Computational Practical 4: Long-read assembly

Module Developers: Dr Fahad Khokar

Table Of Contents

5.1 Introduction

Long-read sequencing has several advantages over short-read sequencing. Firstly, long reads can span over longer genomic regions and thus provide more contiguous sequences, leading to more accurate assemblies and less fragmentation of the genome. This is particularly useful for resolving complex genomic regions such as repetitive sequences, structural variations, and regions with high GC-content. Additionally, long-read sequencing can detect larger structural variations and can be used to phase haplotypes, allowing for the study of the inheritance of genomic regions. Moreover, long-read sequencing can also facilitate the detection and characterization of previously unknown or unannotated elements such as large transposable elements, viral sequences and mitochondrial DNA. Finally, long-reads also enable to sequence and assemble genomes from single cells, providing insights into the diversity and structure of genomes in mixed populations.

The schematic below illustrates the benefits of using long reads in assembling repetitive regions. Short-read sequencing technology often leads to collapse of repeat and high-identity regions during assembly, whereas long reads have a greater chance of encompassing the entire repetitive region (depicted as blue boxes) resulting in a more precise assembly.

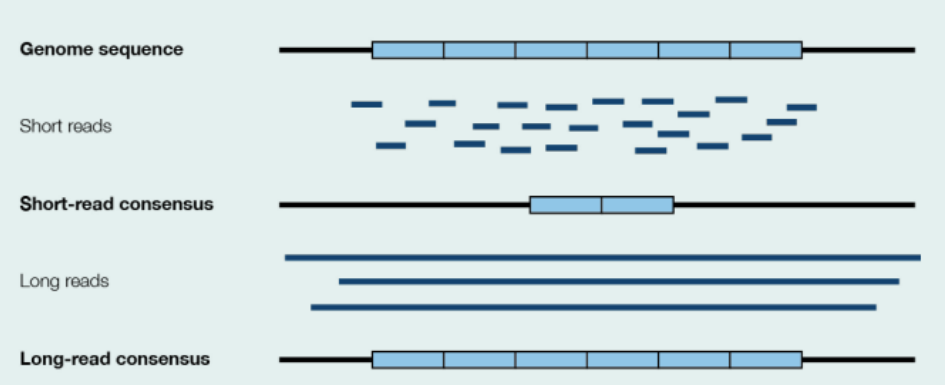


Fig 5.1. Short-read vs long-read in generating consensus sequences of repetitive regions. Image source: https://nanoporetech.com/sites/default/files/s3/white-papers/microbiology-white-paper.pdf (Figure 3).

The objective of this practical is to introduce a typical workflow used for genome assembly of long-read sequencing data. Within this practical you will perform genome assembly of two isolates, generated from a Nanopore sequencing run as demonstrated in Laboratory Practical 5.

Once the data has been basecalled and demultiplexed, a typical assembly workflow will include QC/filtering, genome assembly (and polishing), and finally using the generated files for downstream analyses.

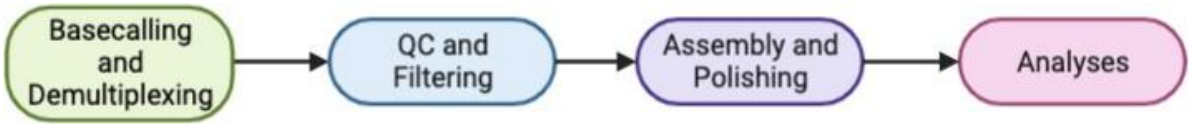


Fig 5.2. Flow chart of the main steps in processing Nanopore sequencing data for genome assembly.

5.2 Data availability and pre-processing

In this practical we will be using Nanopore long-read as well as Illumina short-read sequencing data. There will be two samples used in this session, named “bc01” and “bc02”. The Illumina paired-end data for sample “bc01” was retrieved from Accession: SRR531233, and for sample “bc02” from Accession: SRR6181295. The short-read data for sample “bc01” has been sub- sampled to around 800,000 reads, and both samples have been processed with fastp with default settings.

