The Journal of Open Source Software

Nanoq: fast quality control for nanopore reads

Eike Steinig[1] **and Lachlan Coin**[1]

1 The Peter Doherty Institute for Infection and Immunity, The University of Melbourne, Australia

DOI: 10.21105/joss.02991

Software

- Review
- Repository
- Archive

Editor: Luiz Irber

Reviewers:

- @natir
- @bovee

Submitted: 24 January 2021
Published: 27 January 2021

Summary

Nanopore sequencing is now routinely integrated in a variety of genomics applications, including whole genome assembly (Jain et al., 2018) and real-time infectious disease surveillance (Meredith et al., 2020). As a consequence, the amount of nanopore sequence data in the public domain has increased rapidly in the last few years. One of the first steps in any workflow is to assess the quality of reads and obtain basic summary statistics after basecalling raw nanopore signal, and to filter low quality reads. `NanoPack` (`biopython` parser) (De Coster et al., 2018), `Filtlong` (`Klib` parser) and `MinIONQC` (summary file parser) (Lanfear et al., 2018) are common tools used to filter and obtain summary statistics from nanopore reads. However, these tools can be relatively slow due to bottlenecks in read parsing (`NanoPack`, `Filtlong`), are not immediately usable due to reliance on summary files (`MinIONQC`), or focus on data exploration and visualization. We therefore implement `nanoq`, a command line tool to accelerate summary and quality control for nanopore reads in `Rust`.

Statement of need

A common practice for quality control and filtering of reads for length and quality is to use a sequencing summary file as index to speed up iteration and computation over millions of individual reads and their precomputed metrics from the basecalling process (e.g. the main acess mode for `MinIONQC`), which requires access to signal level data or shared summary files. With increasing throughput on scalable nanopore platforms like GridION or PromethION, fast quality control of sequence reads and the ability to generate summary statistics on-the-fly are required. `Nanoq` is highly competitive in processing speed (see benchmarks) and can be effectively applied to nanopore data from the public domain, where sequencing summaries are unavailable, as part of automated pipelines, in streaming applications, or directly from the command line to check on the progress of active sequencing runs.

Applications

`Nanoq` is implemented in `Rust` using the read parsers from `needletail` and `Rust-Bio` (Köster, 2015).

Tests can be run within the `nanoq` repository:

```
cargo test
```

`Nanoq` accepts a file or stream of sequence reads in `fast{a/q}` and compressed formats on `stdin`:

```
cat test.fq | nanoq
```

Basic summary statistics are output to stderr:

```
100000 400398234 5154 44888 5 4003 3256 8.90 9.49
```

- number of reads
- number of base pairs
- N50 read length
- longest and shorted reads
- mean and median read length
- mean and median read quality

Extended output analogous to NanoStat can be obtained using multiple --detail flags:

```
cat test.fq | nanoq -d -d -d
```

Reads filtered by minimum read length (--length) and mean read quality (--quality) are output to stdout:

```
cat test.fq | nanoq -l 1000 -q 10 > reads.fq
```

Advanced two-pass filtering analogous to Filtlong removes the worst 20% of bases using sorted reads by quality (--keep_percent) or the worst quality reads until approximately 500 Mbp remain (--keep_bases):

```
nanoq -f test.fq -p 80 -b 500000000  > reads.fq
```

Live sequencing run data directory:

```
RUN=/data/nanopore/run
```

Check total run statistics of active run:

```
find $RUN -name *.fastq -print0 | xargs -0 cat | nanoq
```

Check per-barcode statistics of active run:

```
for i in {01..12}; do
  find $RUN -name barcode${i}.fastq -print0 | xargs -0 cat | nanoq
done
```

Benchmarks

Benchmarks evaluate processing speed of a long-read filter and computation of summary statistics on the first 100,000 reads (test.fq.gz in Docker container) of the even Zymo mock community (Nicholls et al., 2019) (GridION) using the nanoq:v0.2.0 Benchmark image with comparison to NanoFilt, NanoStat and Filtlong

