

Genome Assembly Overview 3

BCB 5250 Introduction to Bioinformatics II

Spring 2020

Tae-Hyuk (Ted) Ahn

Department of Computer Science
Program of Bioinformatics and Computational Biology
Saint Louis University



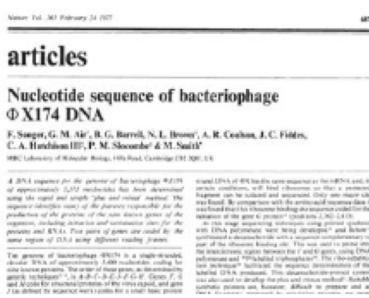
SAINT LOUIS
UNIVERSITY™

— EST. 1818 —

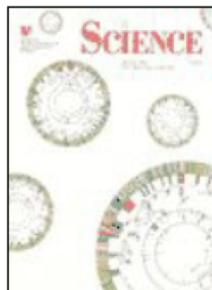
Learning Outcome:

- Learn general knowledge about *de novo* genome assembly.

Milestones in Genome Assembly



1977. Sanger et al.
1st Complete Organism
5375 bp



1995. Fleischmann et al.
1st Free Living Organism
TIGR Assembler. 1.8Mbp



1998. C.elegans SC
1st Multicellular Organism
BAC-by-BAC Phrap. 97Mbp



2000. Myers et al.
1st Large WGS Assembly.
Celera Assembler. 116 Mbp



2001. Venter et al., IHGSC
Human Genome
Celera Assembler/GigaAssembler. 2.9 Gbp



2010. Li et al.
1st Large SGS Assembly.
SOAPdenovo 2.2 Gbp

Like Dickens, we must computationally reconstruct a genome from short fragments

* Slide from Mike Schatz

Assembly Applications

- Novel genomes
- Metagenomes
- Sequencing assays
 - Structural variations
 - Transcript assembly
 -

Assembly Applications

MENU ▾

nature
genetics

Article | Open Access | Published: 31 May 2019

Genome assembly of a tropical maize inbred line provides insights into structural variation and crop improvement

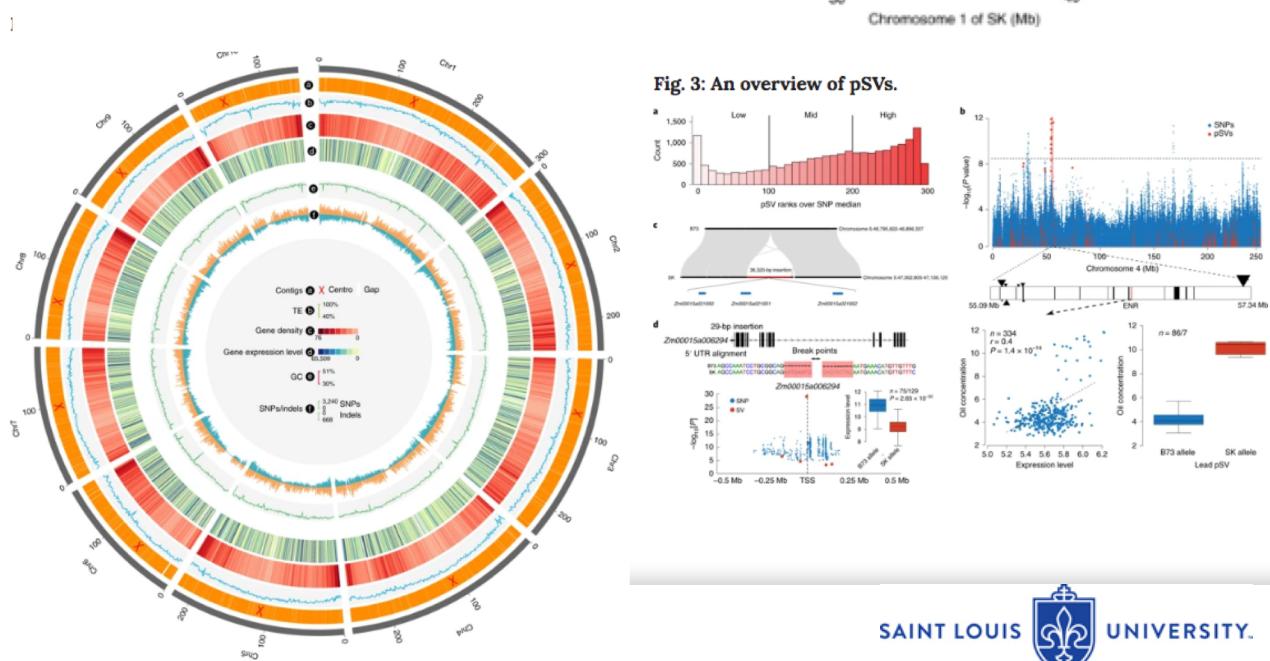
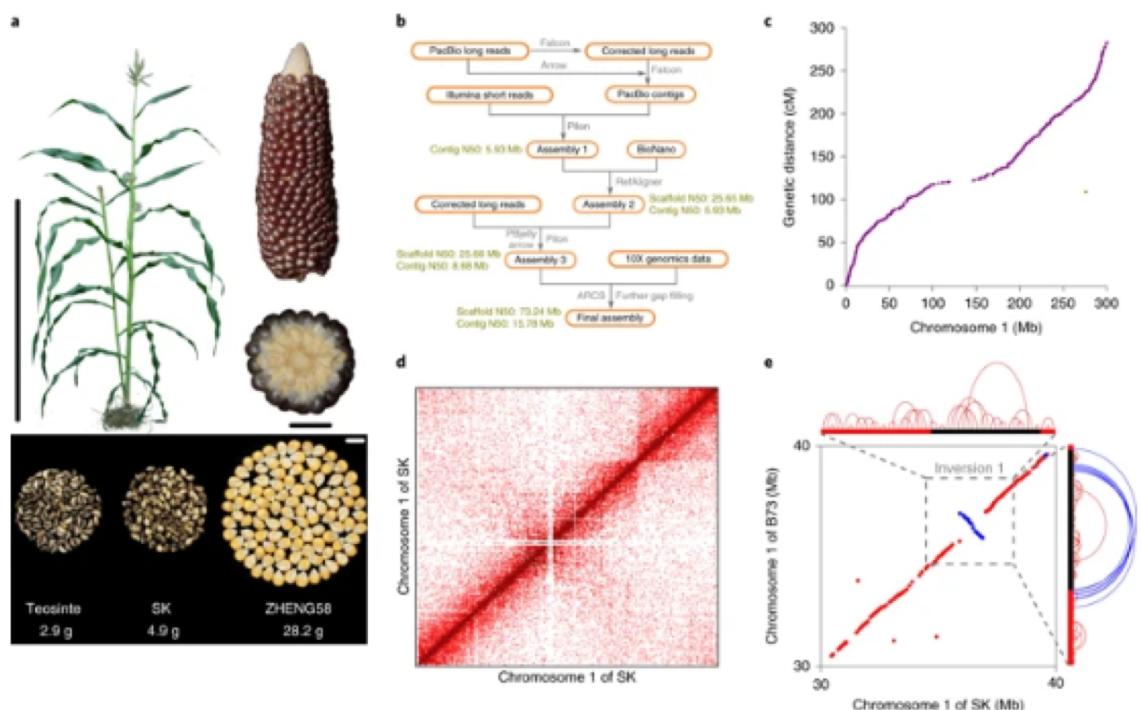
Ning Yang, Jie Liu, Qiang Gao, Songtao Gui, Lu Chen, Linfeng Yang, Juan Huang, Tianquan Deng, Jingyun Luo, Lijuan He, Yuebin Wang, Pengwei Xu, Yong Peng, Zhuoxing Shi, Liu Lan, Zhiyun Ma, Xin Yang, Qianqian Zhang, Mingzhou Bai, San Li, Wenqiang Li, Lei Liu, David Jackson & Jianbing Yan 

Nature Genetics 51, 1052–1059 (2019) | Download Citation 

11k Accesses | 1 Citations | 37 Altmetric | Metrics 

Genomic feature	SK
Estimated genome size (Mb)	2,320.3
Length of SK assembly (bp)	2,161,392,594
Largest scaffold (bp)	301,303,159
Scaffold N50 (bp)	73,237,962
Largest contig (bp)	56,183,869
Contig N50 (bp)	15,776,512
Number of gaps/medium gap size (bp)	238/23,266
Sequences anchored to chromosomes (%)	98.1
Numbers of gene models/transcripts	43,271/95,938
Mean transcript length (bp)	6,172.9
Mean coding sequence length (bp)	1,389.3
Total size of transposable elements (bp)	1,944,366,000

Fig. 1: Overview of the SK genome assembly.



Illumina NGS

- <https://www.illumina.com/science/technology/next-generation-sequencing.html>

Illumina vs PacBio?

