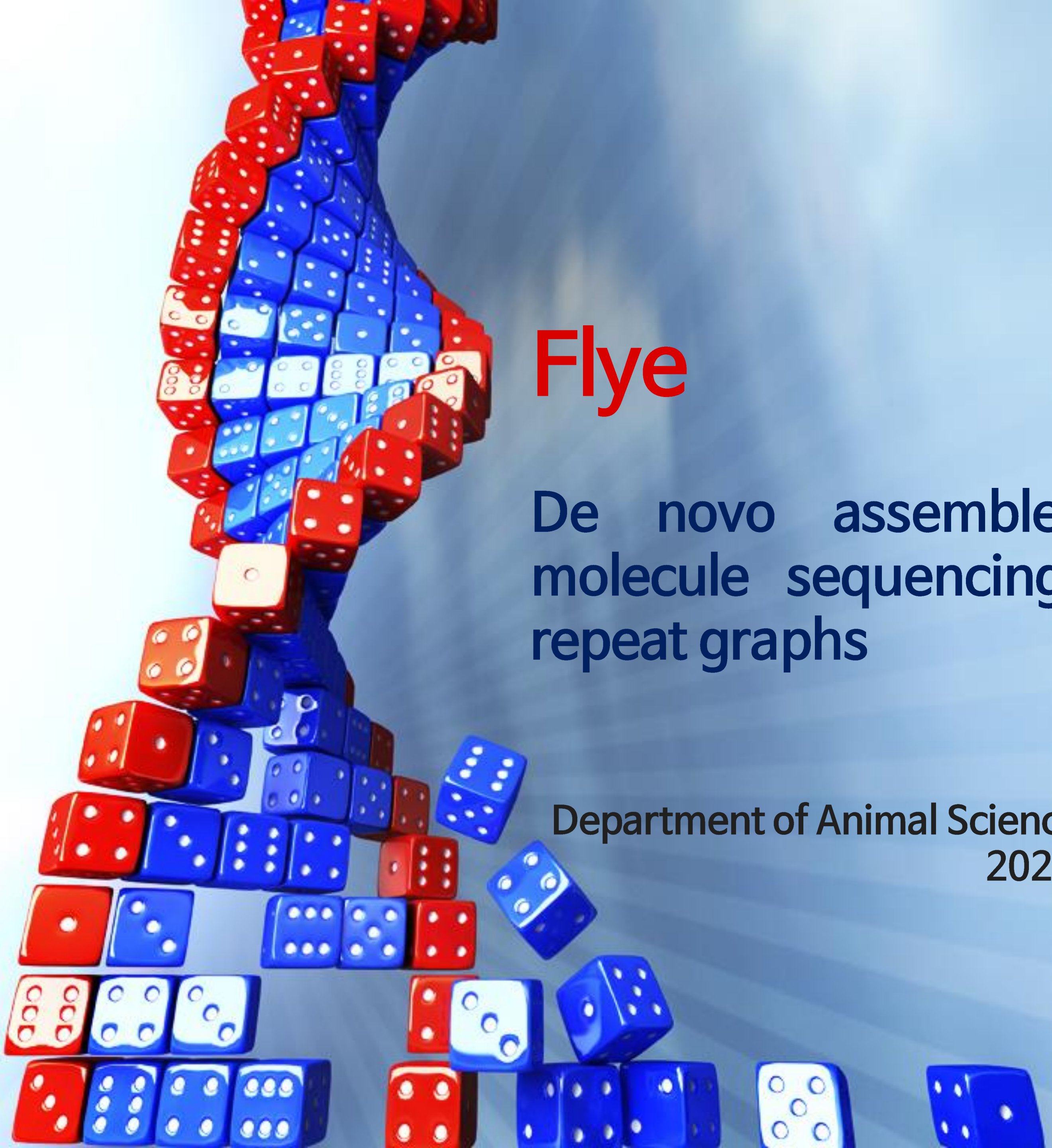


Flye

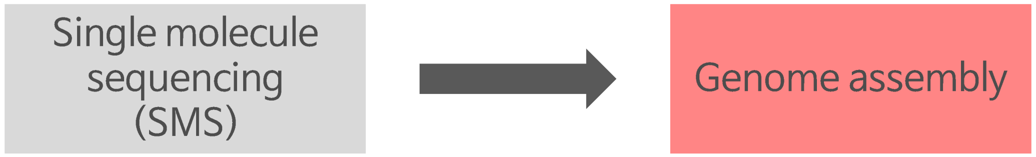
De novo assembler for single molecule sequencing reads using repeat graphs

Department of Animal Science and Biotechnology
2020-24057 Lim Jin Soo

1. Introduction

- **Genome assembly**

  : Process of reconstructing genomes for DNA sequence reads



- **Single molecule seuqencing(SMS)**

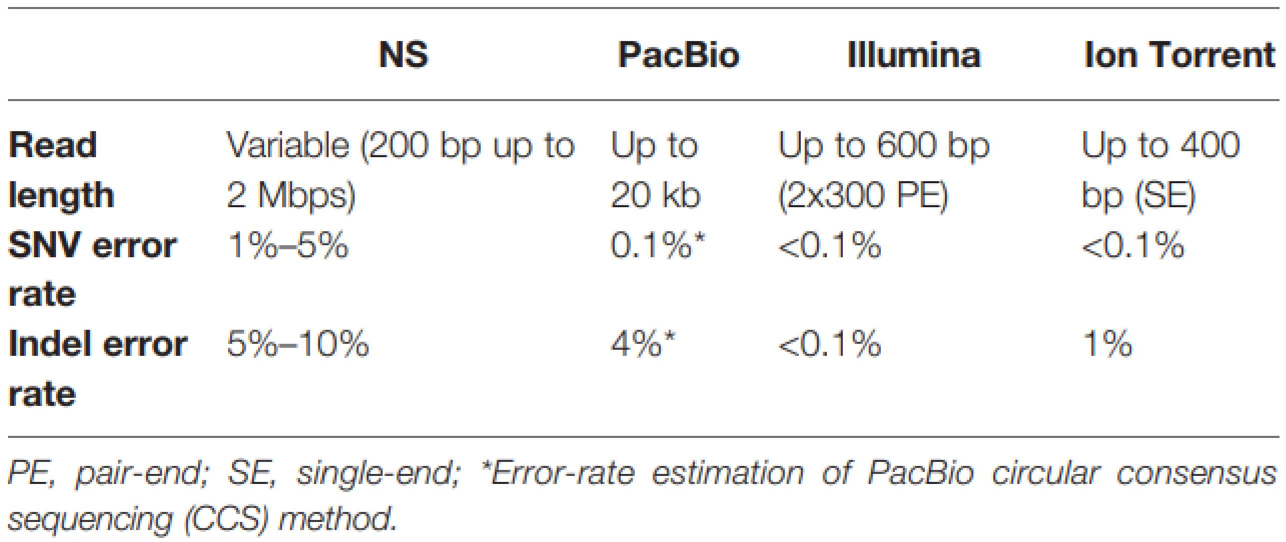
  : Using short-read technologies(Illumina), using long-read technologies(Pacific Biosciences & Oxford Nanopore)

- Genome assembly approach: **'The de Bruijn graph'**

'Flye(assembly of long, error-prone reads using repeat graphs)'

- **Comparison of Single molecule seuqencing(SMS) platforms**

  : Long-read sequencing technologies can produce much more longer reads. But drawback is the relatively high error rate.



