NanoR: a user-friendly R package to analyze and compare nanopore sequencing data

Davide Bolognini[1*], Niccolò Bartalucci[1‡], Alessandra Mingrino[1‡], Alessandro Maria Vannucchi[1], Alberto Magi[1],

1 Department of Experimental and Clinical Medicine, University of Florence, Largo Brambilla 3, Florence, Italy

‡These authors contributed equally to this work.
* davidebolognini7@gmail.com

Abstract

MinION and GridION X5 from Oxford Nanopore Technologies are devices for real-time DNA and RNA sequencing. On the one hand, MinION is the only real-time, low cost and portable sequencing device and, thanks to its unique properties, is becoming more and more popular among biologists; on the other, GridION X5, mainly for its costs, is less widespread but highly suitable for researchers with large sequencing projects. Despite the fact that Oxford Nanopore Technologies' devices have been increasingly used in the last few years, there is a lack of high-performing and user-friendly tools to handle the data outputted by both MinION and GridION X5 platforms. Here we present NanoR, a cross-platform R package designed with the purpose to simplify and improve nanopore data visualization. Indeed, NanoR is built on few functions but overcomes the capabilities of existing tools to extract meaningful informations from MinION sequencing data; in addition, as exclusive features, NanoR can deal with GridION X5 sequencing outputs and allows comparison of both MinION and GridION X5 sequencing data in one command. NanoR is released as free package for R at `https://github.com/davidebolo1993/NanoR`.

Introduction

During the last decade, second-generation sequencing technologies have definitely improved our ability to detect genetic variants in any organism. However, the short reads (100-500 base pairs) generated by these platforms do not allow to detect complex structural variants such as copy number variations [1]. In the last few years, third-generation sequencing (TGS) technologies have risen to prominence; Oxford Nanopore Technologies (ONT) and Pacific Biosciences [2] allow single-molecule and long-read sequencing with different strategies: while Pacific Biosciences uses a single-molecule real-time approach, ONT's devices (MinION, GridION X5 and PromethION) analyze the transit of each DNA molecule through a nanoscopic pore and measure its effect on an electric current.

Among ONT's devices, MinION and GridION X5 are noteworthies.

MinION is the only low-cost (commercially available by paying a starter-pack fee of $1,000), portable ($\sim$ 10 cm in length and weighs under 100 g) and real-time DNA and RNA sequencer: each MinION Flow Cell, which contains the proprietary sensor array Application-Specific Integrated Circuit (ASIC) and 2048 R9.4 (for sequencing kits 1D)

or R9.5 (for sequencing kits $1D^2$) nanopores arranged in 512 channels, is capable to generate 10–20 Gb of DNA sequence data over a 48 hours sequencing run [3]. For its unique properties, from the launch of MinION Access Programme (April 2014), MinION has become increasingly popular among biologists.

Although it is definitely more expensive (∼$50,000.00), GridION X5 fills the gap between MinION and ONT's top-of-the-range PromethION. Indeed, GridION X5 is a small benchtop sequencer that allows to use up to five MinION Flow Cells at the same time; as result, GridION X5 drastically increases the ouput of a single MinION experiment to more than 100 Gb of DNA sequence data over a 48 hours sequencing run, making this instrument suitable for researchers with broader scope projects.

While library preparation is fast and easy when working with ONT's devices, the biggest problem researchers have to face when dealing with these TGS technologies is certainly the analysis of their outputs.

ONT's MinION outputs binary *hdf5* files [4] organized in a hierarchical structure with groups, datasets and attributes; these raw data are processed in real time by the local base caller MinKNOW and, finally, stored again in *hdf5* format with the *.fast5* extension, splitted into *Pass* (*i.e.* basecalled *.fast5* files with mean quality score ≥ 7), *Fail* (*i.e.* not successfully basecalled *.fast5* files with mean quality score ¡ 7) and *Skip* (*i.e.* reads for which insufficient resources were available for basecalling) subclasses.

On the other hand, in ONT's GridION X5 MinKNOW performs device control and data acquisition while the software dogfish is used for passing reads to the basecalling resource Guppy in real-time. By default, GridION X5 outputs *.fast5* files with no basecalled information added, *.fastq* files and sequencing summary *.txt* files that describe each sequenced read.

As for the other next-generation sequencing platforms, data generated by ONT's devices must be rapidly assessed in order to determine how useful the data may be in making biological discoveries as higher the quality of data is, more confident the conclusions are.

In the past years a number of tools designed specifically for MinION data visualization became available, including poretools [5], poRe [6] and HPG pore [7].

Poretools was the first tool released for MinION data analysis and it implements several functions to compute statistics and extract key informations both from entire .fast5 files datasets and single .fast5 files; however, most of poretools' functions designed to return an overview of the sequencing experiment need to be run on the entire .fast5 file dataset, which basically means reading the same dataset multiple times to extract different key informations, causing a waste of time when the user need to call multiple functions. Poretools implements also a function to return a table containing key informations from a dataset of .fast5 files in a single run, but does not provide any function to compute/plot statistics from this table.

PoRe shares most of poretools' features and in 2017 a GUI for parallel and real-time extraction of key statistics from MinION data has been released [8]: by using multiple cores, the GUI version makes it faster to extract informations from .fast5 files but, again, their downstream statistical analysis is extremely limited.

HPG pore, at last, features the capability to extract informations from .fast5 files and plot some statistics by parallelizing the job inside an Hadoop distributed file system, which lets you store and handle massive amount of data on a cloud of machines: for this reasons, HPG pore's true potentiality can be exploited only in a Hadoop cluster. In addition, HPG pore can only be run on Linux machines.

Taking all this into account, we developed NanoR, a package for the cross-platform statistical language and environment R [9], which is growing more and more popular with researchers, to analyze and compare both MinION and GridION X5 1D sequencing data. Our package, which has been tested on all the three main computer operating

systems (Linux, MacOSX and Windows), is designed to be extremely easy to use and allows users to retrive a complete overview of their sequencing runs within acceptable time frames, offering the possibility to parallelize the extraction of key informations from .fast5 files, which is the most time-consuming step of the analysis process.

Results

NanoR is a cross-platform package for the statistical language and environment R, requiring the following additional packages:

- ggplot2 [10]
- seqinr [11]
- reshape2 [12]
- scales [13]
- RColorBrewer [14]
- gridExtra [15]
- rhdf5 [16]
- ShortRead [17]

Further informations on the versions of the aforementioned packages needed to run NanoR are available in the *Availability and requirements* section. NanoR is built on 4 main functions:

- *NanoPrepare()*: prepares your MinION/GridION X5 data for the following analyses
- *NanoTable()*: generates a table that contains metadata for every sequenced read
- *NanoStats()*: plots statistics for nanopore data
- *NanoFastq()*: extract *.fastq* sequences from basecalled *.fast5* reads

Each of these functions exists both in a version for MinION (M) and in a version for GridION X5 (G) data analysis (*e.g.* for *NanoPrepare()* exists both the *NanoPrepareM()* and the *NanoPrepareG()* version). In addition, we provide *NanoCompare()* to allow comparison of both MinION and GridION X5 data in one command and *FastqFilterG()* to allow GridION X5 *.fastq* files filtering when working with *.fastq* files and sequencing summary files returned by this platform.