- <https://www.labpulse.com/index.aspx?sec=sup&sub=ivd&pag=dis&ItemID=800673>

Illumina drops deal to acquire Pacific Biosciences

by Michael McCoy

JANUARY 7, 2020 | APPEARED IN VOLUME 98, ISSUE 2



Illumina's offices in San Diego

Credit: Illumina

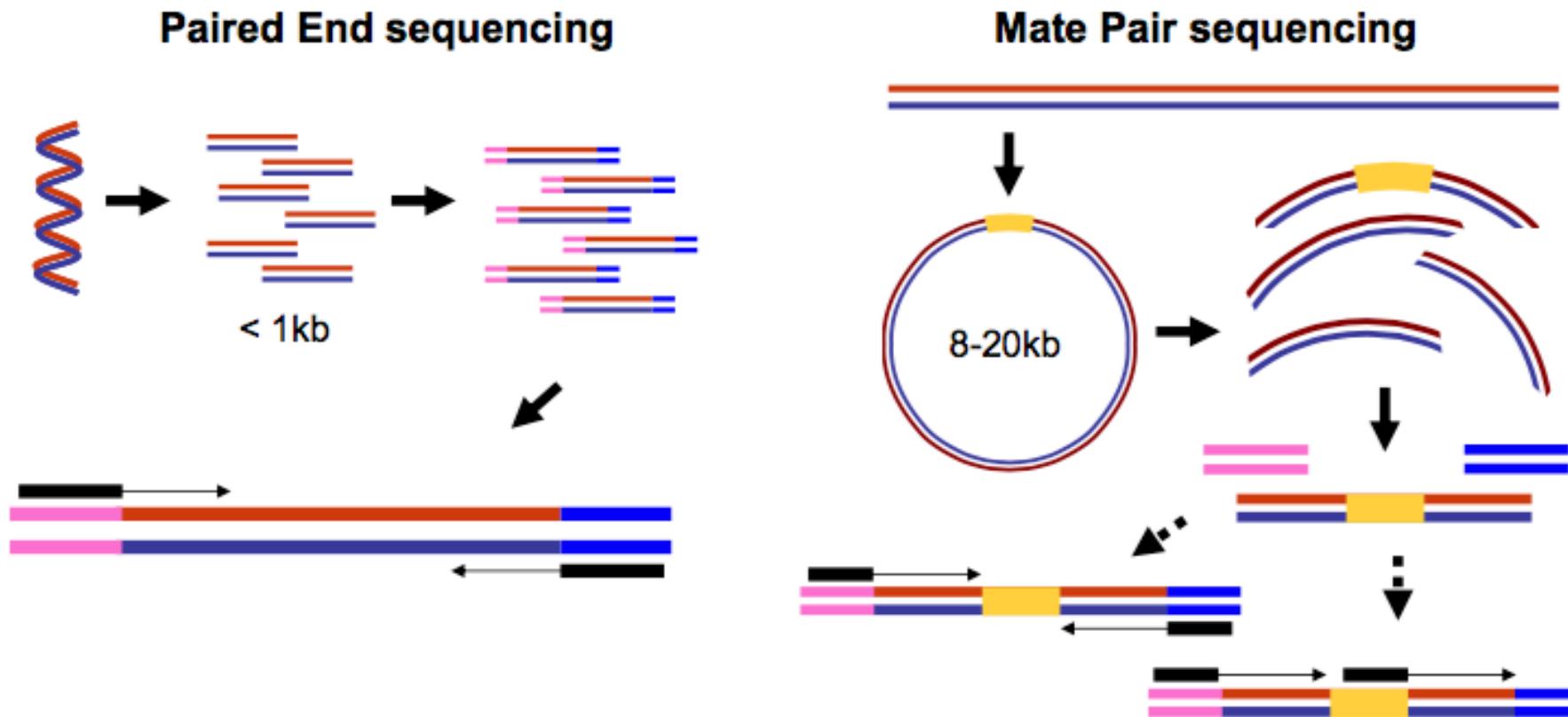
ADVERTISEMENT

MOST POPULAR IN BUSINESS

[C&EN's World Chemical Outlook 2020](#)

[C&EN's Global Top 50 chemical companies of 2017](#)

Pair-end and mate-pair sequencing libraries



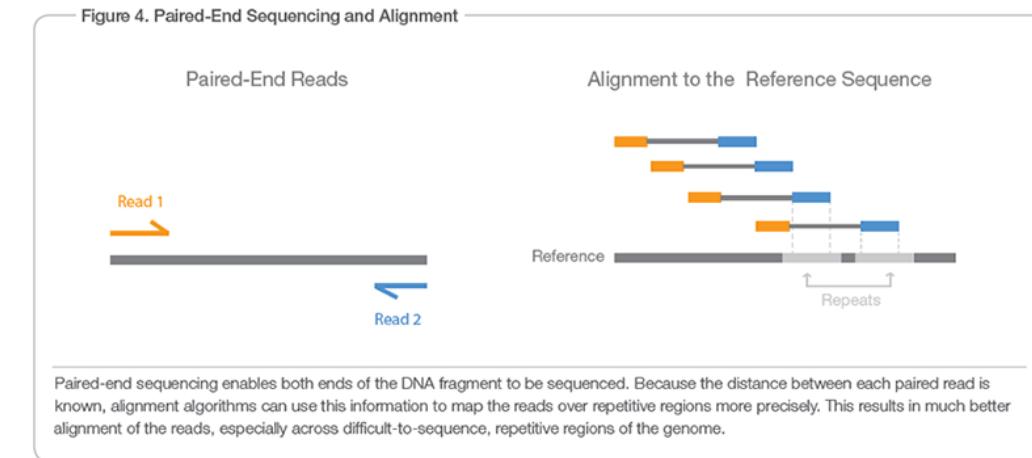
Paired-end and mate-pair reads. In paired end sequencing (left) the actual ends of rather short DNA molecules (less than 1kb) are determined, while for mate pair sequencing (right) the ends of long molecules are joined and prepared in special sequencing libraries. In these mate pair protocols, the ends of long, size-selected molecules are connected with an internal adapter sequence (i.e. linker, yellow) in a circularization reaction. The circular molecule is then processed using restriction enzymes or fragmentation. Fragments are enriched for the linker and outer library adapters are added around the two combined molecule ends. The internal adapter can then be used as a second priming site for an additional sequencing reaction in the same orientation or sequencing can be performed from the second adapter, from the reverse strand. (From Ph.D. dissertation by [Martin Kircher](#))

Paired-End Sequencing

What is Paired-End Sequencing?

Paired-end sequencing allows users to sequence both ends of a fragment and generate high-quality, alignable sequence data. Paired-end sequencing facilitates detection of genomic rearrangements and repetitive sequence elements, as well as gene fusions and novel transcripts.

In addition to producing twice the number of reads for the same time and effort in library preparation, sequences aligned as read pairs enable more accurate read alignment and the ability to detect insertion-deletion (indel) variants, which is not possible with single-read data.¹ All Illumina next-generation sequencing (NGS) systems are capable of paired-end sequencing.



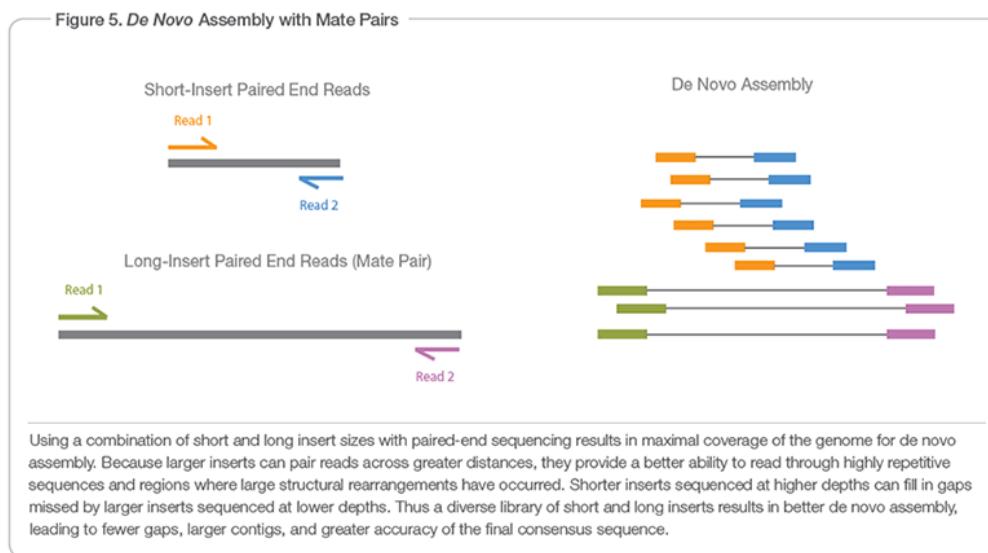
Single-Read Sequencing

Single-read sequencing involves sequencing DNA from only one end, and is the simplest way to utilize Illumina sequencing. This solution delivers large volumes of high-quality data, rapidly and economically. Single-read sequencing can be a good choice for certain methods such as small RNA-Seq or chromatin immunoprecipitation sequencing (ChIP-Seq).

Mate-Pair Sequencing

Mate pair sequencing involves generating long-insert paired-end DNA libraries useful for a number of sequencing applications, including:

- *De novo* sequencing
- Genome finishing
- Structural variant detection
- Identification of complex genomic rearrangements

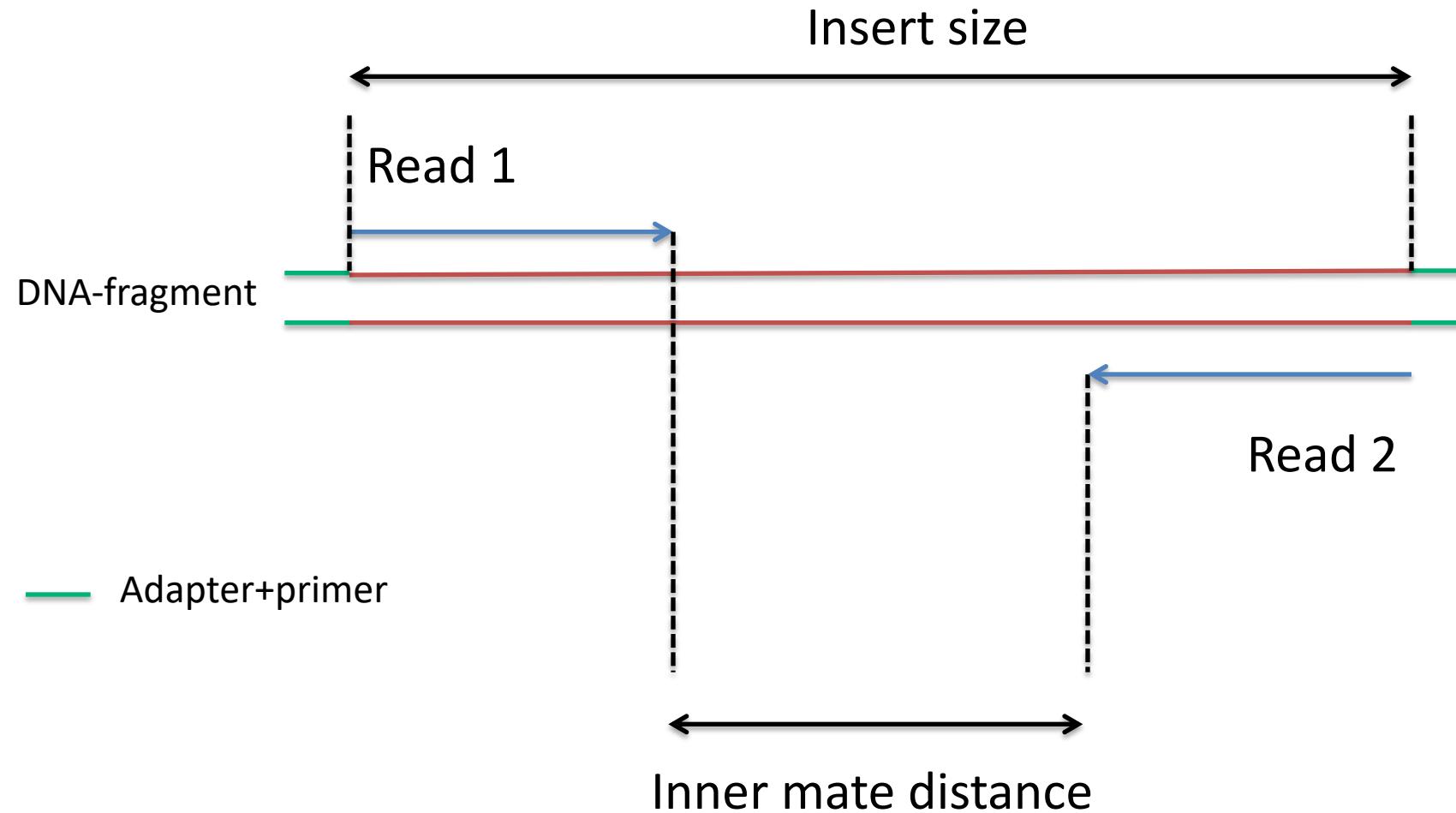


Combining data generated from mate pair library sequencing with that from short-insert paired-end reads provides a powerful combination of read lengths for maximal sequencing coverage across the genome

