch P, Gailly JL: **Zlib compressed data format specification version 3.3**. 1996.
Publisher Full Text
16. Li H, Handsaker B, Wysoker A, *et al.*: **The Sequence Alignment/Map format and SAMtools.** *Bioinformatics.* 2009; **25**(16): 2078–2079.
PubMed Abstract | Publisher Full Text | Free Full Text
17. Jain M, Olsen HE, Paten B, *et al.*: **The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community.** *Genome Biology.* 2016; **17**(1): 239.
PubMed Abstract | Publisher Full Text | Free Full Text
18. Leinonen R., Sugawara H., Shumway M, *et al.*: **The sequence read archive.** *Nucleic Acids Res.* 2010; **39**(Database issue): D19–D21.
PubMed Abstract | Publisher Full Text | Free Full Text
19. Jain M, Koren S, Miga KH, *et al.*: **Nanopore sequencing and assembly of a human genome with ultra-long reads.** *Nature Biotechnology.* 2018; **36**(4): 338–345.
PubMed Abstract | Publisher Full Text | Free Full Text
20. Kassambara A: **fastqcr: Quality control of sequencing data**. R package version 0.1.2. 2019.
Reference Source
21. Chen S, Zhou Y, Chen Y, *et al.*: **fastp: an ultra-fast all-in-one FASTQ preprocessor.** *Bioinformatics.* 2018; **34**(17): i884–i890.
Publisher Full Text
22. Yang X, Liu D, Liu F, *et al.*: **HTQC: a fast quality control toolkit for Illumina sequencing data.** *BMC Bioinformatics.* 2013; **14**(1): 33.
PubMed Abstract | Publisher Full Text | Free Full Text
23. Decato BE, Lopez-Tello J, Sferruzzi-Perri AN, *et al.*: **DNA methylation divergence and tissue specialization in the developing mouse placenta.** *Molecular Biology and Evolution.* 2017; **34**(7): 1702–1712.
PubMed Abstract | Publisher Full Text | Free Full Text
24. Yang J, Zhang L, Jiang Z, *et al.*: **TCF12 promotes the tumorigenesis and metastasis of hepatocellular carcinoma via upregulation of CXCR4 expression.** *Theranostics.* 2019; **9**(20): 5810–5827.
PubMed Abstract | Publisher Full Text | Free Full Text
25. Macosko EZ, Basu A, Satija R, *et al.*: **Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets.** *Cell.* 2015; **161**(5): 1202–1214.
PubMed Abstract | Publisher Full Text | Free Full Text
26. Nusse YM, Savage AK, Marangoni P, *et al.*: **Parasitic helminths induce fetal-like reversion in the intestinal stem cell niche.** *Nature.* 2018; **559**(7712): 109–113.
PubMed Abstract | Publisher Full Text | Free Full Text
27. Buenrostro JD, Giresi PG, Zaba LC, *et al.*: **Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position.** *Nature Methods.* 2013; **10**(12): 1213–1221.
PubMed Abstract | Publisher Full Text | Free Full Text
28. Sievert C, Parmer C, Hocking T, *et al.*: **plotly: Create interactive web graphics via 'plotly. js'**. 2017.
Reference Source

Open Peer Review

Current Peer Review Status: ✔ ✔

Version 2

Reviewer Report 28 January 2021

https://doi.org/10.5256/f1000research.46637.r78323

✔ **R. Henrik Nilsson**
Gothenburg Global Biodiversity Centre, University of Gothenburg, Gothenburg, Sweden

I am happy with the revised manuscript*. It packs quite some punch.

But I attach some few discretionary, cosmetic suggestions in the below.

Methods:

- ”is Open Source” – shouldn’t this be ”is open source” here? Not a proper noun and not a capitonym, after all?
- “calculations, results and text reports.” > “calculations, results, and text reports.”

Results:

- “configuration limits, adapters and contaminants” > “configuration limits, adapters, and contaminants”.
- “trimming, quality filtering and length filtering” > “trimming, quality filtering, and length filtering”.
- “Real run times for Falco, fastp and FastQC” > ““Real run times for Falco, fastp, and FastQC””.