Below are described in more detail two possible workflows (also summarized in fig 1) for users wishing to analyze their MinION or GridION X5 data.

Fig 1. NanoR workflow. NanoR can work with both MinION and GridION X5 sequencing outputs. Users have to rely on NanoFastqM/NanoFastqG to direclty extract .fastq sequences from raw basecalled .fast5 files and on FastqFilterG to filter low-quality .fastq sequences in GridION X5 outputs. NanoPrepare functions, as well as NanoTable and NanoStats can be used one after another to generate a complete overview for the sequencing run. NanoCompare, at last, allows one-command comparison of MinION/GridION X5 analyzed sequencing experiments

MinION data analysis

The MinION local basecaller MinKNOW stores passed, failed and skipped *.fast5* reads in different folders. As passed *.fast5* reads can be considered the high-quality reads from the experiment, NanoR is designed to take into account failed and skipped *.fast5* reads only for calculating their number and percentage with respect to the total number of reads produced during the experiment while the other statistics are computed on passed *.fast5* reads.

Data preparation

In order to prepare MinION sequencing data for further analyses, we provide *NanoPrepareM()*,which reads all the *.fast5* files in the *Pass*, *Fail* and *Skip* directories and organizes data so that they can be used by the other functions from NanoR. The user must provide to *NanoPrepareM()* the path to basecalled *.fast5* reads stored in the *Pass* directory and a label that will be used to name the experiment, while the paths to failed and skipped *.fast5* reads are optional. *NanoPrepareM()* returns an object of class list required by other functions from this package.

Extraction of *.fastq* and *.fasta* files

Each passed *.fast5* file contains sequence informations in *.fastq* format that the user may wish to extract for further analyses by using *NanoFastqM()*. *NanoFastqM()* takes, as inputs, the path to the *Pass .fast5* files folder and a path to a folder where the results of the *.fastq* extraction will be saved; optionally, *NanoFastqM()* can also return a corresponding .fasta file and can be accelerated specifying multiple cores. Users can also set a minimum quality treshold different than the default one for passed .fast5 files (7): this will be used to skip passed sequences that does not meet the specified quality criteria.

Extraction of *.fast5* files metadata

Each basecalled *.fast5* file contains a number of key statistics (*i.e.* metadata) such as the read identifier, the number of both channel and mux that generate a certain read, its generation time as well as its length and quality score; *NanoTableM()* extracts all these informations from each passed *.fast5* read, optionally together with its GC content. Required parameters for this function are the list object returned by *NanoPrepareM()* and the path to a folder where the results of the analysis will be saved. As metadata extraction from *.fast5* sequences can be slow for large datasets, *NanoTableM()* can be accelerated using multiple cores. *NanoTableM()* returns a table with 7 columns that describes each passed *.fast5* read.

Plot statistics

NanoStatsM() is the main function of the NanoR package for MinION data analysis; the user must provide to *NanoStatsM()* the object of class list returned by *NanoPrepareM()*,

the table returned by *NanoTableM()* and the path to the same folder specified for *NanoTableM()*. As results, *NanoStatsM()* returns 4 tables (that contains informations about reads and base pairs number, reads length and reads quality and are required by the function *NanoCompare()* from this package) and 5 *.pdf* files contaning:

- plots of cumulative reads and base pairs produced during the experiment
- plots of reads number, base pairs number, reads length (min, average and max) and reads quality (min, average and max) calculated every half an hour of experiment. (Fig 2)
- plot of read length and read quality compared jointly (Fig 3)
- plots of channels and muxes activity with respect to their real disposition on the MinION flowcell (these plots are useful to diagnose problems in particular areas of the flowcell) (Fig 4)
- plots that summarize number and percentage of passed and failed/skipped *.fast5* reads as well as GC content count (if computed by *NanoTableM()*) calculated for passed ones

Fig 2. Reads number, base pairs number, reads length and reads quality per-time bin.The plots show the number of reads (**Panel A**) and basepairs (**Panel B**), the maximum, average and minimum length of reads in $\log_{10}$ scale (**Panel C**) and the maximum, average and minimum quality of reads (**Panel D**), all calculated every 30 minutes of an experimental MinION run

Fig 3. Reads length and reads quality compared jointly.The plot shows the correlation between length (x axis) and quality (y axis) for $\sim$ 100000 MinION reads. The regression line highlights that longer the reads are, higher their quality score is.

Fig 4. Heatmap of channels and muxes activity. The plots show the base pairs productivity of channels (**Panel A**) and muxes (**Panel B**) with respect to their real disposition on the Flow Cell as described in `https://community.nanoporetech.com/technical_documents/hardware/v/hwtd_5000_v1_revh_03may2016/flow-cell-chip`; inactive channels and muxes are grey-colored.

NanoR with MinION non-classified data

Although NanoR is designed to work with passed MinION *.fast5* files, sometimes users may have to deal with a mix of passed and failed/skipped *.fast5* reads stored in the same folder. In this case, the path to the folder contaning the mixed *.fast5* files can be given to both *NanoFastqM()* and *NanoPrepareM()* instead of the path to the passed *.fast5* files and the list object returned by *NanoPrepareM()* can be given to *NanoTableM()*. Indeed, both *NanoFastqM()* and *NanoPrepareM()* are capable to discriminate between passed and failed/skipped sequences autonomously. After that, the object of class list returned by *NanoPrepareM()* and the table returned by *NanoTableM()* can be given, as inputs, to *NanoStatsM()*, which simply interprets the input table and compute statistics for high-quality reads.

GridION X5 data analysis

By default, GridION X5 outputs *.fast5* reads without basecalled informations inside, *.fastq* files and sequencing summary *.txt* files that describes each read. Due to the fact that *.fastq* sequences and sequencing summary files are generated for both failed and passed reads, NanoR, during the analysis, will filter out failed reads (only their number and percentage will be taken into account) and compute statistics only on the passed ones.

Data preparation

In order to prepare GridION X5 sequencing data, we provide *NanoPrepareG()*. Required inputs for *NanoPrepareG()* are the path to the folder containing *.fastq* and sequencing summary files and a label that will be used, together with the Flow Cell identifier, to name the experiment. This function returns an object of class list required by other functions from this package.

