Genome assembly

EBAII Assembly & Annotation - Roscoff Sept 2022

Delphine NAQUIN
Christophe KLOPP

What are you going to learn?

- What a genome assembly is.
- What a genome assembler is.
- Which assembly strategies can be used.
- Which are the most common genome assemblers used these days.
- Which data/assembler combination work.
- How to assemble a bacterial genome.
- How to perform eukaryote genome assembly with several software packages in Galaxy (TP).
- What are the most used parameters.

What is a genome assembly?

A genome assembly is a set of sequences, usually in **fasta format**, representing the genome content at the nucleotide level.

Today it is very rare to have a chromosomes in a single read. Therefore we assemble a given read **coverage** (nX) to generate a genome assembly.

Depending on the definition assembly builds **contigs** or **scaffolds** or **chromosomes** (pseudo-molecules). Here we will stick to contigs even if some software packages perform scaffolding. For **polyploid organisms** the assembler can output N contig sets : **haplotyped assemblies**.

Assembly sequence correction, called “**polishing**” will be presented in the next section.

Procaryote genome assembly

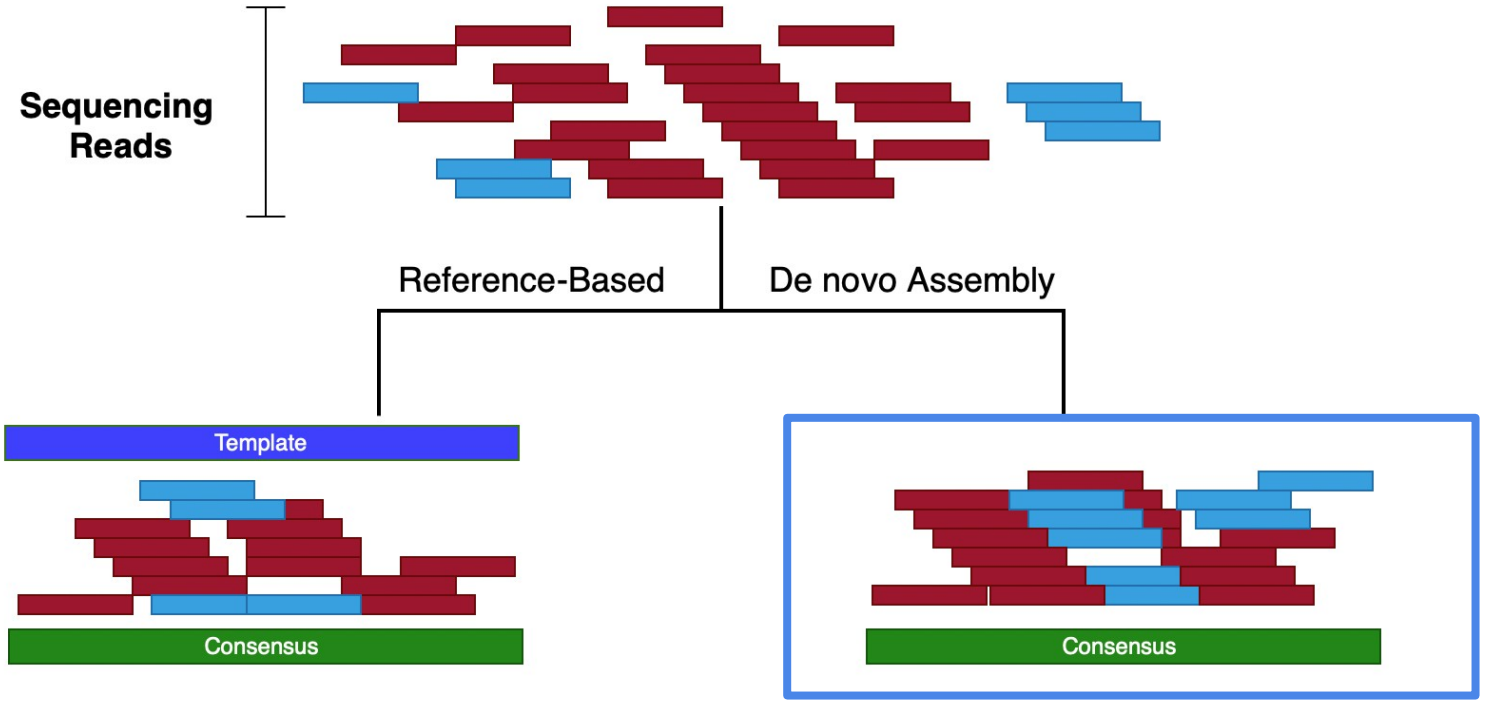
easy (compared to eucaryote) :