After basecalling the long-read raw sequencing files from .fast5 format into .fastq, Porechop was used to remove any additional adapter sequences from the read files. Additionally, Filtlong was used to filter the reads based on length and quality. The commands used are shown below:

```
$ fastp -i bc01_R1.fastq.gz -I bc01_R2.fastq.gz -o bc01_R1_trim.fastq.gz -O
bc01_R2_trim.fastq.gz -l 101

$ porechop -t 4 -i bc01.fastq.gz -o bc01.porechop.fastq.gz

$ filtlong –-min_length 1000 –-keep_percent 95 bc01.porechop.fastq.gz >
bc01_long-filtered.fastq.gz
```

After generating a long-read only assembly (section 5.5 Flye), the standard approach is to use the same long-read data to polish the draft assembly. This involves mapping the .fastq reads back to the .fasta assembly file to find overlaps which may be corrected. To save time this has already been done for both samples. The tool used to generate the new assembly file was Medaka with the following commands:

```
$ medaka_consensus -i reads/bc01_long-filtered.fastq.gz -d
medaka/bc01_flye.fasta -o medaka/ -t 4 -m r941_min_sup_g507 && cp
medaka/consensus.fasta assemblies/flye/bc01_flye_medaka.fasta && rm
-r medaka/
```

Multiple rounds of polishing can be performed, each time starting with the new .fasta assembly file generated. The two generated files are “bc01_flye_medaka.fasta” and “bc02_flye_medaka.fasta” in the /assemblies/flye/ folder. These will be used from section 5.7 Polish genomes using Polypolish.

In addition to the above pre-processing of the sequencing reads, short-read only and hybrid genome assemblies have already been generated for the two samples using Unicycler (see section 5.6).

Fastp	Doi: https://doi.org/10.1093/bioinformatics/bty560
	Github: https://github.com/OpenGene/fastp
Porechop	https://github.com/rrwick/Porechop
Filtlong	https://github.com/rrwick/Filtlong
Medaka	https://github.com/nanoporetech/medaka

5.3 Software check

To check the tools required are installed on the VM, open a new terminal window, and run the following commands:

```
$ NanoPlot -h
$ flye -h
$ polypolish -h
$ polypolish_insert_filter.py -h
```

If all are installed correctly, you should see the help page of each tool, and

no error. Navigate to the directory for Practical 5:

```
$ cd ~/course/cp5/
```

You should see three folders (assemblies, reads and trycycler) inside the directory, with the following files in their subdirectories:

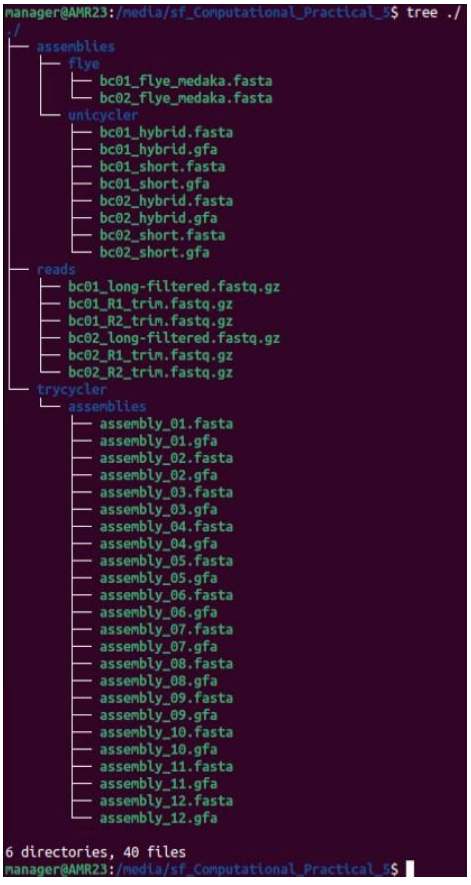


Fig 5.3. Directory listing in VM that should be visible. Sequencing reads are in the “/reads” folder, with assemblies already generated and to generate stored in “/assemblies”.

