Introduction to Nanopore: Principles, Updates, and Bioinformatics

Genome assembly using Nanopore data

Rania Ouazahrou 04-09-2025

Biomics platform, Institut Pasteur Paris

Bioinformatics and biostatistics hub, Institut Pasteur Paris

Intended Learning Objectives

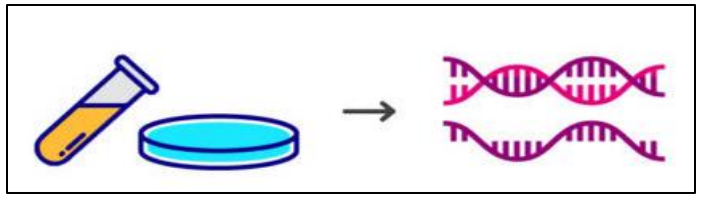
Specific objectives of this session:

1. Learn how to perform quality control of Nanopore data
2. Learn about genome assembly using Nanopore data

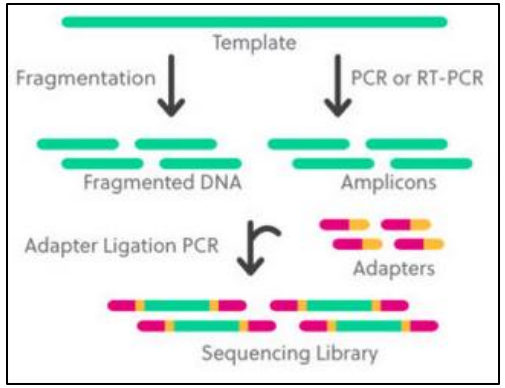
Outline

This session consists of the following elements

1. Perform quality control of Nanopore reads
   - Filter Nanopore reads based on length and quality
2. Introduction to genome assembly steps
   - Reads correction
   - Assembly
   - Polishing assembly
   - Assessment of assembly quality



Step 1: DNA/RNA extraction



Step 2: Library preparation

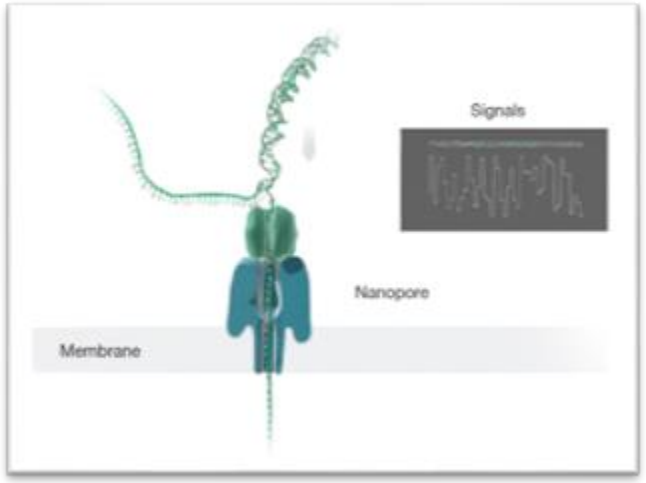
Step 3: Sequencing

Step 4: Data analysis

Step 1: Basecalling, QC

Basecalling

Sequencing signal



Basecalling

Nucleotidic sequences (fastq/bam)

Fastq files (reads)

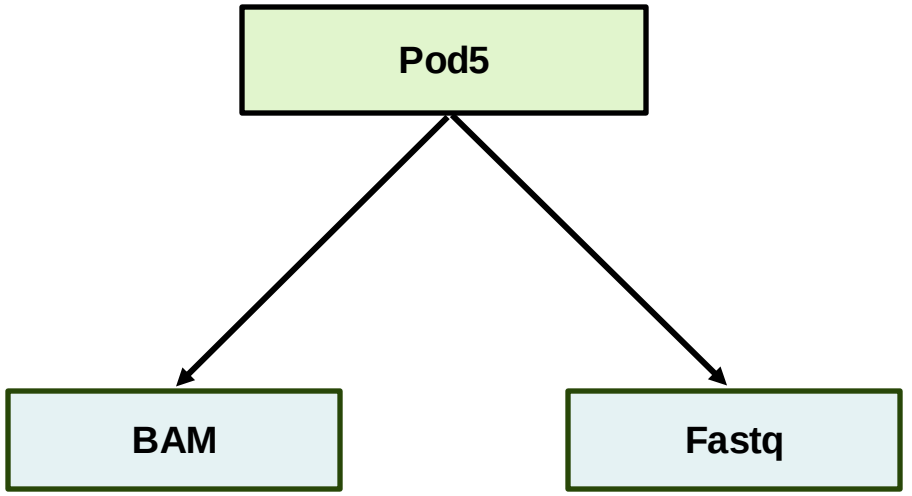

```
@3d9bb75a-f130-459f-9eee-d2998a98bbca runid=ec82a2362d33965586054b536eb745225170338f read=104298 ch=1794 start_time=2023-10-12T08:57:49.470672+01:00 flow_cell_id=PAS17198 protocol_group_id=20231011_B16959_P_1 sample_id=B16959_P1 barcode=barcode21 barcode_alias=barcode21 parent_read_id=3d9bb75a-f130-459f-9eee-d2998a98bbca basecall_model_version_id=2021-05-05_dna_r9.4.1_promethion_384_dd219f32
AAGCAGTGGTATCAACGCAGAGTACATGGGGGCCGTGAGACAGCAAGTGTCGGATCCGCAGGCGCAGCCGTGCGATGTTGTCCTCTACAGCCATGTATTCGGCTCCTGGCAGAGACTTGGGGATGGAACCACCTGAAACCGCGGGCCCTTTGCAGCTGCGATTTTCGCCCTACGTTTGCTGGAGGTACTATACTAGCAATTGCTGGAGAAGATTTTGCAATTGTTGCTTCTGATACTCGATTGAGTGAAGGGTTTTCAATTCATACGCGGGATAGCCCCAAATGTTACAAATTAACAGACAAAACAGTCATTGGATGCAGCGGTTTTCATGGAGACTGTCTTACGCTGACAAAGATTATTGAAGCAAGACTAAAGATGCCAGCATTCCAATAATAAGGCCATGACTACAGGGGCAATTGCTGCAATGCTGTCTACAATCCTGTATTCAAGGCGCTTCTTTCCATACTATGTTTACAACATCTTCGGTGGACTTGATGAAGAAGGAAAGGGGCTGTATACAGCTTTGATCAATGGGGTCTTACCAGAGAGACTCCTTCAAGGCTGGAGGCTCAGCAAGTGCCATGCTACAGCCCCTGCTTGACAACCAGGTTGGTTTTAAGAACATGCAGAATGTGGAGCATGTTCCGCTGTCCTTGGACAGAGCCATACGGCTGGTGAAAGATGTCTTCATTTCTGCGGCTGAGAGAGATGTGTACACTGGGGACACTACTCCGGATCTGCATAGTGACCAAAGAGGGCATCAGGAGGAAACTGTTCCTTAAGGAAGGACTGATCGTATTGCTCTTATCACCAATCAGTTCAGACCTGGTTGATTTTGTACTTTGGAACTGTACCTTGGATGGTTTTGTTTGGTAAAGAAAACCTGAACTAAAAAAAAAAAAAAAAAAAAAAAAACTCGATATCCATTGATTTCTGGAAATACAGATCGGAAGAGCGTCGTGTAG
+
JECB?@@A?>???DAA>>@ABA9<=>C@?@DB?<//00577('''556BD?98;9:::;=88B---.=?<<<<AA=?ABABC<<<?AA>;;;=CDBDDC@??>??426E@=203522:<><DDAA?>;;;+)+)***++,,;<;;<>;<<1***)/35>@ACKDKD;:8;>>=740*('''(0883=>>??//.+,,-A@A<CCF@?<<A<BBCFEJEEDGF9888PBDN::?99:<=<=>AEBCCD?BB=98:<=A;:;=<>@>>;9444:==****//7=>?@@@ACDKFEDE?6669<{FE?;?>=55557<=>DFBA7666?58;?ADD=>;<<<@?;:8787,,5=?<=?>?===::00002=@??A/),))(&%%%(,,@;.7:>?97789@??@ABB8555300420001334===?BBD@@BAAC@ACB=A?:>>>>>=@?9865577965679::>;;;<>>@>=======><;2210289:=:::;>@BBH?@?99269;998777:::9::<:9678:9+*//./1:;;<<=<<<<{{{{09::;>?@<;=9;=>?=<:::;<A?<?>=<;<10.-/023369:<:<9B@@@@>====>?=<<=>@BEA?BB?=;:::;6/,.1=>=<<==<==>A74447 8:,+++.:;<9888;<;('''++,---.937<<;<;<>??>=??>10//24=?>{{{{{8<>><;;<<>@??@@>..-)''/3**-46999:9888<<=>>@A79>::<::9;/.-,.135689<=/;992//11=8;99-)*/3''&&&&)4:;>@>>=>>@?>=9899=?==;:;;=<>>??CHEEA???AA<;;<<<<?><<=?>>>;:::?E=:90('')+.7-,.0><5::::1149<AG{H{GGCA@=;<<=>>AA>:5/01389=@?77777@@A>::56:>@=><:<=;;:<@;D>;998789;==
@55b6bc89-35d8-4200-acae-60c27b5dda43 runid=ec82a2362d33965586054b536eb745225170338f read=96442 ch=1795 start_time=2023-10-12T08:57:47.470672+01:00 flow_cell_id=PAS17198 protocol_group_id=20231011_B16959_P_1 sample_id=B16959_P1 barcode=barcode21 barcode_alias=barcode21 parent_read_id=55b6bc89-35d8-4200-acae-60c27b5dda43 basecall_model_version_id=2021-05-05_dna_r9.4.1_promethion_384_dd219f32
AAGCAGTGGTATCAACGCAGAGTACATGGGGCTTCCTGTCTGTACCAGGGCGGCGCGTGGTCACGCCGAGTGACAGAGACGCTCAGGCTGTGTTCTCAGGATGACCGAGTGGGAGACGGCAGCACCAGCGGTGGCAGAGACCCCGGACATCAAGCTCTTTGAAGTGGAGCACCGATGATGTGCAGATCAATGACATTTCCCTGCAGGATTACATTGCAGTGAAGGAGAAGTATGCCAAGTACCTGCCTCACAGTGCAGGGCGGTATGCCGCCAAACGCTTCCGCAAAGCTCAGTGTCCCATTGTGGAGCGCCTCACTAACTCCATGATGATGCACGGCTTTAACAACGGCAAGAAGCTCATGACTGTGCGCATCGTCAAGCATGCCTTCGAGATCATACACCTGCTCACAGGCGAGAACCCTCTGCAGGTCCTGGTGAACGCCATCATCAACAGTGGTCCCGGGAGGACTCCACACGCATTGGGCGCGCCGGGACTGTGAGACATTTGTGGATGTGTCCCCCCTGCGCCGTGTGAACCAGGCCATCTGGCTGCTGTGCACAGGCGCTCGTGAAACTGCCTTCCGGAACATTAAGACCATTGCTGAGTGCCTGGCAGATGAGCTCCTCAATGCTGCCAAGGGCTCCTCGAACTCCTATGCCATTAAGAAGAAGGACGAGCTGGAGCGTGTGGCCAAGTCCAACCGCTGATTTTTCCCAGCTGCTGCCCAATAAAACCTGTCTGCCCTTTGGGGCAGTCCCAGCCAAAAAAAAAAAAAAAA
```


https://www.researchgate.net/publication/340832159_The_third_generation_sequencing_The_advanced_approach_to_genetic_diseases

