Check for updates

SOFTWARE TOOL ARTICLE

Picopore: A tool for reducing the storage size of Oxford Nanopore Technologies datasets without loss of functionality

[version 1; peer review: 2 approved]

Scott Gigante

Walter & Eliza Hall Institute of Medical Research, Parkville, Victoria, 3121, Australia

V1 First published: 07 Mar 2017, 6:227
<https://doi.org/10.12688/f1000research.11022.1>
 Second version: 12 Apr 2017, 6:227
<https://doi.org/10.12688/f1000research.11022.2>
 Latest published: 28 Sep 2017, 6:227
<https://doi.org/10.12688/f1000research.11022.3>

Abstract

Oxford Nanopore Technologies' (ONT) MinION and PromethION long-read sequencing technologies are emerging as genuine alternatives to established Next-Generation Sequencing technologies. A combination of the highly redundant file format and a rapid increase in data generation have created a significant problem both for immediate data storage on MinION-capable laptops, and for long-term storage on lab data servers.

We developed Picopore, a software suite offering three methods of compression. Picopore's lossless and deep lossless methods provide a 25% and 44% average reduction in size, respectively, without removing any data from the files. Picopore's raw method provides an 88% average reduction in size, while retaining biologically relevant data for the end-user. All methods have the capacity to run in real-time in parallel to a sequencing run, reducing demand for both immediate and long-term storage space.

Keywords

DNA Sequencing, Genome Informatics, Nanopore Sequencing, Compression, Data Storage

This article is included in the **Nanopore Analysis** gateway.

This article is included in the **Python** collection.

Open Peer Review**Approval Status**

	1	2
version 3		
(revision)
28 Sep 2017		view
version 2		
(revision)
12 Apr 2017		view
version 1
07 Mar 2017	view	view

1. **Matthew Loose**, University of Nottingham, Nottingham, UK
2. **David A. Eccles** , Malaghan Institute of Medical Research, Wellington, New Zealand
Gringene Bioinformatics, Wellington, New Zealand

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

Corresponding author: Scott Gigante (gigante.s@wehi.edu.au)

Competing interests: No competing interests were disclosed. **Author Endorsement:** Chris Woodruff confirms that the author has an appropriate level of expertise to conduct this research, and confirms that the submission is of an acceptable scientific standard. Chris Woodruff declares he has no competing interests. Affiliation: Visiting Scientist at Bioinformatics Division of Walter and Eliza Hall Institute of Medical Research, Parkville, Victoria, Australia.

Grant information: The work discussed in this article was funded by the Speed Lab in the Bioinformatics Division of the Walter & Eliza Institute of Medical Research. The Speed Lab is supported by the Australian NHMRC Program Grant number 1054618. *The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.*

Copyright: © 2017 Gigante S. This is an open access article distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. Data associated with the article are available under the terms of the [Creative Commons Zero "No rights reserved" data waiver](#) (CC0 1.0 Public domain dedication).

How to cite this article: Gigante S. **Picopore: A tool for reducing the storage size of Oxford Nanopore Technologies datasets without loss of functionality [version 1; peer review: 2 approved]** F1000Research 2017, 6:227
<https://doi.org/10.12688/f1000research.11022.1>

First published: 07 Mar 2017, 6:227 <https://doi.org/10.12688/f1000research.11022.1>

Introduction

Oxford Nanopore Technologies' (ONT) nanopore sequencing technology MinION provides a high-throughput, low-cost alternative to traditional Next-Generation Sequencing (NGS) technologies¹. The sequencing device itself is handheld and connects by USB to a laptop computer. Together with all equipment and reagents required for DNA library preparation, the equipment required to use MinION is minimal; entire laboratories have even been transported overseas in a suitcase, allowing a versatile and agile approach towards DNA and RNA sequencing².

Over the course of ONT's Early Access Program, several improvements in software and chemistry have led to a rapid increase in yield, through an increase in average read length, an improvement in basecalling accuracy rates and an increase in total number of reads. In October 2015, the MinION Analysis and Reference Consortium (MARC), using R7.3 flow cells and SQK-MAP005 (2D) chemistry, reported a median of 60,600 reads with a median yield of 650,000 events across 20 MinION experiments³. In contrast, ONT claim to have obtained a total base yield of 17 gigabases using an R9.4 flowcell on the latest version of their MinKNOW software (<https://nanoporetech.com/about-us/news/minion-software-minknow-upgraded-enable-increased-data-yield-other-benefits>). The drastic increase in data generation from a single experiment is rapidly becoming the limiting factor in uptake of the technology.