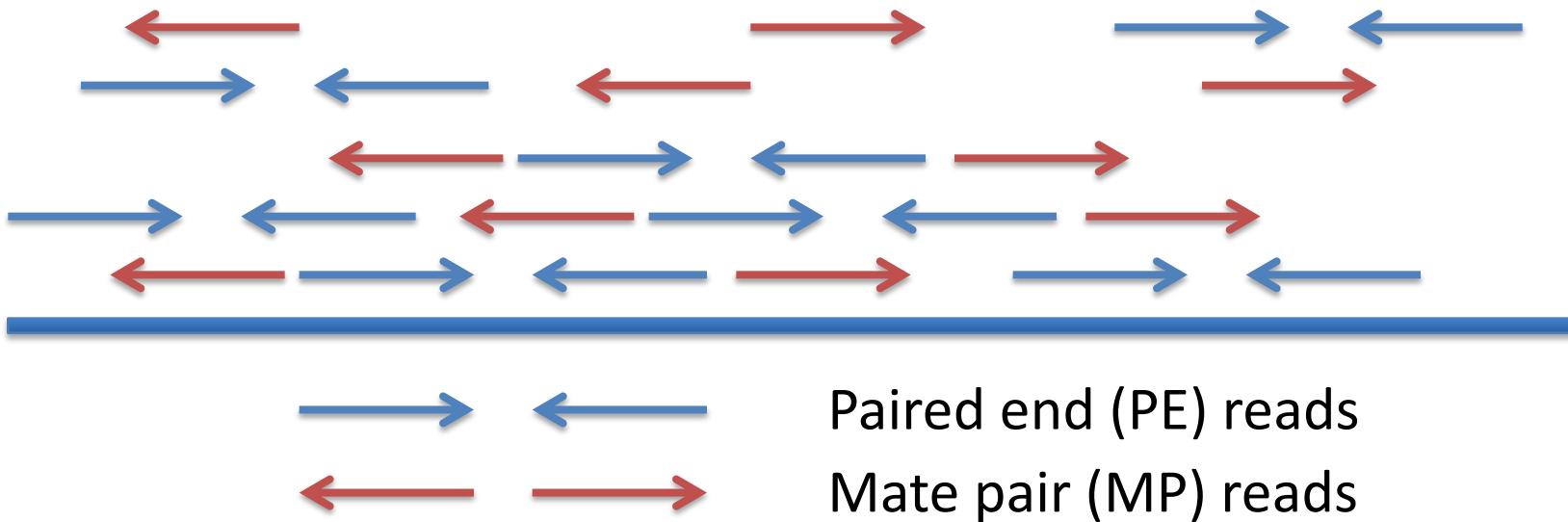
Mate Pair Library Preparation Process

https://www.illumina.com/technology/next-generation-sequencing/mate-pair-sequencing_assay.html

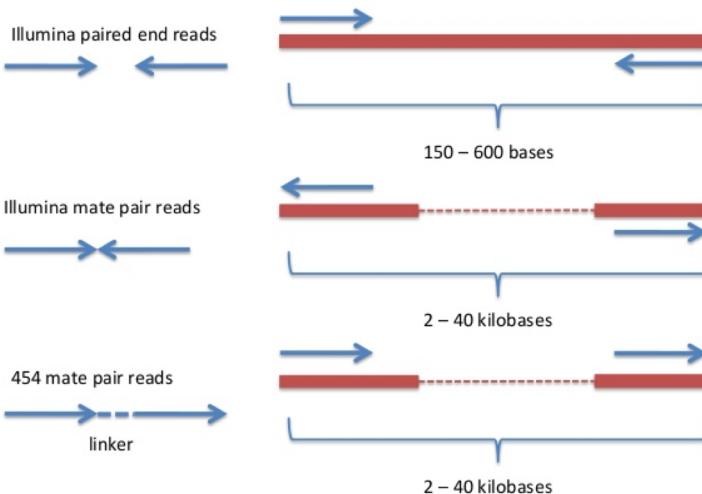
Insert Size



Orientation of paired reads



Mate pair splitting and orientation



Reads (FASTQ)

- Thus in both cases (paired-end and mate-pair) a single physical piece of DNA (or RNA in the case of RNA-seq) is sequenced from two ends and so generates two reads. These can be represented as separate files (two fastq files with first and second reads) or a single file where reads for each end are interleaved. Here are examples:

Two single files

File 1

```
@M02286:19:00000000-AA549:1:1101:12677:1273 1:N:0:23
CCTACGGGTGGCAGCAGTGAGGAATATTGGTCAATGGACGGAAGTCT
+
ABC8C,:@F:CE8,B-,C,-6-9-C,CE9-CC--C-<-C++,,+;CE
@M02286:19:00000000-AA549:1:1101:15048:1299 1:N:0:23
CCTACGGGTGGCTGCAGTGAGGAATATTGGACAATGGTCGGAAGACT
+
ABC@CC77CFCEG;F9<F89<9--C,CE,--C-6C-,CE:++7:,CF
```

File 2

```
@M02286:19:00000000-AA549:1:1101:12677:1273 2:N:0:23
CACTACCCGTGTATCTAATCCTGTTGATACCCGACCTTCGAGCTTA
+
--8A,CCE+,,,,<CC,,<CE@,CFD,,C,CFF+@+@CCEF,,,B+C,
@M02286:19:00000000-AA549:1:1101:15048:1299 2:N:0:23
CACTACCGGGGTATCTAATCCTGTTGCTCCCACGCTTCGTCCATC
+
-6AC,EE@::CF7CFF<<FFGGDFFF,@FGGGG?F7FEGGGDEFF>FF
```

Note that read IDs are **identical** in two files and they are listed in **the same** order. In some cases read IDs in the first and second file may be appended with /1 and /2 tags, respectively.

Reads (FASTQ)

- Thus in both cases (paired-end and mate-pair) a single physical piece of DNA (or RNA in the case of RNA-seq) is sequenced from two ends and so generates two reads. These can be represented as separate files (two fastq files with first and second reads) or a single file where reads for each end are interleaved. Here are examples:

Interleaved file

```
@1/1
AGGGATGTGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTA
+
EGGEGGGDFGEEEAEECGDEGGFEEGEFGBEEDDECfefDD@CDD<ED
@1/2
CTAACCCCTAACCCCTAACCCCTAACCCCTAACCCCTAACCCCTAAC
+
GHHHDFDFGEGFBGEGGEGEGGGHGFHFHFFFFHHHEF?EFEFF
@2/1
AGGGATGTGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTA
+
HHHHHHHEGFHEEFEEHEEHGGEGGGGFEGFGGGGHHHHFBEEEEEEFG
@2/2
CTAACCCCTAACCCCTAACCCCTAACCCCTAACCCCTAACCCCTAAC
+
HHHHHHHHHHHHHHGHHHHHHGHHHHHHHHHHFHHHFHHHHHHHHHHHH
```

Planning a genome sequencing project

- How large is my genome?
- How much of it is repetitive, and what is the repeat size distribution?
- Is a good quality genome of a related species available?
- What will be my strategy for performing the assembly?

How large is my genome?

- The size of the genome can be estimated from the ploidy of the organism and the DNA content per cell
 - This will affect:
 - How many reads will be required to attain sufficient coverage (typically 10x to 100x)
 - What sequencing technology to use
 - What computational resources will be needed

[Methods Mol Biol.](#) 2011;772:3-12. doi: 10.1007/978-1-61779-228-1_1.

Genome size determination using flow cytometry of propidium iodide-stained nuclei.

Hare EE¹, Johnston JS.

 Author information

Abstract

With the rapid expansion of whole-genome sequencing and other genomic studies in nonmodel organisms, there is a growing demand for robust and user-friendly methods for estimating eukaryotic genome sizes across a broad range of taxa. Propidium iodide (PI) staining with flow cytometry is a powerful method for genome sizing because it is relatively fast, works with a wide variety of materials, and provides information on a very large number of nuclei. In this method, nuclei are stained with PI, which intercalates into the major groove of DNA. Unknown samples are typically costained with standard nuclei of a known genome size, and the relative fluorescence is used to calculate the genome size of the unknown.

PMID: 22065429 DOI: [10.1007/978-1-61779-228-1_1](https://doi.org/10.1007/978-1-61779-228-1_1)

[Indexed for MEDLINE]



Repetitive Sequences

- Most common source of assembly errors
- If sequencing technology produces reads > repeat size, impact is much smaller
- Most common solution: generate mate pairs with spacing > largest known repeat
- New solution: use new sequencing (3rd generation) long reads