Filter *.fastq* files

As previously said, the *.fastq* files returned by GridION X5 contains informations for both passed and failed sequences. We provide *FastqFilterG()* to filter out failed *.fastq* files; the user must provide to *FastqFilterG()* the path to the folder that contains *.fastq* files and sequencing summary files, the path to a folder in which pass *.fastq* files will be stored and a label that will be used, together with the Flow Cell identifier, to name the experiment. Optionally, *FastqFilterG()* can translate the outputted .fastq sequences into .fasta ones. As for *NanoFastqM()*, users can also set a minimum quality treshold different than the default one for passed sequences (7), which will be used to skip sequences whose quality is lower than the specified value.

Metadata selection

The sequencing summary files returned by GridION X5 contain metadata required to compute significant statistics on the sequenced reads. *NanoTableG()* takes, as inputs, the list returned by *NanoPrepareG()* and the path to a user-defined folder in which the result of this function will be saved: as result, *NanoTableG()* simply selects the most meaningful metadata from the sequencing summary files previously arranged by *NanoPrepareM()* and extracts GC content from *.fastq* files[1]: the resulting table contains 7 columns that describe each GridION X5 outputted sequence.

Plot statistics

NanoStatsG(), as it is the counterpart of *NanoStatsM()* for the GridION X5 data analysis, is the main function of NanoR package for GrdiION X5 data analysis; it takes exactly the same inputs required by *NanoStatsM()* and returns exactly the same outputs with the only exception that, as there are no informations about the active muxes in the sequencing summary files, only the channels activity heatmap will be plotted.

NanoR with GridION X5 basecalled *.fast5* files

NanoR functions for GridION X5 data analysis can also deal with basecalled *.fast5* files. Indeed, when analyzing basecalled GridION X5 *.fast5* files, the user has simply to

[1] It has been reported by some members of the Oxford Nanopore community that, occasionally, *.fastq* files returned by GridION X5 are ill-formatted: this problem was theoretically solved with the release of Guppy 1.4.0. However, if NanoR encounter one or more corrupted *.fastq* files, GC content calculation and *.fastq* files filtering will stop and NanoR will print the number of the ill-formatted *.fastq* file

switch to "TRUE" the *BasecalledFast5* parameter of *NanoPrepareG()* and give to the *Data* parameter of this function the path to the folder that contains the basecalled *.fast5* files (*e.g* the path to the *Pass .fast5* files folder returned by ONT's official command-line basecaller Albacore) as well as a label that will be used, together with the Flow Cell identifier, to name the experiment. All the other analyses will work as described in the "MinION data analysis" subsection.

Compare MinION/GridION X5 data

When dealing with 2 or more MinION/GridION X5 experiments, users may wish to compare their sequencing runs: to this puprose we provide *NanoCompare()*. *NanoCompare()* takes, as inputs, a vector containing the paths to the folders where results from *NanoStatsM()*/*NanoStatsG()* are stored, an ordered vector contaning the identifier used to name each experiment (for MinION data, the labels chosen when running *NanoStatsM()*; for GridION X5 data, the labels used when running *NanoStatsG()* together with the Flow Cell identifier, separated with an underscore character), the path to the folder in which *NanoCompare()* plots will be saved and a logical expression indicating whether GC content count have been computed for all the data. *NanoCompare()* returns 2 *.pdf* files contaning:

- violin plots that compare reads number, base pairs number, reads length and reads quality of the given experiments every 10 hours of sequencing run-time (the default sequencing run-time is 48 hours) (Fig 5)
- histogram plots that compare reads length, reads quality and, optionally, reads GC content count

Fig 5. Violin plots for comparison between experiments. From top to bottom, the plots show the comparison between 4 GridION X5 experiments in term of reads number, base pairs number, reads mean length and reads mean quality. Comparison is done every 10 hours of experimental run, as the standard duration of a MinION/GridION X5 run is 48 hours.

Discussion

As described in detail in the previous section, NanoR is based on few functions (4 for MinION data Analysis, 5 for GridION X5 data analysis and 1 to compare MinION/GridION x5 experiments), thus being intuitive and easy to use, even for biologists with little knowledge of R language. It also produces meaningful data within acceptable time frames. Indeed, the most time-consuming step when dealing with .fast5 files is the metadata extraction. In order to demonstrate how our tool performs compared to other tools that run in the same environment, we compared the user waiting-time when extracting metadata from MinION passed *.fast5* files with 3 different R packages: the NanoR package (using *NanoTableM()*), with and without GC content calculation), the poRe package (using *extract.fast5()*, taken from the poRe parallel GUI and used to extract only metadata[2]) and the IONiseR [18] package (using

[2]The modified version of the poRe function we used extracts the .fast5 filename, the channel number, the read number, the read start time, the length of the sequence, the run identifier, the read identifier and the experiment start time and assesses the read quality returning a logical expression indicating whether the read is passed (T) or not (F).

readFast5Summary() and *readInfo()*[3]); In particular we randomly sampled increasing number of reads (25000,50000,100000,500000,1000000) from a total of $\sim$ 2000000 reads taken from 5 different MinION sequencing runs and we calculated the user waiting-time when extracting metadata from them. The aforementioned functions were parallelized with the R "parallel" package using 10 Intel®Xeon®CPU E5-46100 @ 2.40GHz cores on a 48 cores SUSE Linux Enterprise Server 11.

For each number of reads analyzed, we repeated the sampling-extraction step 5 times (all the time measures, together with their averages and standard deviations are attached in S1 Supporting Information). Cache was cleaned after each metadata extraction.

Fig 6 shows these results as LOESS curves: as expected, the user waiting-time increases, for all the three packages, when increases the number of the reads analyzed, with IONiseR that turned out to be the slowest and NanoR the fastest, if it doesn't have to read the *.fastq* information to compute GC content.

Indeed, *NanoTable()* function, instead of dumping .fast5 files, which requires to stream each file entirely, makes use of pointers, which allow direct access to that part of the file structure that contains the required information. This behaviour, also known as "lazy evaluation principle", allows users to save time when retrieving informations from files with nested architectures, as it avoids to stream the entire file before fetching the desired informations. Moreover, as the basecalled .fast5 files structure is nowadays fixed and the presence/absence of the .fastq dataset inside them allows to discriminate between passed and failed/skipped sequences, NanoR checks for this dataset before looking for other informations, skipping the file if it is not found, thus avoiding to spend time uselessly. In addition, NanoR pre-allocates a portion of computer memory to store informations from each read before the extraction and fills in values as it goes, making it fast to save retrived metadata. We have also assessed the user waiting-time when retrieving metadata from reads of different size, as the length-independent speed of analysis can be an useful feature when dealing with ultra-long reads [19]; in this case, all the three tools perform equally, as there isn't, as expected, an increase of the user-waiting time when the size of the reads increases (except for NanoR if it had to read the *.fastq* information to compute GC content: in that case, the user-waiting time increases in a length-dependent manner). NanoR has a number of features in common with other tools built to deal with MinION sequencing data such as poRe, IONiseR, poretools and HPG pore. Table 1 summarizes NanoR features and compares them to those implemented in the aformentioned tools. As shown in Table 1, NanoR reproduces most of the output generated by the other tools for MinION data analysis, overcoming their limitations. In particular, it extends the statistical analysis carried out by all the other tools (*e.g.*, it allows users to check the trend of the sequencing run every 30-minutes ((Fig 2)) and to compare reads length and reads quality jointly ((Fig 3)) as exclusive features) and is also faster then competitors in the R environment. Moreover, NanoR allows easy parallelization without the need to rely on distributed clusters, as it strictly depends for that on the R base package "parallel". As other unique features, NanoR is, at the moment, the only tool capable to deal with GridION X5 sequencing data and allows quick comparison of MinION/GridION X5 analyzed experiments, making it easy to visualize differences between experimental sequencing runs. An example of application of NanoR on GridION X5 data is shown in S2 Supporting Information

