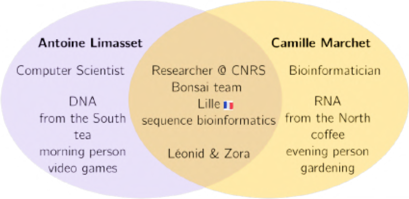
A "little" tour of assembly methods

Antoine Limasset & Camille Marchet
CRIStAL, Université de Lille, CNRS, France

Evomics Workshop on Genomics 01-09-2025

antoine.limasset@gmail.com @npmalfoy
camille.marchet@univ-lille.fr @camillemrcht



Antoine Limasset

Computer Scientist

DNA
from the South
tea
morning person
video games

Researcher @ CNRS
Bonsai team
Lille 🇫🇷
sequence bioinformatics

Léonid & Zora

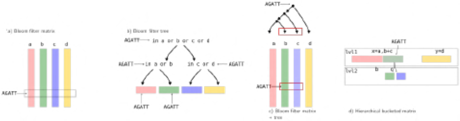
Camille Marchet

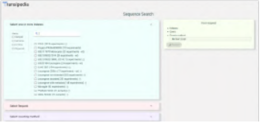
Bioinformatician

RNA
from the North
coffee
evening person
gardening

Things you might want to discuss with us:

- **methodological/scalability questions?**
- young kids/work balance
- impostor syndrome when doing CS stuff
- nice tenured positions in France





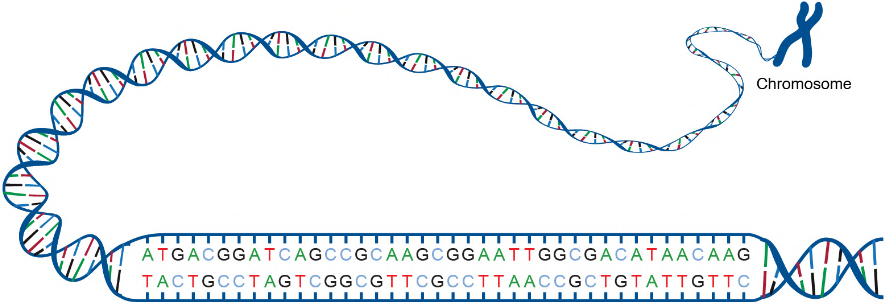
Content of this course

- How to reconstruct a genome with sequencing data?
- What are the main challenges?
- Which solutions have been proposed?

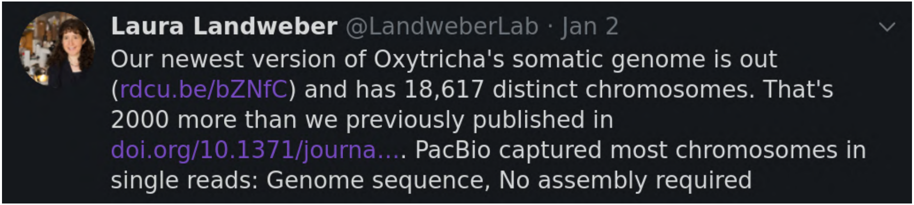
Bingo: find a book that we both love (French title).

genome size: $\sim$ 40 gigabases

- Accessing a genome


Chromosome
ATGACGGATCAGCCGCAAGCGGAATTGGCGACATAACAAG
TACTGCCTAGTCGGCGTTCGCCTTAACCGCTGTATTGTTC

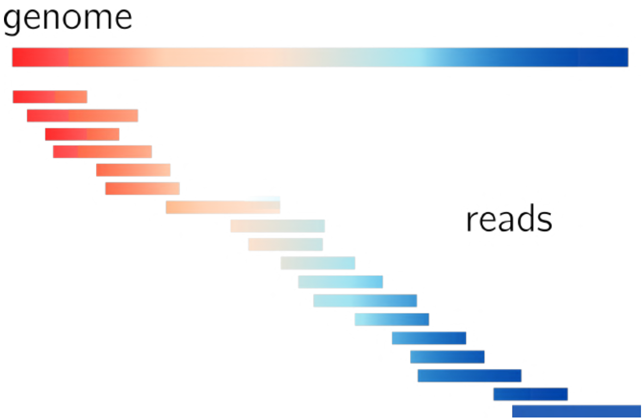
- Why do we need assembly?



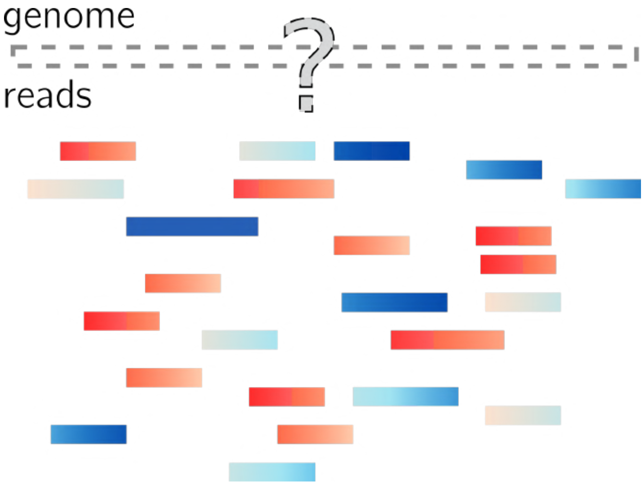
Laura Landweber @LandweberLab · Jan 2

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/journa.... PacBio captured most chromosomes in single reads: Genome sequence, No assembly required