Repetitive Sequences

- Assemblies can collapse around repetitive sequences

True structure of genomic region

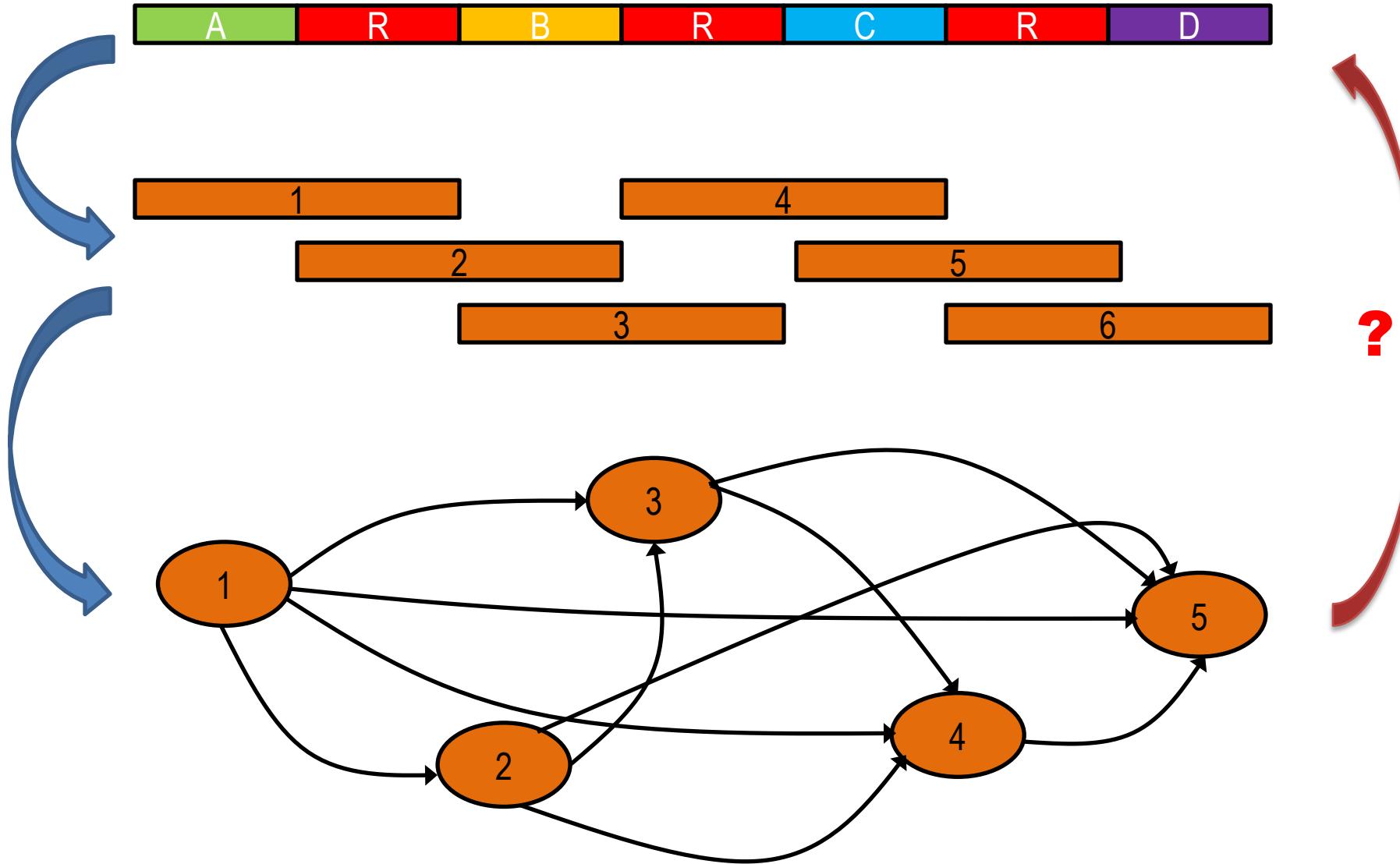


Incorrect assembly with “orphan” contig (red)



Salzberg S L , and Yorke J A Bioinformatics 2005;21:4320-4321

Assembly Complexity



Genome(s) from related species

- Preferably of good quality, with large reliable scaffolds
- Help guiding the assembly of the target species
- Help verifying the completeness of the assembly
- Can themselves be improved in some cases
- **But** to be used with caution – can cause errors when architectures are different!

New Sequencing Technology Help Assembly

- Illumina Sequencing Platform
 - <https://www.illumina.com/systems/sequencing-platforms.html>
 - Huge yield, cheap, reliable, read length “long enough” (100-300 bp), industry standard=huge amount of available software
- Moleculo libraries
 - Acquired by Illumina in 2013, allows generation of fully assembled 10 kb sequences, but...



The TruSeq Synthetic Long-Read DNA Library Prep Kit has been discontinued. Illumina does not have an alternative solution. We remain committed to providing you with high-quality support and service.

New Sequencing Technology Help Assembly

● Pacific Biosciences

- Long reads, single molecules
- High error rate on longer fragments (15%), expensive
- Circular consensus sequencing (CCS) drops the error rate to 0.2%.

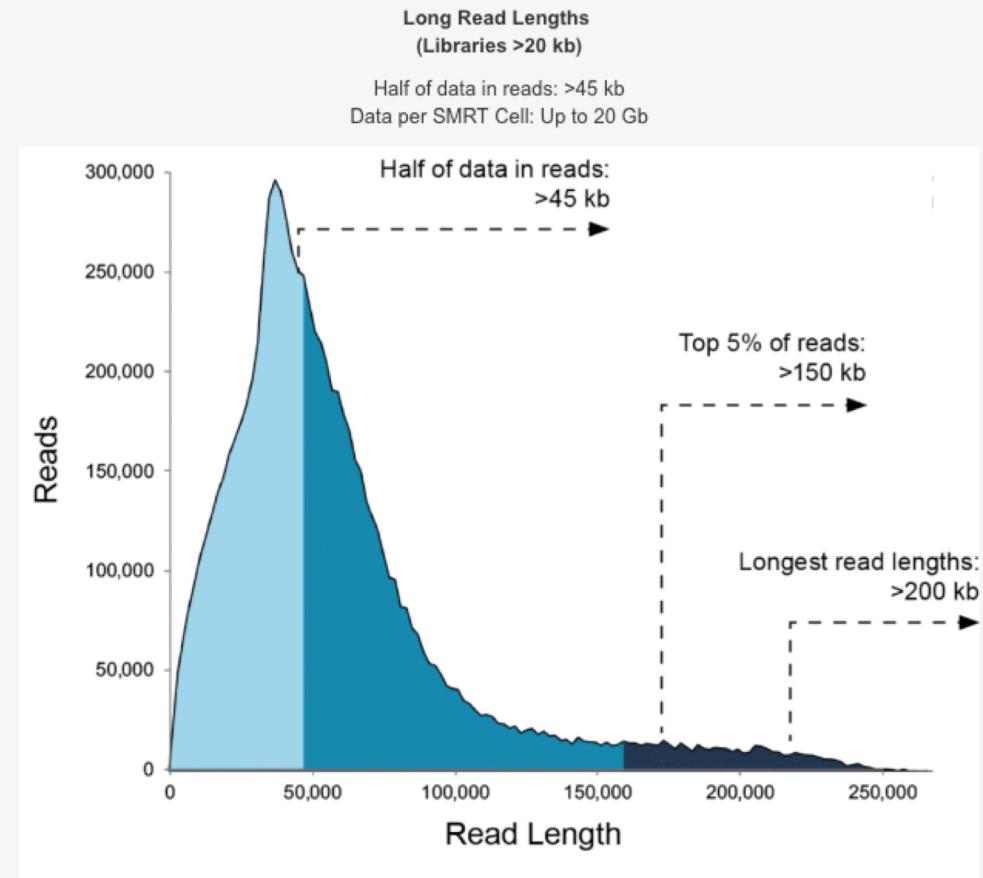
<https://doi.org/10.1038/s41587-019-0217-9>

● Nanopore

- Extremely long sequences, single molecule, portable (MinION)
- Very high error rates (up to 38% reported earlier version, but now dropped a lot)

<https://www.biostars.org/p/380759/>

PacBio long-read sequencing provides exceptional read lengths without compromising throughput or accuracy



Read length data shown above from a 35 kb size-selected human library using the SMRTbell Express Template Prep Kit on a Sequel System* (3.0 Chemistry, Sequel System Software v6.0, 20-hour movie).

Sequencing Technology & Assembly Algorithms

- Sanger sequencing: ~ 1000 bp
 - ***Overlap-layout-consensus*** algorithm dominated the assembly field
 - E.g., Celera assembler
- Illumina sequencing: <100 bp → 100 bp → 250~300 bp
 - OLC algorithm broke down completely for those short reads, because finding overlaps between all read pairs scaled quadratically with the number of reads
 - ***de Bruijn*** method works well with the short sequences
 - E.g., ABySS, IDBA, ALLPATHS-LG, Velvet, SOAPdenovo
 - Best k-mer: 35~75
- PacBio or Oxford Nanopore (~ 250,000 bp)
 - OLC algorithm (Canu) is tuned for these long sequences.
 - <https://canu.readthedocs.io/en/latest/index.html>

Assembly Strategies and Algorithms

- In all cases, start with cleanup and error correction of raw reads
- For long reads (>500 nt), Overlap/Layout/Consensus (OLC) algorithms work best
- For short reads, De Bruijn graph-based assemblers are most widely used
- <https://doi.org/10.1093/bib/bbx147>

Comprehensive evaluation of non-hybrid genome assembly tools for third-generation PacBio long-read sequence data 

Vasanthan Jayakumar, Yasubumi Sakakibara 

Briefings in Bioinformatics, Volume 20, Issue 3, May 2019, Pages 866–876,

<https://doi.org/10.1093/bib/bbx147>

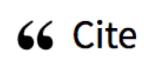
Published: 03 November 2017 **Article history** ▾



PDF



Split View



Cite



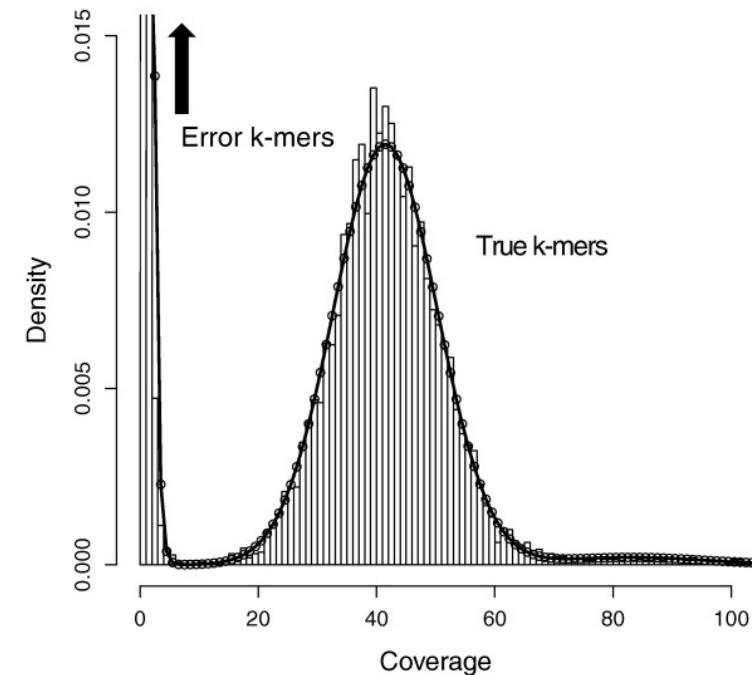
Permissions



Share ▾

Cleaning up the data

- Trim reads with low quality calls
- Remove short reads
- Correct errors:
 - Find all distinct k-mers (typically k=15) in input data
 - Plot coverage distribution
 - Correct low-coverage k-mers to match high-coverage

