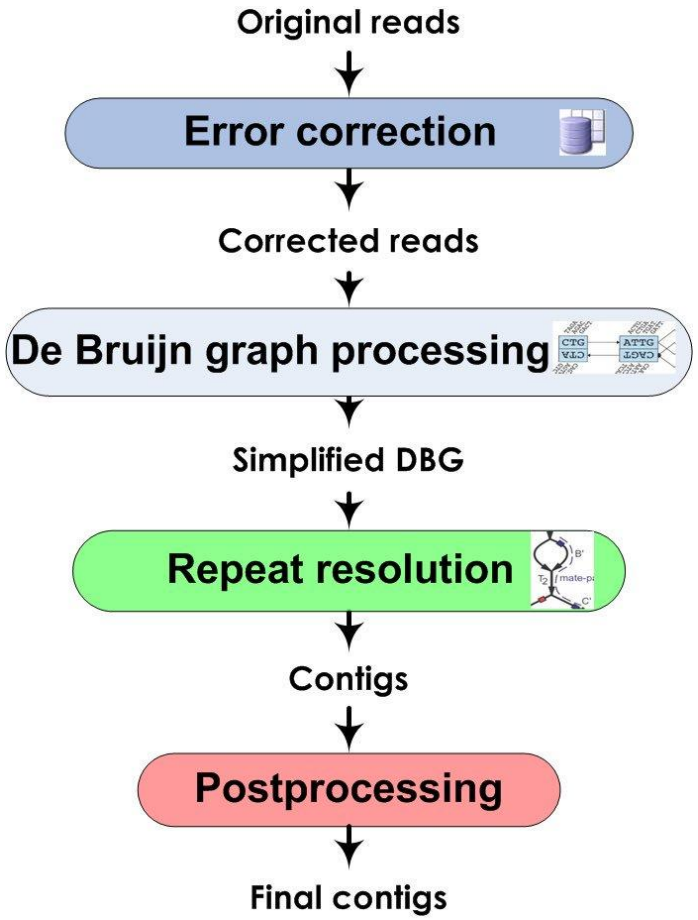
Assembly programs - how do they work

karin.lagesen@vetinst.no

Short read assembly - overview

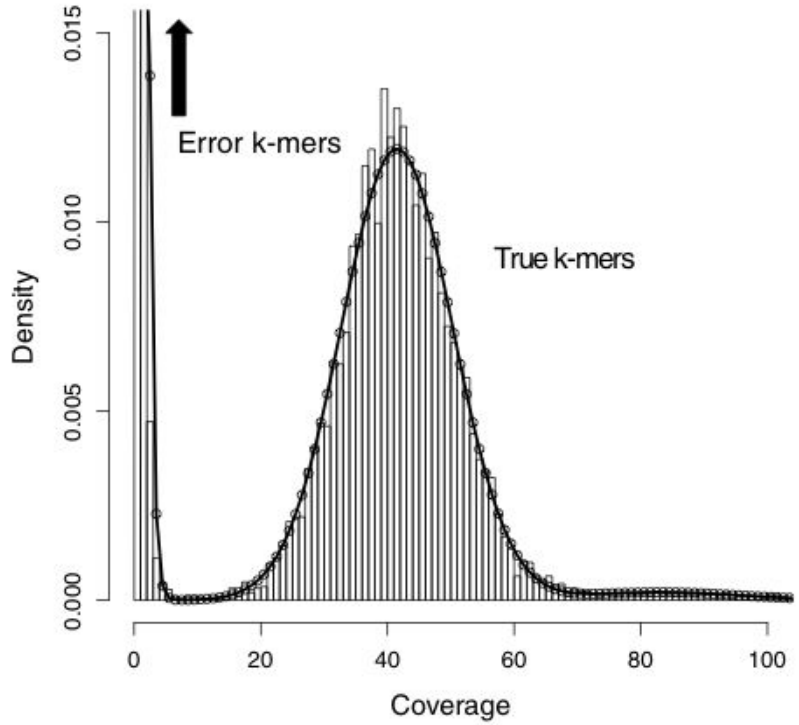


Velvet vs SPAdes

- SPAdes developed to be able to assemble single-cell sequence data
- Single-cell data:
  - Not uniform coverage
- Three main differences between Velvet and SPAdes
  - Error correction
  - Graph construction
  - Graph simplification/resolution
- Other differences too, but won’t go into that here