[3]The *readInfo()* function from IONiseR returns the filename, the number of both channel and mux that generate a certain sequence and its pass/fail status.

Fig 6. LOESS curves comparing NanoR (dark blue, light blue), poRe (red) and IONiseR (green) performances when extracting metadata informations from increasing number of *.fast5* reads (25000,50000,100000,500000,1000000) randomly sanpled from ∼ 2000000 reads coming from 5 MinION runs, using 10 Intel®Xeon®CPU E5-46100 @ 2.40GHz cores. Each sampling-extraction step was repeated 5 times. Under the same conditions (*i.e.* without GC content calculation), NanoR is the fastest in extracting metadata informations for all the groups of *.fast5* file considered (*e.g.* NanoR takes approximately 60 minutes to extract metadata from 1000000 *.fast5* files, poRe takes approximately 80 minutes and IONiseR takes over 10 hours) while the extraction of metadata together with GC content computation (light blue line) makes NanoR working slightly slower (it takes approximatley 100 minutes to both extract metadata informations from 1000000 *.fast5* files and calculate their GC content).

Table 1. Comparison between NanoR, poRe, IONiseR, poretools and HPG pore

Feature	NanoR	poRe	IONiseR	poretools	HPG Pore
Extract *.fastq*[*]	✓	✓	✓	✓	✓
Extract *.fasta*[*]	✓	✓		✓	✓
Extract metadata[*]	✓	✓	✓	✓	✓
Calculate/plot GC content[*]	✓				✓
Plot yield[*]	✓	✓	✓	✓	✓
Plot reads and base pairs number per time bin[*]	✓				
Plot min/mean/max reads quality and length per time bin[*]	✓				
Plot reads length versus reads quality[*]	✓				
Plot reads length histograms[*]	✓[4]	✓		✓	✓
Plot reads quality histograms[*]	✓[5]				
Plot channels and muxes occupancy[*]	✓		✓	✓[6]	
Plot reads and base pairs number per channel histograms[*]		✓			✓
Plot pass/fail/skip *.fast5* files number and percentage[*]	✓				
Extract informations and plots statistics from GridION X5 outputs[**]	✓				
Compare MinION/GridION X5 data	✓				

[4] NanoR plots reads length histograms for each experiment when comparing experiments with *NanoCompare()*
[5] NanoR plots reads quality histograms for each experiment when comparing experiments with *NanoCompare()*
[6] Poretools does not plot muxes occupancy
[*] These features refers to MinION data analysis
[**] NanoR computes statistics described for MinION data also for GridION X5 data

Conclusion

MinION and GridION X5 from ONT are real-time sequencing devices capable to generate very long reads required for precise characterisation of complex structural variants such as copy number alterations. While library preparation is not the "hard part" when working with these devices, researchers may have some difficulties in analyzing their outputted data. Here we presented NanoR, a cross-platform package for the widespread statistical language R, that allows easy and fast analysis of both MinION and GridION X5 outputs. Currently, NanoR has been extensively tested on (and can deal with) the following 1D sequencing data:

- MinION basecalled *.fast5* files, returned by both MinION local basecaller MinKNOW and ONT's official command-line basecaller Albacore

- GridION X5 default output files (*.fastq* files and sequencing summary *.txt* files)
- GridION X5 basecalled *.fast5* files returned by Albacore

NanoR can be easily improved in the future in order to be up-to-date with the expected releases of MinION and GridION X5 softwares and to extend its data analysis capability (*e.g.* we have already plan to make NanoR capable to deal with $1D^2$ sequencing data and with the new bulk .fast5 format) and, even though there are other programs that can be used to analyze MinION data such as poRe, IONiseR, poretools and HPG pore, NanoR has unique features that make this tool extremely useful for researchers that work with ONT's sequencing data.

Software availability and requirements

- Project name: NanoR
- Project home page: `https://github.com/davidebolo1993/NanoR`
- Operating systems: Linux, Windows, MacOSX
- Programming language: R
- Other requirements: $R \geq 3.1.3$, $ggplot2 \geq 2.2.1$, $reshape2 \geq 1.4.3$, $RColorBrewer \geq 1.1.2$, $scales \geq 0.5.0$, $seqinr \geq 3.4.5$, $gridExtra \geq 2.3$, $ShortRead \geq 1.24.0$, $rhdf5 \geq 2.14$
- License: GPLv3

A dataset ($\sim$ 11 GB) of MinION passed *.fast5* files supporting the figures of this article together with a dataset ($\sim$ 17 GB) coming from a GridION X5 run supporting data shown in S2 Supporting Information can be found at `https://faspex.embl.de/aspera/faspex/external_deliveries/611?passcode=6f668c8921cea5cc30d0a6c17ec228f6ba75936c&expiration=MjAxOS0wMi0yMlQxMDo1NjoOOVo=`. The complete set of plots returned by NanoR as well as exhaustive installation istructions and examples of how to run NanoR functions and a folder containing all the functions on which NanoR is built can be found at `https://github.com/davidebolo1993/NanoR`, together with the manual of the tool.

Declarations

Acknowledgements

This work has been supported by Ministero della Salute (GR-2011-02352026), and by Associazione Italiana per la Ricera sul Cancro (Investigator Grant 20307).

Competing interests

The authors declare that they have no competing interests.

Author's contributions

Davide Bolognini wrote and tested the R code. Niccolò Bartalucci and Alessandra Mingrino prepared MinION and GridION X5 libraries. Alberto Magi and Alessandro Maria Vannucchi supervised the project. Davide Bolognini and Alberto Magi wrote the manuscript.

References

1. Magi A, Semeraro R, Mingrino A, Giusti B, D'Aurizio R. Nanopore sequencing data analysis: state of the art, applications and challenges. Brief Bioinform. 2017;doi:10.1093/bib/bbx062.