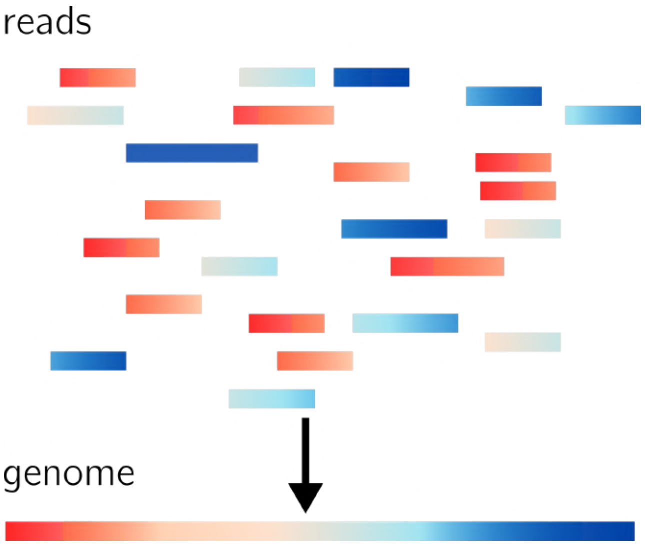
- Reads are subsequences from the genome



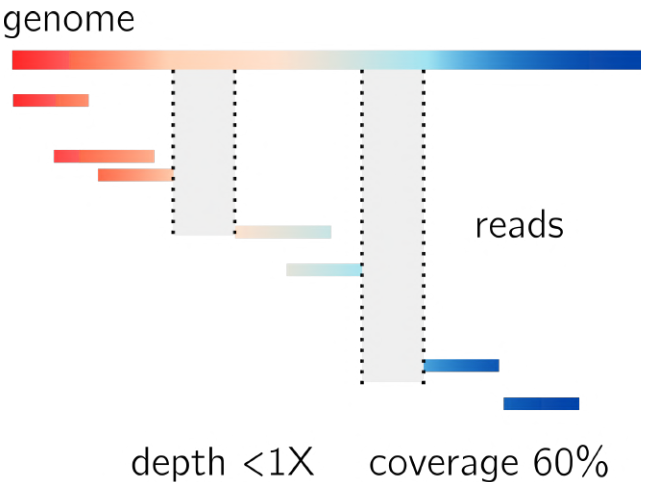
- Reads are **shuffled** subsequences from the genome



