

Genome assembly

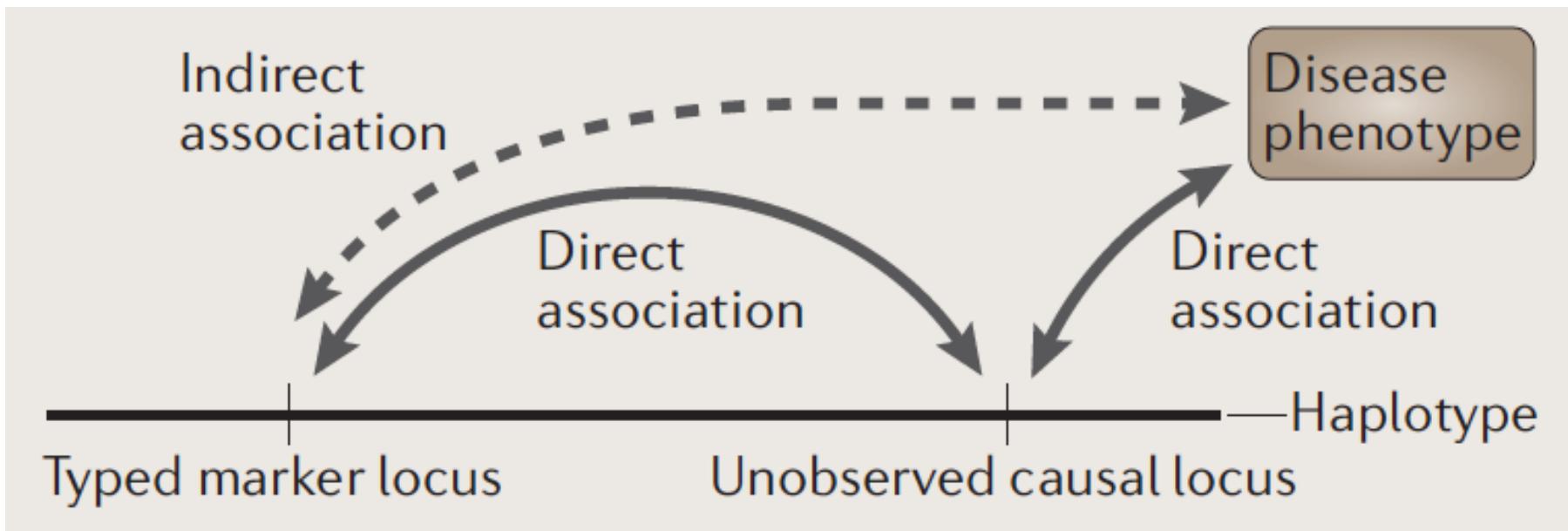
Bioinformatics Applications (PLPTH813)

Sanzhen Liu

3/25/2021

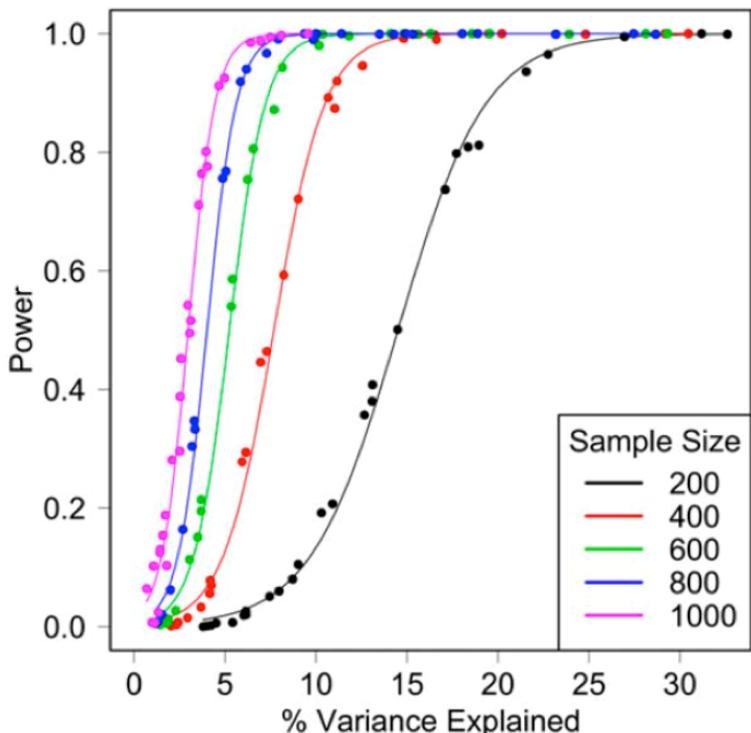
Linkage disequilibrium (LD)

Linkage disequilibrium (LD): a non-random association of alleles at different loci; genotyping data at two loci have some level of correlations



Association is NOT equal to causality

The lower the effect of causal sites, the larger the sample size is needed



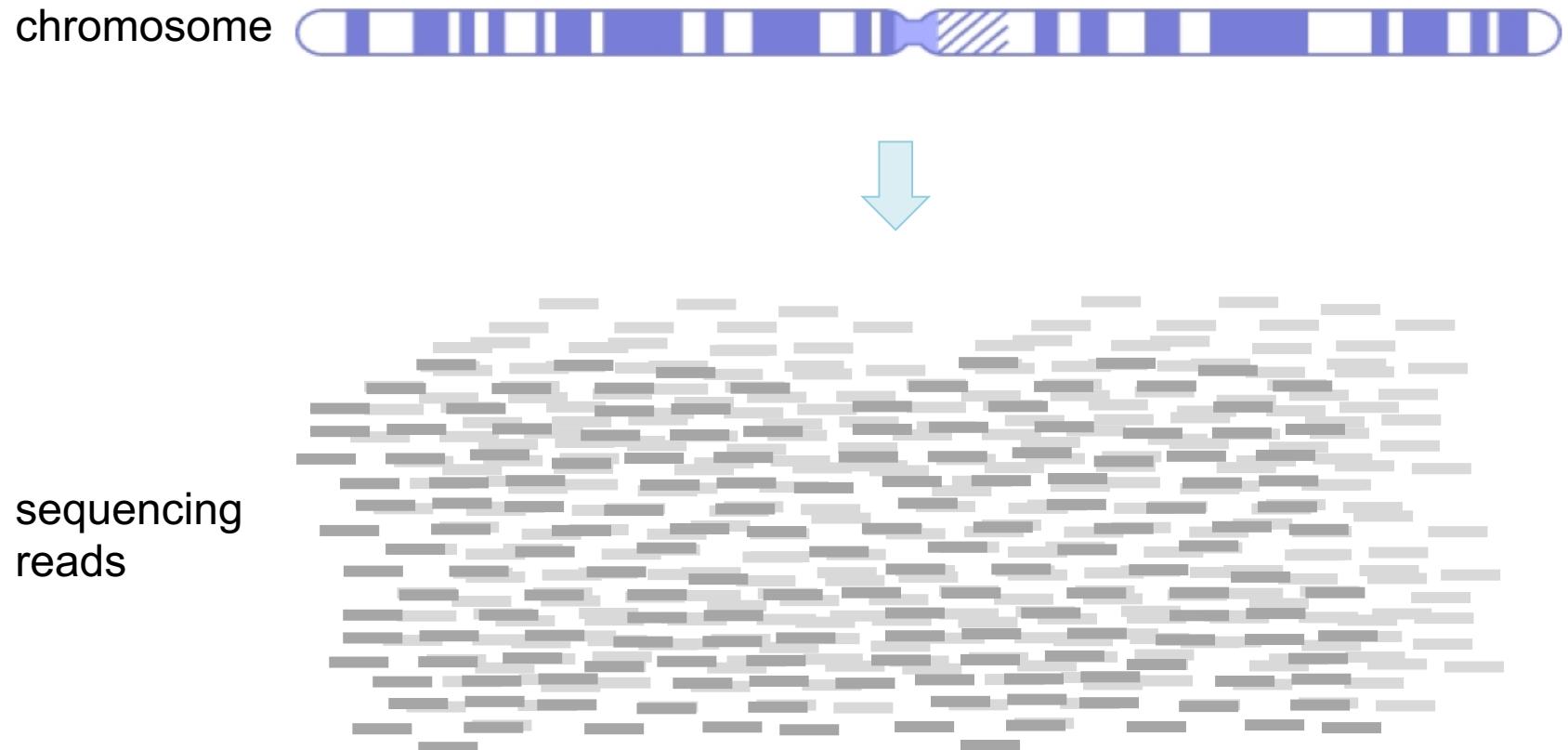
Species	year	Journal	Population size	Trait
Cotton	2017	NG	318	Fiber and yield
Cotton	2017	NG	352	Fiber quality
Rice	2016	NG	176	Agronomic traits
Rice	2016	NG	342	Grain shape
Rice	2016	Nature	10,074	Heterotic effects

from previous students (2019)
Ruolin, Upasana, Diriba

Outline

- Genome assembly: concept
- Assembly algorithms: OLC/De Bruijn graph
- Error correction (k-mer counts)
- Assembly strategy to cope with the repeat problem
- Assembly evaluation (N50, comparison to a reference genome)
- Long-read genome assembly
- Genome annotation
- Case study

Whole genome shotgun (WGS) sequencing

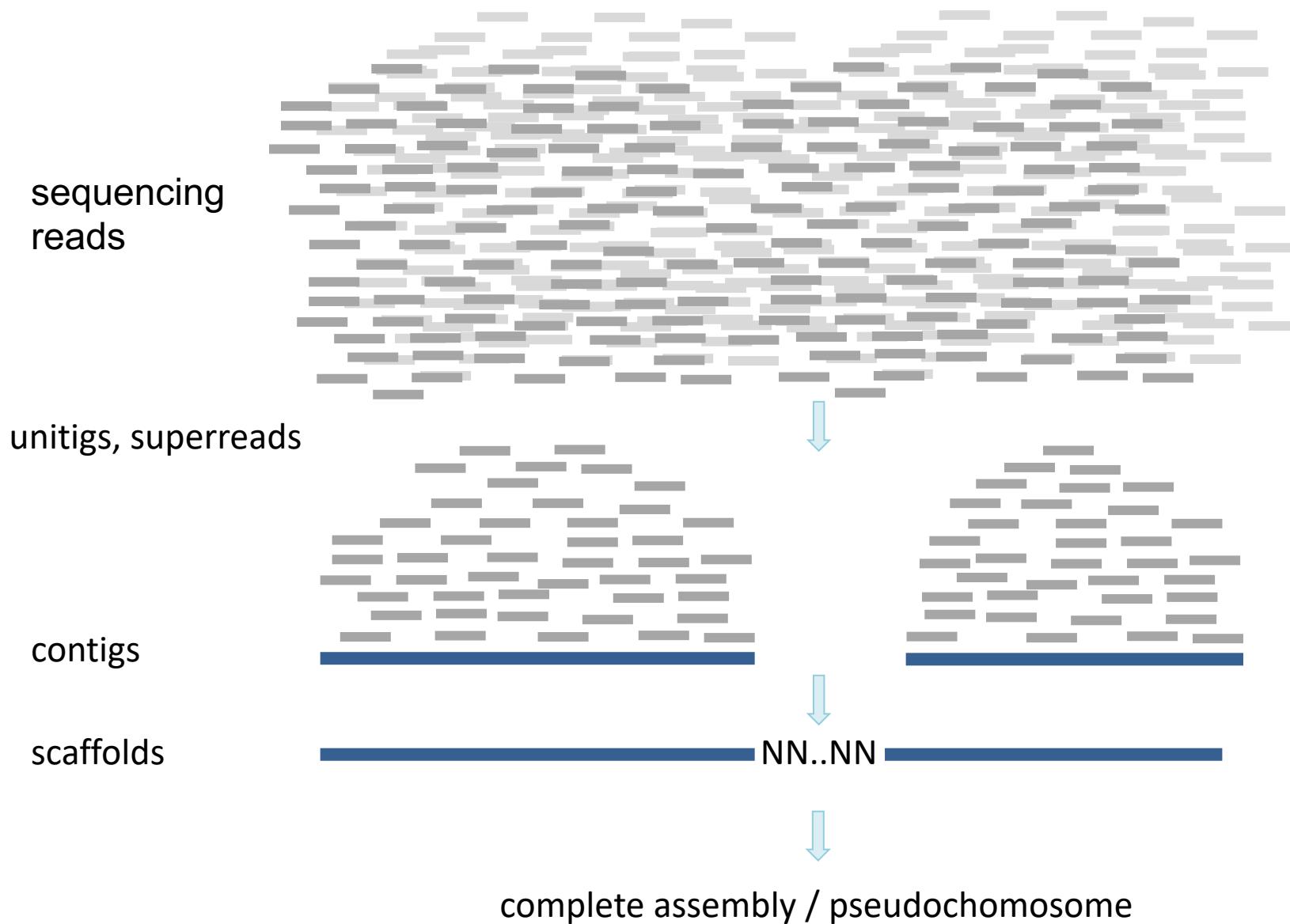


Complexity of genome assembly



Algorithm can Solve 10, 000
Piece jigsaw in 24 Hours

Genome assembly



Assembly algorithms

- **Overlap layout consensus (OLC)**
 1. all-to-all pair-wise read alignments
 2. Construct graph based on overlaps
 3. Trace paths for the assembly

- **De Bruijn graph**
 1. Determine k-mer from reads
 2. Construct k-mer graph
 3. Trace paths for the assembly

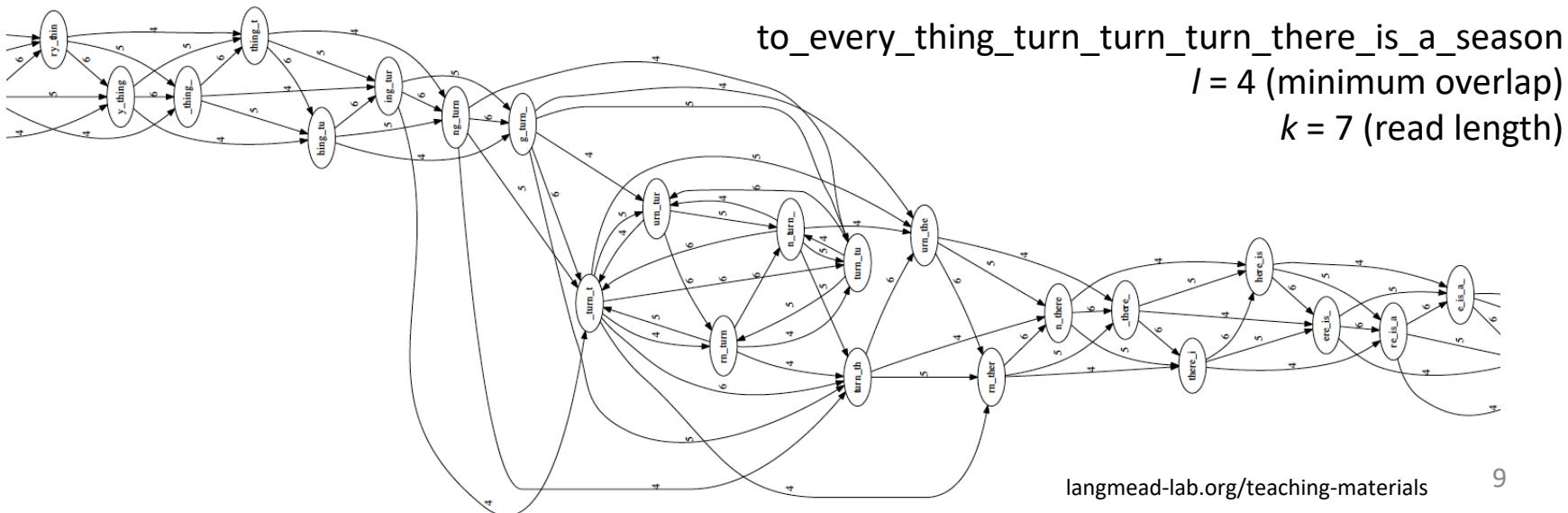
Overlap layout consensus (I)

Overlap

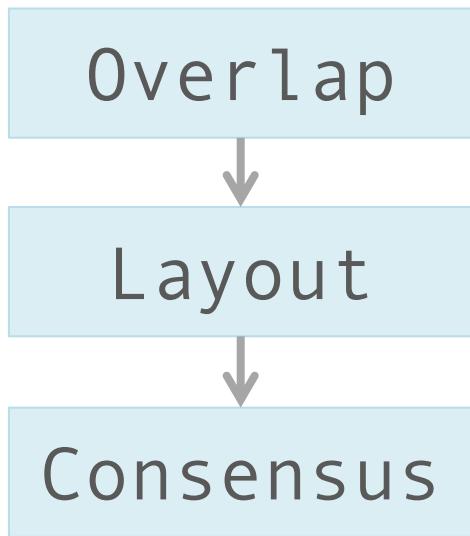
Layout

Consensus

- Overlap graph
- 1. to identify all pairs of reads that overlap **sufficiently well**
- 2. to organize the overlapping information into a graph containing a **node** for every read and an **edge** between any pair of reads that overlap each other.

