Genome Assembly Overview

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.

Sequencing Techniques

Next (Second) Generation Sequencing

- ILLUMINA
- MGI
- ION TORRENT
- 454 Life Sciences (Roche)
- SOLID, ….

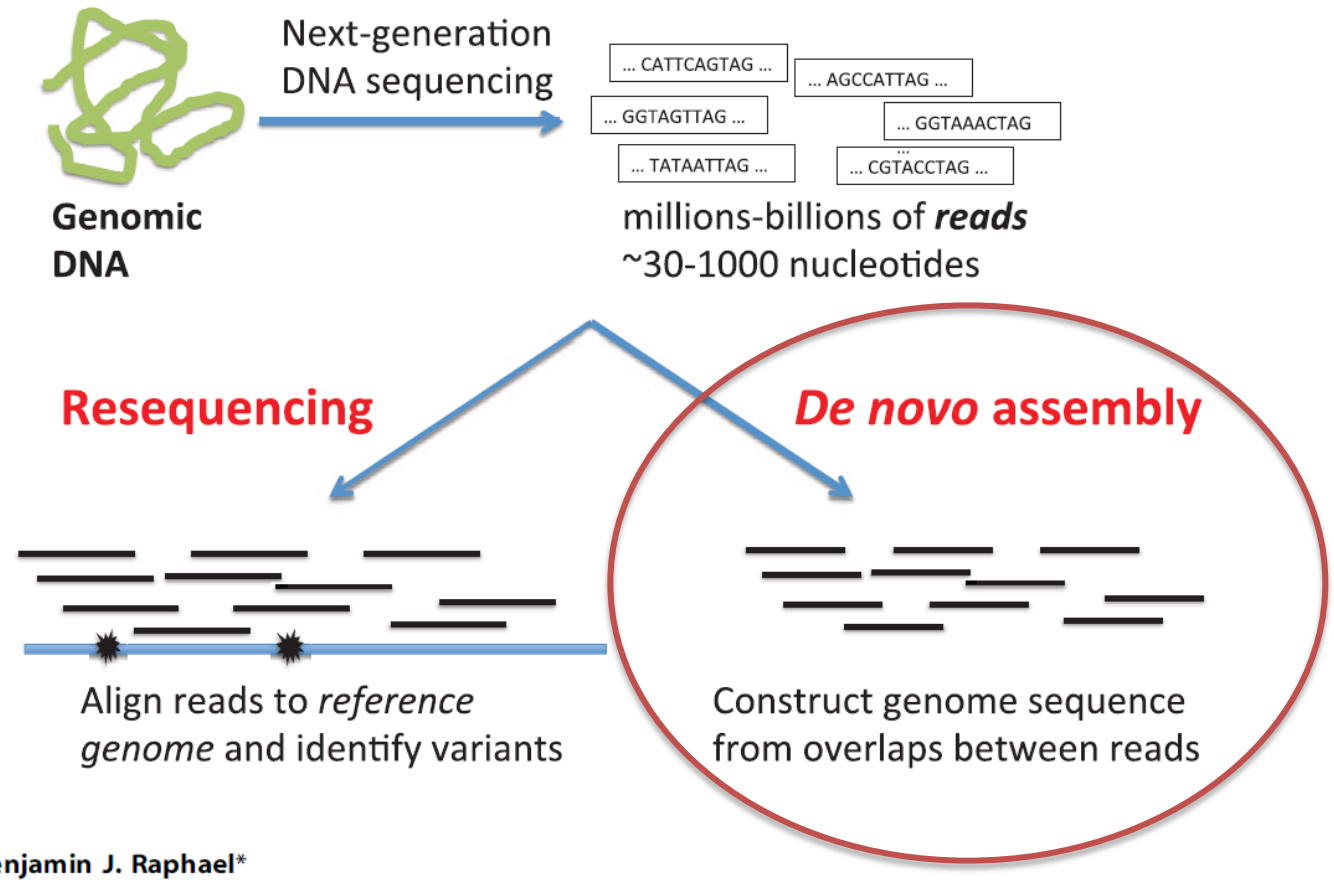
- Read length ≤ 300 bases
- Extremely high read number
- Low cost

Third Generation Sequencing

- PACIFIC BIOSCIENCES
- OXFORD NANOPORE
- BioNano Genomics
- (optic cards)

- Single molecule sequencing
- Real time sequencing
- Maximum read length > 50 kb

Genome Assembly with or without Reference



Benjamin J. Raphael*

Department of Computer Science and Center for Computational Molecular Biology, Brown University, Providence, Rhode Island, United States of America

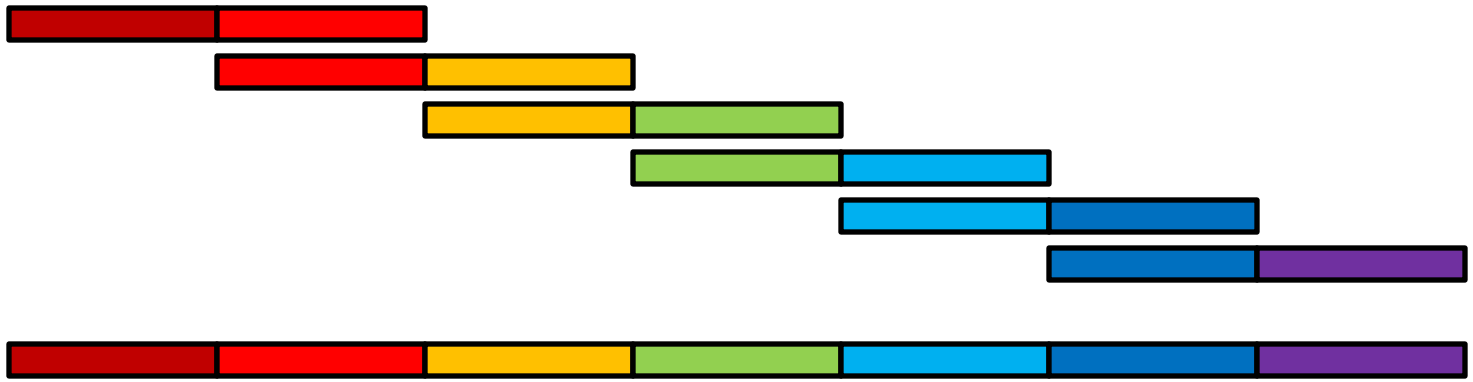
De Novo Assembly of Genome

1. Shear & Sequence DNA

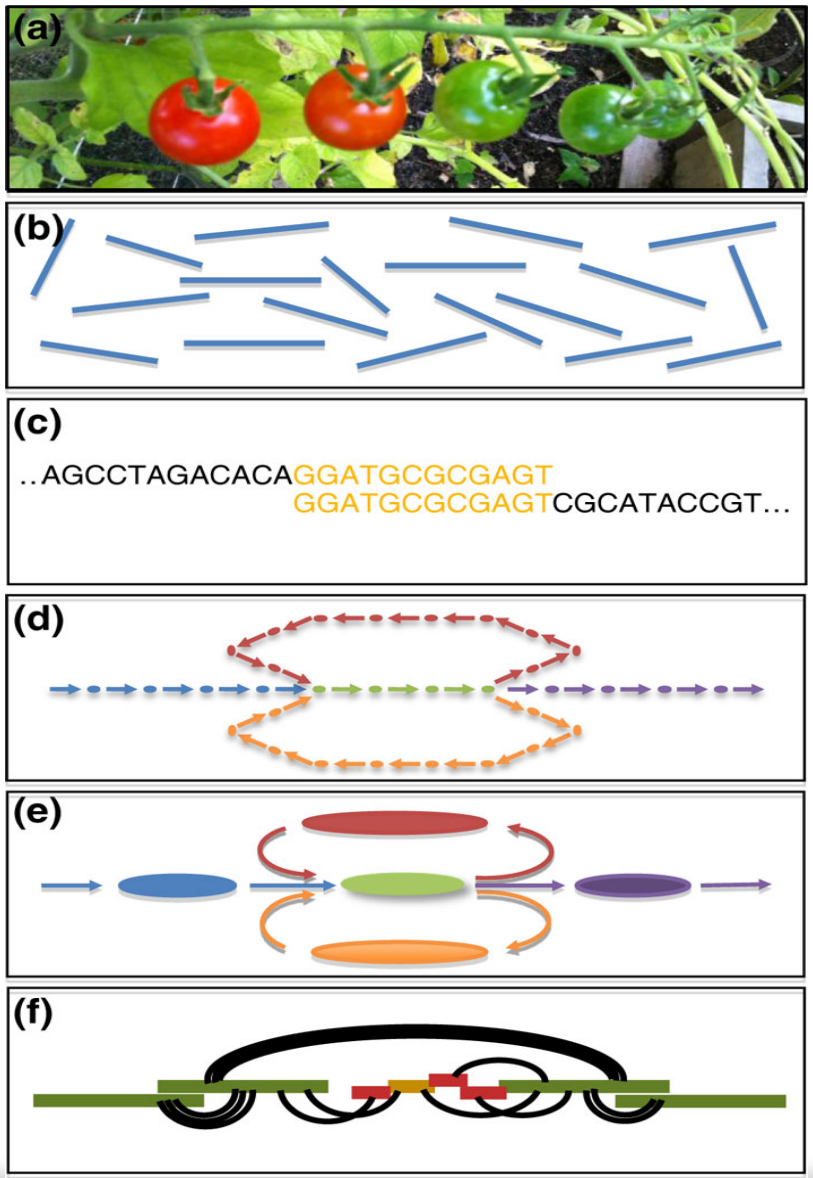
2. Construct assembly graph from overlapping reads

```
…AGCCTAGGATGCGCGACACGT
            TGCGCGACACGTCGCATATCCGGTTGAT
                                TCCGGTTGATCACGAATACG…
```