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Metabarcoding ; molecular ecology ; systematics ; mycology

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Reviewer Report 30 October 2020

https://doi.org/10.5256/f1000research.23273.r72941

Weihong Qi
Functional Genomics Center Zurich, Zürich, Switzerland

The authors developed falco, an emulation of the popular FastQC tool, which is faster and can handle very long Nanopore reads. It is a useful development, especially for core facilities and research labs that produce high volumes of sequencing data regularly, where generating read QC reports in a timely fashion is indeed helpful. I only have a few questions and one minor comment:

Main questions:

1. The implementation session could be expanded with more details. From my understanding, the major improvement was identified duplicated analysis in FastQC analysis modules, and implemented a single analysis workflow that was sufficient to generate the same modularized results. But it is not clear to me which changes make falco to handle long ONT reads successfully, while the original FastQC failed.

2. The original FastQC is portable (Unix. Mac and Windows). It also has a GUI version for less experienced users. These features are not important for experienced users and automated workflows where analyzing large amounts of data in a short time is the focus. But they can be important for other type of end users. The authors should at least point out these differences.

3. In results, run times of multiple QC tools analyzing different datasets were compared, how about the RAM usages?

Minor comment:
The sentence “While FastQC is capable of making summaries for protocols such as 454[27] PacBio[28], which generate sequences with around 10,000 bases per read ” should be updated to "While FastQC is capable of making summaries for protocols such as 454[27] PacBio[28], which generate sequences with around 10,000-20,000 bases per read".

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Genome informatics.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 09 Jan 2021

Guilherme de Sena Brandine, University of Southern California, Los Angeles, USA

The reviewer has raised some important questions about points not addressed by the manuscript. We provide our responses below, and highlight changes made to the manuscript to address the reviewer's comments.

The authors developed falco, an emulation of the popular FastQC tool, which is faster and can handle very long Nanopore reads. It is a useful development, especially for core facilities and research labs that produce high volumes of sequencing data regularly, where generating read QC reports in a timely fashion is indeed helpful. I only have a few questions and one minor comment:

Main questions:

1) The implementation session could be expanded with more details. From my understanding, the major improvement was identified duplicated analysis in FastQC analysis modules, and implemented a single analysis workflow that was sufficient to generate the same modularized results. But it is not clear to me which changes make falco to handle long ONT reads successfully, while the original FastQC failed.

We really appreciate the reviewer highlighting the missing details regarding FastQC's behavior. Upon trying to address this comment, we have further explored the FastQC code to understand why it failed for long reads, and we have learned that the perl script that wraps the FastQC call imposes a maximum memory limit of 250 MB per thread, which we were not aware of at the time we wrote the manuscript, and was prohibitive for the long read samples we have selected. Changing this configuration internally allowed us to run FastQC in the samples that we were previously unable to, thus allowing us to report a more comprehensive comparison of both time and memory for every test we have gathered. For

this reason, we have removed the section named “Falco scales for larger nanopore reads”, instead replacing it with a paragraph at the end of the subsection named “Falco is faster than popular QC tools”, where the memory usage of each mapper is discussed. We have also filled Tables 3 and 4 with the results of running FastQC in the three long-read samples in our tests under the same hardware settings used for other tests.

2) The original FastQC is portable (Unix. Mac and Windows). It also has a GUI version for less experienced users. These features are not important for experienced users and automated workflows where analyzing large amounts of data in a short time is the focus. But they can be important for other type of end users. The authors should at least point out these differences.

We thank the reviewer for this observation. We have made modifications to the first paragraph of the “implementation choices” subsection under “methods” to clarify that falco was designed for UNIX systems and does not include a graphical user interface.

3) In results, run times of multiple QC tools analyzing different datasets were compared, how about the RAM usages?

We appreciate the observation about RAM comparison. We have added two paragraphs in the manuscript that address memory requirement in more detail. A “system requirement” subsection under “methods” was added to emphasize that falco requires about 100 MB for short read samples and under 1 GB for samples with read lengths of at most 1 million. As stated in question (1), we also reported the RAM usage for the software tools compared in the tests, both for short-read and long-read tests in the section “Falco is faster than popular QC tools”.

