The Journal of Open Source Software

Nanoq: ultra-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: 24 November 2021

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

Nanopore sequencing is now routinely used in a variety of genomics applications, including whole genome assembly (Jain et al., 2018) and real-time infectious disease surveillance (Meredith et al., 2020). One of the first steps in many workflows is to assess the quality of reads and to obtain summary statistics, as well as to filter fragmented or low quality reads. 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. Benchmarks indicate that `nanoq` is as fast as `seqtk` for small datasets (100,000 reads) and ~1.5x as fast for large datasets (3.5 million reads). Without quality scores, computing summary statistics is around ~2-3x faster than `rust-bio-tools` and `seq-kit stats`, 44x faster than `seqtk` and up to ~450x faster than `NanoStats` (> 1.2 million reads per second). In read filtering applications, `nanoq` is considerably faster than other commonly used tools (`NanoFilt`, `Filtlong`). Memory consumption is consistent and tends to be lower than other applications (~5-10x). `Nanoq` offers nanopore-specific quality scores, read filtering options and output compression. It can be applied to data from the public domain, as part of automated pipelines, in streaming applications, or to rapidly check progress of active sequencing runs.

Statement of need

`NanoPack` (`biopython`) (De Coster et al., 2018), `Filtlong` (`Klib`) and `MinIONQC` (basecalling summary) (Lanfear et al., 2018) are common tools used to filter and obtain summary statistics from nanopore reads. However, their performance may be bottlenecked at read iteration (`NanoPack`, `Filtlong`), they may not immediately applicable due to reliance on basecalling summary files (`MinIONQC`) or implement more complex filters and data visualization for research applications (`NanoFilt`, `Filtlong`). We wrote `nanoq` to accelerate quality control and summary statistics for large nanopore data sets.

Benchmarks

Benchmarks evaluate processing speed and memory consumption of a basic read length filter and summary statistics on the even Zymo mock community (Nicholls et al., 2019) (`GridION`) with comparisons to `seqtk fqchk`, `seqkit stats` (Shen, 2016), `rust-bio-tools`(Köster, 2015), `NanoFilt`, `NanoStat` (De Coster et al., 2018) and `Filtlong`. Time to completion and maximum memory consumption were measured using `/usr/bin/time -f "%e %M"`, speedup is relative to the slowest command in the set. We note that summary statistics from `rust-bio-tools` and `seqkit stats` do not compute read quality scores and are therefore comparable to `nanoq-fast`.

Tasks:

- `stats`: basic read set summaries
- `filter`: minimum read length filter (into `/dev/null`)

Tools:

- `rust-bio-tools=0.28.0`
- `nanostat=1.5.0`
- `nanofilt=2.8.0`
- `filtlong=0.2.1`
- `seqtk=1.3-r126`
- `nanoq=0.8.2`
- `seqkit=2.0.0`

Commands used for `stats` task:

- `nanostat` (fq + fq.gz) –> `NanoStat --fastq test.fq --threads 1`
- `nanostat-t8` (fq + fq.gz) –> `NanoStat --fastq test.fq --threads 8`
- `seqtk-fqchk` (fq + fq.gz) –> `seqtk fqchk`
- `seqkit-stats` (fq + fq.gz) –> `seqkit stats -j1`
- `rust-bio` (fq) –> `rbt sequence-stats --fastq < test.fq`
- `rust-bio` (fq.gz) –> `zcat test.fq.gz | rbt sequence-stats --fastq`
- `nanoq` (fq + fq.gz) –> `nanoq --input test.fq --stats`
- `nanoq-fast` (fq + fq.gz) –> `nanoq --input test.fq --stats --fast`

Commands used for `filter` task:

- `filtlong` (fq + fq.gz) –> `filtlong --min_length 5000 test.fq > /dev/null`

- `nanofilt` (fq) –> `NanoFilt --fastq test.fq --length 5000 > /dev/null`
- `nanofilt` (fq.gz) –> `gunzip -c test.fq.gz | NanoFilt --length 5000 > /dev/null`
- `nanoq` (fq + fq.gz) –> `nanoq --input test.fq --min-len 5000 > /dev/null`
- `nanoq-fast` (fq + fq.gz) –> `nanoq --input test.fq --min-len 5000 --fast > /dev/null`

Files:

- `zymo.fq`: uncompressed (100,000 reads, ~400 Mbp)
- `zymo.fq.gz`: compressed (100,000 reads, ~400 Mbp)
- `zymo.full.fq`: uncompressed (3,491,078 reads, ~14 Gbp)

Data preparation:

```
wget "https://nanopore.s3.climb.ac.uk/Zymo-GridION-EVEN-BB-SN.fq.gz"
zcat Zymo-GridION-EVEN-BB-SN.fq.gz > zymo.full.fq
head -400000 zymo.full.fq > zymo.fq && gzip -k zymo.fq
```