	NS	PacBio	Illumina	Ion Torrent
Read length	Variable (200 bp up to 2 Mbps)	Up to 20 kb	Up to 600 bp (2x300 PE)	Up to 400 bp (SE)
SNV error rate	1%–5%	0.1%*	<0.1%	<0.1%
Indel error rate	5%–10%	4%*	<0.1%	1%

*PE, pair-end; SE, single-end; *Error-rate estimation of PacBio circular consensus sequencing (CCS) method.*

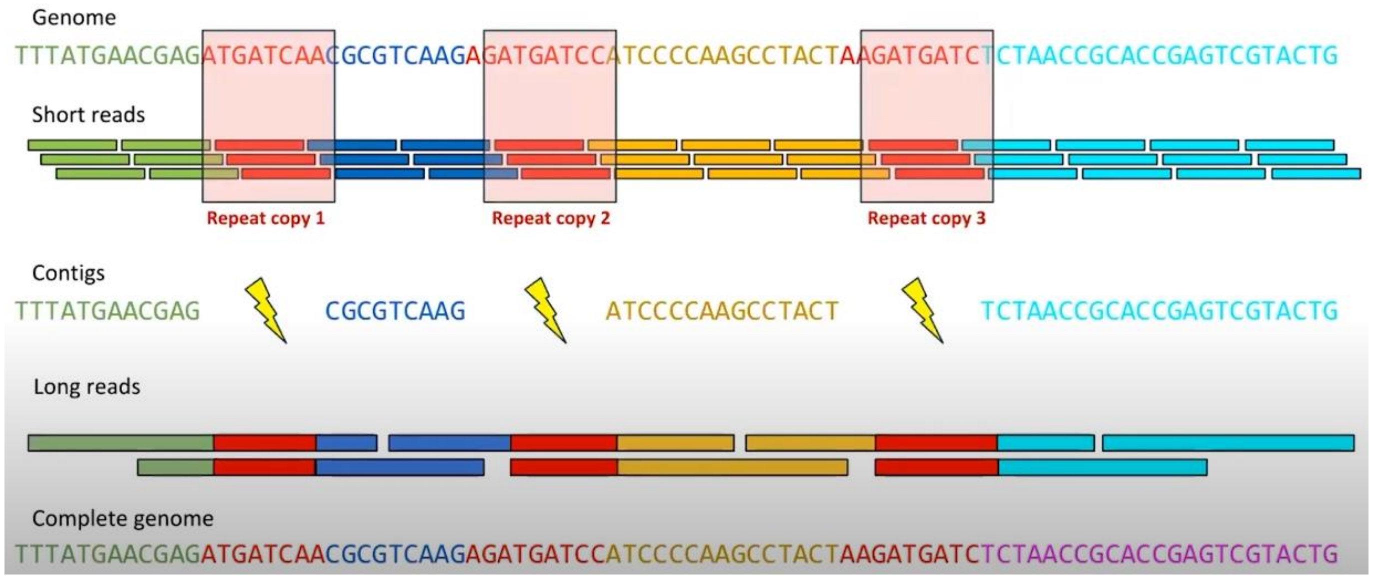
〈 Error-rate comparison of NS, PacBio, Ilumina, and Ion Torrent sequencing platforms〉

“Nanopore Sequencing in Blood Diseases: A Wide Range of Opportunities”, Crescenzio Francesco Minervini et al.

- **Challenges for genome assembly**

  - SMS short-read assembly

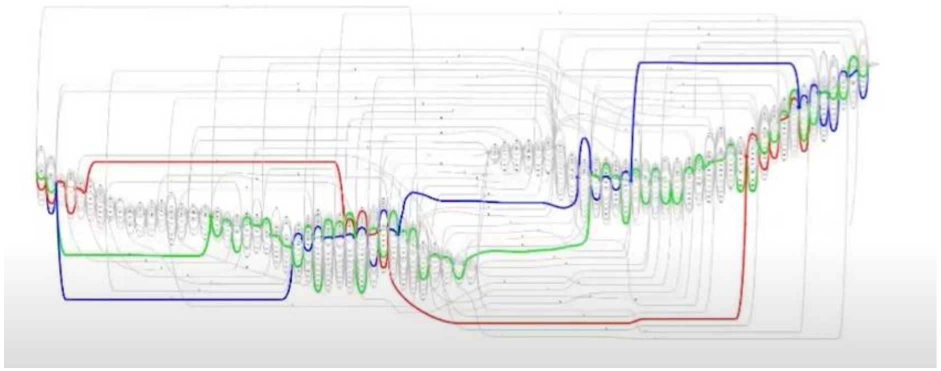
  : In repetitive regions of the genome, accurately assembling short reads is challenging


Genome
TTTATGAACGAGATGATCAACGCGTCAAGAGATGATCCATCCCCAAGCCTACTAAGATGATCTCTAACCGCACCGAGTCGTACTG
Short reads
Repeat copy 1
Repeat copy 2
Repeat copy 3
Contigs
TTTATGAACGAG
CGCGTCAAG
ATCCCCAAGCCTACT
TCTAACCGCACCGAGTCGTACTG
Long reads
Complete genome
TTTATGAACGAGATGATCAACGCGTCAAGAGATGATCCATCCCCAAGCCTACTAAGATGATCTCTAACCGCACCGAGTCGTACTG

〈 Assemblies can be fragmented〉

- The de Bruijn graph approach assembly

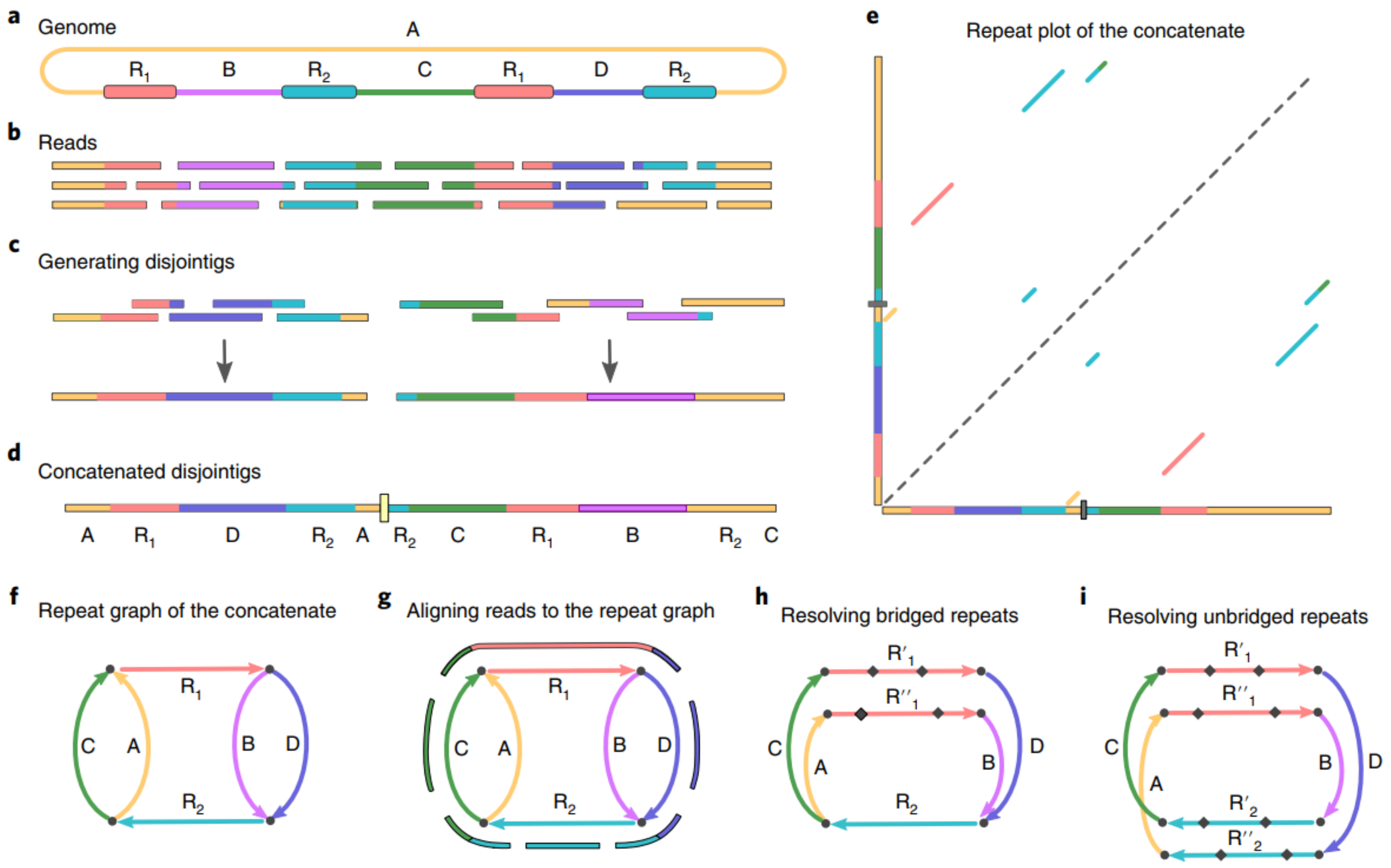
: Due to long-read sequencing error rate, how to deal with fragmented de Bruijn graph and how to transform it into an assembly graph is challenging