Minor comment:
The sentence “While FastQC is capable of making summaries for protocols such as 454[27] PacBio [28], which generate sequences with around 10,000 bases per read ” should be updated to "While FastQC is capable of making summaries for protocols such as 454[27] PacBio[28], which generate sequences with around 10,000-20,000 bases per read".

We thank the reviewer for the observation. The section containing this sentence was removed given the shift in the focus of the manuscript to memory comparison, as addressed in questions (1) and (3).

Competing Interests: No competing interests were disclosed.

Reviewer Report 07 July 2020

https://doi.org/10.5256/f1000research.23273.r66327

?

R. Henrik Nilsson
Gothenburg Global Biodiversity Centre, University of Gothenburg, Gothenburg, Sweden

The authors present a welcome addition to the flora of FastQC-style read processing packages. The fact that it is a drop-in replacement for FastQC is particularly nice.

The manuscript is a bit too short in my opinion. I miss some background information and some performance-related data. If the authors want to address a wide audience, they should probably work a bit more on the installation instructions and documentation too. The authors should probably also consider defining who their target audience is.

Introduction:

- As a discretionary comment: the authors sometimes, but not always, use the Oxford comma (see example below). I wonder if this is something that should be streamlined. "sequencing data is growing in abundance, dataset size and read length[9]." vs. "...adapter trimming, filtering contaminants and low-quality reads, and mapping reads to a reference genome or transcriptome."

- "as a safety criteria" should probably be "as a safety criterion".

Methods:

- Good thinking behind the "We designed falco[13] to faithfully..." paragraph. Nobody would be helped by yet another set of new file formats. A drop-in replacement is the way to go, if you ask me. And that is indeed what the authors deliver.

- I must make the observation, though, that the name "falco" may not be available (?): https://www.falcoseed.com/ca/article/cibus-registers-new-falco-brand-79k-canola-hybrid/ (some other more or less commercial uses of "Falco" can be found on https://en.wikipedia.org/wiki/Falco). I'm not sure how North American trademark/proprietary laws operate, but I suggest that the authors consult with the lawyers of their university. Better safe than sorry, right. (I consulted with our lawyers at one point, and they had me change the name of a software package we were working on at the time.)

- Two unintended line breaks:

  "uniformly,
  falco's design centralizes"

Results:

- Please cite the Sequence Read Archive formally; see Kodama *et al.* (2012[1]).

- I like the reproducible nature of the "Results" section.

- Two superfluous line breaks:

  "gzip
  compressed files."

Conclusions:

- Both "Falco" and "falco" are used in the manuscript. You'd think that the name would be fixed as either "Falco" or "falco".

Software availability:

- Why not use the term "open source" at least once in the manuscript?

Miscellaneous questions and observations:

- Out of curiosity: how does falco compare to Liu *et al.* (2019[2])?

- I've seen one software tool for quality-score-based trimming of sequences that actually reads the entire query file into memory, and then started to process it. This works less well as data files continues to grow, obviously. Is it worth pointing out that falco does not do this? What is, in fact, the maximum file size allowed by falco? Or is this dictated solely by the operating system?

- You can produce some pretty funny behavior in some other tools for sequence QC/trimming by feeding them a file with a single FastQC entry in it, speaking of nothing. Is there, then, a minimum file size or number of query sequences for falco?

- Unless I'm mistaken, there are no fastq files available on https://github.com/smithlabcode/falco. I think the authors should make one or two available, so that it will be smooth and easy for the reader to try the software out. "The scripts used to download files and reproduce the bench-marking steps described are also available in the same repository within the "benchmark" directory." comes across as somewhat indirect to me.

- What, exactly, are the hardware and software requirements for falco? This may be worth pointing out. Between the lines I read "any computer you can install the GNU GCC compiler on" – but not all readers will probably read it this way. Instead you'll get the question: "Does it run on Windows 10?".

- Also, how much memory is needed to run falco? And how much memory is used up by falco when it processes a large file?

- Between the lines ("Programs were instructed to run using a single thread."), I take it that falco can use multiple threads. Is that correct? And if so, why not point it out more explicitly? And out of curiosity: when you run falco on 4 cores, do you see a 4x speedup? Or is the bottleneck something else (disk IO?) than raw computational power? How does it scale with the number of cores, in other words?