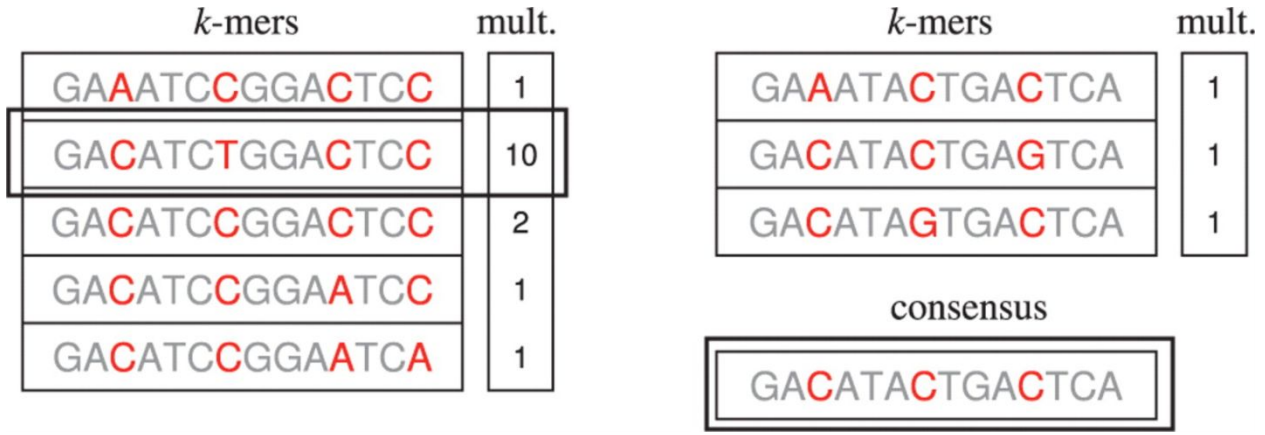
Velvet error correction

- Velvet: expects uniform coverage
- Uses high coverage k-mers to error correct low coverage k-mers



SPAdes error correcton - Hammer

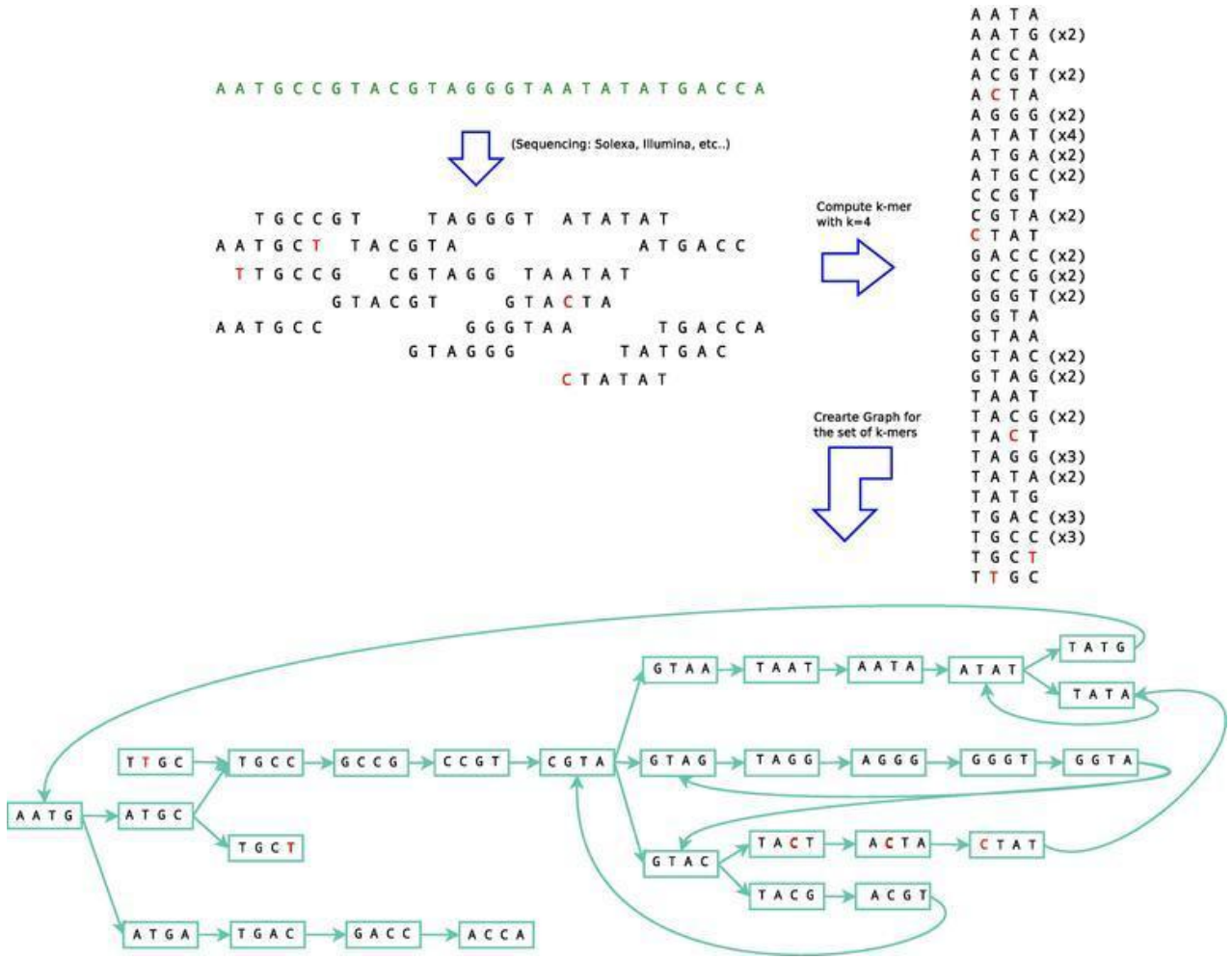
- BayesHammer (Illumina) or IonHammer (IonTorrent)
- How Hammer works:



BayesHammer does the same as Hammer, but looks at the problem probabilistically

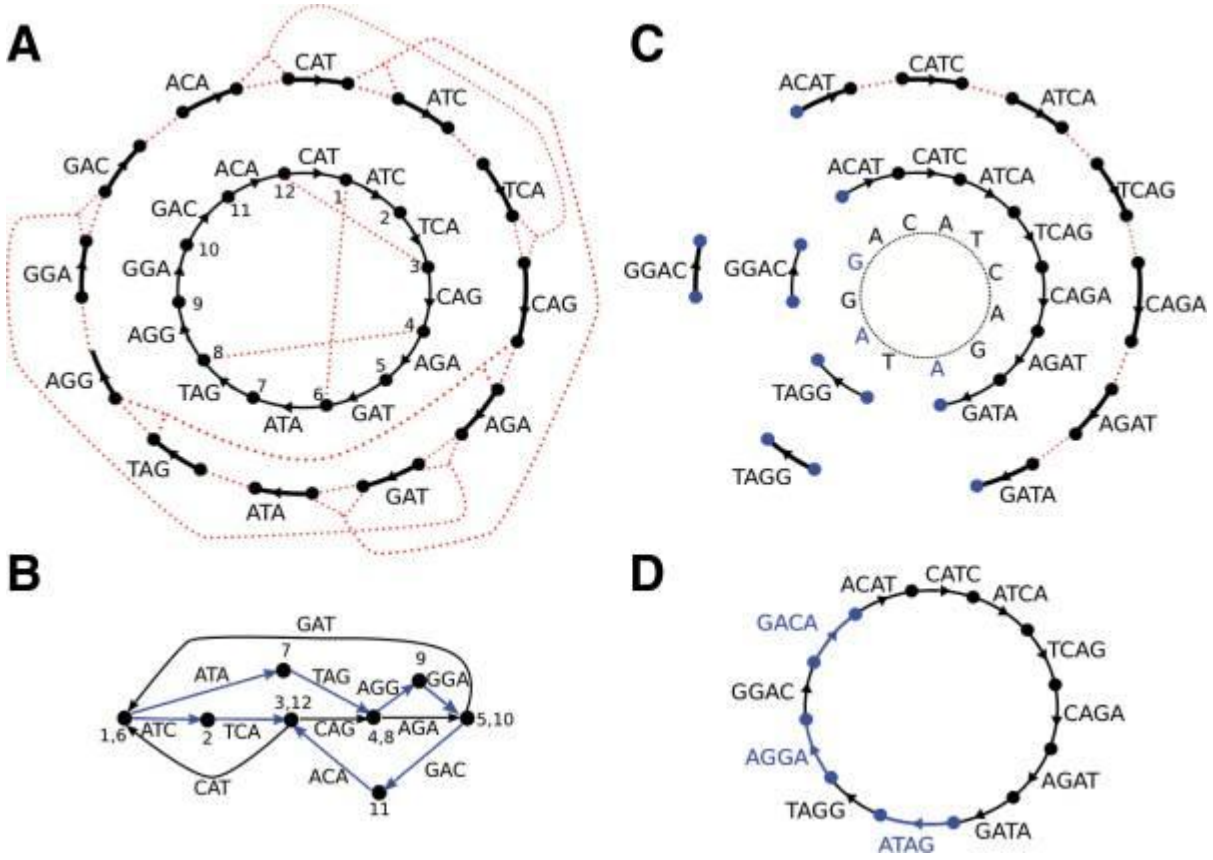
On the left is an example of a typical cluster with good coverage. There are five *k*-mers clustered together, with five loci having mis-alignments. We compute the consensus string (taking multiplicities into account), which we find is already in the cluster (boxed in). All the *k*-mers are then corrected to the consenus. On the right is an example of a common cluster in low coverage regions. The generating *k*-mer was sequenced three times but each time with a single error. There are three *k*-mers in the cluster, but the consensus (boxed in) has not been sequenced and therefore is not in the cluster. Nevertheless, we correct all the *k*-mers to the consensus, allowing Hammer to reconstruct new *k*-mers that are not present in the original data.
Medvedev *et. al.*, Bioinformatics. 2011