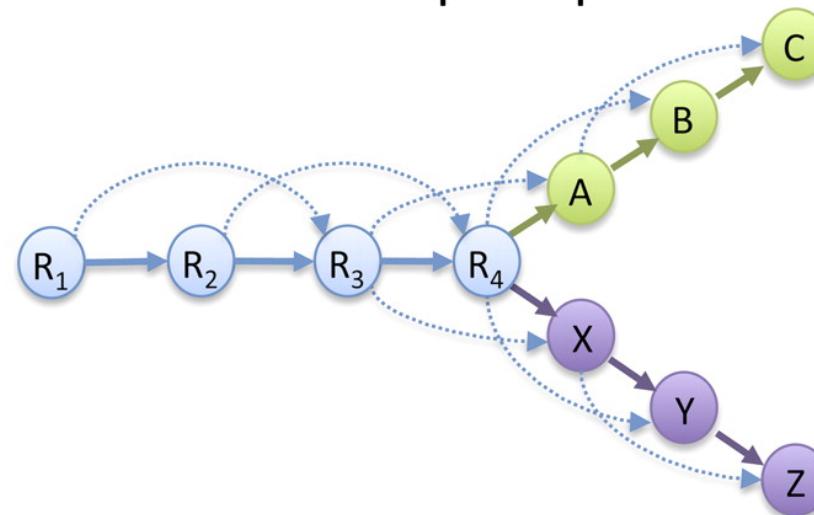


OLC vs De Bruijn

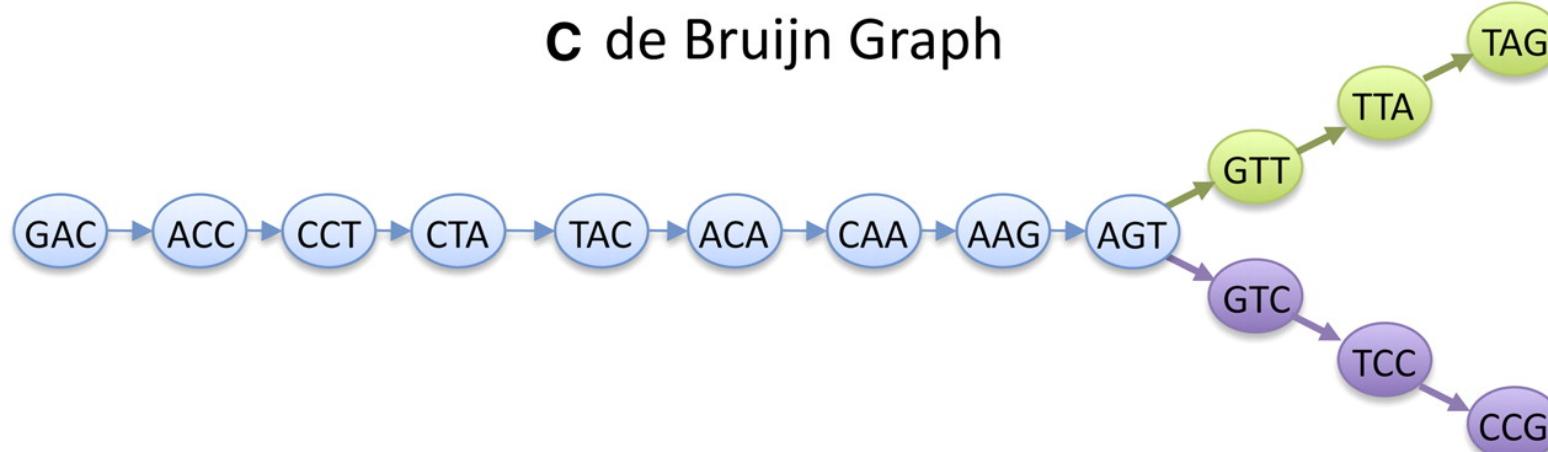
A Read Layout

R_1 :	GACCTACA
R_2 :	ACCTACAA
R_3 :	CCTACAAG
R_4 :	CTACAAGT
A:	TACAAGTT
B:	ACAAGTTA
C:	CAAGTTAG
X:	TACAAGTC
Y:	ACAAGTCC
Z:	CAAGTCCG

B Overlap Graph



C de Bruijn Graph



Overlap-Layout-Consensus

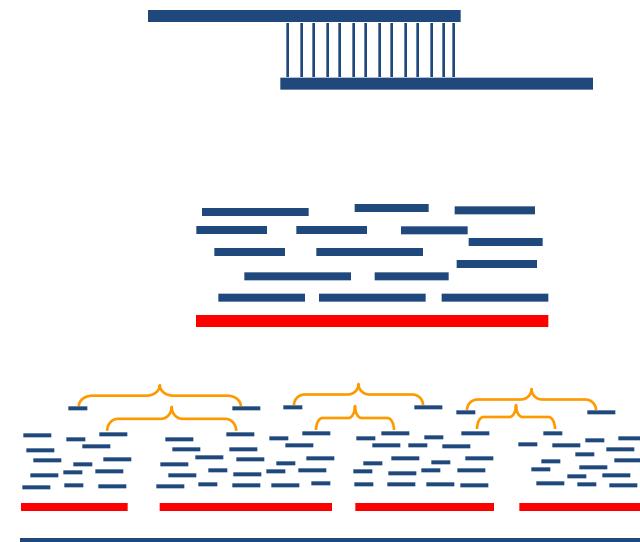
Assemblers (Old): ARACHNE, PHRAP, CAP, TIGR, CELERA

Assemblers (New): PBcR, Canu, Falcon

Overlap: find potentially overlapping reads

Layout: merge reads into contigs and
contigs into supercontigs

Consensus: derive the DNA sequence and
correct read errors



..ACGATTACAATAGGTT..

Finding Overlaps

Can we be less naive than this?

Say $l = 3$

Look for this in Y ,
going right-to-left

X: CTCTAG**GCC**
Y: TAGGCCCTC



X: CTCTAG**GCC**
Y: TAG**GCC**CTC

Found it

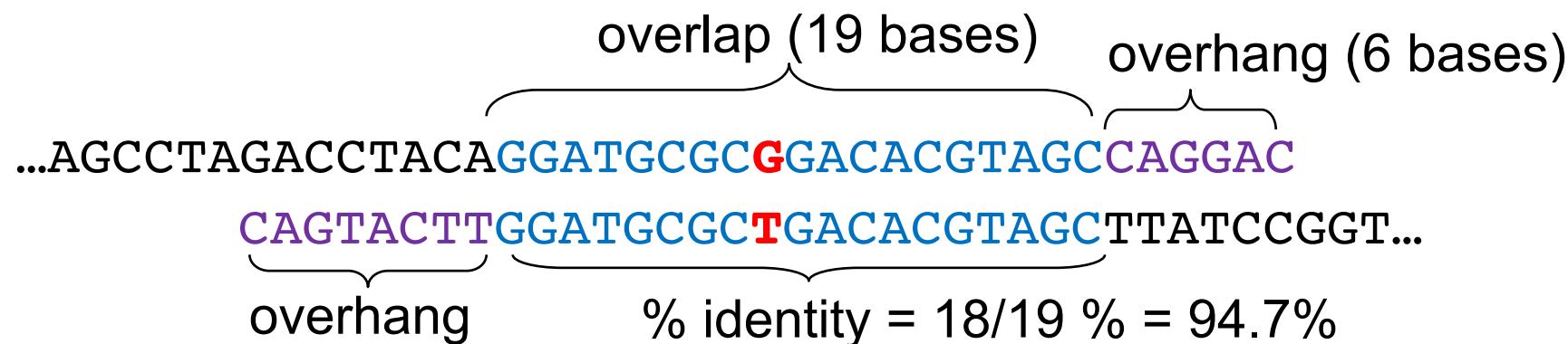
Extend to left; in this case, we
confirm that a length-6 prefix
of Y matches a suffix of X

X: CT**C**TAGGCC
Y: TAG**GCC**CTC



We're doing this for every pair of input strings

Overlap between two sequences



overlap - region of similarity between regions

overhang - un-aligned ends of the sequences

The assembler screens merges based on:

- length of overlap
- % identity in overlap region
- maximum overhang size.

Finding Overlaps

- Overlap Graph
 - Overlap-Layout-Consensus (mostly by min overlap length)
 - de Bruijn Graph (by k -mers)
- Allow mismatches / indels

Overlap Graph

Bi-directed Graph Format

Edge Types:

Regular Dovetail



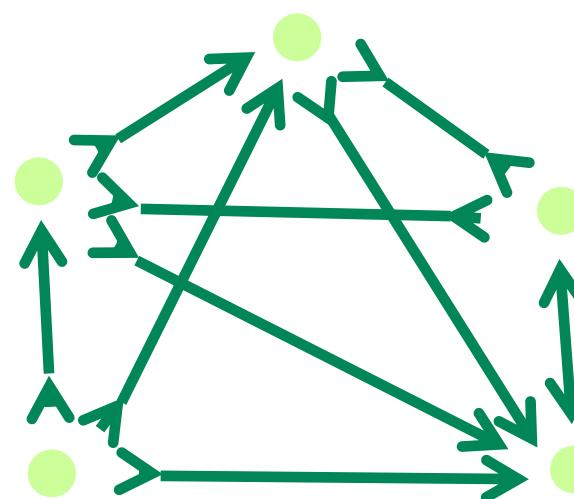
Prefix Dovetail



Suffix Dovetail



E.G.:



Edges are annotated
with deltas of overlaps

Layout

- Overlap graph is big and messy. Contigs don't "pop out" at us.
- Anything redundant about this part of the overlap graph?
- Bundle stretches of the overlap graph into unitigs

Layout

Read r_1 : AGCTAACGATTTACGATAGCCGATAGCTAAATTAC
Read r_2 : CGTAATTAGCTATCGGCTATCGTAAATGCTTAGC
Read r_3 : AACGTAATTAGCTATCGGCTATCGTAAATGCTTA
Read r_4 : GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read r_5 : GTATAACGTAATTAGCTATCGGCTATCGTAAATG
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read r_8 : ATATAACGTAATTAGCTATCGGCTATCGTAAATG
Read r_9 : ATTTACGATAGCCGATAGCTAAATTACGTTATA
Read r_{10} : CTATATAACGTAATTAGCTATCGGCTATCGTAAA

Set of given reads

Read r_1 : AGCTAACGATTTACGATAGCCGATAGCTAAATTAC
Read \bar{r}_2 : GCTAACGATTTACGATAGCCGATAGCTAAATTACG
Read \bar{r}_3 : TAAGCATTACGATAGCCGATAGCTAAATTACGTT
Read r_4 : GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read \bar{r}_5 : CATTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read \bar{r}_8 : CATTACGATAGCCGATAGCTAAATTACGTTATAT
Read r_9 : ATTTACGATAGCCGATAGCTAAATTACGTTATA
Read \bar{r}_{10} : TTTACGATAGCCGATAGCTAAATTACGTTATA