Basecalling

Dorado

- Converting signals into sequences
- Deep learning algorithms
- Several basecalling models:
  - fast,
  - high accuracy (HAC),
  - super high accuracy (SUP)



https://github.com/nanoporetech/dorado

Basecalling

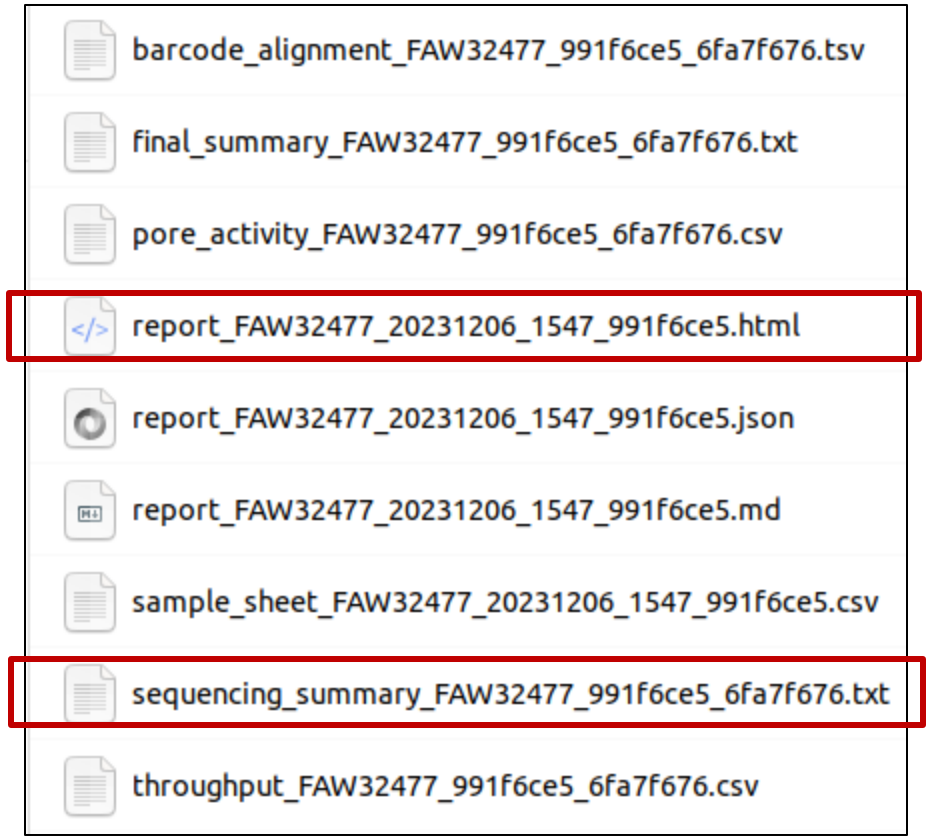
Basecalling models

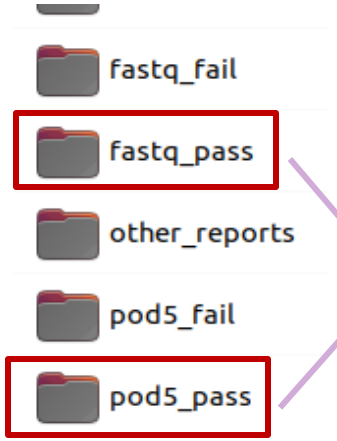
Model	Speed	Accuracy	Application
Fast	Very fast	Average accuracy	Real-time sequencing, fast applications (clinical..)
High Accuracy (HAC)	Medium	High	Genomic studies, analysis of variants
Super Accuracy (Sup)	Slow	Very high	Metagenomics, applications requiring maximum precision

https://github.com/nanoporetech/dorado

Slido

Nanopore run outputs





Pass files contain reads with a higher quality score than the one specified at the start of the run.

sequencing summary file: necessary for basecall and quality control tools

Quality control - Nanopore report

Reads with a quality score higher than 10, will be in the pass folder, and those lower than the threshold will be in the fail folder.

Quality control - Nanopore report

Sequence output

READ LENGTHS · OUTLIERS REMOVED

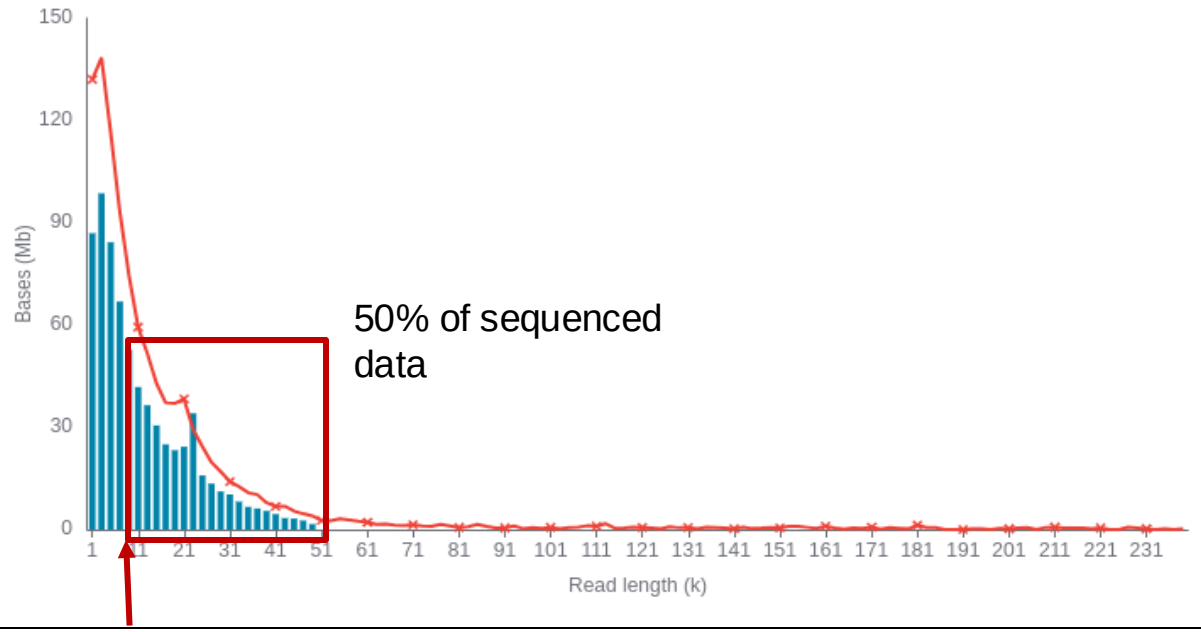
The read length graph shows the total number of bases vs the read length. The longest 1% of strands are classified as outliers, and excluded to allow focus on the main body of data.

Legend

Basecalled Estimated

Estimated N50: **8.64 kb**

% Basecalled: **100%**



50% of sequenced data

N50

OUTLIERS

The longest 1% of strands are classified as outliers, and aggregated into groups to show their relative amounts.

Read length (kb)	Aggregated reads (Mb)
50 - 66	5.46
66 - 82	1.32
82 - 95	0.27

N50: shortest read length such that 50% of the total length is made up of sequences that are at least that long.

Visualising reads/runs quality

- Several tools :
  - Nanoplot
  - PyqoQC
  - MinionQC
  - ToulligQC
  - ...

roblanf/**minion_qc**

Quality control for MinION sequencing data

7 Contributors | 15 Issues | 208 Stars | 42 Forks

https://github.com/roblanf/minion_qc

toulligQC Public

https://github.com/GenomiqueENS/toulligQC

wdecoster/
NanoPlot

Plotting scripts for long read sequencing data

9 Contributors | 30 Used by | 393 Stars | 48 Forks

https://github.com/wdecoster/NanoPlot

https://github.com/a-slide/pycoQC

Visualising reads/runs quality

```
pycoQC -f sequencing_summary.txt -a [bam_file / fastq_file] -o pycoQC_output.html
```

- Can be run with fastq or bam files
- Requires the complete sequencing summary
- Interactive plots

https://github.com/a-slide/pycoQC

PycoQC

General run summary

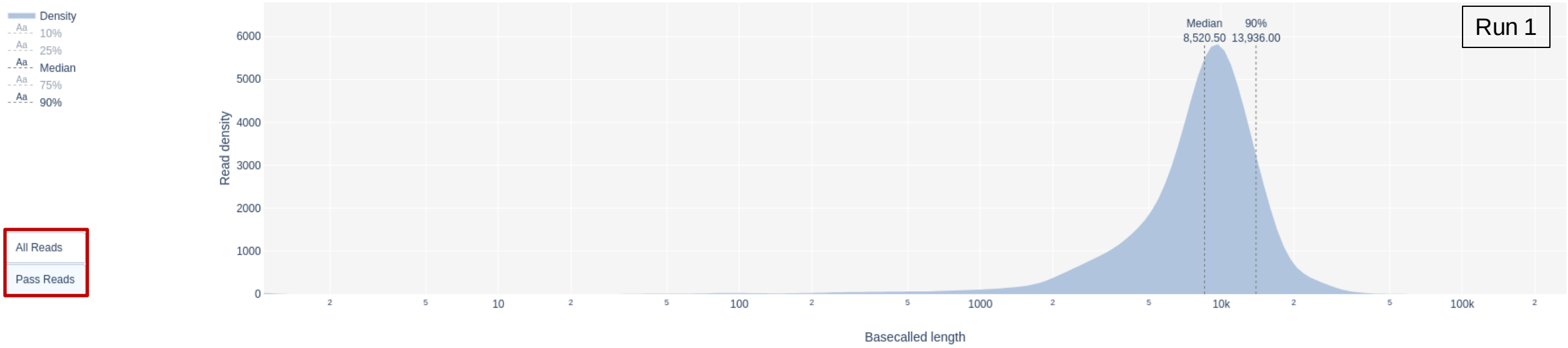
Status	Run Duration (h)	Active Channels	Number of Runids	Number of Barcodes
All Reads	72.00	502	1	0
Pass Reads	71.99	491	1	0