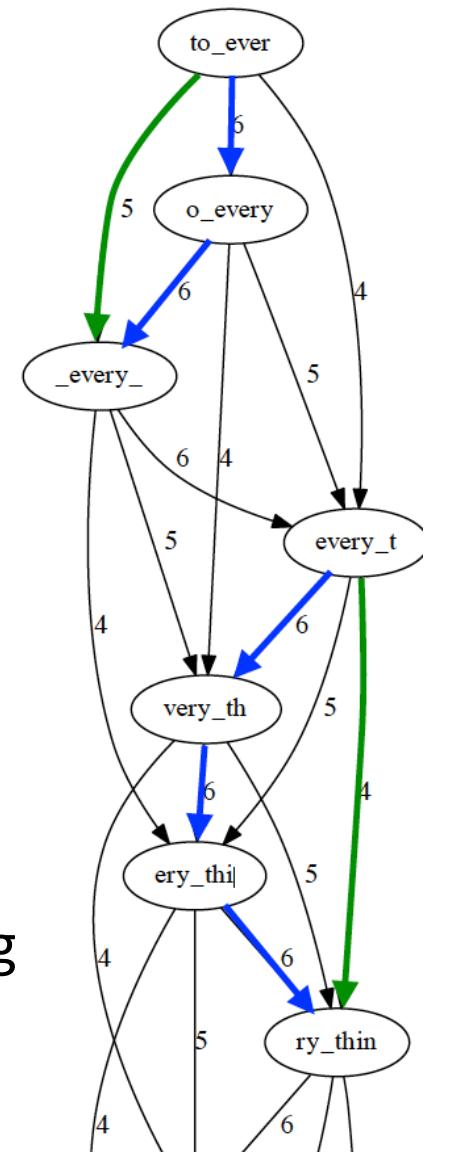
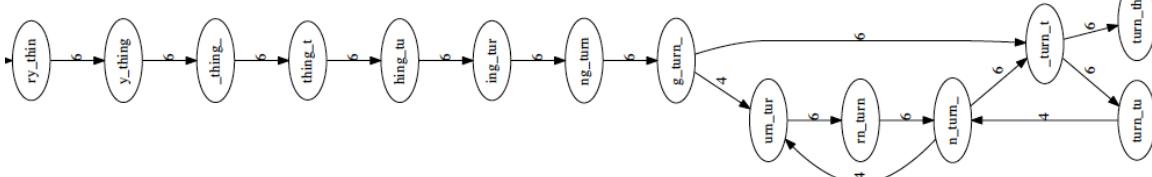


Overlap layout consensus (II)

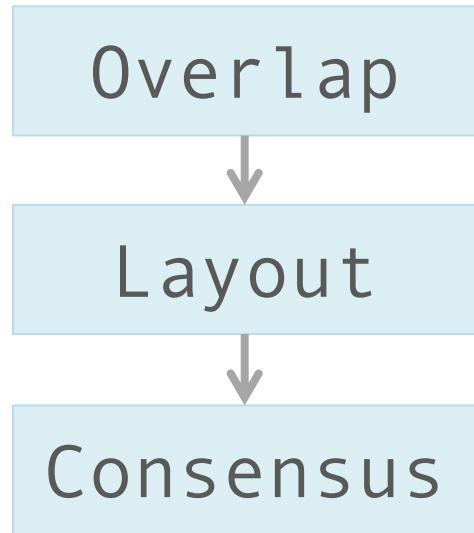


transitive edges: edges that skip one or more nodes

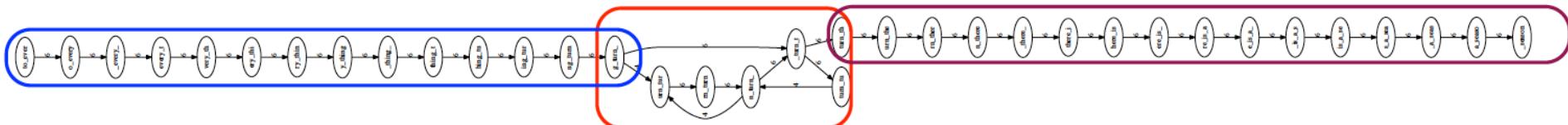
- Layout (**string graph**)
 1. to simplify the global overlap graph by removing redundant information (transitive edges)



Overlap layout consensus (III)



- Layout
 1. Simplifies the global overlap graph by removing redundant information
 2. Emits contigs corresponding to the non-branching paths



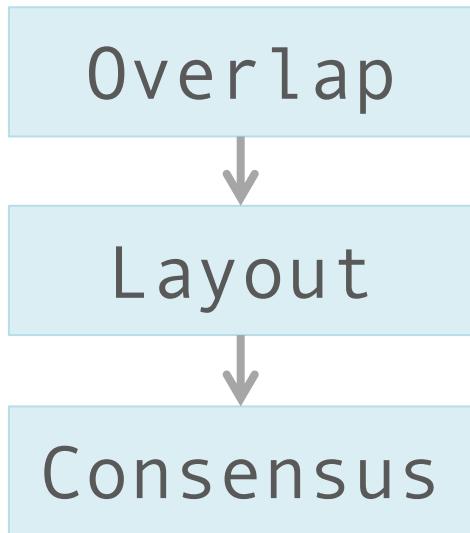
contig 1
to_every_thing_turn_

unresolved repeat (gap)

contig 2
turn_there_is_a_season

to_every_thing_turn_turn_there_is_a_season

Overlap layout consensus (IV)



- Consensus

Extract consensus sequences (major votes) based on the alignment of reads that make up a contig

TAGATTACACAGATTACTGA TTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAACTA
TAG TTACACAGATTATTGACTTCATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA

The diagram shows five lines of DNA sequence, each starting with TAGATTACACAGATTACTGA. The second line has a T in the 11th position highlighted in red. The third line has a T in the 12th position highlighted in red. The fourth line has a C in the 13th position highlighted in red. The fifth line has a G in the 14th position highlighted in red. Red arrows point from the 11th, 12th, 13th, and 14th positions of the first line down to the corresponding positions in the consensus sequence below, indicating the extraction of the consensus sequence.

TAGATTACACAGATTACTGACTTGATGGCGTAA CTA

OLC drawbacks

- Identifying all-to-all overlaps is a slow procedure, especially when read number is millions or billions.
- Overlap graph is big when the read number is huge. One node per read, and in practice the number of edges grows superlinearly with the number of reads.
- The computational complexity limits its application.

De Bruijn graph – k-mer

De Bruijn graph assemblers model the relationship between **exact substrings** of length k (k -mer) extracted from input reads.

($k=3$)

reads A T G G C G T

k-mer ($k=3$)

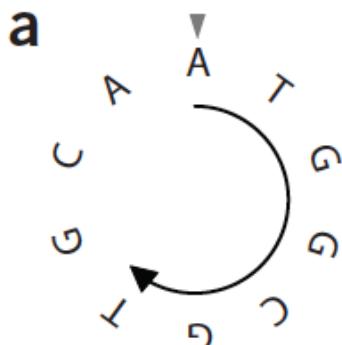
- 1 . ATG
- 2 . TGG
- 3 . GGC
- 4 . GCG
- 5 . CGT

Problem: list all K-mers ($k=4$)

CGATCGATCGATCGAT

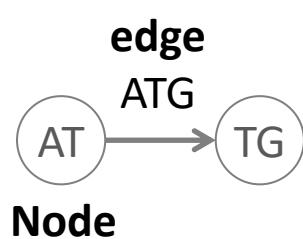
De Bruijn graph - assembly

genome
sequence



reads

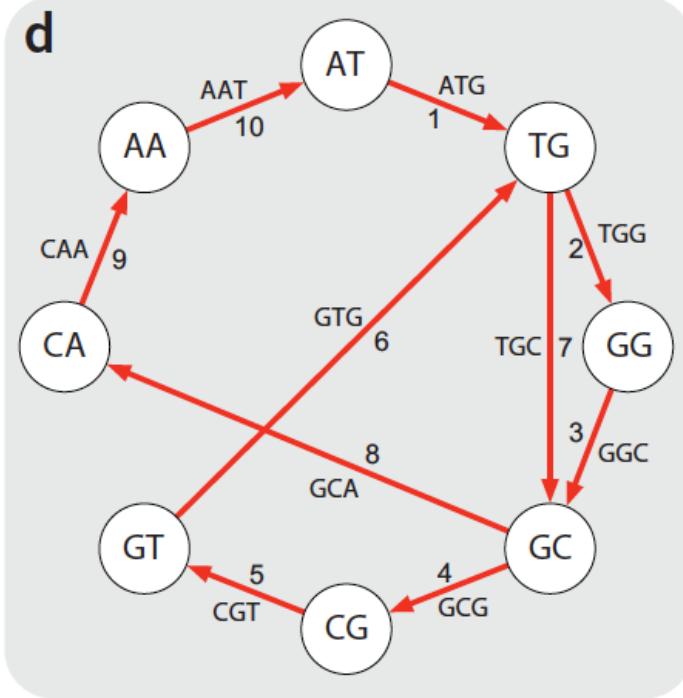
- b
1. ATGGCGT
 2. GGCCTCG
 3. CTTGCAA
 4. TGCAATG
 5. CAATGGC
 6. ATGGCGT



k-mer (k=3)

- c
1. ATG
 2. TGG
 3. GGC
 4. GCG
 5. CGT
 6. GTG
 7. TGC
 8. GCA
 9. CAA
 10. AAT

De Bruijn graph

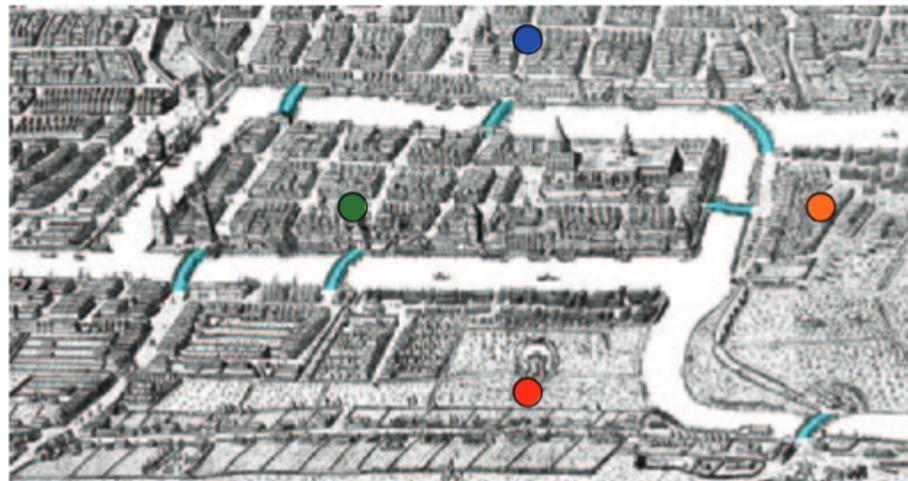


ATGGCGTGCAATG

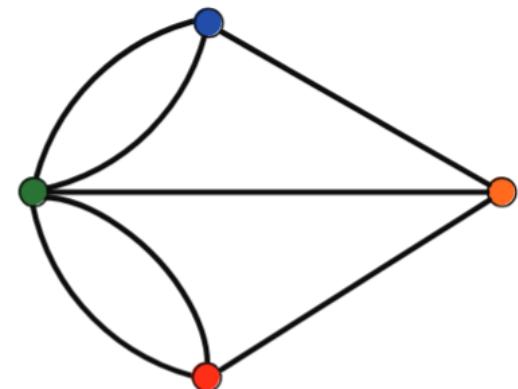
de Bruijn algorithm

- To find a path containing all k-mers but also is as short as possible, as it contains each k-mer exactly once.
- De Bruijn answered this question by borrowing Euler's solution of the Bridges of Königsberg problem. An Eulerian cycle represents a shortest path that contains each k-mer exactly once.

a



b



Sequencing errors complicate assemblies

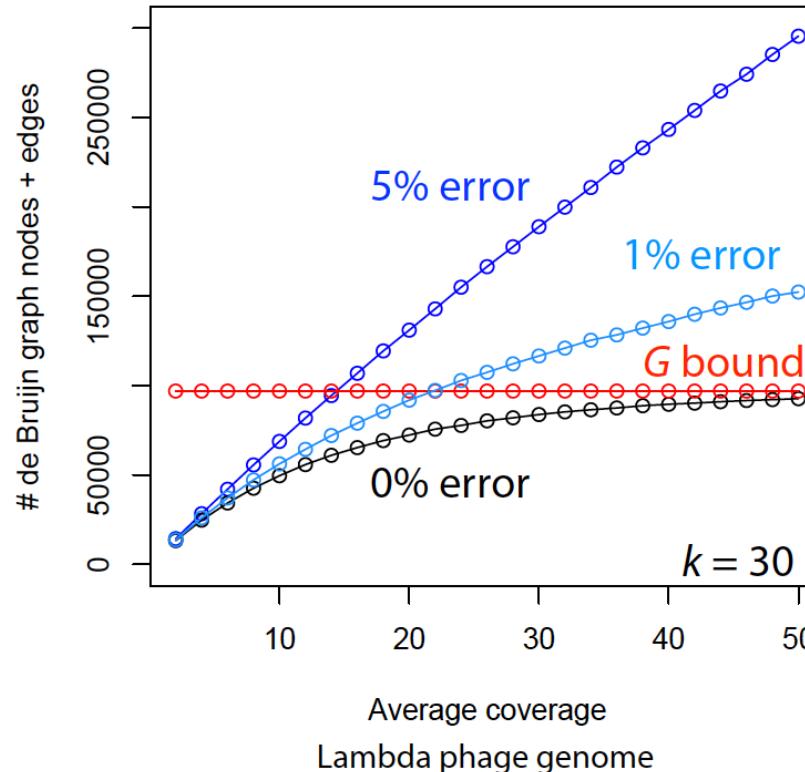
The complexity of De Bruijn graph grows when reads contain errors.

Reads

TGCGTGA
TGC~~G~~TGA
TGC~~G~~TGA
TGGGTGA

k-mer count profile ($k=5$)

TGCGT
GCGTG
CGTGA
TGGG
T
GGGT
G
GGTG
A



G: genome

langmead-lab.org/teaching-materials

Sequence error correction before graph construction is a critical step for De Bruijn graph assembly.