The concerns over data storage extend beyond the data generation capabilities of a single flowcell. Recent attempts to perform *de novo* assembly of eukaryotic genomes have combined the data generated by multiple flowcells in order to gain sufficient coverage of the genome⁴. To this end, ONT have begun the pre-commercial release of the PromethION, a benchtop nanopore sequencing device with 48 flowcells. In addition, each of these flowcells contains 3000 channels, as opposed to the 512 channels in a single MinION flowcell, with data generation projected at 6 terabases per flowcell per day⁵.

Numerous methods have been developed for the efficient analysis of the increasingly large nanopore datasets. However, current methods to reduce the data storage footprint are extremely limited. Nanopore runs uploaded to online repositories, such as the European Nucleotide Archive, are bundled into a tarball, a process which facilitates upload as a single file, but does not decrease file size. Moreover, ONT runs bundled into a tarball (which could then be compressed using traditional means) are not able to be read by any existing nanopore analysis tools. Moreover, traditional compression technologies are poorly adapted to the needs of the individual user, many of whom have no need for a large portion of the data stored by ONT's basecallers. Therefore, we developed Picopore, a tool for reducing the storage footprint of the ONT runs without preventing users from using their preferred analysis tools. Picopore uses a combination of storage reduction techniques, including built-in dynamic compression in the HDF5 file format, reduction of data duplication, efficient allocation of memory within the file, and the removal of intermediate data generated during basecalling, which is deemed unnecessary by the end-user.

Methods

Implementation

Picopore is developed using the Python h5py module (<http://www.h5py.org/>), an interface to the HDF5 file format (<http://www.hdf-group.org/HDF5>), used by ONT under the FAST5 file extension. Picopore implements a number of different compression methods, a selection of which are applied according to user preferences, before using HDF5's `h5repack` to rebuild the file according to the reduced file size requirements.

Compression techniques

Built-in GZIP compression. The HDF5 file format allows for both files and datasets within files to be written using a number of different compression filters, the most universally implemented being GZIP. GZIP applies traditional compression to the data stored in the HDF5 file with choices of compression level between 1 and 9. ONT's default compression uses GZIP at level 1; Picopore increases this to level 9 in order to decrease disk space usage.

Dynamic memory allocation for variables. Data stored in the HDF5 file format uses fixed-size file formats provided by NumPy, which provides a vast array of options for storing integers, decimals and strings within high-dimensional datasets⁶. ONT's native data is written using the largest data types provided by NumPy: 64-bit integers, 64-bit decimals, and "variable-size" strings. Picopore vastly reduces disk space usage by analyzing each dataset to determine the minimum number of bytes required by a given variable in the file, changing the data type accordingly.

Collapsing of file structure. The advantage of the HDF5 file format is that it provides a file directory-like storage format for datasets and properties, making reading and writing to the files straightforward and easy to understand. However, the inherent nature of the highly-structured file format requires HDF5 to allocate slots of memory to "groups", which represent the internal directory structure of the file. Picopore reduces the disk space used by this file metadata by collapsing the directory structure, while retaining the option for users to reverse this action when tools that only recognize the original file format are required.

Indexing of duplicated data. ONT's most widely used basecalling software, the cloud-based Metrichor service (<https://metrichor.com/s/>), performs feature recognition (or "event detection"). This segments the electrical signal representing each nanopore read into events, each of which represents a period of time when the DNA was stationary in the nanopore. These events are then converted into basecalled data, which provides a single k-mer (at present a 5-mer) of DNA representing the bases in the nanopore contributing to the signal at that time. Each event corresponds to a single row in the basecalled dataset, and both the event detection and basecalled datasets thus store the mean signal, standard deviation, start time and length of the event. Picopore reduces disk space usage by indexing the basecalled dataset to the event detection dataset, removing the duplicated data while retaining the option for users to reverse this action when tools that require access to this data are required.

Removal of intermediate data. The primary function of all basecalling software is to generate a FASTQ file containing the genomic sequence and associated quality scores representing the read stored in each FAST5 file. While some software tools, such as *nanopolish*⁷ and *nanoraw*⁸, do make use of the signal, event detection and basecalled datasets, the large majority of analyses, including alignment, assembly and variant calling, simply require access to the FASTQ data. Picopore allows users to remove the intermediate data generated during the process of converting raw signal to FASTQ, while retaining the signal data, should they ever want to re-basecall the run to attain improved FASTQ data or to access this intermediate data at a later stage.