〈 Transformation of de Bruijn graph into an assembly graph〉

Very complicated and time consuming!!

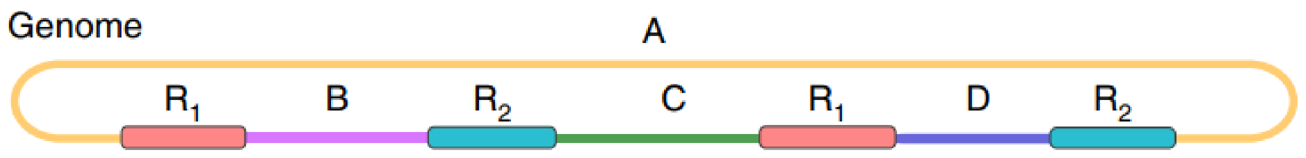
- Flye outline



〈 Flye outline 〉

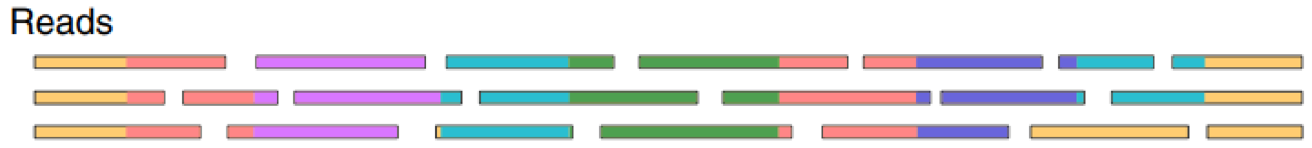
- A ‘genome’ with two 99% identical copies of a repeat R1 and two 99% identical copies of a repeat R2. Segments A, B, C, and D represent non-repetitive regions.

a

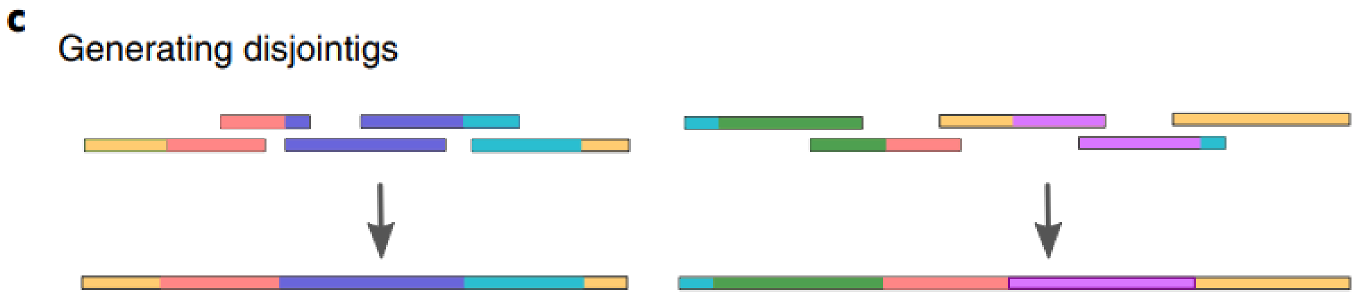


- A set of reads from the genome.