Error identification

Reads (depth = 10)

TGCGTGATAACG

TGCGTGATAACG

TGGGTGATAACG

TGCGTGATAACG

TGCGTGATAACG

TGCGTGATAACG

TGCGTGATAACG

TGCGTGATAACG

TGCGTGATAACG

TGCGTGATAACG

k-mer count profile (k=5)

TGGGT1

GGGTG1

GGTGA1

TGCGT9

GCGTG9

CGTGA9

GTGAT10

TGATA10

GATAC10

ATACG10

Error identification

Reads (depth = 10)

TGCGTGATAACG

TGCGTGATAACG

TGGGTGATAACG

TGCGTGATAACG

TGCGTGATAACG

TGCGTGATAACG

TGCGTGATAACG

TGCGTGATAACG

TGCGTGATAACG

TGCGTGATAACG

k-mer count profile (k=5)

TGGGT 1

GGGTG 1

GGTGA 1

TGCGT 9

GCGTG 9

CGTGA 9

GTGAT 10

TGATA 10

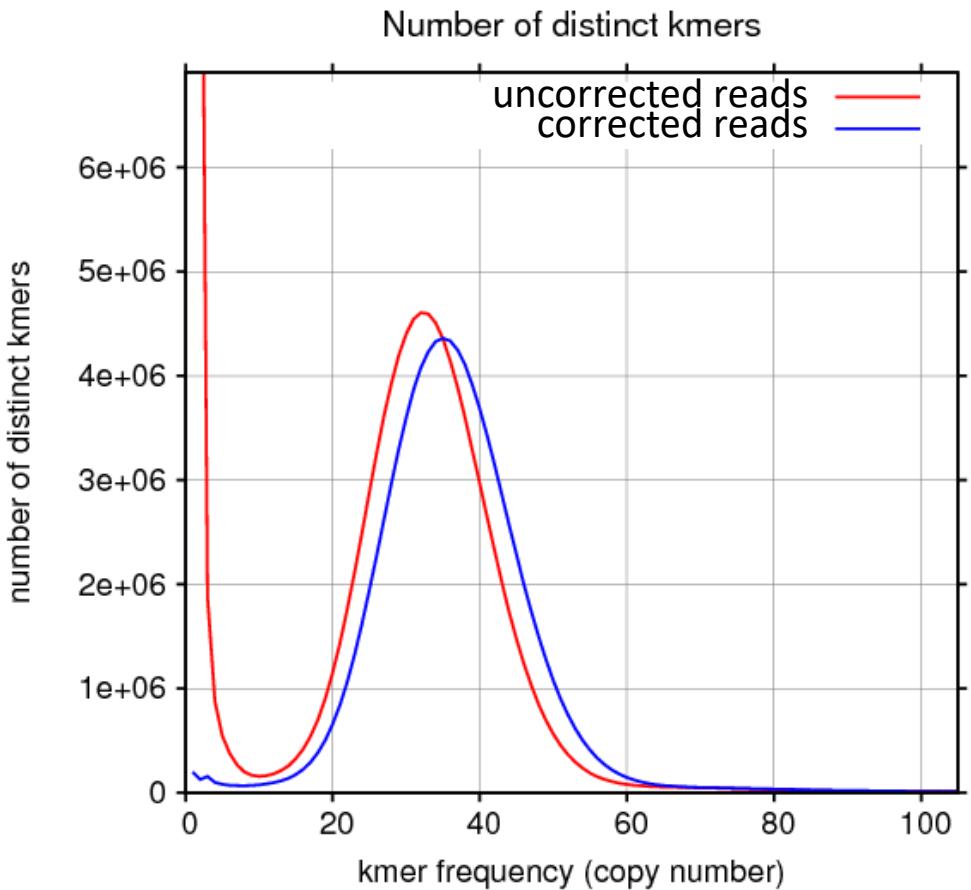
GATAC 10

ATACG 10

k-mer count profile indicates where errors are.

Error correction

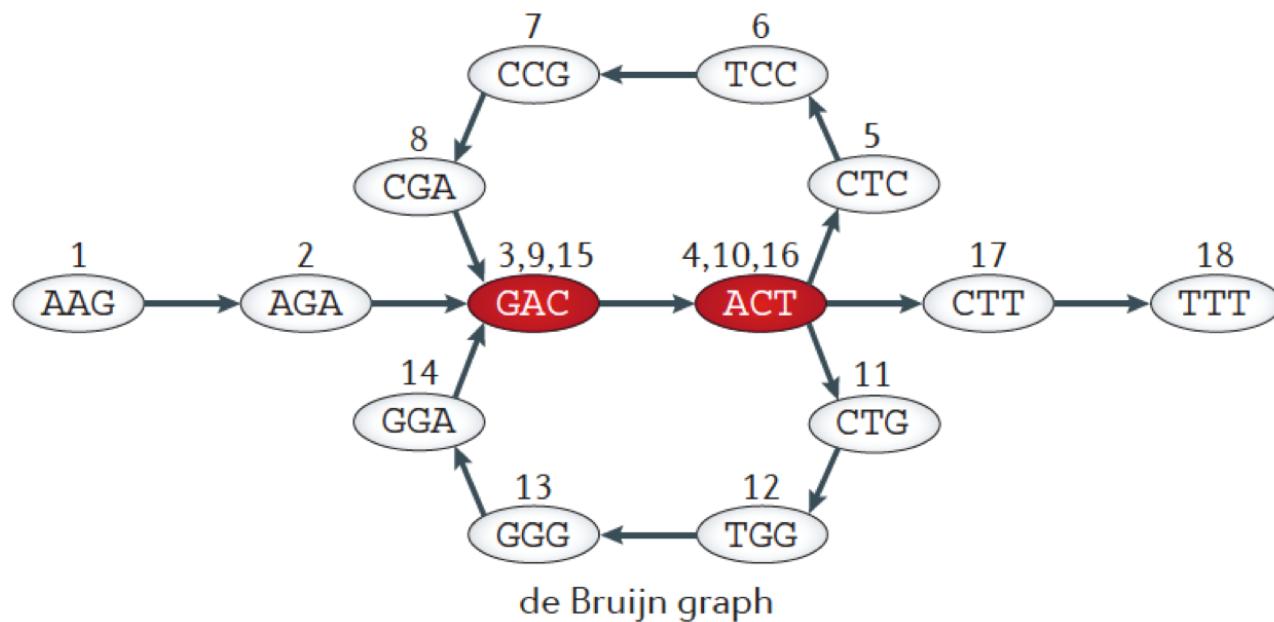
Essentially, the error correction algorithm removes k-mers with a few copies and correct reads carrying these errors.



from Allpaths-LG

Repeats

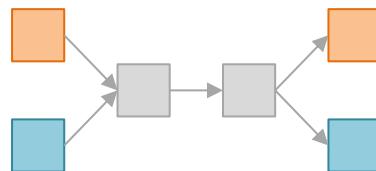
AGAC	CCGT	CTCC	CTTT
....	AAGA	GGAC
GGGA	GACA	CCGA	CTGG



AAGACTCCGACTGGGACTTT

Unresolved repeats produce gaps

Long repeats, especially for those whose lengths are longer than read length, and high-copy repeats are challenging to resolve. If no other information can be used to assist in resolving repeats, gaps will be introduced.



What strategies can be used to resolve repeats?

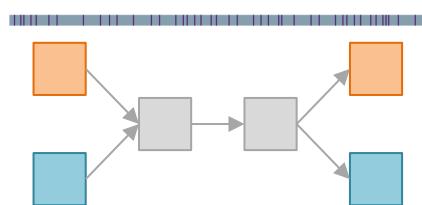
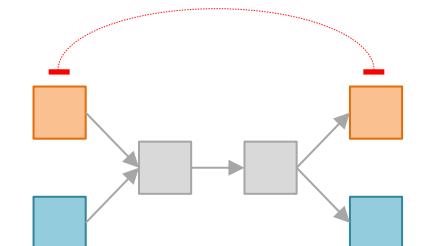
Some strategies to resolve repeats

Mate-pair reads, long reads (e.g., PacBio), genetic map, Hi-C data, and physical map (BioNano) can resolve some repeats

Illumina mate-pair (MP) reads



PacBio long reads (average 4-8 kbp, longest over 30 kbp)



Miller et al., 2010. Genomics 95: 315-327

Gaps due to other reasons

- Low sequence coverage (depth)
- Sequencing biases (e.g., regions recalcitrant to sequencing)

CGGATCTGCGT**G**ATACGGAATAGCCTAGCA
GATCTGCGT**G**ATACGGAAT
ATCTGCGT**G**ATACGGAATA
TCTGCGT**G**ATACGGAATAG
CTGCGT**G**ATACGGAATAGC
TGC**G**TGATACGGAATAGCC
CGT**G**ATACGGAATAGCCT
6 (coverage; depth)

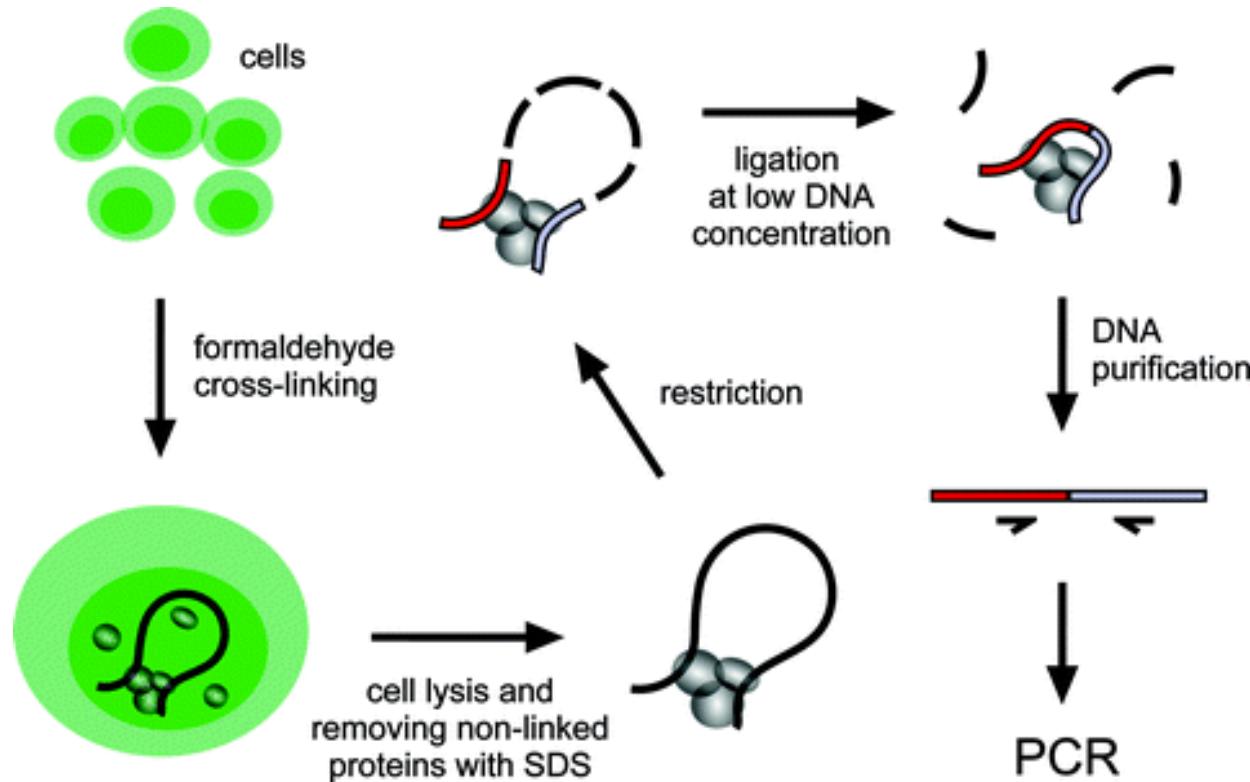
Recalcitrant genomic regions:
1. extremely high GC or AT
2. complicated secondary
structure

Preferred genome sequencing reads:

1. Long reads
2. High-quality reads
3. High sequencing depth (at least 30x)
4. No sequencing biases

Hi-C: the method to quantify physical interactions between all possible pairs of genomic regions simultaneously

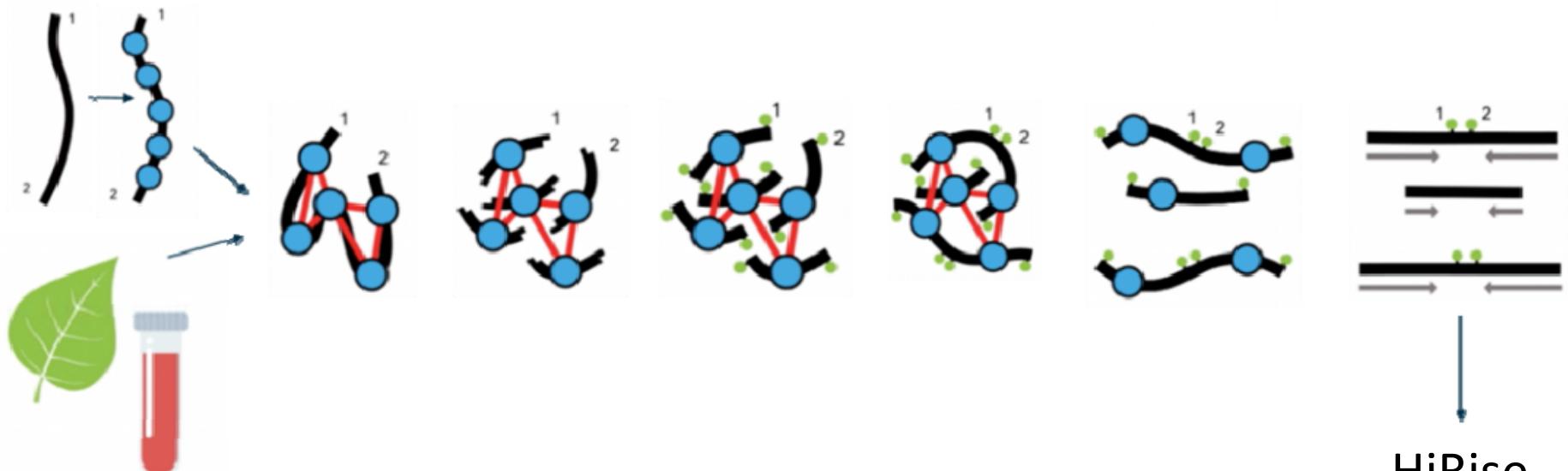
Inside a cell, genomic regions in a chromosome are physically connected to form chromosome conformation. **Closer genomic regions generally have higher frequent physical contacts than regions far apart in distance.**



MMB, 567:171-188

Chicago + Dovetail Hi-C

Chicago generated libraries starting from pure DNA that is reconstituted into chromatin



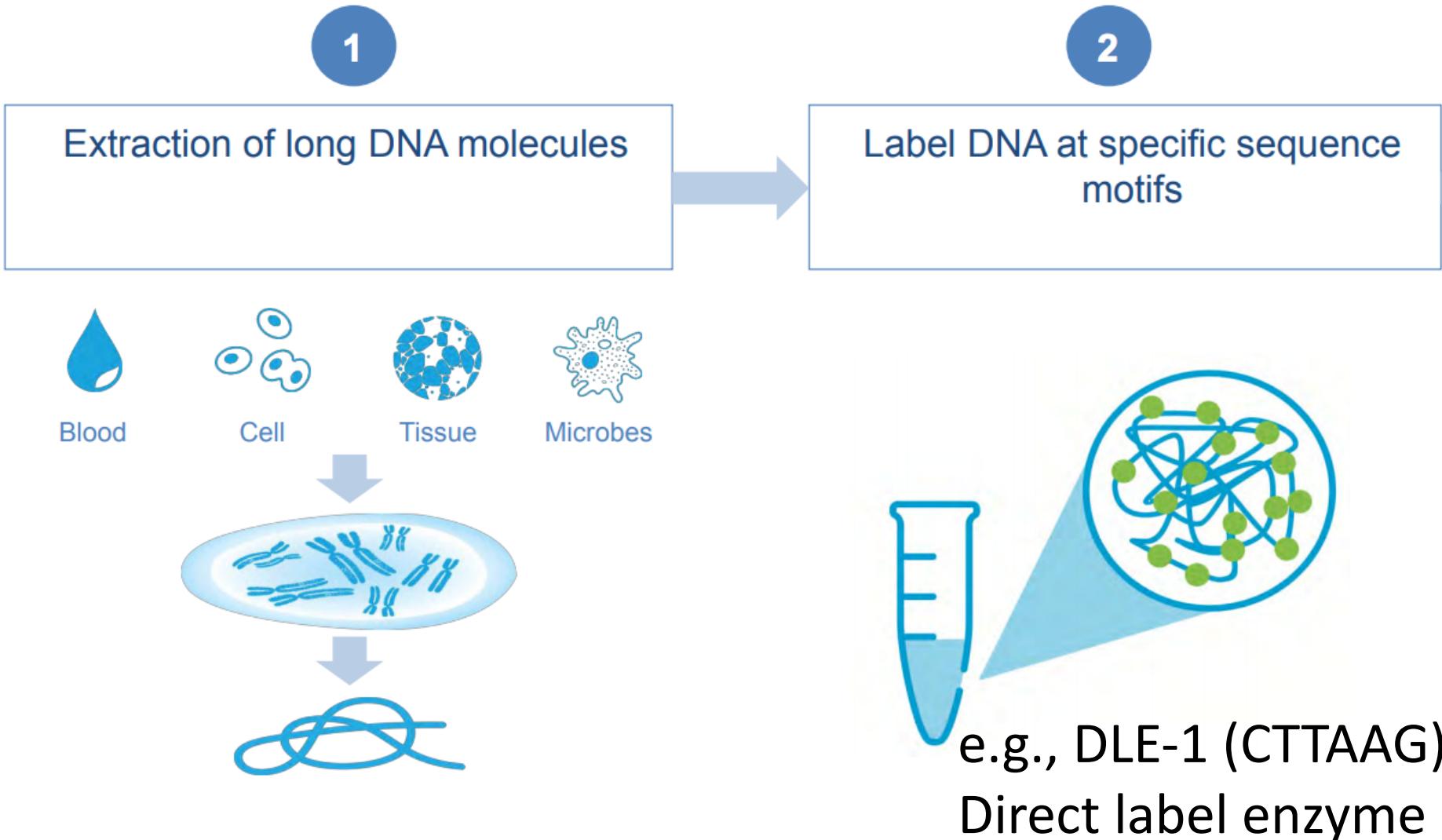
Dovetail Hi-C generated libraries starting from tissue or cell culture and endogenous chromatin