- smaller (< 12 Mb)
- less repetition (longest < 10 Kb)
- haploïd

but :

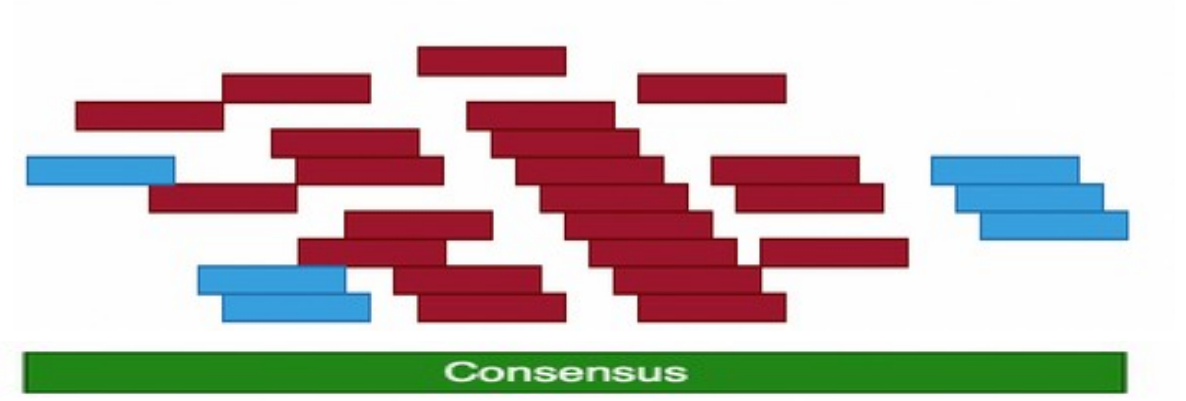
- presence of a plasmid whose copy number differs from that of the chromosome -> different coverage
- circular but without a clear beginning or end

Reference based vs de novo assembly



from https://upload.wikimedia.org/wikipedia/commons/b/b6/Types_of_sequencing_assembly.png

What is the problem?



No unique read yet covering the complete chromosome.

Strategy : bridging chromosome with several reads using sequence similarities to organize them.

What is a de novo genome assembler?

It is a piece of software taking reads as input and producing a set of contigs or scaffolds representing the genome content a the nucleotide level.

There are several categories of assemblers depending on the read length :

- short read assemblers
- long read assemblers
- hybrid assemblers

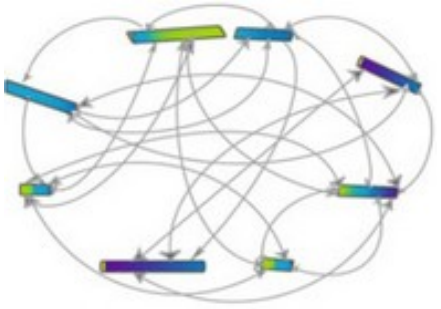
Assembler algorithms

Several genome assembly algorithms have been imagined :