Operation

Requirements. Picopore is built in Python 2.7 (www.python.org) and runs on Windows, Mac OS and Linux. It requires the following Python packages:

- `h5py 2.2.0` or later
- `watchdog 0.8.3` or later

In addition, Picopore requires HDF5 1.8.4 or newer, with development headers (`libhdf5-dev` or similar), including the binary utility `h5repack`, which is included therein.

Installation. The latest stable version of Picopore is available on PyPi and bioconda (see Software availability). It can be installed according to the following commands:

Linux and Mac OS: `pip install picopore`

Windows: `conda install picopore -c bioconda -c conda-forge`

Picopore can also be installed from source (see Software availability) using the command `python setup.py install`.

Execution. Picopore is run from the command-line as a binary executable.

Picopore accepts both folders and FAST5 files as input. If a folder is provided, it will be searched recursively for FAST5 files, and all files found will be considered as input.

There are three modes of compression available, each of which performs a selection of the techniques described above.

- **lossless:** performs built-in GZIP compression and dynamic memory allocation for variables. This mode is both fast and allows continued analysis of data by any existing software.
- **deep-lossless:** performs lossless compression, as well as collapsing of file structure and indexing of duplicated data. This mode obtains the best compression results without removing any data, but comes at the cost of requiring reversion before most software tools can analyse the data.

- **raw:** performs lossless compression, as well as removal of intermediate data, partially reverting files to the “raw” pre-basecalled file format. This mode is fast, obtains the best filesize reduction, and allows continued analysis by tools that extract FASTQ and related data, but comes at the cost of removing intermediate basecalling data required for some niche applications, such as *nanopolish*, which cannot be retrieved by Picopore (but can be regenerated using basecalling software.)

Optional arguments include:

- `-revert:` reverts lossless compressed files to their original state to allow high-speed access at the cost of disk usage;
- `-realtime:` watches for file creation in the given input folder(s) and performs the selected mode of compression on new files in real time to reduce the footprint of an ongoing MinION run;
- `-prefix:` allows the user to specify a filename prefix to prevent in-place overwriting of files;
- `-group:` allows the user to select only one of the analysis groups on files that have been processed by multiple basecallers;
- `-threads:` allows the user to specify the number of files to be processed in parallel.

Results

To demonstrate the effectiveness of Picopore’s compression, we ran all three modes of compression on four toy datasets of 40 FAST5 files run using the R9 SQK-RAD001 (R9_1D), R9 SQK-NSK007 (R9_2D), R9.4 SQK-RAD002 (R9.4_1D) and R9.4 SQK-LSK208 (R9.4_2D) protocols. The files for the toy datasets were chosen randomly from the *pass* folder of four MinION datasets generated at the Australian Genome Research Facility (see Software and data availability). For the R9_1D dataset, DNA was extracted from the lung of a juvenile 129/Sv mouse using the DNeasy Blood and Tissue kit (Qiagen). For the R9_2D, R9.4_1D and R9.4_2D datasets, DNA was extracted from a culture of *escherichia coli* K12 MG1655 using the Blood & Cell Culture DNA Kit (Qiagen). Quality control performed by visualisation on the TapeStation (Agilent). Run metadata is shown in [Table 1](#).

Each file was compressed and tarred using each of five methods: no compression, `gzip` (applied after tarring, as per convention), `picopore lossless`, `picopore deep-lossless` and `picopore raw`. Each of these methods was run on a single core. [Figure 1](#) shows that `lossless` achieves only slightly less compression than `gzip`, giving an average reduction in size of 25% compared to `gzip`’s 32%, while `deep-lossless` and `raw` perform significantly better, giving average reductions in size of 44% and 88%, respectively. A dependent sample t-test (R v3.3.0) was run on individual compressed file sizes. [Table 2](#) shows that each successive method of compression (excluding `gzip`, which does not compress individual files) gives a significant reduction in size from the previous. [Figure 2](#) shows that while all of Picopore’s

Table 1. Metadata for MinION datasets sampled to produce toy datasets.