2. Korlach J, Bjornson KP, Chaudhuri BP, Cicero RL, Flusberg BA, Gray JJ, et al. Real-time DNA sequencing from single polymerase molecules. Methods Enzymol. 2010;472:431–455. doi:10.1016/S0076-6879(10)72001-2.

3. Jain M, Olsen HE, Paten B, Akeson M. The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community. Genome Biol. 2016;17(1):239. doi:10.1186/s13059-016-1103-0.

4. The HDF Group. Hierarchical Data Format, version 5;.

5. Loman NJ, Quinlan AR. Poretools: a toolkit for analyzing nanopore sequence data. Bioinformatics. 2014;30(23):3399–3401. doi:10.1093/bioinformatics/btu555.

6. Watson M, Thomson M, Risse J, Talbot R, Santoyo-Lopez J, Gharbi K, et al. poRe: an R package for the visualization and analysis of nanopore sequencing data. Bioinformatics. 2015;31(1):114–115.

7. Tarraga J, Gallego A, Arnau V, Medina I, Dopazo J. HPG pore: an efficient and scalable framework for nanopore sequencing data. BMC Bioinformatics. 2016;17:107. doi:10.1186/s12859-016-0966-0.

8. Stewart RD, Watson M. poRe GUIs for parallel and real-time processing of MinION sequence data. Bioinformatics (Oxford, England). 2017;33(14):2207–2208. doi:10.1093/bioinformatics/btx136.

9. R Core Team. R: A Language and Environment for Statistical Computing; 2017. Available from: `https://www.r-project.org/`.

10. Wickham H. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York; 2009. Available from: `http://ggplot2.org`.

11. Charif D, Lobry JR. Seqin{R} 1.0-2: a contributed package to the {R} project for statistical computing devoted to biological sequences retrieval and analysis. In: Bastolla U, Porto M, Roman HE, Vendruscolo M, editors. Structural approaches to sequence evolution: Molecules, networks, populations. Biological and Medical Physics, Biomedical Engineering. New York: Springer Verlag; 2007. p. 207–232.

12. Wickham H. Reshaping Data with the {reshape} Package. Journal of Statistical Software. 2007;21(12):1–20.

13. Wickham H. scales: Scale Functions for Visualization; 2017. Available from: `http://cran.r-project.org/package=scales`.

14. Neuwirth E. RColorBrewer: ColorBrewer Palettes; 2014. Available from: `http://cran.r-project.org/package=RColorBrewer`.

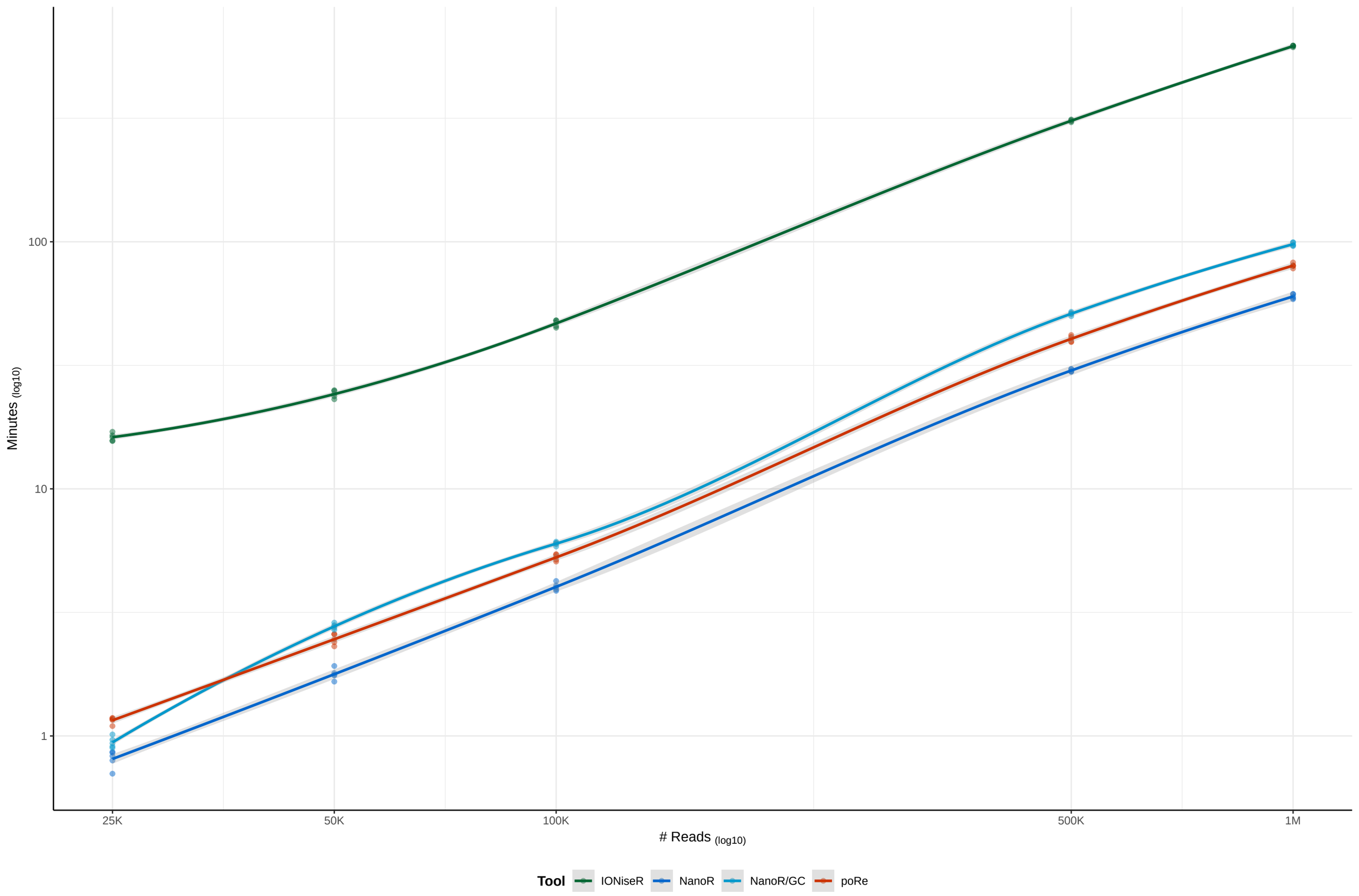
15. Auguie B. gridExtra: Miscellaneous Functions for "Grid" Graphics; 2017. Available from: `http://cran.r-project.org/package=gridExtra`.