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

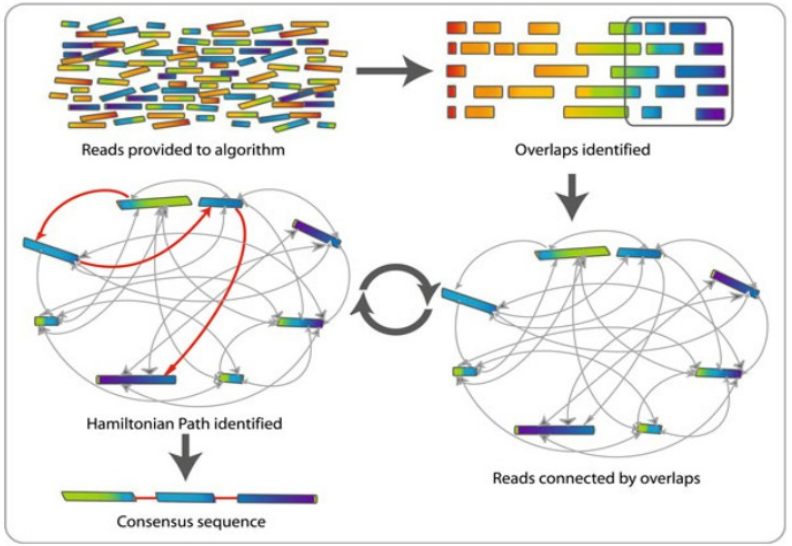
The most used ones are OLC and DBG.

Both represent the assembly as a graph with nodes and edges.



OLC : Overlap Layout Consensus

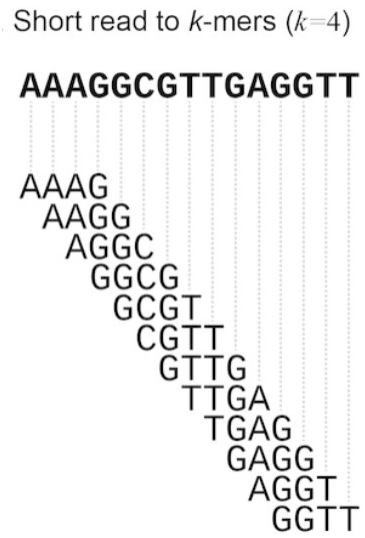
1. Identify all overlaps
2. Identify the paths corresponding to the genome segments
3. Find a single path that explores all **nodes** exactly once.



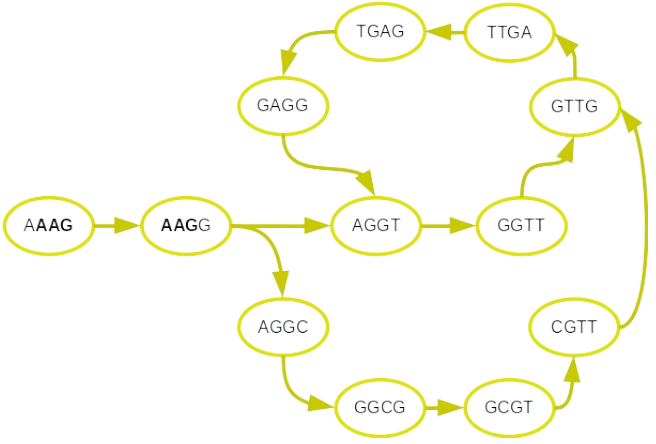
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3055744/