Name	Chemistry	Protocol	MinION ID	Flowcell ID	Sample	Strain
R9_1D	R9 1D	SQK-RAD001	MN17324	FAD24340	Mouse	129/Sv
R9_2D	R9 2D	SQK-NSK007	MN17324	FAD24193	<i>Escherichia coli</i>	K12 MG1655
R9.4_1D	R9.4 1D Rapid	SQK-RAD002	MN17324	FAF04136	<i>Escherichia coli</i>	K12 MG1655
R9.4_2D	R9.4 2D	SQK-RAD002	MN17324	FAF04232	<i>Escherichia coli</i>	K12 MG1655

Figure 1. Size of tarball containing FAST5 files compressed using various methods.

Table 2. Significance of difference in size of files compressed with different methods using a dependent sample t-test (R v3.3.0).

Mode 1	Mode 2	t-statistic	p-value
uncompressed	lossless	33.58	< 10 ⁻¹⁵
lossless	deep-lossless	20.14	< 10 ⁻¹⁵
deep-lossless	raw	17.69	< 10 ⁻¹⁵

Figure 2. Time taken for single-thread compression and tarring of FAST5 files using various methods.

compression methods are much slower than `gzip`, `raw` is the fastest of these, followed by `lossless` and `deep-lossless`. Note that the tarring time makes up a maximum of 0.04 s/read in each case and is largely negligible.

To demonstrate the effectiveness of Picopore's multithreading, we ran `deep-lossless`, the most computationally expensive of the Picopore compression methods, on each dataset using 1, 2, 5, and 10 cores. **Figure 3** shows an almost linear improvement in speed, showing that even on a small dataset, the multithreading overhead is relatively small.

Dataset 1. Output data generated using Picopore on four toy datasets

<http://dx.doi.org/10.5256/f1000research.11022.d153370>

Each of the four toy datasets was compressed using each of Picopore's three compression modes and GZIP. Datasets compressed using Picopore were tarred after compression, while the GZIP dataset was tarred before compression.

Discussion

It is clear that, due to the enormous reduction in disk space and low total time requirements, Picopore's `raw` compression is the optimal compression mode for users who have no need for the intermediate

Figure 3. Speed of deep lossless compression of FAST5 files using multiple threads. The dotted blue line shows the theoretical linear maximum increase in speed for the R9 2D run.

event detection and basecalling data. The superior running speed of lossless compression over deep lossless compression may mean that this is the preferred method for users who wish to retain all data and need to compress the data in real-time; however, for users with these data retention requirements who wish to store data long-term on their file server, for whom speed of compression is not an issue, deep lossless compression provides the best option. All Picopore compression methods provide significant improvements over uncompressed or traditionally compressed files, and `lossless` and `raw` methods carry the added benefit that files can be processed using analysis tools in compressed form.

Although the compression has a high CPU cost, the capability of Picopore to run on multiple threads signifies that, if computing resources are available, the files can be compressed in a reasonably short period of time. Finally, extrapolating the running time per file to a real-time run of 500,000 reads over 48 hours (0.34 s/read), Picopore has the capability to run `lossless` (0.33 s/read) and `raw` (0.25 s/read) modes on a single core in real time. While `deep-lossless` requires multiple cores to keep pace with the MinION data generation, this adds little to the overall computational cost, and can reach real-time speed with just five cores (0.24 s/read). As data generation continues to increase in scale, further gains could be made by incorporating the compression methods used in Picopore into the basecalling software itself.

Conclusions

ONT' MinION and PromethION sequencing devices promise to produce increasingly large datasets as the technology progresses toward commercial release. The disk space required to run and store one or multiple datasets from these poses a problem for service providers and users alike; Picopore provides three different solutions that cater to the different needs of users.

Although the trade-off between data retention, computing time and disk space is a compromise that cannot be perfectly resolved, Picopore provides user options to reduce their ONT datasets to the minimum viable size based on their intended use. This may involve real-time compression for laptop disk space concerns, reduction of bandwidth usage for transfer of datasets between laboratories, or reduction of the storage footprint on shared data servers.

