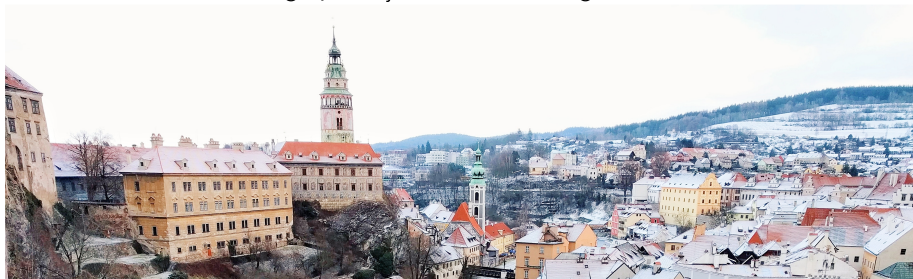


A little tour of assembly methods

Antoine Limasset & Camille Marchet
CRISAL, Université de Lille, CNRS, France
Evomics '22 – Český Krumlov



@Camillemrcht camille.marchet@univ-lille.fr
@BQPMalfoy antoine.limasset@univ-lille.fr



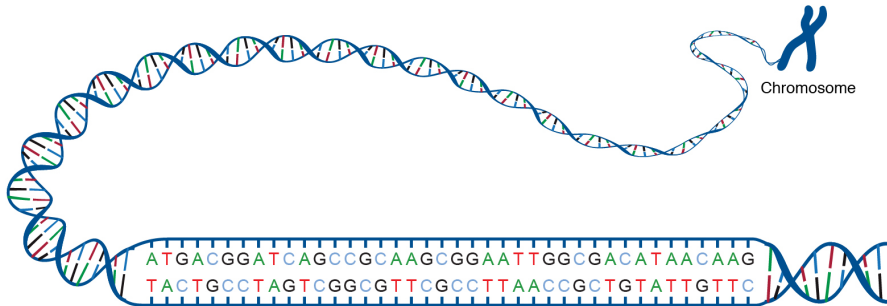
● Content of this course

- How to reconstruct a genome with sequencing data?
 - What are the main challenges?
 - Which solutions have been proposed?
-
- Easter egg Warning: 2 assembler names hidden in the slides



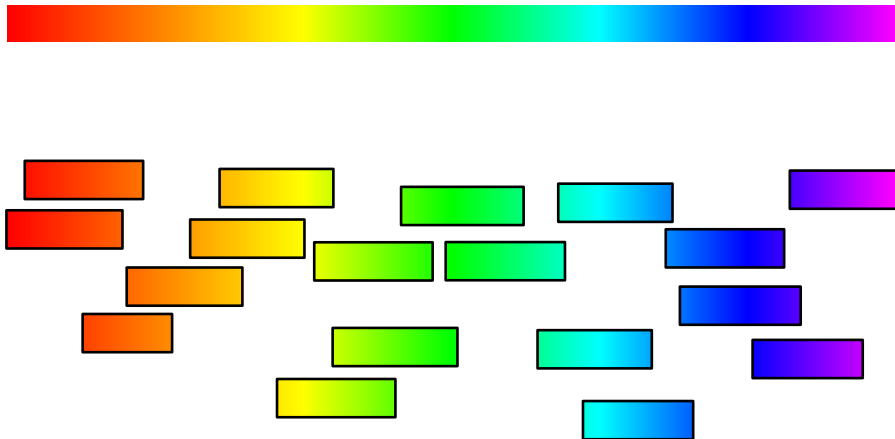
genome size: ~ 32 gigabases

- Accessing a genome

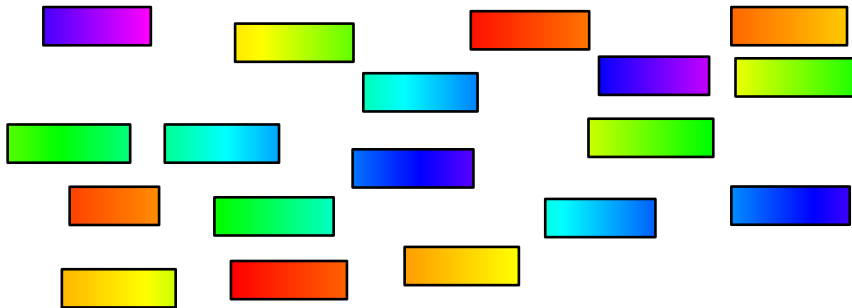


From www.genome.gov/genetics-glossary/acgt

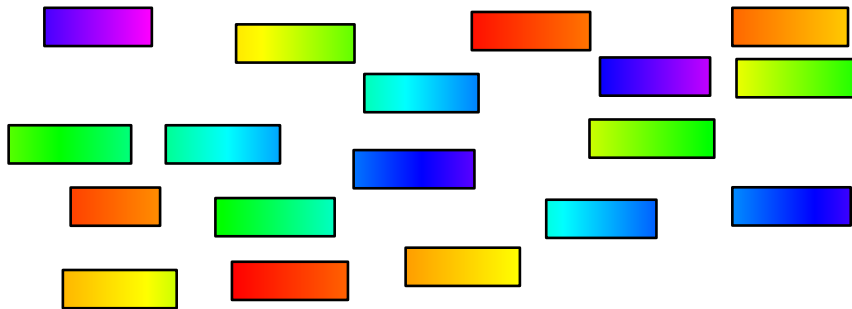
- Reads are words from the genome



- Reads are **shuffled** words from the genome



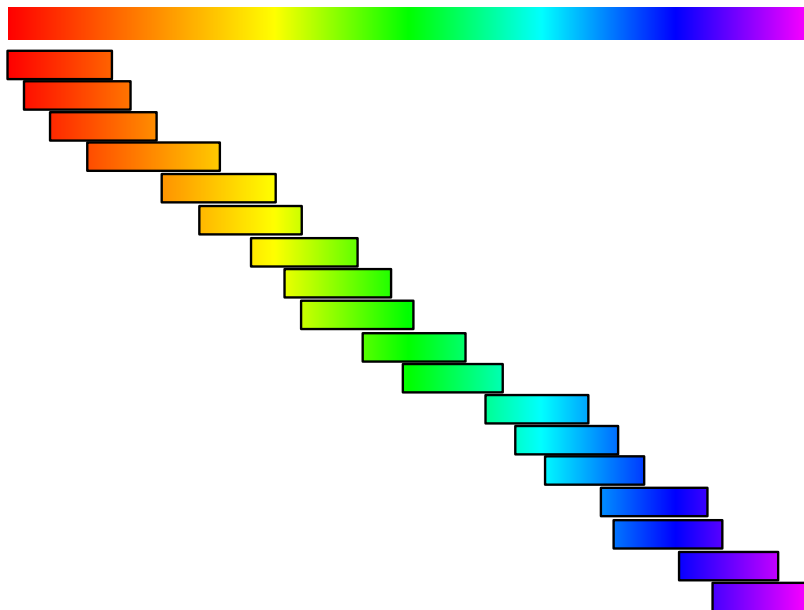
- Genome assembly task



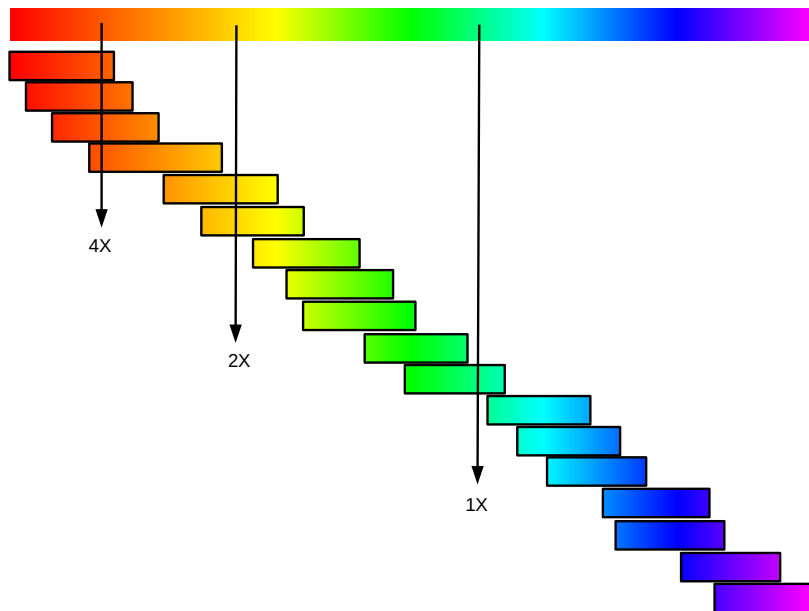
Genome assembly



- Using read overlaps



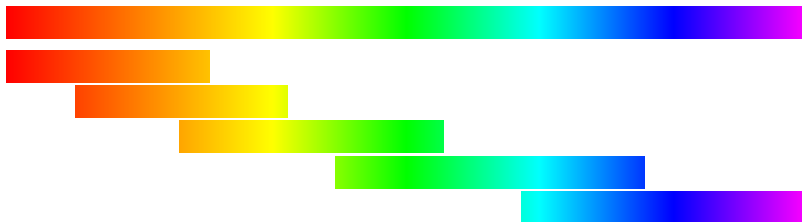
Genome sequencing: coverage



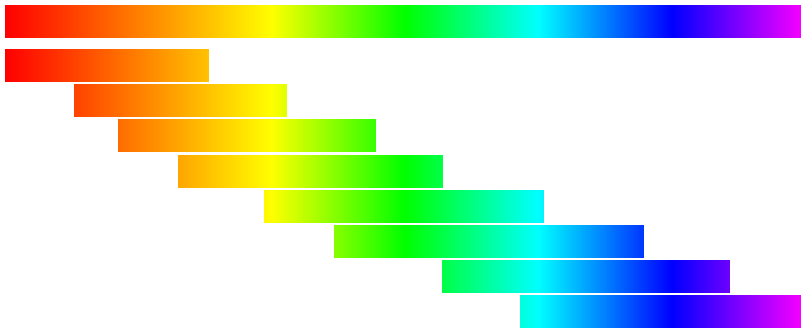
- Genome sequencing: coverage



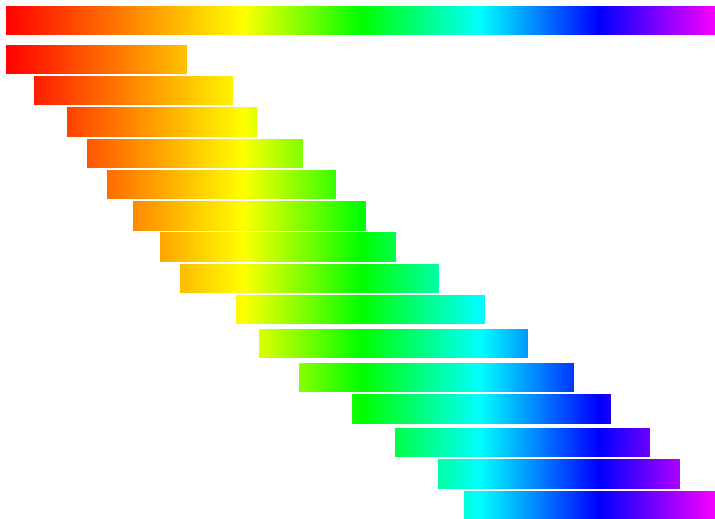
- Genome sequencing: coverage



- Genome sequencing: coverage



- Genome sequencing: coverage



NB: 30-100X are often required for assembly projects

• How to assemble

Reads :

1: ATCGGTATCG
2: GGTATCGTTA
3: ATCGTTACGG
4: GTTACGGTAT
5: ACGGTATACC

1: ATCGGTATCG
2: GGTATCGTTA

Overlap length:7

1: ATCGGTATCG
3: ATCGTTACGG

Overlap length:4

1: ATCGGTATCG
4: GTTACGGTAT

Overlap length:1

1: ATCGGTATCG
5: ACGGTATACC

No Overlap

- Assembly idea number 1: assemble the longest overlaps

Reads :

1: ATCGGTATCG

2: GGTATCGTTA

3: ATCGTTACGG

4: GTTACGGTAT

5: ACGGTATACC

Best overlaps:

1: ATCGGTATCG

2: GGTATCGTTA

3: ATCGTTACGG

4: GTTACGGTAT

5: ACGGTATACC

supplementary
information
brought by
each read



Output "genome" :

ATCGGTATCG + TTA + ATCGTTACGG + TAT + ACC

- Your time to shine!

Let assemble this genome!

Your read set:

1: A T T T A C G G G T

2: T T A C G G G T G G

3: A C G G G T C C T T

4: G T C C T T T C T T

5: T T T C T T A C G G

For each read:

Find the best overlap (length>5)

Merge the two reads

- The Greedy solution

The best overlaps:

ATTTACGGGT

TTACGGGTGG

ACGGGTCCTT

GTCCTTTCTT

TTTCTTACGG

Output “genome”

ATTTACGGGTGG

ACGGGTCCTTTCTTACGG

- The actual solution

The actual genome:

ATTTACGGGTCCTTTCTTACGGGTGG

How the reads should be ordered:

ATTTACGGGT
ACGGGTCCTT
6 GTCCTTTCTT
TTTCTTACGG
TTACGGGTGG

~~longest overlap we found~~

~~ATTTACGGGT
TTACGGGTGG~~
8

- What happened?

The actual genome:

ATTTACGGGTCTTTCTTACGGGTGG

How the reads should be ordered:

ATTTACGGGT
ACGGGTCTTT
6 GTCCTTTCTT
TTTCTTACGG
TTACGGGTGG

longest overlap we found

~~ATTTACGGGT
TTACGGGTGG~~
8

ATTTACGGGTGG
not in the genome
ACGGGTCTTTCTTACGG
not in the genome

- Do we expect many repeats?