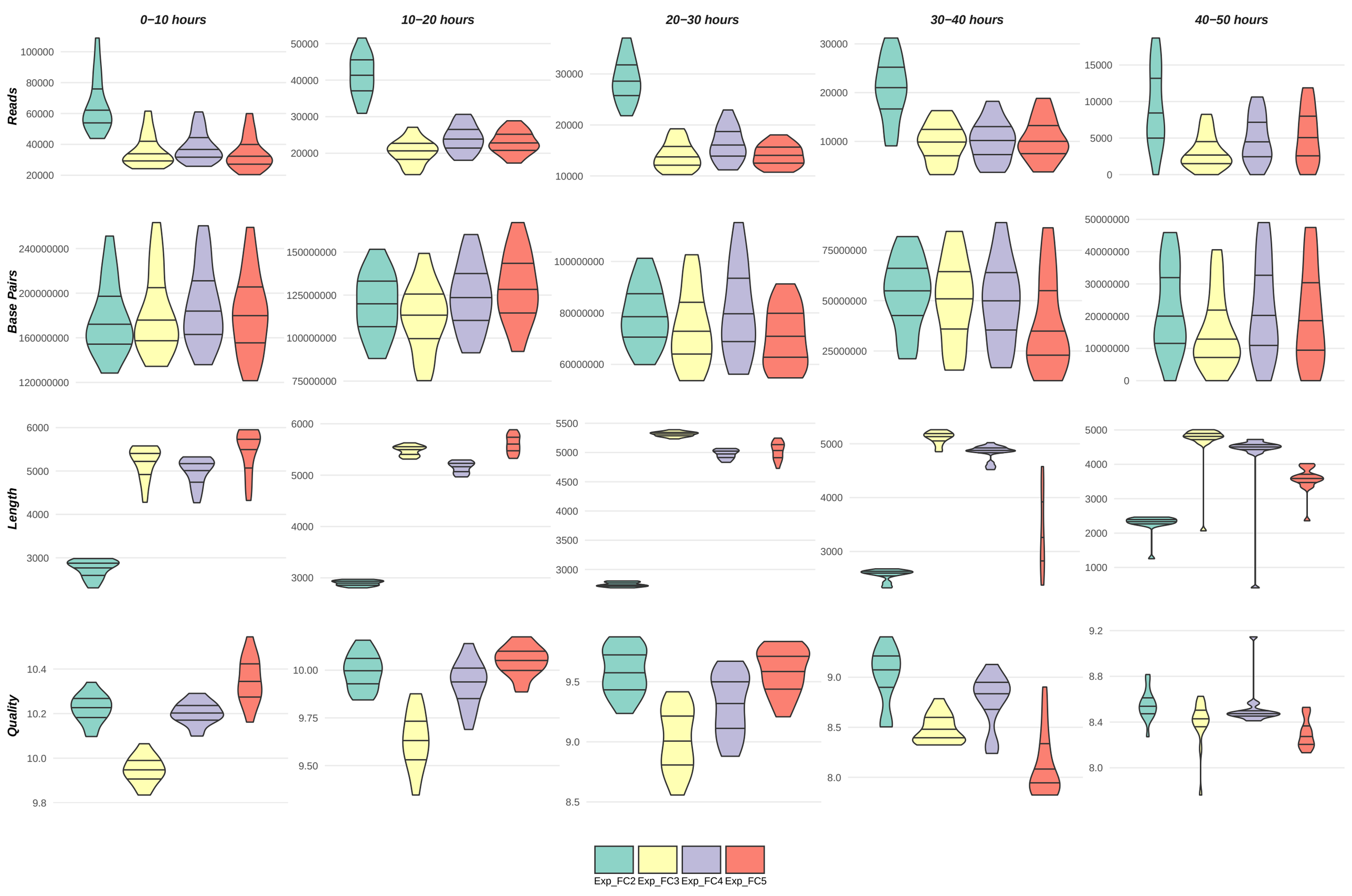
16. Fischer B, Pau G. rhdf5: HDF5 interface to R;.

17. Morgan M, Anders S, Lawrence M, Aboyoun P, Pagès H, Gentleman R. {ShortRead}: a {B}ioconductor package for input, quality assessment and exploration of high-throughput sequence data. Bioinformatics. 2009;25:2607–2608. doi:10.1093/bioinformatics/btp450.

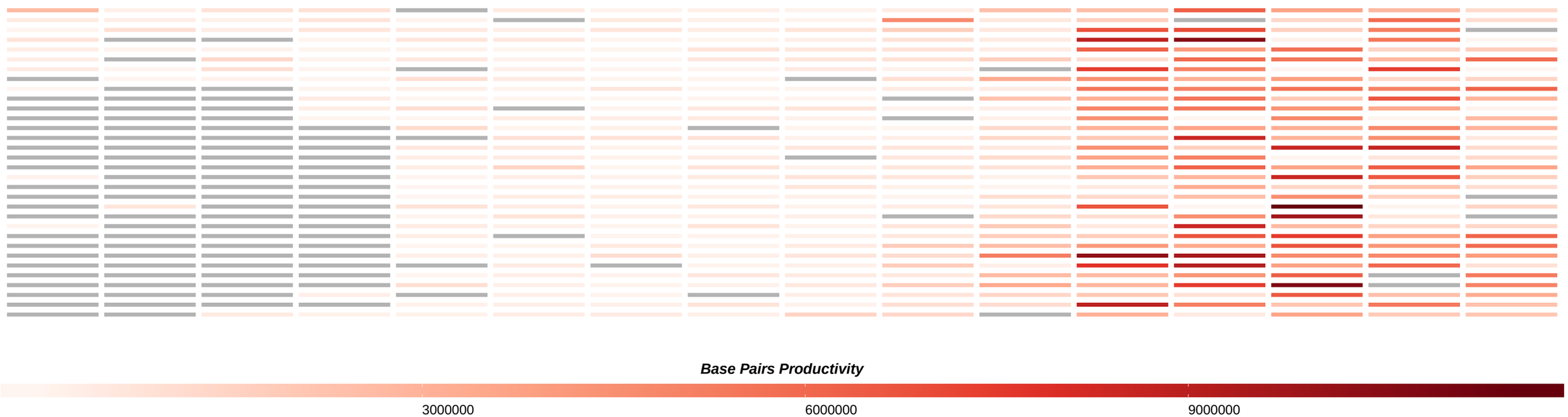
18. Smith M. IONiseR: Quality Assessment Tools for Oxford Nanopore MinION data; 2017.

19. Jain M, Koren S, Miga KH, Quick J, Rand AC, Sasani TA, et al. Nanopore sequencing and assembly of a human genome with ultra-long reads. Nature Biotechnology. 2018;36(4):338–345. doi:10.1038/nbt.4060.