de Bruijn graph

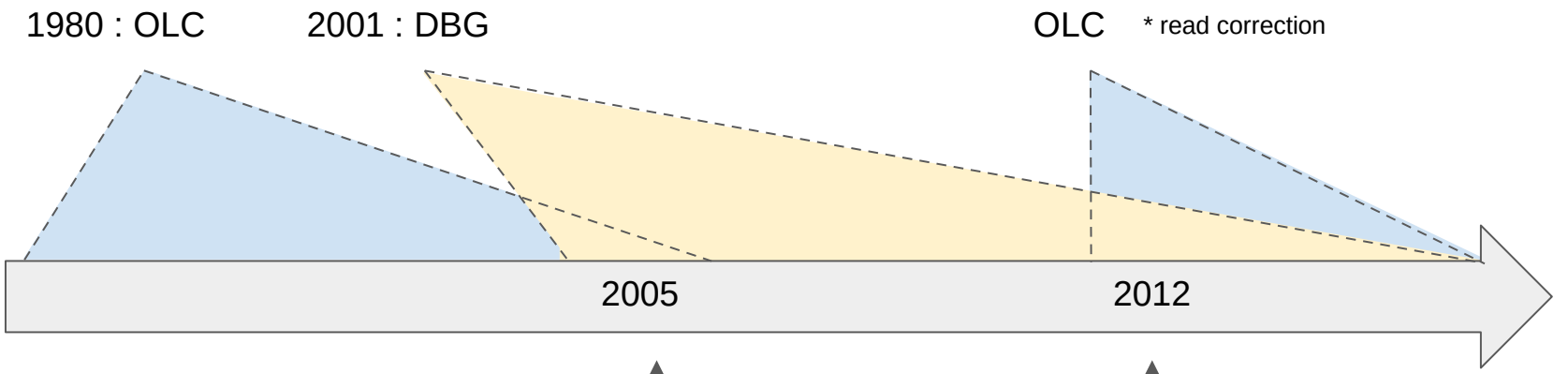
1. Construct the k-mers graph of the reads (substring of length k)
   - nodes: all k-mers present in the reads
   - a link connects 2 nodes if an overlap of length k-1 exists between the 2 k-mers.
2. Find a path that crosses all nodes at least once



de Bruijn graph



OLC and DBG over time



few long
sequences
few errors

Sanger

many short
sequences
few errors

Second generation
sequencing
(Illumina)

2009
cucumber
panda

Third generation
sequencing
(PacBio/Nanopore)

2016
sunflower

2021
wheat

many long
sequences
many errors

Commonly used assemblers

Name	short reads	ONT, CLR reads	hifi reads	algorithm	polishing	scaffolding
SPAdes	X			DBG		X
(Uni-Try)cycler **(SPAdes, miniasm+Racon)**	X	X		DBG first	X	
Flye		X	X	OLC	X	X
wtdbg2 **(redbean)**		X	X	DBG		
(Hi)Canu		X	(X)	OLC		
hifiasm	haplotyping		X	OLC		

Other assemblers

soapdenovo (DBG) : short reads, quick and low memory consumption, short contigs

smartdenovo (OLC) : ONT & CLR long reads, redbean ancestor : same ideas

MaSuRCA (Hybrid) : long read correction with short reads

But also : **Peregrin**, **Nextdenovo**, **NECAT**,**...**

Read length and quality / assembler combination

- Some assembler can work with different data types : usually low quality data assembler can also run on high quality data.
- Assemblers are often long reads or short reads specific. Some combine both data type for read correction but not for assembly (with some exception).
- Check if the assembler is adapted for your data and use the corresponding parameter(s).

What should you take into account?

Read type :

- length
- error rate

Sequencing depth : too low depth = fragmented assembly, too high depth can also impair assembly metrics

Genome repeat fraction and repeat structure : large and very similar repeats are difficult to assemble. Long high quality reads will enable to build through repeats using few variations.

Heterozygocity : high heterozygocity will render the assembly more difficult and you will need more read coverage

Recent whole genome duplication and auto-polyploïdy : multiple copies of the same genome part is not taken into account by the assemblers.

Partial endoreplication : having parts of the genome more represented than others is not taken into account by the assemblers.

Some advices

Know your genome!

- size
- heterozygosity
- repeat content

Try different assemblers : large genome assembly could take weeks and sometimes months a few years ago. Now it is hours or sometimes days. Still you should try different assemblers. Try at least two.

Do not use too much data : Assembler have an optimal coverage range in which they perform best. Assembly metrics are going to worsen with too much data.