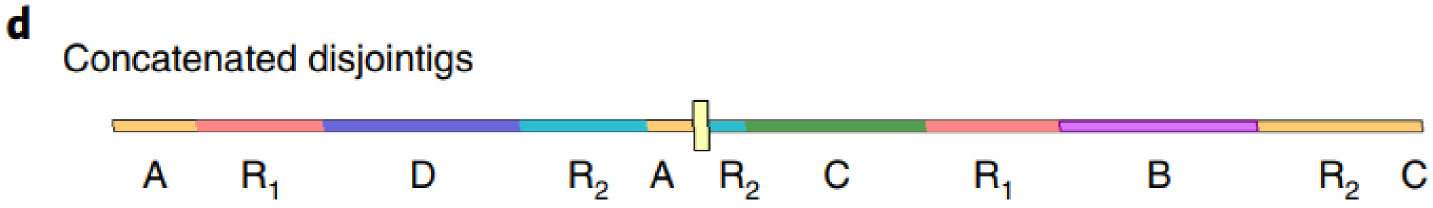
b



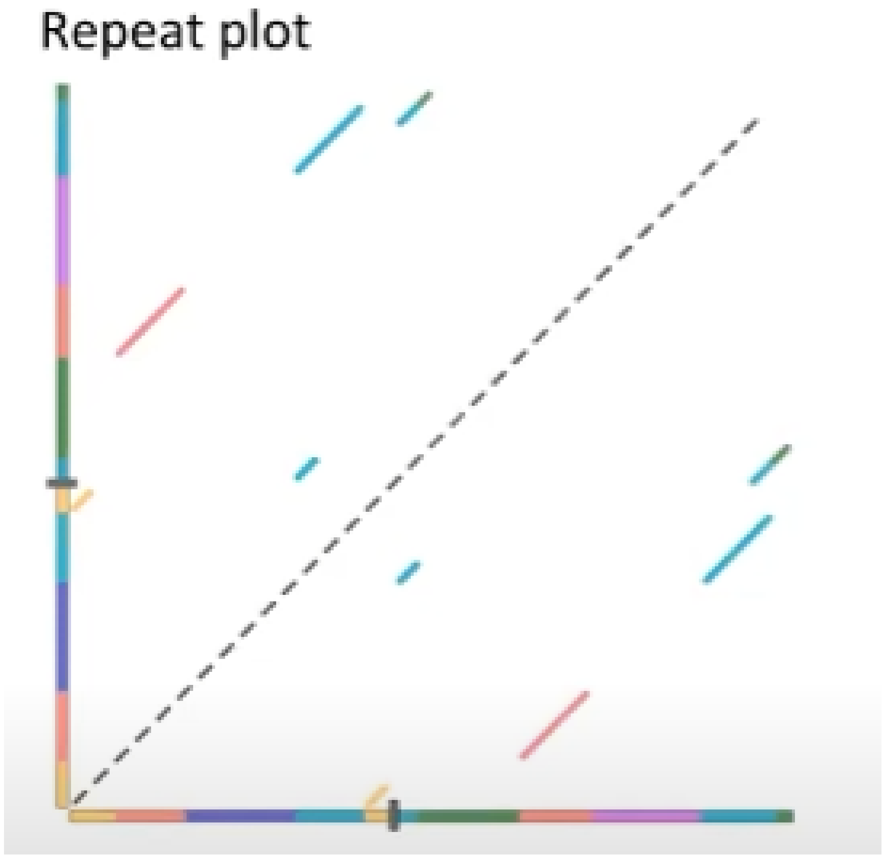
- Two (misassembled) disjointigs AR1DR2A and R2CR1BR2C derived from the reads.



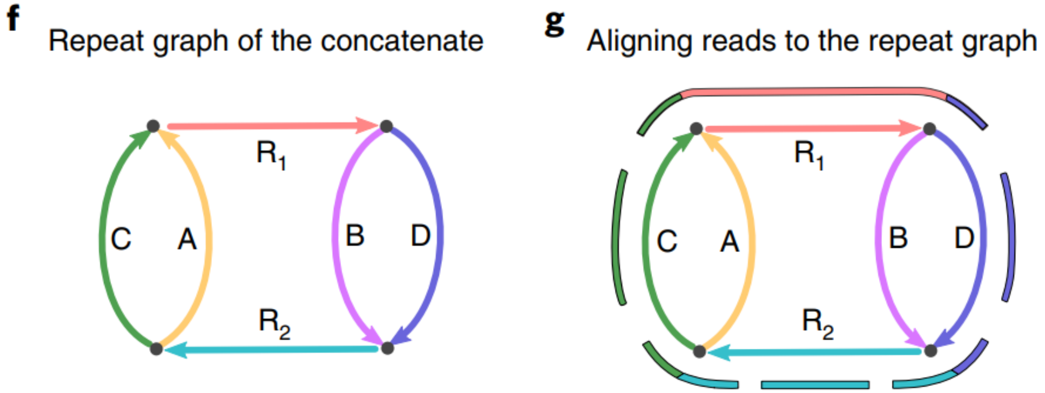
- Concatenate of the disjointigs.



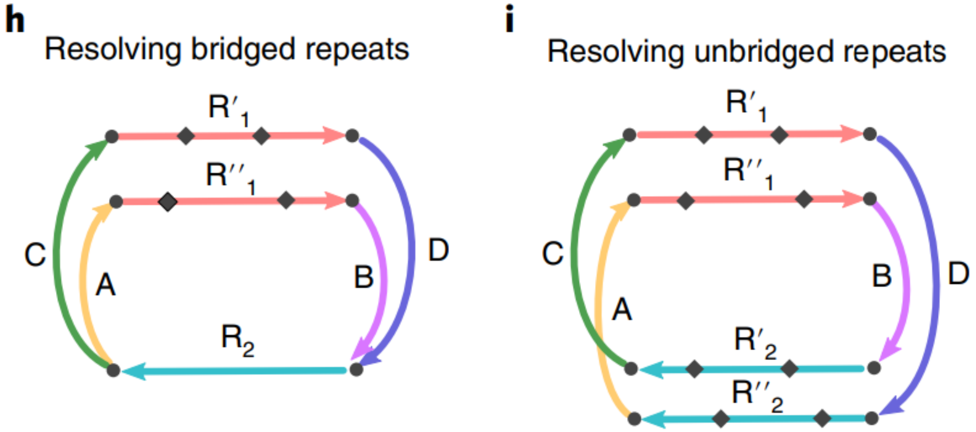
- Repeat plot of the concatenate.



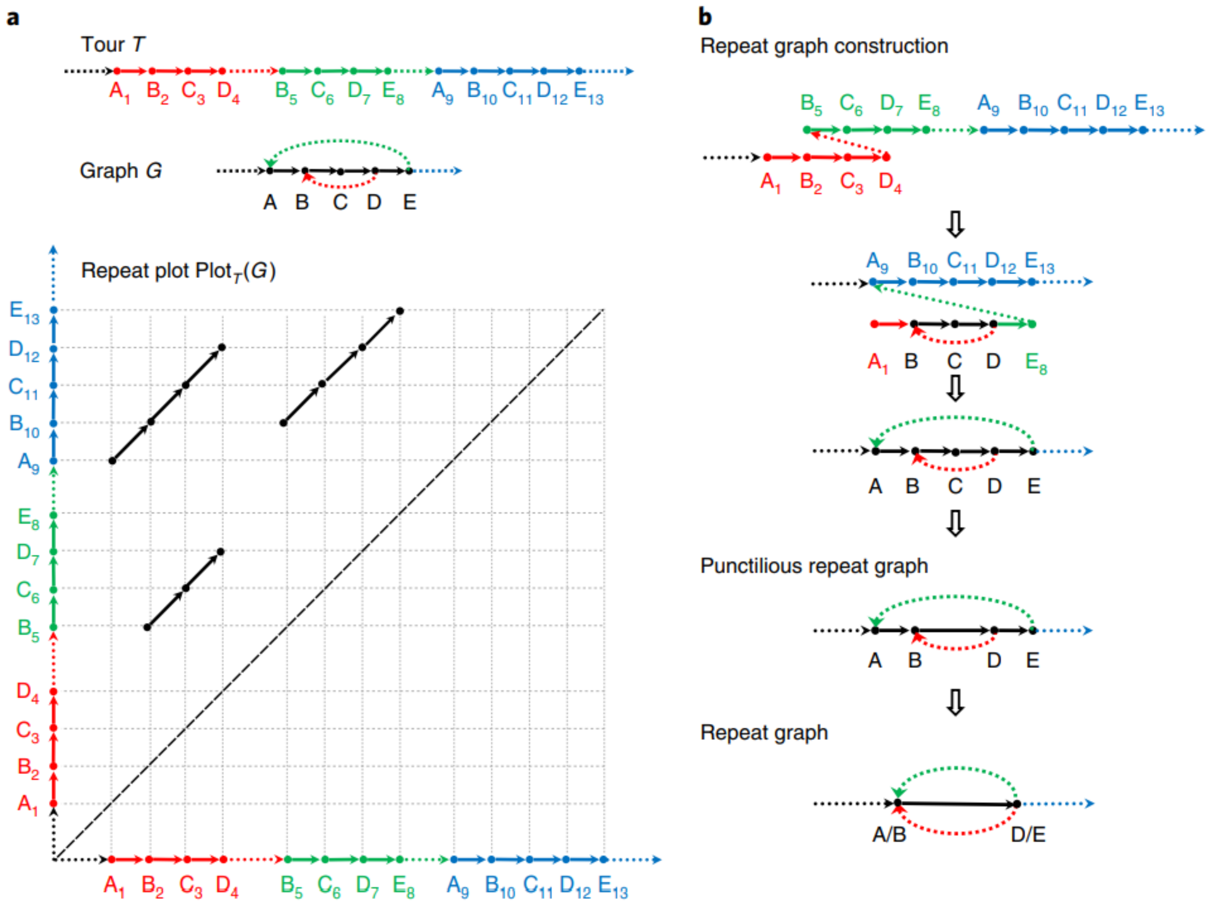
- Repeat graph constructed by gluing vertices in the concatenate according to the repeat plot. Aligning reads against the repeat graph.


f
Repeat graph of the concatenate
R1
C
A
B
D
R2
g
Aligning reads to the repeat graph
R1
C
A
B
D
R2

- Resolving the bridged repeat R1 and reconstructing its two copies R′1 and R′′1. Resolving the unbridged repeat R2 with two slightly diverged copies.