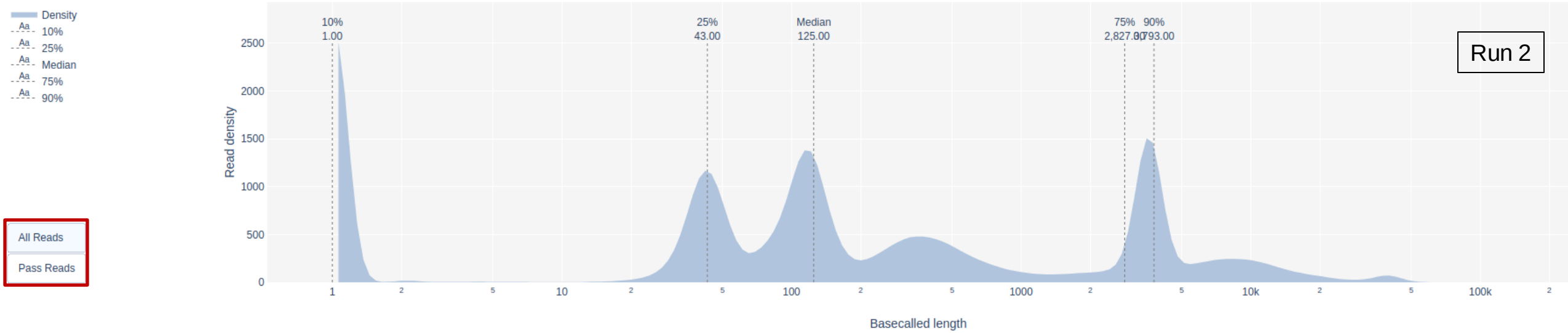
Basecall summary

Status	Reads	Bases	N50	Median Read Length	Median PHRED score
All Reads	1.858227e+7	1.074439e+10	531	507	4.987
Pass Reads	3.563872e+6	2.241761e+9	560	526	8.331

PycoQC

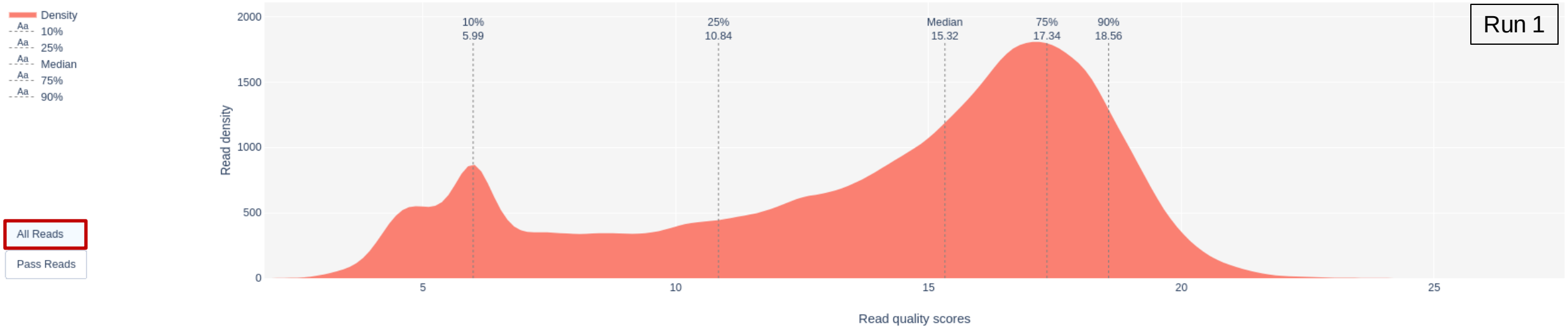
Basecalled read length

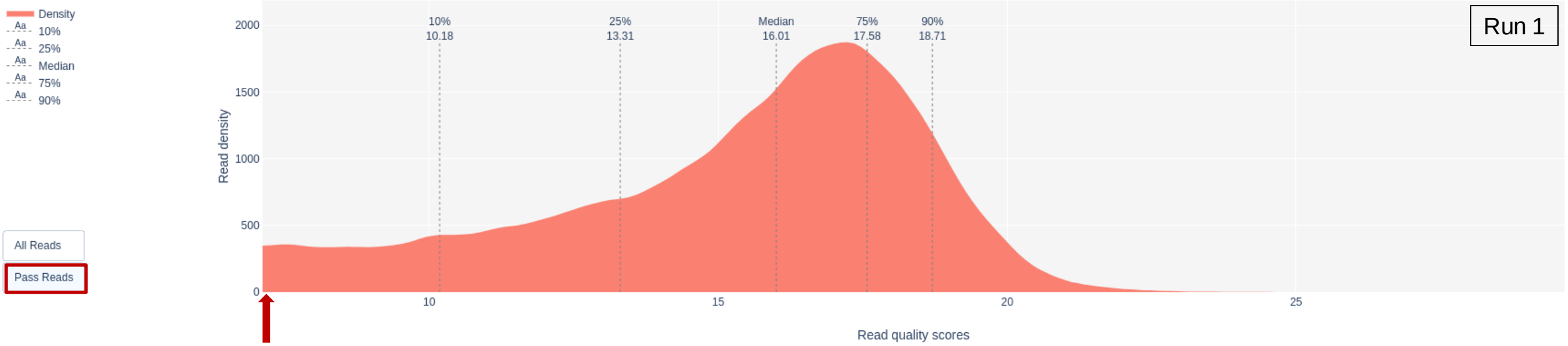




PycoQC

Basecalled reads PHRED quality

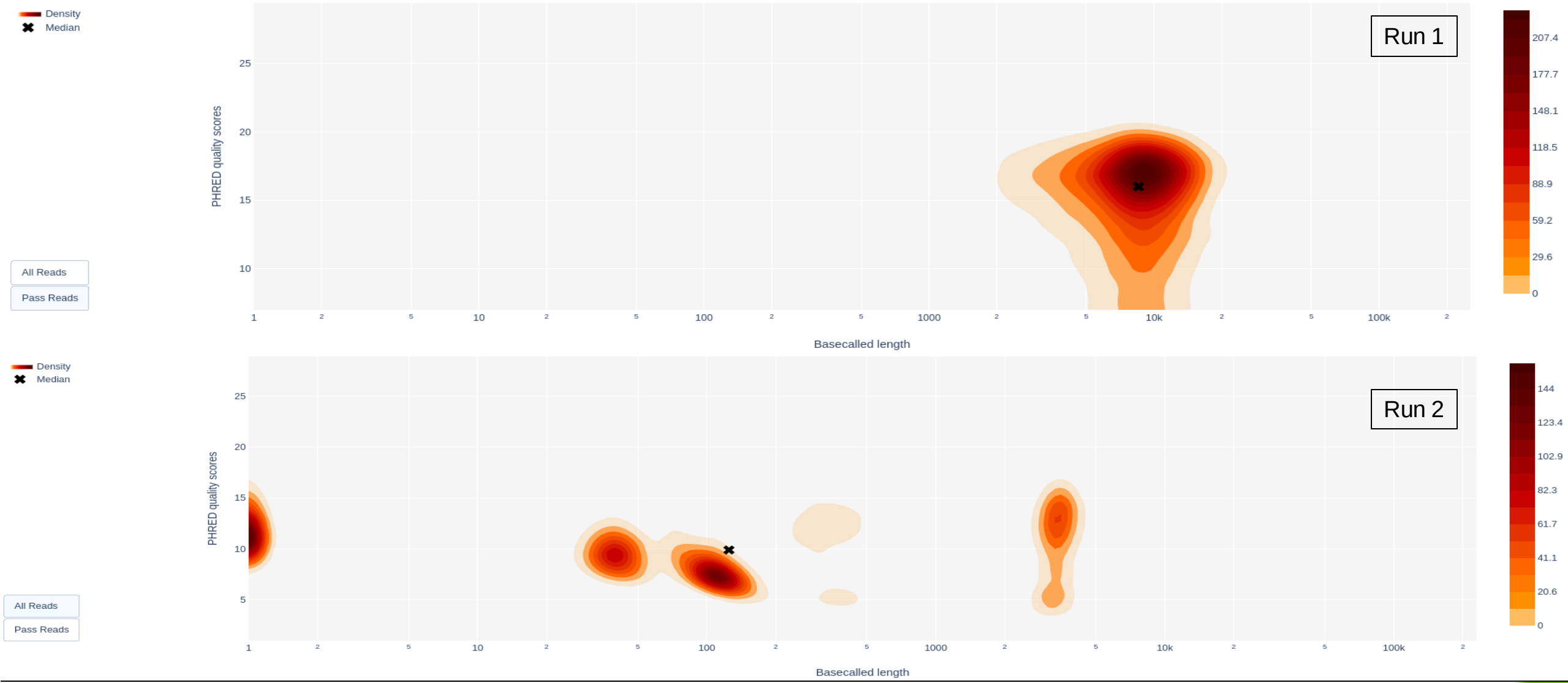




7

PycoQC

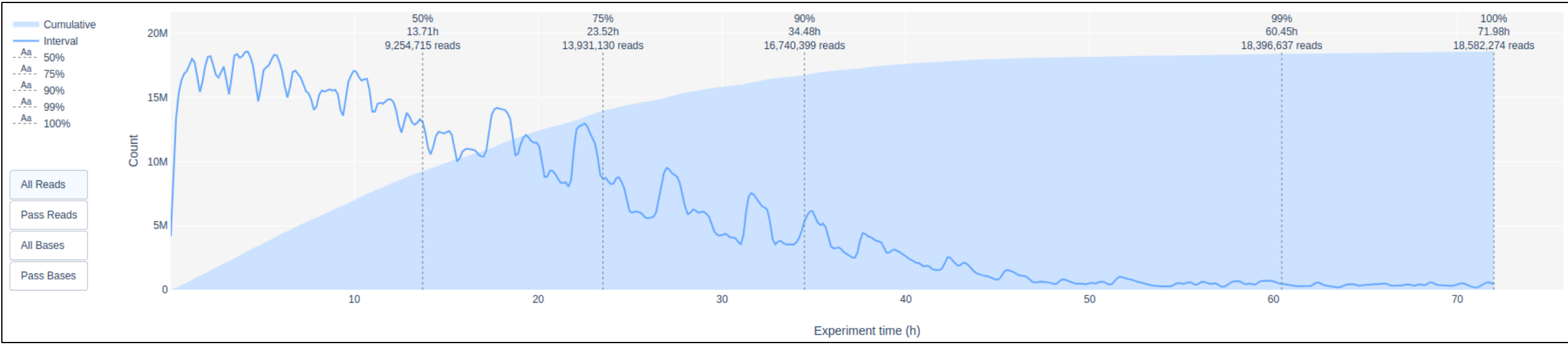
Basecalled read length vs reads PHRED quality