5.4 Sequence QC

The first step before proceeding with genome assembly is to perform a quality check of the sequencing data. A popular tool to use for long-read data is NanoPlot, which produces a .html report for each sample we want to analyse.

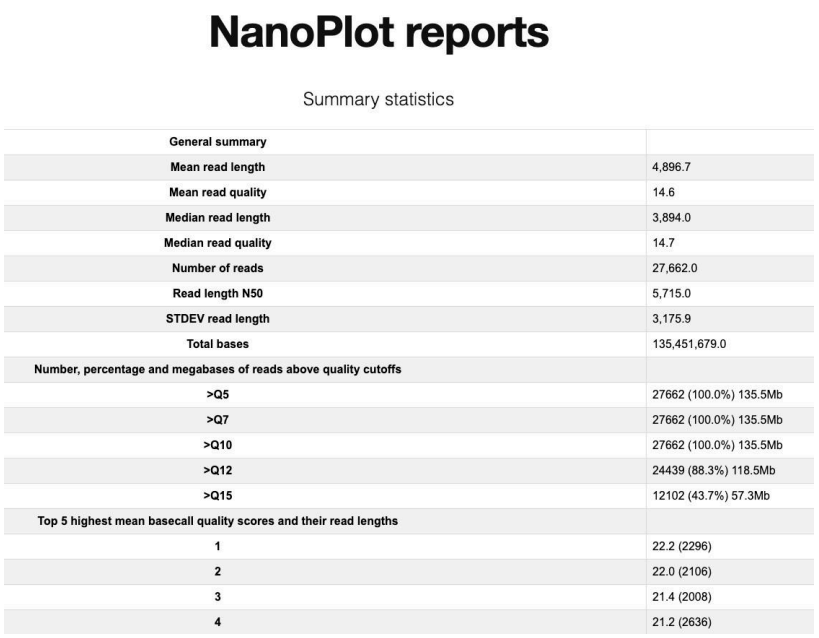
The long-read data for the samples we are processing are in the "/reads" folder, bc01_long- filtered.fastq.gz and bc02_long-filtered.fastq.gz. To run NanoPlot on sample bc01, enter the following command in the terminal:

```
$ NanoPlot -t 4 –-fastq reads/bc01_long-filtered.fastq.gz -o reads/NanoPlot/ -p bc01.
&& cp reads/NanoPlot/bc01.NanoPlot- report.html reads/bc01.NanoPlot-report.html
&& rm -r reads/NanoPlot
```

(Note: if processing sample "bc02" simply edit the command above accordingly)

This should produce the file reads/bc01.NanoPlot-report.html (or reads/bc02.NanoPlot- report.html). Open the file to view the report. Either navigate to the folder using the File Viewer software, or type the following into the terminal:

```
$ firefox reads/bc01.NanoPlot-report.html
```



NanoPlot reports

Summary statistics

General summary	
Mean read length	4,896.7
Mean read quality	14.6
Median read length	3,894.0
Median read quality	14.7
Number of reads	27,662.0
Read length N50	5,715.0
STDEV read length	3,175.9
Total bases	135,451,679.0
Number, percentage and megabases of reads above quality cutoffs	
>Q5	27662 (100.0%) 135.5Mb
>Q7	27662 (100.0%) 135.5Mb
>Q10	27662 (100.0%) 135.5Mb
>Q12	24439 (88.3%) 118.5Mb
>Q15	12102 (43.7%) 57.3Mb
Top 5 highest mean basecall quality scores and their read lengths	
1	22.2 (2296)
2	22.0 (2106)
3	21.4 (2008)
4	21.2 (2636)

Fig 5.4. Screenshot showing the top of the NanoPlot report generated for sample bc01 using the above command.