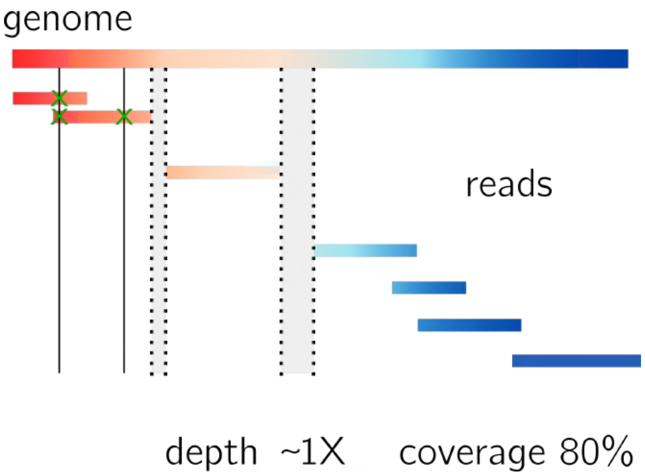
Genome assembly task



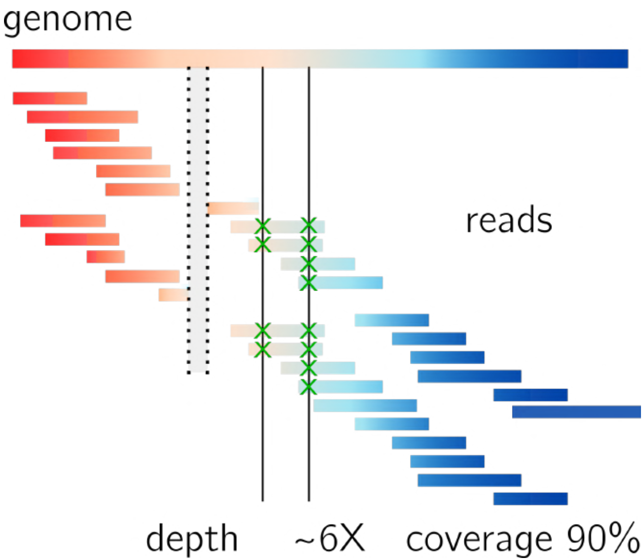
- Genome sequencing: depth & coverage



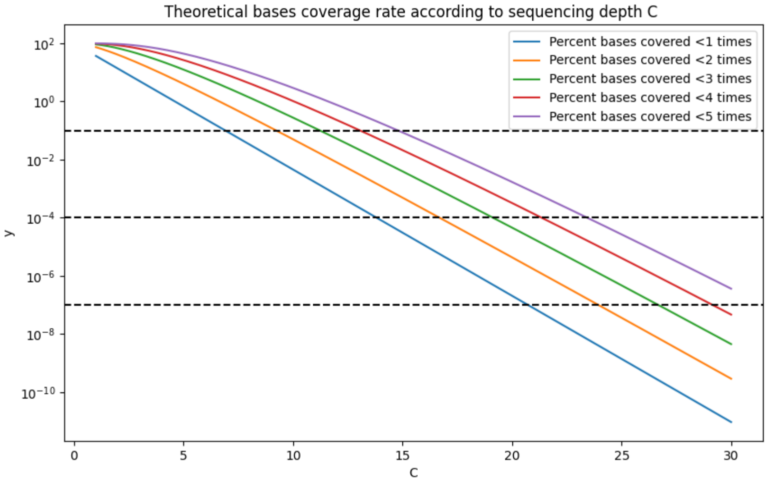
- Genome sequencing: depth & coverage



- Genome sequencing: depth & coverage

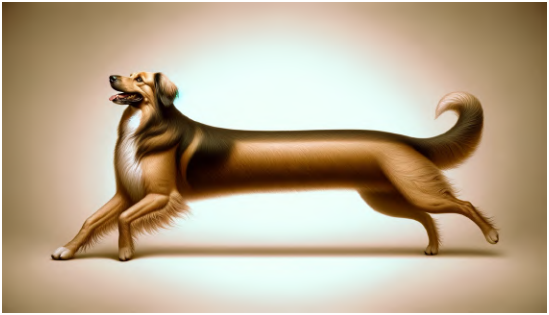


Poisson law



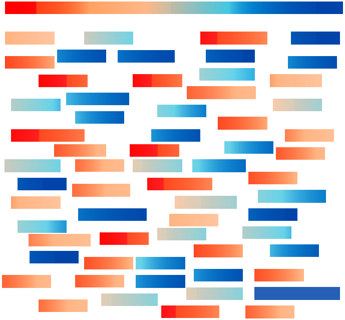
30X are often required for assembly projects

•First experiment: *Long, perfect boy*'s genome



Genome size
1 billion bases

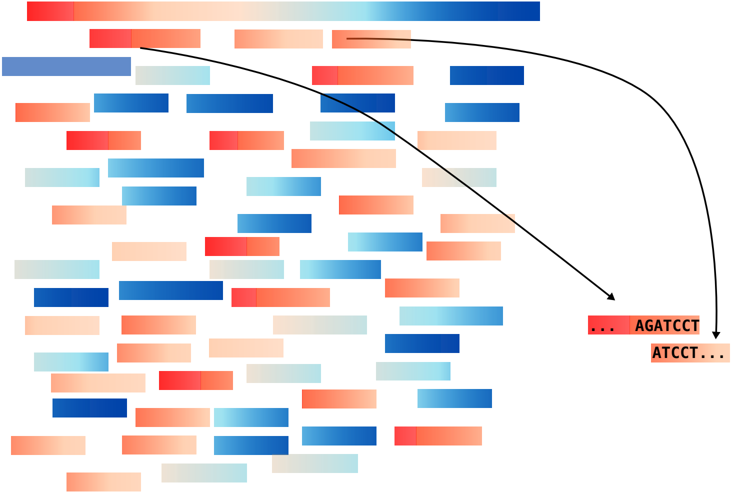
100kb region from the genome
(only for the record, we actually don't have it)



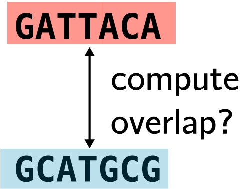
Reads
10 million
mean size 10kb

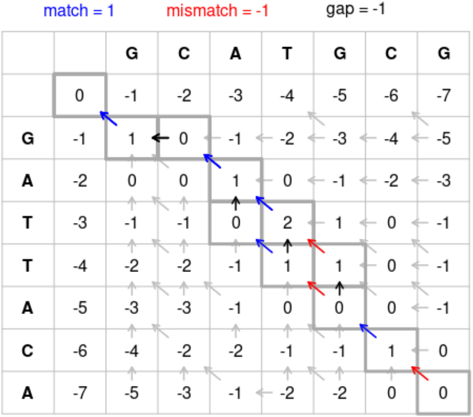
Order according to overlaps

Overlaping reads are likely successive part of the genome



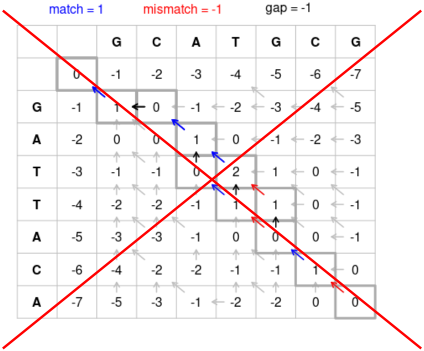
•How to compute the overlaps? Alignment?



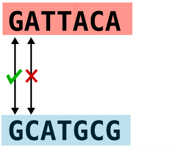

match = 1   mismatch = -1   gap = -1

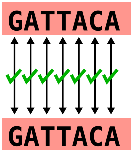
		G	C	A	T	G	C	G
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

- How to compute the overlaps? Quick exact match!


match = 1 mismatch = -1 gap = -1

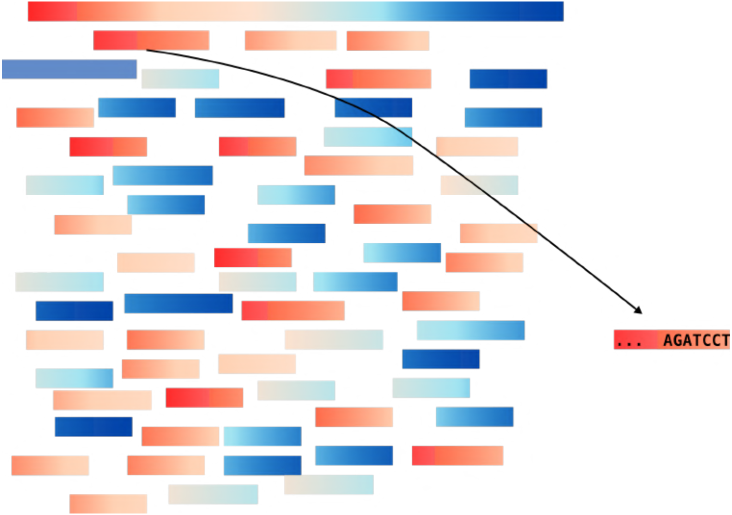
		G	C	A	T	G	C	G
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0