Slido

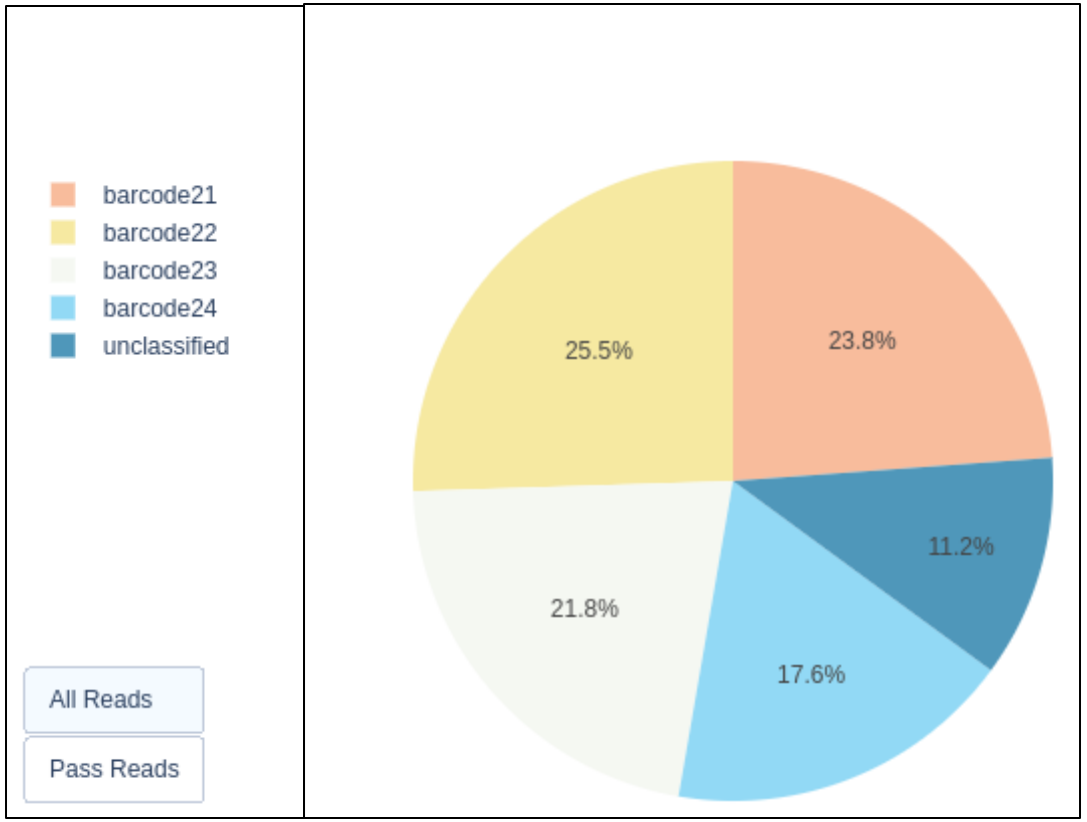
PycoQC

Output over experiment time



PycoQC

Distribution of reads across barcodes



Data filtering

- How do we filter our data?

Two major criteria:

Quality of reads

Length of reads

Data filtering

Quality

- Default threshold when run is launched (can be modified): 7
- Can be more stringent

Length

- A default threshold can be defined when the run is launched
- Depends on the type of analyses and manipulations carried out beforehand
- Ampliseq: short fragments

Data filtering tools

Nanofilt

```
NanoFilt -l 500 -q 10 reads.fastq > reads_filtered.fastq
```

https://github.com/wdecoster/nanofilt

filtlong

```
filtlong --min_length 1000 --min_mean_q 12 reads.fastq > filtered.fastq
```

https://github.com/rrwick/Filtlong

seqtk

```
seqtk seq -L 1000 input.fastq > output_min1000.fastq
```

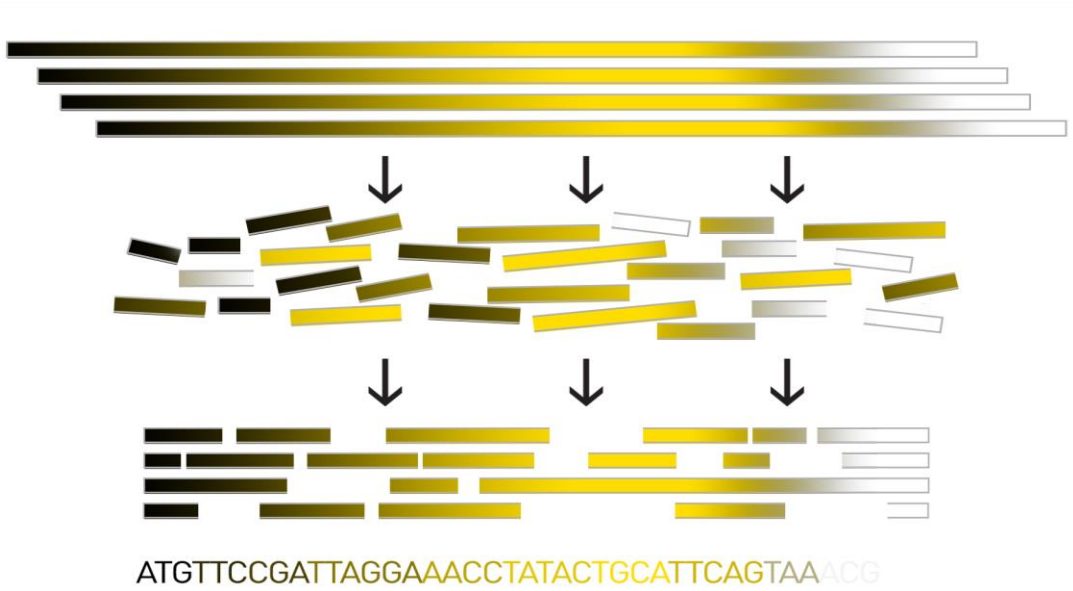
https://github.com/lh3/seqtk

General tool for fatsq/fasta filtering based on length, or subsampling ...

Slido

Genome assembly with Nanopore data

What is assembly ?



https://www.hudsonalpha.org/sequencing-from-scratch-reference-genomes-and-de-novo-sequence-assembly/

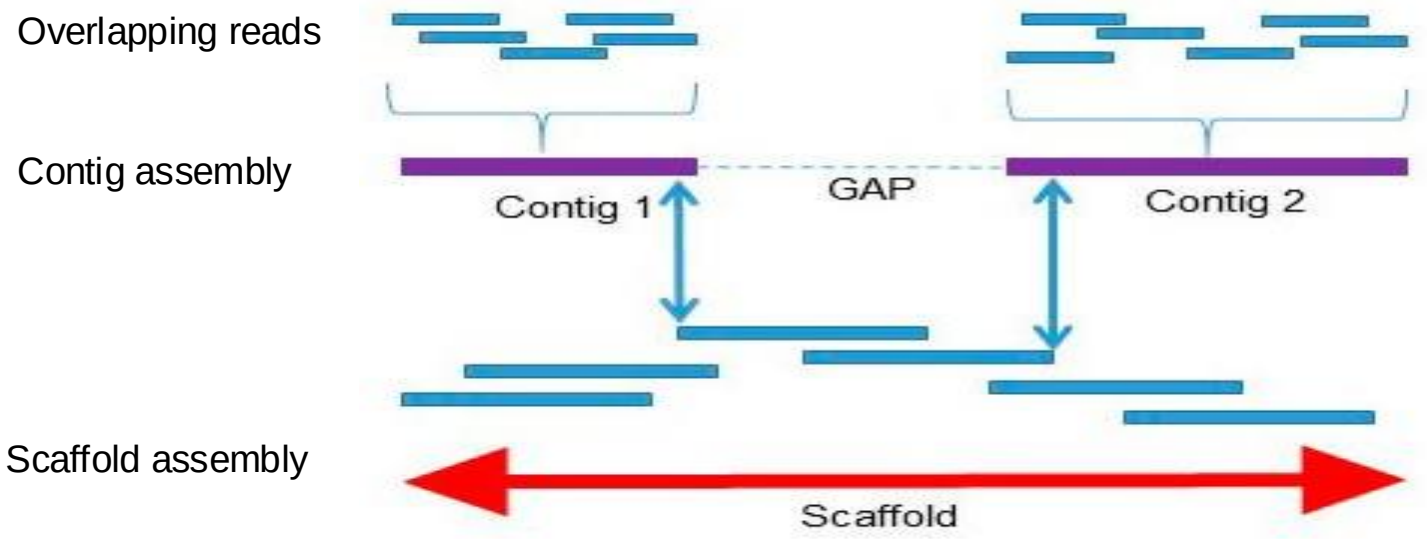
Sample: DNA extraction

Library preparation and sequencing

Assembly

- **Genome assembly** is the construction of a consensus sequence from scratch (reads) by finding overlapping regions.