Probability to have NO repeated word of size 31 in a 5 megabases genome

Input interpretation:

$$\left(\frac{4^{31} - 1}{4^{31}} \right)^{1/2 (5 \times 10^6 (5 \times 10^6 - 1))}$$

Decimal approximation:

0.999997289498784302383172055421363836712023171938932024106...

From en.wikipedia.org/wiki/Birthday_problem

- The burden of assembly: genomic repeats

Amount of repeats larger than a given size in *E. coli* genome

- 15: 44,994
- 21: 1,169
- 31: 559
- 41: 323
- 51: 225
- 61: 192

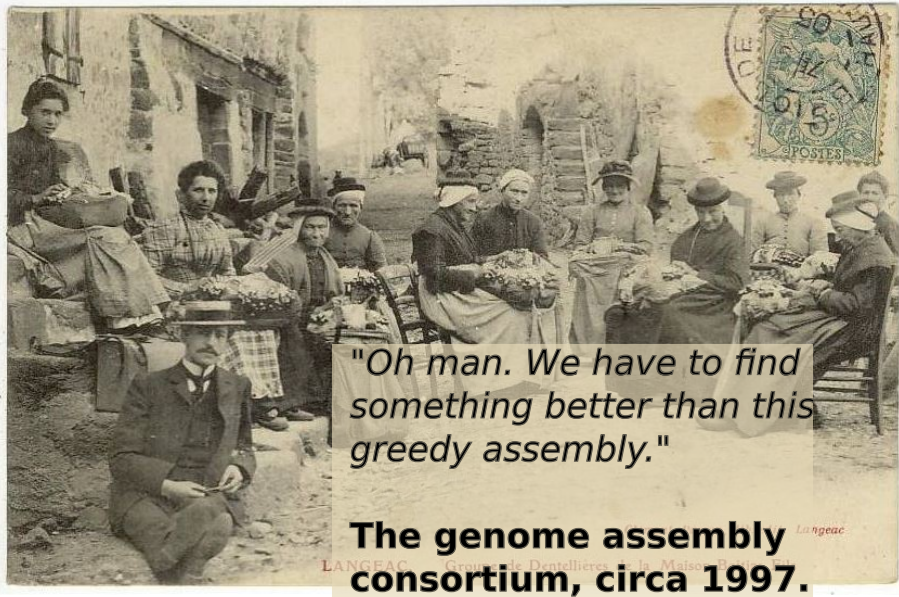
Genomic repeats are NOT random events

• Greedy assemblers

- Simple and efficient scheme
- Rely on **local** best choice (greedy)
- May create errors because of local choices when there are repeats



- History from the last century



"Oh man. We have to find something better than this greedy assembly."

The genome assembly consortium, circa 1997.

- Graph representation

ACGGGTCCTT a node is a sequence

GTCCCTTTCTT

TTTCTTACGG

an arc oriented between
a source node and
a sink node

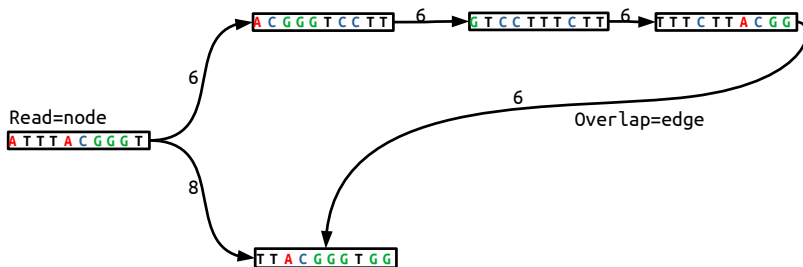
an arc means there is an overlap between
the end of the source node and the beginning
of the sink node

- Assembly **idea number 2**: consider all overlaps

Genome:

A T T T A C G G G T C C T T T C T T A C G G G T G G

Overlap graph:



● Greedy solution

Genome:

A TTTACGGGT CCTTTC TTAACGGGT GG

Overlap graph:



Read=node

A T T T A C G G G T

Overlap=edge

8

T T A C G G G T G G

Greedy assembly output:

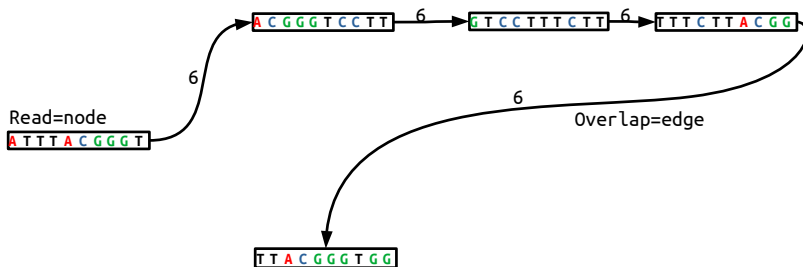
A T T T A C G G G T G G
A C G G G T C C T T T C T T A C G G

• One piece solution

Genome:

A T T T A C G G G T C C T T T C T T A C G G G T G G

Overlap graph:



Overlap graph output:

A T T T A C G G G T C C T T T C T T A C G G G T G G

• Multiple repeats

Reads:

GCTGATTT

ATTTGTAT

GTATTGTC

TGTCAAGT

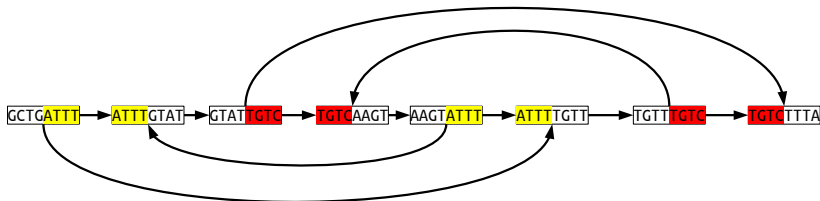
AAGTATTT

ATTTTGTT

TGTTTGTC

TGTCCTTA

Overlap graph:



• First solution

Reads:

GCTGATTT

ATTTGTAT

GTATTGTC

TGTCAAGT

AAGTATTT

ATTTTGTT

TGTTTGTC

TGCTTTA

Overlap graph:



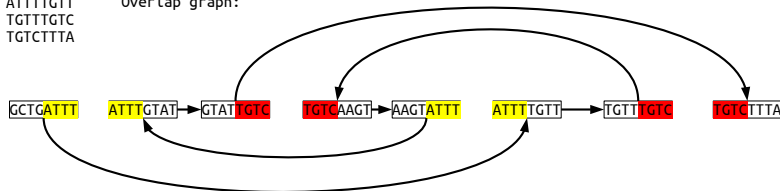
Possible assemblies:

GCTGATTTGTATTGTCAAGTATTTTGTTTGCTTTA

• Second solution

Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCITTA

Overlap graph:



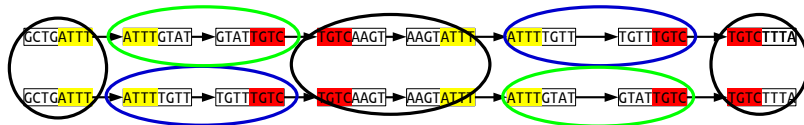
Possible assemblies:

GCTGATTTGTATTGTC AAGTATTTTGTTTGTCITTA
GCTGATTTTGTTTGTC AAGTATTTGTATTGTCITTA

Those two solutions are indistinguishable

● Parsimonious solution: do not assemble

Possible assemblies:

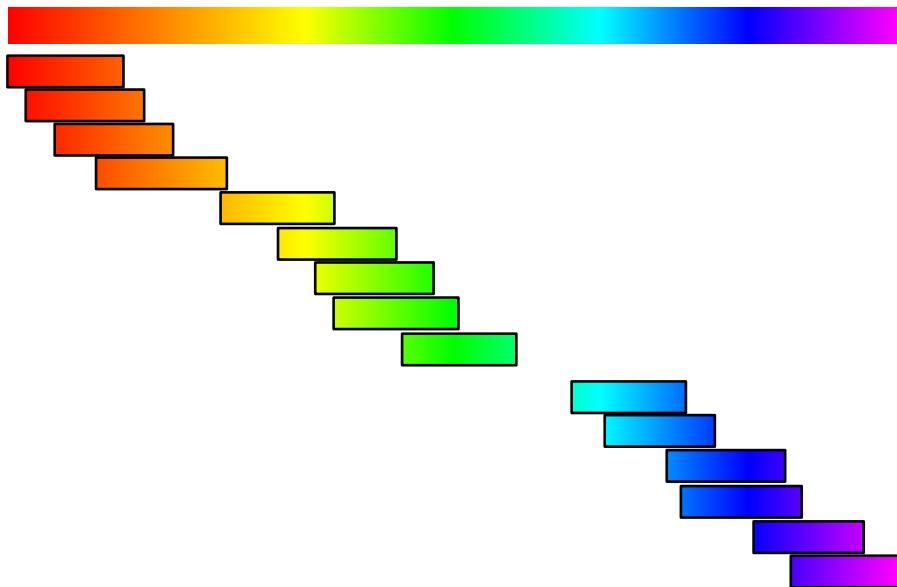


Genome pieces:

GCTGATT ATTTGTATTGC TGTCAAGTATT ATTTTGTTTGC

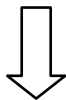
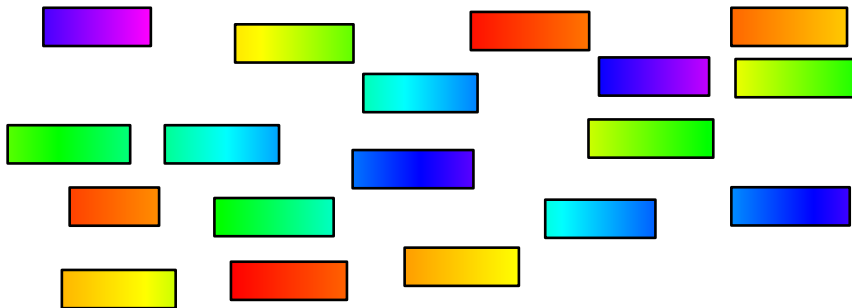
Repeats lead to the fragmentation of the assembly

- Missing information also fragments the assembly



- Assembly **concession number 1: output fragments**

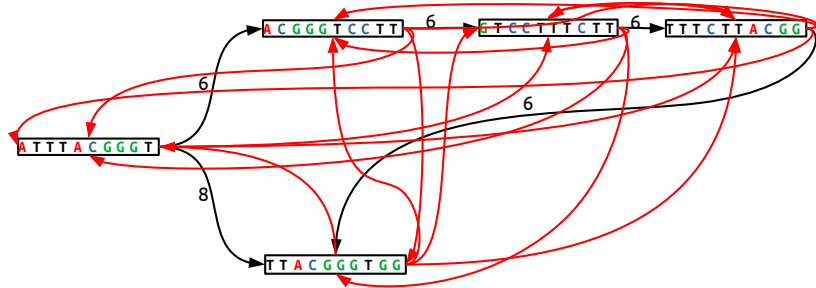
In the real world, assemblers often provide pieces of genomes rather than complete ones



Genome assembly

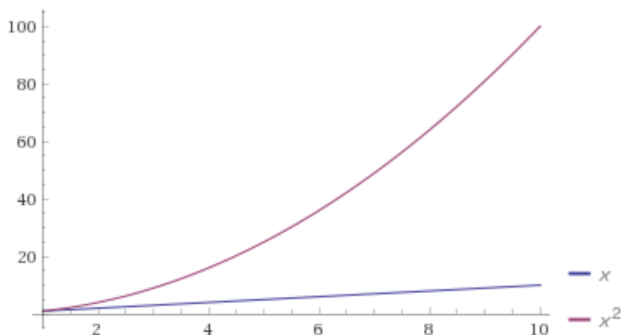


- Overlap graph prerequisite : all overlaps



- Overlap graph burden: number of reads

$n(n-1)/2 = \mathcal{O}(n^2)$ possible overlaps for n reads



Linear: 2X data 2X time

Quadratic: 2X data 4X time

- Overlap graph burden: number of reads

$n(n-1)/2 = \mathcal{O}(n^2)$ possible overlaps for n reads

# Reads	# Overlaps
1000	499,500
10,000	50 million
100,000	5 billion
1 million	500 billion
10 million	50 trillion...

Most overlaps are too small to be considered...

The overlap computation is not linear

Talking about CPU years on large genomes...

● Overlap graphs in a nutshell

- Graphs of overlaps between the reads
- Can provide a global solution for assembly
- Can be difficult in real cases because it requires a lot of computation (overlaps)



S. cerevisiae, *D. melanogaster*, human could be assembled using overlap graphs approaches (Celera (Myers et al. 2000), SGA (Simpson & Durbin 2011), ...)

- Fast forward



Genome Assembly Campus in the 2000's

- Assembly **idea number 3: Focus on genome words**

Let's select a word from the genome:

AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC

in the genome, after TAGCAT

only AGCATG appears

- Genome words / read words

In real cases we don't have the genome

AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC

in the genome, after TAGCAT

only AGCATG appears

but we have the reads

ATGACCGTAGCATGCT
ATGACCGTAGCATGCT
GACCGTAGCATGCTAA

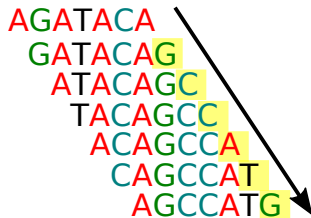
in the reads, after TAGCAT

only AGCATG appears

- Reconstitute larger genomic words

extract all k -mers ($k = 7$):

AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC



AGATACA + G + C + C + A + T + G

AGATACAGCCATG a sequence from the genome

- The de Bruijn graph

Read

AGATACAGCCA

De Bruijn graph

Kmer=node



k-1 overlap=edge

AGATACA + G + C + C + A
=AGATACAGCCA

- de Bruijn graph assembly

Overlapping reads

AGATACAGCCA
TACAGCCATGG

De Bruijn graph



Resulting sequence

AGATACAGCCATGG

- de Bruijn graph time!