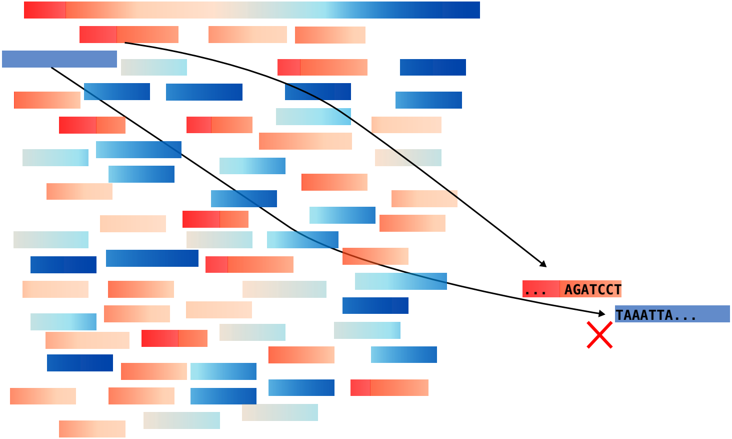
Check all reads for overlaps

For a given read, scan all others



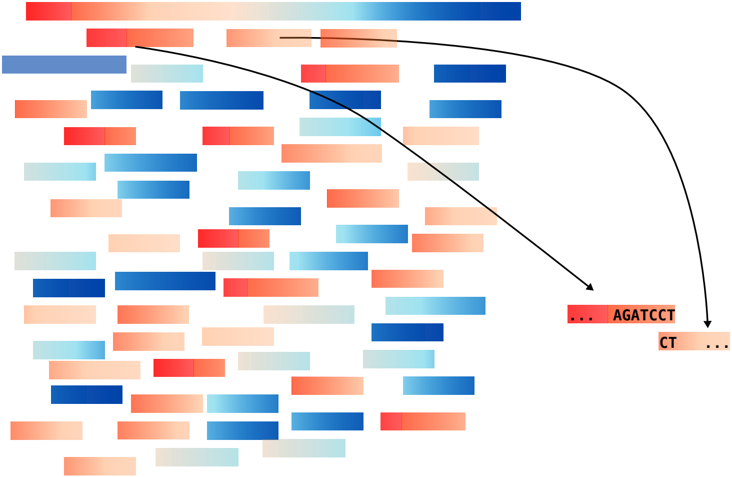
Most cases

No overlap



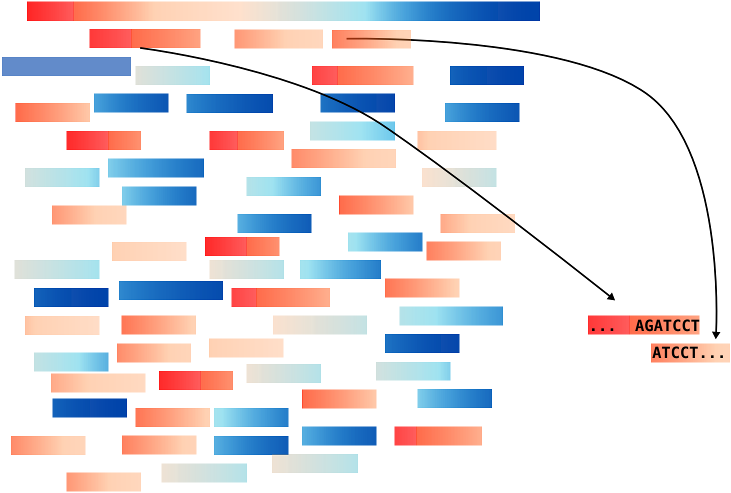
•Small overlaps

Can happen "by chance"

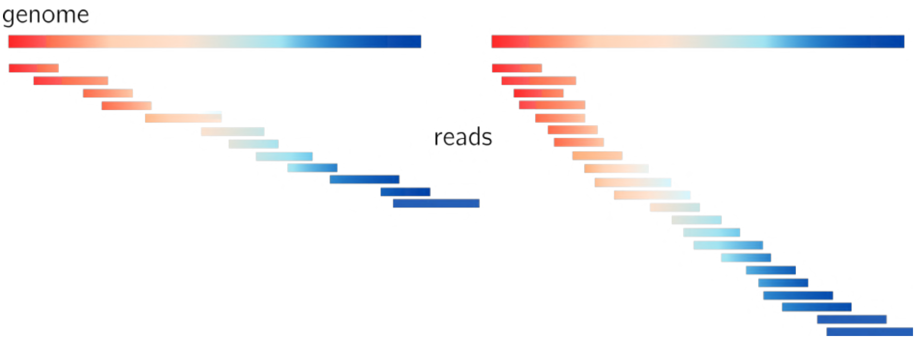


•Longest overlaps

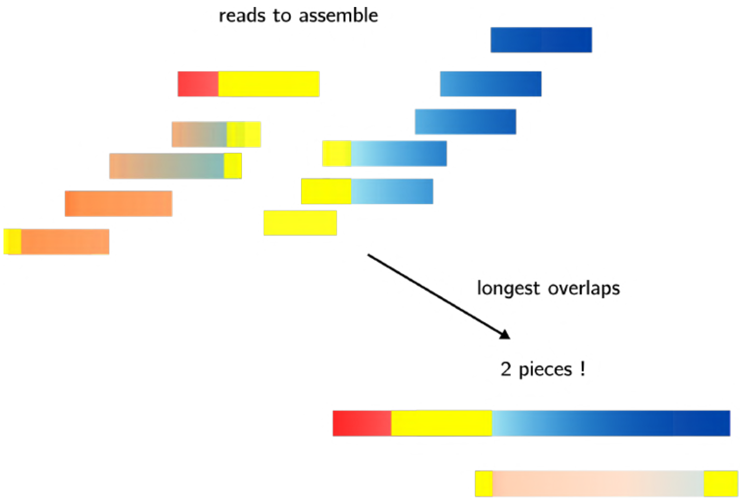
We are more confident in longer overlaps