Contigs, scaffolds definition



https://www.france-genomique.org/expertises-technologiques/genome-entier/

- **Contig** is a set of DNA sequences (reads) that overlap in a way that provides a continuous representation of a genomic region with no gaps in between.
- **Scaffold** is composed of contigs linked together by gaps represented by N in the final sequence.

Assembly limitations

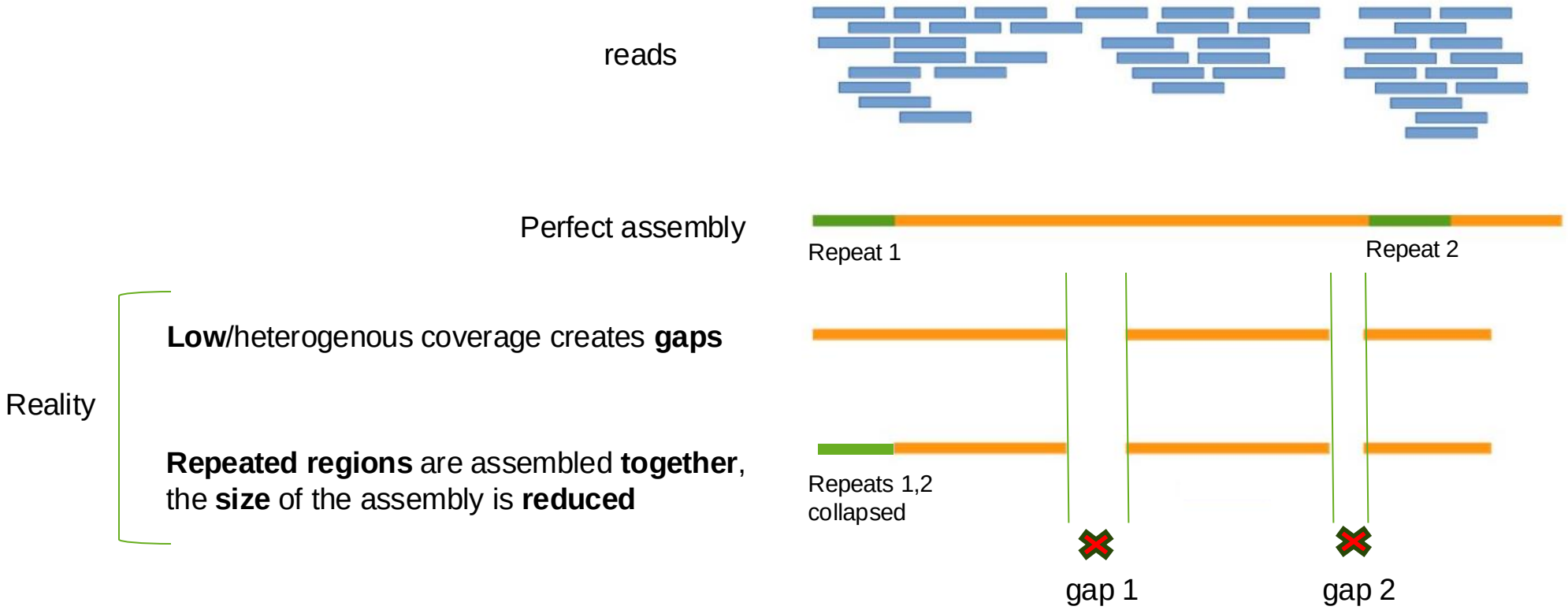
Millions of reads

Final assembly

- need to find overlapping words to assemble two parts of the paper (overlapping reads)
- repeats (same text written more than once in the page) --> very long reads
- erased words or misspelling errors (handwritten) (= sequencing errors) --> reads correction + high coverage
- some parts are missing (= low coverage) --> high coverage

Assembly limitations

Most of cases (depending on genome complexity)

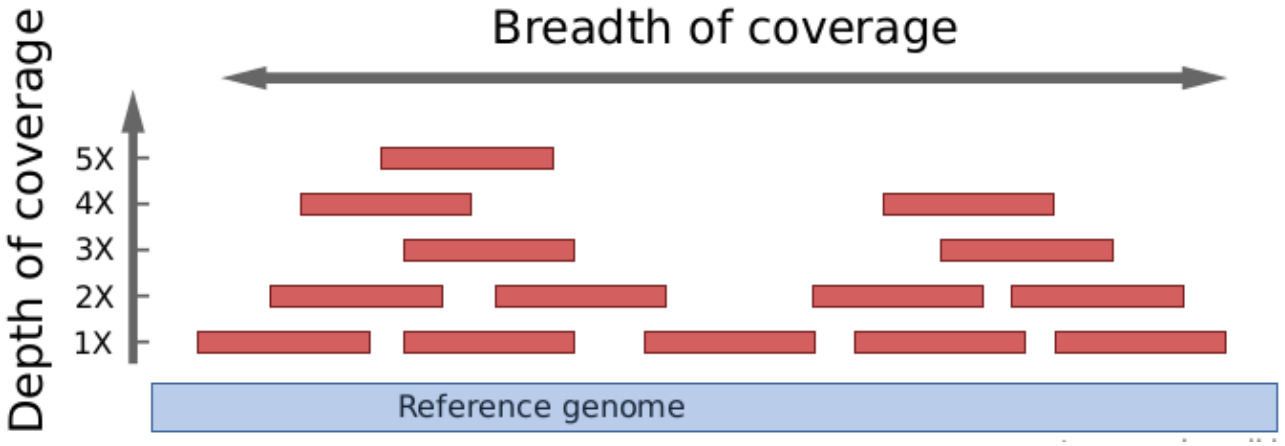


Nanopore data advantages

Resolving assembly errors with Nanopore data

- Long reads (up to 1Mb) --> solve repetitions problems
- Decrease % of errors thanks to the last versions of basecalling and the new flowcells.
- Highest yield with last flowcells (Promethion) --> solve low coverage gaps

Sequencing depth vs coverage



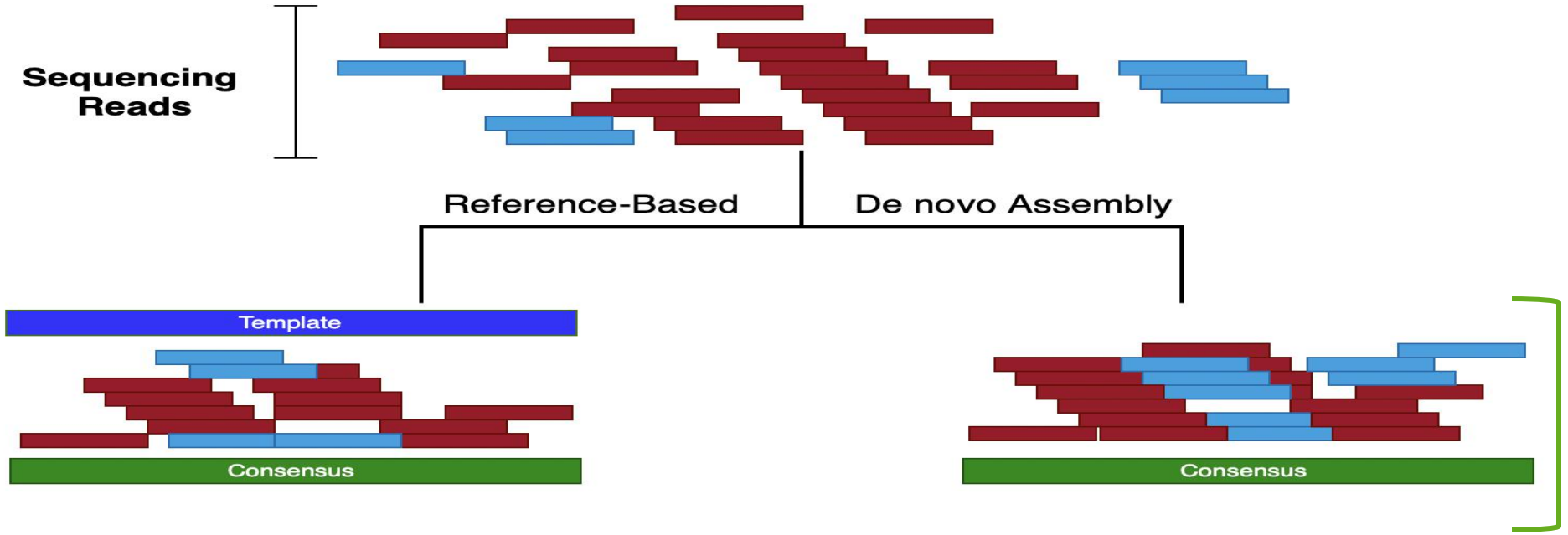
- **Depth of coverage** (how many times each base is sequenced)
- **Breadth of coverage** (the percentage of the genome that is covered by reads)

https://bioinformatics.stackexchange.com/questions/2726/what-is-deep-sequencing

Assembly types

Two types of assembly :