HiRise
scaffolding
pipeline

<https://youtu.be/OIxFFB3dalM>

<https://www.biorxiv.org/content/biorxiv/early/2018/08/15/392126.full.pdf>

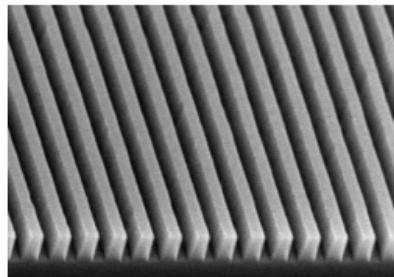
BioNano – library preparation



BioNano – running on Saphyr

3

Saphyr Chip linearizes DNA in NanoChannel arrays



4

Saphyr automates imaging of single molecules in NanoChannel arrays



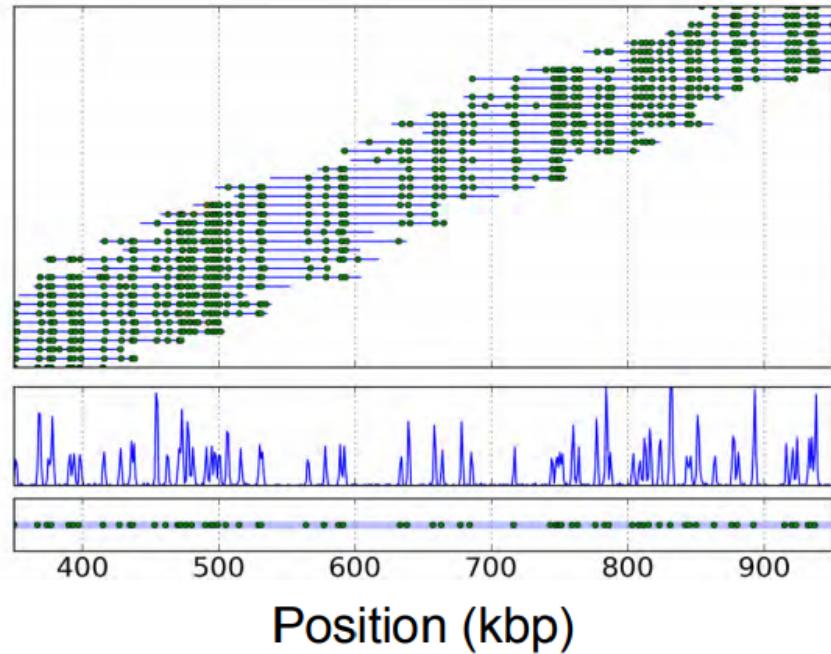
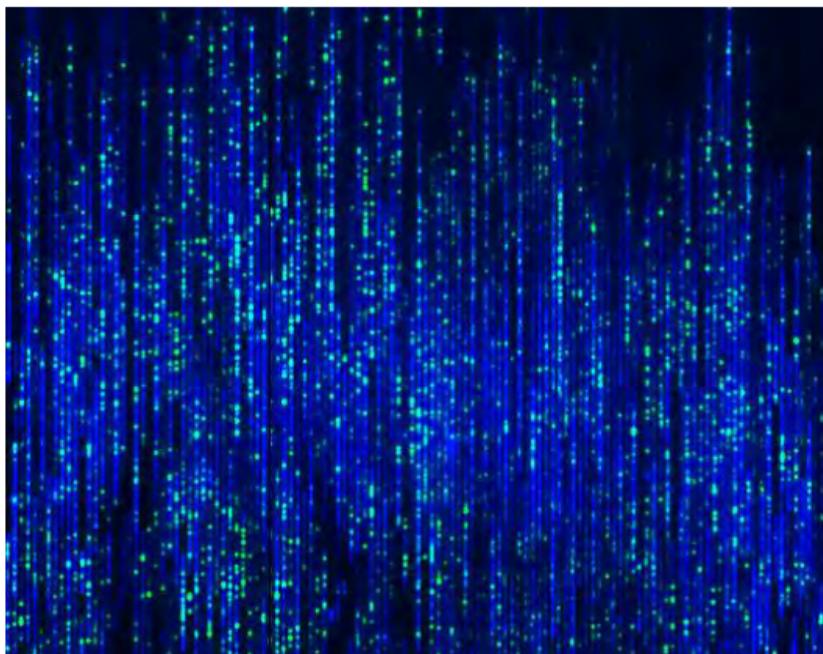
BioNano – image analysis and assembly of molecular data

5

Molecules and labels detected in images

6

Bionano Access software assembles optical maps



BioNano scaffolding

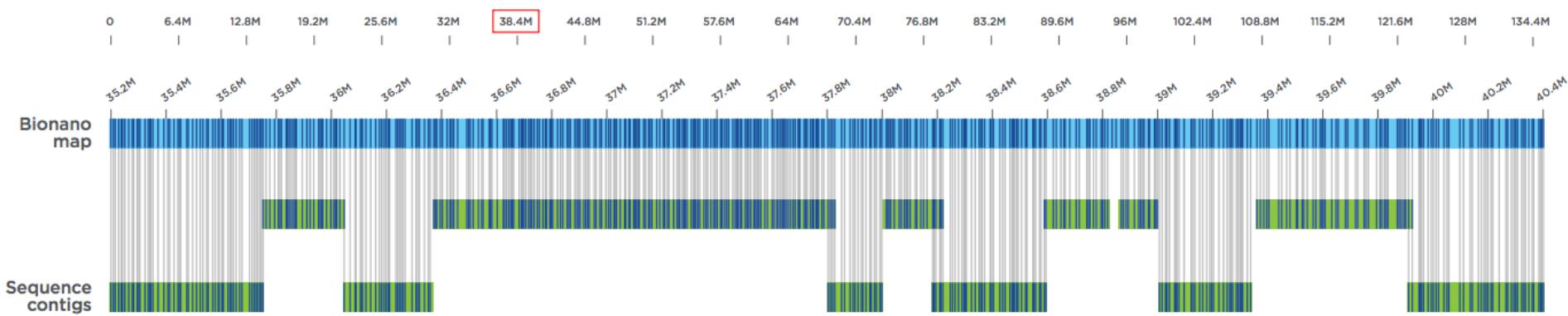


Figure 1:

Combining Bionano maps with sequence assemblies. Sequence contigs are anchored and oriented using the de novo generated Bionano maps.

Assembly statistics – N50 and L50

- N50: A statistic used for assessing the contiguity of a genome assembly.
- L50: The number of contigs whose summed length is N50.
- The contigs in an assembly are sorted by size and added, starting with the largest. The contig N50 is the size of the contig that makes the total greater than or equal to 50% of the total contig size.
- **OR:** The contig N50 is the length of the smallest contig in the set that contains the fewest (largest) contigs whose combined length represents at least 50% of the assembly.

An example to determine N50 and NG50

- assembly: 20 contigs

contig N50? 360 kbp

ctg_ID	ctg_size (kbp)	Accumulated (kbp)	% ASM
1	615	615	13%
2	515	1,130	25%
3	512	1,642	36%
4	450	2,092	46%
5	360	2,452	53%
6	356	2,808	61%
7	310	3,118	68%
8	201	3,319	72%
9	189	3,508	76%
10	186	3,694	80%
11	160	3,854	84%
12	150	4,004	87%
13	120	4,124	90%
14	102	4,226	92%
15	95	4,321	94%
16	86	4,407	96%
17	82	4,489	98%
18	54	4,543	99%
19	32	4,575	100%
20	21	4,596	100%
Total	4,596		

Assembly statistics – NG50

Problem of N50:

N50 values of the assemblies with significantly different total assembly space (lengths) are usually not comparable. Even if the same data set is used for the assembly.

50,000	50,000
30,000	40,000
10,000	30,000
	20,000
	10,000

NG50: The NG50 statistic is the same as N50 except that it is 50% of the known or estimated genome size. This allows for meaningful comparisons between different assemblies.

An example to determine N50 and NG50

- Genome size: 5 Mbp
- 20 contigs

contig N50 360 bp
contig NG50? 356 kbp

ctg_ID	ctg_size (kbp)	Accumulated (kbp)	% ASM	% genome
1	615	615	13%	12%
2	515	1,130	25%	23%
3	512	1,642	36%	33%
4	450	2,092	46%	42%
5	360	2,452	53%	49%
6	356	2,808	61%	56%
7	310	3,118	68%	62%
8	201	3,319	72%	66%
9	189	3,508	76%	70%
10	186	3,694	80%	74%
11	160	3,854	84%	77%
12	150	4,004	87%	80%
13	120	4,124	90%	82%
14	102	4,226	92%	85%
15	95	4,321	94%	86%
16	86	4,407	96%	88%
17	82	4,489	98%	90%
18	54	4,543	99%	91%
19	32	4,575	100%	92%
20	21	4,596	100%	92%
Total	4,596			

Software for short-read assemblies

Software	Description	URL and reference
Velvet	Original de Bruijn graph assembler	http://github.com/dzerbino/velvet
SOAPdenovo	De Bruijn graph assembler with error-correction step	http://soap.genomics.org.cn/
Meraculous	Hybrid k-mer/read-based	https://jgi.doe.gov/data-and-tools/meraculous/
ALLPATHS-LG	Uses unipath graph to collapse repeats	http://software.broadinstitute.org/allpaths-lg/blog/
SGA	Uses string graphs	https://github.com/jts/sga
ABySS	Represents de Bruijn graph with a Bloom filter	https://github.com/bcgsc/abyss
DISCOVAR de novo	Requires 250-bp PCR-free reads	https://software.broadinstitute.org/software/discover/blog/
Supernova	Assembles 10× linked reads	https://github.com/10XGenomics/supernova

Long-read assemblies and polishing

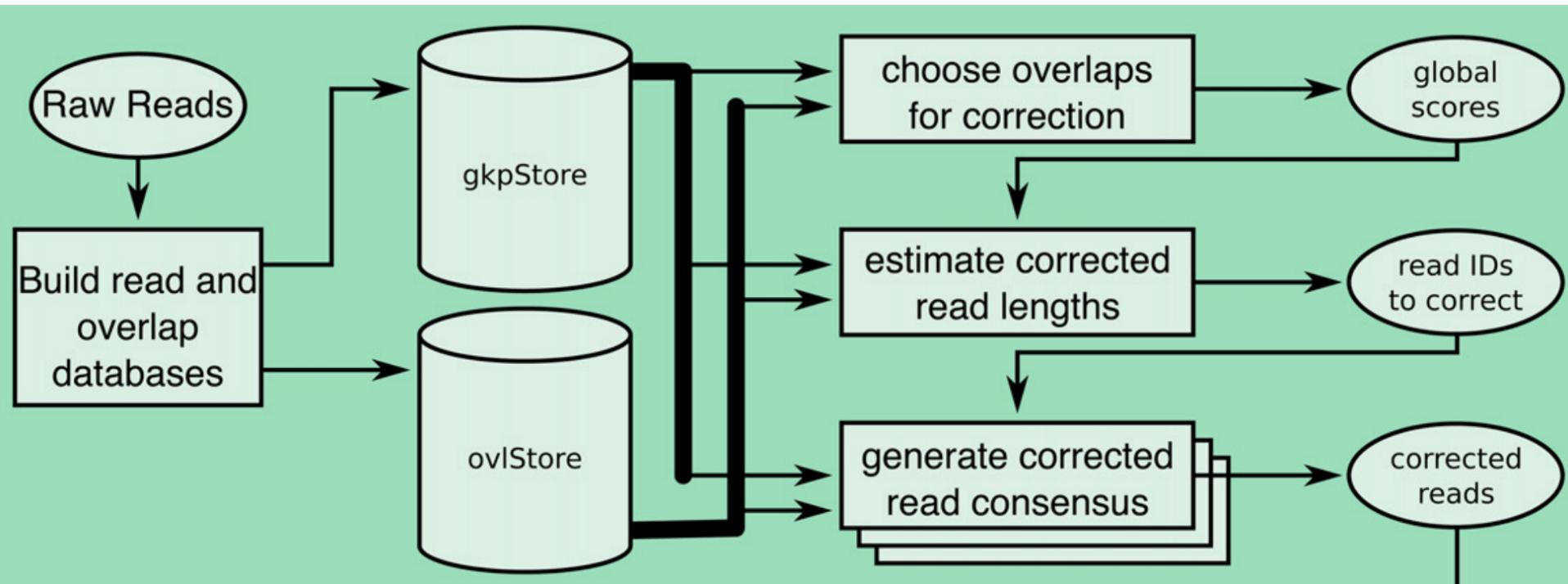
Long-read assemblies

Software	Description	URL and reference
HGAP	Error correction, OLC assembly, and polishing	https://github.com/PacificBiosciences/Bioinformatics-Training/wiki/HGAP
Canu	K-mer-based overlap computation	https://github.com/marbl/canu
FALCON	Assembles phased diploid genomes	https://github.com/PacificBiosciences/FALCON
Flye	Uses A-Brujin graph	https://github.com/fenderglass/Flye
Miniasm	Fast, but no error correction	https://github.com/lh3/miniasm

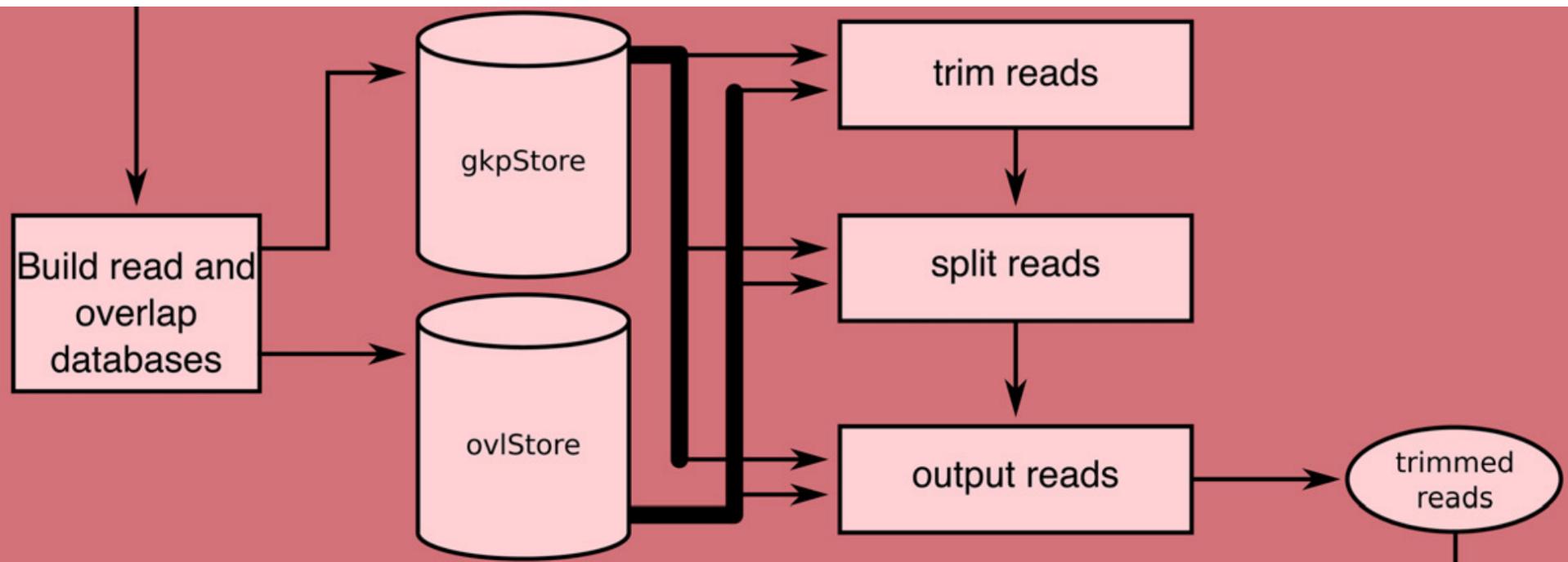
Polishing

Software	Description	URL and reference
Pilon	Uses short-read alignments to correct errors	https://github.com/broadinstitute/pilon
Arrow	Hidden Markov model and long-read alignments	https://github.com/PacificBiosciences/GenomicConsensus
Nanopolish	Nanopore only; uses voltage data to correct errors	https://github.com/jts/nanopolish