3. Simplify assembly graph



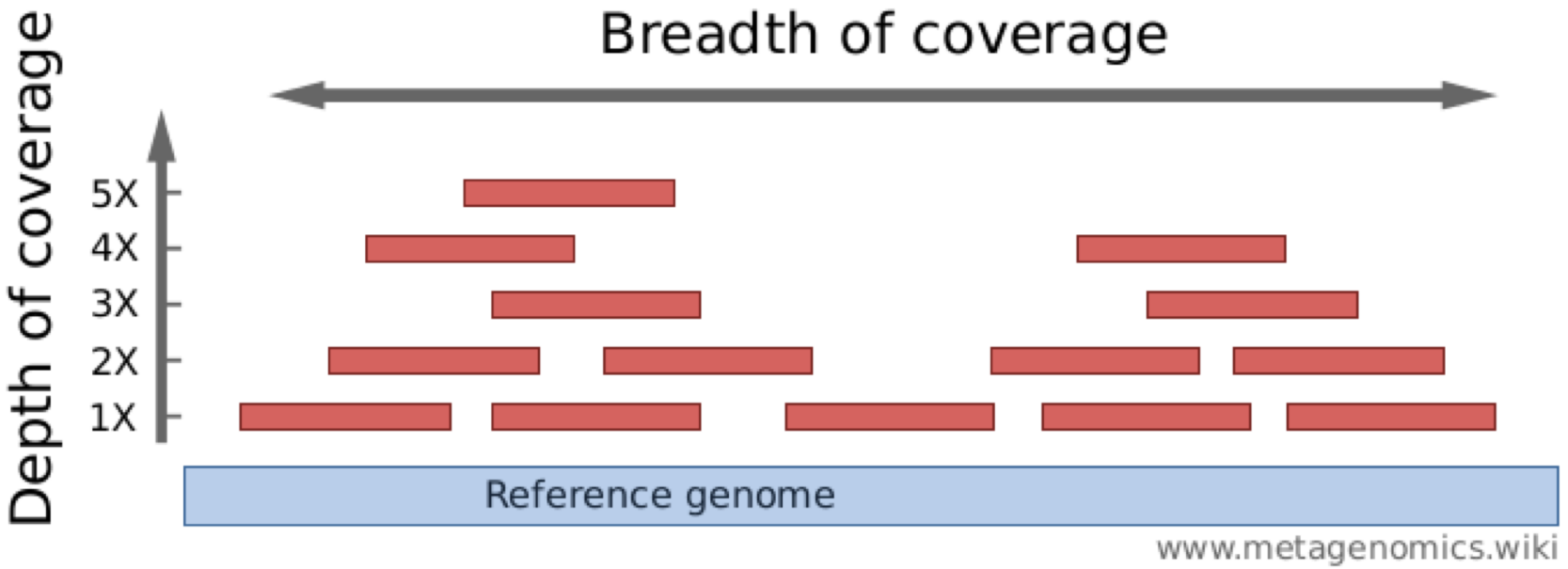
Schematic Overview of Genome Assembly via NGS



Schematic overview of genome assembly. (a) DNA is collected from the biological sample and sequenced. (b) The output from the sequencer consists of many billions of short, unordered DNA fragments from random positions in the genome. (c) The short fragments are compared with each other to discover how they overlap. (d) The overlap relationships are captured in a large assembly graph shown as nodes representing *k-mers* or reads, with edges drawn between overlapping *k-mers* or reads. (e) The assembly graph is refined to correct errors and simplify into the initial set of contigs, shown as large ovals connected by edges. (f) Finally, mates, markers and other long-range information are used to order and orient the initial contigs into large scaffolds, as shown as thin black lines connecting the initial contigs.

Schatz et al. Genome Biology 2012 13:243

Coverage

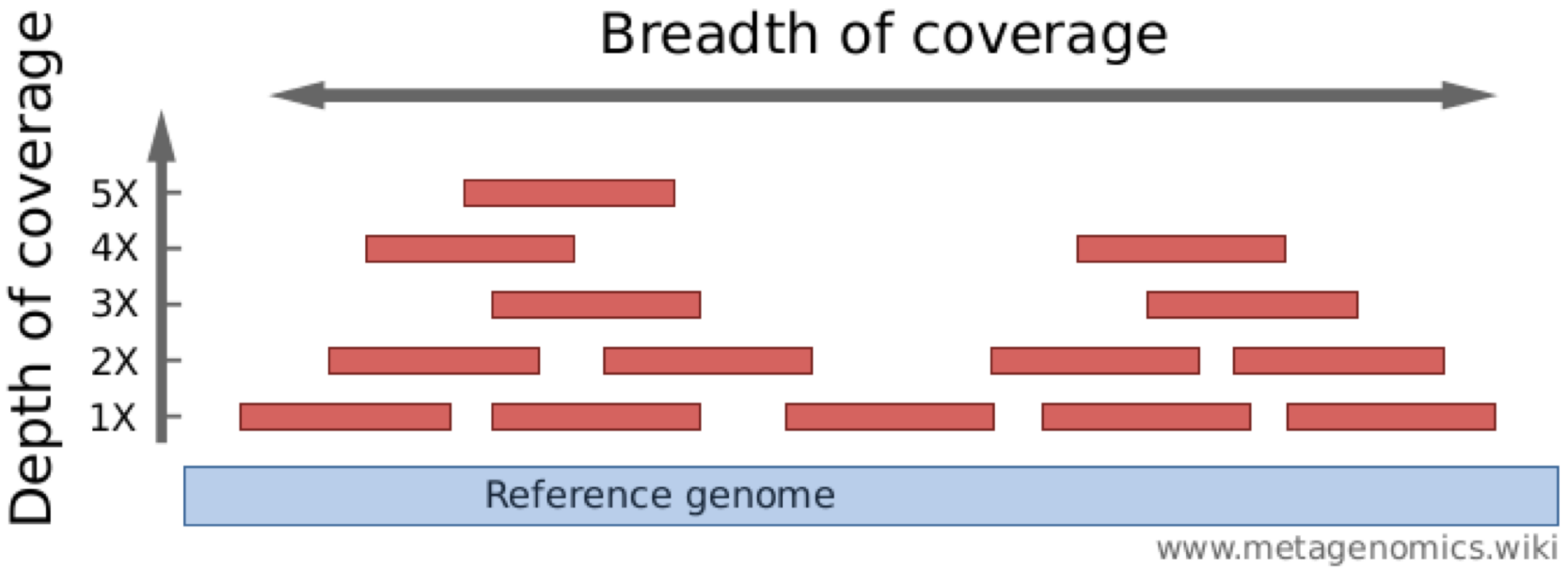


Depth of coverage

How strong is a genome "covered" by sequenced fragments (short reads)?

Per-base coverage is the average number of times a base of a genome is sequenced. The coverage depth of a genome is calculated as the number of bases of all short reads that match a genome divided by the length of this genome. It is often expressed as 1X, 2X, 3X,... (1, 2, or, 3 times coverage).

Coverage



Breadth of coverage

How much of a genome is "covered" by short reads? Are there regions that are not covered, even not by a single read?

Breadth of coverage is the percentage of bases of a reference genome that are covered with a certain depth. For example: 90% of a genome is covered at 1X depth; and still 70% is covered at 5X depth.

Be careful to use Coverage

- In genome assembly, “coverage” usually refers “coverage depth”

Length of genomic segment: ***L***

Number of reads: ***n*** (Depth of) Coverage ***C = n l / L***

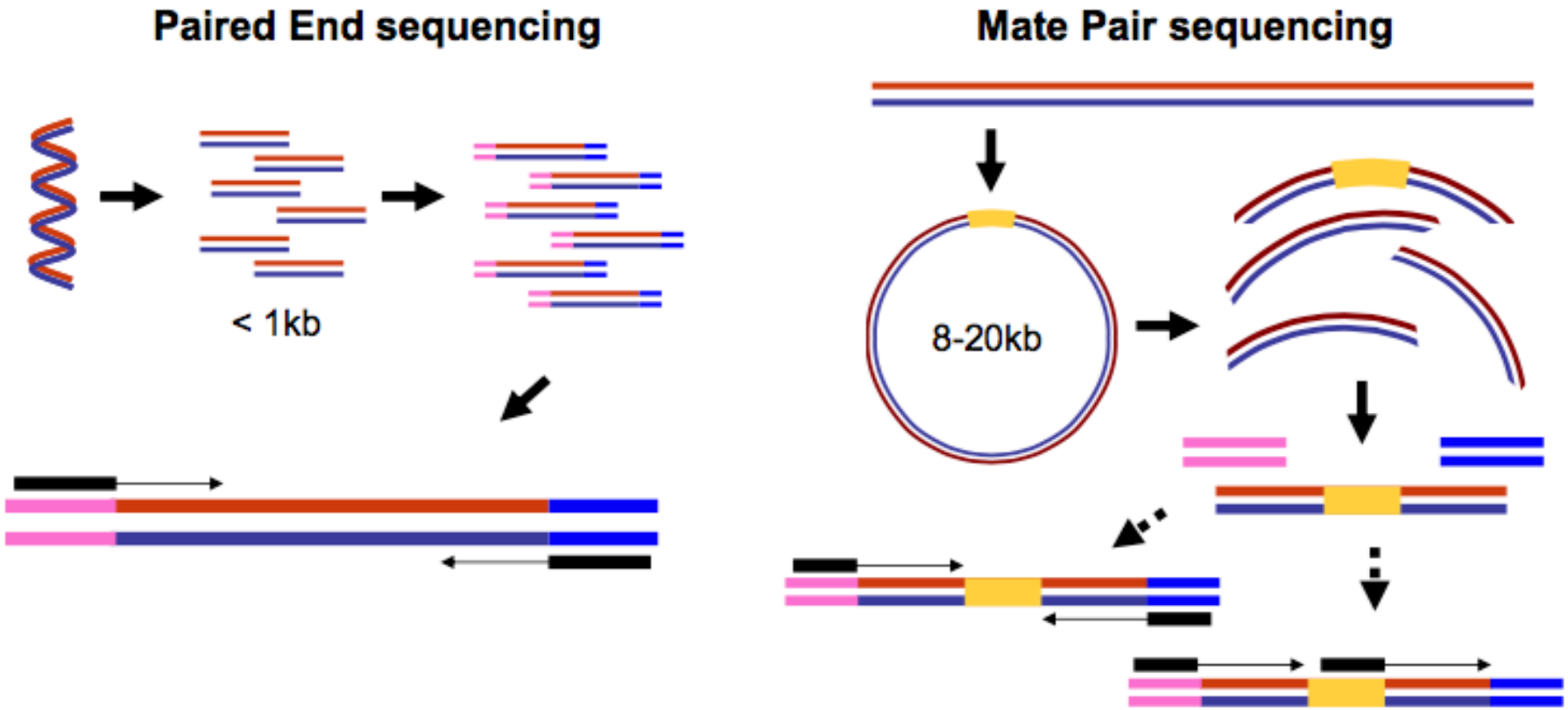
Length of each read: ***l***

- You can calculate the “Breath of coverage” by simple calculation of physically covered region of the genome.

Illumina NGS

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

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

- 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.