- Suppose my dataset is 10 Gb, and that I have 15 Gb left of free disk space. Do I dare to run

falco on that dataset? How large is the output compared to the input? In a “minimum” (no extra features) as well as “maximum” (all extra features) mode?

- Suppose a user finds a bug, or wants to put forward a feature request. How can the user do that? Should this be mentioned in the manuscript?

- The first question I always get from users of my software tools is: “where can I download the Windows binaries?”. Is it, actually, not a good idea to be explicit about the fact that this is a command-line tool that you compile on your own computer? Would probably save the authors some time to be upfront with this.

- The list of references comes across as somewhat untidy. Some examples follow below. The authors should probably go through all references to make sure they comply with journal specifications.

- () Journal names are sometimes abbreviated, sometimes not. Ref 2 is abbreviated, whereas ref 3 should be abbreviated “BMC Bioinform.”

- () Article titles: should verbs and key nouns in article titles have a leading uppercase letter, as in, e.g., ref 5, or should they not, as in ref 1?

- () Should page ranges be written out in full (ref 23, “1202–1214.”) or should they be abbreviated (ref 25, “1213–8.”)?

Are the installation instructions a bit too thin? I’d say yes, at least if the intention of the authors is to address a diverse audience and not just readers with Linux-style experience. To simulate a less experienced user, I used my son’s MacBook Pro and tried to install falco on it following the manual:

“Upon downloading, inflating and moving to the source
directory, installation can be done through the following commands:
...
$ ./configure CXXFLAGS="-O3 -Wall"
$ make all
$ make install”

So I did:

$ cd src
$ ./configure CXXFLAGS="-O3 -Wall"
-bash: ./configure: No such file or directory

And then

conda install -c bioconda falco
-bash: conda: command not found

And that was it. No further clues or assistance to be found in the instructions.

If the authors are happy with this behavior, then they should make it clear in the manuscript that falco is not for everyone, but rather only for those with significant Linux-style experience.

"Source code for falco available at: https://github.com/smithlabcode/falco." – the trailing "." should be removed, I'd say. The link won't work for users who copy-and-paste it into their browser. The same thing goes for

"Archived source code at time of publication: http://doi.org/10.5281/zenodo.3520933[13]." where both the reference and the "." cause problems.

References

1. Kodama Y, Shumway M, Leinonen R, International Nucleotide Sequence Database Collaboration: The Sequence Read Archive: explosive growth of sequencing data.*Nucleic Acids Res*. 2012; **40** (Database issue): D54-6 PubMed Abstract | Publisher Full Text
2. Liu X, Yan Z, Wu C, Yang Y, et al.: FastProNGS: fast preprocessing of next-generation sequencing reads.*BMC Bioinformatics*. 2019; **20** (1): 345 PubMed Abstract | Publisher Full Text

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Metabarcoding ; molecular ecology ; systematics ; mycology

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 09 Jan 2021

Guilherme de Sena Brandine, University of Southern California, Los Angeles, USA

The reviewer has presented a thorough feedback to the description of the Falco software tool, as well as its implementation, description and documentation. We truly appreciate the very helpful comments, and provide our responses to improvement suggestions below. Comments were divided and numbered to allow us to refer to them in other places in our response if certain modifications to the manuscript pertain to multiple comments.

(1) The authors present a welcome addition to the flora of FastQC-style read processing packages. The fact that it is a drop-in replacement for FastQC is particularly nice.

The manuscript is a bit too short in my opinion. I miss some background information and some performance-related data. If the authors want to address a wide audience, they should probably work a bit more on the installation instructions and documentation too. The authors should probably also consider defining who their target audience is.

We appreciate the comments on the introduction. We have expanded on the target audience on further comments (we address these in more detail on comment 16). We also fully agree that more background information can be provided. We expanded the second paragraph of the "introduction" section to add a brief description of some common quality control tests applied to most next-generation sequencing datasets.

(2) As a discretionary comment: the authors sometimes, but not always, use the Oxford comma (see example below). I wonder if this is something that should be streamlined. "sequencing data is growing in abundance, dataset size and read length[9]." vs.
"...adapter trimming, filtering contaminants and low-quality reads, and mapping reads to a reference genome or transcriptome."