Overlapping reads

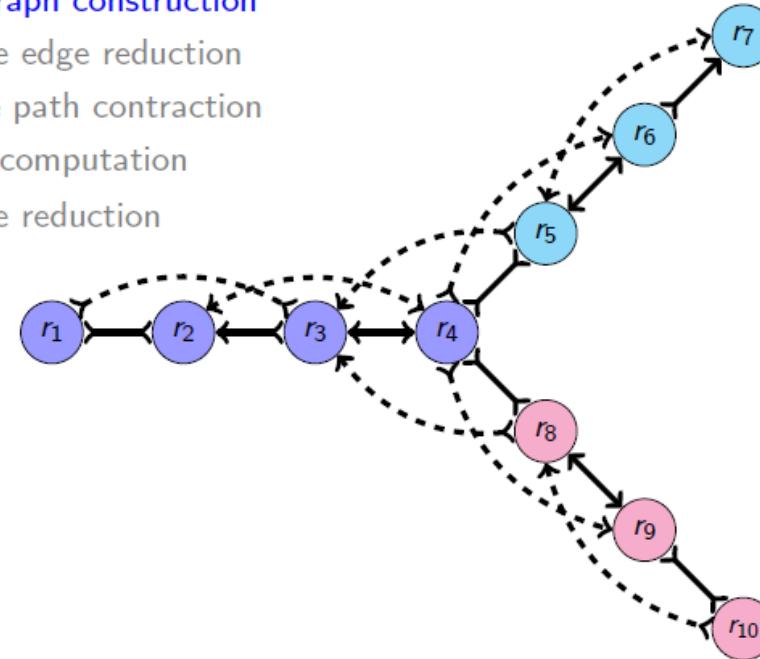
Overlap graph construction

Transitive edge reduction

Composite path contraction

Flow computation

Tree reduction



Layout

Read r_1 : AGCTAACGATTTACGATAGCCGATAGCTAAATTAC
Read r_2 : CGTAATTAGCTATCGGCTATCGTAAATGCTTAGC
Read r_3 : AACGTAAATTAGCTATCGGCTATCGTAAATGCTTA
Read r_4 : GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read r_5 : GTATAACGTAATTAGCTATCGGCTATCGTAAATG
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read r_8 : ATATAACGTAATTAGCTATCGGCTATCGTAAATG
Read r_9 : ATTTACGATAGCCGATAGCTAAATTACGTTATATA
Read r_{10} : CTATATAACGTAATTAGCTATCGGCTATCGTAAA

Set of given reads

Read r_1 : AGCTAACGATTTACGATAGCCGATAGCTAAATTAC
Read \bar{r}_2 : GCTAACGATTTACGATAGCCGATAGCTAAATTACG
Read \bar{r}_3 : TAAGCATTTACGATAGCCGATAGCTAAATTACGTT
Read r_4 : GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read \bar{r}_5 : CATTACGATAGCCGATAGCTAAATTACGTTATAC
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read \bar{r}_8 : CATTACGATAGCCGATAGCTAAATTACGTTATAT
Read r_9 : ATTTACGATAGCCGATAGCTAAATTACGTTATATA
Read \bar{r}_{10} : TTTACGATAGCCGATAGCTAAATTACGTTATATAG

Overlapping reads

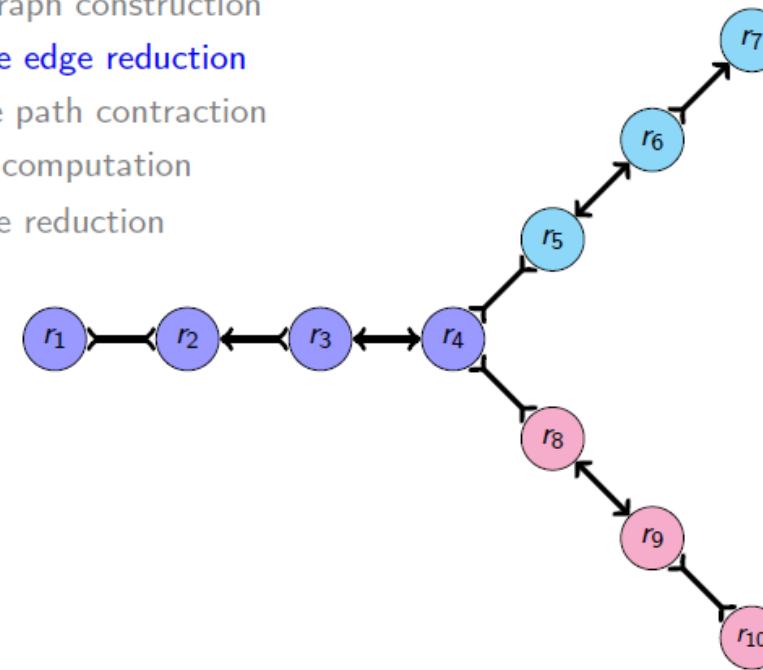
Overlap graph construction

Transitive edge reduction

Composite path contraction

Flow computation

Tree reduction



Layout

Read r_1 : AGCTAACGATTTACGATAGCCGATAGCTAAATTAC
Read r_2 : CGTAATTTAGCTATCGGCTATCGTAAATGCTTAGC
Read r_3 : AACGTAATTTAGCTATCGGCTATCGTAAATGCTTA
Read r_4 : GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read r_5 : GTATAACGTAATTTAGCTATCGGCTATCGTAAATG
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read r_8 : ATATAACGTAATTTAGCTATCGGCTATCGTAAATG
Read r_9 : ATTTACGATAGCCGATAGCTAAATTACGTTATATA
Read r_{10} : CTATATAACGTAATTTAGCTATCGGCTATCGTAAA
Set of given reads

Read r_1 : AGCTAACGATTTACGATAGCCGATAGCTAAATTAC
Read \bar{r}_2 : GCTAACGATTTACGATAGCCGATAGCTAAATTACG
Read \bar{r}_3 : TAAGCATTACGATAGCCGATAGCTAAATTACGTT
Read r_4 : GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read \bar{r}_5 : CATTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read \bar{r}_8 : CATTACGATAGCCGATAGCTAAATTACGTTATAT
Read r_9 : ATTTACGATAGCCGATAGCTAAATTACGTTATATA
Read \bar{r}_{10} : TTTACGATAGCCGATAGCTAAATTACGTTATATAG
Overlapping reads

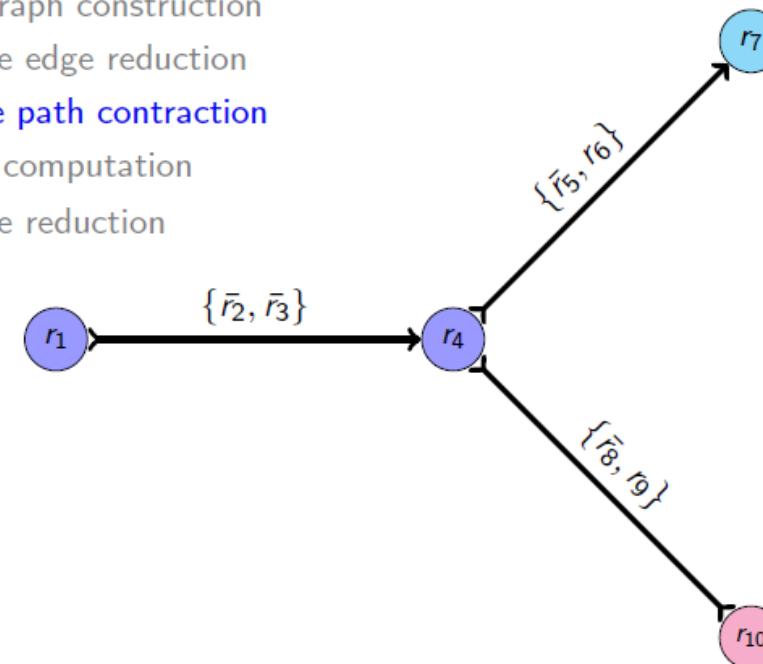
Overlap graph construction

Transitive edge reduction

Composite path contraction

Flow computation

Tree reduction



Layout

Read r_1 : AGCTAACGATTACGATAGCCGATAGCTAAATTAC
Read r_2 : CGTAATTAGCTATCGGCTATCGTAAATGCTTAGC
Read r_3 : AACGTAAATTAGCTATCGGCTATCGTAAATGCTTA
Read r_4 : GCATTACGATAGCCGATAGCTAAATTACGTTATA
Read r_5 : GTATAACGTAATTAGCTATCGGCTATCGTAAATG
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read r_8 : ATATAACGTAATTAGCTATCGGCTATCGTAAATG
Read r_9 : ATTTACGATAGCCGATAGCTAAATTACGTTATATA
Read r_{10} : CTATATAACGTAATTAGCTATCGGCTATCGTAAA

Set of given reads

Read r_1 : AGCTAACGATTACGATAGCCGATAGCTAAATTAC
Read \bar{r}_2 : GCTAACGATTACGATAGCCGATAGCTAAATTACG
Read \bar{r}_3 : TAAGCATTTACGATAGCCGATAGCTAAATTACGTT
Read r_4 : GCATTACGATAGCCGATAGCTAAATTACGTTATA
Read \bar{r}_5 : CATTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read \bar{r}_8 : CATTACGATAGCCGATAGCTAAATTACGTTATAT
Read r_9 : ATTTACGATAGCCGATAGCTAAATTACGTTATATA
Read \bar{r}_{10} : TTTACGATAGCCGATAGCTAAATTACGTTATATAG

Overlapping reads

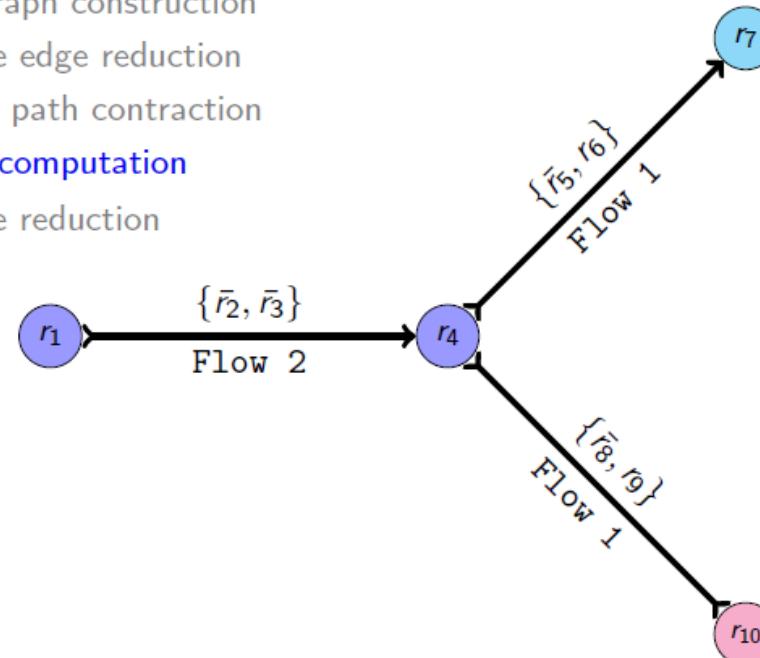
Overlap graph construction

Transitive edge reduction

Composite path contraction

Flow computation

Tree reduction



Layout

Read r_1 : AGCTAACGATTTACGATAGCCGATAGCTAAATTAC
Read r_2 : CGTAATTAGCTATCGGCTATCGTAAATGCTTAGC
Read r_3 : AACGTAAATTAGCTATCGGCTATCGTAAATGCTTA
Read r_4 : GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read r_5 : GTATAACGTAATTAGCTATCGGCTATCGTAAATG
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read r_8 : ATATAACGTAATTAGCTATCGGCTATCGTAAATG
Read r_9 : ATTTACGATAGCCGATAGCTAAATTACGTTATATA
Read r_{10} : CTATATAACGTAATTAGCTATCGGCTATCGTAAA