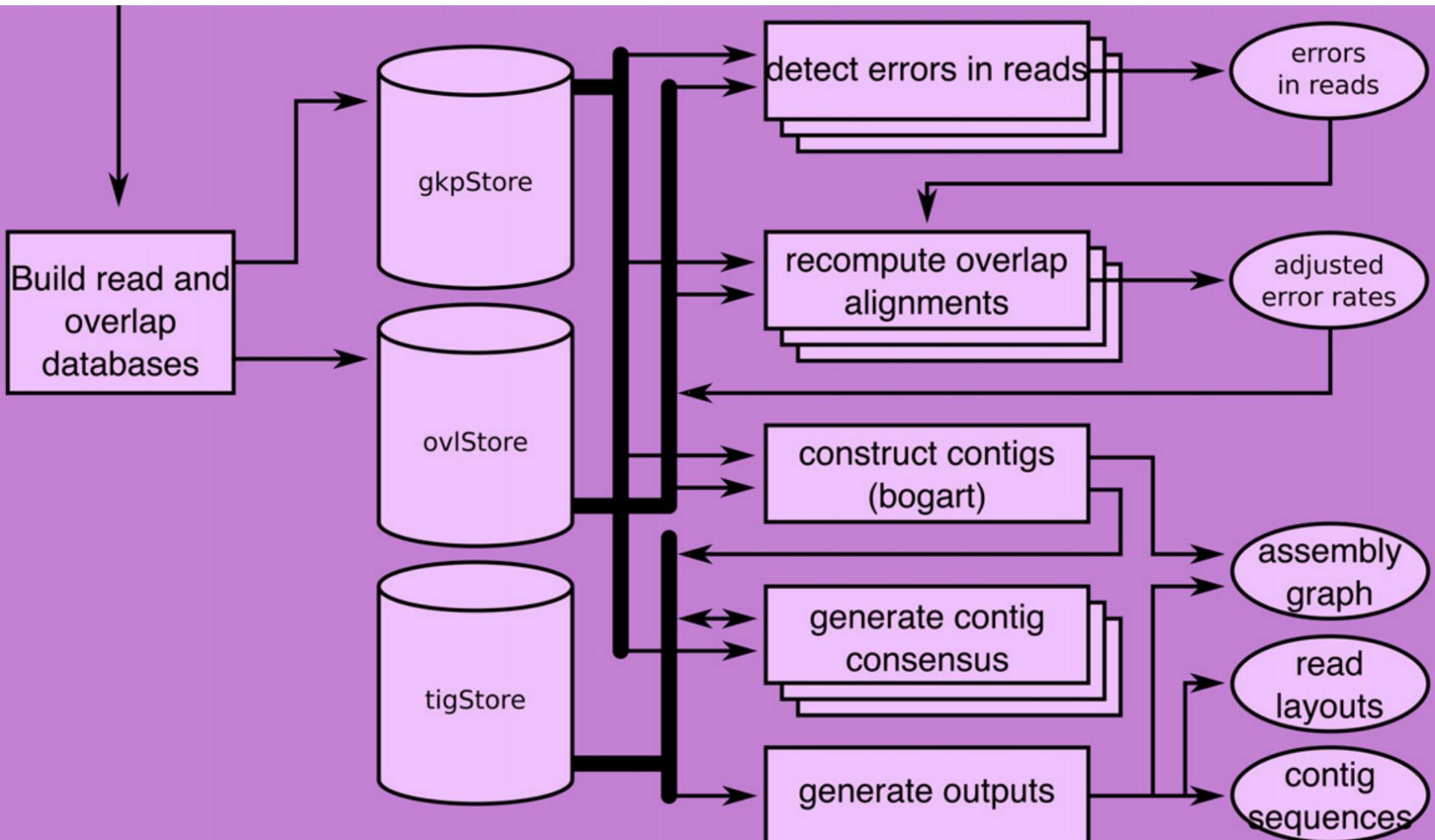
Canu - read correction



Canu - read trimming



Canu - assembly (contigs and consensus)



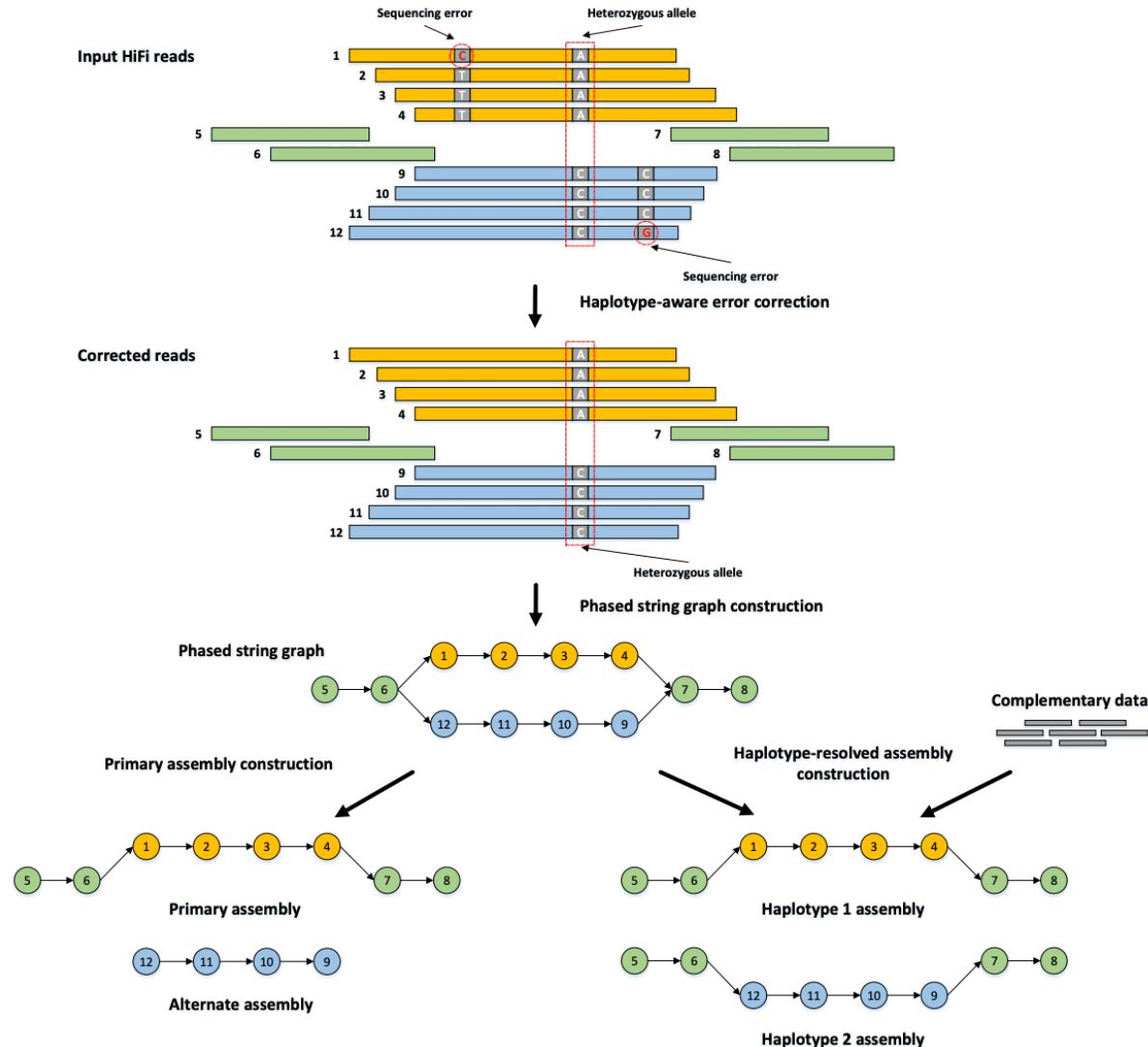
Assembly meets long HiFi PacBio reads

Assembly with hifiasm

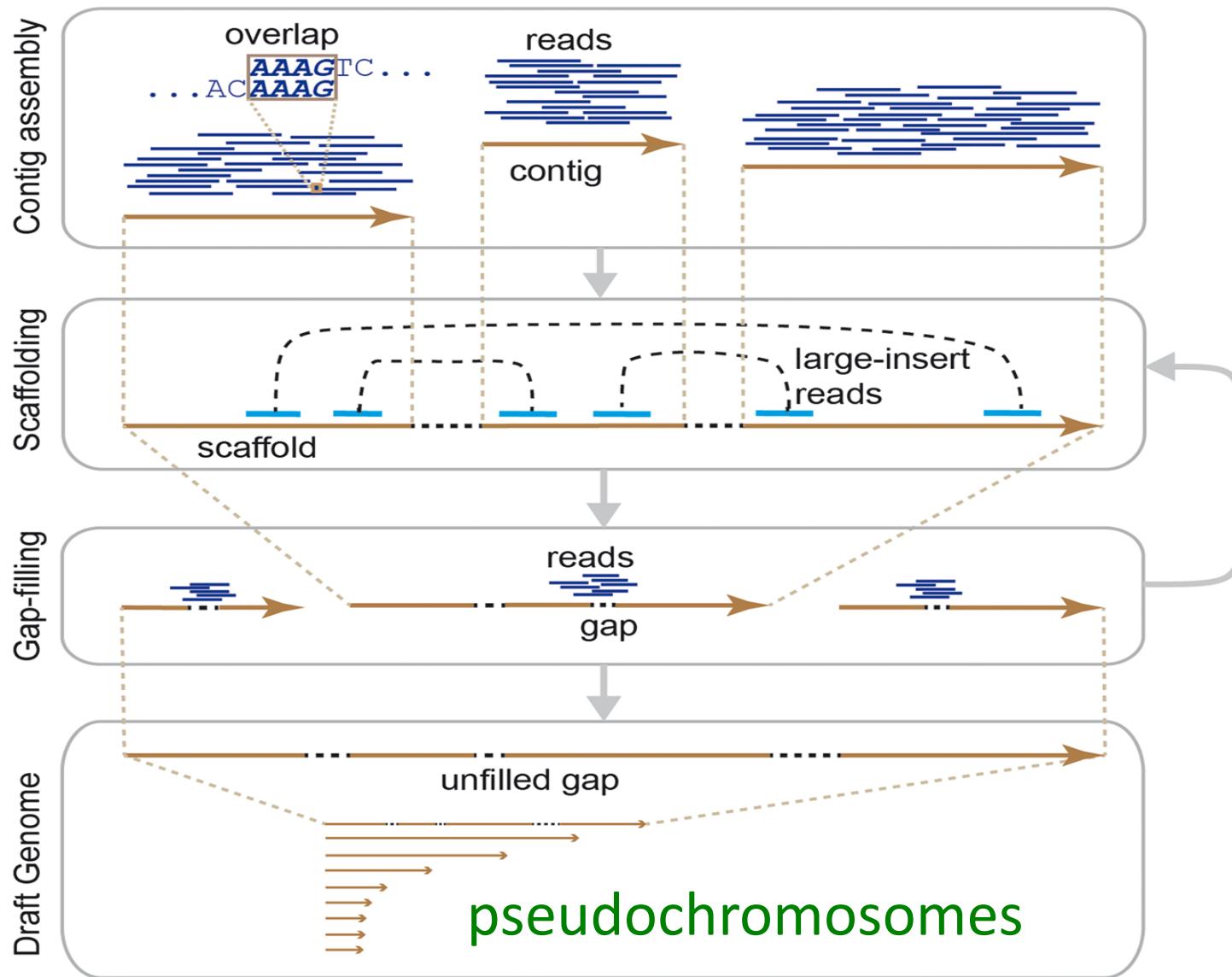
Dataset	Size	Coverage	CPU time	RAM	N50
<u>Maize (B73)</u>	2.2Gb	×22	203.2h	68G	36.7Mb
<u>Human (CHM13)</u>	3.1Gb	×32	310.7h	114G	88.9Mb

<https://github.com/chhylp123/hifiasm>

Haplotype-resolved *de novo* assembly (Phased assembly using hifiasm)



General workflow of *de novo* assemblies



Assemblies are hypotheses

... there will always be *some degree of error* in the characterized genome sequence, both on the level of individual nucleotides and in the ordering of sequence blocks. ..., every genome assembly is the result of a series of assembly heuristics and should accordingly be treated as a **working hypothesis**.

Assembly evaluation

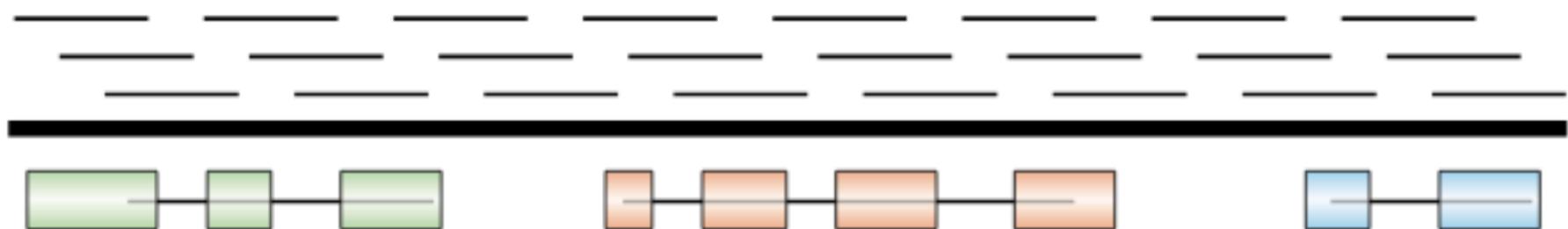
1. No. of contigs: the total number of contigs in the assembly.
2. Largest contig: the length of the largest contig in the assembly.
3. Total length: total assembly length
4. N_{xx} (N₅₀), NG_{xx} (NG₅₀)
5. Using a reference genome, No. of misassemblies, No. INDEL per 100 kbp, and No. of mismatches per 100 kbp are reported
6. Evaluation with BUSCO

Outline

- Genome assembly: concept
- Assembly algorithms: OLC/De Bruijn graph
- Error correction (Kmer counts)
- Assembly strategy to cope with the repeat problem
(mate-pair reads, long reads, others)
- Assembly evaluation (N50, comparison to a reference genome)
- **Genome annotation**

Gene annotation of eukaryotic genomes

Nucleotide-level annotation

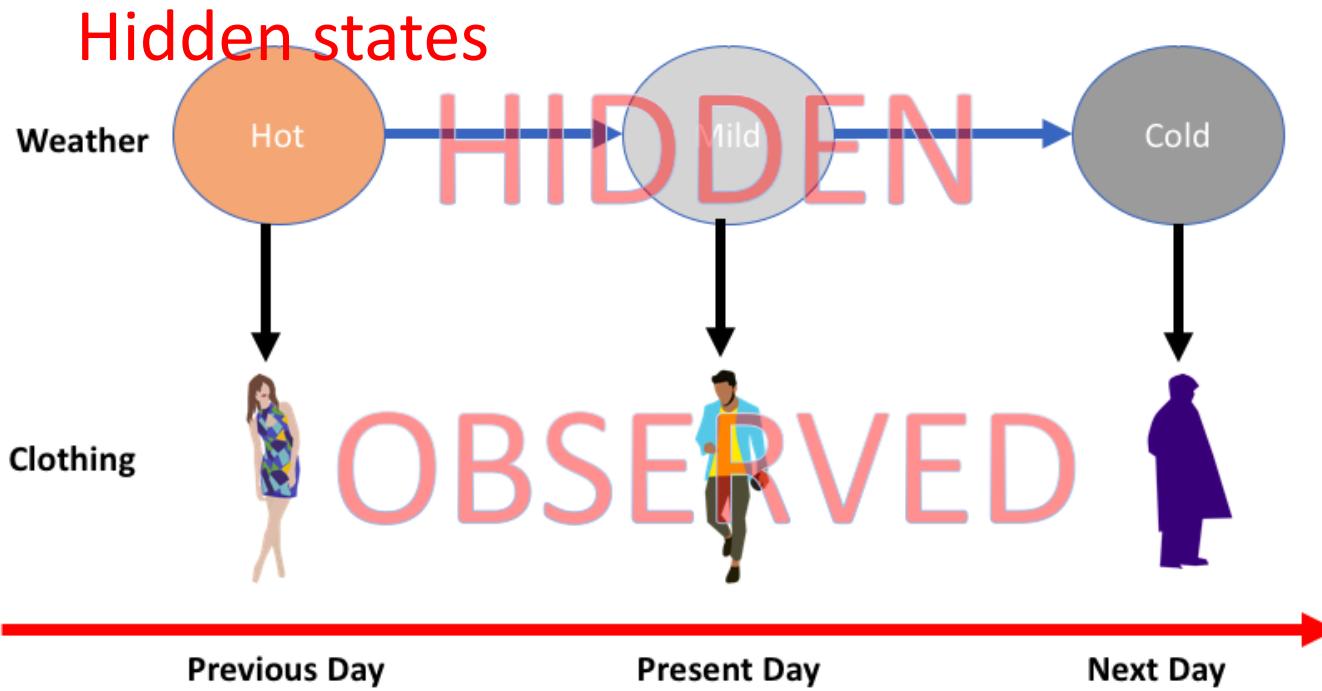


Gene prediction from genomic DNA sequences:
to predict the ***exon-intron*** structures of the protein-encoding portions of transcripts (open reading frames or ORFs) and ***untranslated regions*** (UTR).