Reads

GCCATGGGTTT
TACAGCCATGG
AGCCATGGGTT
GCCATGGGTTT
AGATACAGCCA
ACAGCCATGGG
GATACAGCCATG
CATGGGTTTAA
ACAGCCATGGG
GATACAGCCATG
CATGGGTTTAA
CAGCCATGGGT

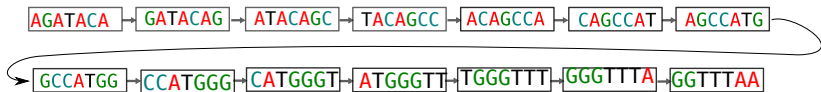
Hint: Use 7-
mers

• Solution

Overlapping reads

```
AGATACAGCCA
GATACAGCCATG
GATACAGCCATG
TACAGCCATGG
ACAGCCATGGG
ACAGCCATGGG
CAGCCATGGGT
AGCCATGGGTT
GCCATGGGTTT
GCCATGGGTTT
CATGGGTTTAA
CATGGGTTTAA
```

De Bruijn graph



Resulting sequence

AGATACAGCCATGGGTTTAA

• de Bruijn graph versus overlap graph

reads



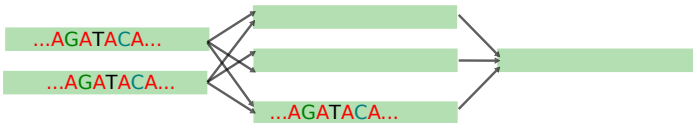
words from the reads

AGATACA
GATACAG TACAGCC AGCCATG
ATACAGC ACAGCCA CAGCCAT
...

word graph (de Bruijn graph)



Overlap graph from the reads



• de Bruijn graphs abstract redundancy

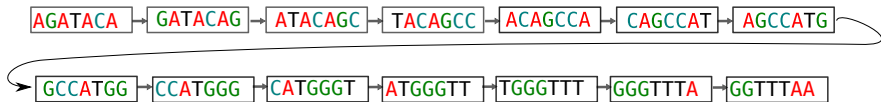
Overlapping reads

```
AGATACAGCCA
GATACAGCCATG
GATACAGCCATG
TACAGCCATGG
ACAGCCATGGG
ACAGCCATGGG
CAGCCATGGGT
AGCCATGGGTT
GCCATGGGTTT
GCCATGGGTTT
CATGGGTTTAA
CATGGGTTTAA
```

62 **(non distinct)** 7-mers in the reads

De Bruijn graph

14 **distinct** 7-mers in the De Bruijn graph

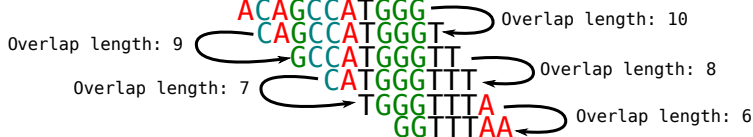


- de Bruijn graphs only rely on $k - 1$ overlaps

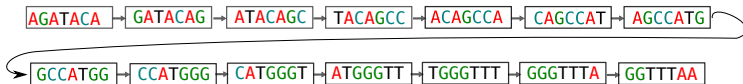
Overlapping reads

```

AGATACAGCCA
GATACAGCCATG
GATACAGCCATG
TACAGCCATGG
ACAGCCATGGG
CAGCCATGGGT
GCCATGGGTTT
CATGGGTTTT
TGGGTTTTA
GGTTTTAA
  
```

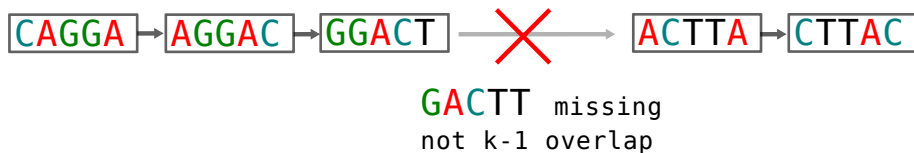


De Bruijn graph overlap length: 6



- de Bruijn graphs limitation

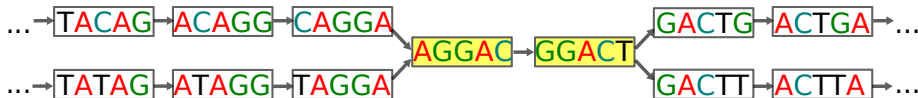
Fixed overlaps



- de Bruijn graphs limitation

Repeats...

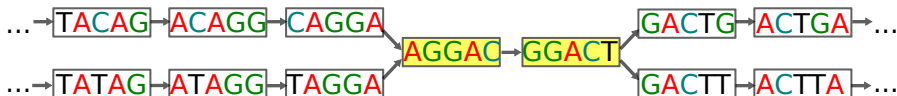
...TACAGGACTTA... ...TATAGGACTGA...



each k -mer appears only once in a de Bruijn graph

- de Bruijn graph limitation

...TACAGGACTTA... ...TATAGGACTGA...



...TATAGGA

GACTGA...

genome pieces

AGGACT

...TACAGGA

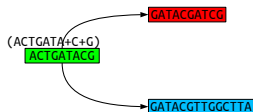
GACTTA...

- On the representation of de Bruijn graphs

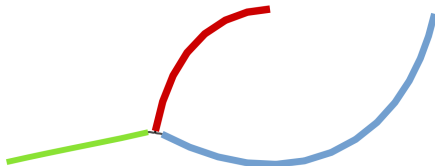
De Bruijn graph:



Compacted De Bruijn graph:



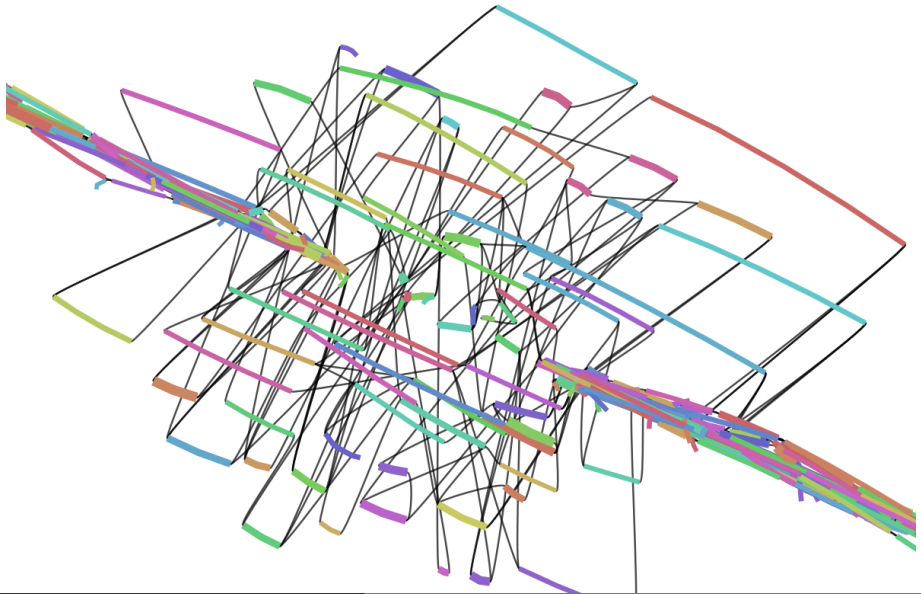
Graphical representation
(.gfa plot using Bandage):



- de Bruijn graph on a real dataset



- de Bruijn graph on a real dataset ZOOMED IN



- Sequencing errors

Genome:

ATCGGTATCGTTACGGTATACC

Reads:

ATCGCTATCG
GGTTTCGTTA
ATCGATACGG

TCGCTA
GGTTTC
ATCGAT

...

Are not genomic kmers...

• Erroneous k -mers vs genomic k -mers

Genome:

TAAGAAAGCTCTGAATCAACGGACTGCGACA

Reads:

TAAGAAAGCTCTGAATCA

AAGAAAGCTCT**A**AATCAAC

AGAAAGCTCTGAATCAACG

GAAAGCTCTGAATCAACGGA

AAAGCTCTGAATCAACGGAC

AAGCTCTGAATCAACGGACT

AGCTCTGAATCAACGGACTG

GCTCTGAATCAACGG**T**CTGC

CTCTGAATCAACGGACTGCG

TCTGAATCAACGGACTGCGA

9 times TCTGAAT

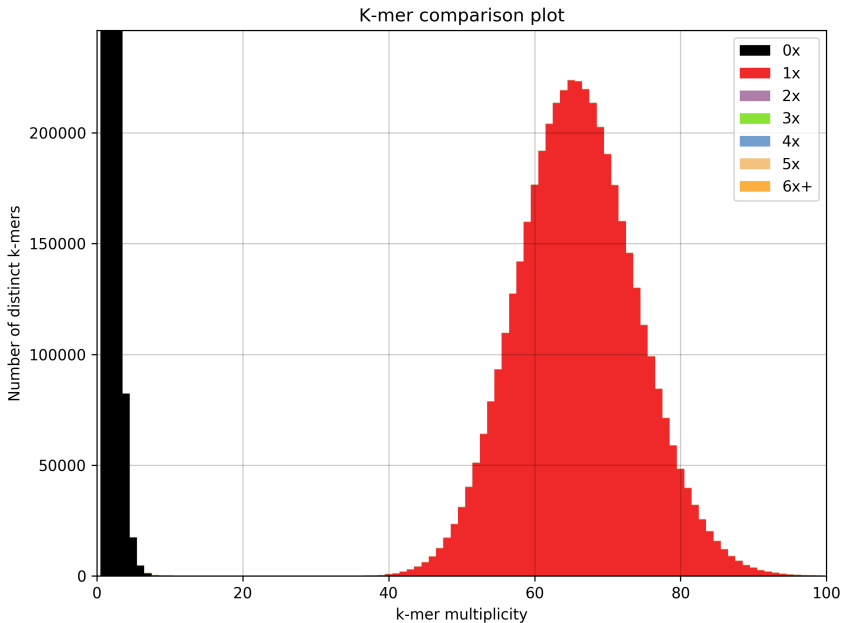
1 time TCT**A**AAT

6 times CAACGGA

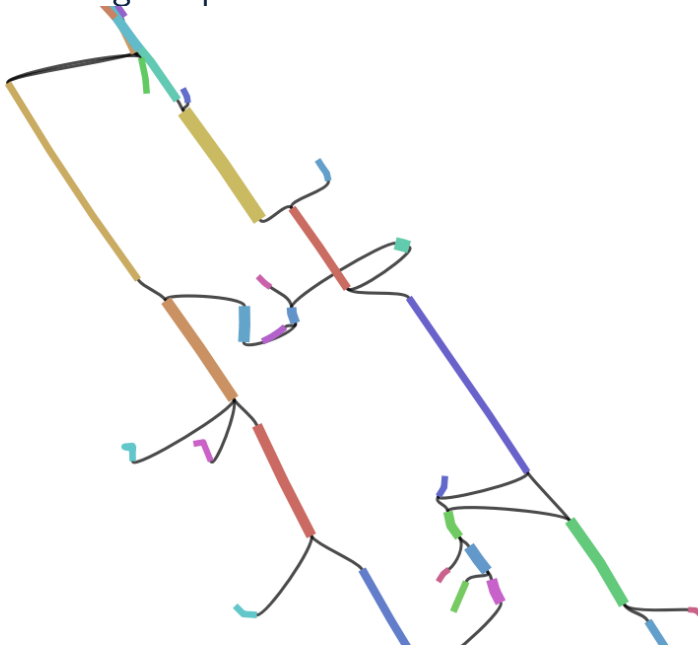
1 time CAACGG**T**

Erroneous k -mers are seen less than genomic ones

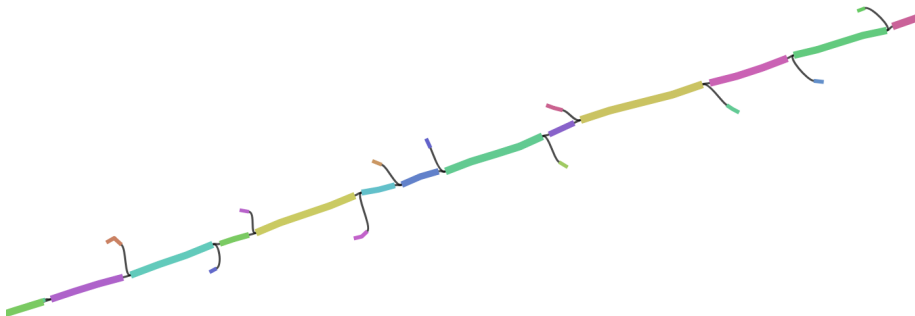
• K -mer histogram



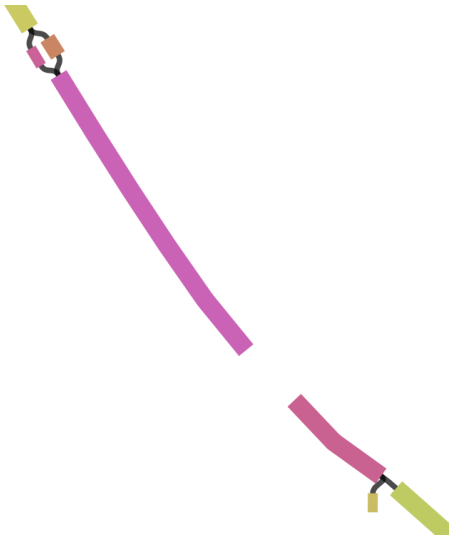
- Removing unique k -mers



- Removing k -mers seen less than 3 times



- Removing k -mers seen less than 4 times



- Errors in de Bruijn graphs

...TACAGGACTTACTGA... genome

reads

CAGGACTTA	
AGGACGTAC	← sequencing error
AGGACTTAC	
GGACTTACT	



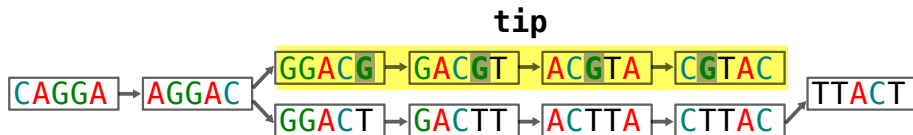
- Errors in de Bruijn graphs

...TACAGGACTTACTGA... genome

reads

C	A	G	G	A	C	T	T	A
A	G	G	A	C	G	T	A	C
A	G	G	A	C	T	T	A	C
G	G	A	C	T	T	A	C	T

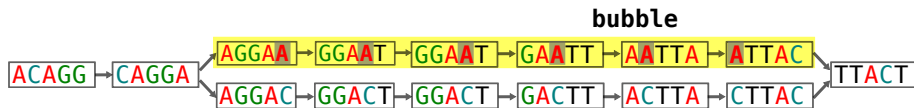
← sequencing error



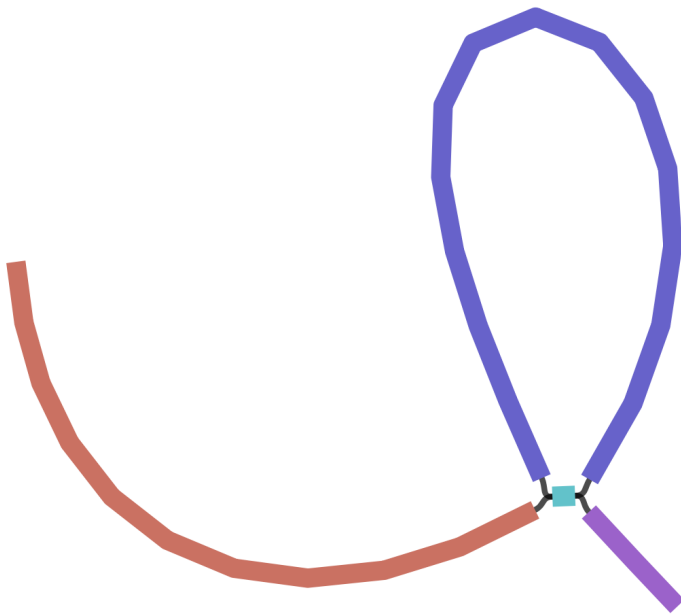
- Errors in de Bruijn graphs

...TACAGGACTTACTGA... genome

reads
 ACAGGACTTA
 CAGGAATTAC ← sequencing error
 CAGGACTTAC
 AGGACTTACT



- (Almost assembled phage !)



- de Bruijn graphs in a nutshell

- Graph of words of size k , $k-1$ overlaps
- Collapses identical k -mers
- Very successful, have replaced the overlap graphs with high throughput sequencing data
- Still outputs fragments of the genome



White spruce, 20 gigabases

- Multiple k assembly

Most de Bruijn graph assemblers can now perform several assemblies with different k -mer sizes to produce an improved "super" assembly

Exercise

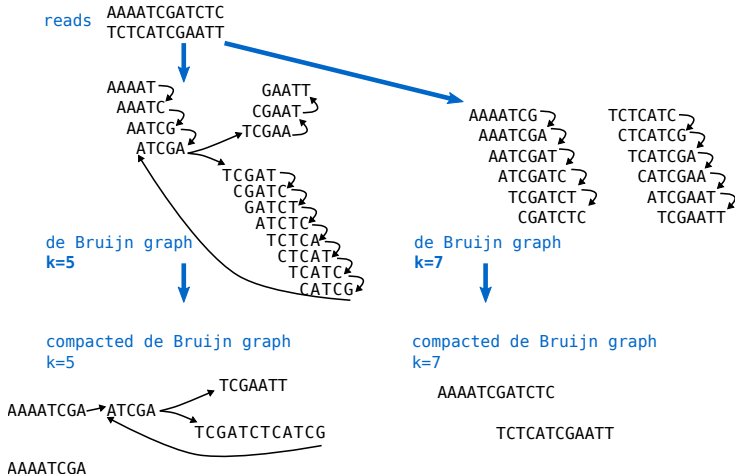
Build DBG with $k=5$ and $k=7$ from those reads

AAAATCGATCTC

TCTCATCGAATT

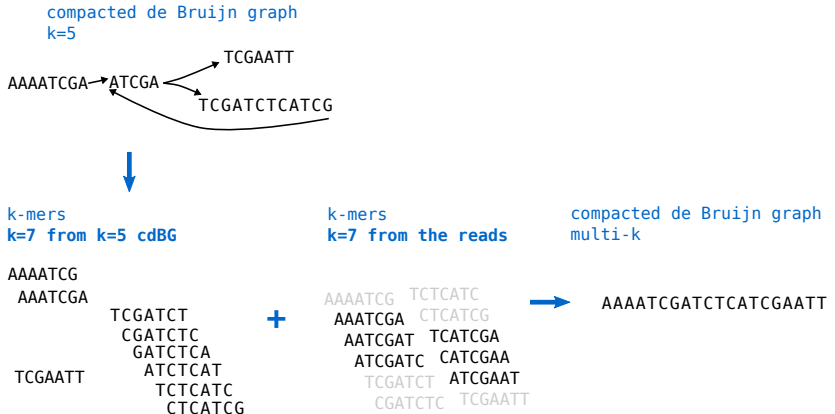
Multiple k assembly

Most de Bruijn graph assemblers can now perform several assemblies with different k -mer sizes to produce an improved "super" assembly

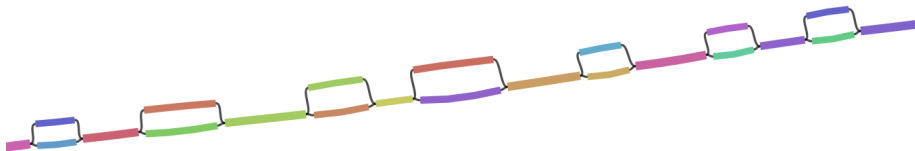


Multiple k assembly

Most de Bruijn graph assemblers can now perform several assemblies with different k -mer sizes to produce an improved "super" assembly



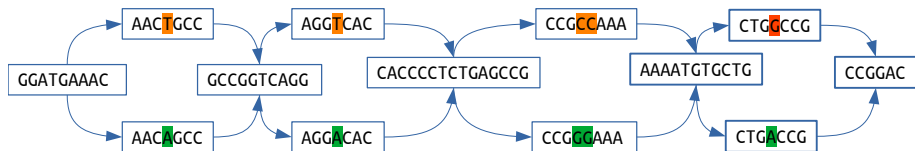
- de Bruijn graph on an eukaryota



- Two or more genomes per individual

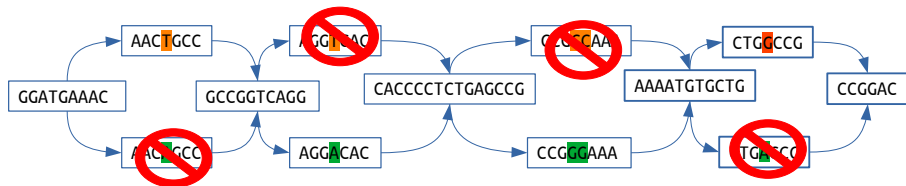
♀ GGATGAAACTGCCGGTCAGGTACCCCTCTGAGCCGCCAAAATGTGCTGCCGGAC

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC



- Two or more genomes per individual

♀ GGATGAAAC**T**GCCGGTCAGG**T**CACCCCTCTGAGCCG**CC**AAAATGTGCTG**G**CCGGAC
 ♂ GGATGAAAC**A**GCCGGTCAGG**A**CACCCCTCTGAGCCG**GG**AAAATGTGCTG**A**CCGGAC



Assembly:

GGATGAAAC**T**GCCGGTCAGG**A**CACCCCTCTGAGCCG**GG**AAAATGTGCTG**G**CCGGAC

- Assembly **concession number 2**: collapse variability

♀ GGATGAAACT**T**GCCCGGTCAGG**T**CACCCCTCTGAGCCG**CC**AAAATGTGCTG**G**CCGGAC

♂ GGATGAAAC**A**GCCCGGTCAGG**A**CACCCCTCTGAGCCG**GG**AAAATGTGCTG**A**CCGGAC

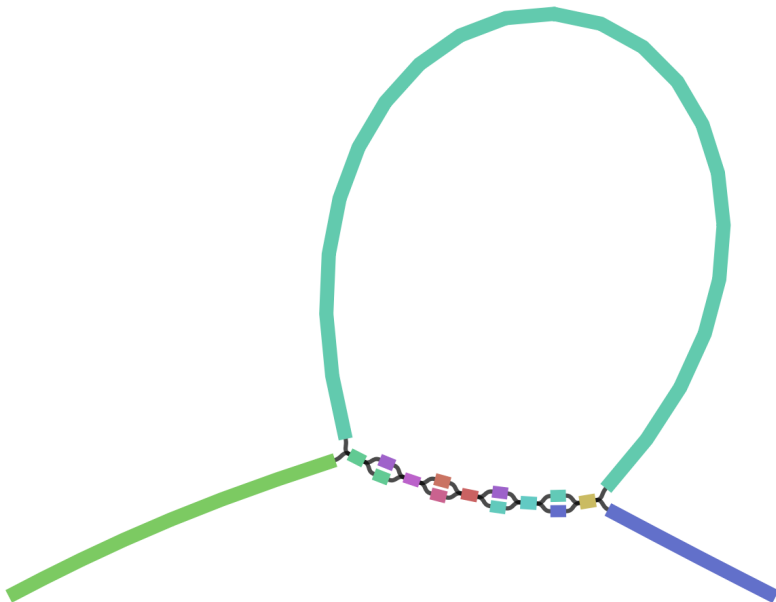
Assembly:

GGATGAAACT**T**GCCCGGTCAGG**A**CACCCCTCTGAGCCG**GG**AAAATGTGCTG**G**CCGGAC

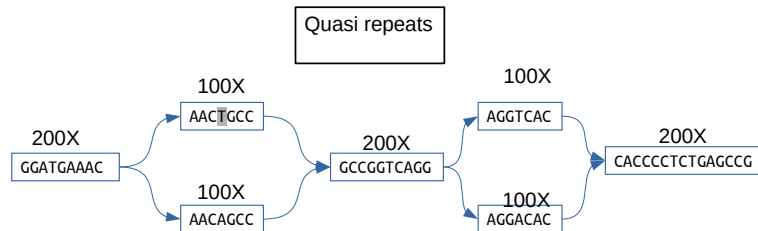
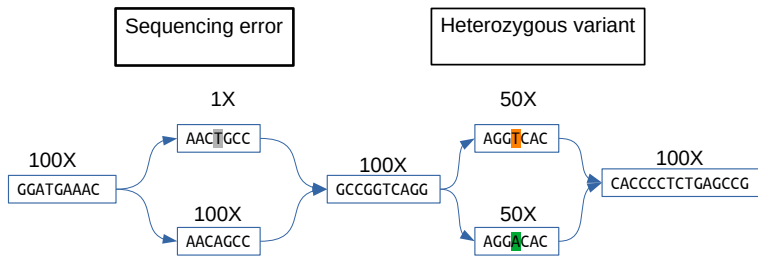
Reads:

GATGAAACT**T**
ATGAAAC**A**GC
TGAAAC**A**GCCG
GAAACT**T**GCCGG
AAACT**T**GCCGGT
AAC**A**GCCGGTC
AC**A**GCCGGTCA
CT**T**GCCGGTCAG

- Paralog genes/repeats



- Paralog genes/repeats in graph



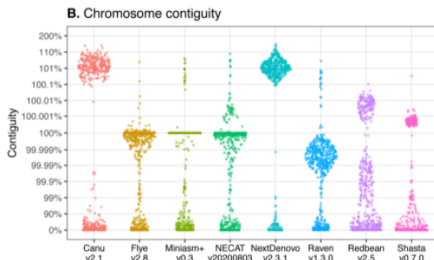
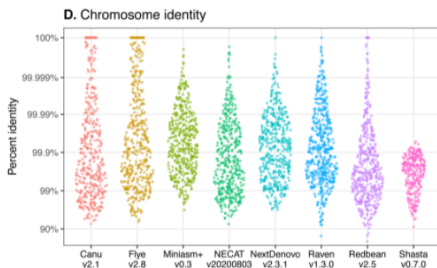
- An assembler is a set of heuristics

Graph cleaning heuristics

- Nodes coverage
- Graph local/global topology
- Reads that can be mapped on nodes
- Estimated coverage/genome size
- ...

● An assembly is a model

- 1 Assemblies contain errors
- 2 Different tools can produce very similar assemblies
- 3 A single tool can produce very different assemblies with small changes of parameters(!)