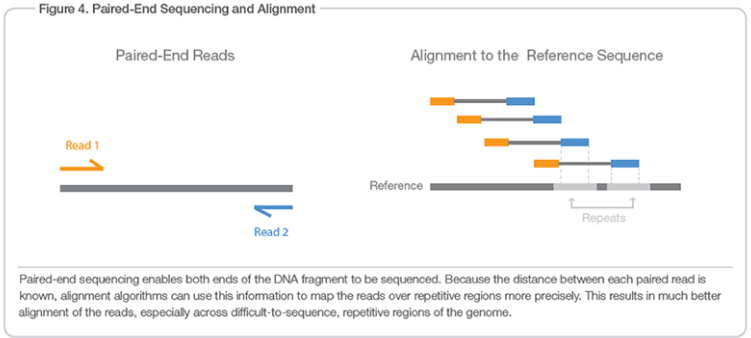
Paired-End Sequencing Highlights

Simple Paired-End Libraries: Simple workflow allows generation of unique ranges of insert sizes

Efficient Sample Use: Requires the same amount of DNA as single-read gDNA or cDNA sequencing

Broad Range of Applications: Does not require methylation of DNA or restriction digestion; can be used for bisulfite sequencing

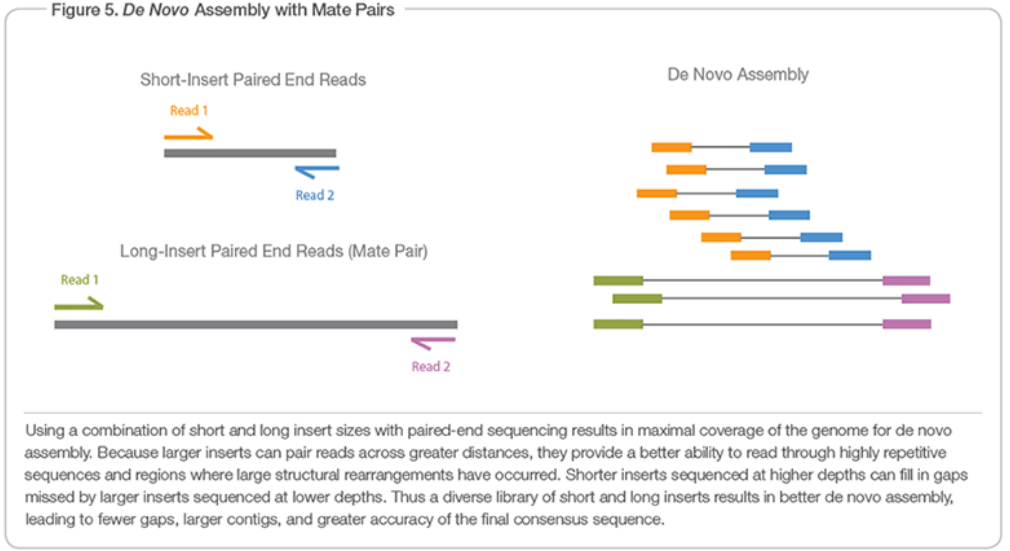
Simplified Data Analysis: Higher quality sequence assemblies with short-insert libraries



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

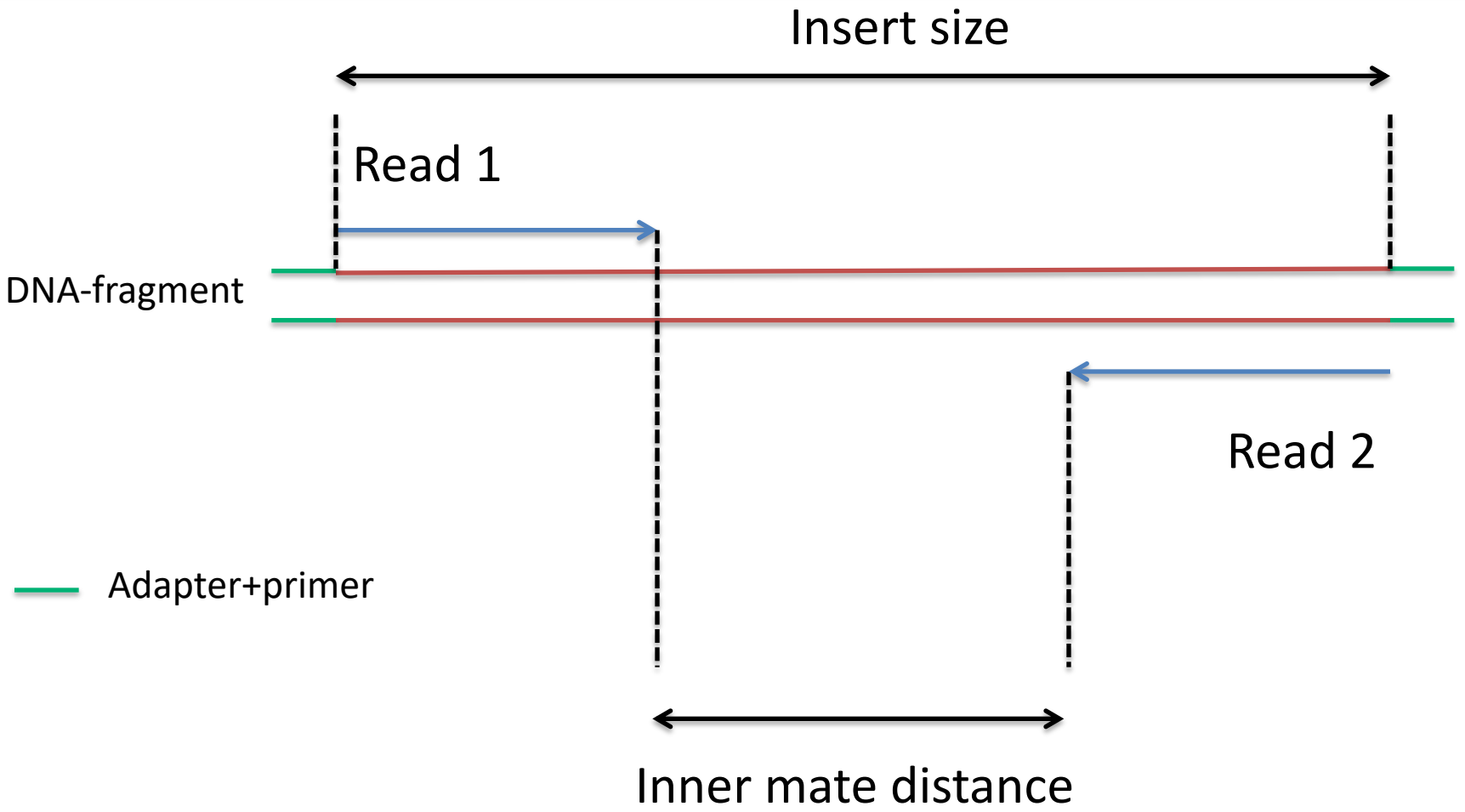


Combining data generated from mate pair library sequencing with that from short-insert paired-end reads provides a powerful combination of read lengths for maximal sequencing coverage across the genome

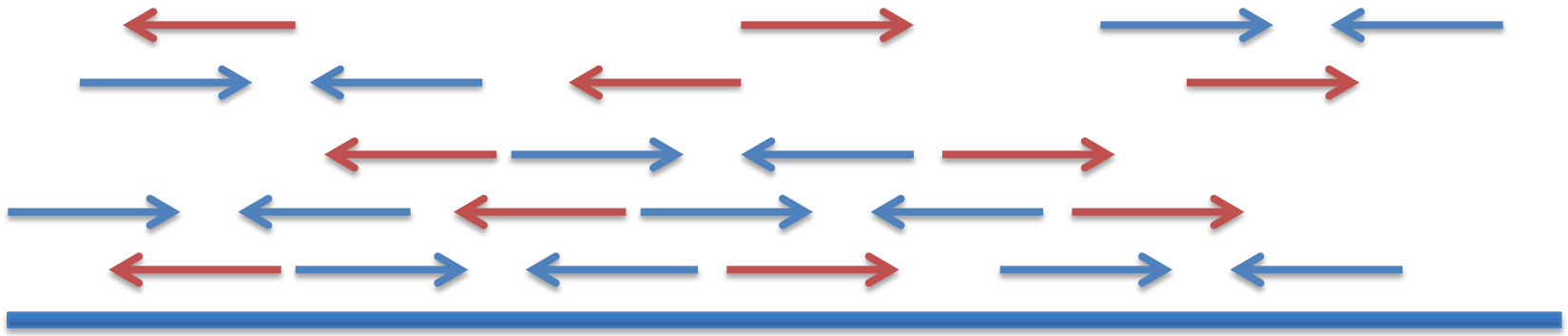
Mate Pair Library Preparation Process

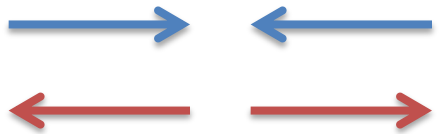
https://www.illumina.com/technology/next-generation-sequencing/mate-pair-sequencing_assay.html

Insert Size


Insert size
Read 1
DNA-fragment
Read 2
Adapter+primer
Inner mate distance

Orientation of paired reads

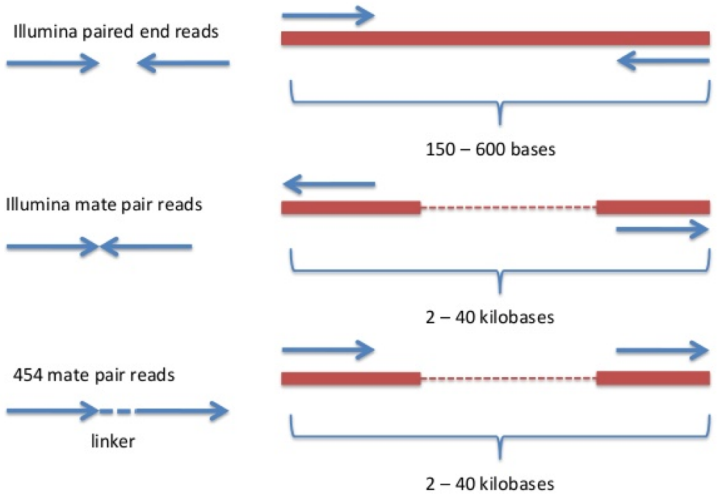




Paired end (PE) reads

Mate pair (MP) 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 were reads for each end are interleaved. Here are examples:

Two single files

File 1

```
@M02286:19:000000000-AA549:1:1101:12677:1273 1:N:0:23
CCTACGGGTGGCAGCAGTGAGGAATATTGGTCAATGGACGGAAGTCT
+
ABC8C,:@F:CE8,B-,C,-6-9-C,CE9-CC--C-<-C++,,+;CE
@M02286:19:000000000-AA549:1:1101:15048:1299 1:N:0:23
CCTACGGGTGGCTGCAGTGAGGAATATTGGACAATGGTCGGAAGACT
+
ABC@CC77CFCEG;F9<F89<9--C,CE,--C-6C-,CE:++7:,CF
```

File 2

```
@M02286:19:000000000-AA549:1:1101:12677:1273 2:N:0:23
CACTACCCGTGTATCTAATCCTGTTTGATACCCGCACCTTCGAGCTTA
+
--8A,CCE+,,;,<CC,,<CE@,CFD,,C,CFF+@+@CCEF,,,B+C,
@M02286:19:000000000-AA549:1:1101:15048:1299 2:N:0:23
CACTACCGGGGTATCTAATCCTGTTCGCTCCCCACGCTTTCGTCCATC
+
-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 were reads for each end are interleaved. Here are examples:

Interleaved file

```
@1/1
AGGGATGTGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTA
+
EGGEGGGDFGEEEAEECGDEGGFEEGEFGBEEDDECFEFDD@CDD<ED
@1/2
CCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAAC
+
GHHHDFDFGFGEGFBGEGGEGEGGGHGFGHFHFHHHHHHHHEF?EFEFF
@2/1
AGGGATGTGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTA
+
HHHHHHEGFHEEFEEHEEHHGGEGGGGEFGFGGGGHHHHFBEEEEEFG
@2/2
CCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAAC
+
HHHHHHHHHHHHHGHHHHHHGHHHHHHHHHHHHFHHHFHHHHHHHHHH
```

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 EE[1], 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

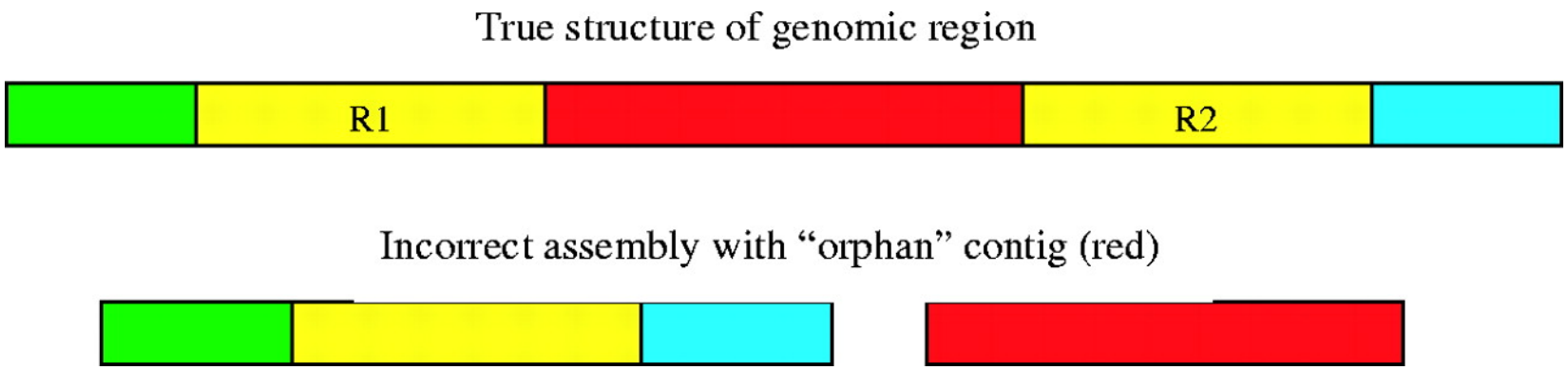
[Indexed for MEDLINE]

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

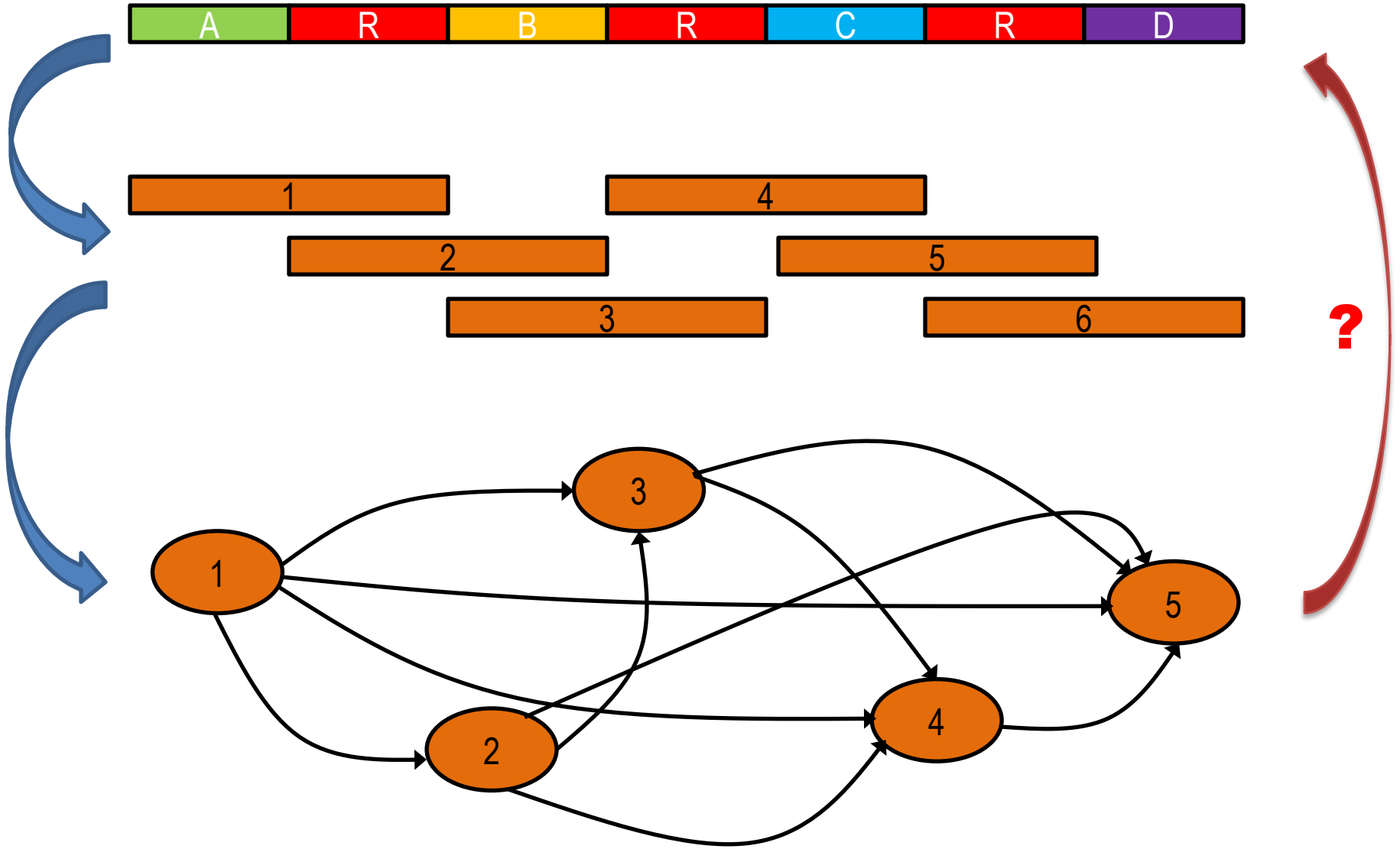
Repetitive Sequences

- Assemblies can collapse around repetitive sequences



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!

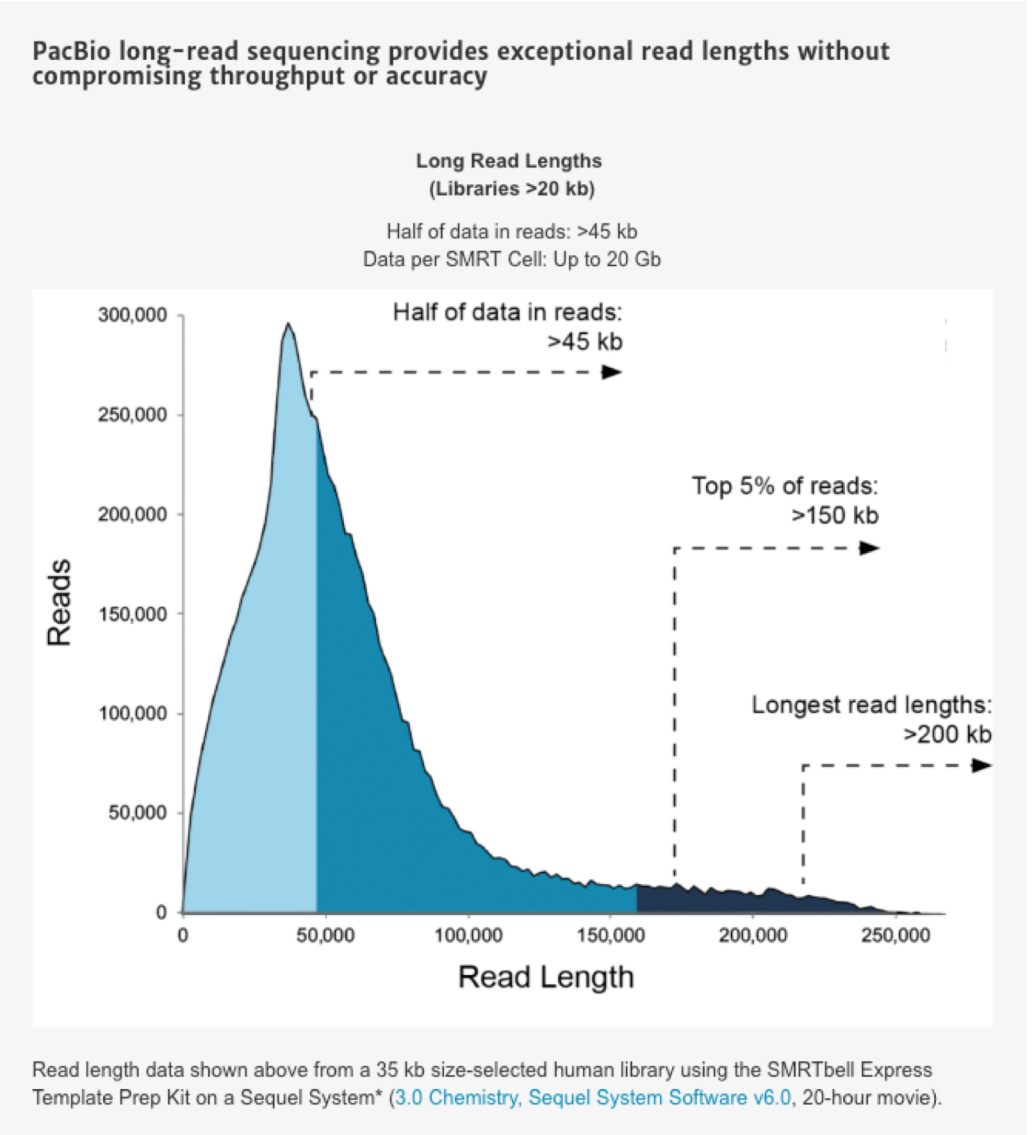
New Sequencing Technology Help Assembly

- Illumina
  - Huge yield, cheap, reliable, read length “long enough” (100-300 bp), industry standard=huge amount of available software
- Moleculo libraries
  - New technology acquired by Illumina, allows generation of fully assembled 10 kb sequences
- Illumina NovaSeq

https://www.illumina.com/content/dam/illumina-marketing/documents/products/datasheets/novaseq-series-specification-sheet-770-2016-025.pdf

New Sequencing Technology Help Assembly

- Pacific Biosciences
  - Long reads, single molecules
  - High error rate on longer fragments (15%), expensive
- Nanopore
  - Extremely long sequences, single molecule, portable (MinION)
  - Very high error rates (up to 38% reported)



Sequencing Technology & Assembly Algorithms

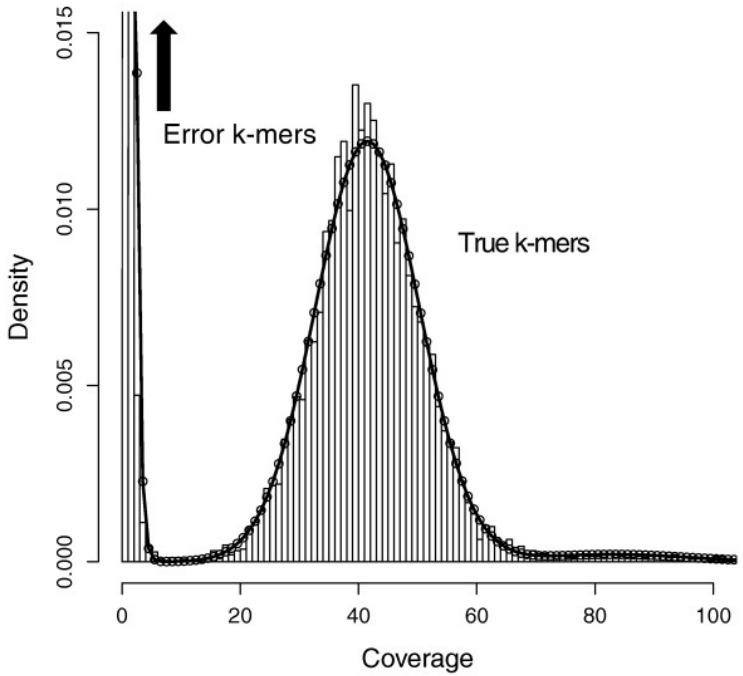
- Sanger sequencing: ~ 1000 bp
  - ***Overlap-layout-consensus*** algorithm dominated the assembly field
  - E.g., Celera assembler
- Illumina sequencing: <100 bp ➔ 100 bp ➔ 250~300 bp
  - OLC algorithm broke down completely for those short reads, because finding overlaps between all read pairs scaled 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 Oxforad Nanomore (~ 250,000 bp)
  - OLC algorithm (Canu) is tuned for these long sequences.

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

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
  - Part of several assemblers, also stand-alone Quake of khmer programs

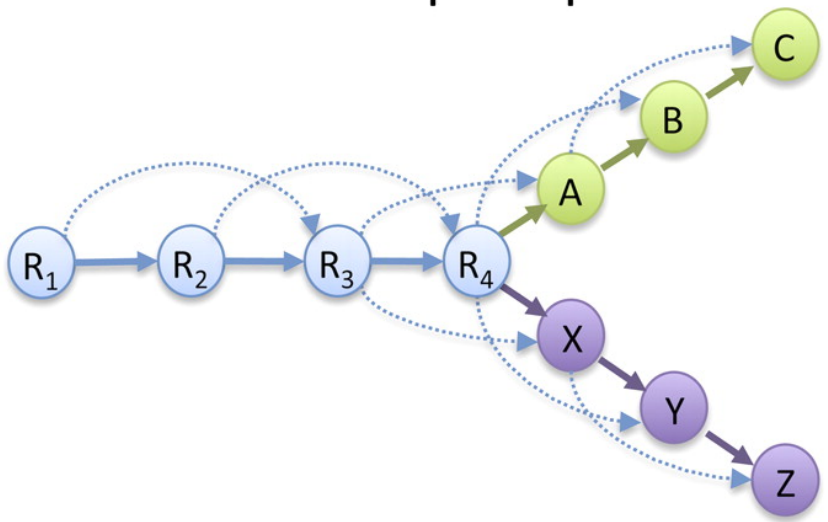


OLC vs De Bruijn

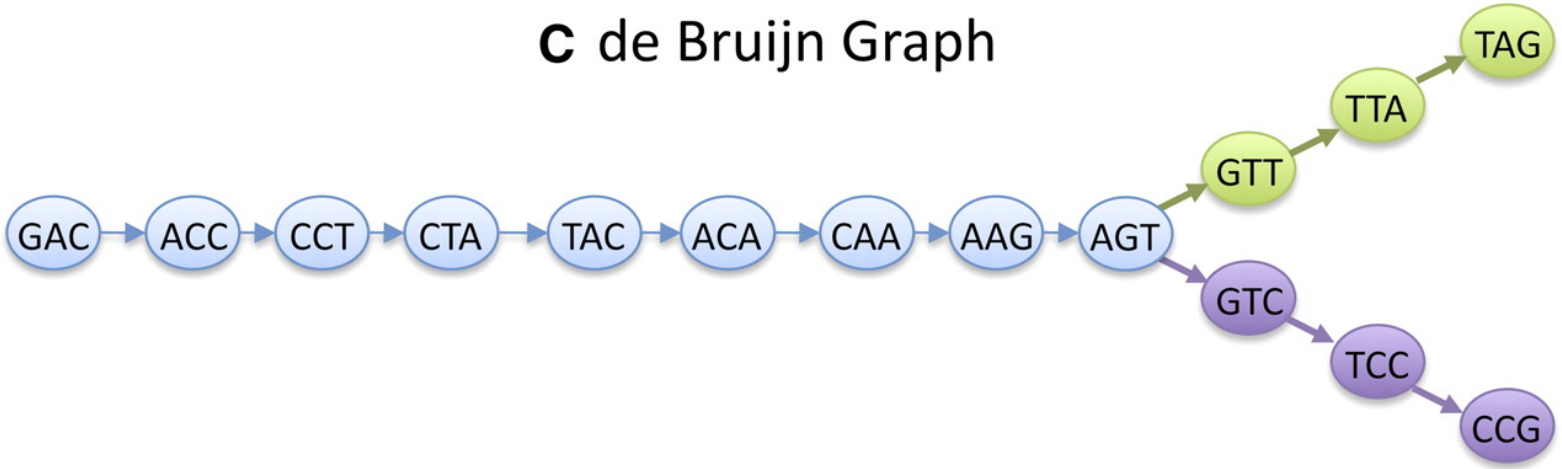
A Read Layout