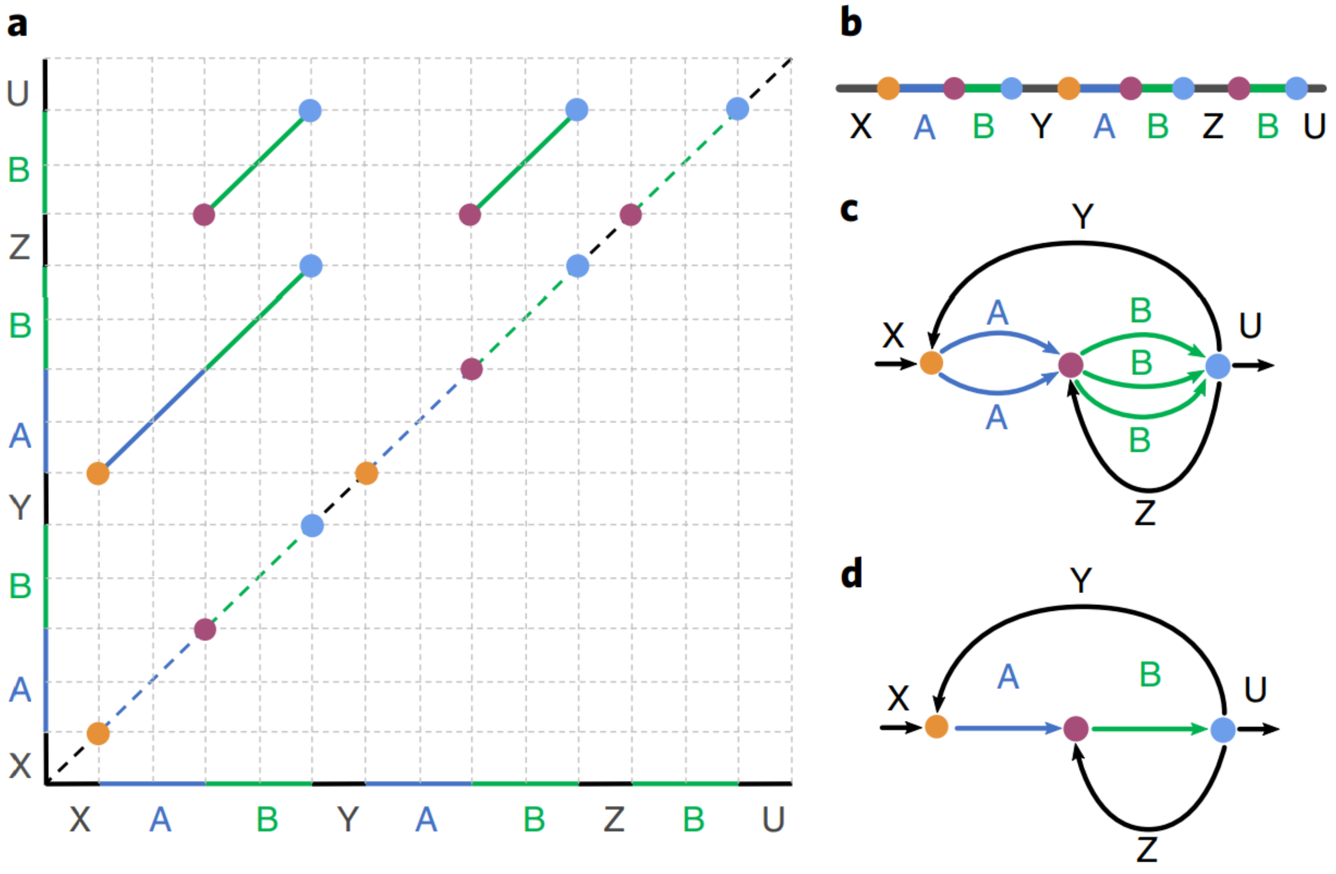
- **Theoretical framework for the repeat graph construction**



⟨Constructing the repeat plot of a tour in the graph and constructing the repeat graph from a repeat plot⟩

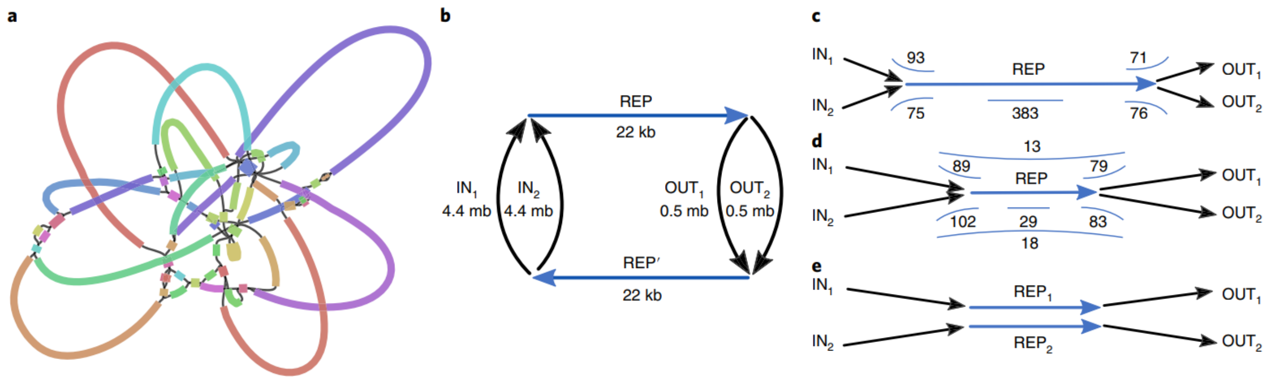
2. Results

- **Repeat graph construction**
  : Repeat graph compactly represents all repeats in a genome and reveals their mosaic structure