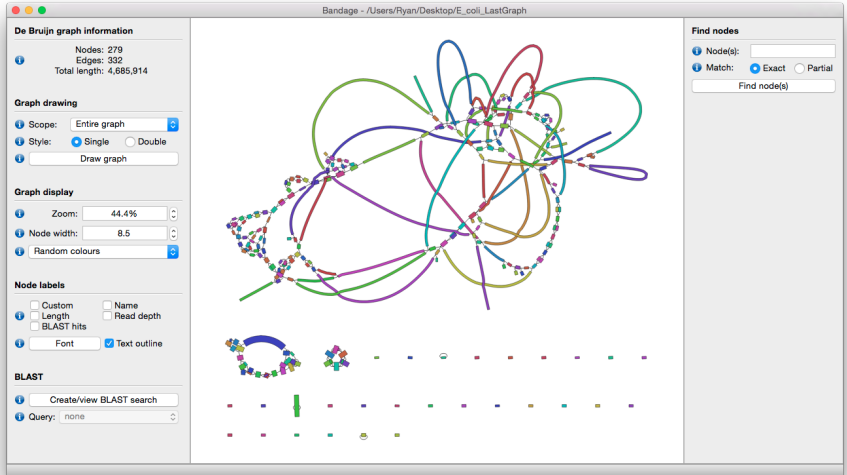
From github.com/rrwick/Long-read-assembler-comparison

- What do we do post-assembly?

- ① Assess its quality
- ② Improve it
- ③ Use it!

● Visualize assembly

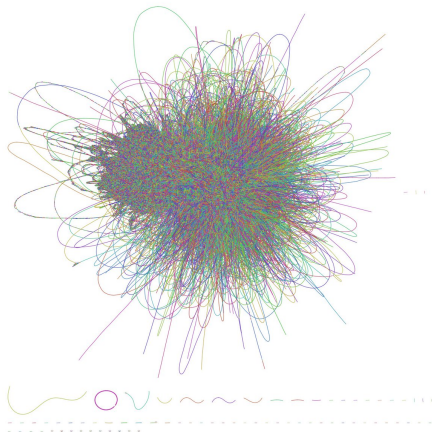
Bandage tool can visualize assembly graphs (GFA)



From [rwick.github.io/Bandage](https://github.com/rwick/Bandage)

- Visualize assembly

Bandage tool can visualize assembly graphs (GFA)



- Assembly continuity

N50

N50 can be described as a weighted median statistic such that 50% of the entire assembly is contained in contigs or scaffolds equal to or larger than this value.

Example: 1 Mbp genome 50%



- Assembly continuity

N50

N50 can be described as a weighted median statistic such that 50% of the entire assembly is contained in contigs or scaffolds equal to or larger than this value.

N75

N75 is the same statistic for 75% of the assembly

NGA50

Similar to the N50 but only takes into account contigs/scaffolds that can be **aligned** on the reference genome and consider 50% of the **genome size** instead of the assembly size

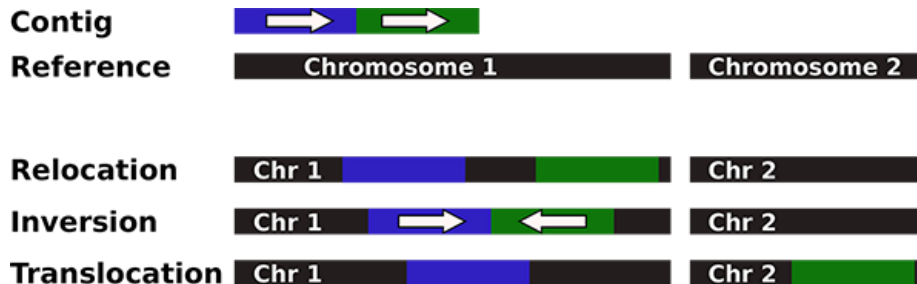
Evaluate assembly

Contigs can be mapped and compared to a reference/closely related genome



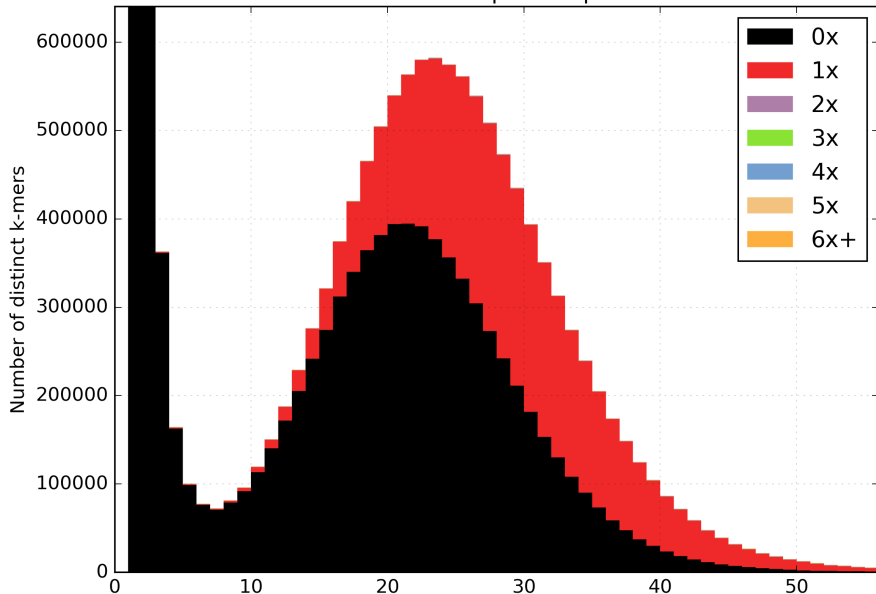
From quast.bioinf.spbau.ru/manual.html

● Misassemblies

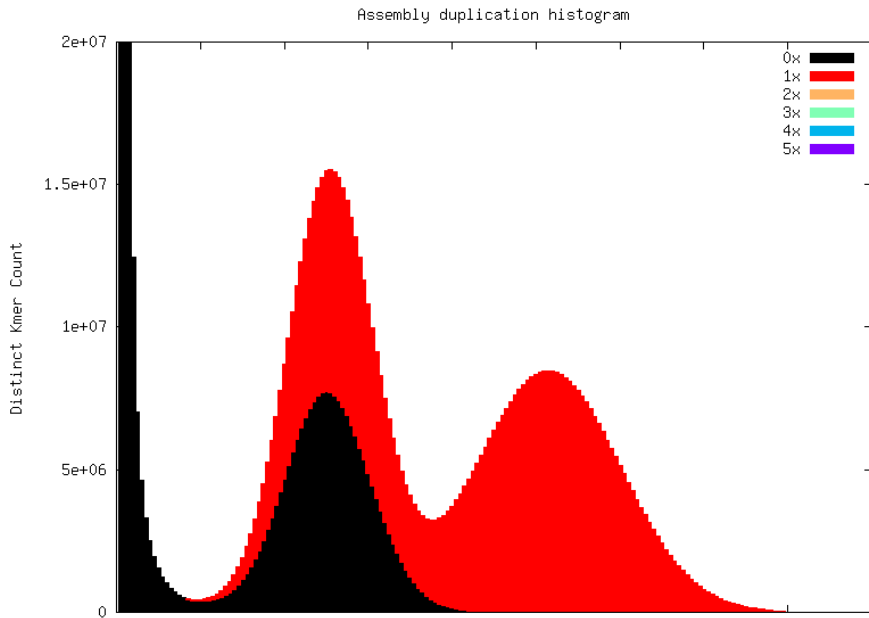


● K -mer spectrum visualization with KAT

K-mer comparison plot

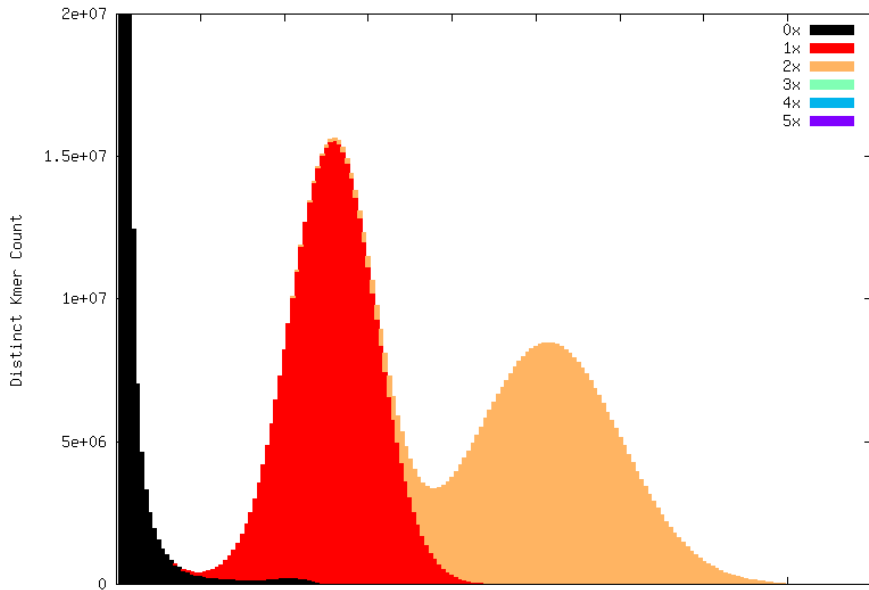


● *K*-mer spectrum visualization with KAT

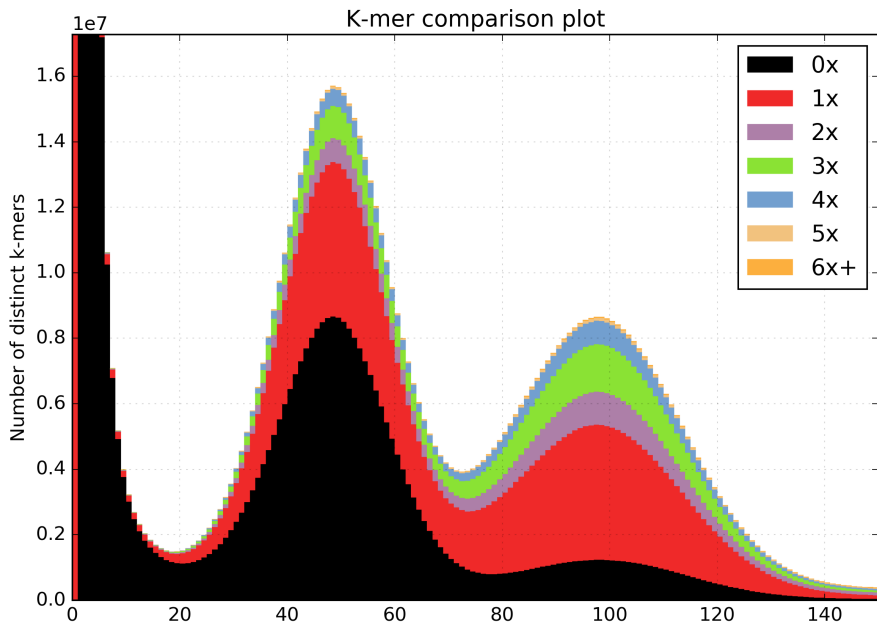


● *K*-mer spectrum visualization with KAT

Assembly duplication histogram

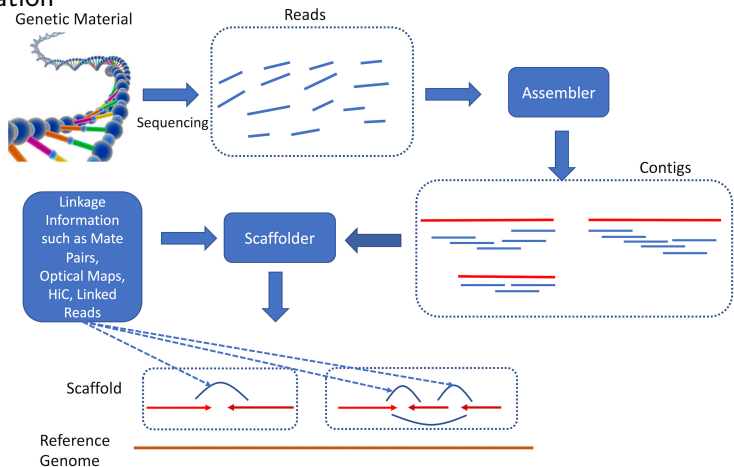


● K -mer spectrum visualization with KAT



● Scaffolding

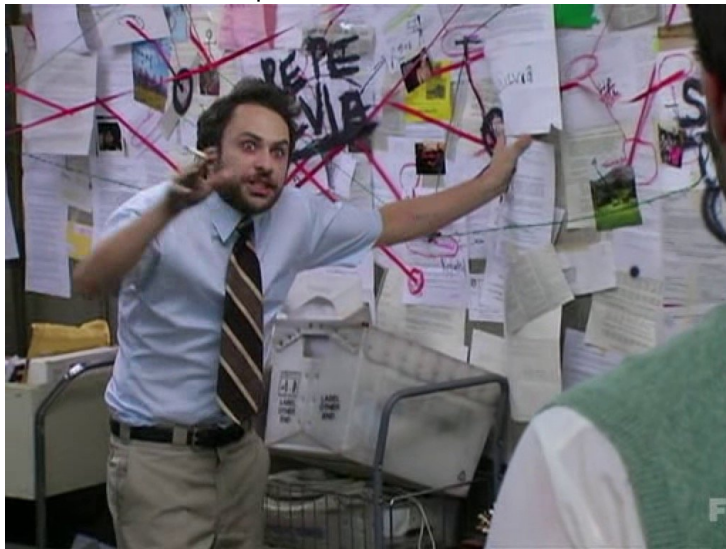
Softwares can improve the assembly continuity by using other kinds of information