Velvet graph construction – fixed size *k-mer*



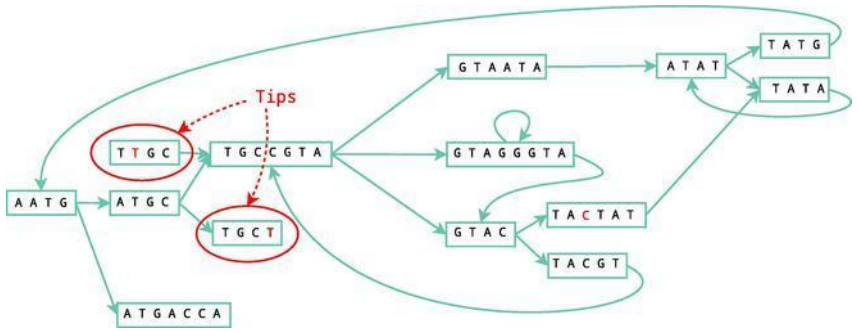
SPAdes: multisized graphs

- Uses several different *k-mer* sizes

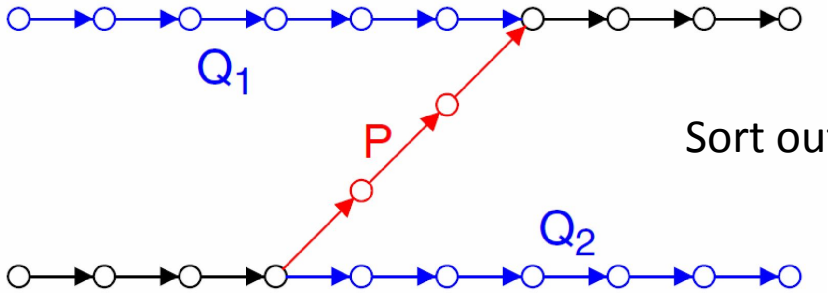


Standard and multisized de Bruijn graph. A circular GENOME CATCAGATAGGA is covered by a set READS consisting of nine 4-mers, {ACAT, CATC, ATCA, TCAG, CAGA, AGAT, GATA, TAGG, GGAC}. Three out of 12 possible 4-mers from GENOME are missing from READS (namely {ATAG,AGGA,GACA}), but all 3-mers from GENOME are present in READS. (**A**) The outside circle shows a separate black edge for each 3-mer from READS. Dotted red lines indicate vertices that will be glued. The inner circle shows the result of applying some of the glues. (**B**) The graph DB(READS, 3) resulting from all the glues is tangled. The three h-paths of length 2 in this graph (shown in blue) correspond to h-reads ATAG, AGGA, and GACA. Thus $READS_{3,4}$ contains all 4-mers from GENOME. (**C**) The outside circle shows a separate edge for each of the nine 4-mer reads. The next inner circle shows the graph DB(READS, 4), and the innermost circle represents the GENOME. The graph DB(READS, 4) is fragmented into 3 connected components. (**D**) The multisized de Bruijn graph DB(READS, 3, 4).
Bankevitch *et. al.*,.J Comput Biol. 2012

Graph simplification



Trim tips

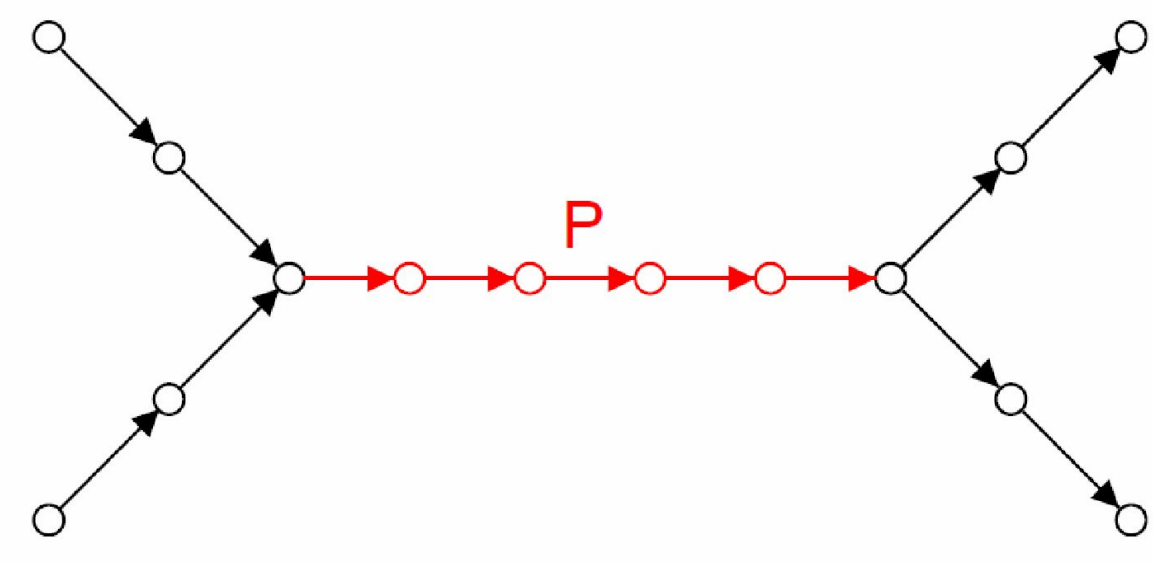


Sort out chimeras

Sort out bulges

- Velvet and SPAdes do these things in similar ways, but: SPAdes needs to keep track of covare in case there is an alternate path in the single cell data

Repeat resolution



How do you sort out which end goes which which end?

Repeat resolution

- Velvet
  - Looks at the reads connecting longer contigs
  - Uses paired read information to “straighten” out the repeats
- SPAdes
  - Uses read pair information
  - Creates a paired de Bruijn graph – each node a pair
  - Much sparser than the “normal” graph