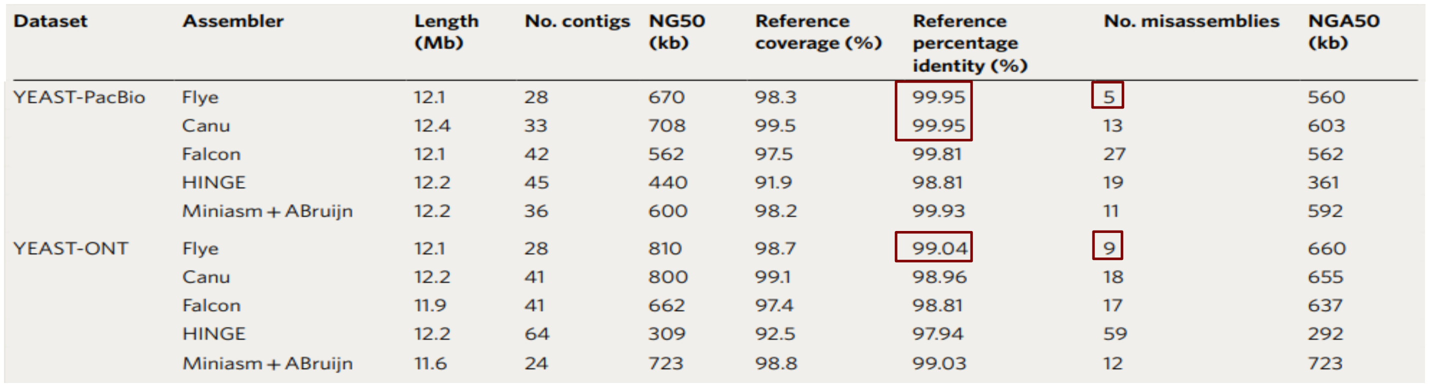
〈Constructing the approximate repeat graph from local self-alignments〉

- **Resolving unbridged repeats with Flye**
  : Flye utilizes the constructed repeat graph for the resolution of unbridged repeats. The repeat graph constructed by Flye offers an approach for resolving unbridged repeats based on analyzing the topology of the repeat graph.



〈Resolving an unbridged repeat〉

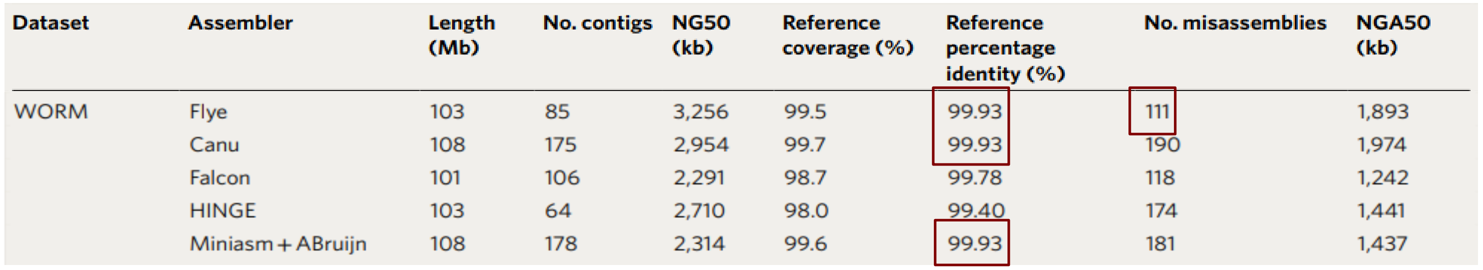
- **Benchmarking with the YEAST dataset**



Dataset	Assembler	Length (Mb)	No. contigs	NG50 (kb)	Reference coverage (%)	Reference percentage identity (%)	No. misassemblies	NGA50 (kb)
YEAST-PacBio	Flye	12.1	28	670	98.3	$\boxed{99.95}$	$\boxed{5}$	560
	Canu	12.4	33	708	99.5	$\boxed{99.95}$	13	603
	Falcon	12.1	42	562	97.5	99.81	27	562
	HINGE	12.2	45	440	91.9	98.81	19	361
	Miniasm + ABruijn	12.2	36	600	98.2	99.93	11	592
YEAST-ONT	Flye	12.1	28	810	98.7	$\boxed{99.04}$	$\boxed{9}$	660
	Canu	12.2	41	800	99.1	98.96	18	655
	Falcon	11.9	41	662	97.4	98.81	17	637
	HINGE	12.2	64	309	92.5	97.94	59	292
	Miniasm + ABruijn	11.6	24	723	98.8	99.03	12	723

- The YEAST dataset contains PacBio and Oxford Nanopore Technology (ONT) reads from the *Saccharomyces cerevisiae* S288c genome of length 12.1Mb at 30× coverage.

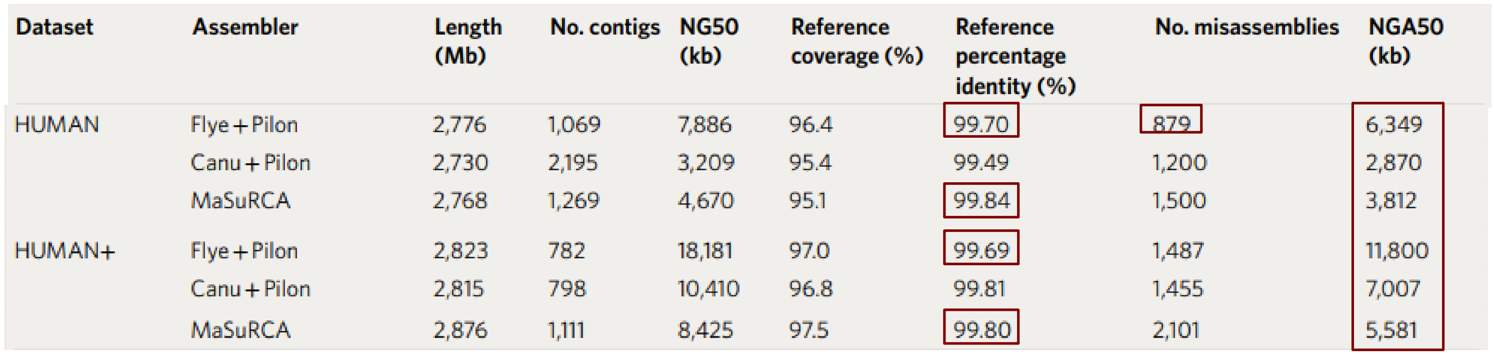
- **Analyzing the WORM dataset**



Dataset	Assembler	Length (Mb)	No. contigs	NG50 (kb)	Reference coverage (%)	Reference percentage identity (%)	No. misassemblies	NGA50 (kb)
WORM	Flye	103	85	3,256	99.5	99.93	111	1,893
	Canu	108	175	2,954	99.7	99.93	190	1,974
	Falcon	101	106	2,291	98.7	99.78	118	1,242
	HINGE	103	64	2,710	98.0	99.40	174	1,441
	Miniasm + ABruijn	108	178	2,314	99.6	99.93	181	1,437

-The WORM dataset contains PacBio reads from *the Caenorhabditis elegans* genome of length ~ 100Mb at 40× coverage.