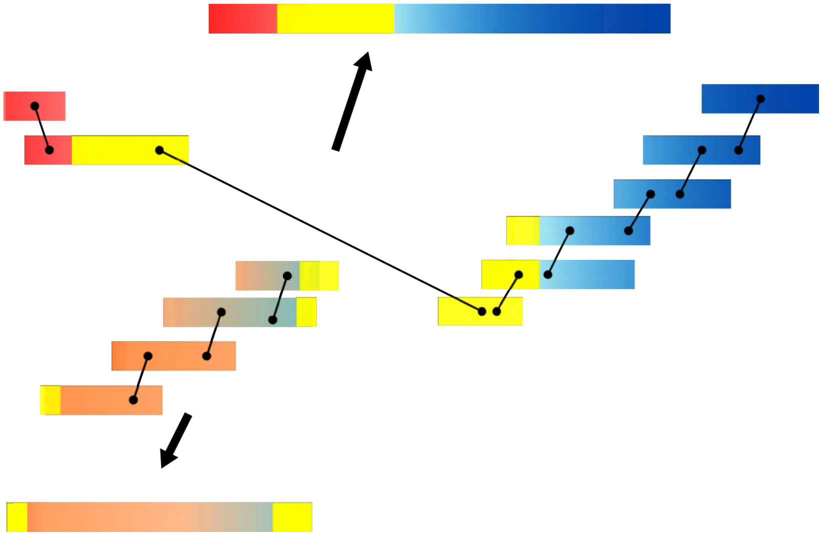
- Higher depth, longer overlaps



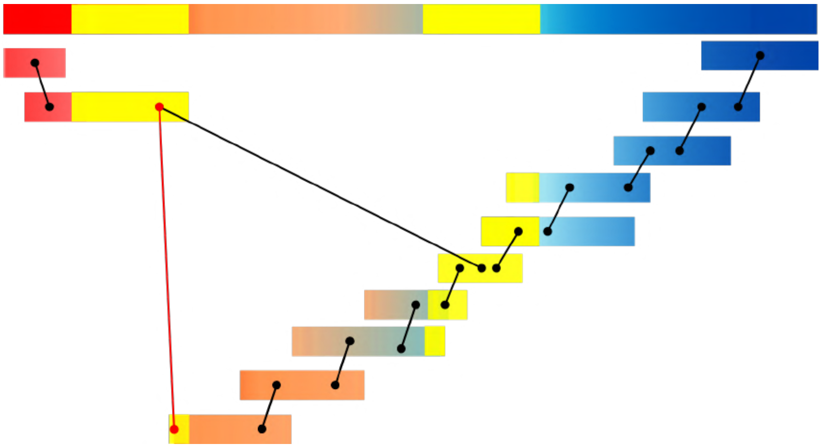
•Something weird happened



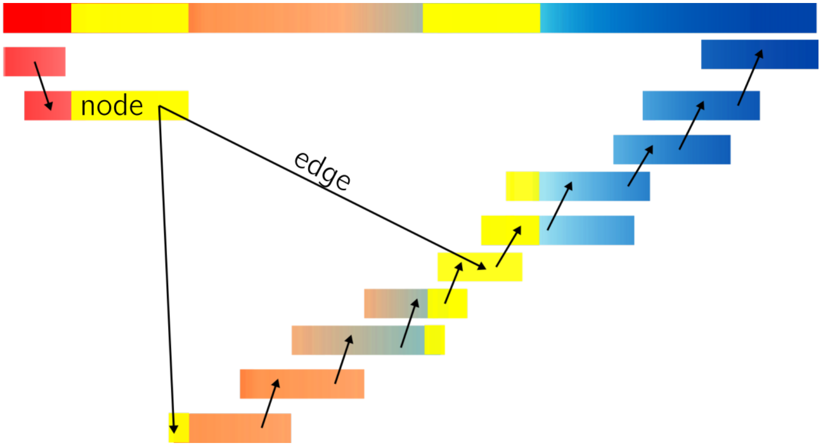
- All longest overlaps



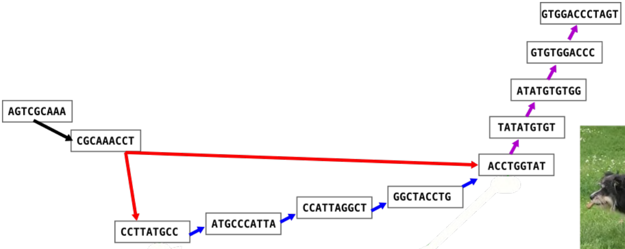
- Take into account other overlaps?



Look, a graph!



Unsafe paths in an overlap graph



AGTCGCAAA

CGCAAACCT

CCTTATGCC

ATGCCCATTA

CCATTAGGCT

GGCTACCTG

ACCTGGTAT

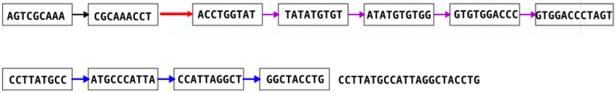
TATATGTGT

ATATGTGTGG

GTGTGGACCC

GTGGACCCTAGT

spurious assembly result



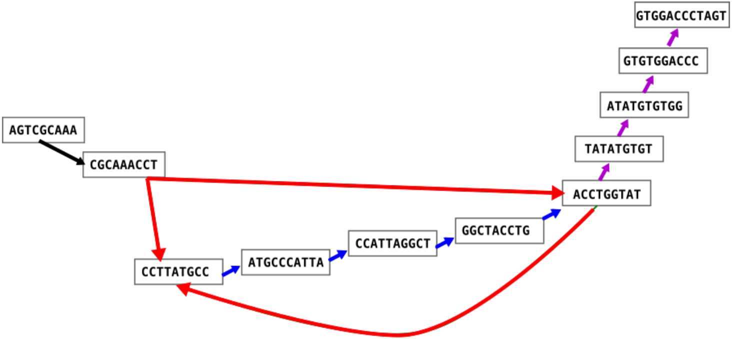
AGTCGCAAA → CGCAAACCT → ACCTGGTAT → TATATGTGT → ATATGTGTGG → GTGTGGACCC → GTGGACCCTAGT