We thank the reviewer for this observation. We have revised the manuscript and ensured that the Oxford comma is adopted across the entire manuscript.

(3) "as a safety criteria" should probably be "as a safety criterion".

We fully agree with the reviewer. This correction was made on the manuscript.

(4) Good thinking behind the "We designed falco[13] to faithfully..." paragraph. Nobody would be helped by yet another set of new file formats. A drop-in replacement is the way to go, if you ask me. And that is indeed what the authors deliver.

We appreciate the comments, thank you!

(5) I must make the observation, though, that the name "falco" may not be available (?): https://www.falcoseed.com/ca/article/cibus-registers-new-falco-brand-79k-canola-hybrid/ (some other more or less commercial uses of "Falco" can be found on https://en.wikipedia.org/wiki/Falco). I'm not sure how North American trademark/proprietary laws operate, but I suggest that the authors consult with the lawyers of their university. Better safe than sorry, right. (I consulted with our lawyers at one point, and they had me change the

name of a software package we were working on at the time.)

We really appreciate the reviewer raising the possible legal issue that may arise from the program name. We have researched the matter and believe that the software name should not raise legal issues, both due to the program not having any commercial or profitable goals and the name "Falco" being a common proper noun in many Latin languages.

(6) Two unintended line breaks: "uniformly, falco's design centralizes"

We thank the reviewer for this observation. This line was removed from the manuscript.

(7) Please cite the Sequence Read Archive formally; see Kodama et al. (2012[1]).

We followed the reviewer's suggestion and added the appropriate citation when referring to the Sequence Read Archive. We also corrected the meaning of the acronym from "Sequencing Read Archive", as it was previously written.

(8) I like the reproducible nature of the "Results" section.

We appreciate the comment, thank you!

(9) Two superfluous line breaks: "gzip compressed files."

We have removed the line break from the manuscript.

(10) Both "Falco" and "falco" are used in the manuscript. You'd think that the name would be fixed as either "Falco" or "falco".

We thank the reviewer for this observation. The notation used in the manuscript used "Falco" at the start of sentences and "falco" everywhere else, to resemble the name of the binary program used in the command-line interface. We have modified the manuscript to use "Falco" everywhere except in the "use cases" section, where an example command-line call for the program is shown.

(11) Why not use the term "open source" at least once in the manuscript?

We thank the reviewer for this observation. To address this, we started the "Implementation choices" subsections with the following sentence: *Falco is an Open Source C++ implementation of the FastQC software tool built for UNIX-based operating systems.*

(12) Out of curiosity: how does falco compare to Liu et al. (2019[2])?

We have downloaded the software from the URL provided in the manuscript (github.com/megagenomics/fastprongs) and performed comparisons on identical hardware to what was used in the manuscript. We tested on two datasets: Dataset 1 consisted of 76 million 150 base reads from arabidopsis (SRR12075121 in SRA), and dataset 2 contained 139 million 100 base reads from chicken (SRR5015166 in SRA). On dataset 1, Falco ran in 9:40

and FastProNGS ran in 5:16 with 3 threads (the default program configuration) and 10:42 on a single thread. On dataset 2, Falco ran in 14:30 and FastProNGS ran in 7:55 with 3 threads and 16:13 with a single thread. We noticed that FastProNGS only reports the following modules in their output: Basic statistics, adapter content, per base sequence quality, per base sequence content and sequence length distribution. We also tried to run Falco by enabling only these modules. Under these settings Falco ran in 4:35 for dataset 1 and 7:12 for dataset 2. In all tests, Falco ran with 92 MB of RAM, whereas FastProNGS used 1.26 GB. In a long-read dataset (test 12 in the manuscript), FastProNGS did not run successfully unless we configured it to only consider the first 200 bases of each read, in which case FastProNGS ran in 2 seconds, but only reported summaries for the first 200 bases of all reads. Falco ran in 12 seconds for this dataset.