SPAdes options

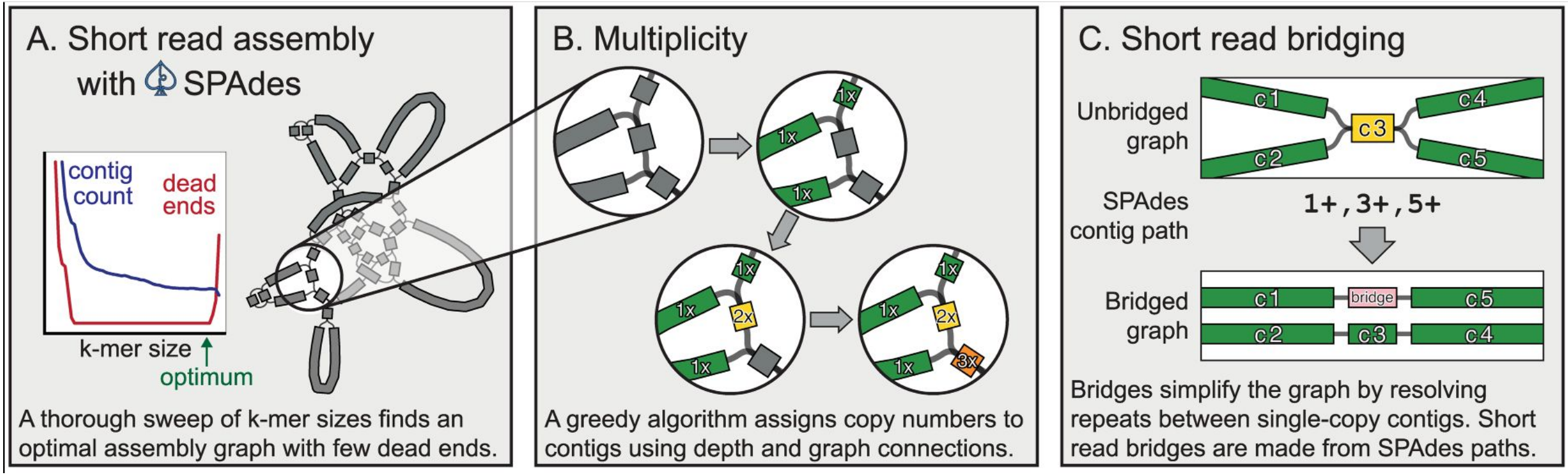
- hybrid spades - short and long read data
- metaSPAdes – a pipeline for metagenomic data sets
- plasmidSPAdes – a pipeline for extracting and assembling plasmids from WGS data sets
- metaplasmidSPAdes – a pipeline for extracting and assembling plasmids from metagenomic data sets
- rnaSPAdes – a de novo transcriptome assembler from RNA-Seq data.
- truSPAdes – a module for TruSeq barcode assembly
- biosyntheticSPAdes – a module for biosynthetic gene cluster assembly with paired-end reads

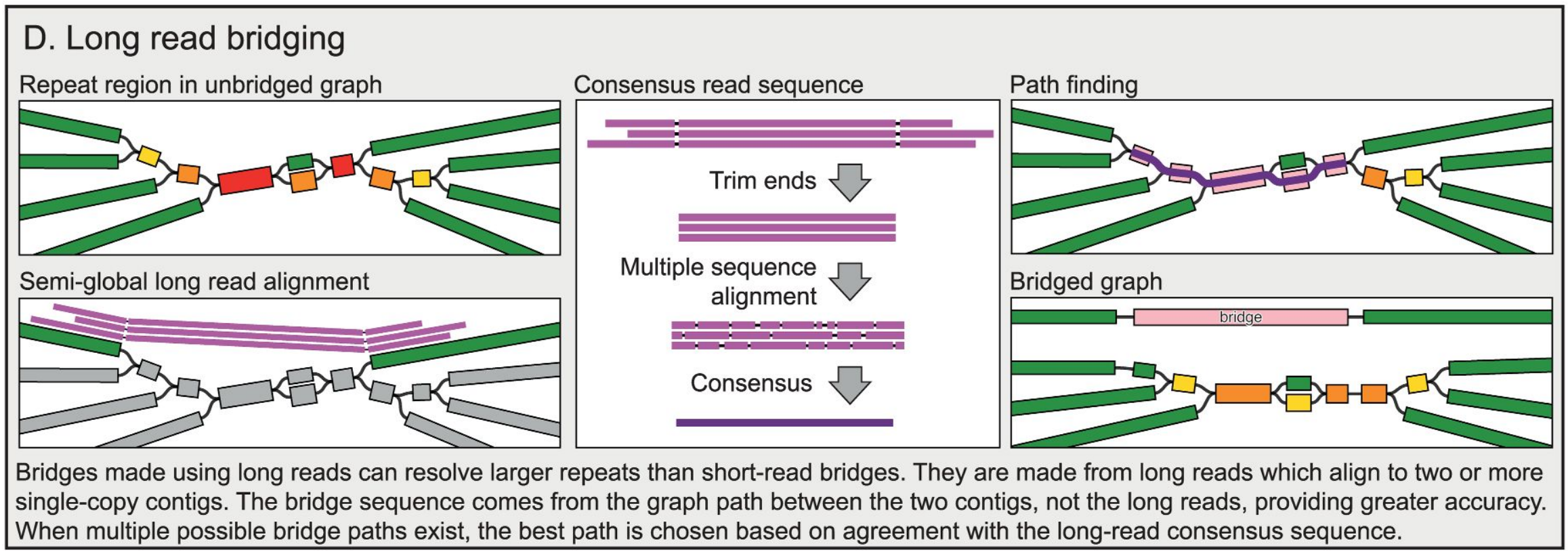
Unicycler - hybrid assembler

- Short read, long read and hybrid assembler
- Short only - SPAdes optimizer
- Long reads - map (miniasm), assemble w. overlap, polish (racon)