Set of given reads

Read r_1 : AGCTAACGATTTACGATAGCCGATAGCTAAATTAC
Read \bar{r}_2 : GCTAACGATTTACGATAGCCGATAGCTAAATTACG
Read \bar{r}_3 : TAAGCATTACGATAGCCGATAGCTAAATTACGTT
Read r_4 : GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read \bar{r}_5 : CATTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read \bar{r}_8 : CATTACGATAGCCGATAGCTAAATTACGTTATATA
Read r_9 : ATTTACGATAGCCGATAGCTAAATTACGTTATATA
Read \bar{r}_{10} : TTTACGATAGCCGATAGCTAAATTACGTTATATACT

Overlapping reads

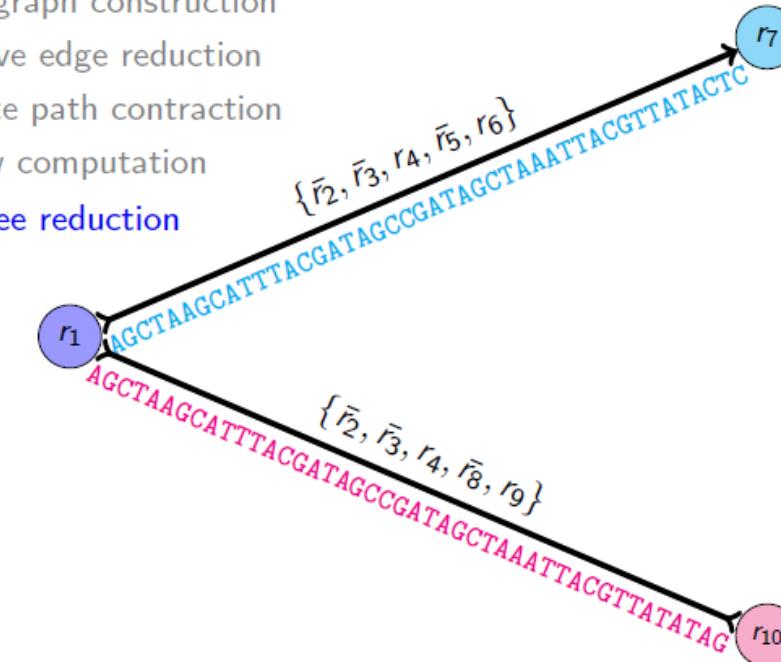
Overlap graph construction

Transitive edge reduction

Composite path contraction

Flow computation

Tree reduction



Contig 1:

AGCTAACGATTTACGATAGCCGATAGCTAAATTACGTTATACTC

Contig 2:

AGCTAACGATTTACGATAGCCGATAGCTAAATTACGTTATATA

Contig is a set of overlapping DNA segments.

Consensus

TAGATTACACAGATTACTGA TTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAAACTA
TAG TTACACAGATTATTGACTTCATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA

↓ ↓ ↓ ↓ ↓

TAGATTACACAGATTACTGACTTGATGGCGTAA CTA



Take reads that make up a contig and line them up

Take *consensus*, i.e. majority vote

At each position, ask: what nucleotide (and/or gap) is here?

Complications: (a) sequencing error, (b) ploidy

Say the true genotype is AG, but we have a high sequencing error rate and only about 6 reads covering the position.

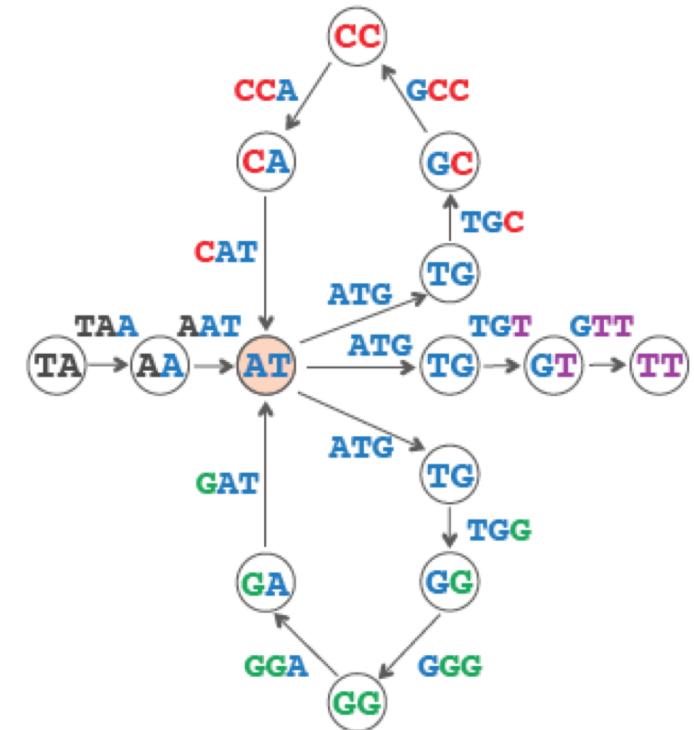
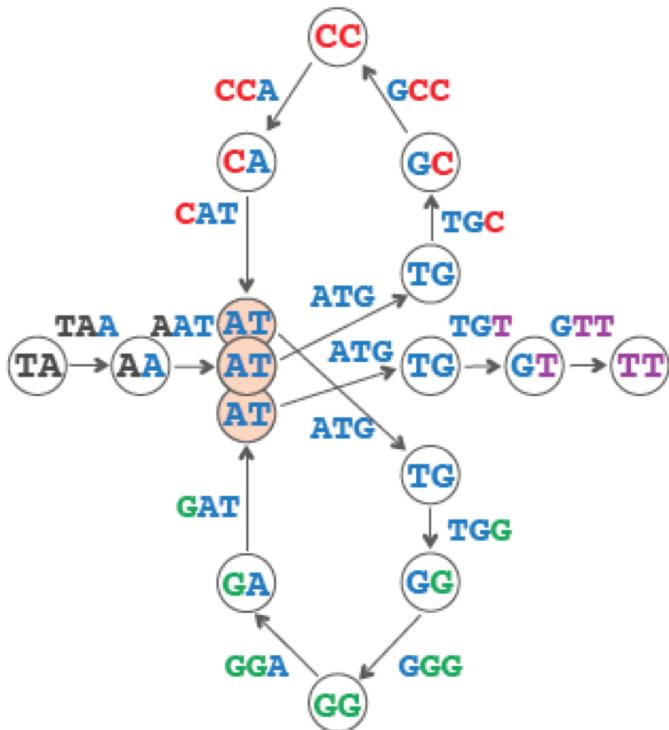
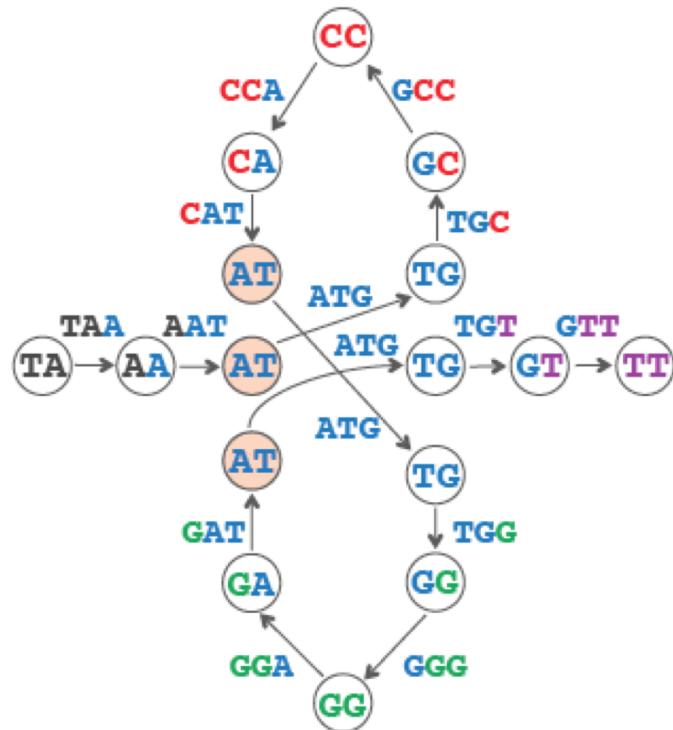
Some OLC-based assemblers

- Celera Assembler → CANU
- Celera Assembler with the Best Overlap Graph (CABOG)
 - Designed for Sanger sequences, but works with 454 and error-corrected PacBio reads
- Newbler, a.k.a. GS *de novo* Assembler
 - Designed for 454 sequences, but works with Sanger reads
- Omega, overlap-graph *de novo* assembler for metagenomics

OLC drawbacks

- Building overlap graph is slow.
- Overlap graph is big; one node per read, and in practice # edges grows superlinearly with # reads
- 2nd-generation sequencing datasets are ~ 100s of millions or billions of reads, hundreds of billions of nucleotides total