Software and data availability

Software for Linux or Mac OS available from: <https://pypi.python.org/pypi/picopore>

Software for Linux, Mac OS and Windows available from: <https://anaconda.org/bioconda/picopore>

Source code available from: <https://github.com/scottgigante/picopore>

Archived source code from: <https://doi.org/10.5281/zenodo.3219579>

License: GPLv3

Dataset 1. Output data generated using Picopore on four toy datasets. Each of the four toy datasets was compressed using each of Picopore's three compression modes and GZIP. Datasets compressed using Picopore were tarred after compression, while the GZIP dataset was tarred before compression. doi, [10.5256/f1000research.11022.d153370](https://doi.org/10.5256/f1000research.11022.d153370)¹⁰

The toy dataset used for the analyses in this paper is available on Zenodo: Toy datasets for compression by Picopore, doi: [10.5281/zenodo.3219591](https://doi.org/10.5281/zenodo.3219591)¹¹.

Author endorsement

Chris Woodruff confirms that the author has an appropriate level of expertise to conduct this research, and confirms that the submission is of an acceptable scientific standard. Chris Woodruff declares he has no competing interests. Affiliation: Visiting Scientist at Bioinformatics Division of Walter and Eliza Hall Institute of Medical Research, Parkville, Victoria, Australia.

Author contributions

SG designed and developed the source code, published the software and wrote the article.

Competing interests

No competing interests were disclosed.

Grant information

The work discussed in this article was funded by the Speed Lab in the Bioinformatics Division of the Walter & Eliza Institute of Medical Research. The Speed Lab is supported by the Australian NHMRC Program (grant number 1054618).

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgments

The author is grateful to Terry Speed and Chris Woodruff from the Walter & Eliza Hall Institute of Medical Research for their supervision of this and related work, to Alexis Lucattini and Lavinia Gordon from the Australian Genome Research Facility and Matthew Ritchie, Andrew Kinery and Marnie Blewitt from the Walter & Eliza Hall Institute of Medical Research for their assistance in providing MinION data and discussing ongoing analyses, to Bio-medical Research Victoria and CSL Limited for the provision of a stipend to begin his work at the Walter & Eliza Hall Institute, and to the organisers and participants of PorecampAU for inspiring this work.

References

1. Eisenstein M: **Oxford Nanopore announcement sets sequencing sector abuzz.** *Nat Biotechnol.* 2012; **30**(4): 295–296.
[PubMed Abstract](#) | [Publisher Full Text](#)
2. Quick J, Loman NJ, Duraffour S, *et al.*: **Real-time, portable genome sequencing for Ebola surveillance.** *Nature.* 2016; **530**(7589): 228–232.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
3. Ip CL, Loose M, Tyson JR, *et al.*: **MinION Analysis and Reference Consortium: Phase 1 data release and analysis [version 1; referees: 2 approved].** *F1000Res.* 2015; **4**: 1075.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
4. Tyson JR, O'Neil NJ, Jain M, *et al.*: **Whole genome sequencing and assembly of a *Caenorhabditis elegans* genome with complex genomic rearrangements using the MinION sequencing device.** *bioRxiv.* 2017.
[Publisher Full Text](#)
5. Jain M, Olsen HE, Paten B, *et al.*: **The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community.** *Genome Biol.* 2016; **17**(1): 239.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
6. van der Walt S, Colbert SC, Varoquaux G: **The numpy array: a structure for efficient numerical computation.** *Comput Sci Eng.* 2011; **13**(2): 22–30.
[Publisher Full Text](#)
7. Loman NJ, Quick J, Simpson JT: **A complete bacterial genome assembled *de novo* using only nanopore sequencing data.** *Nat Methods.* 2015; **12**(8): 733–735.
[PubMed Abstract](#) | [Publisher Full Text](#)
8. Stoiber MH, Quick J, Egan R, *et al.*: ***De novo* identification of DNA modifications enabled by genome-guided nanopore signal processing.** *bioRxiv.* 2016.
[Publisher Full Text](#)
9. Gigante S, Sjödin A: **scottgigante/picopore: 1.0.0 Stable [Data set].** *Zenodo.* 2017.
[Data Source](#)
10. Gigante S: **Dataset 1 in: Picopore: A tool for reducing the storage size of Oxford Nanopore Technologies datasets without loss of functionality.** *F1000Research.* 2017.
[Data Source](#)
11. Lucattini A, Kinery A: **Toy datasets for compression by Picopore [Data set].** *Zenodo.* 2017.
[Data Source](#)

Open Peer Review

Current Peer Review Status:

Version 1

Reviewer Report 21 March 2017

<https://doi.org/10.5256/f1000research.11886.r20757>

© 2017 Eccles D. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

David A. Eccles

¹ Malaghan Institute of Medical Research, Wellington, New Zealand

² Gringene Bioinformatics, Wellington, New Zealand

It's quite difficult to knock out a useful software tool for Oxford Nanopore devices before they change the protocol, or release their own tool that does similar things. Scott Gigante has done an admirable job in this regard by developing and publishing a much-needed tool within the space of time between two updates of the MinKNOW software (v1.4 -> v1.5). While the newest version of MinKNOW no longer produces event data by default in the FAST5 files (substantially reducing file sizes), Scott's tool will still be useful for existing file sets and event-called files in the future.