- Hybrid:
  - Create SPAdes assembly
  - Scaffold with long reads

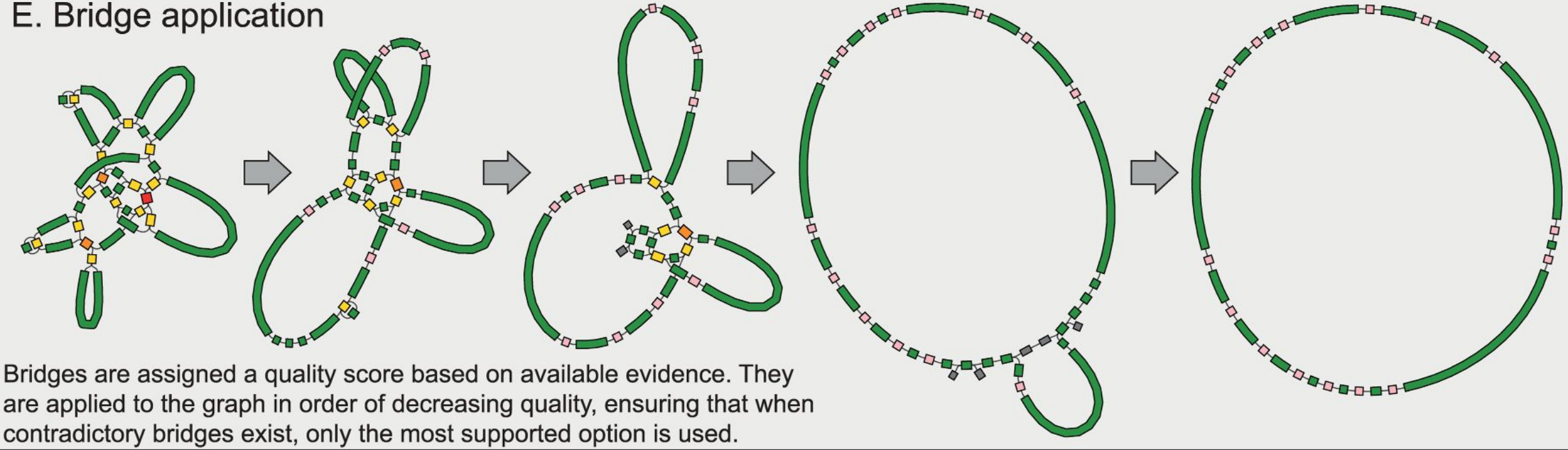
- Note: only for bacterial genomes!


A. Short read assembly with SPAdes
contig count
dead ends
k-mer size
optimum
A thorough sweep of k-mer sizes finds an optimal assembly graph with few dead ends.
B. Multiplicity
1x
2x
3x
A greedy algorithm assigns copy numbers to contigs using depth and graph connections.
C. Short read bridging
Unbridged graph
c1
c2
c3
c4
c5
SPAdes contig path
1+,3+,5+
Bridged graph
bridge
Bridges simplify the graph by resolving repeats between single-copy contigs. Short read bridges are made from SPAdes paths.



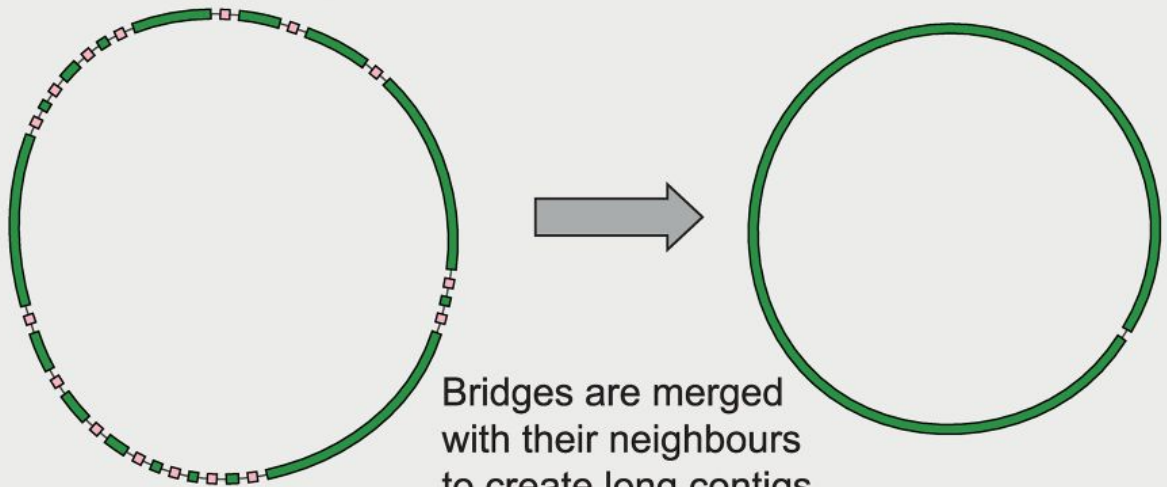
Bridges made using long reads can resolve larger repeats than short-read bridges. They are made from long reads which align to two or more single-copy contigs. The bridge sequence comes from the graph path between the two contigs, not the long reads, providing greater accuracy. When multiple possible bridge paths exist, the best path is chosen based on agreement with the long-read consensus sequence.