Gene prediction approaches

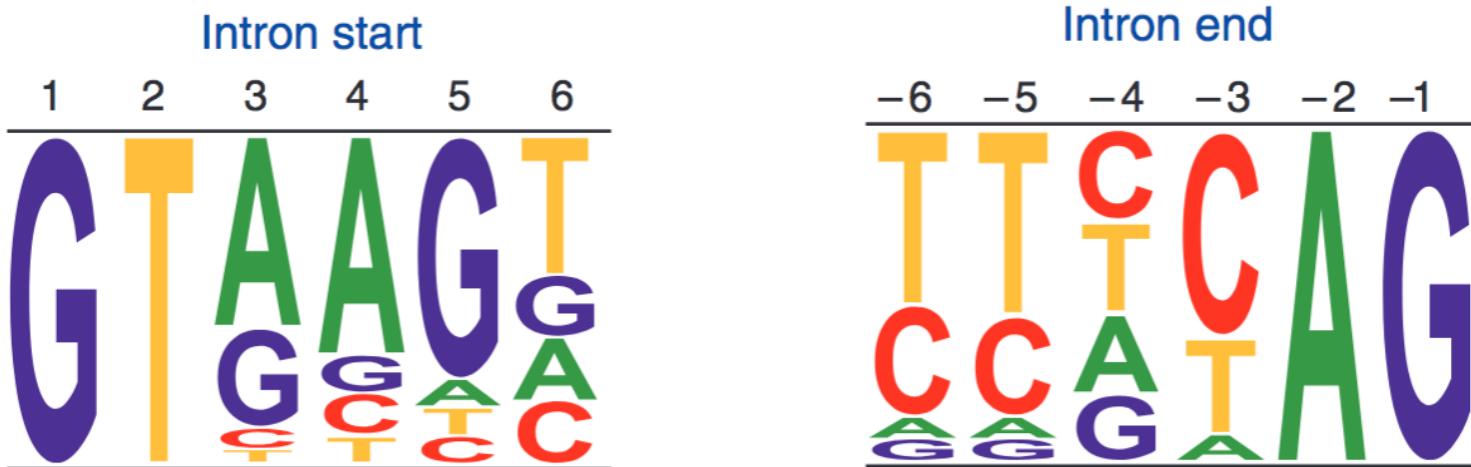
- ***ab initio* gene prediction** (Hidden Markov Model) to assign probability scores to many potential coding exons, and then join consecutive high-scoring exons with consistent reading frames to form high-scoring *exon–intron structures*
- **Evidence-driven gene prediction** (e.g., RNA sequencing data)

Hidden Markov Model (HMM)



HMM is a class of probabilistic graphical model that is used to predict a sequence of *unknown* (hidden) variables from *observed* variables.

Exon-intron structure



Sequence logos represent weight matrices for the first six bases of an intron (left) and the last six bases of an intron (right). In plants and animals, ~99% of introns begin with GT.

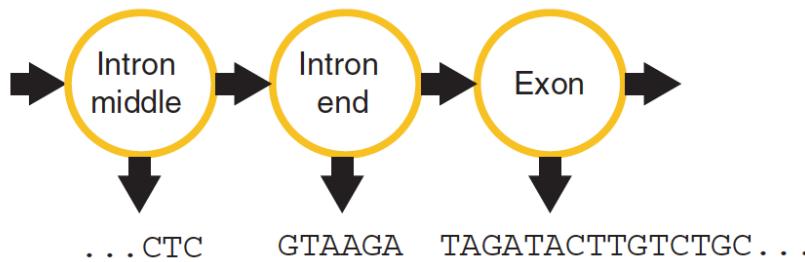


Other genomic sequence patterns: translation initiation and termination sites.

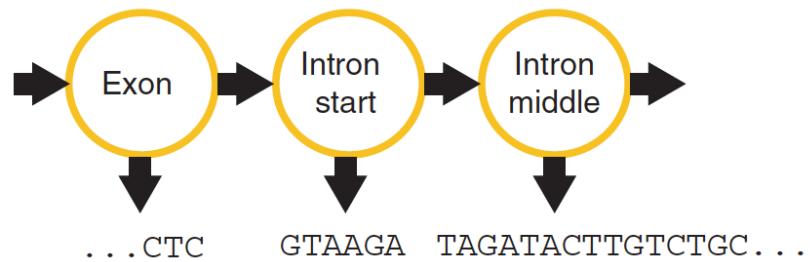
Generalized Hidden Markov Model (GHMM)

b

Less likely hypothesis

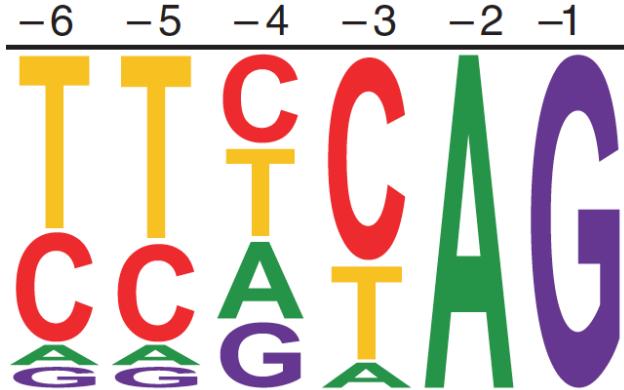


More likely hypothesis

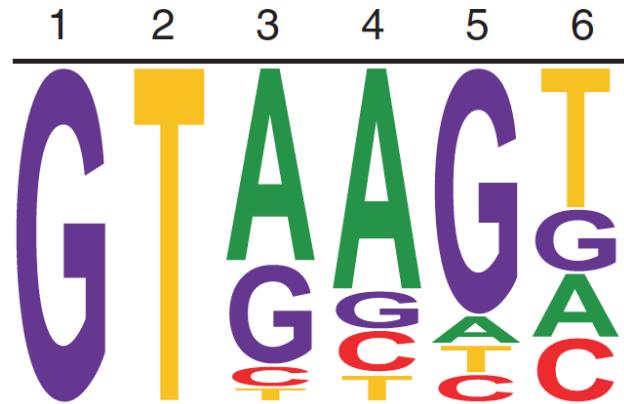


c

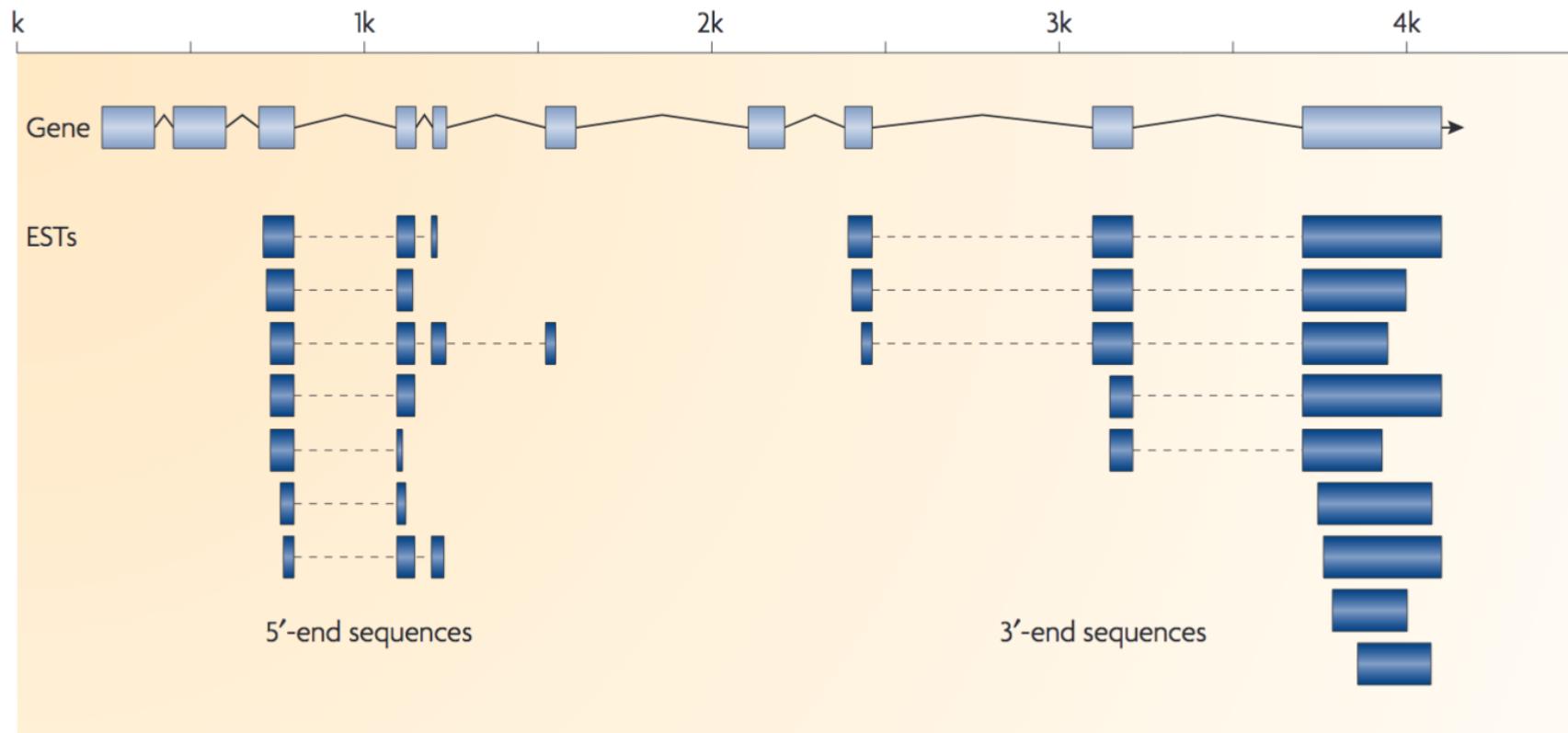
Intron end



Intron start



Evidence-driven approaches



Ideally, full-length cDNA sequences of all genes are collected for evidence-based genome annotation.

evidence and limitations

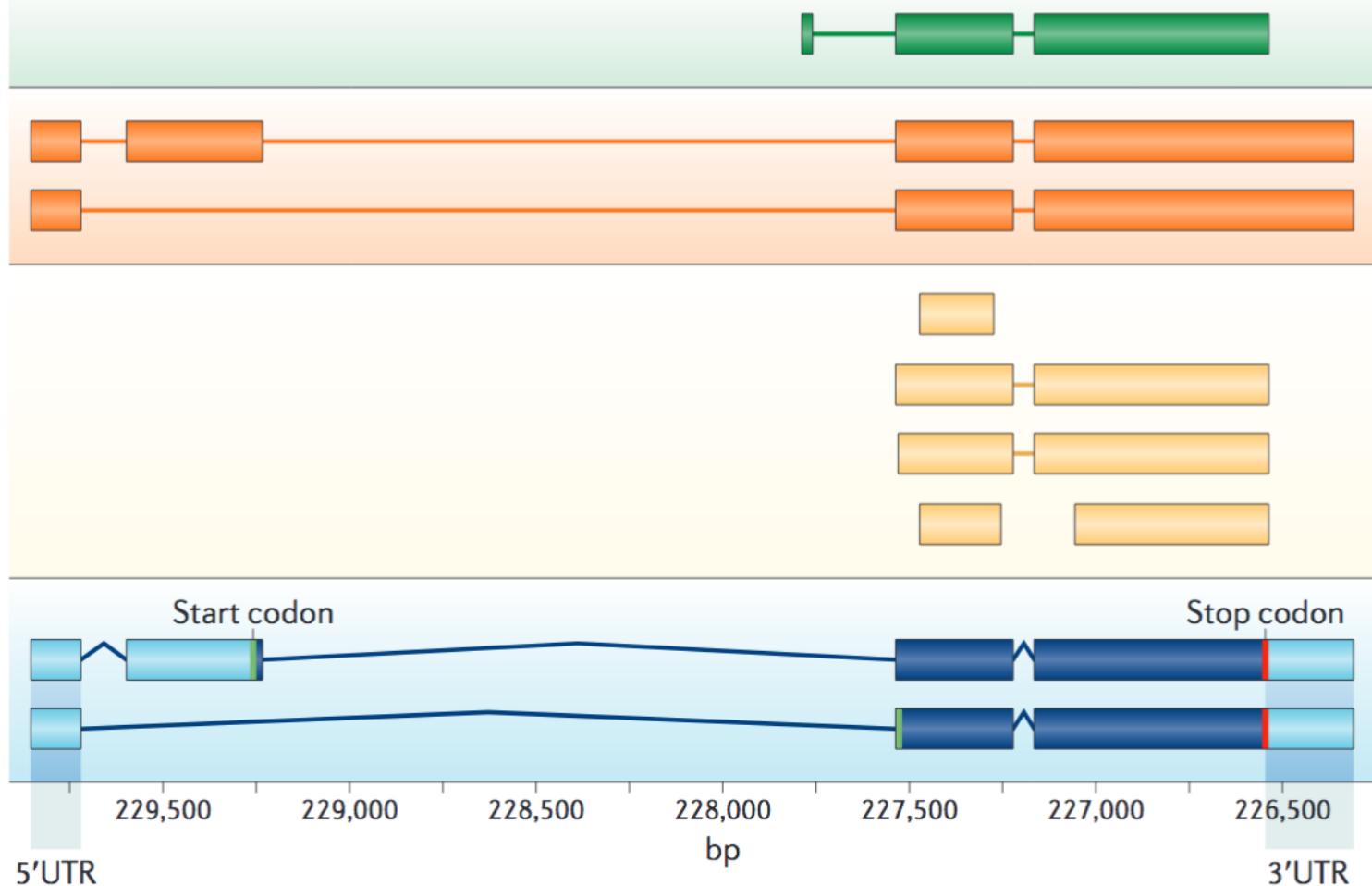
- Full-length cDNAs
- Protein sequences
- Expression sequence tags (ESTs)
- RNA-Seq (de novo assembly and alignment-based)