De Bruijn graphs

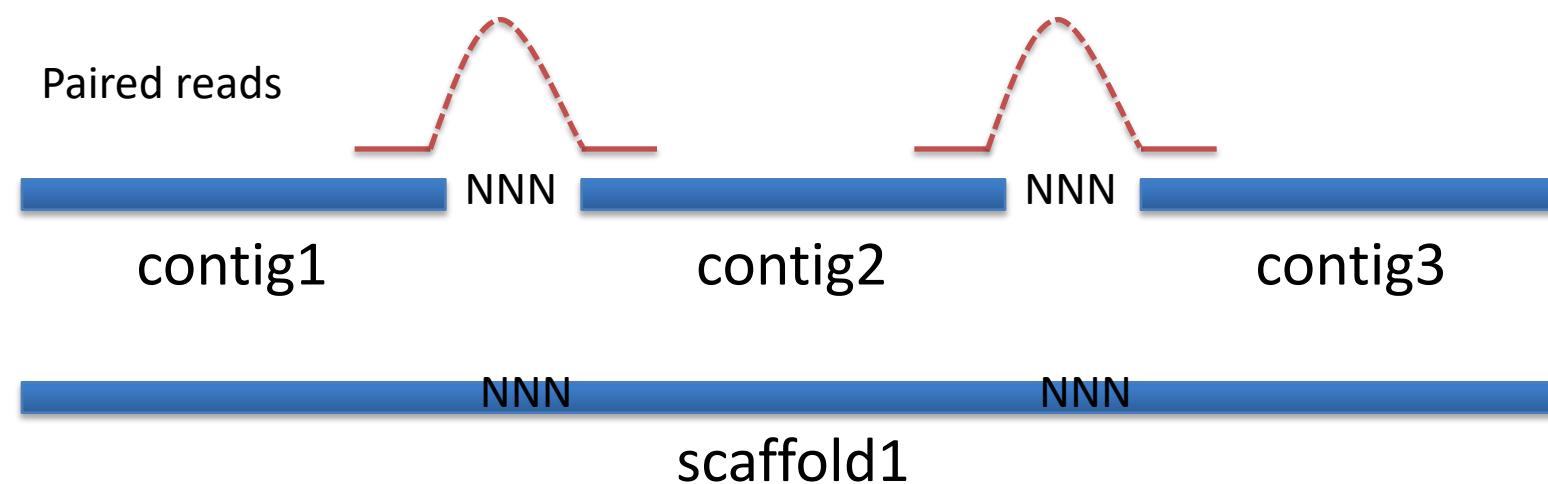


Examples of DBG-based assemblers

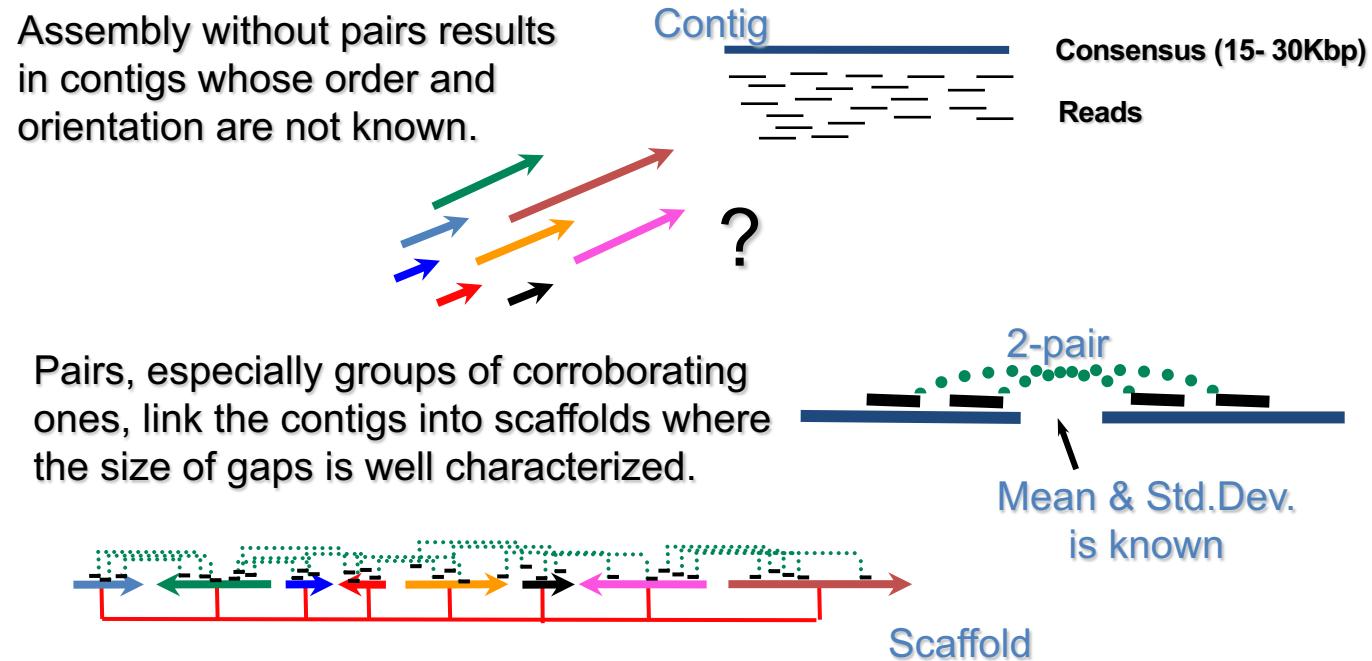
- **EULER** (P. Pevzner), the first assembler to use DBG
- **Velvet** (D. Zerbino), a popular choice for small genomes
- **SOAPdenovo** (BGI), widely used by BGI and for relatively unstructured assemblies
- **ALLPATHS-LG**, probably the most reliable assembler for large genomes (but with strict input requirements)

Contig vs Scaffold

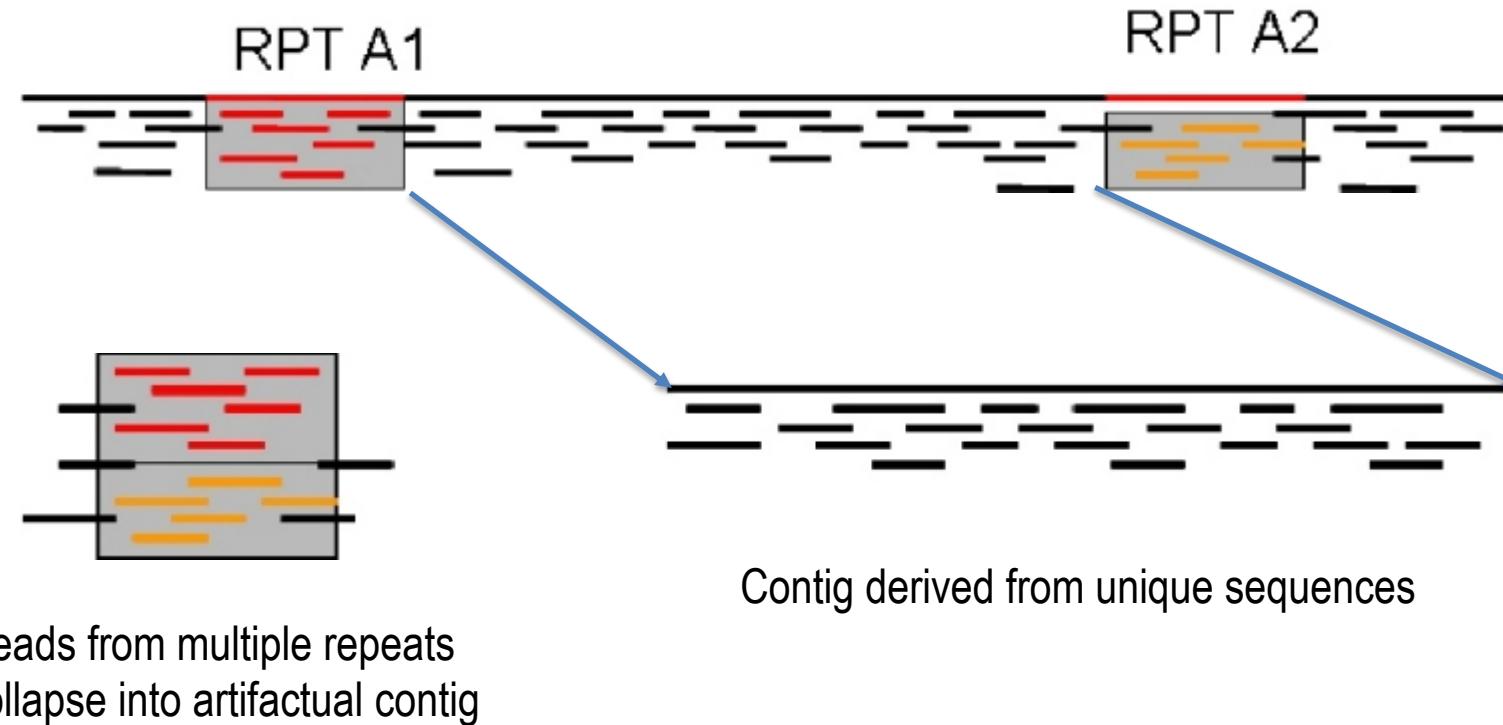
- A contig (from contiguous) is a set of overlapping DNA segments that together represent a consensus region of DNA.
- A scaffold is composed of contigs and gaps.
 - Gap length can be guessed by incorporating information from paired ends or mate pairs of different insert sizes.
 - Several contigs stitched together with NNNs in between



Pairs give order and orientation



Repeats often split genome into contigs



Handling Repeats

1. Repeat detection

- **pre-assembly:** find fragments that belong to repeats
 - statistically (most existing assemblers)
 - repeat database (*RepeatMasker*)
- **during assembly:** detect "tangles" indicative of repeats (Pevzner, Tang, Waterman 2001)
- **post-assembly:** find repetitive regions and potential mis-assemblies.
 - *Reputer, RepeatMasker*
 - "unhappy" mate-pairs (too close, too far, mis-oriented)

2. Repeat resolution

- find DNA fragments belonging to the repeat
- determine correct tiling across the repeat
- Obtain long reads spanning repeats

How good is my assembly?

- How much total sequence is in the assembly relative to estimated genome size?
- How many pieces, and what is their size distribution?
- Are the contigs assembled correctly?
- Are the scaffolds connected in the right order / orientation?
- How were the repeats handled?
- Are all the genes I expected in the assembly?

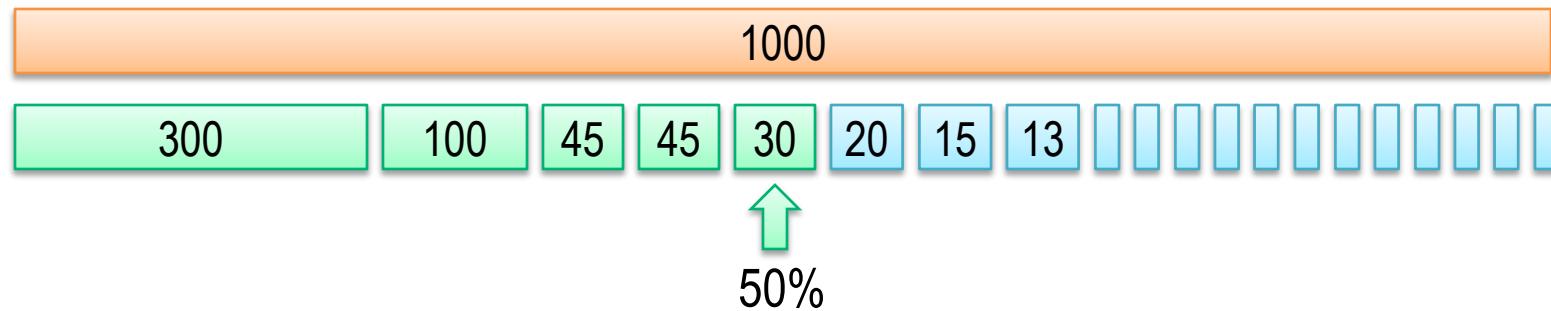
Quality of Assembly? What is N50?

Genome assembly evaluation tool

- GAGE, REARP, QUAST

Def: 50% of the genome is in contigs as large as the N50 value

Example: 1Mbp genome



N50 size = 30 kbp

$$(300k + 100k + 45k + 45k + 30k = 520k \geq 500\text{kbp})$$

NG50 - compared with genome size rather than assembly size

- N50 - contigs of this size or larger include 50 % of the assembly
- NG50 - contigs of this size or larger include 50 % of the genome
- NG50 is a better approximation of assembly quality, but can sometimes not be calculated, e.g., the genome size is unknown
- Can be quite different from N50, e.g., genome is 1,5 Gb but assembly is 1 Gb due to non-assembled repeats

De Novo Genome Assembly Steps

- Quality control with [FastQC](#)
- Data trimming/filtering and error correction using [BBTools](#)
- De Novo Genome assembly with [Velvet](#)
- Another De Novo Genome assembly with [Spades](#)
- Validate the assembly with [Quast](#)