- **Analyzing the HUMAN and HUMAN+ datasets**



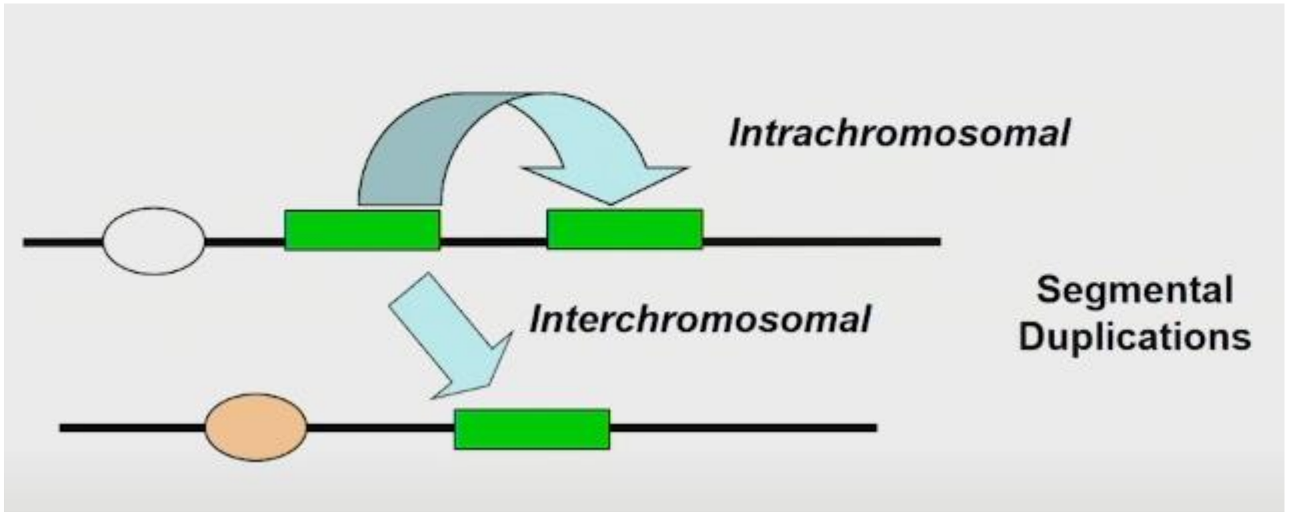
Dataset	Assembler	Length (Mb)	No. contigs	NG50 (kb)	Reference coverage (%)	Reference percentage identity (%)	No. misassemblies	NGA50 (kb)
HUMAN	Flye+Pilon	2,776	1,069	7,886	96.4	99.70	879	6,349
	Canu+Pilon	2,730	2,195	3,209	95.4	99.49	1,200	2,870
	MaSuRCA	2,768	1,269	4,670	95.1	99.84	1,500	3,812
HUMAN+	Flye+Pilon	2,823	782	18,181	97.0	99.69	1,487	11,800
	Canu+Pilon	2,815	798	10,410	96.8	99.81	1,455	7,007
	MaSuRCA	2,876	1,111	8,425	97.5	99.80	2,101	5,581

- The HUMAN dataset contains ONT reads from the GM12878 human cell line at 30× coverage complemented by a set of short Illumina reads at 50× coverage.

- The HUMAN+ dataset combines the HUMAN dataset with a dataset of ultra-long ONT reads (those with reads N50>100 kb; that is, 50% of the total sequence data in reads longer than100 kb) at 5× coverage.

- **Complex mosaic structure of segmental duplication(SD)**

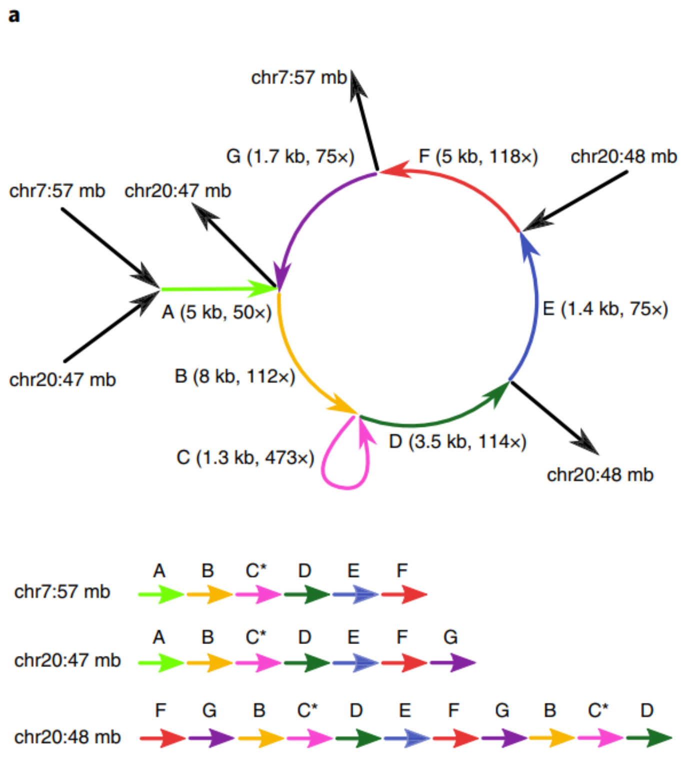
  : SDs are long DNA sequences (typically defined as being 〉1kb in length) that have nearly identical sequences (90–100%) and exist in multiple locations as a result of duplication events.
  (tandem or interspersed & interchromosomal or intrachromosomal)

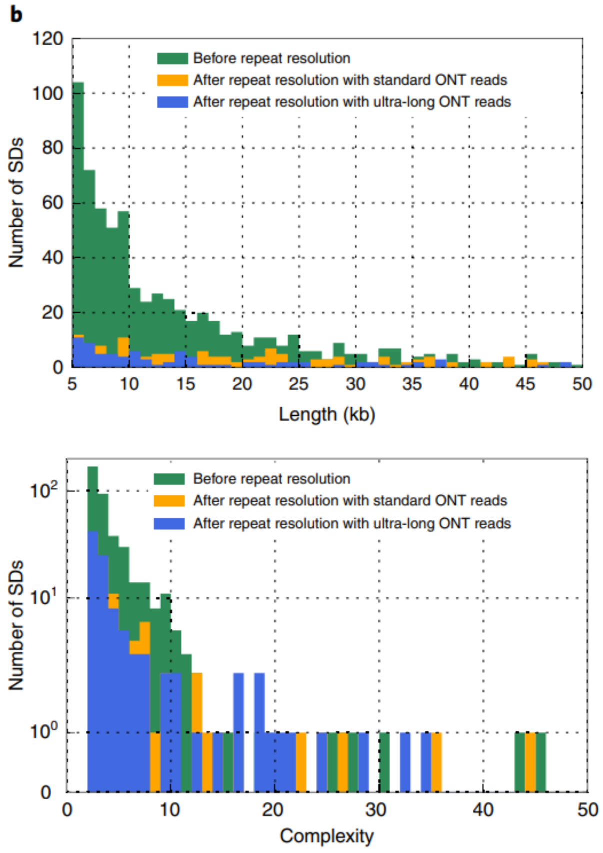


〈Simple image showing Segmental Duplication〉

- **SDs in the human genome**

  : The repeat graph constructed by Flye reveals the complex mosaic structure of SDs.