- Denovo assembly (exploratory approach)
- Reference based assembly (existing reference)



https://en.wikipedia.org/wiki/Sequence_assembly

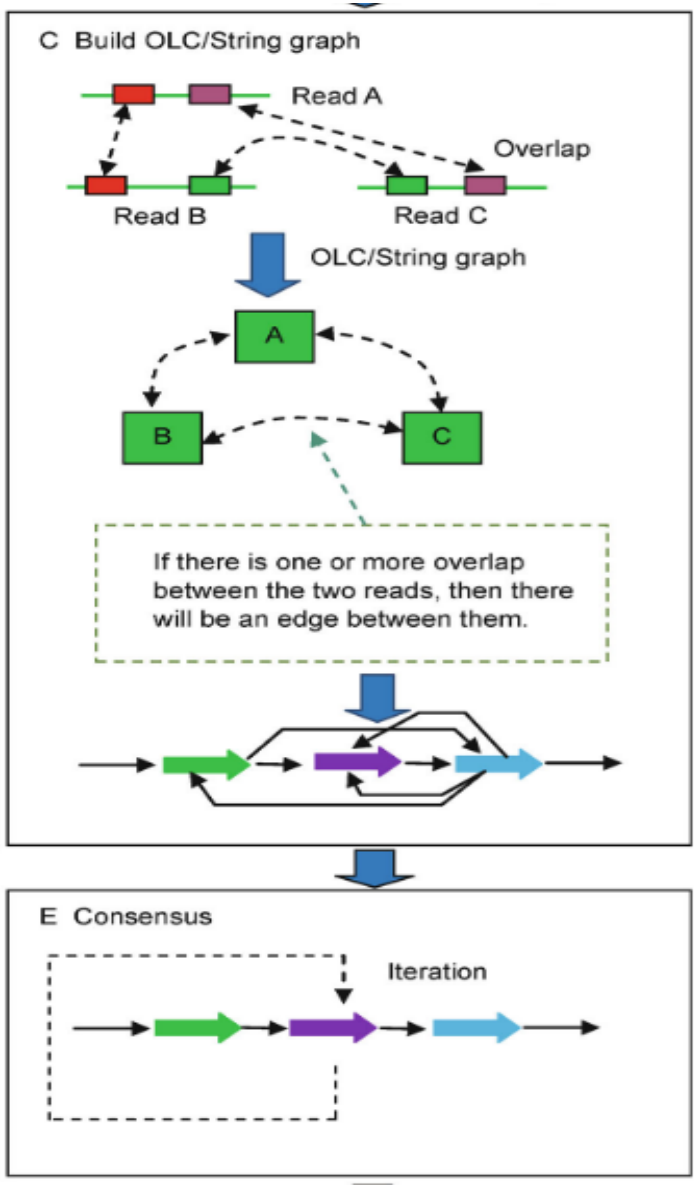
Assembly algorithms

Several genome assembly algorithms have been used :

- OLC (Overlap Layout Consensus)
- DBG (de Bruijn graph)
- string graph
- repeat graph
- ...

The most used ones are **OLC** and **DBG** (for short reads).

Overlap Layout Consensus (OLC)



Overlap discovery between all reads

Reads are compared pairwise to find overlaps.

Layout (Building a Graph)

A **graph** is constructed where:

- **Nodes** represent sequencing reads
- **Edges** represent the overlaps between them

Consensus

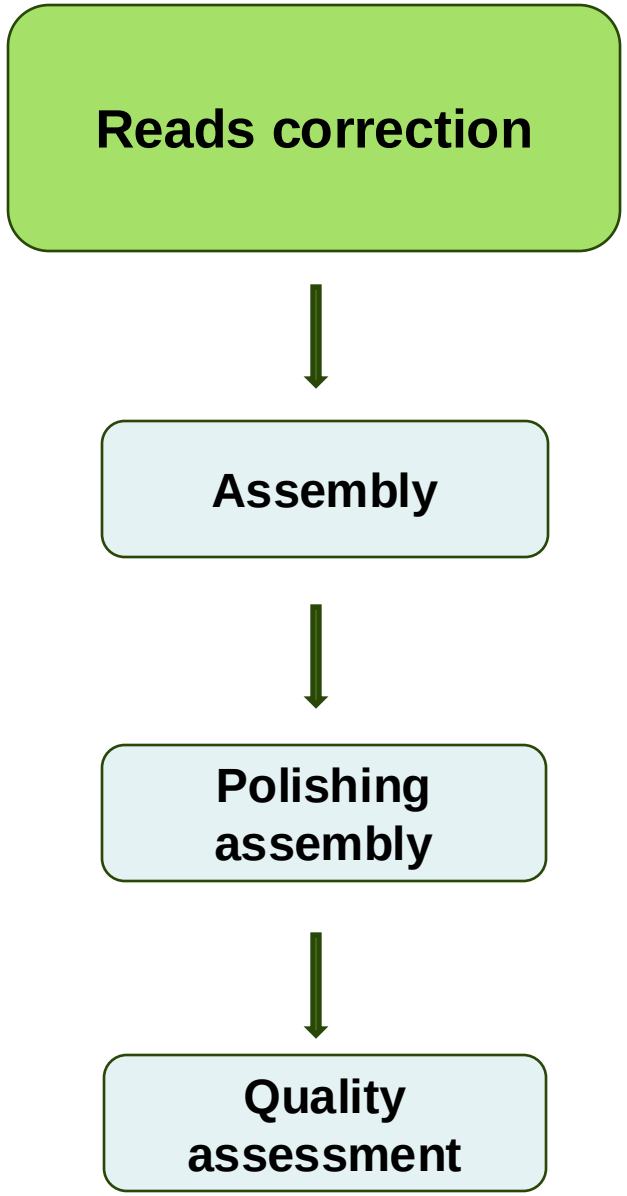
The algorithm derives a **final consensus sequence**, assembling the genome.

* where one node is visited only once (Hamiltonian path).

https://link.springer.com/content/pdf/10.1007/s40484-019-0166-9.pdf

Slido

Assembly steps



Before assembly: read correction

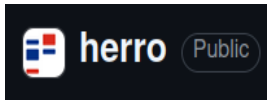
- Read correction helps improve assembly quality by filtering and refining sequencing data.
- Canu and NECAT, include a built-in read correction step before performing the actual assembly

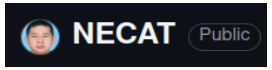
Dorado correct

- Keeps only **reads > 10kb**

➡ Dataset of very long reads

```
dorado correct {raw_reads} > {output.fasta}
```

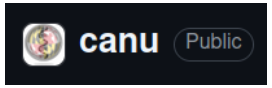
Uses Herro model 



Necat: error correction and de-novo assembly tool for Nanopore long noisy reads.

*Requires a config file (see documentation)
*Only corrects longest 40X

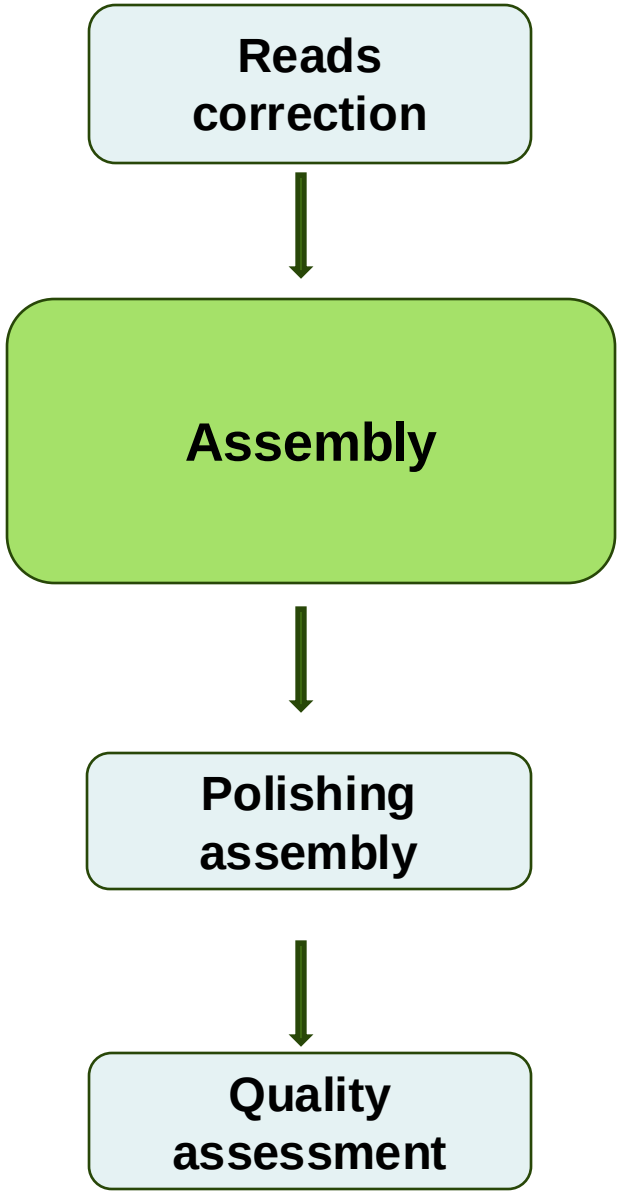
```
necat.pl correct {config.txt}
```



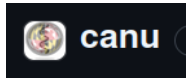
Canu: assembly + read correction

```
canu -correct -p {prefix} -d {output_dir}  genomeSize=4.8m -nanopore {fastq}
```

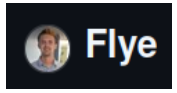
Assembly steps



Commonly used assemblers

- Canu 

https://github.com/marbl/canu

- Flye 

https://github.com/mikolmogorov/Flye

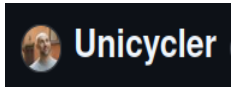
- Nextdenovo 

https://github.com/Nextomics/NextDenovo

- Necat 

https://github.com/xiaochuanle/NECAT

…

- Unicycler (prokaryotes) 

https://github.com/rrwick/Unicycler

- Autocycler (prokaryotes) 

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

Canu

```
canu  -p {output_prefix} -d {output_directory} genomeSize=4.8m maxInputCoverage=100  -nanopore reads.fastq
```

- Supports **Oxford Nanopore & PacBio** long reads
- Can assemble **eukaryotic and prokaryotic genomes**
- Uses an **OLC graph**
- Three steps:
  - **Correction**
  - **Trimming**
  - **Assembly**
- Input: **fastq/fasta** files
- Assembly is not deterministic, different assemblies with the same parameters and data.
- **Limitation**: can be very slow, does not include circularization,