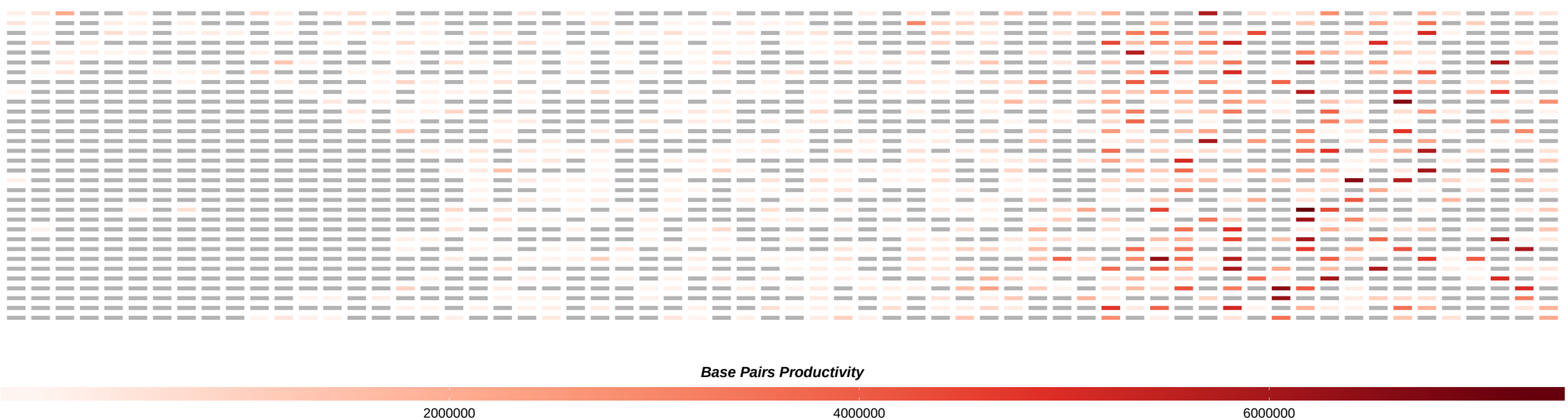

100
10
1
Minutes (log10)
25K
50K
100K
500K
1M
Reads (log10)
Tool
IONiseR
NanoR
NanoR/GC
poRe

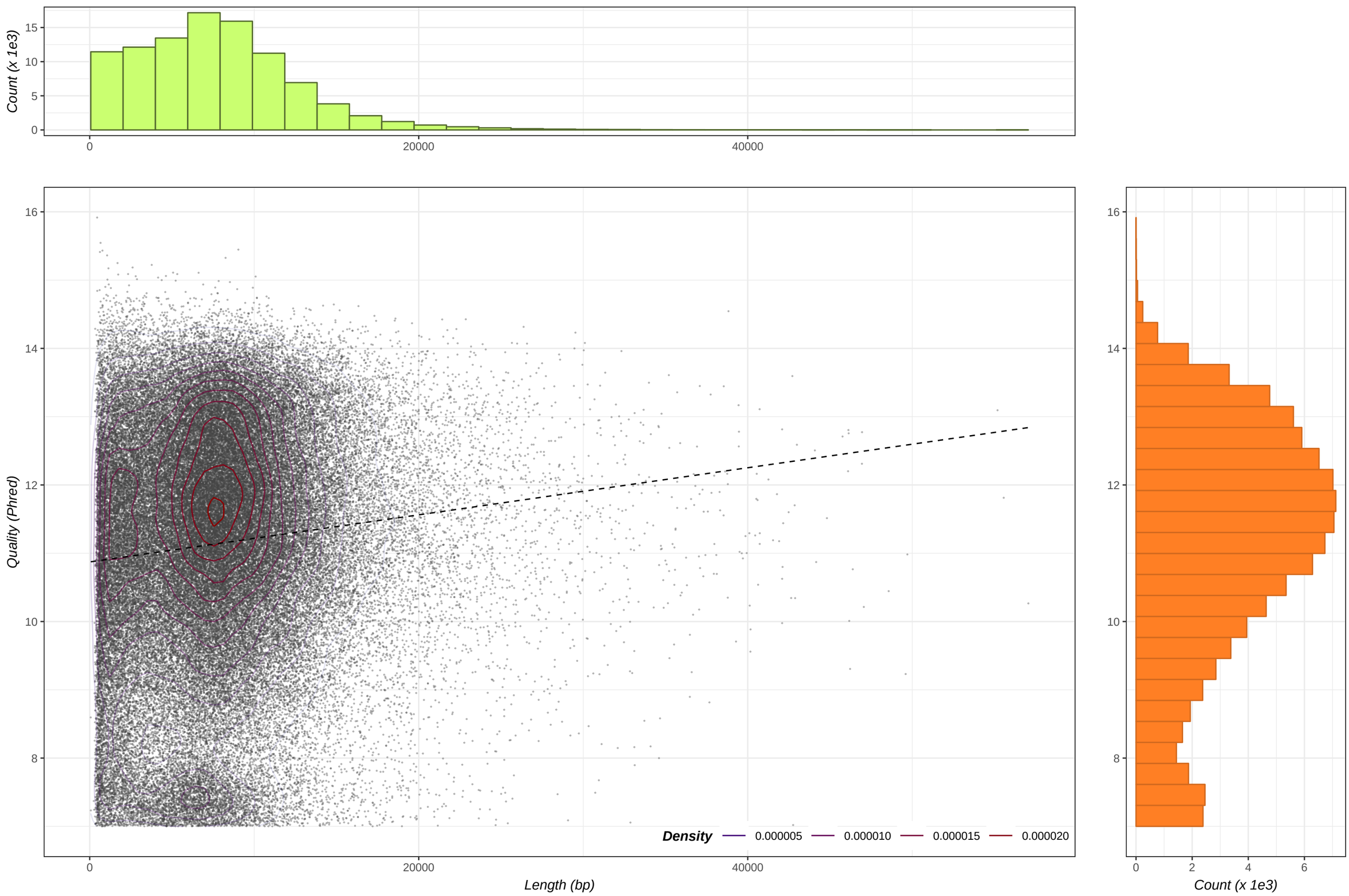

0−10 hours
10−20 hours
20−30 hours
30−40 hours
40−50 hours
Reads
Base Pairs
Length
Quality
Exp_FC2
Exp_FC3
Exp_FC4
Exp_FC5