AGTCGCAAACCTGGTATATGTGTGGACCCTAGT

CCTTATGCC → ATGCCCATTA → CCATTAGGCT → GGCTACCTG

CCTTATGCCATTAGGCTACCTG

Safe paths in an overlap graph



assembly result

AGTCGCAAA → CGCAAACCT AGTCGCAAACCT

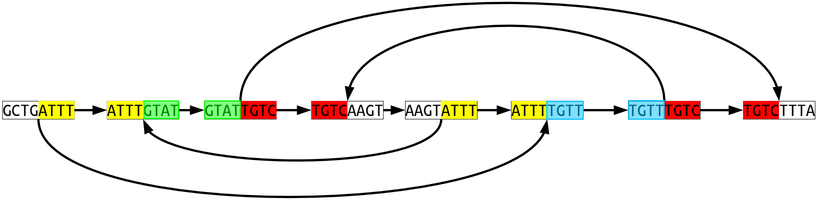
CCTTATGCC → ATGCCCATTA → CCATTAGGCT → GGCTACCTG CCTTATGCCATTAGGCTACCTG

ACCTGGTAT → TATATGTGT → ATATGTGTGG → GTGTGGACCC → GTGGACCCTAGT ACCTGGTATATGTGTGGACCCTAGT

Multiple repeats

```
Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA
```

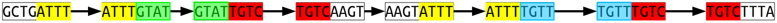
Overlap graph:



First solution

```
Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA
```

Overlap graph:



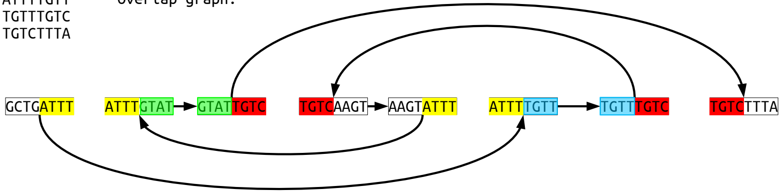
GCTGATTT → ATTTGTAT → GTATTGTC → TGTCAAGT → AAGTATTT → ATTTTGTT → TGTTTGTC → TGTCTTTA

```
Possible assemblies:
GCTGATTTGTATTGTCAAGTATTTTGTTTGTCTTTA
```

- Second solution

```
Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA
```

Overlap graph:



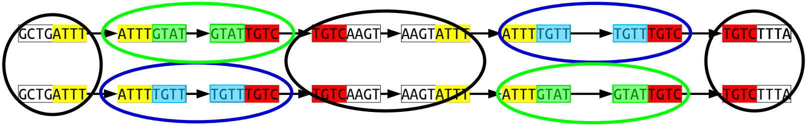
GCTGATTT ATTTGTAT GTATTGTC TGTCAAGT AAGTATTT ATTTTGTT TGTTTGTC TGTCTTTA

```
Possible assemblies:
GCTGATTTGTATTGTCAAGTATTTTGTTTGTCTTTA
GCTGATTTTGTTTGTCAAGTATTTGTATTGTCTTTA
```

Those two solutions are indistinguishable

- Parsimonious solution: do not assemble

Possible assemblies:


GCTGATTT → ATTTGTAT → GTATTGTC → TGTCAAGT → AAGTATTT → ATTTTGTT → TGTTTGTC → TGTCTTTA
GCTGATTT → ATTTTGTT → TGTTTGTC → TGTCAAGT → AAGTATTT → ATTTGTAT → GTATTGTC → TGTCTTTA

Genome pieces:

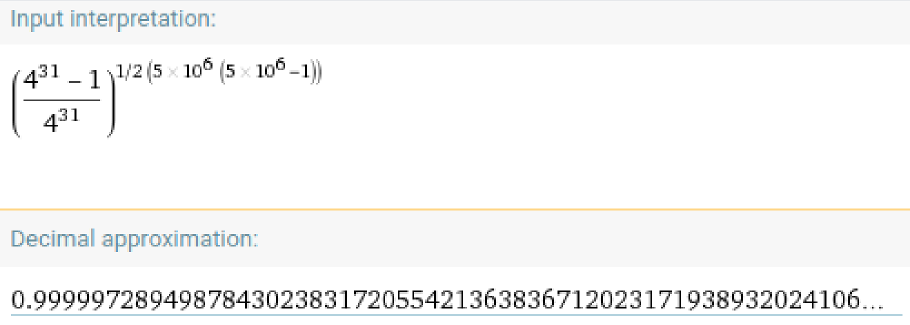
GCTGATTT   ATTTGTATTGTC   TGTCAAGTATTT   ATTTTGTTTGTC

Repeats lead to the fragmentation of the assembly

Genomes pieces that make **con**sensus across the differents solution are called **Con**tigs

Do we expect many repeats?