Flye

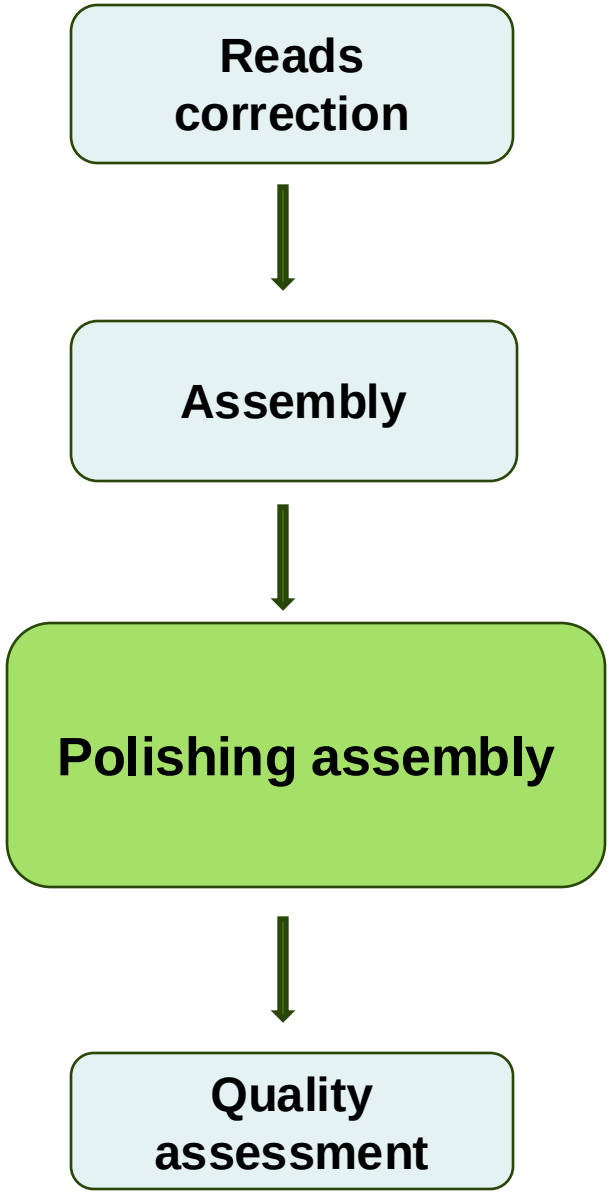
```
flye --nano-raw {reads.fastq.gz} --out-dir {output_directory}
```

- Supports **Oxford Nanopore & PacBio** long reads
- Can assemble **eukaryotic and prokaryotic genomes**
- Uses a **repeat graph** (allow approximate sequence matches) for assembly
- Special mode for **metagenomes** (handles uneven coverage)
- **Designed for raw reads** (tolerates high sequencing noise)
- Include **circularization** (bacterial genomes)
- **Limitation**: May fail with **very high coverage** (>1000x) ??

Nanopore reads options

Option	Error rate	Data type
--nano-raw	< 20%	Raw reads
--nano-corr	< 3%	Corrected reads (ex: canu)
--nano-hq	< 5%	High quality reads (Q20)

Assembly steps



Polishing assembly

Polishing an assembly involves mapping long or short reads to the assembled sequence(s) to correct errors.

Several tools are available:

- **Racon**: uses longs and short reads, adapted for substitution errors and small indels <7bp

https://github.com/lbcb-sci/racon

  - Input: fasta file (assembly), fastq file (reads), alignement file (sam)
- **Pilon**: uses Illumina reads (in case of hybrid assembly)

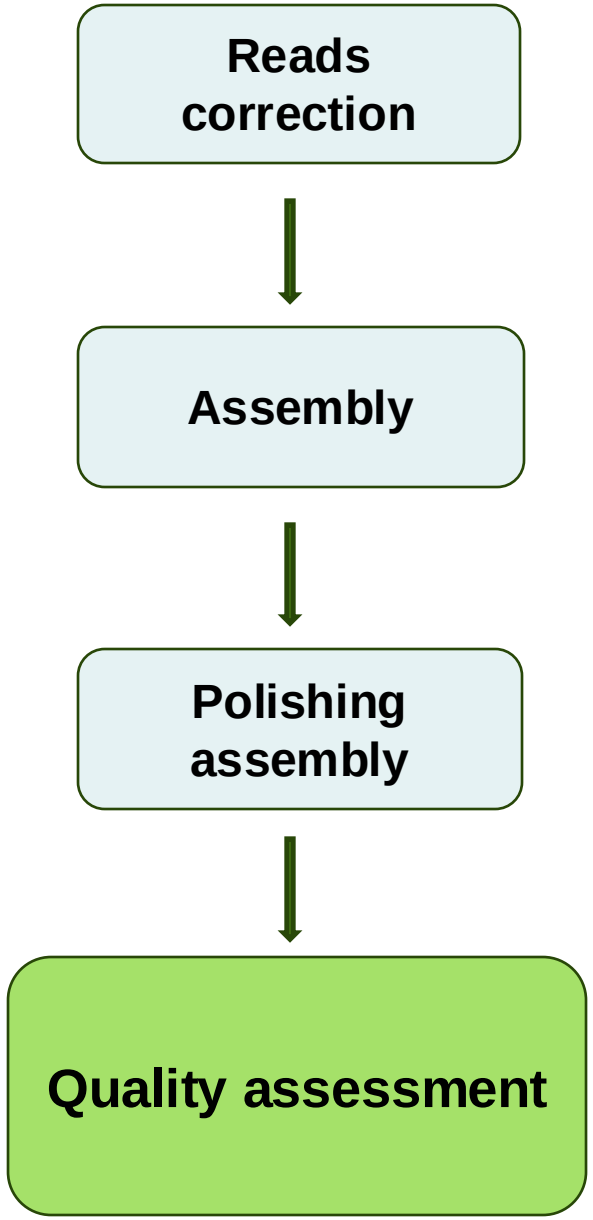
https://github.com/broadinstitute/pilon/wiki

  - Input: fasta file (assembly), bam file (reads mapped to the assembly)
- **Flye**: uses long reads

https://github.com/mikolmogorov/Flye

- ...

Assembly steps



Assembly quality

Assembly quality, **Quast** (QUality ASsessment Tool)

https://github.com/ablab/quast

- provides assembly statistics:
  - number of contigs, their length, the NG50 and the GC% ...

```
quast {assembly.fasta} -o {output_directory}
```

Evaluate the completeness of the assembly:

- **Busco (Benchmarking Universal Single-Copy Orthologs)**

BUSCO compares the assembled genome against a reference database containing conserved orthologous gene groups. https://busco.ezlab.org/busco_userguide.html#running-busco

```
busco -i {assembly.fasta} -l bacteria_odb10 -o {busco_output} -m genome
# -m BUSCO analysis mode to run. Can be 'genome', 'protein' or 'transcriptome'.
# -l Specify the BUSCO lineage dataset to be used for scoring

busco --list-datasets
```

Databases can be downloaded in advance or automatically

Busco

- For example, if working with E.coli, we can use a general bacterial database **bacteria**, or a more specific one, such as **Enterobacteria**.
- A **high-quality assembly** should ideally show **100% completeness**, meaning all expected conserved genes are present.

```
C:89.0%[S:85.8%,D:3.2%],F:6.9%,M:4.1%,n:3023
```

BUSCO generates a report with the percentages of genes found, categorized into four groups:

- ✅ **Complete BUSCOs (C):** Genes fully recovered (ideally close to 100%).
- 🟡 **Fragmented BUSCOs (F):** Genes partially present in the assembly.
- 🔴 **Missing BUSCOs (M):** Genes absent.
- 🔄 **Duplicated BUSCOs (D):** Genes found in multiple copies.

CheckM

- Place the genome on a phylogenetic tree
- Identify sets of marker genes that should be present in the phylogenetic group
- Search for the genes in the assembly

Outputs:

- **Completeness**: estimate of the fraction of genes that are expected to be ubiquitous and single-copy within a phylogenetic lineage
- **Contamination**: identifying the number of single copy genes, that should only be there once.
- **Strain heterogeneity (SH) index:** indicates the proportion of the contamination that appears to be from the same or similar strains

https://github.com/Ecogenomics/CheckM/wiki

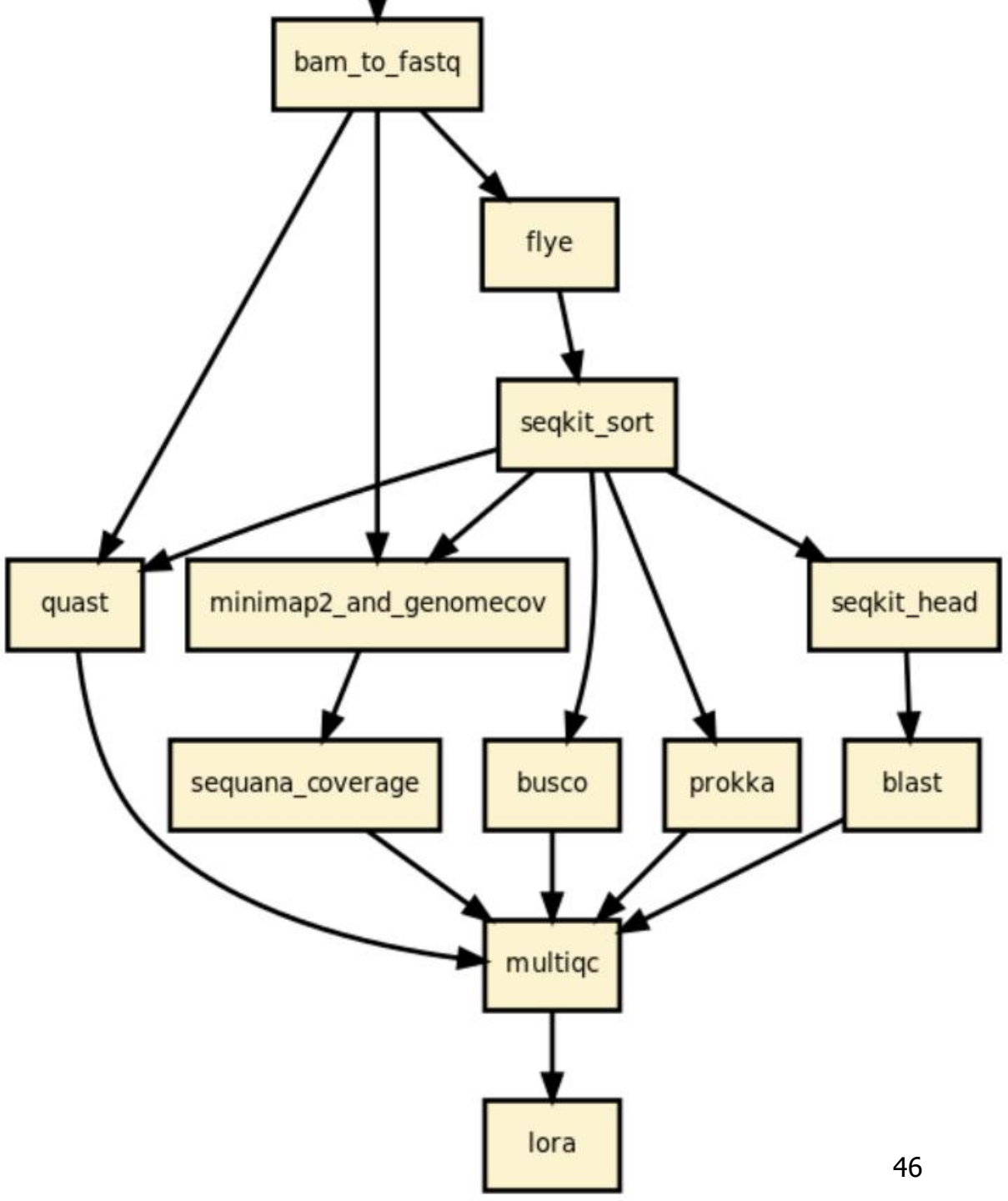
Lora pipeline

Combines all the previous steps in only one pipeline

https://github.com/sequana/lora/tree/main

```
sequana_lora --assembler canu --pacbio --input-directory
{fastq} --input-pattern *.fastq.gz --mode {bacteria} --
busco-lineage {bacteria_odb10} --blastdb {path to blast db}
--working-directory {canu_output} --use-apptainer –
apptainer-prefix {folder_path_for_appatiners}
```