Do not stick to N50 : It is better to have a lower N50 with less assembly error than the opposite. Check your assembly versus the reads and/or a reference when available. Check transcript content : BUSCO, RNA-Seq de novo contig alignment,..

What should you expect?

Assembly length should be close to genome size. With error prone reads you expect repeat compression and therefore a **smaller** assembly size. For heterozygous genomes you can find **longer** assemblies than expected because both haplotypes have been kept. This should be check with kmers and corrected with purge_dups or purge haplotigs.

Contig N50 with a correct read depth (**50X** short reads, CLR or ONT and **20X** HiFi or Q20+):

- with short reads only : 10Kb to 200Kb depending on the repeat content and genome size
- with long reads 2Mb to 50Mb depending on heterozygosity, repeat content

Average contig coverage (Nx) should be close to your sequencing coverage for most of the contig, mainly the long ones.

Most of the non error kmer found in the reads should be present in the assembly (in both haplotypes for diploid species).

Assembler parameters

Parameters are different between assemblers.

Categories :

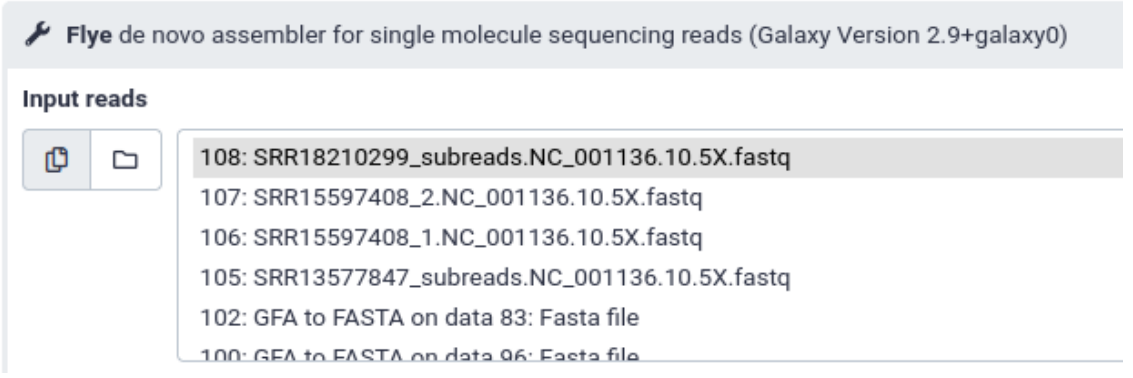
- performance related (CPU, memory)
- genome information (genome size)
- **read type** : when accepting different types
- coverage related : min coverage to keep links between reads, expect cov
- overlap related : when should two reads be seen as having an overlap
- assembly related : graph pruning, type of output : primary, haplotypes
- haplotyping related : Hi-C, trio data
    - purge related : removing duplicated contigs