5.5 De novo genome assembly with Flye

Manuscript: Assembly of long, error-prone reads using repeat graphs. Doi: https://doi.org/10.1038/s41587-019-0072-8

Github: https://github.com/fenderglass/Flye

To perform this genome assembly step, we will first start a new screen session so Flye will continue to run in the background whilst we continue onto the next step. To do this enter the following command:

```
$ screen -S flye
```

If the data has passed the initial QC, we can try to assemble the genomes of the samples. We will use the popular Flye tool for de novo assembly. We will need to use the same long-read sequence files as the previous step. Ensure you are currently in the same directory as before (~/course/cp5/). The command to process sample "bc01" using Flye is:

```
$ flye -t 4 --nano-hq reads/bc01_long-filtered.fastq.gz -o assemblies/flye/bc01/ &&
cp assemblies/flye/bc01/assembly.fasta assemblies/flye/bc01_flye.fasta && cp
assemblies/flye/bc01/assembly_graph.gfa
assemblies/flye/bc01_flye.gfa && rm -r assemblies/flye/bc01
```

(Note: if processing sample "bc02" simply edit the command above accordingly)

This will now go through the stages of genome assembly with Flye, which will take around 10 and 20 minutes for samples bc01 and bc02 respectively.

To continue using the terminal for other commands, we can detach/hide the current screen session. To do this press Ctrl + A + D, which should bring you back out to the main screen. To re-enter the Flye screen, enter the command "screen -r flye".

The resulting output from genome assemblers will include; assembled contigs in .fasta format ("bc0x_flye.fasta"), and an assembly graph file in .gfa format ("bc0x_flye.gfa").

5.6 Unicycler

Unicycler: Resolving bacterial genome assemblies from short and long sequencing reads. Doi: https://doi.org/10.1371/journal.pcbi.1005595

https://github.com/rrwick/Unicycler

Let's first look at the short-read only assemblies generated using the paired-end fastq sequence files from an Illumina HiSeq. These were assembled using the Unicycler tool (which uses the SPAdes assembler) with the following command:

```
$ unicycler -t 4 -1 reads/bc01_R1_trim.fastq.gz -2 reads/bc01_R2_trim.fastq.gz -o assemblies/unicycler/bc01 && cp assemblies/unicycler/bc01/assembly.fasta assemblies/unicycler/bc01_short.fasta && cp assemblies/unicycler/bc01/assembly.gfa assemblies/unicycler/bc01_short.gfa && rm -r assemblies/unicycler/bc01/
```

As with the output from Flye, Unicycler also produces a .fasta and .gfa file. Using a program called Bandage, we can visualise the genome assembly using these files. Type the following command to activate the conda environment:

```
$ conda activate bandage
```

Once activated we can load the .gfa file into the program to visualise. Type the following in the terminal to generate the visualisation:

```
$ Bandage load /assemblies/unicycler/bc01_short.gfa --draw
```

This should produce graphs similar to the ones below, depending on which assembly you are viewing.