```
R1: GACCTACA
R2:    ACCTACAA
R3:     CCTACAAG
R4:      CTACAAGT
A:        TACAAGTT
B:         ACAAGTTA
C:          CAAGTTAG
X:        TACAAGTC
Y:         ACAAGTCC
Z:          CAAGTCCG
```

B Overlap Graph



C de Bruijn Graph



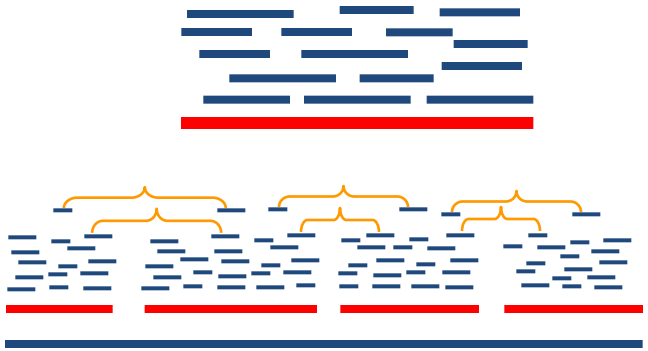
Overlap-Layout-Consensus

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



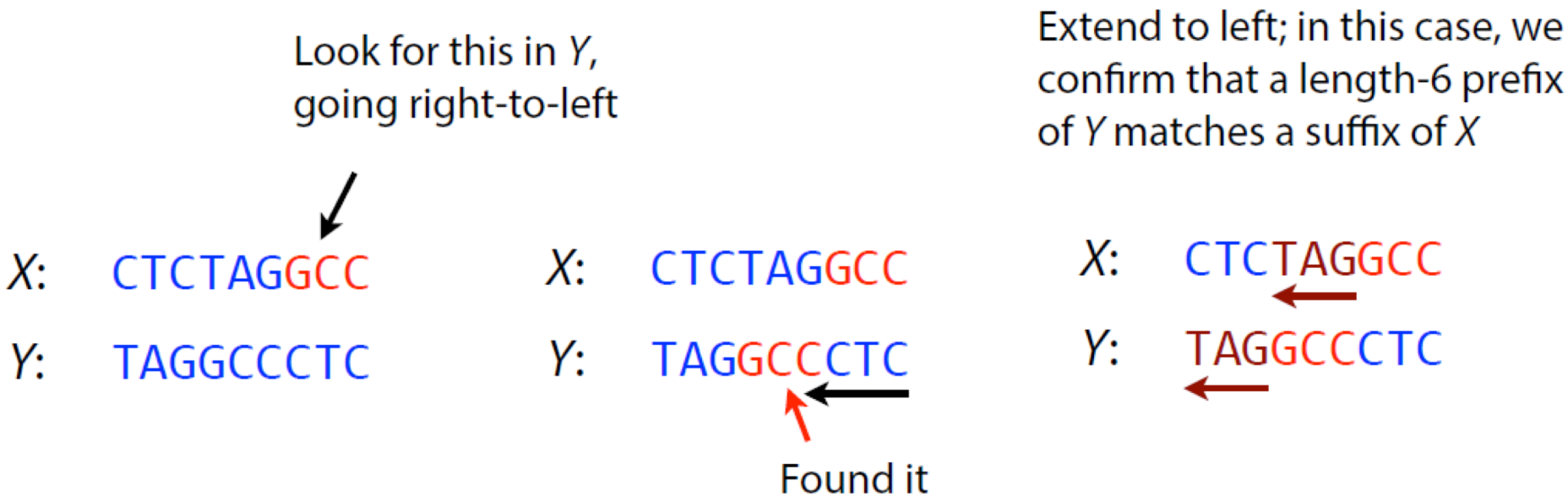
Consensus: derive the DNA sequence and correct read errors

..ACGATTACAATAGGTT..

Finding Overlaps

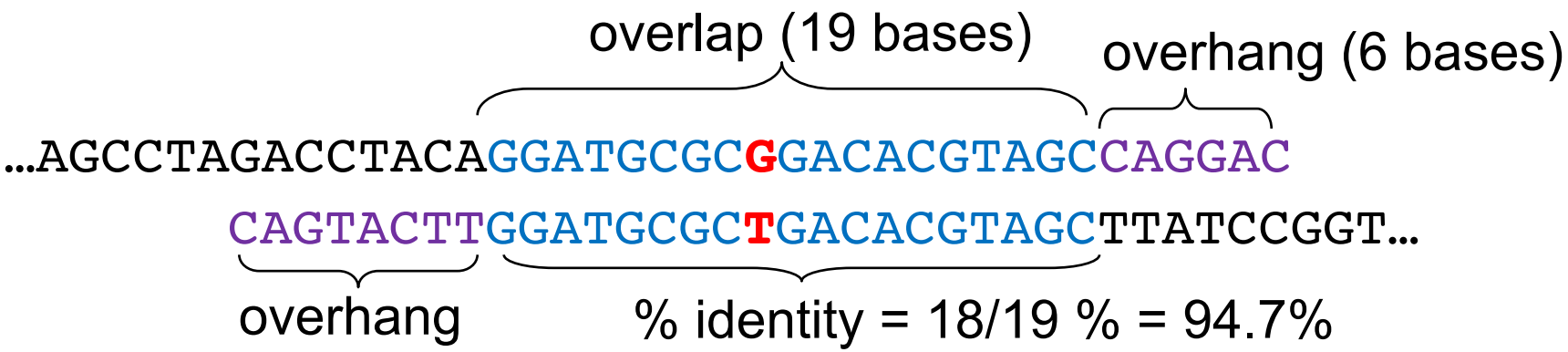
Can we be less naive than this?

Say $l = 3$



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

- Can we use suffix trees for overlapping?
- Problem: Given a collection of strings *S*, for each string *x* in *S* find all overlaps involving a prefix of *x* and a suffix of another string *y*
- Build a generalized suffix tree of the strings in *S*

Allow Mismatches?

What if we want to allow mismatches and gaps in the overlap?

I.e. How do we find the best *alignment* of a suffix of *X* to a prefix of *Y*?