Manuscript

The manuscript is sufficiently verbose for a short software release publication: explaining modes of operation, demonstrating differences in file compression in different modes, and showing processing speed on different datasets.

Title / abstract

The title and abstract sufficiently summarise the manuscript

Introduction

- "ONT" -> "ONT's"
- "The drastic increase... the limiting factor in uptake of the technology" -- cite, alter, or remove. My experience is that storage space is only an issue for *existing* users of the technology, not people deciding whether or not to use the MinION.

Operation

- At least one running example in the 'Execution' section would be useful, similar to the Usage section of the pypi repository. Something like this:

'picopore --mode lossless --prefix shrunk '

Discussion

- The introduction suggests ONT's internal runs are approaching 2M reads, yet the discussion suggests 0.5M reads per run. This is the difference between real-time processing and almost a week of waiting for processing to finish.
- The discussion mentions future potential capabilities for basecalling software, but not Picopore itself. Are there any planned updates on the horizon?

Software testing

The current version of Picopore (installed by *pip* on 2017-Mar-17) had a few minor issues on my system that would prevent most users from being able to run the software. Once these issues were dealt with, Picopore was able to substantially reduce the file size of two FAST5 files with wildly different internal structures, while retaining important base call and raw signal information.

Installation

The program appeared to install fine on my Debian Linux desktop by running '*pip install picopore*', as per the manuscript instructions. Unfortunately there was a problem with module import when showing the help dialog:

```
$ picopore -h
Traceback (most recent call last):
  File "/usr/local/bin/picopore", line 7, in
    from picopore.__main__ import main
  File "/usr/local/lib/python2.7/dist-packages/picopore/__main__.py", line 22, in
    from picopore.parse_args import parseArgs, checkSure
  File "/usr/local/lib/python2.7/dist-packages/picopore/parse_args.py", line 20, in
    from builtins import input
ImportError: No module named builtins
```

Commenting out the offending line in 'parse_args.py' fixed this error.

Use of other data

Instead of using the provided data, I did a stress test of sorts on Picopore by running it on two files which were put in subdirectories of a parent directory:

1. A 2kb R7.3 tomato read produced by me in March 2016 (channel 342, read 13) [[David Eccles' read](#)]
2. A 771kb R9.4 E. coli read produced by Nick Loman and Josh Quick in March 2017 [[Nick Loman & Josh Quick's read](#)]

Equivalence testing

It is appreciated that Picopore includes a test for equivalence to make sure information is retained. Picopore was able to recursively descend through the directories, but the deep-lossless equivalence test reported failure for both of these sequences. In the case of the second file, it appears that the only failure was a missing //Picopore directory (which should probably be excluded from the failure modes):

- Equivalence test 1

```
$ picopore --prefix pico_ -t 10 --test --mode deep-lossless tested_picopore
Performing deep lossless compression on 2 files...
No conversion path for dtype: dtype('    Complete.
Original size: 67048849
Compressed size: 66616200
Checking equivalence of
/home/gringer/bioinf/reviews/tested_picopore/1/lambda_TEDxWellington_DavidEccles_3637_1_ch342_read13_s
(file 1) and
/home/gringer/bioinf/reviews/tested_picopore/1/picopore.test.lambda_TEDxWellington_DavidEccles_3637_1_ch
(file 2)...
Failure: //Analyses missing from file 2
Failure: //Raw missing from file 2
Failure: //Sequences missing from file 2
Failure: //UniqueGlobalKey missing from file 2
Failure: //Picopore missing from file 1
Traceback (most recent call last):
  File "/usr/local/bin/picopore", line 11, in
    sys.exit(main())
  File "/usr/local/lib/python2.7/dist-packages/picopore/__main__.py", line 80, in main
    runTest(args)
  File "/usr/local/lib/python2.7/dist-packages/picopore/__main__.py", line 63, in runTest
    checkEquivalent(f, compressedFile)
  File "/usr/local/lib/python2.7/dist-packages/picopore/test.py", line 72, in checkEquivalent
    recursiveCheckEquivalent(file1, file2, group.name)
  File "/usr/local/lib/python2.7/dist-packages/picopore/test.py", line 56, in
recursiveCheckEquivalent
    recursiveCheckEquivalent(file1, file2, ".".join([name, key]))
  File "/usr/local/lib/python2.7/dist-packages/picopore/test.py", line 50, in
recursiveCheckEquivalent
    if not attr2[key] == value:
ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or
a.all()
```