Elapsed real time and maximum resident set size:

```
/usr/bin/time -f "%e %M"
```

Task and command execution:

Commands were run in replicates of 10 with a mounted benchmark data volume in the provided `Docker` container. An additional cold start iteration for each command was not considered in the final benchmarks.

```
for i in {1..11}; do
  for f in /data/*.fq; do
    /usr/bin/time -f "%e %M" nanoq -f- s -i $f 2> benchmark
    tail -1 benchmark >> nanoq_stat_fq
  done
done
```

Benchmark results

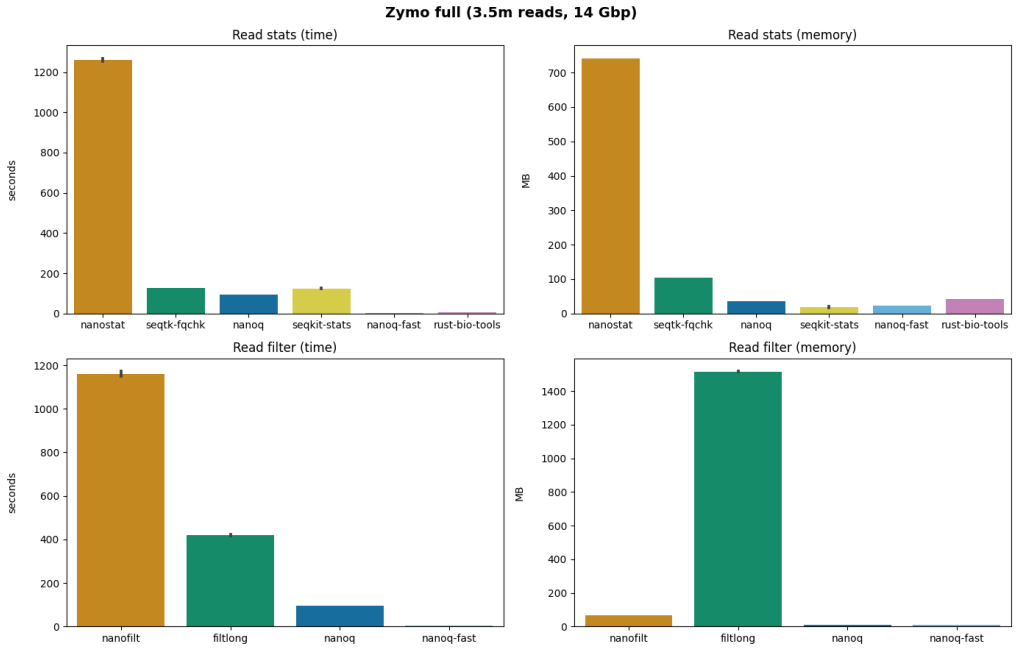

Figure 1: Nanoq benchmarks on 3.5 million reads of the Zymo mock community (10 replicates)

`stats + zymo.full.fq`

command	mem (sd)	sec (sd)	reads / sec	speedup	quality scores
nanostat	741.4 (0.09)	1260. (13.9)	2,770	01.00 x	true
seqtk-fqchk	103.8 (0.04)	125.9 (0.15)	27,729	10.01 x	true
seqkit-stats	**18.68** (3.15)	125.3 (0.91)	27,861	10.05 x	false
nanoq	35.83 (0.06)	94.51 (0.43)	36,938	13.34 x	true
rust-bio	43.20 (0.08)	06.54 (0.05)	533,803	192.7 x	false
nanoq-fast	22.18 (0.07)	**02.85** (0.02)	1,224,939	442.1 x	false

`filter + zymo.full.fq`

command	mem (sd)	sec (sd)	reads / sec	speedup
nanofilt	67.47 (0.13)	1160. (20.2)	3,009	01.00 x
filtlong	1516. (5.98)	420.6 (4.53)	8,360	02.78 x
nanoq	11.93 (0.06)	94.93 (0.45)	36,775	12.22 x
nanoq-fast	**08.05** (0.05)	**03.90** (0.30)	895,148	297.5 x