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



Input interpretation:

$$\left(\frac{4^{31}-1}{4^{31}}\right)^{1/2\left(5\times10^{6}\left(5\times10^{6}-1\right)\right)}$$

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

- The burden of assembly: genomic repeats

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

- 15: 44,994
- 21: 1,169

- 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

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

- 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

The burden of assembly: genomic repeats

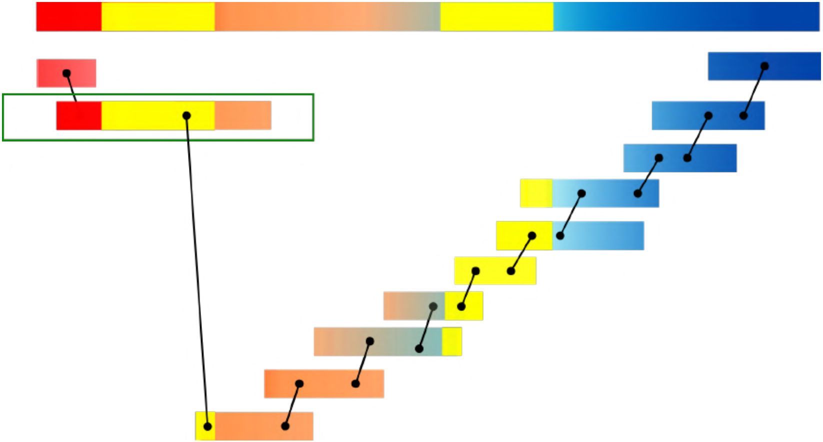
Amount of repeats larger than a given size in *E. coli* genome