E. Bridge application



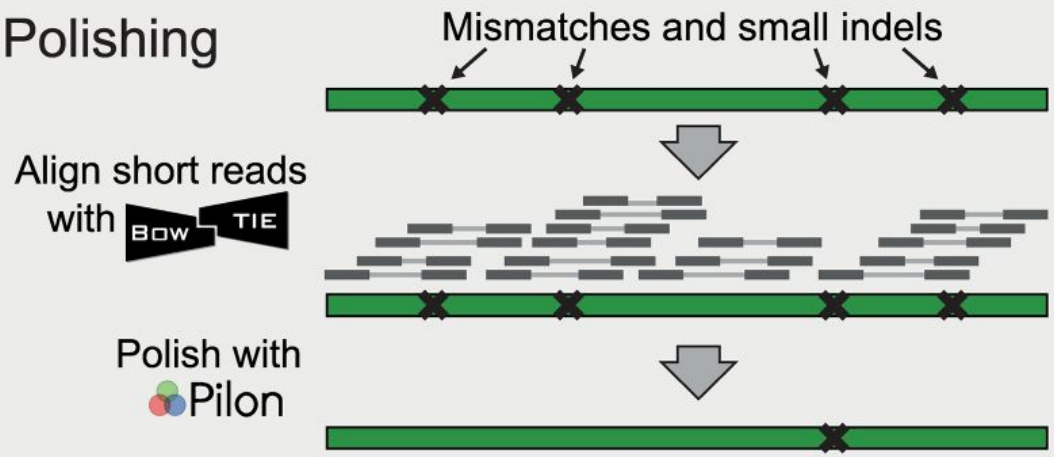
Bridges are assigned a quality score based on available evidence. They are applied to the graph in order of decreasing quality, ensuring that when contradictory bridges exist, only the most supported option is used.

F. Contig merging



Bridges are merged
with their neighbours
to create long contigs.

G. Polishing



The final assembly is polished using the accurate short reads to
reduce the rate of mismatches and small insertions/deletions.

Flye - long read assembly

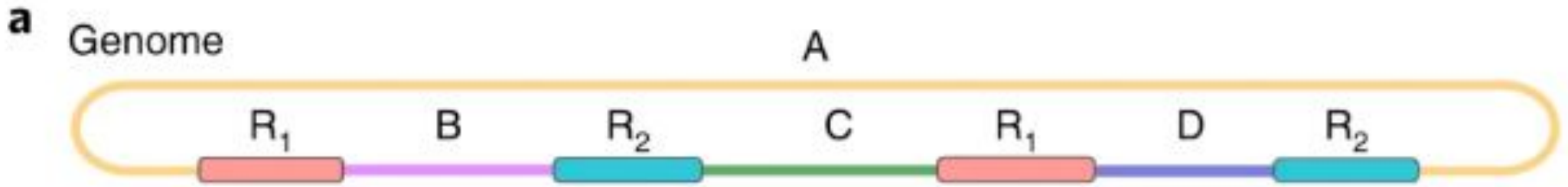
- Only works on long reads
- Follows the assemble-then-correct approach
- Quite fast, and especially good for resolving repeats

- Alternative: canu
  - Correct-assemble
  - Thorough, but takes a long time

Flye procedure

- Find k-mers in reads, find reads with shared k-mers
- Find reads that have overlaps
- Assemble contigs from overlaps - called disjointigs
- Reconstruct graph
- Resolve repeats
- Polish
- Output assembly