A very meaningful conclusion of this comparison is the potential advantage of multithreading in QC, as evidenced by the steep decrease in processing time from FastProNGS when multithreading is enabled. We noticed, upon inspecting its source code, that this performance improvement can be explained by FastProNGS reading multiple reads in batch and allowing a new set of reads to be loaded while the previous batch of reads is processed. In contrast, Falco loads and processes each read sequentially, which reduces RAM usage but makes multithreading difficult in its current implementation. The performance of FastProNGS suggests that switching to a "batch processing" paradigm may have significant speed advantages when multiple cores are used and enough RAM is available to load reads in batch, and this is something we will incorporate in future versions of Falco, especially in order to address (18). We thank the reviewer for bringing this tool to our attention.

(13) I've seen one software tool for quality-score-based trimming of sequences that actually reads the entire query file into memory, and then started to process it. This works less well as data files continues to grow, obviously. Is it worth pointing out that falco does not do this? What is, in fact, the maximum file size allowed by falco? Or is this dictated solely by the operating system?

Both disk and memory requirements will depend on the length of the largest read in the dataset, as Falco processes the input FASTQ one read at a time. To address the computational requirements necessary to run Falco in more detail in the manuscript, we created an additional subsection named "system requirements" under the "Methods" section, where the computational resources (memory and disk) required to run Falco successfully are discussed. Furthermore, we have added a paragraph in the "results" section summarizing the memory and disk usage for the tests used for comparison across programs.

(14) You can produce some pretty funny behavior in some other tools for sequence QC/trimming by feeding them a file with a single FastQC entry in it, speaking of nothing. Is there, then, a minimum file size or number of query sequences for falco?

There are no minimum or maximum file sizes required by Falco. We have tested (although not disclosed in the manuscript) that Falco successfully runs on empty files and single-read files. We thank the reviewer for having raised this issue, and have addressed that there are no constraints in file size or number of reads in the "system requirements" section stated in

(13).

(15) Unless I'm mistaken, there are no fastq files available on https://github.com/smithlabcode/falco. I think the authors should make one or two available, so that it will be smooth and easy for the reader to try the software out. "The scripts used to download files and reproduce the benchmarking steps described are also available in the same repository within the "benchmark" directory." comes across as somewhat indirect to me.

We thank the reviewer for this observation. While we cannot provide the full FASTQ files used to perform our comparisons in the GitHub repository, we do agree that the documentation of our tests can be made simpler for users who wish to test the program. We made modifications in our repository to simplify both testing in an example file and testing in the FASTQ files used in the manuscript for comparison. Specifically, we added (1) direct links to the SRA files under the "benchmark" directory and (2) an "example.fq" file, consisting of a FASTQ file of 1000 reads, which is used as input for the example commands provided in the README.

(16) What, exactly, are the hardware and software requirements for falco? This may be worth pointing out. Between the lines I read "any computer you can install the GNU GCC compiler on" – but not all readers will probably read it this way. Instead, you'll get the question: "Does it run on Windows 10?".

We agree that constraints should be disclosed in more detail, and that the limited support for usage of Falco on Windows should be more explicit. We have rephrased the first paragraph in the "implementation choices". The last sentences disclose more explicitly that Falco, by design, is a UNIX-centric program made to be run on a command line and that, unlike FastQC, it cannot be run in a graphical user interface.

(17) Also, how much memory is needed to run falco? And how much memory is used up by falco when it processes a large file?

Falco requires under 1 GB of memory for any short or long read file generated by the current sequencing technologies. More memory will be required when technologies expand read lengths to the order of millions or billions of bases per read. We have added a discussion of disk and memory requirements under the "systems requirement" section, and also discussed the memory usage of the programs compared in the manuscript under the section "Falco is faster than popular QC tools".

(18) Between the lines ("Programs were instructed to run using a single thread."), I take it that falco can use multiple threads. Is that correct? And if so, why not point it out more explicitly? And out of curiosity: when you run falco on 4 cores, do you see a 4x speedup? Or is the bottleneck something else (disk IO?) than raw computational power? How does it scale with the number of cores, in other words?