- Several assemblers are available within the pipeline
- All tools are available within containers with Damona –> no manual installation required
  https://github.com/cokelaer/damona/
- Easy to use
- Output all the necessary steps to a good assembly: from read correction to assembly evaluation



Lora pipeline

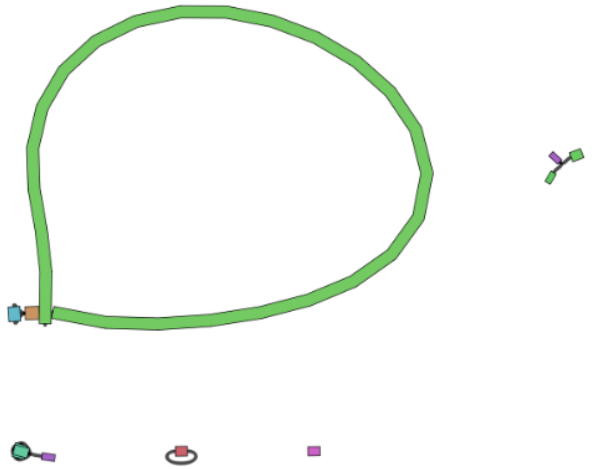
General Information

The following table gives you basic statistics about the final assembly. Please click on the sample of interest in the first column to switch from one sample to the other. The default page shows the first sample.

Sample	# contigs	Largest contig	Total length	Mapped (%)	Avg. coverage depth	Complete BUSCOs	Complete and duplicated BUSCOs	Fragmented BUSCOs	Missing BUSCOs
nCCUG4856T	7	5124147	5272023	100.0	55	57	0	48	19

nCCUG4856T sample:

Download contig file for nCCUG4856T: here.

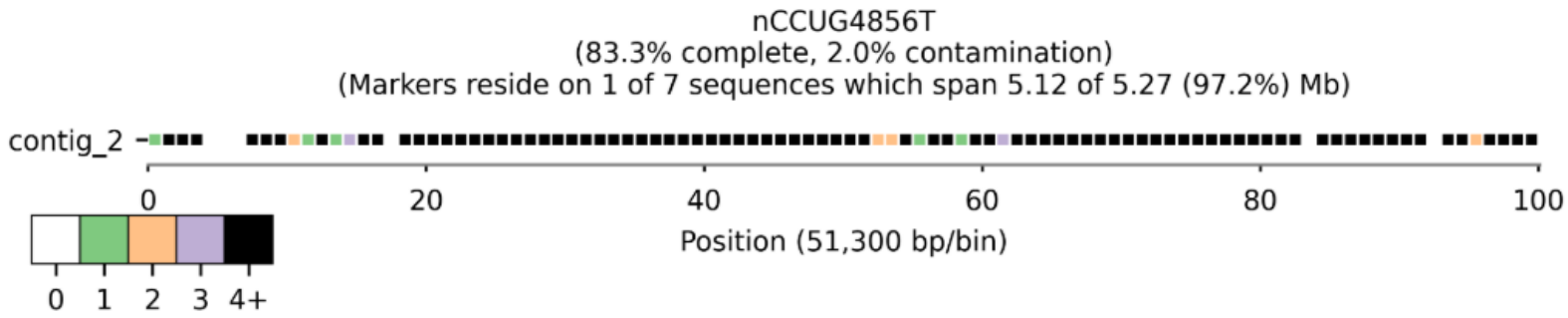


https://github.com/rrwick/Bandage/wiki

Lora pipeline

General CheckM results using species / Bacteroides fragilis

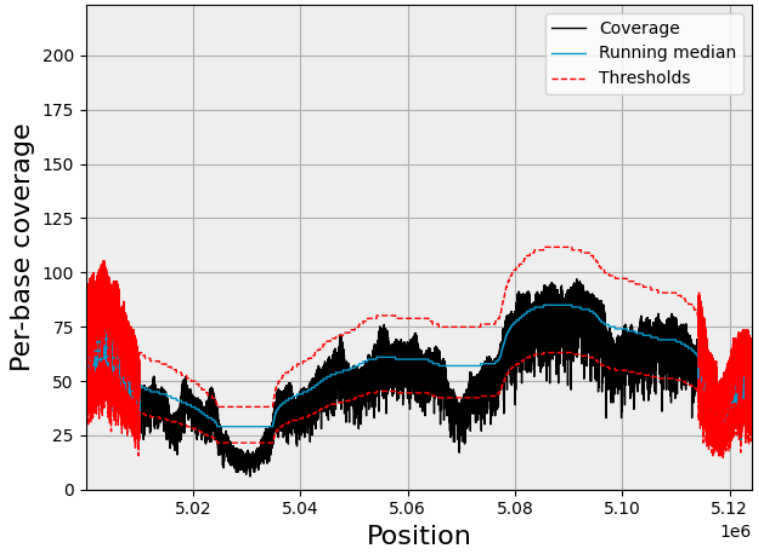
Completeness	83.34
Contamination	2.04
Heterogenity	0.00



Lora pipeline

Sequana coverage for contig_2

length	10248294
DOC	54.588
BOC	100.000



Blast results for contig_2

sseqid	pident	length	mismatch	gapopen	qstart	qend	sstart	send	evalue	bitscore	staxids	stitle
gi|1663148248|gb|CP036555.1|	99.665	1891581	321	5581	3237617	5123414	777664	2669006	0.0	3452000.0	817	Bacteroides fragilis strain CCUG4856T chromosome, complete genome
gi|60491031|emb|CR626927.1|	99.662	1733764	302	5155	3395714	5124147	1	1733536	0.0	3164000.0	272559	Bacteroides fragilis NCTC 9343, complete genome
gi|1042799677|gb|CP012706.1|	99.189	392181	1809	1184	1465266	1856156	2402456	2794555	0.0	705300.0	817	Bacteroides fragilis strain S14, complete genome
gi|52214156|dbj|AP006841.1|	99.097	199290	1146	601	4309505	4508195	977282	1176517	0.0	357400.0	295405	Bacteroides fragilis YCH46 DNA, complete genome
gi|794204258|gb|CP011073.1|	99.113	190106	1066	546	4465384	4654928	1035111	1225157	0.0	341100.0	817	Bacteroides fragilis strain BOB25, complete genome

Nanopore assembly strategy

- **Define the genome's characteristics:** heterozygosity, genome size, repeat content...
- **Use different assemblers** and **compare results**
- Choose the adapted assembler to your data
- **Test** assembly with **several depths**:
  - low depth results in small contigs (fragmented assembly),
  - high depth can impact assembly (e.g: Flye)
- **Test** assembly with several **read lengths** (e.g: reads > 10kb)
- **Final assembly size ~ genome size**

In summary

List of learning points in this session:

- Carry out a quality control analysis of Oxford Nanopore data
- Filter Nanopore data based on the quality and read length
- Correct Nanopore reads before assembly
- Run a genome assembly
- Polish the assembly
- Assess the completeness and the quality of the assembled genome

Practical part

Data requirements:

► Microb Genom. 2019 Nov 7;5(11):e000312. doi: 10.1099/mgen.0.000312

Complete hybrid genome assembly of clinical multidrug-resistant *Bacteroides fragilis* isolates enables comprehensive identification of antimicrobial-resistance genes and plasmids

Thomas V Sydenham [1,2,3,*], Søren Overballe-Petersen [4], Henrik Hasman [4], Hannah Wexler [5], Michael Kemp [1,2], Ulrik S Justesen [1,2]

https://pmc.ncbi.nlm.nih.gov/articles/PMC6927303/#abstract1

- Fastq files (data summary section):

https://zenodo.org/records/2677927

Practical part

- Control the quality of your data (Nanoplot)
- Filter data if necessary
- Run Flye assembler
- Assess the quality of your assembly (Busco + Quast)

References

Liao X, et al. Current challenges and solutions of de novo assembly. *Quantitative Biology*. 2018.

Sun J, et al. Benchmarking Oxford Nanopore read assemblers for high-quality molluscan genomes. *Biological Sciences*. 2021.

Gavrielatos M, et al. Benchmarking of next and third generation sequencing technologies and their associated algorithms for de novo genome assembly. *Molecular Medicine Reports*. 2021.

Wick R. Benchmarking of long-read assemblers for prokaryote whole genome sequencing. *F1000Research*. 2021.

Links for tools:

- Canu: https://canu.readthedocs.io/en/latest/quick-start.html
- Flye: https://github.com/mikolmogorov/Flye
- Necat: https://github.com/xiaochuanle/NECAT
- Unicyler: https://github.com/rrwick/Unicycler?tab=readme-ov-file#introduction
- Autocycler: https://github.com/rrwick/Autocycler/wiki/Illustrated-pipeline-overview
- Pilon: https://github.com/broadinstitute/pilon/wiki
- Racon: https://github.com/lbcb-sci/racon
- Dorado: https://github.com/nanoporetech/dorado
- Lora: https://github.com/sequana/lora
- Quast: https://github.com/ablab/quast
- Busco: https://busco.ezlab.org/busco_userguide.html#running-busco
- CheckM: https://github.com/Ecogenomics/CheckM/wiki
- Nanoplot: https://github.com/wdecoster/NanoPlot

Acknowledgements

The creation of this training material was commissioned by ECDC to Institut Pasteur with the direct involvement of Rania Ouazahrou

Q&A session