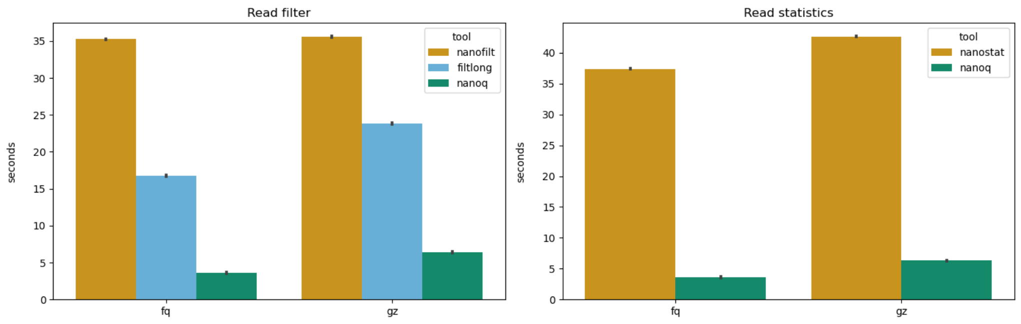

Figure 1: Nanoq benchmarks compared to Filtlong and Nanopack on 100,000 reads of the Zymo mock community

program	ftype	task	mean sec (+/- sd)	~ reads / sec	speedup
nanofilt	fq	filter	35.25 (0.35)	2,836	1.00 x
filtlong	fq	filter	16.71 (0.47)	5,984	2.11 x
nanoq	fq	filter	03.63 (0.45)	27,548	9.71 x
nanostat	fq	stats	37.39 (0.50)	2,674	1.00 x
nanoq	fq	stats	03.57 (0.57)	28,011	10.4 x
nanofilt	fq.gz	filter	35.58 (0.36)	2,810	1.00 x
filtlong	fq.gz	filter	23.84 (0.60)	4,195	1.49 x
nanoq	fq.gz	filter	06.37 (0.41)	14,858	5.28 x
nanostat	fq.gz	stats	42.21 (0.37)	2,369	1.00 x
nanoq	fq.gz	stats	06.30 (0.28)	15,873	6.70 x

Availability

`Nanoq` is open-source on GitHub (https://github.com/esteinig/nanoq) and available through:

- Cargo: `cargo install nanoq`
- Docker: `docker pull esteinig/nanoq`
- BioConda: `conda install -c bioconda nanoq`
- Singularity: `singularity pull docker://esteinig/nanoq`

`Nanoq` is integrated with pipelines servicing research projects at Queensland Genomics using nanopore sequencing to detect infectious agents in septic patients, reconstruct transmission dynamics of bacterial pathogens, and conduct outbreak sequencing at the Townsville University Hospital (QLD, Australia).

Acknowledgements

We would like to thank the `Rust-Bio` community, the `seq_io` project and the `OneCodex` team for developing the `Rust` read parsing crates. ES was funded by the Queensland Genomics Far North Queensland project and a joint grant by HOT NORTH and the Center for Policy Relevant Infectious Disease Simulation and Mathematical Modelling (NHMRC: #1131932).

References

De Coster, W., D'Hert, S., Schultz, D. T., Cruts, M., & Van Broeckhoven, C. (2018). NanoPack: visualizing and processing long-read sequencing data. *Bioinformatics*, *34*(15), 2666–2669. https://doi.org/10.1093/bioinformatics/bty149

Jain, M., Koren, S., Miga, K. H., Quick, J., Rand, A. C., Sasani, T. A., Tyson, J. R., Beggs, A. D., Dilthey, A. T., Fiddes, I. T., Malla, S., Marriott, H., Nieto, T., O'Grady, J., Olsen, H. E., Pedersen, B. S., Rhie, A., Richardson, H., Quinlan, A. R., … Loose, M. (2018). Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nat. Biotechnol.*, *36*(4), 338–345. https://doi.org/10.1038/nbt.4060

Köster, J. (2015). Rust-Bio: a fast and safe bioinformatics library. *Bioinformatics*, *32*(3), 444–446. https://doi.org/10.1093/bioinformatics/btv573

Lanfear, R., Schalamun, M., Kainer, D., Wang, W., & Schwessinger, B. (2018). MinIONQC: fast and simple quality control for MinION sequencing data. *Bioinformatics*, *35*(3), 523–525. https://doi.org/10.1093/bioinformatics/bty654

Meredith, L. W., Hamilton, W. L., Warne, B., Houldcroft, C. J., Hosmillo, M., Jahun, A. S., Curran, M. D., Parmar, S., Caller, L. G., Caddy, S. L., Khokhar, F. A., Yakovleva, A., Hall, G., Feltwell, T., Forrest, S., Sridhar, S., Weekes, M. P., Baker, S., Brown, N., … Goodfellow, I. (2020). Rapid implementation of SARS-CoV-2 sequencing to investigate cases of health-care associated COVID-19: A prospective genomic surveillance study. *Lancet Infect. Dis.*, *20*(11), 1263–1272. https://doi.org/10.1016/S1473-3099(20)30562-4

Nicholls, S. M., Quick, J. C., Tang, S., & Loman, N. J. (2019). Ultra-deep, long-read nanopore sequencing of mock microbial community standards. *Gigascience*, *8*(5). https://doi.org/10.1093/gigascience/giz043