Falco, like FastQC currently does not use multiple threads to process a single file, and no significant speed difference was observed when running fastp with multiple threads, which is why we focused our comparison on single-thread across the software tools. Despite QC

computations being fast relative to IO, our comparison with FastProNGS described in (12) suggests that multithreading can lead to speed improvements if reading and processing are done in parallel, and we certainly plan on exploring this paradigm in the next release of Falco. We have also rephrased the third paragraph of the subsection "Falco is faster than popular QC tools" to say "Both fastp and fastqc were instructed to run on a single thread" to avoid ambiguities regarding Falco's multithread option.

(19) Suppose my dataset is 10 Gb, and that I have 15 Gb left of free disk space. Do I dare to run falco on that dataset? How large is the output compared to the input? In a "minimum" (no extra features) as well as "maximum" (all extra features) mode?

We have addressed the disk requirement on the "system requirements" section disclosed in (13) and (14), specifically adding the sentence "The total disk space required to store the three output files generated by Falco is under 1 MB". Like FastQC, Falco's output is a set of reports whose size scales with the maximum read length of the input but are never under 1 MB in total. We fully agree that the fact that disk space is not crucial to run Falco should be made more explicit.

(20) Suppose a user finds a bug, or wants to put forward a feature request. How can the user do that? Should this be mentioned in the manuscript?

We thank the reviewer for this observation. All issues and bug reports can be done through our GitHub page, the same one provided for the source code. To make this clearer for users, we have added a sentence at the "software availability" section, stating that errors, installation problems and bugs can be reported in the "Issues" section in the same URL provided to download the source code.

(21) The first question I always get from users of my software tools is: "where can I download the Windows binaries?". Is it, actually, not a good idea to be explicit about the fact that this is a command-line tool that you compile on your own computer? Would probably save the authors some time to be upfront with this.

We fully agree with the reviewer, and have added the statements of a more specific target audience as discussed in (16).

(22) The list of references comes across as somewhat untidy. Some examples follow below. The authors should probably go through all references to make sure they comply with journal specifications.
Journal names are sometimes abbreviated, sometimes not. Ref 2 is abbreviated, whereas ref 3 should be abbreviated "BMC Bioinform."
Article titles: should verbs and key nouns in article titles have a leading uppercase letter, as in, e.g., ref 5, or should they not, as in ref 1?
Should page ranges be written out in full (ref 23, "1202–1214.") or should they be abbreviated (ref 25, "1213–8.")?

We really appreciate the keen observations on the reference standards. We have reviewed our citations and standardized journals to their non-abbreviated names, uppercase letters

only in leading words, and pages written in full.

(26) Are the installation instructions a bit too thin? I'd say yes, at least if the intention of the authors is to address a diverse audience and not just readers with Linux-style experience. To simulate a less experienced user, I used my son's MacBook Pro and tried to install falco on it following the manual:

"Upon downloading, inflating and moving to the source directory, installation can be done through the following commands:
...
$ ./configure CXXFLAGS="-O3 -Wall"
$ make all
$ make install"

So I did:

$ cd src
$ ./configure CXXFLAGS="-O3 -Wall"
-bash: ./configure: No such file or directory

And then

conda install -c bioconda falco
-bash: conda: command not found

And that was it. No further clues or assistance to be found in the instructions.

If the authors are happy with this behavior, then they should make it clear in the manuscript that falco is not for everyone, but rather only for those with significant Linux-style experience.

We really appreciate the reviewer bringing up this observation about our documentation. We agree that the wording of "moving to the source directory" was misleading and may cause users to try to run the commands on the "src" directory. We have updated our README with clearer command line instructions that show the user how to clone the repository or download a release file, as well as which directory to move to in order to run the commands. We are striving to make the documentation as clear and simple as possible, and truly appreciate these suggestions on how these can be improved.

(27) "Source code for falco available at: https://github.com/smithlabcode/falco." – the trailing "." should be removed, I'd say. The link won't work for users who copy-and-paste it into their browser. The same thing goes for
"Archived source code at time of publication: http://doi.org/10.5281/zenodo.3520933[13]." where both the reference and the "." cause problems.

We appreciate this observation and the potential problems punctuation near links may cause. We have ensured that the trailing dots and citations are clearly separated from the links and that they will not cause problems when copying URLs directly from the

manuscript.

Competing Interests: No competing interests were disclosed.