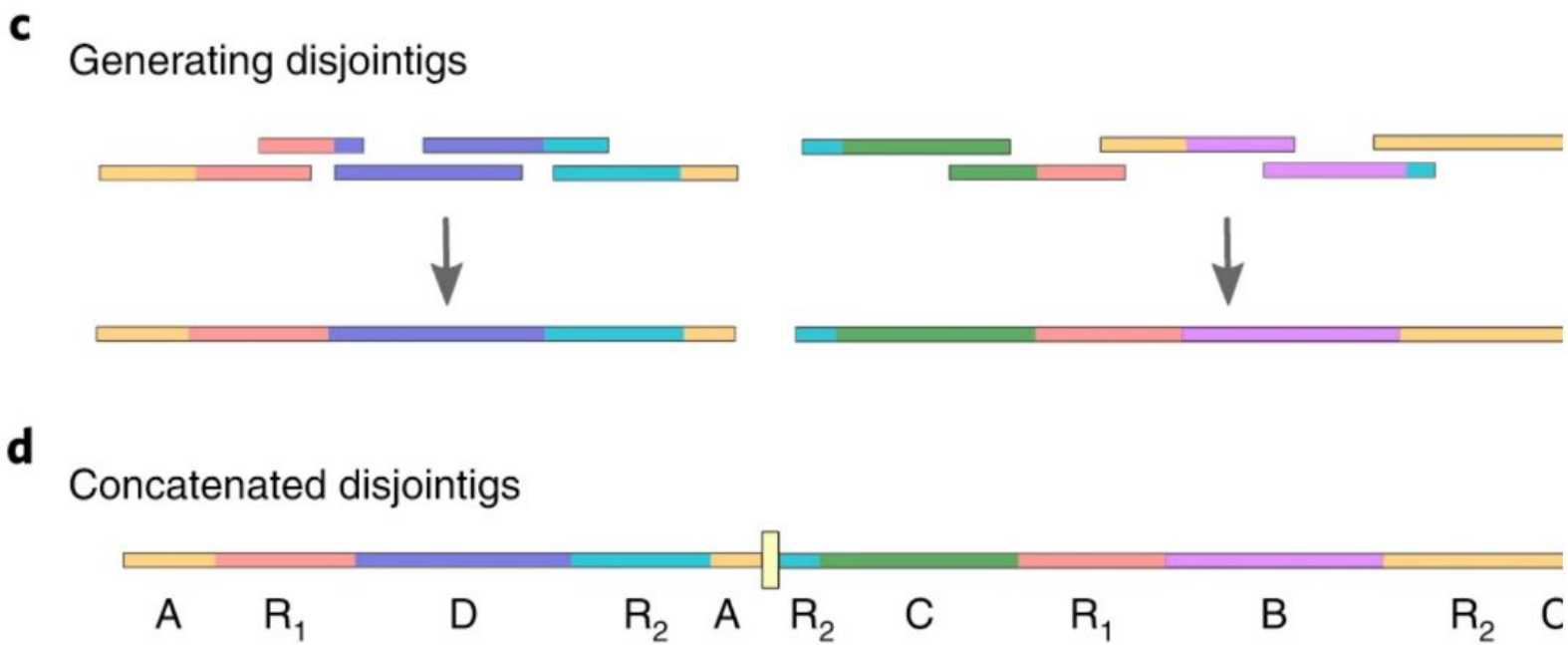
Genome



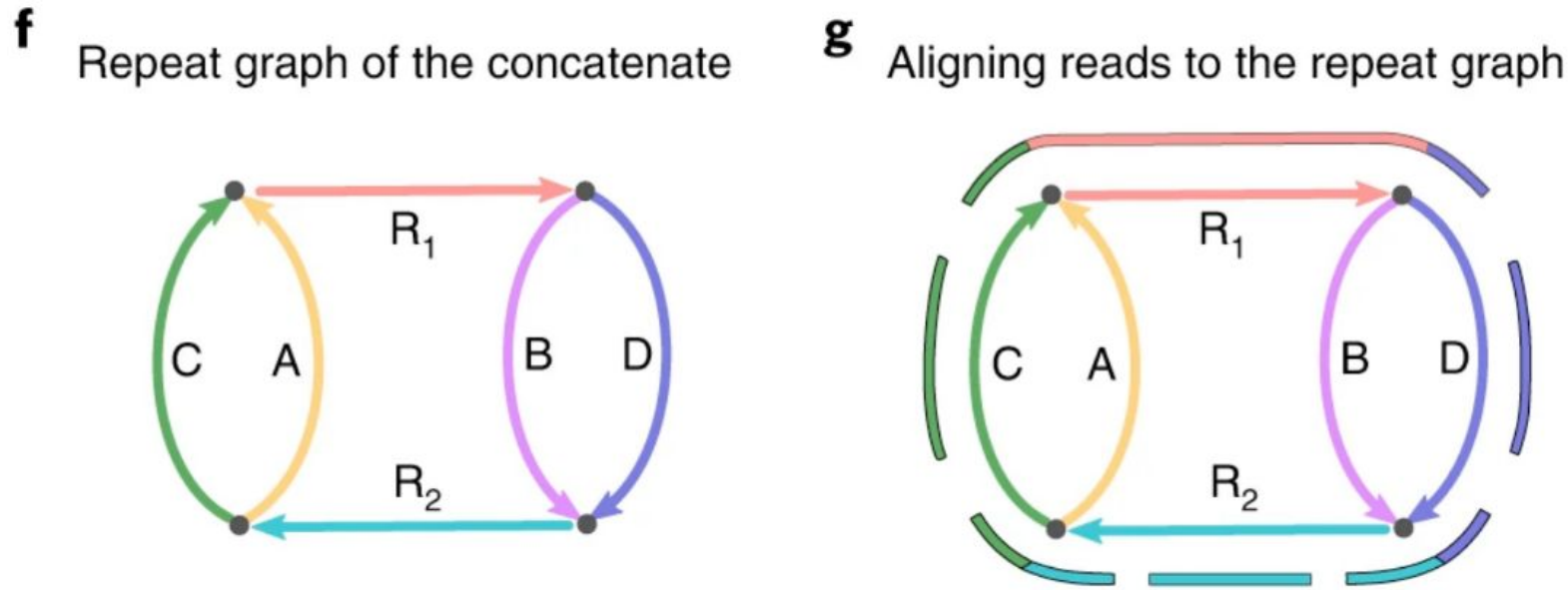
Disjointigs

Build disjointigs by taking reads at random and trying to extend left and right using overlaps.

Concatenate all disjointigs, in an arbitrary order, into one string called *Concatenate*.



Align reads to the repeat graph



Resolving *unbridged* repeats in the assembly graph