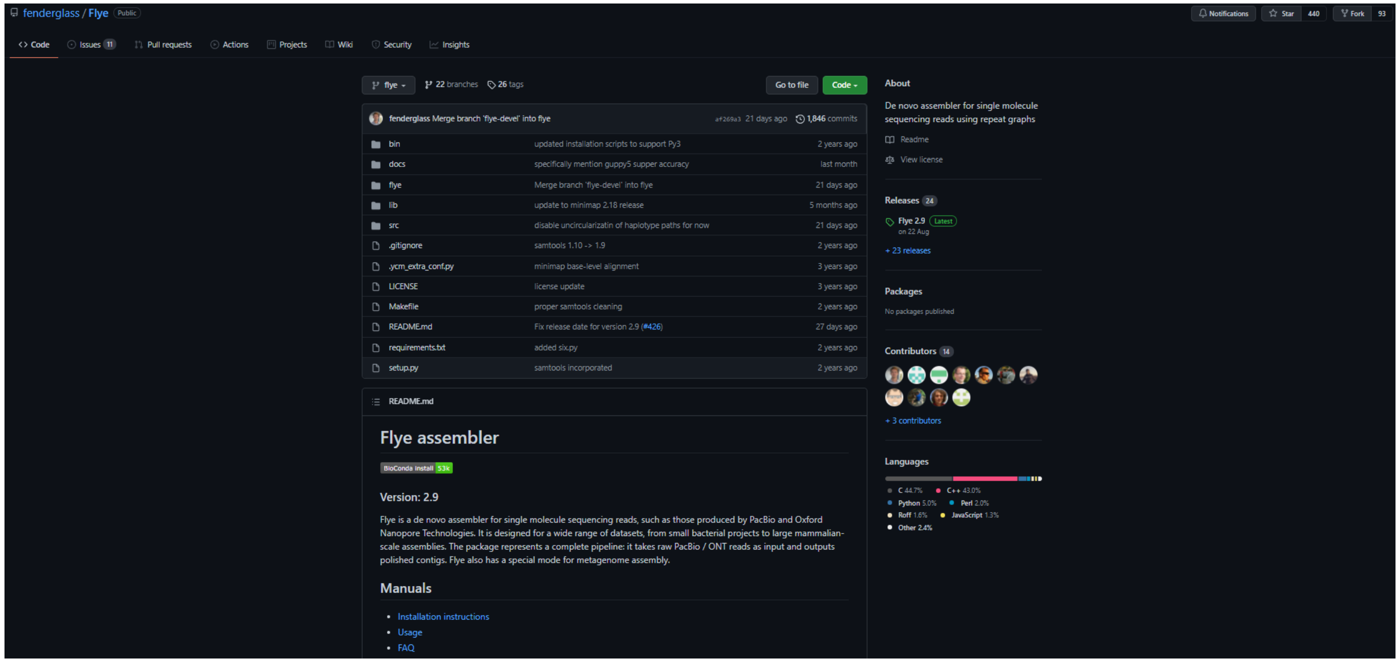




⟨An SD from the Flye assembly of the HUMAN dataset and the distribution of the lengths and complexities of all SDs from the same assembly⟩

3. Github : Flye assembler

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



fenderglass Merge branch 'flye-devel' into flye af269a3 21 days ago 1,846 commits

bin	updated installation scripts to support Py3	2 years ago
docs	specifically mention guppy5 supper accuracy	last month
flye	Merge branch 'flye-devel' into flye	21 days ago
lib	update to minimap 2.18 release	5 months ago
src	disable uncircularizatin of haplotype paths for now	21 days ago
.gitignore	samtools 1.10 -> 1.9	2 years ago
.ycm_extra_conf.py	minimap base-level alignment	3 years ago
LICENSE	license update	3 years ago
Makefile	proper samtools cleaning	2 years ago
README.md	Fix release date for version 2.9 (#426)	27 days ago
requirements.txt	added six.py	2 years ago
setup.py	samtools incorporated	2 years ago

README.md

Flye assembler

BioConda Install 53k

Version: 2.9

Flye is a de novo assembler for single molecule sequencing reads, such as those produced by PacBio and Oxford Nanopore Technologies. It is designed for a wide range of datasets, from small bacterial projects to large mammalian-scale assemblies. The package represents a complete pipeline: it takes raw PacBio / ONT reads as input and outputs polished contigs. Flye also has a special mode for metagenome assembly.

Manuals

- Installation instructions
- Usage
- FAQ

About

De novo assembler for single molecule sequencing reads using repeat graphs

Readme

View license

Releases 24

Flye 2.9 Latest
on 22 Aug

+ 23 releases

Packages

No packages published

Contributors 14

+ 3 contributors

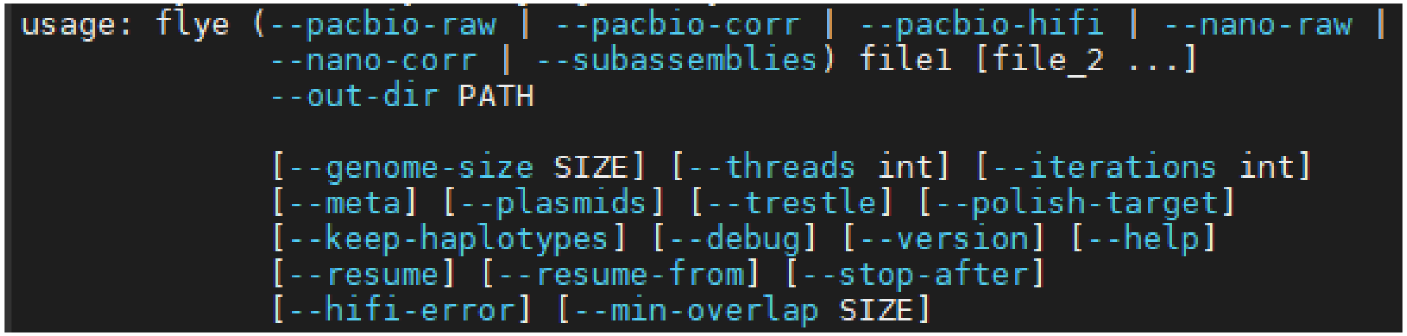
Languages

● C 44.7% ● C++ 43.0%
● Python 5.0% ● Perl 2.0%
● Roff 1.6% ● JavaScript 1.3%
● Other 2.4%

⟨Github: Flye assembler⟩

- **Flye assembler input & parameters**

  :Input reads can be in FASTA or FASTQ format, uncompressed or compressed with gz. Currently, PacBio (raw, corrected, HiFi) and ONT reads (raw, corrected) are supported.



```
usage: flye (--pacbio-raw | --pacbio-corr | --pacbio-hifi | --nano-raw |
             --nano-corr | --subassemblies) file1 [file_2 ...]
             --out-dir PATH

             [--genome-size SIZE] [--threads int] [--iterations int]
             [--meta] [--plasmids] [--trestle] [--polish-target]
             [--keep-haplotypes] [--debug] [--version] [--help]
             [--resume] [--resume-from] [--stop-after]
             [--hifi-error] [--min-overlap SIZE]
```

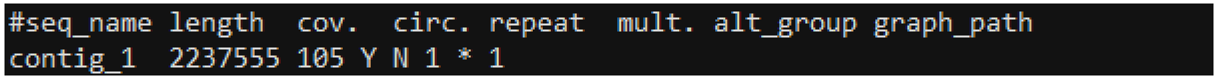
⟨Flye short manual⟩

- **Flye assembler output**

  - assembly.fasta
  : Final assembly. Contains contigs and possibly scaffolds.

  - assembly_graph. {gfa | gv}
  : Final repeat graph. Note that the edge sequences might be different (shorter) than contig sequences, because contigs might include multiple graph edges.

  - assembly_info.txt : Extra information about contigs (such as length or coverage)



```
#seq_name length   cov.   circ. repeat   mult. alt_group graph_path
contig_1   2237555 105 Y N 1 * 1
```

⟨assembly_info.txt of *Bifidobacterium bifidum*⟩

4. Discussion

- Flye algorithms for constructing an assembly graph from sequencing reads and repeat characterization improves the genome assembly.

- Benchmarking flye against five state-of-the-art assemblers and show that it generates better or comparable assembles, while being an order of magnitude faster.

- Flye algorithm for resolving unbridged repeats resolved only a small fraction of various long SDs since it is currently limited to simple SDs. Moreover it currently has difficulties resolving highly similar SDs(SDs with less than 1% divergence).

5. Q & A

Thank You for listening
Any Questions?