```
X: CTCGGCCCTAGG
       |||  |||||
Y:     GGCTCTAGGCCC
```

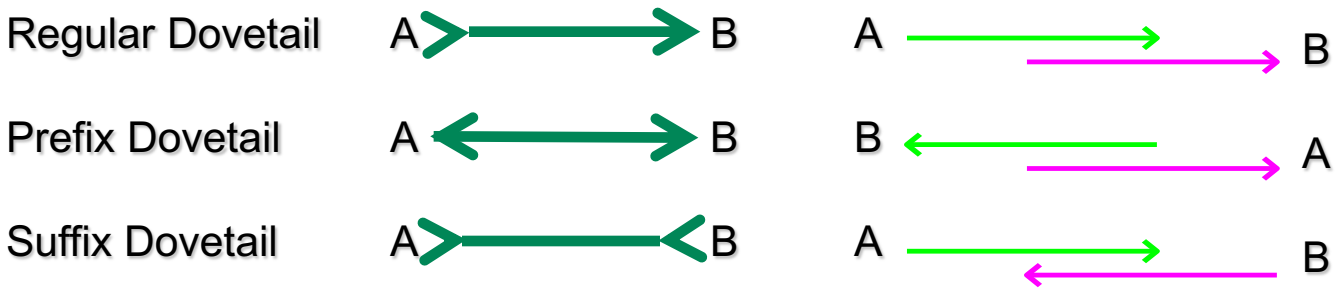
Dynamic programming

But we must frame the problem such that only backtraces involving a suffix of *X* and a prefix of *Y* are allowed

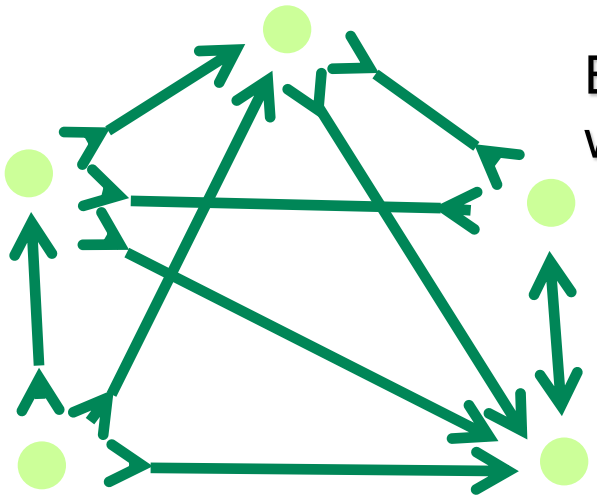
Overlap Graph

Bi-directed Graph Format

Edge Types:



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 r1   :  AGCTAAGCATTTACGATAGCCGATAGCTAAATTAC
Read r2   :  CGTAATTTAGCTATCGGCTATCGTAAATGCTTAGC
Read r3   :  AACGTAATTTAGCTATCGGCTATCGTAAATGCTTA
Read r4   :  GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read r5   :  GTATAACGTAATTTAGCTATCGGCTATCGTAAATG
Read r6   :  ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r7   :  TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read r8   :  ATATAACGTAATTTAGCTATCGGCTATCGTAAATG
Read r9   :  ATTTACGATAGCCGATAGCTAAATTACGTTATATA
Read r10  :  CTATATAACGTAATTTAGCTATCGGCTATCGTAAA
```

Set of given reads

```
Read r1   :  AGCTAAGCATTTACGATAGCCGATAGCTAAATTAC
Read r̄2   :   GCTAAGCATTTACGATAGCCGATAGCTAAATTACG
Read r̄3   :     TAAGCATTTACGATAGCCGATAGCTAAATTACGTT
Read r4   :        GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read r̄5   :         CATTTACGATAGCCGATAGCTAAATTACGTTATAC
Read r6   :          ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r7   :           TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read r̄8   :         CATTTACGATAGCCGATAGCTAAATTACGTTATAT
Read r9   :          ATTTACGATAGCCGATAGCTAAATTACGTTATATA
Read r̄10  :           TTTACGATAGCCGATAGCTAAATTACGTTATATAG
```

Overlapping reads

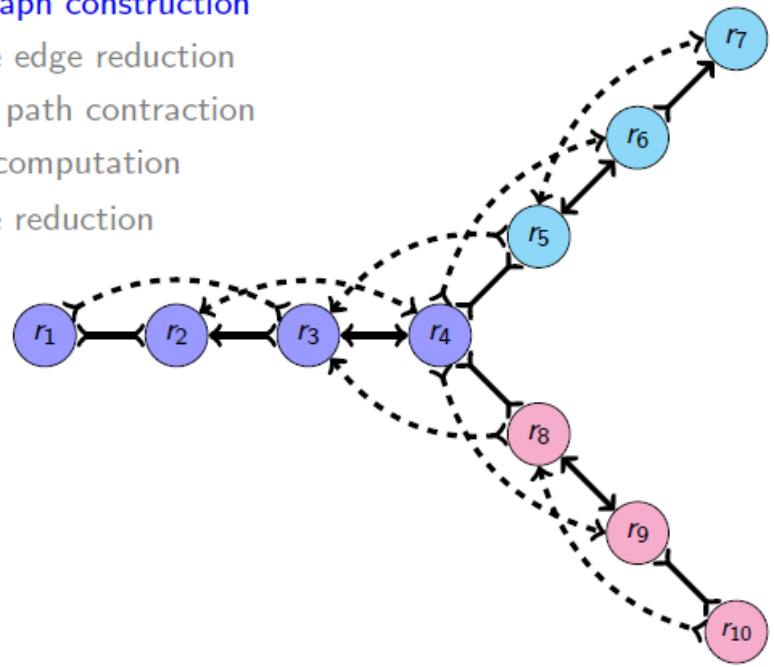
Overlap graph construction

Transitive edge reduction

Composite path contraction

Flow computation

Tree reduction



Layout

```
Read r1   :  AGCTAAGCATTTACGATAGCCGATAGCTAAATTAC
Read r2   :  CGTAATTTAGCTATCGGCTATCGTAAATGCTTAGC
Read r3   :  AACGTAATTTAGCTATCGGCTATCGTAAATGCTTA
Read r4   :  GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read r5   :  GTATAACGTAATTTAGCTATCGGCTATCGTAAATG
Read r6   :  ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r7   :  TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read r8   :  ATATAACGTAATTTAGCTATCGGCTATCGTAAATG
Read r9   :  ATTTACGATAGCCGATAGCTAAATTACGTTATATA
Read r10  :  CTATATAACGTAATTTAGCTATCGGCTATCGTAAA
```

Set of given reads

```
Read r1   :  AGCTAAGCATTTACGATAGCCGATAGCTAAATTAC
Read r̄2   :   GCTAAGCATTTACGATAGCCGATAGCTAAATTACG
Read r̄3   :     TAAGCATTTACGATAGCCGATAGCTAAATTACGTT
Read r4   :       GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read r̄5   :        CATTTACGATAGCCGATAGCTAAATTACGTTATAC
Read r6   :         ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r7   :          TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read r̄8   :        CATTTACGATAGCCGATAGCTAAATTACGTTATAT
Read r9   :         ATTTACGATAGCCGATAGCTAAATTACGTTATATA
Read r̄10  :          TTTACGATAGCCGATAGCTAAATTACGTTATATAG
```