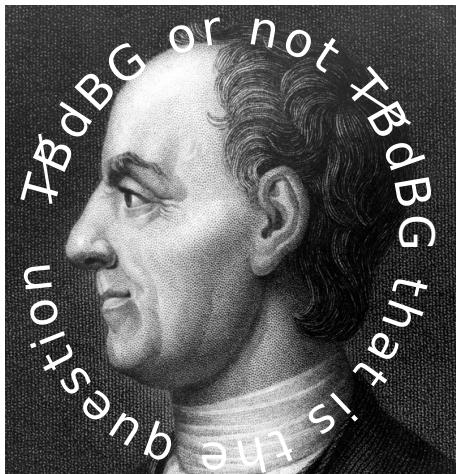
From "Modern technologies and algorithms for scaffolding assembled genomes" Plos Computational Biology

• The end

... of the theoretical part



- Intermission



• Sanger



- Medium reads $\approx 1000bp$
- Very low error rate $\approx 0.01\%$
- low throughput (up to billion of reads per run)
- Costly (\$500/Mb)

No longer used for assembly

• Second generation sequencing



NextSeq Series +



HiSeq 4000 System



HiSeq X Series†



NovaSeq 6000
System

- Short reads $\approx 150bp$
- Low error rate $< 1\%$
- High throughput (up to billion of reads per run)
- Cheap (\$0.50/Mb)
- GC bias

Mainly assembled using de Bruijn graphs

• State-of-the-art

Well performing assemblers

- SPAdes [Bankevich 2012]
- Megahit [Li 2015]
- IDBA [Peng 2012]



Other notable assemblers

- SGA [Simpson 2012]
- Discover denovo [Weisenfeld 2014]
- Abyss [Simpson 2009]

- Third generation sequencing



- Long reads $\approx 10 - 100\text{ kbp}$
- High error rate $\approx 10\%$
- High throughput (up to millions of reads per run)

• Nanopore VS Pacbio

Nanopore

- Portable
- Ultra long reads (100kbases, some reads reach the megabase level)
- Mostly deletions

Pacbio

- More mature
- HiFi reads (99.9% identity)
- Mostly insertions

Repeats spanning

Genome:

GGTA**ATGG****TTTTTT****GGTG**CTAAT**TGCG****TTTTTT****CATG**GATGTCGTA**ATTTTT**ATCTG

Reads:

GGTA**ATG** **TTTTTT** **GTG**CTAAT **TTTTTT** **ATG**GATG **TTTTTT**A
 ATGG**TTT** AAT**TGCG****TTT** ATGTCGT **TTT**ATCTG
 TTTGGTG **TTTT****CATG** CGTA**ATTT**

Contexts of the repeat:

... ATGG		ATCT ...
	??? TTTTTT ???	GGTG ...
... TGCG		CATG ...
... GTAA		

• Repeats spanning

Genome:

GGTAATGGTTTTTGGTGCTAATGCGTTTTTCATGGATGTCTGAATTTTTATCTG

Reads:

GGTAATG TTTTT GTGCTAAT G TTTTT ATGGATG TTTTTA
ATGGTTT AATGCGTT ATGTCTG TTTATCTG
TTTGGTG TTTTCATG CTGAATTT

Long reads:

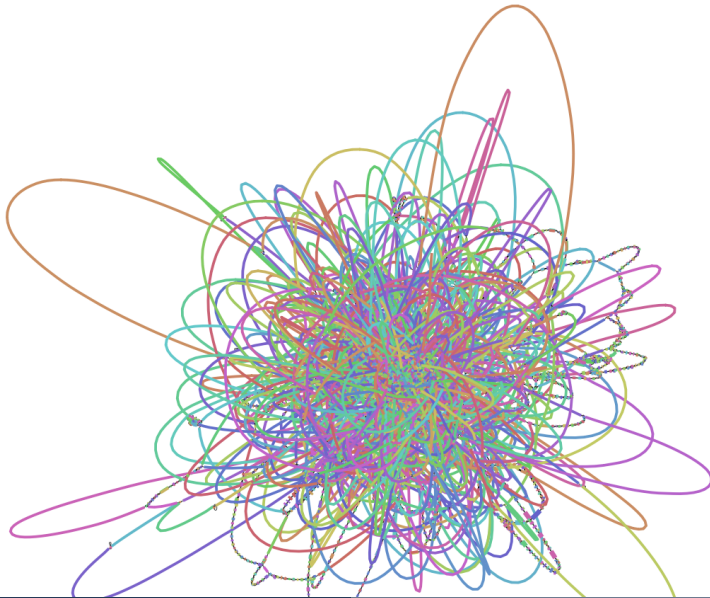
TGGTTTTTGGT TGCCTTTTTCAT TGAATTTTATCT

Contexts of the repeats:

...ATGG → GGTG... ...TGCG → CATG... ...GTAA → ATCT...

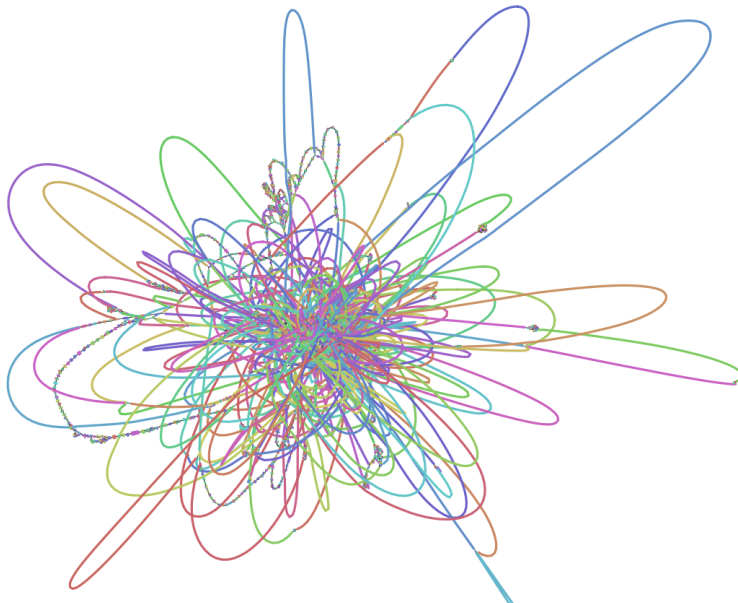
- Read length matter

Read size=21



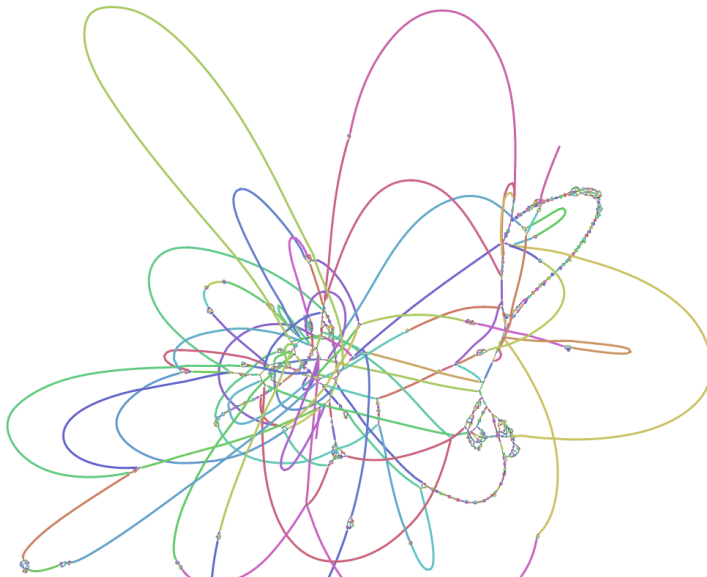
- Read length matter

Read size=31



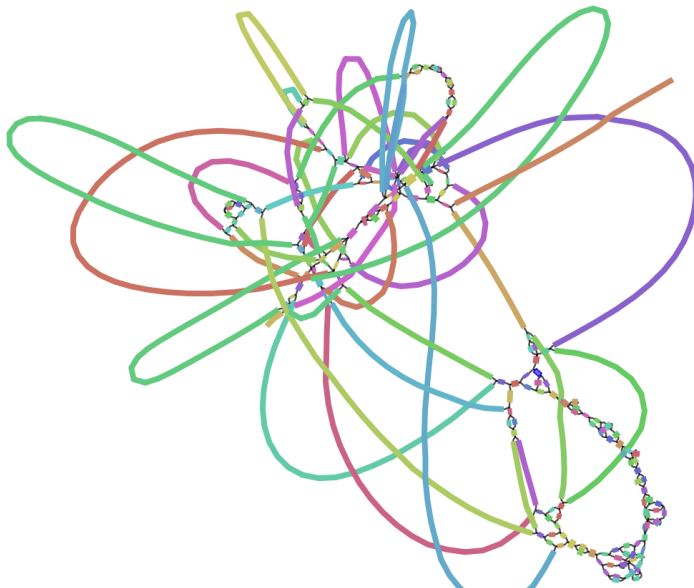
● Read length matter

Read size=63



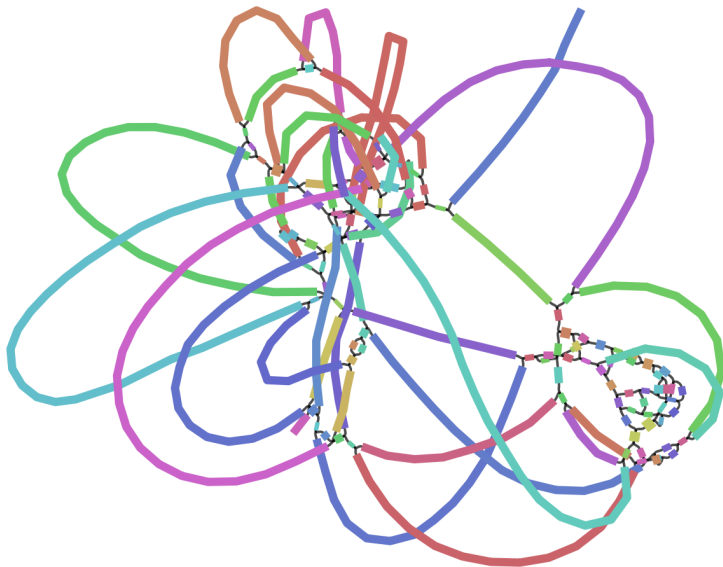
● Read length matter

Read size=255



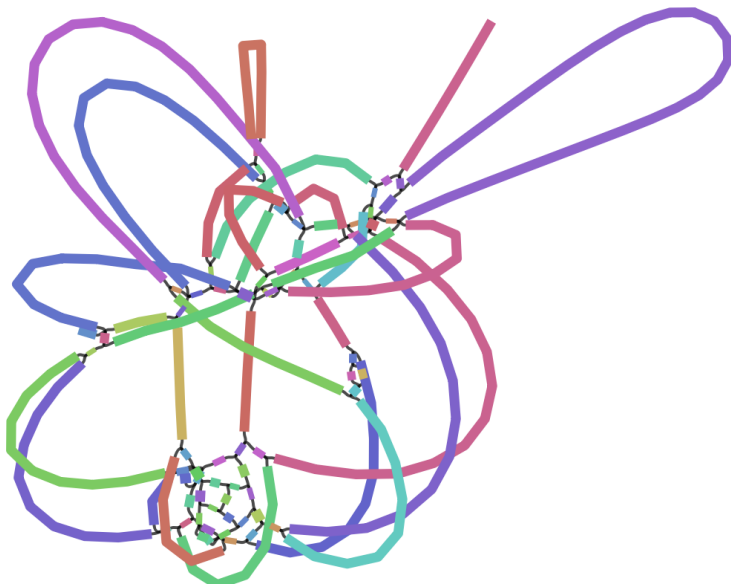
● Read length matter

Read size=500



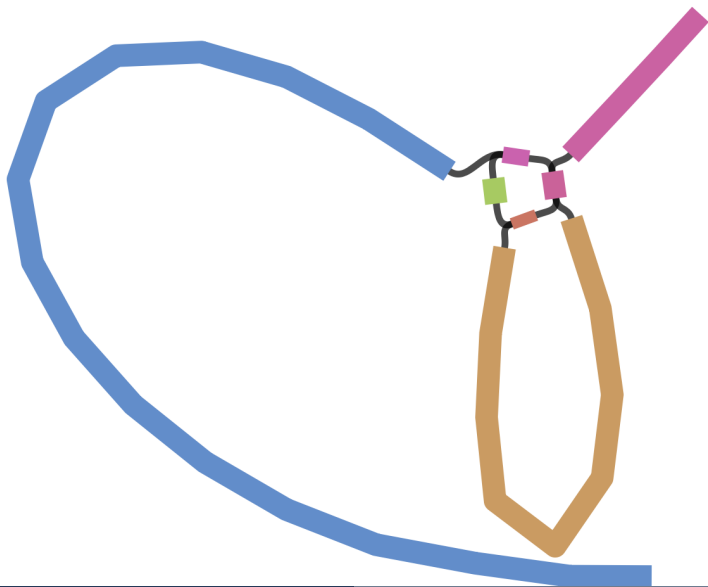
● Read length matter

Read size=1000

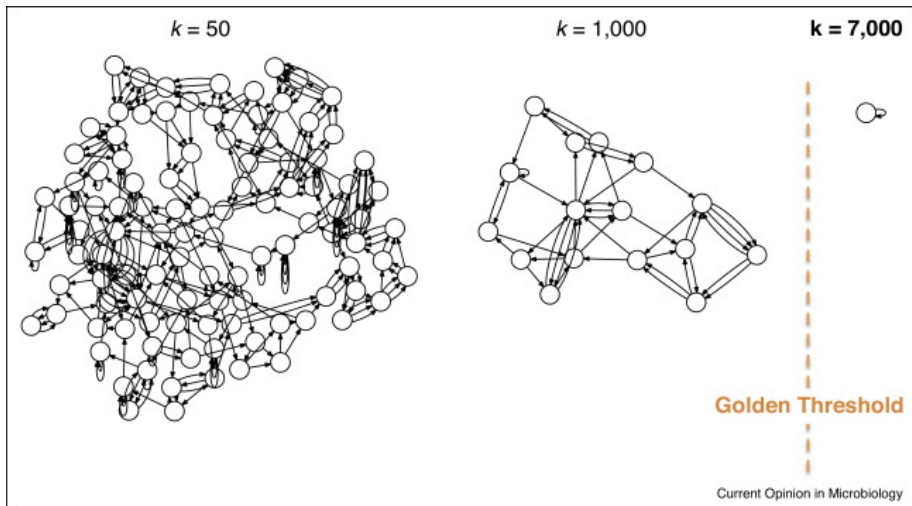


- Read length matter

Read size=2000



- Great hope for assembly



From "One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly" Current Opinion in Microbiology 2015