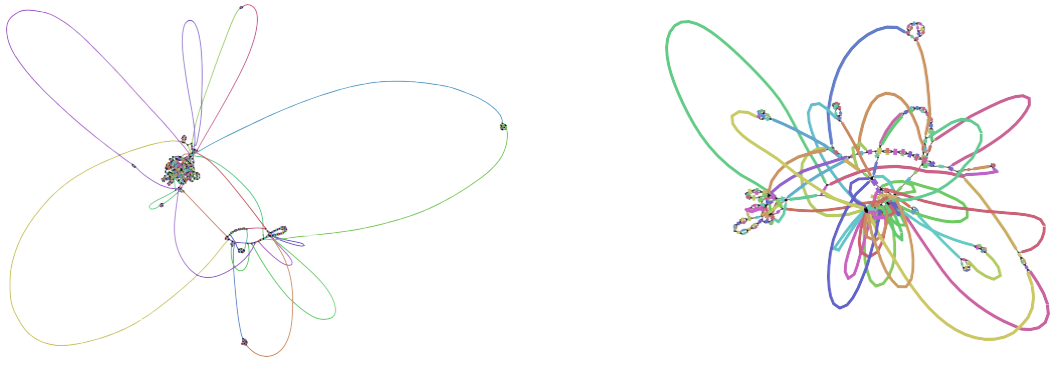


Fig 5.5. Output of bc01_short.gfa (left) and bc02_short.gfa (right), viewed in

Bandage

Unicycler can also be used with both short- and long-read sequencing data, to produce a “hybrid” genome assembly. We can use the same long-read sequencing file that was used with Flye, in combination with the same short-read data used. To run a hybrid assembly using Unicycler, we can simply add the “-l” tag (l = long reads) into the command as below:

```
$ unicycler -1 reads/bc01_R1_trim.fastq.gz -2 reads/bc01_R2_trim.fastq.gz -l
reads/bc01_long-filtered.fastq.gz -o assemblies/unicycler/bc01 && cp
assemblies/unicycler/bc01/assembly.fasta assemblies/unicycler/bc01_hybrid.fasta
&& cp assemblies/unicycler/bc01/assembly.gfa
assemblies/unicycler/bc01_hybrid.gfa && rm -r
assemblies/unicycler/bc01
```

The first part of the hybrid assembly is the same as the short-read only method, generating an assembly using SPAdes. Unicycler then uses the long-reads to build bridges, which often allows it to resolve all repeats in the genome, resulting in a complete genome assembly.

To view the output of the hybrid assemblies, we can repeat the previous commands for Bandage, replacing the input file with the newly generated .gfa:

```
$ Bandage load assemblies/unicycler/bc01_hybrid.gfa --draw
```

The output should look similar to the figures below for bc01 and bc02.

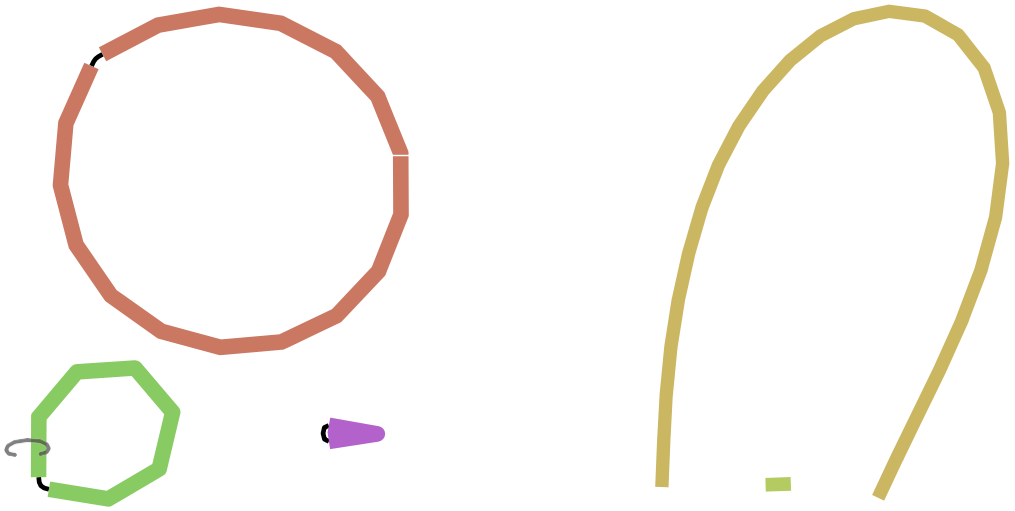


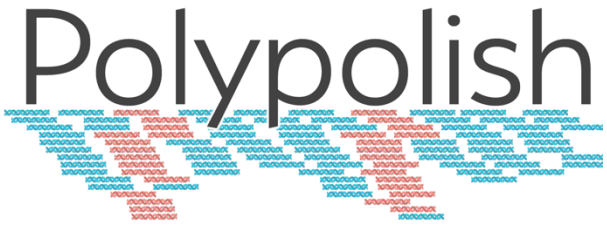
Fig 5.6. Output of bc01_hybrid.gfa (left) and bc02_hybrid.gfa (right) viewed in Bandage.

When finished using Bandage, remember to “deactivate” from the conda environment:

```
$ conda deactivate bandage
```

5.7 Polish genomes using Polypolish

Polypolish is a tool for improving genome assemblies using short reads. It stands out from other similar tools by utilizing SAM files that have been aligned to multiple locations, rather than just one. This feature enables Polypolish to repair errors in repeat regions that other alignment-based tools cannot fix.