Overlapping reads

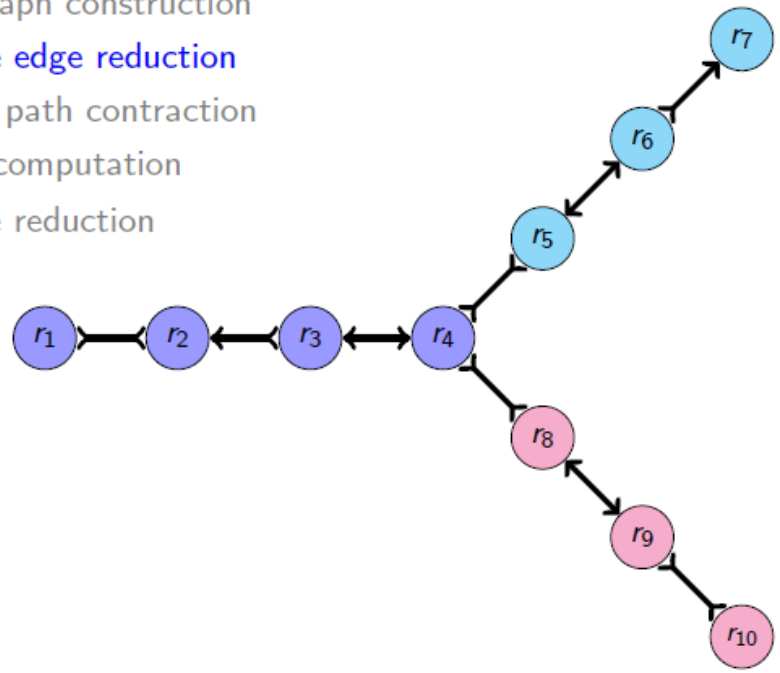
Overlap graph construction

Transitive edge reduction

Composite path contraction

Flow computation

Tree reduction



Layout

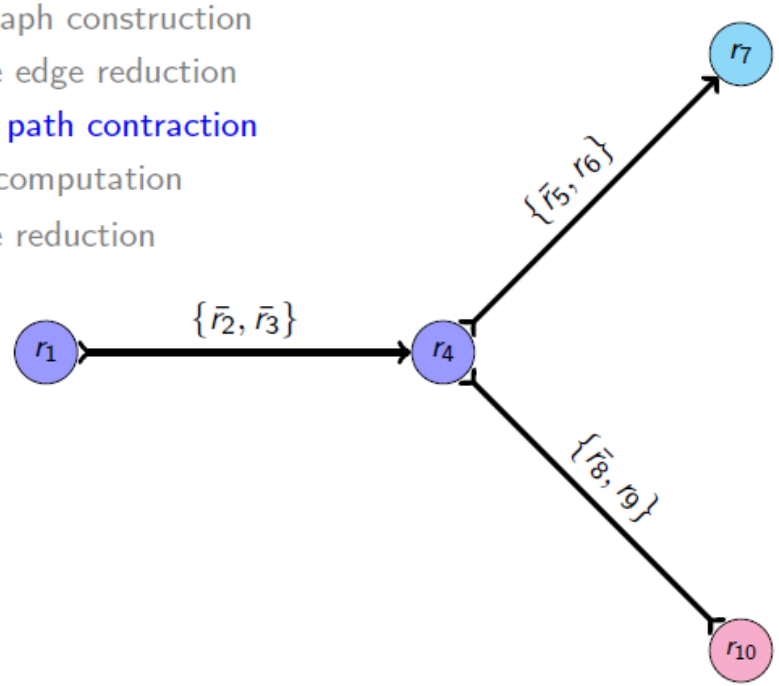
Read r_1 : AGCTAAGCATTTACGATAGCCGATAGCTAAATTAC
Read r_2 : CGTAATTTAGCTATCGGCTATCGTAAATGCTTAGC
Read r_3 : AACGTAATTTAGCTATCGGCTATCGTAAATGCTTA
Read r_4 : GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read r_5 : GTATAACGTAATTTAGCTATCGGCTATCGTAAATG
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC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

Set of given reads

Read r_1 : AGCTAAGCATTTACGATAGCCGATAGCTAAATTAC
Read $\bar{r}_2$: GCTAAGCATTTACGATAGCCGATAGCTAAATTACG
Read $\bar{r}_3$: TAAGCATTTACGATAGCCGATAGCTAAATTACGTT
Read r_4 : GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read $\bar{r}_5$: CATTTACGATAGCCGATAGCTAAATTACGTTATAC
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read $\bar{r}_8$: CATTTACGATAGCCGATAGCTAAATTACGTTATAT
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

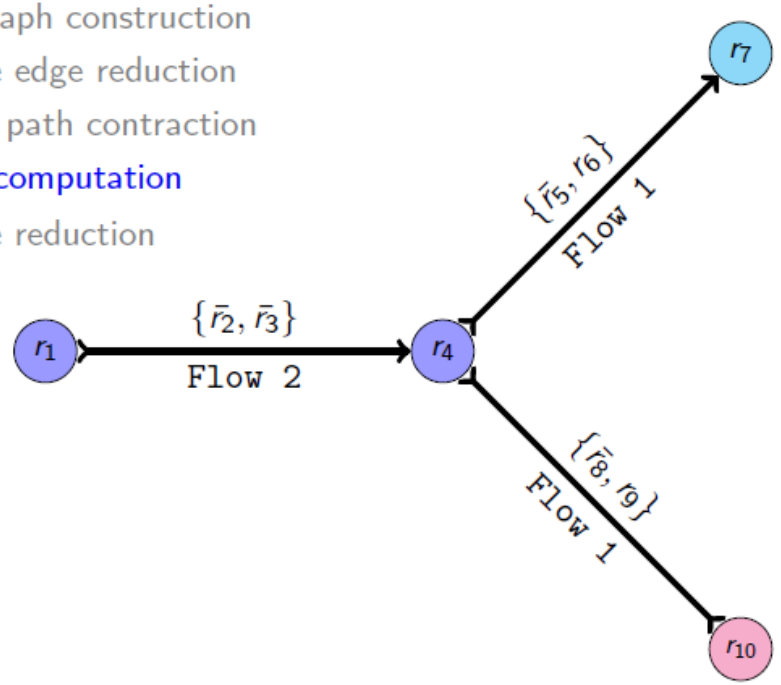
Read r_1 : AGCTAAGCATTTACGATAGCCGATAGCTAAATTAC
Read r_2 : CGTAATTTAGCTATCGGCTATCGTAAATGCTTAGC
Read r_3 : AACGTAATTTAGCTATCGGCTATCGTAAATGCTTA
Read r_4 : GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read r_5 : GTATAACGTAATTTAGCTATCGGCTATCGTAAATG
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC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

Set of given reads

Read r_1 : AGCTAAGCATTTACGATAGCCGATAGCTAAATTAC
Read $\bar{r}_2$: GCTAAGCATTTACGATAGCCGATAGCTAAATTACG
Read $\bar{r}_3$: TAAGCATTTACGATAGCCGATAGCTAAATTACGTT
Read r_4 : GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read $\bar{r}_5$: CATTTACGATAGCCGATAGCTAAATTACGTTATAC
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read $\bar{r}_8$: CATTTACGATAGCCGATAGCTAAATTACGTTATAT
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

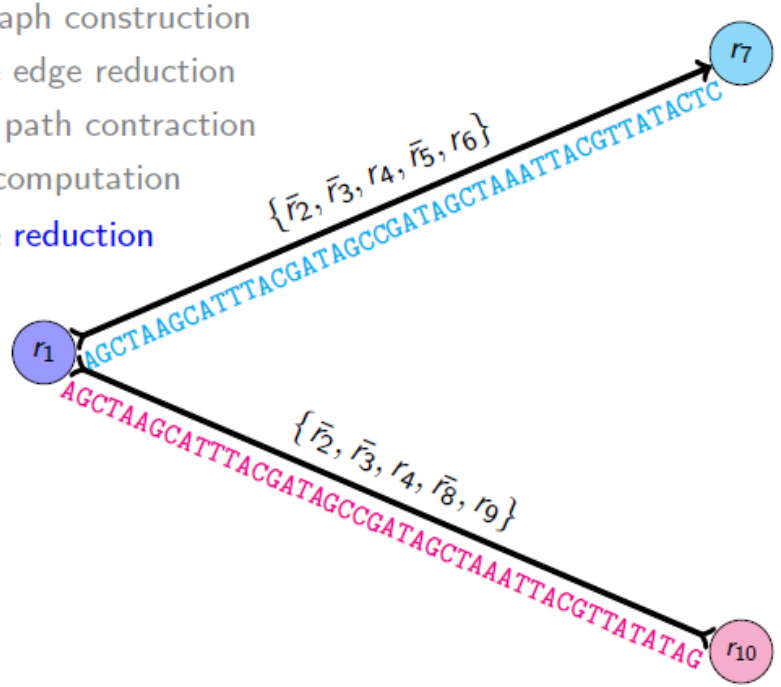
Read r_1 : AGCTAAGCATTTACGATAGCCGATAGCTAAATTAC
Read r_2 : CGTAATTTAGCTATCGGCTATCGTAAATGCTTAGC
Read r_3 : AACGTAATTTAGCTATCGGCTATCGTAAATGCTTA
Read r_4 : GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read r_5 : GTATAACGTAATTTAGCTATCGGCTATCGTAAATG
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC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

Set of given reads

Read r_1 : AGCTAAGCATTTACGATAGCCGATAGCTAAATTAC
Read $\bar{r}_2$: GCTAAGCATTTACGATAGCCGATAGCTAAATTACG
Read $\bar{r}_3$: TAAGCATTTACGATAGCCGATAGCTAAATTACGTT
Read r_4 : GCATTTACGATAGCCGATAGCTAAATTACGTTATA
Read $\bar{r}_5$: CATTTACGATAGCCGATAGCTAAATTACGTTATAC
Read r_6 : ATTTACGATAGCCGATAGCTAAATTACGTTATACT
Read r_7 : TTTACGATAGCCGATAGCTAAATTACGTTATACTC
Read $\bar{r}_8$: CATTTACGATAGCCGATAGCTAAATTACGTTATAT
Read r_9 : ATTTACGATAGCCGATAGCTAAATTACGTTATATA
Read $\bar{r}_{10}$: TTTACGATAGCCGATAGCTAAATTACGTTATATAG

Overlapping reads

Overlap graph construction
Transitive edge reduction
Composite path contraction
Flow computation
Tree reduction



Contig 1:
AGCTAAGCATTTACGATAGCCGATAGCTAAATTACGTTATACTC
Contig 2:
AGCTAAGCATTTACGATAGCCGATAGCTAAATTACGTTATATAG

Contig is a set of overlapping DNA segments.