How to assemble a bacterial genome

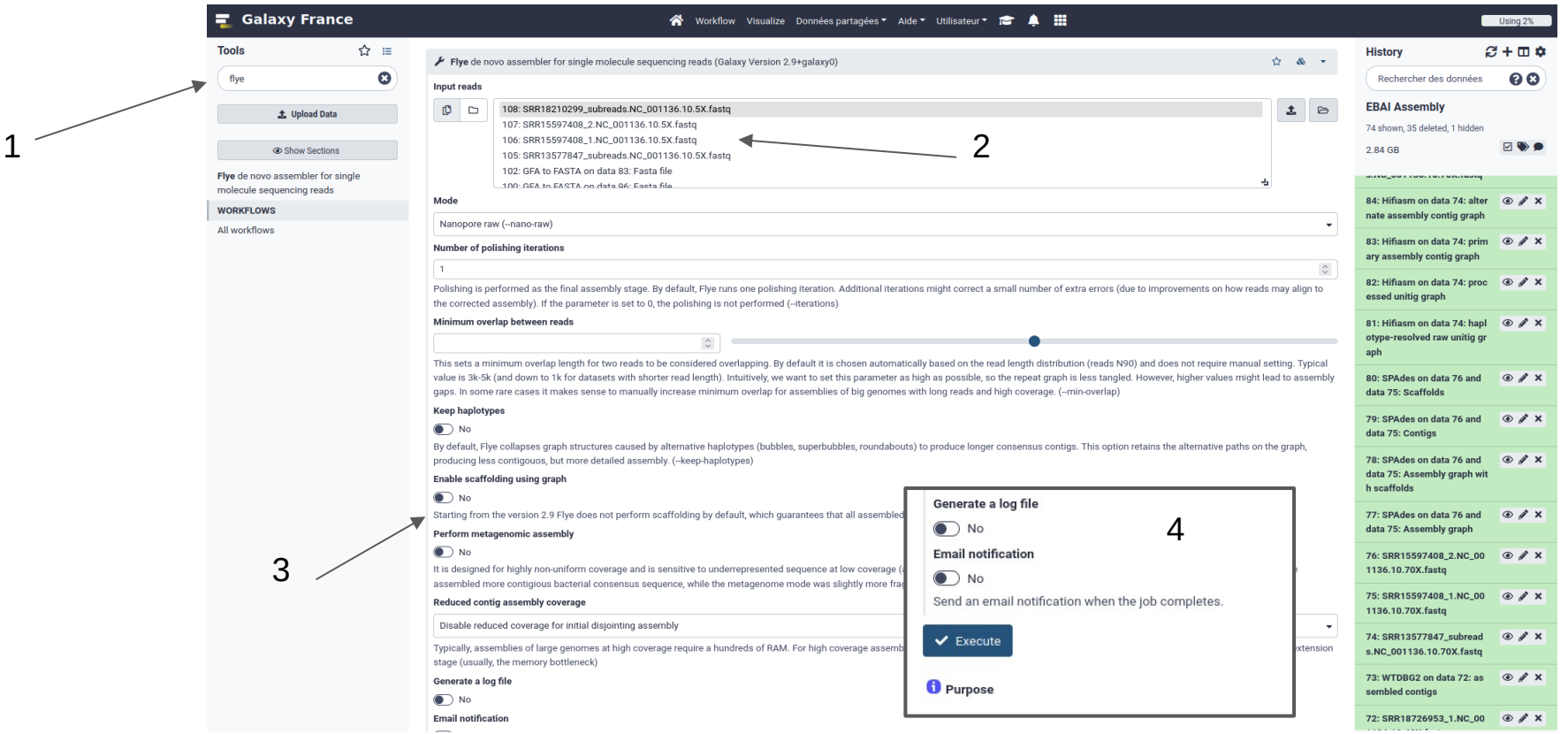
- long reads sequencing !
- short reads sequencing
- if the long reads coverage is greater than ~60X, filter reads on **size** and **quality** (NanoFilt, Filtlong)
- run 2 or 3 different assemblers (specific tool : plasmidSPAdes)
- metrics comparison (N50, # contigs, assembly size…)
- choose the best assembly
- polishing the best assembly (or all and compare the BUSCO scores, reads mapping)

Running an assembly in Galaxy

- Upload your data (usually a fastq file) or have access to it locally.
- select the assembler in the software package list (on the left).
- select your dataset in the list (available fastq datasets)
- set parameters (usually first run with default)
- hit the “execute” button



Running a flye genome assembly in usegalaxy.fr



Conclusions

Know your genome before producing your data and adapt your data production (data type, coverage) to your genome characteristics.

With the right data type(s) genome assembly is now usually easy for genomes up to 1Gb. This kind of assemblies can now be performed by a **single scientist** using a large enough computing infrastructure.

Generate **several** assemblies (different software packages, different coverages (nX),...) and **compare them** to select the best.

Check your assemblies (metrics, protein content, organels, telomeric repeats,...)