Polypolish: Short-read polishing of long- read bacterial genome assemblies. Doi: https://doi.org/10.1371/journal.pcbi.100 98 02

https://github.com/rrwick/Polypolish/wiki

The first step of Polypolish is to create alignments from the short-read data. To do this, we need to index our draft long-read genome (bc0x_flye_medaka.fasta), then input the R1 and R2 files separately to create to new alignment files. These alignments files are then trimmed before being used to polish the genome. For a full explanation of why using Polypolish is the preferred method of short-read polishing, follow the example on this link https://github.com/rrwick/Polypolish/wiki/Toy-example

Run the following commands to generate a “polypolished” assembly file for our samples:

```
$ bwa index assemblies/flye/bc01_flye_medaka.fasta

$ bwa mem -t 4 -a assemblies/flye/bc01_flye_medaka.fasta
reads/bc01_R1_trim.fastq.gz > bc01_alignments_1.sam

$ bwa mem -t 4 -a assemblies/flye/bc01_flye_medaka.fasta
reads/bc01_R2_trim.fastq.gz > bc01_alignments_2.sam

$ polypolish_insert_filter.py –-in1 bc01_alignments_1.sam –-in2
bc01_alignements_2.sam –-out1 bc01_filtered_1.sam –-out2 bc01_filtered_2.sam


$ polypolish assemblies/flye/bc01_flye_medaka.fasta bc01_filtered_1.sam
bc01_filtered_2.sam > assemblies/bc01_polypolished.fasta && rm *.sam
```

5.8 QUAST

QUAST: quality assessment tool for genome assemblies. Doi: https://doi.org/10.1093/bioinformatics/btt086

QUAST (Quality Assessment Tool for Genome Assemblies) is a tool to evaluate genomes assemblies. It works both with or without a reference genome, on command line or via the web interface. To use the web interface, open the Firefox application and type the QUAST online website http://cab.cc.spbu.ru/quast.

Navigate to the appropriate assembly folder and drag and drop the following files into the box on the website:

1. /assemblies/unicycler/bc01_short.fasta
2. /assemblies/unicycler/bc01_hybrid.fasta
3. /assemblies/flye/bc01_flye.fasta
4. /assemblies/flye/bc01_polypolished.fasta

Once uploaded select the “Prokaryotic” option and click on “Evaluate”. After a few minutes the report will be ready to view by clicking on the date on the right side (may need to refresh page). This should produce basic stats on the genome assemblies provided.

5.9 Trycycler

Trycycler: consensus long-read assemblies for bacterial genomes. Doi: https://doi.org/10.1186/s13059-021-02483-z

https://github.com/rrwick/Trycycler/wiki

Recent advancements in long-read assembly have led to the availability of multiple efficient assemblers such as Canu, Flye, Raven, and Redbean. Due to the straightforward nature of bacterial genomes (small size and minimal repetitions), it is often possible to achieve a final assembly with one contig per replicon when utilizing long reads.

Despite their effectiveness, even the most advanced assemblers have their limitations. Common issues include difficulty in circularizing some sequences, resulting in duplication or omission of sequence at the start/end of a contig. Additionally, they may generate spurious contigs by assembling repetitive parts of the chromosome separately, omit entire replicons, introduce medium-scale indel errors such as deleting 50 bp from the genome and even create large-scale misassemblies such as significant structural rearrangements.

Trycycler is a tool that utilizes multiple separate long-read assemblies of the same genome (e.g. from different assemblers or different read subsets) to create a consensus long-read assembly. The tool performs the following functions:

- Clustering of contig sequences to distinguish complete contigs (i.e. those that correspond to an entire replicon) from spurious and/or incomplete contigs.
- Reconciliation of alternative contig sequences and correction of circularization issues.
- Multiple sequence alignment (MSA) of alternative sequences.
- Construction of a consensus sequence from the MSA by choosing between variants where the sequences differ. The outcome is a long-read assembly that is more accurate and trustworthy.