Consensus

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

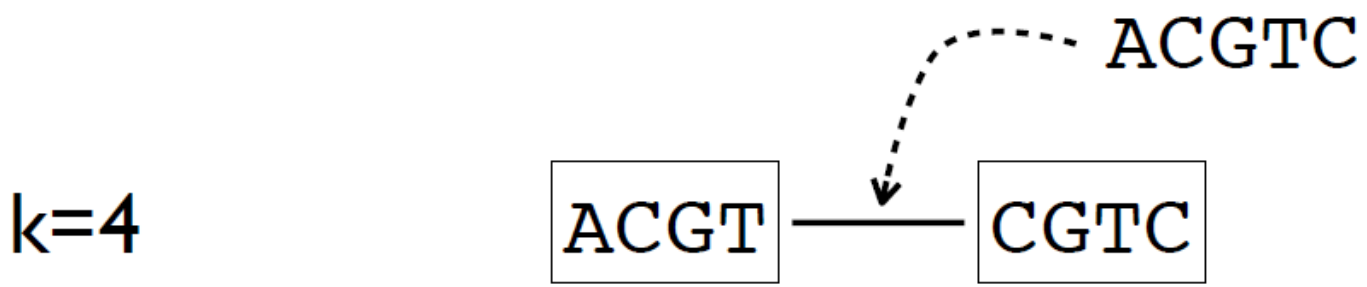
- Celera Assembler with the Best Overlap Graph (CABOG)
  - Designed for Sanger sequences, but works with 454 and error-corrected PacBio reads
- Newbler, a.k.a. GS *de novo* Assembler
  - Designed for 454 sequences, but works with Sanger reads
- Omega, overlap-graph *de novo* assembler for metagenomics

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

- de Bruijn graph
  - k-dimensional graph over four symbols {A, C, G, T}
  - vertex: k-mer -- a string of k nucleotides
  - edge: (k+1)-mer

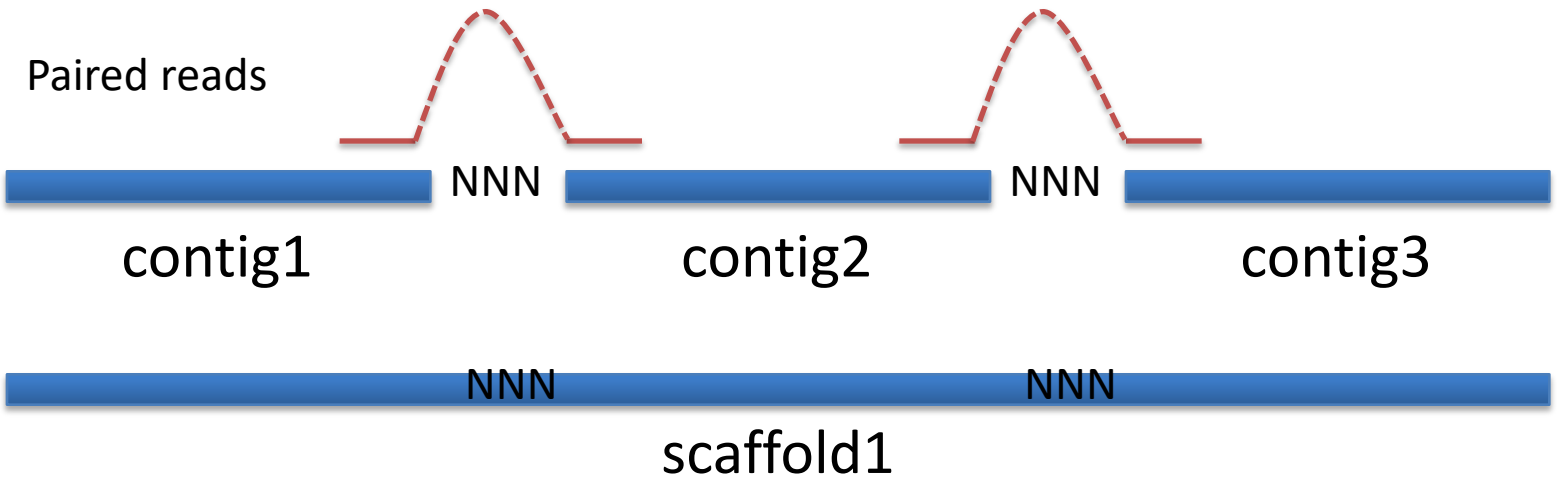


Examples of DBG-based assemblers

- **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)

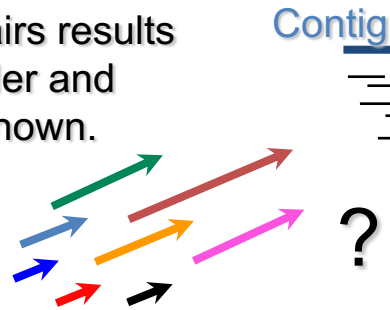
Contig vs Scaffold

- 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

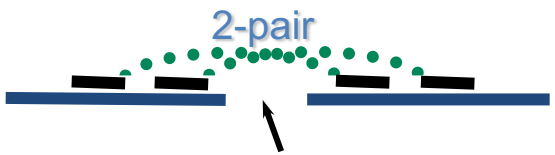


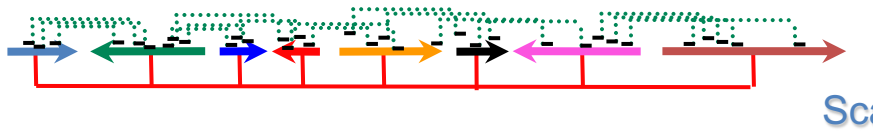
Pairs give order and orientation

Assembly without pairs results in contigs whose order and orientation are not known.

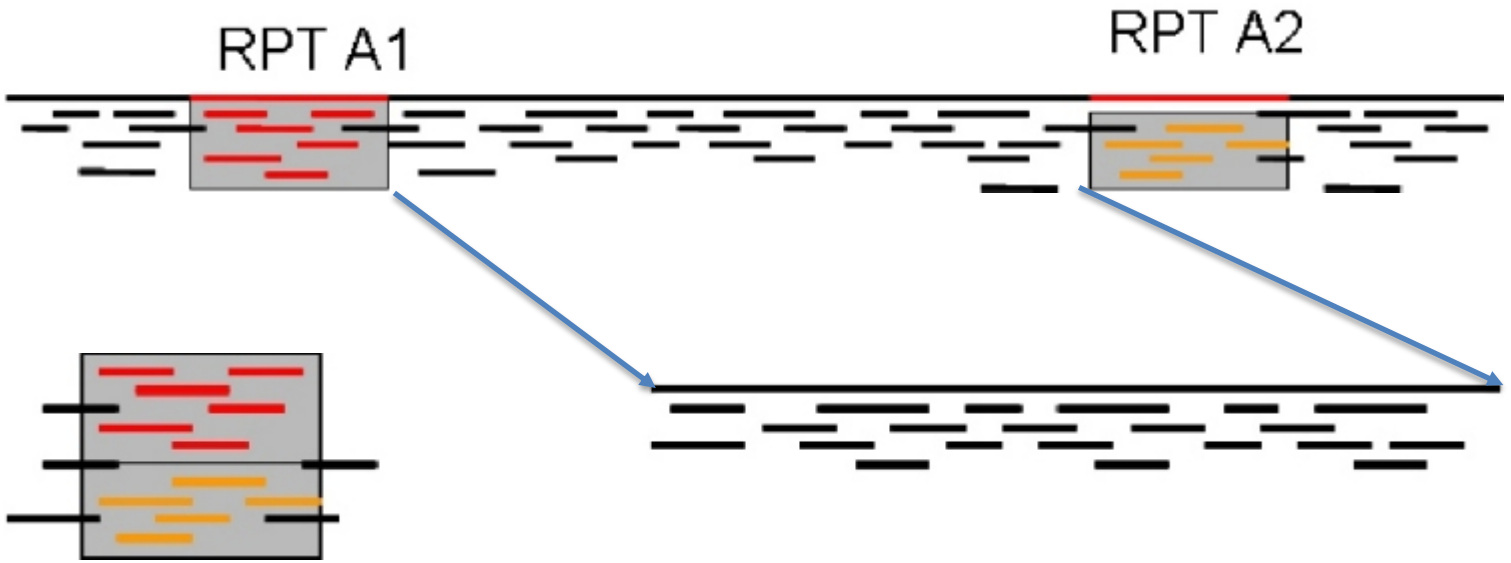


Pairs, especially groups of corroborating ones, link the contigs into scaffolds where the size of gaps is well characterized.





Repeats often split genome into contigs



Reads from multiple repeats collapse into artifactual contig

Contig derived from unique sequences

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?

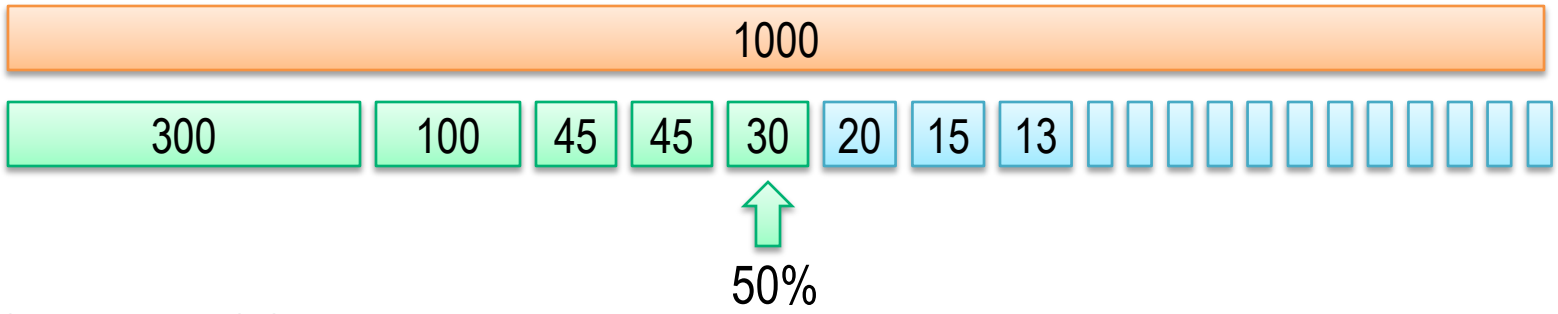
Quality of Assembly? What is N50?

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 >= 500kbp)

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

Learning Outcome: Yes?

- Learn basic knowledge about *de novo* genome assembly and required techniques for it.