- Equivalence test 2

```
$ picopore --prefix pico_ -t 10 --test --mode deep-lossless tested_picopore/2
Performing deep lossless compression on 1 files...
No conversion path for dtype: dtype('    Complete.
Original size: 65686831
Compressed size: 65688039
Checking equivalence of
/home/gringer/bioinf/reviews/tested_picopore/2/loman_771kb_ch181_read4882_strand.fast5 (file
1) and
/home/gringer/bioinf/reviews/tested_picopore/2/picopore.test.loman_771kb_ch181_read4882_strand.fast5
(file 2)...
Failure: //Picopore missing from file 1
Complete.
```

Confirmation dialog

The confirmation of writing files is also a great idea, but produces an error when both responses are given. I wonder if this is due to python version incompatibilities (and my attempted prior bugfix):

- Confirmation test 1

```
$ picopore --prefix pico_ -t 10 --mode raw tested_picopore
Performing raw compression with FASTQ and no summary on 2 files...
Are you sure? (yes | no): no
Traceback (most recent call last):
  File "/usr/local/bin/picopore", line 11, in
    sys.exit(main())
  File "/usr/local/lib/python2.7/dist-packages/picopore/__main__.py", line 84, in main
    run(args.revert, args.mode, args.input, args.y, args.threads, args.group, args.prefix,
args.fastq, args.summary)
  File "/usr/local/lib/python2.7/dist-packages/picopore/__main__.py", line 34, in run
    if y or checkSure():
  File "/usr/local/lib/python2.7/dist-packages/picopore/parse_args.py", line 109, in checkSure
    response = input("Are you sure? (yes | no): ")
  File "", line 1, in
NameError: name 'no' is not defined
```

- Confirmation test 2

```
$ picopore --prefix pico_ -t 10 --mode raw tested_picopore
Performing raw compression with FASTQ and no summary on 2 files...
Are you sure? (yes | no): yes
Traceback (most recent call last):
  File "/usr/local/bin/picopore", line 11, in
    sys.exit(main())
  File "/usr/local/lib/python2.7/dist-packages/picopore/__main__.py", line 84, in main
    run(args.revert, args.mode, args.input, args.y, args.threads, args.group, args.prefix,
args.fastq, args.summary)
  File "/usr/local/lib/python2.7/dist-packages/picopore/__main__.py", line 34, in run
    if y or checkSure():
  File "/usr/local/lib/python2.7/dist-packages/picopore/parse_args.py", line 109, in checkSure
    response = input("Are you sure? (yes | no): ")
  File "", line 1, in
NameError: name 'yes' is not defined
```

The 'input' function in my version of python does an evaluation after reading input; replacing 'input(...)' with 'raw_input(...)' fixed this error.

After these errors were fixed enough to allow the code to proceed, Picopore was able to successfully strip out event data (in 'raw' mode) from both the R7.3 and R9.4 FAST5 files, while retaining called FASTQ sequences and raw signal (i.e. it did what it said on the box). File sizes were

reduced from 1.3MB down to 577kB for the R7.3 file, and 63Mb down to 9.8MB for the R9.4 file.

Picopore also retained the 'Model' section from the R7.3 FAST5 files, indicating that it probably does a blacklist removal of known analysis components and retains unknown things in the file hierarchy; this should ensure that Picopore will be reasonably forward-compatible for future file format changes even without updates.

Threaded mode

I tried to run Picopore on the toy dataset in single-threaded mode, broke out of it because it was taking too long, then restarted in threaded mode and realised I wanted to stop *that* as well [Picopore leaves temporary files of a predictable name in the working directories that are not deleted on failure, and I had not deleted them between runs]. Unfortunately, when running Picopore in threaded mode, I was not able to break out of the running program (and needed to kill it using another console).