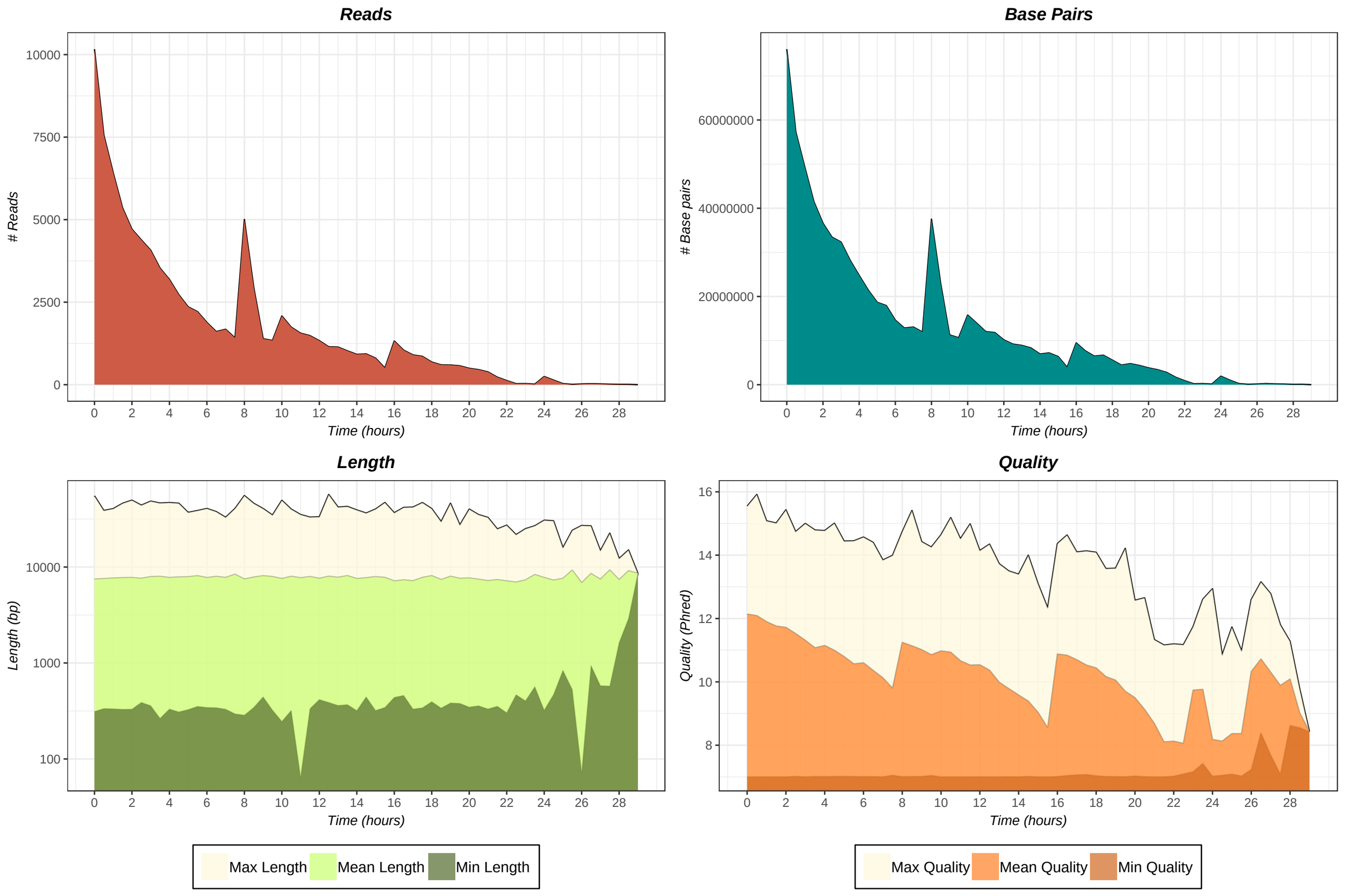
Channels Activity

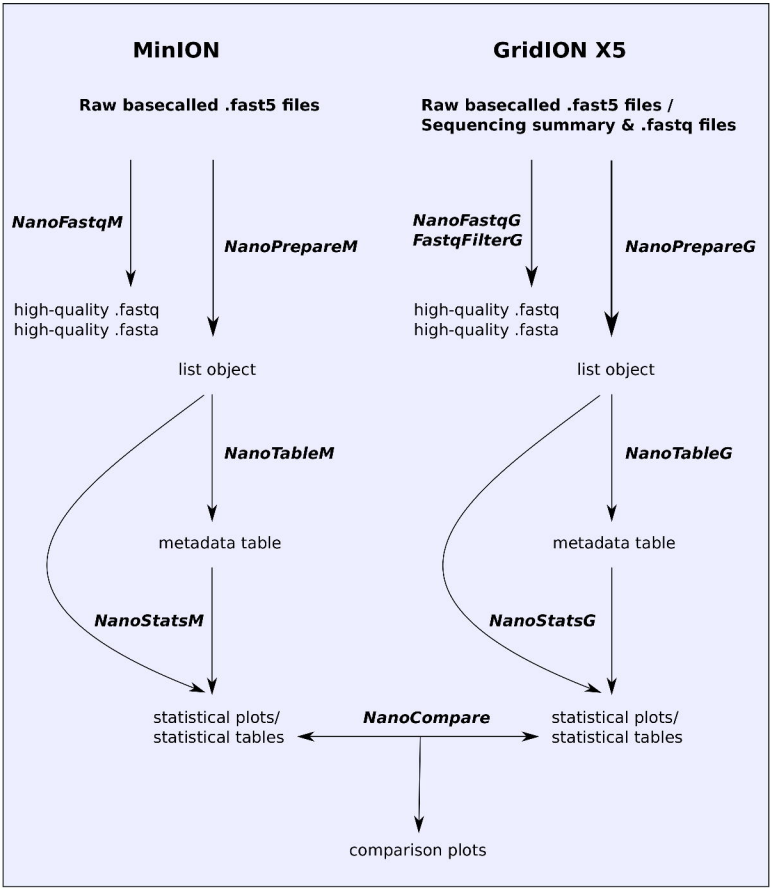


Muxes Activity




Count (x 1e3)
15
10
5
0
0
20000
40000
16
14
12
10
8
Quality (Phred)
Density
0.000005
0.000010
0.000015
0.000020
0
20000
40000
Length (bp)
16
14
12
10
8
0
2
4
6
Count (x 1e3)


Reads
Reads
10000
7500
5000
2500
0
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28
Time (hours)
Base Pairs
Base pairs
60000000
40000000
20000000
0
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28
Time (hours)
Length
Length (bp)
10000
1000
100
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28
Time (hours)
Max Length
Mean Length
Min Length
Quality
Quality (Phred)
16
14
12
10
8
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28
Time (hours)
Max Quality
Mean Quality
Min Quality


MinION
GridION X5
Raw basecalled .fast5 files
Raw basecalled .fast5 files /
Sequencing summary & .fastq files
NanoFastqM
NanoPrepareM
NanoFastqG
FastqFilterG
NanoPrepareG
high-quality .fastq
high-quality .fasta
high-quality .fastq
high-quality .fasta
list object
list object
NanoTableM
NanoTableG
metadata table
metadata table
NanoStatsM
NanoStatsG
statistical plots/
statistical tables
NanoCompare
statistical plots/
statistical tables
comparison plots