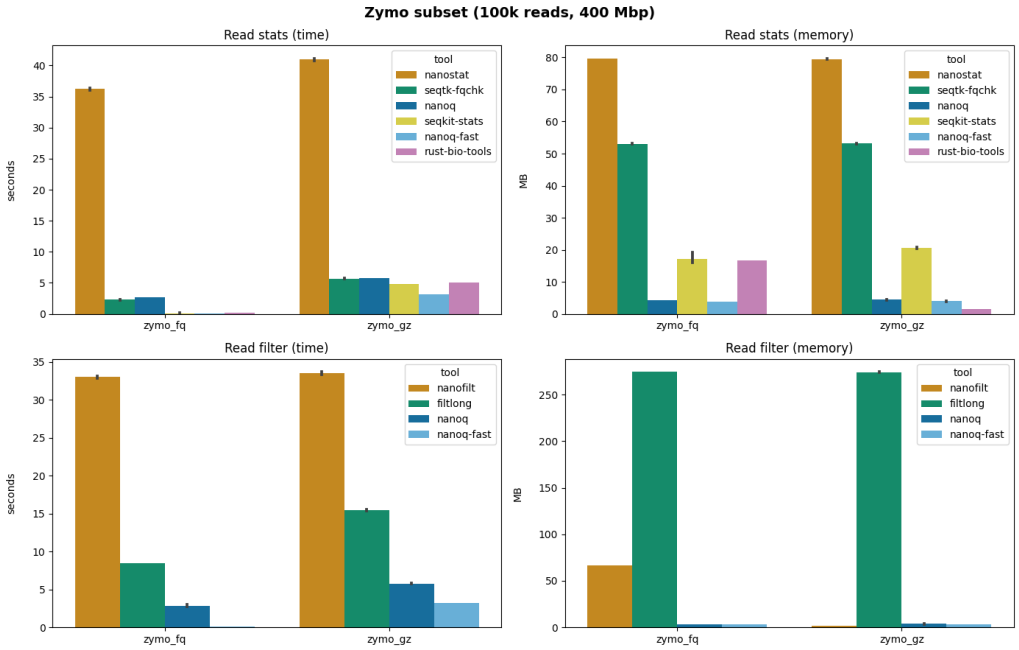

Figure 2: Nanoq benchmarks on 100,000 reads of the Zymo mock community (10 replicates)

`stats + zymo.fq`

command	mem (sd)	sec (sd)	reads / sec	speedup	quality scores
nanostat	79.64 (0.14)	36.22 (0.27)	2,760	01.00 x	true
nanoq	04.26 (0.09)	02.69 (0.02)	37,147	13.46 x	true
seqtk-fqchk	53.01 (0.05)	02.28 (0.06)	43,859	15.89 x	true
seqkit-stats	17.07 (3.03)	00.13 (0.00)	100,000	36.23 x	false
rust-bio	16.61 (0.08)	00.22 (0.00)	100,000	36.23 x	false
nanoq-fast	**03.81** (0.05)	**00.08** (0.00)	100,000	36.23 x	false

`stats + zymo.fq.gz`

command	mem (sd)	sec (sd)	reads / sec	speedup	quality scores
nanostat	79.46 (0.22)	40.98 (0.31)	2,440	01.00 x	true
nanoq	04.44 (0.09)	05.74 (0.04)	17,421	07.14 x	true
seqtk-fqchk	53.11 (0.05)	05.70 (0.08)	17,543	07.18 x	true
rust-bio	**01.59** (0.06)	05.06 (0.04)	19,762	08.09 x	false
seqkit-stats	20.54 (0.41)	04.85 (0.02)	20,619	08.45 x	false
nanoq-fast	03.95 (0.07)	**03.15** (0.02)	31,746	13.01 x	false

`filter + zymo.fq`

command	mem (sd)	sec (sd)	reads / sec	speedup
nanofilt	66.29 (0.15)	33.01 (0.24)	3,029	01.00 ×
filtlong	274.5 (0.04)	08.49 (0.01)	11,778	03.89 ×
nanoq	03.61 (0.04)	02.81 (0.28)	35,587	11.75 ×
nanoq-fast	**03.26** (0.06)	**00.12** (0.01)	100,000	33.01 ×

`filter + zymo.fq.gz`

command	mem (sd)	sec (sd)	reads / sec	speedup
nanofilt	**01.57** (0.07)	33.48 (0.35)	2,986	01.00 ×
filtlong	274.2 (0.04)	16.45 (0.09)	6,079	02.04 ×
nanoq	03.68 (0.06)	05.77 (0.04)	17,331	05.80 ×
nanoq-fast	03.45 (0.07)	**03.20** (0.02)	31,250	10.47 ×

Availability

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

- Cargo: `cargo install nanoq`
- Conda: `conda install -c bioconda 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.

Acknowledgements

We would like to thank the `OneCodex` team for developing `needletail`, Luiz Irber and Pierre Marijon for developing `niffler` and Michael Hall for code adoption from Rasusa. ES was funded by HOT NORTH and the Center for Policy Relevant Infectious Disease Simulation and Mathematical Modelling (NHMRC: #1131932). LC was funded by a NHMRC grant (NHMRC: GNT1195743).

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

Shen, S. A. L., Wei AND Le. (2016). SeqKit: A cross-platform and ultrafast toolkit for FASTA/q file manipulation. *PLOS ONE*, *11*(10), 1–10. https://doi.org/10.1371/journal.pone.0163962