Limitations:

- Full-length cDNA databases are not available
- Not all genes are transcribed at sufficient levels at certain conditions
- Alignments produce errors

annotation: *ab initio* prediction + evidence

Gene prediction
(SNAP)



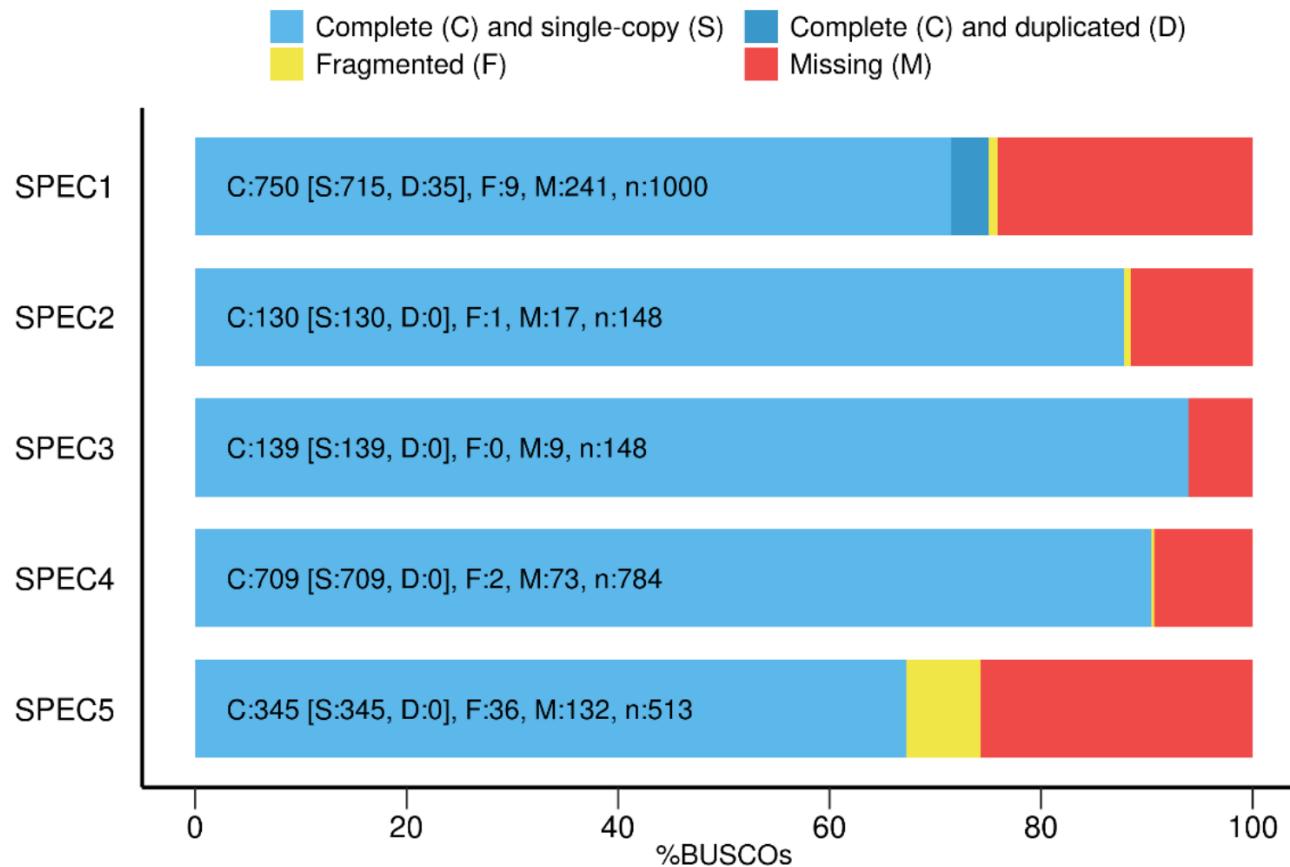
Question?

What method can you figure out to assess the completeness of a genome annotation?

Evaluation with BUSCO

Assessing genome assembly and annotation completeness with Benchmarking Universal Single-Copy Orthologs.

BUSCO Assessment Results



Outline

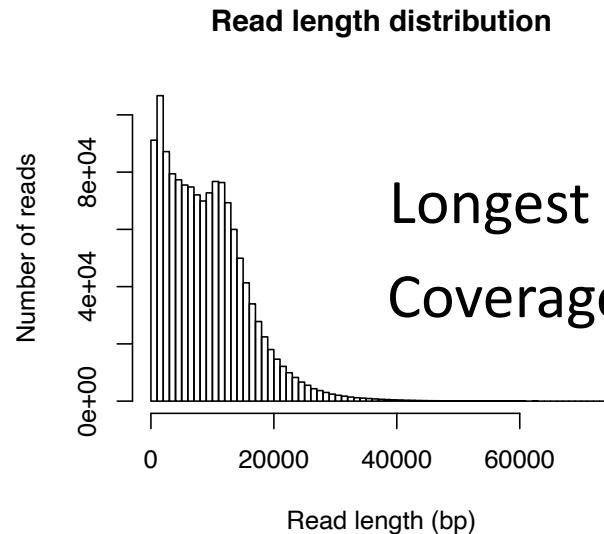
- Genome assembly: concept
- Assembly algorithms: OLC/De Bruijn graph
- Error correction (Kmer counts)
- Assembly strategy to cope with the repeat problem
(mate-pair reads, long reads, others)
- Assembly evaluation (N50, comparison to a reference genome)
- Genome annotation
- **Case study**

Case 1: genome assembly of a wheat blast fungal strain

- 10 SMRTCell PacBio data (P6-C4)



PacBio RS II



K-State BRI

Longest reads: 74,474 bp
Coverage*: 276x

- PCR-free Illumina TruSeq data



HiSeq 2500

>2x10⁷ pairs of 2x250 paired-end reads
Coverage*: 222x

* assuming the genome size is 45 Mb

Summary of the assembly

Statistics of the assembly*

Item	Statistics
total contig number	31
total contig length	44,522,920
longest contig	7,902,655
shortest contig	12,906
N50	5,402,116
overall GC	0.50
min contig GC	0.28

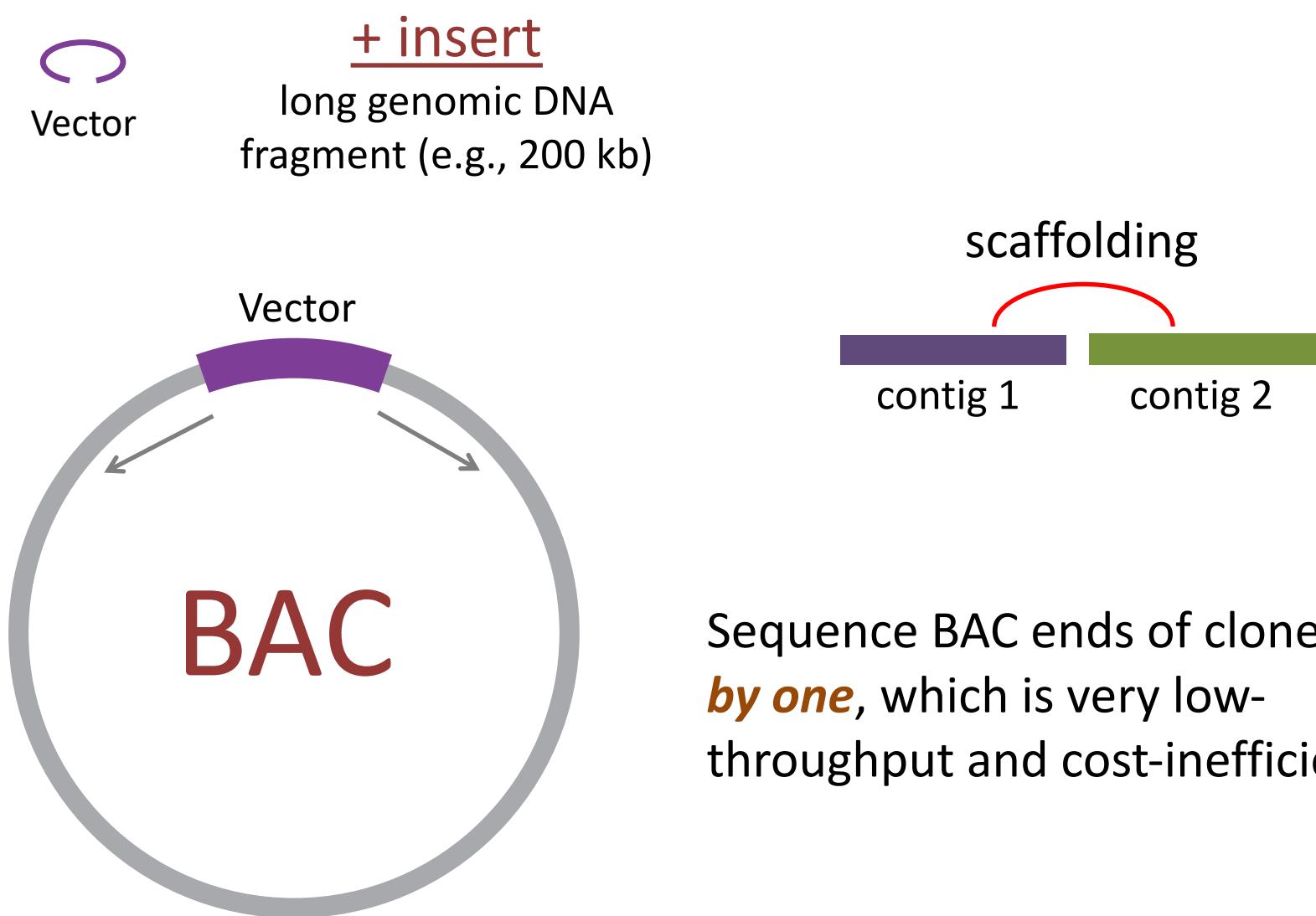
* *Canu* for PacBio assembly
Quiver polishing
Further error correction using Illumina data

mitochondrial sequence

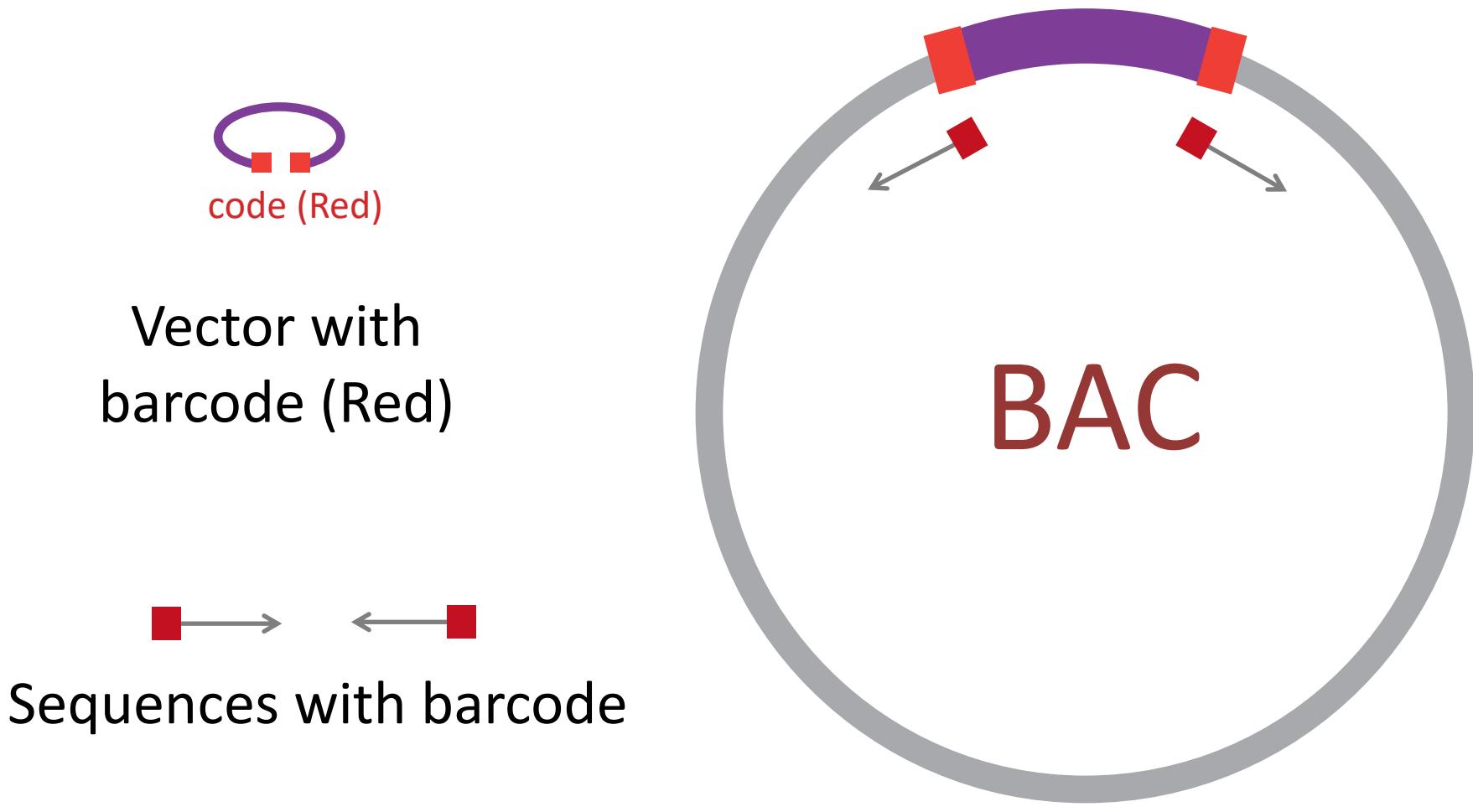
List of contigs (N=31)

Contig	Length (bp)	GC%
tig00000000	7,902,655	51.2
tig00000024	7,531,155	49.1
tig00000030	6,090,985	50.2
tig00000003	5,402,116	51.2
tig00000011	4,657,194	49.2
tig00000004	4,442,877	51.2
tig00000005	4,042,640	49.2
tig00000001	1,778,515	46.4
tig00000014	461,599	49.4
tig00000025	398,015	44.3
tig00000039	253,502	45.1
tig00000026	240,676	43.8
tig00000018	237,459	47.4
tig00000016	184,678	48
tig00000007	138,010	49.9
tig00000022	110,918	46
tig00000020	78,737	49.6
tig00000006	69,533	51.1
tig00000033	69,131	49.2
tig00000043	57,199	28.4
tig00000015	51,862	48.5
tig00000008	47,134	47.6
tig00000031	42,261	48.6
tig00000041	36,641	52.5
tig00000023	36,455	44.2
tig00000037	35,037	48.9
tig00000032	32,608	49.7
tig00000017	28,099	51.7
tig00000038	27,162	49.5
tig00000019	25,161	51.2
tig00000027	12,906	47.3