Equivalence check on included dataset

Running the equivalence check in *lossless* mode on the provided toy dataset produced no errors. No additional testing was done on the toy dataset.

Competing Interests: David Eccles was a demonstrator / speaker at PoreCampAU 2017, an event which inspired the creation of Picopore.

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 10 Apr 2017

Scott Gigante, Walter & Eliza Hall Institute of Medical Research, Parkville, Australia

Dear Dr. Eccles,

Thank you for your review.

I have amended the manuscript according to your suggested revisions.

You note that an equivalent to Picopore's *raw* compression is now the default behaviour for MinKNOW v1.5. I will continue to examine the output from the latest version of MinKNOW to find new mechanisms for size reduction; however, in the short term, I anticipate that Picopore's major use cases will be to reduce the size of older datasets, and to reduce the size of datasets produced by power users who continue to use MinKNOW with event data enabled.

Thank you also for your extensive testing of Picopore. The errors you pointed out have been resolved in the latest version of Picopore, available on Pypi, Bioconda and Github.

Thank you once again for your comments.

Kind regards,

Scott Gigante

Competing Interests: No competing interests were disclosed.

Reviewer Report 20 March 2017

<https://doi.org/10.5256/f1000research.11886.r20761>

© 2017 Loose M. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Matthew Loose

School of Life Sciences, University of Nottingham, Nottingham, UK

Picopore is a well written package that installs quickly and easily, has clear guidance on its use and addresses a relevant issue in Nanopore sequencing at this time.

The tools function as described (certainly on OSX).

I have some reservations about reporting speed in terms of reads/s. I would like to see metrics which take in to account the number of bases being processed per unit time as I suspect the alternative compression options will perform differently by this metric.

I also have some reservations about the use of some modes of picopore compression. Users will need to think carefully about the application of modes which are not immediately compatible with existing tool chains. Given recent announcements from Nanopore with respect to provision of off line base calling I suspect a better long term storage of data will be as simple raw files with an associated fastq.gz.

Competing Interests: No competing interests were disclosed.

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 10 Apr 2017

Scott Gigante, Walter & Eliza Hall Institute of Medical Research, Parkville, Australia

Dear Dr. Loose,

Thank you for your review.

I would like to note that since the compression modes were tested all on the same dataset of a total of 160 files across four runs, the number of bases is fixed, with a total of 1.7 Mb in each instance. Thus, the comparison of speeds and sizes between compression modes will be equivalent by either metric. However, I agree that bases/s is a more useful metric than reads/s in comparing results between different protocols, and have included this in my analyses.

In regards to your recommended mode of long term storage, this is indeed my expectation and is Picopore's recommended mode for end-users. Picopore's *raw* mode stores by default only the raw signal and the FASTQ, which is compressed using HDF5's built-in GZIP compression. Picopore's *lossless* and *deep-lossless* modes are aimed at developers who wish to retain the event data for active use or long-term storage respectively; I acknowledge that these use cases are only suitable for a particular subgroup of users.

Thank you once again for your comments.

Kind regards,

Scott Gigante

Competing Interests: No competing interests were disclosed.

Comments on this article

Version 1

Author Response 20 Mar 2017

Scott Gigante, Walter & Eliza Hall Institute of Medical Research, Parkville, Australia

Hi Wouter,

Thanks very much for your comments.

My apologies for the error in the PromethION throughput, I had misunderstood the statement in the original reference. I will issue a new version of the paper correcting this.

In regards to kmer length, ONT did indeed move to a 6-mer for the later HMM basecallers, but moved back to a 5-mer for the RNN - there is a brief introduction of this [here](#) on the Nanopore Community.

Cheers,
Scott Gigante

Competing Interests: No competing interests were disclosed.

Reader Comment 16 Mar 2017

Wouter De Coster, VIB Center for Molecular Neurology Antwerp, Belgium

Hi Scott,

Interesting paper and great work.

About PromethION you wrote "...data generation projected at 6 terabases per flowcell per day.", but I believe that's the theoretical throughput for the entire machine, all 48 flowcells together rather than per flowcell.

I'm also not sure if the software still uses DNA 5-mers for converting signal to basecalling. If I'm not mistaken that was the case in the early HMM basecaller, which then changed to 6-mers and now an RNN basecaller.

Cheers,
Wouter De Coster

Competing Interests: No competing interests were disclosed.

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com

F1000Research