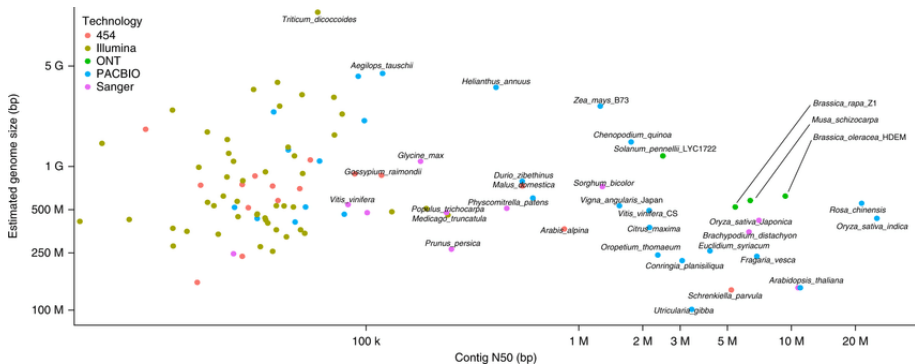
- Long reads killed the assembly star



Laura Landweber @LandweberLab · Jan 2

Our newest version of Oxytricha's somatic genome is out (rdcu.be/bZNfC) and has 18,617 distinct chromosomes. That's 2000 more than we previously published in doi.org/10.1371/journal.pgen.1004000. PacBio captured most chromosomes in single reads: Genome sequence, No assembly required

Great hope for assembly



From "Chromosome-scale assemblies of plant genomes using nanopore long reads and optical maps" Nature Plants 2018

● Which assembly strategy is best suited?

- Long reads $\approx 10kbp$
- High error rate $\approx 10\%$
- High throughput (up to millions of reads per run)

Based on long reads properties, which assembly solution would you choose and why?

Vote!

- Greedy
- Overlap graph
- de Bruijn graph

- Long reads for assembly: de Bruijn graph?

genome

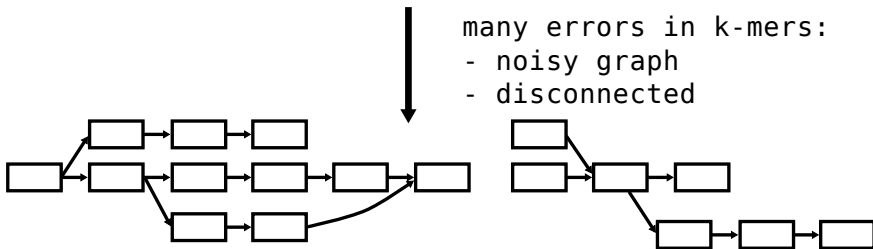


reads



many errors in k-mers:

- noisy graph
- disconnected



• Long reads for assembly: overlap graph?

Supposed to be super expensive!

TAAGAAAGCTCTGAATCAACGGACTGCGACAATAAGTGGTGGTATCCAGAATTTGCACTTCAAGTAAAAACACCTCACGAGTTAAAAACCTAAGTTC

TAAGAAAGCT

AAGAAAGCTC

AGAAAGCTCT

GAAGCTCTG

AAAGCTCTGA

AAGCTCTGAA

AGCTCTGAAT

GCTCTGAATC

CTCTGAATCA

TCTGAATCAA

CTGAATCAAC

TGAATCAACG

GAATCAACGG

AATCAACGGA

ATCAACGGAC

TCAACGGACT

CAACGGACTG

AACGGACTGC

ACGGACTGCG

CGGACTGCGA

GGACTGCGAC

GACTGCGACA

ACTGCGACAA

CTGCGACAAT

TGCGACAATA

...

TAAGAAAGCTCTGAATCAACGGACTGCGACAATAAGTGGTGGTATCCAGAATTTGCACTTCAAGTAAAAACACCTCACGAGTTAAAAACCTAAGTTC

TAAGAAAGCTCTGAATCAACGGACTGCGACA

GAAAGCTCTGAATCAACGGACTGCGACAAT

AGCTCTGAATCAACGGACTGCGACAATAAG

TCTGAATCAACGGACTGCGACAATAAGTGG

GAATCAACGGACTGCGACAATAAGTGGTGG

TCAACGGACTGCGACAATAAGTGGTGGTAT

ACGGACTGCGACAATAAGTGGTGGTATCCA

GACTGCGACAATAAGTGGTGGTATCCAG

TGGGACAATAAGTGGTGGTATCCAGAAT

GACAATAAGTGGTGGTATCCAGAATTTG

AATAAGTGGTGGTATCCAGAATTTGTCA

Average coverage: 10

Read length: 10

Average overlap: 9

Read number: 100

Average coverage: 10

Read length: 30

Average overlap: 27

Read number: 33

• Longer reads, better overlaps

- Less reads for the same coverage
- Larger overlaps

5Mb bacteria example with 100X coverage

Short reads

- 5 million 100bp reads
- 99 bp average overlap

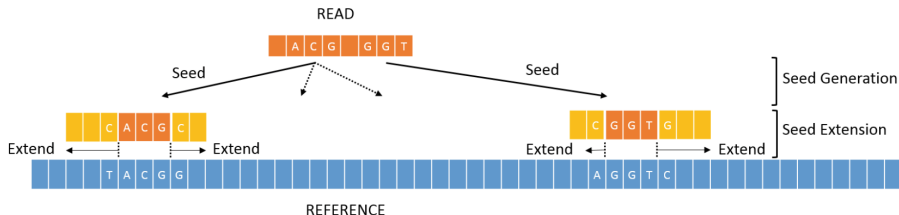
Long reads

- 50,000 10kbp reads
- 9,900 bp average overlap

Very long reads

- 5,000 100kbp reads
- 99,000 bp average overlap

- Are large overlaps hard to compute?



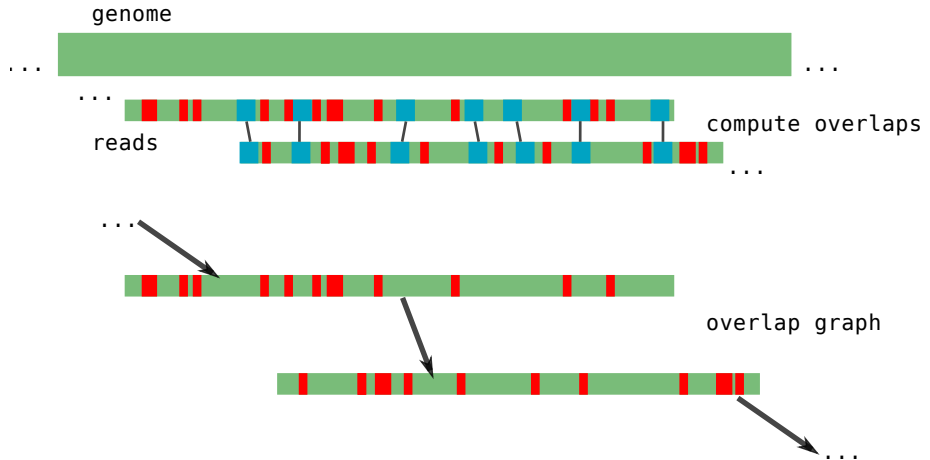
Aligning very long and highly erroneous regions is expected to be expensive, as alignment is quadratic $\approx \mathcal{O}(n^2)$!

- "Anchor chaining" in overlap graph



- For long reads: typically Minimap's [Li 2016] job
- "Anchor chaining": find common chains of anchors (k -mers) in the same order in 2 sequences (can be **linear** in practice in most cases)

- Long reads for assembly: overlap graphs



- Sequencing errors

Genome:

ATCGGTATCGTTACGGTATACC

Reads:

ATCGCTATCG

GGTATCGTCTA

ATGTTACGG

(Substitution)

(Insertion)

(Deletion)

Insertion and deletion made calling almost impossible

Using coverage to remove noise: Consensus

Genome:

TAAGAAAGCTCTGAATCAACGGACTGCGACAATAAGTGGTGGTATCCAGAATTTGTCACCTT

Reads:

AAAGAAAGCACTGAATCATGGGACTTCGAG
GAAAGCTCTCAACCAACGGACTGCGACTTT
AACTCTCAAGCAACGGACTGCGACAAAAG
TCTGAATCACCGGACTGCGTCAAAAAGTGC
GAATCACCGGACTGCGACAGTTTGTGGTGG
TCAACGCACTGCGACAATAAGTCTGGTAT
ACGGACTGCGACAAAAGTGTGGGTATCCA
GACTGCCACAAAAGTGGTGGTATCCAG
TCGGACAAAAGTGGGGGTATCCAGAAT
GACAATAAGGGGGGTATCCAAAATTTG
AAAAAGGGGTGGTATCCAGAATTTTCA
TAAGTGGGGGTATCCAAAATTTTTCAGTT

Consensus:

AAAGATAGCTCTGAATCAACGGACTGCGACAAAAGTGGTGGTATCCAGAATTTTTCAGTT

1/1 4/7 9/10 6/11 3/4

- Exercise: Perform a consensus

Erroneous reads:

TAAGAAAGCTCTGAATCAAACGGTACTGCGA
GAAAGCTTGAATCAAACGGACTGCGACAA
AGCTCTGAATCAACGGACTGCGACAATAA

Contig to polish:

TAAGAAAGCTTGAATCAACGGAATGGCGACAATAA

- Exercise: Perform a consensus - solution

Correct contig:

TAAGAAAGCTCTAATCAA-CGGACTG-CGACAATAA

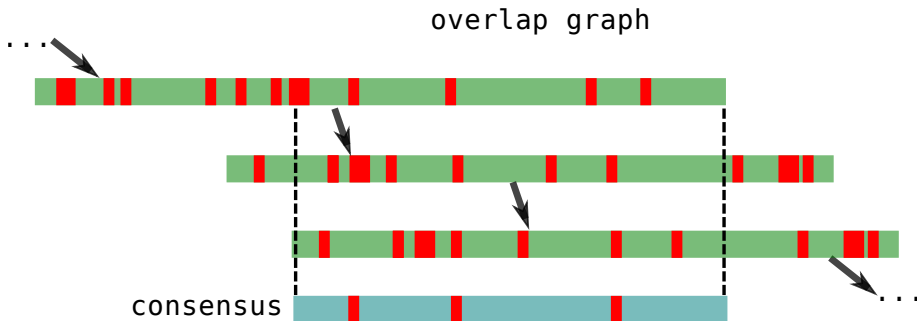
Aligned reads:

TAAGAAAGCTCTAATCAAACGGACTGTCGA

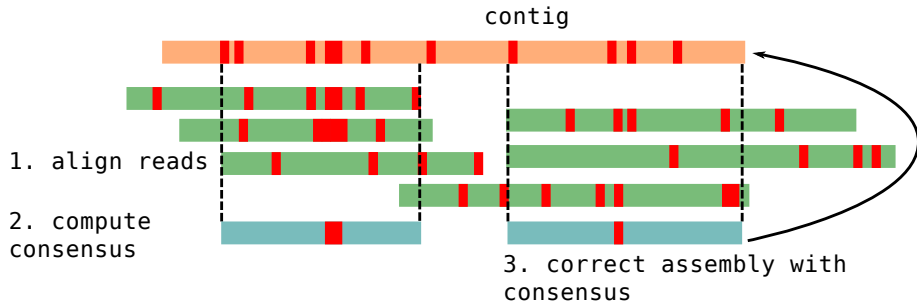
AAAGCT-TAATCAA-CGGACTG-CGACAAT

GCTCTAATCAA-CGG-CTG-CGACAATAA

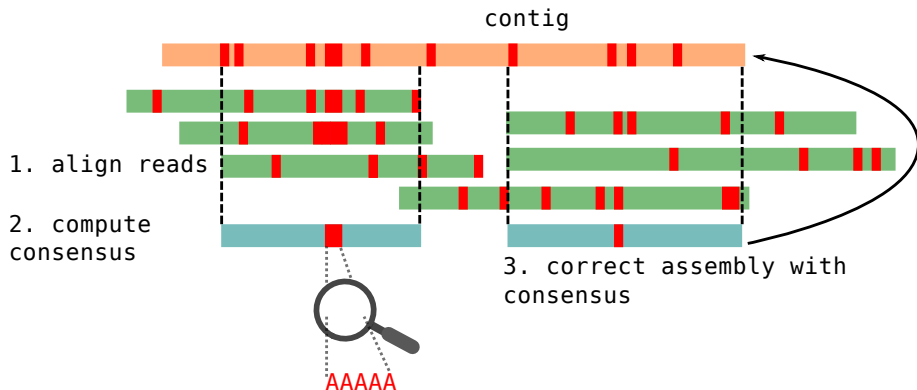
- Consensus during assembly



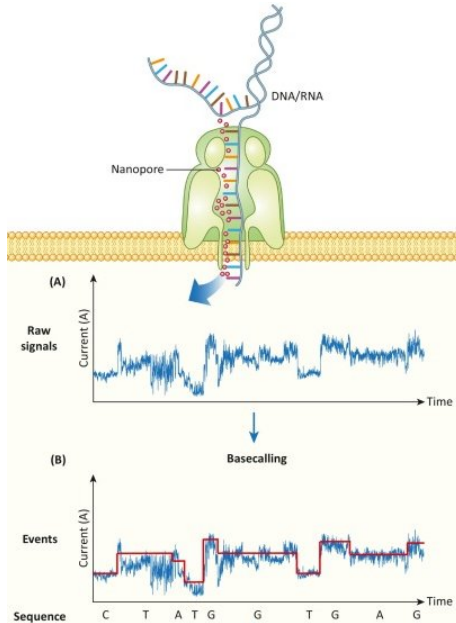
- Consensus after assembly: polishing



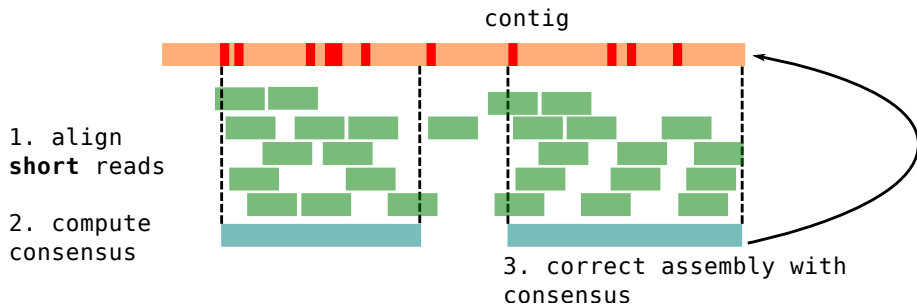
- Consensus after assembly: polishing



- Homopolymers are hard to read



- Polishing using accurate reads

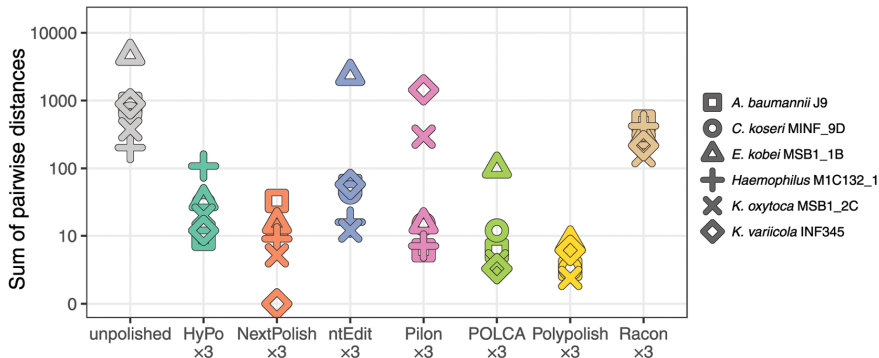


Systematics errors

Polishing with Illumina data can improve the final error rate

A. Single-tool short-read polishing

ALE change: 0 110696 113366 87707 113056 113061 115623 82446
total distance: 7635 212 74 2519 1775 128 28 1867

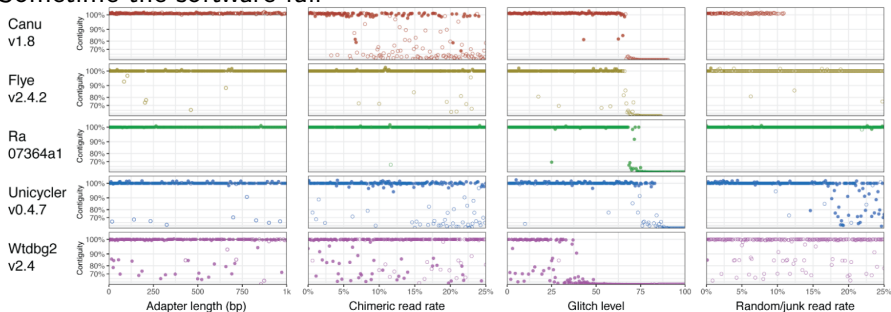


From Polypolish: Short-read polishing of long-read bacterial genome assemblies

● Long reads for assembly: assembly solved?

Assembly is not solved yet

Sometime the software fail



From github.com/rrwick/Long-read-assembler-comparison

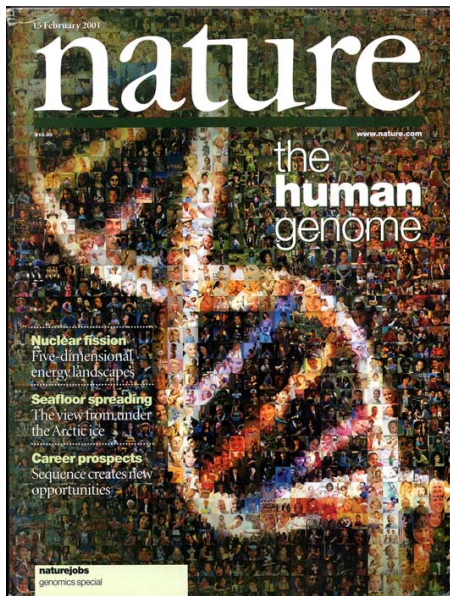
- Long reads for assembly: assembly solved?

Assembly is not solved yet

Sometimes the data cannot solve the problem

- Very large repeated region
- Low local coverage
- Chimeric/noisy reads

- 20 years later



● Telomere-to-Telomere consortium

Has produced in 2021 a complete human genome with one contig per chromosomes !

- 30x PacBio HiFi
- 120x coverage of Oxford Nanopore (ultra long reads)
- 70x PacBio CLR
- 10X Genomics, BioNano DLS and Arima Genomics HiC
- 100 authors from 50 labs

- Long reads assemblers

Best performing assemblers

- Flye (Repeat graph) [Kolmogorov et al 2019]
- Raven (OLC) [Vaser et al 2021]
- NECAT/MECAT(OLC) [Xiao et al 2017]

Other notable assemblers

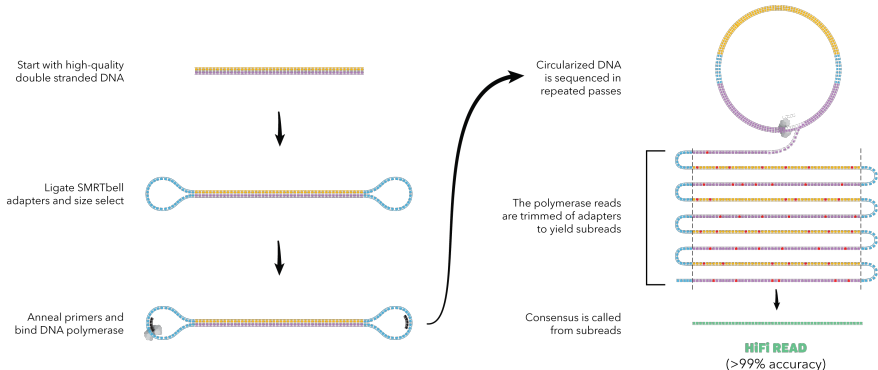
- Canu (Greedy) [Koren et al 2017]
- Shasta (OLC) [Saffin et al 2020]
- Redbean (fuzzy de Bruijn graph) [Ruan 2019]
- ...

- Long read assembly summary

- Overlap graphs with quick overlap computation
- Long reads can span repeats and improve assemblies
- Methods to polish contigs



• Consensus during sequencing



HiFi data

Stands for "High Fidelity"

Very low error rates $\approx 0.1\%$ 0.01%

Almost only homopolymer errors remain

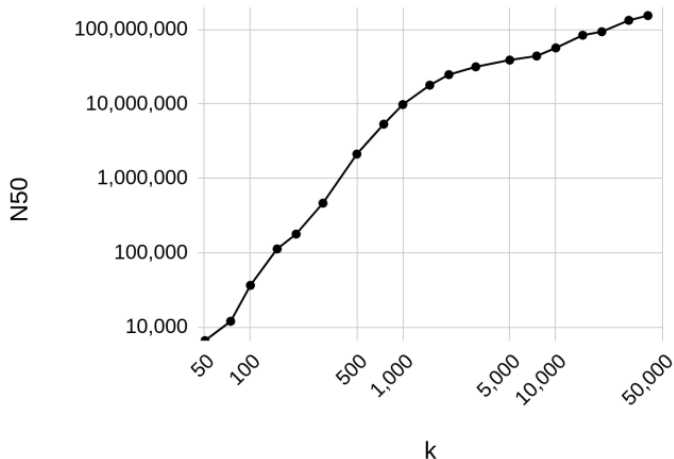
● HiFi Assembly

With almost error-less long reads we have several promising improvements ahead:

- Use de Bruijn graph (more efficient data structures)
- Assemble large genomes very fast
- Perform diploid assembly

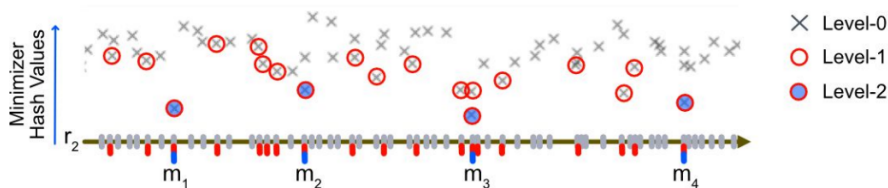
• de Bruijn graph Assembly

Using $K=500$ and $K=5000$ de Bruijn graphs to assemble

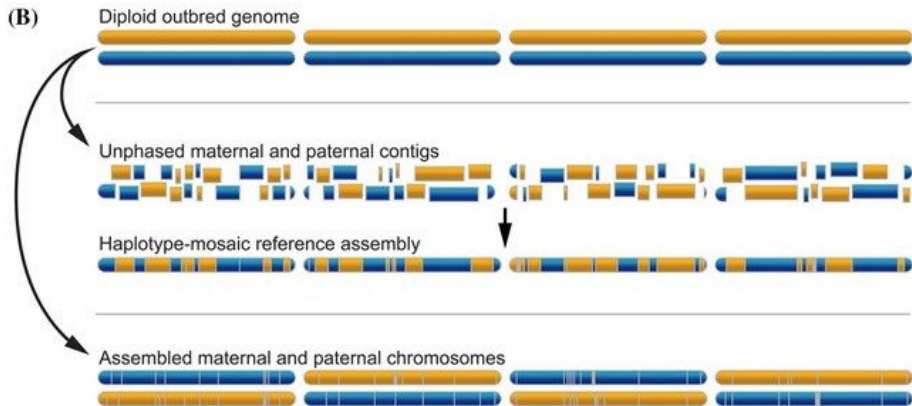


- Very fast genome assembly

Human genome assembled within 2 hours (Peregrine assembler) and 10 minutes (RMBG assembler)

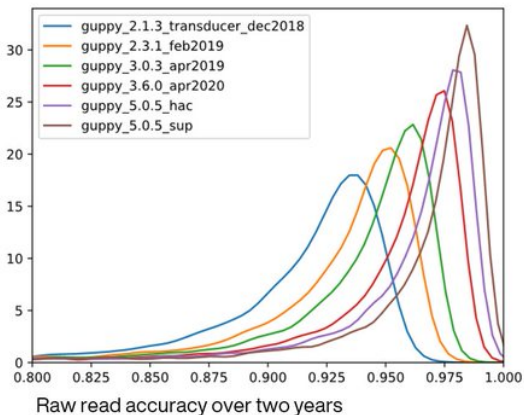


● Diploid assembly



● Ongoing progress

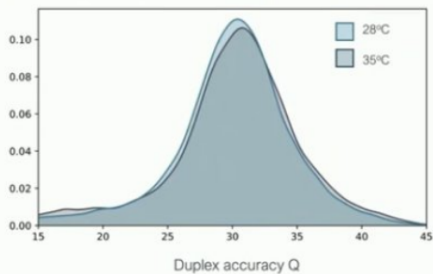
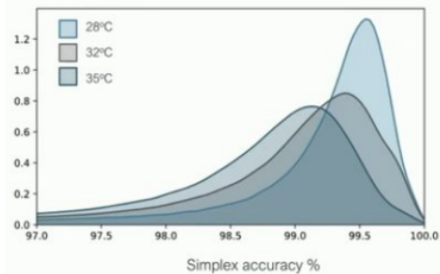
Errors in Nanopore sequencing data are rapidly diminishing



Q20 chemistry achieved modal accuracy > 99%

- High fidelity nanopore incoming?

Nanopore duplex reads could deliver long and precise reads in the future



- The end



- Take home messages

Ultra fast summary

- Short reads: de Bruijn graphs / Long reads: Overlap graph
- Repeats are the core issue
- Output fragments of genomes (**contigs**)
- Several parameters and heuristics used in practice

- On going work

Assembly Challenges

- Reconstruct haplotypes
- Scaling on large genomes
- Robustness to noisy data
- Repetitive regions

- The end



PopGenGoogling
@popgengoogleing



i trust you to figure out your own genome

[Traduire le Tweet](#)

3:01 AM · 14 déc. 2019 · [Twitter for iPhone](#)

37 Retweets **267** J'aime