An illustrated pipeline overview of Trycycler can be found on this link: https://github.com/rrwick/Trycycler/wiki/Illustrated-pipeline-overview.

To use Trycycler on sample “bc02”, first activate the conda environment:

```
$ conda activate trycycler
```

Navigate to the “/trycycler” directory. “Step 1: Generating Assemblies” has already

been completed, the output of which is in the /trycycler/assemblies folder. This contains a total of 12 assemblies generated using read subsets of bc02_long-filtered.fastq.gz. These assemblies can now be used from “Step 2: Clustering contigs” onwards.

5.11 Additional

5.11.1 To polish, or not to polish?

Recent developments in the Nanopore chemistry and technologies have further increased the accuracy of sequencing. This is due to a new type of protein nanopore, version 10.4, which has two read sensors rather than one, as illustrated in the image below.

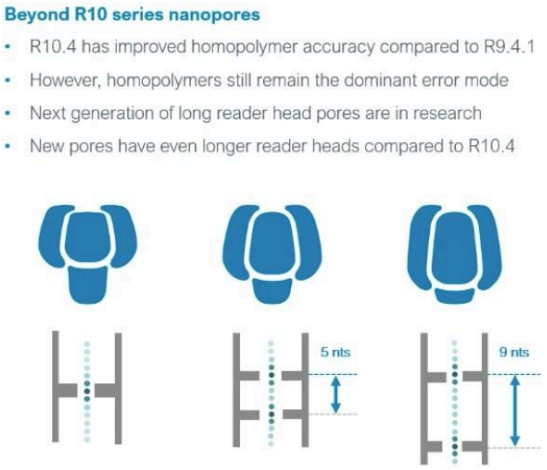


Fig 5.7. Schematic comparison of R9.4.1and R10.4 pores.

This paper by Sereika et. al., from 2022 describes the use of the latest R10.4 nanopore to generate high-quality bacterial genome assemblies. This is available at Doi: https://doi.org/10.1038/s41592-022-01539-7


Brief Communication | Open Access | Published: 04 July 2022

Oxford Nanopore R10.4 long-read sequencing enables the generation of near-finished bacterial genomes from pure cultures and metagenomes without short-read or reference polishing

Mantas Sereika, Rasmus Hansen Kirkegaard, Søren Michael Karst, Thomas Yssing Michaelsen, Emil Aarre Sørensen, Rasmus Dam Wollenberg & Mads Albertsen

Nature Methods **19**, 823–826 (2022) | Cite this article

23k Accesses | **15** Citations | **158** Altmetric | Metrics

The authors demonstrate that using this technology, it is possible to generate near-finished genomes from pure cultures and metagenomes without the need for additional short-read or reference polishing. The results show that the R10.4 pore flow cell generates assemblies with a high level of accuracy, completeness, and continuity, and that it is a cost-effective alternative to traditional short-read sequencing. The study concludes that this technology provides a powerful tool for bacterial genome assembly and will be useful for bacterial strain identification, comparative genomics, and functional genomics.

5.11.2 Nature Method of the Year 2022

In June 2022, a special issue was released that showcased the success of the Telomere-to- Telomere (T2T) Consortium in completing the first full human genome. A combination of experimental and computational techniques were used to achieve this milestone, with long- read sequencing being the primary method for generating the T2T data. This technology is the foundation of this achievement. However, the T2T Consortium's work is just one example of the many breakthroughs that long-read sequencing is enabling in reading genomes, transcriptomes and epigenomes in humans and other species. Due to its significant methodological advancement and wide range of applications, long-read sequencing was chosen as the Method of the Year 2022.

Editorial | Published: 12 January 2023

Method of the Year 2022: long-read sequencing

Nature Methods **20**, 1 (2023) | Cite this article

12k Accesses | **693** Altmetric | Metrics

Long-read sequencing powers a more complete reading of genomic information.

You can read the full article using this link: Doi: https://doi.org/10.1038/s41592-022-01759-x