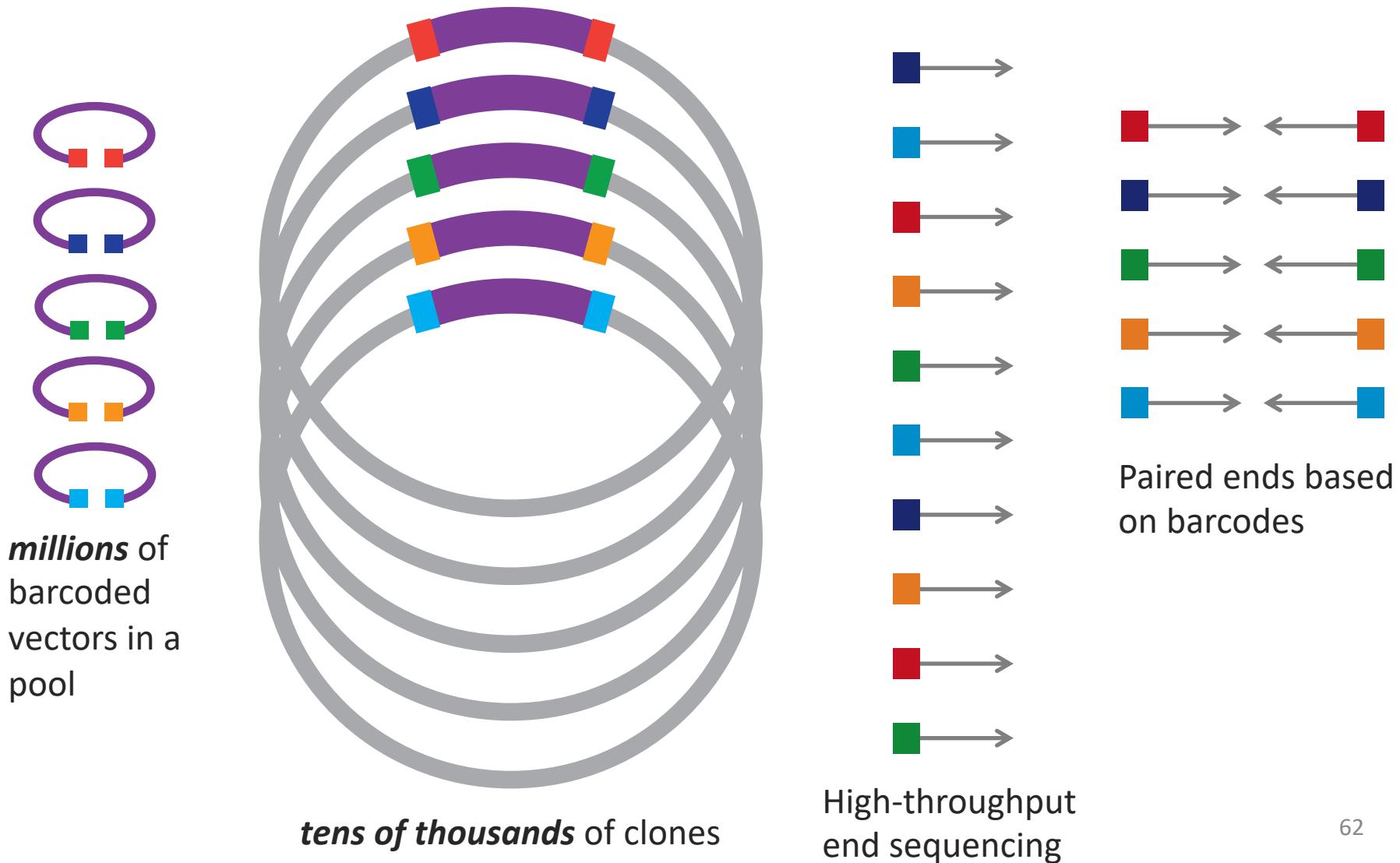
BAC-end sequencing for assembly improvement?



Idea: build random barcodes in vectors to enable a high throughput (I)

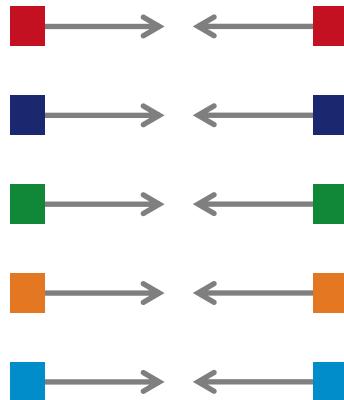


Idea: build random barcodes in vectors to enable a high throughput (II)

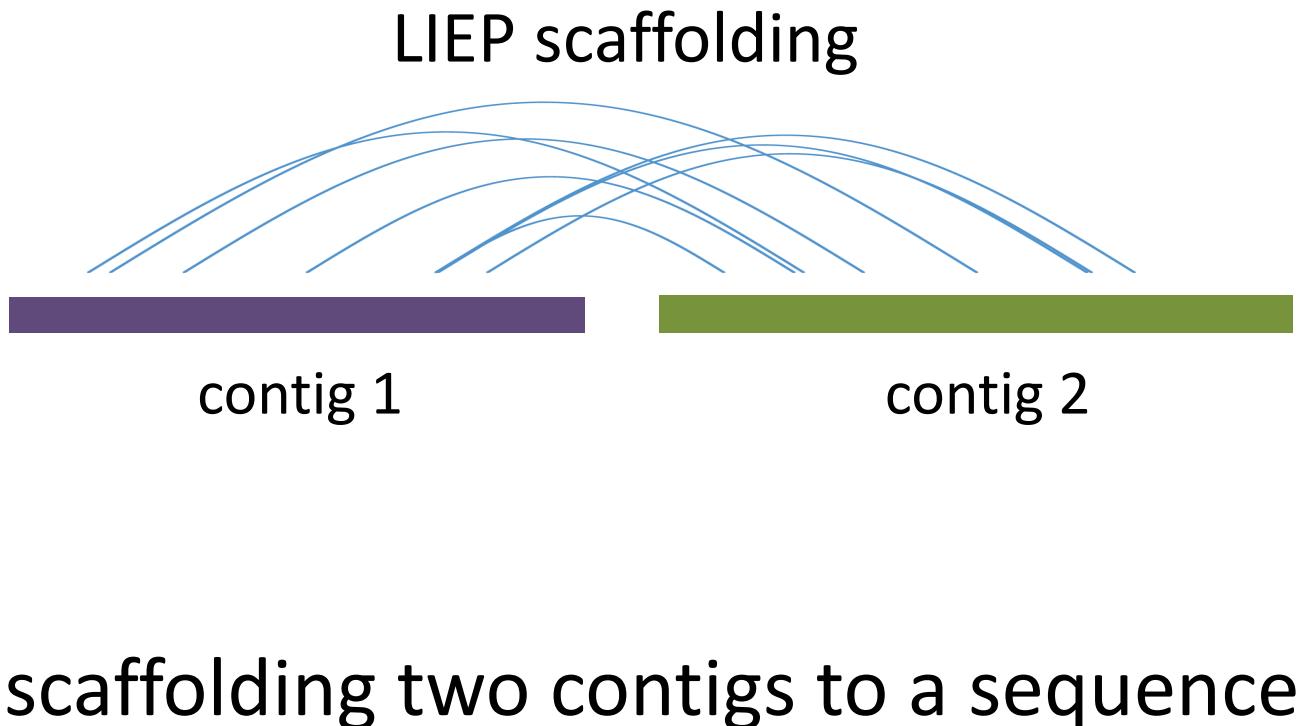


LIEP Scaffolding to improve the B71 assembly

Long Distance
End Pairs (LIEP)



~95% are long-
distance read pairs



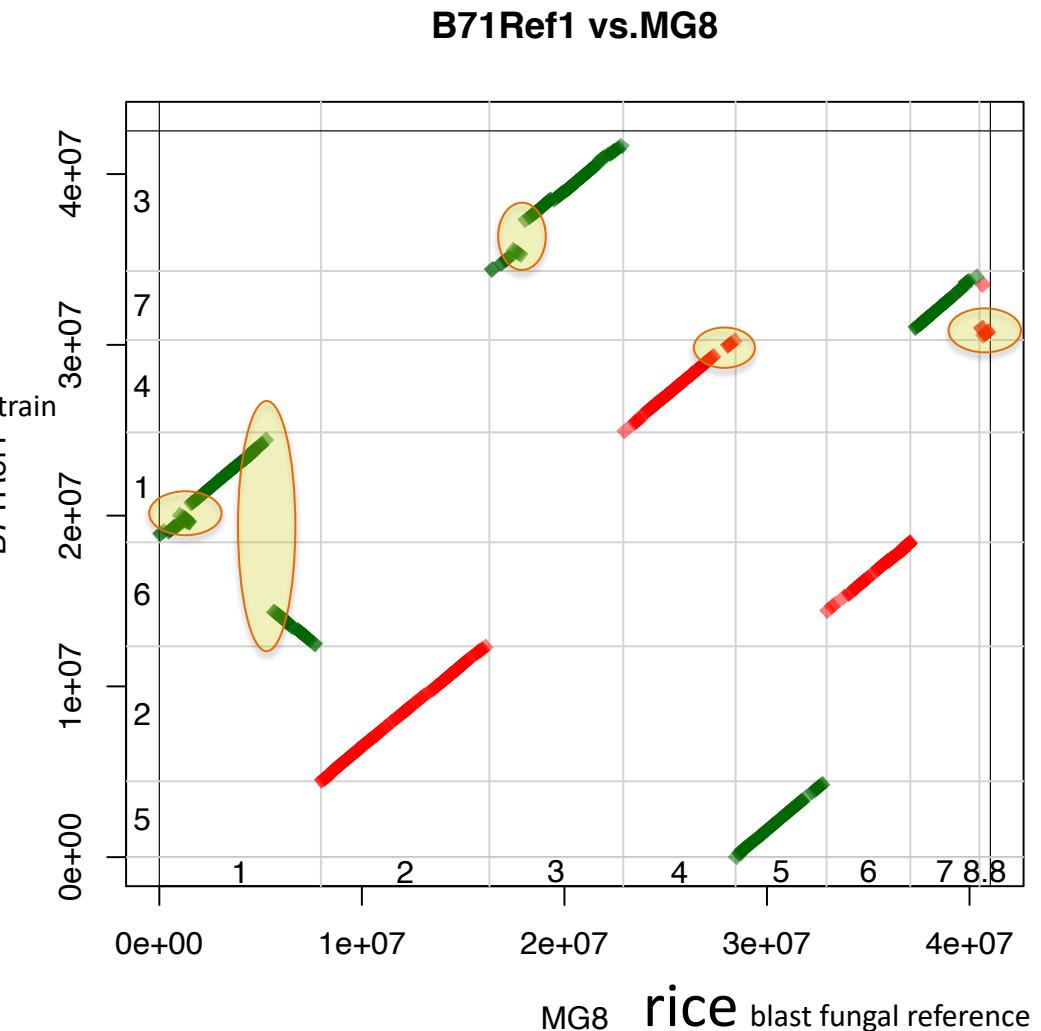
scaffolding two contigs to a sequence

MoT B71 final assembly vs. MG8 (70-15, rice strain)

Chr	Length (bp)
chr1	6,442,091
chr2	7,902,655
chr3	8,206,304
chr4	5,402,116
chr5	4,442,877
chr6	6,090,985
chr7	4,042,640
scaf1	941,816
scaf2	739,928
scaf3	104,670
scaf4	74,396
scaf5	69,131

Wheat

blast fungal strain
B71Ref1



>10 kb overlap and >95% identity

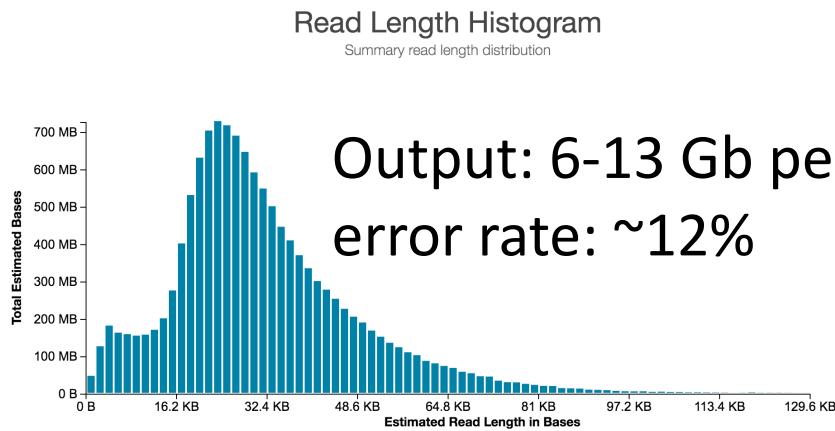
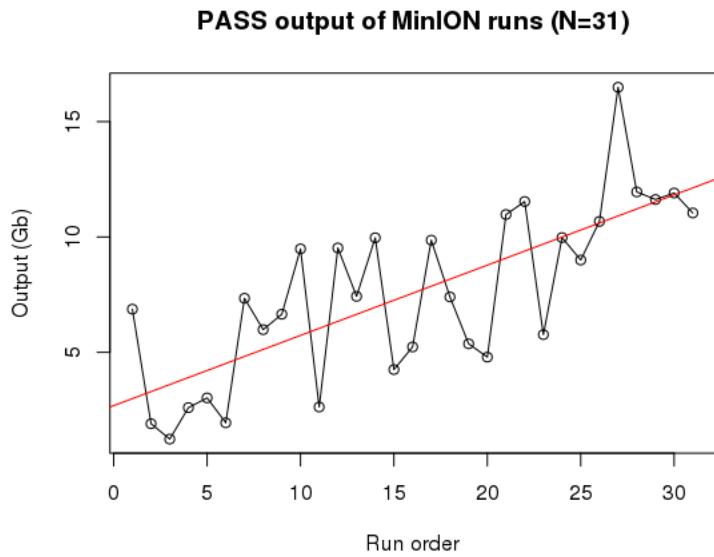
Case 2: Maize genome assembly with Nanopore MinION sequencing

Nanopore

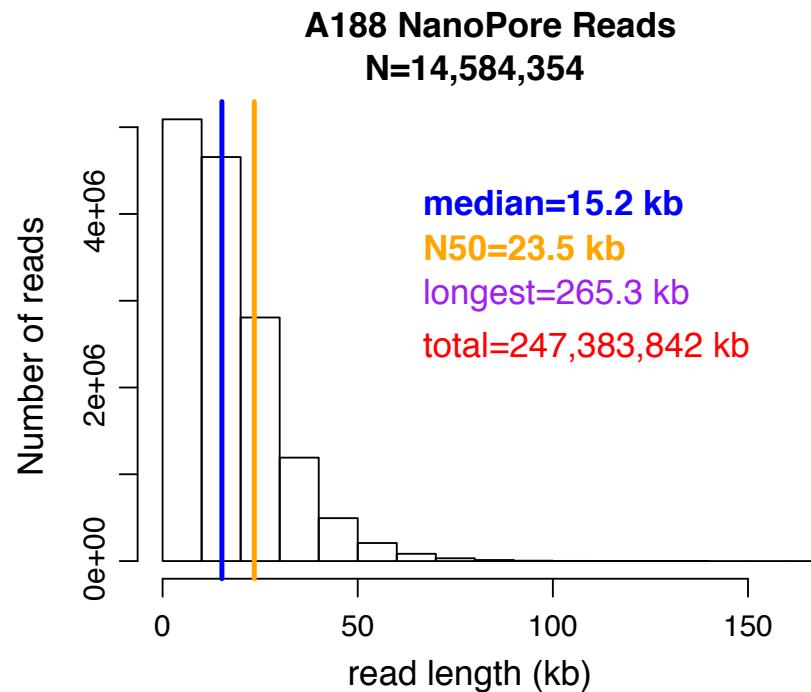
MinION (N=31)



FLO-MIN106
SQK-LSK109



Output: 6-13 Gb per flowcell
error rate: ~12%



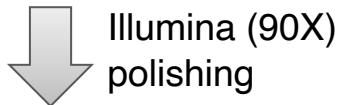
Phase - I

Nanopore long reads (90X)



canu

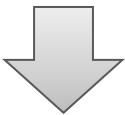
Canu asm, Nanopolish



Illumina (90X)
polishing

Polished contigs

Mt; Pt



BioNano map



hybrid
scaffold

scaffolds + contigs

tGBS of DH lines



mstmap

Genetic map



ALLMAPS

pseudochr

scaffolds +
contigs

B73 genome



Ragoo

Locate
Unanchored contigs

Assembly strategy

Citations

- Berger, B., J. Peng, M. Singh, Computational solutions for omics data. *Nature Reviews Genetics* 14: 333-346 (2013).
- Compeau, P. E. C., P. A. Pevzner, G. Tesler, How to apply de Bruijn graphs to genome assembly. *Nature Biotechnology* 29: 987-991 (2011).
- Nagarajan, N., M. Pop, Sequence assembly demystified. *Nature Reviews Genetics* 14: 157-167 (2013).
- Miller, J. R., S. Koren, G. Sutton, Assembly algorithms for next-generation sequencing data. *Genomics* 95: 315-327 (2010).
- Brent, M. R. How does eukaryotic gene prediction work? *Nat. Biotechnol.* 25, 883–885 (2007).
- Stein, L. Genome annotation: from sequence to biology. *Nat. Rev. Genet.* 2, 493–503 (2001).
- Yandell, M., Ence, D. A beginner’s guide to eukaryotic genome annotation. *Nat. Rev. Genet.* 13, 329–342 (2012).
- Simão, F. A., et al. BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics* 31, 3210–3212 (2015).