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

Genomic repeats are NOT random events

With longer reads

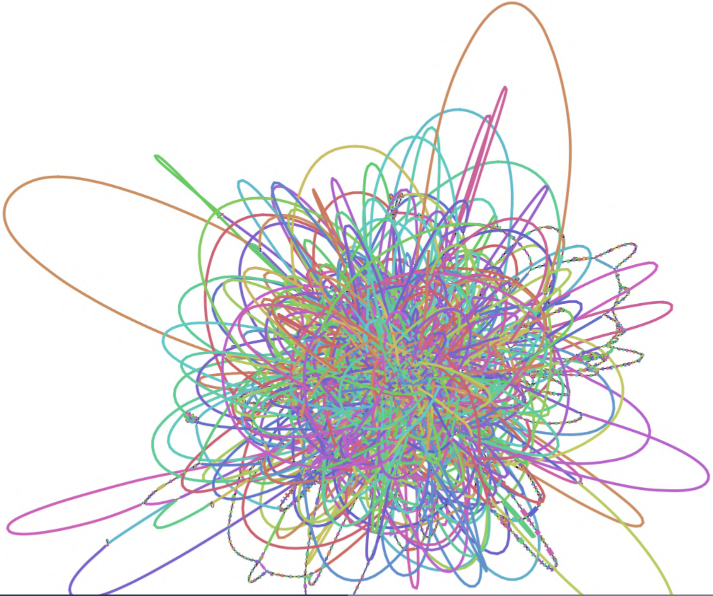
Reads longer than the repeat "solve" it



The graph becomes trivial to traverse

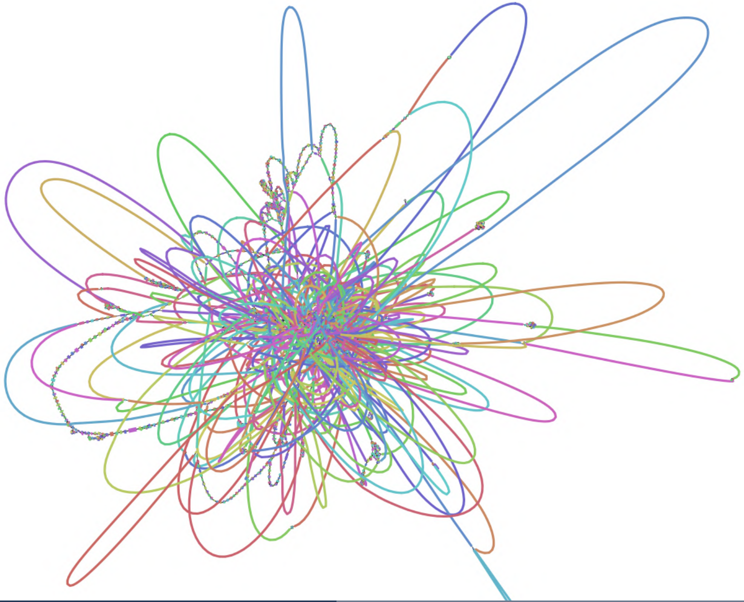
- Read length matters

Read size=21



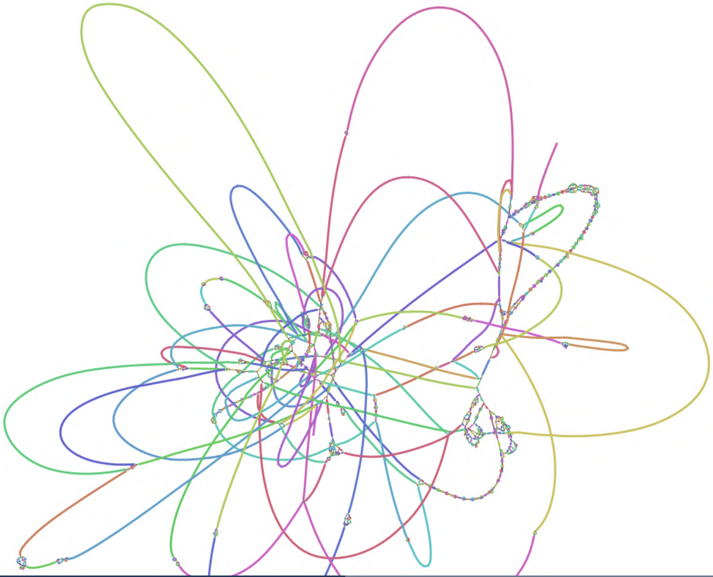
- Read length matters

Read size=31



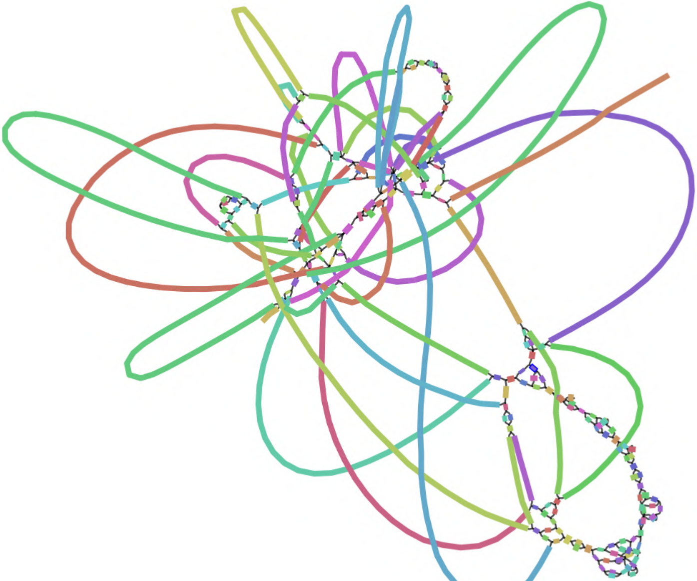
Read length matters

Read size=63



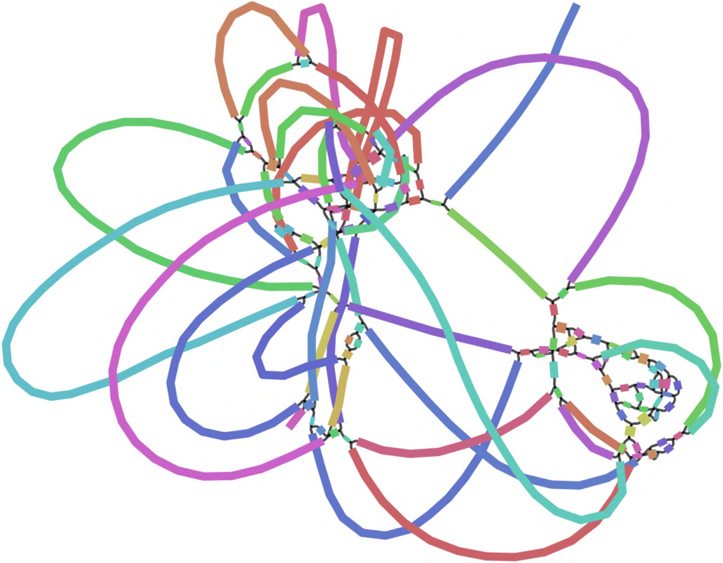
- Read length matters

Read size=255



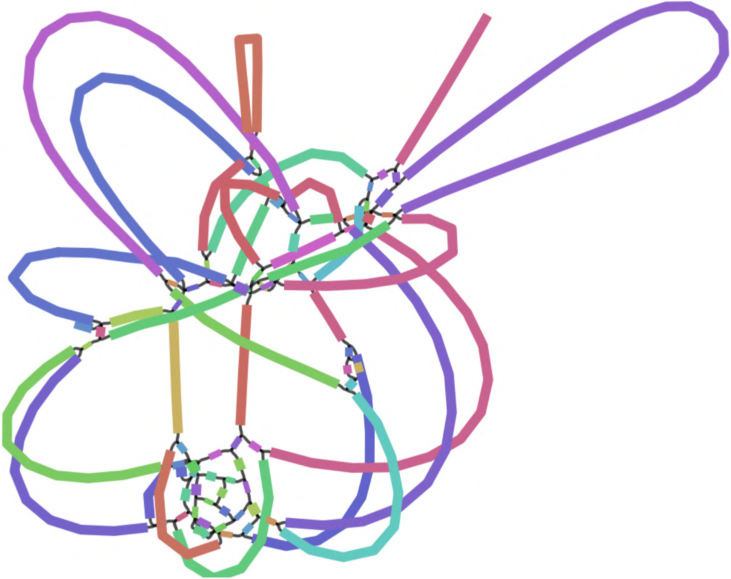
- Read length matters

Read size=500



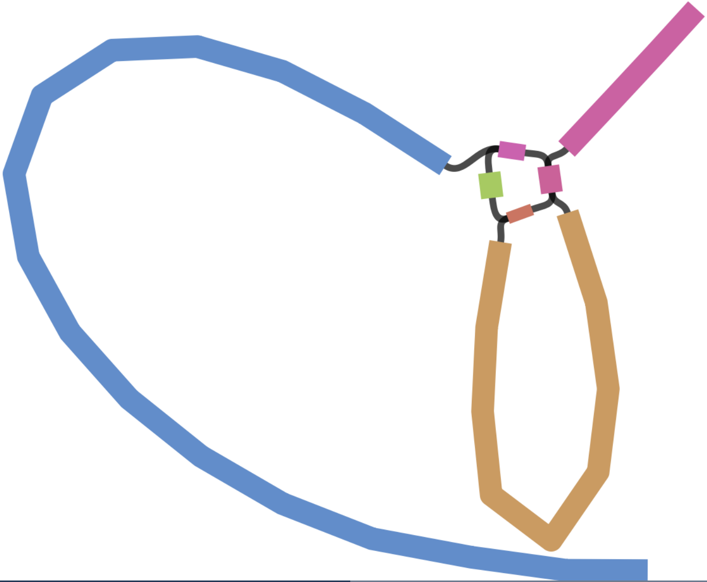
- Read length matters

Read size=1000

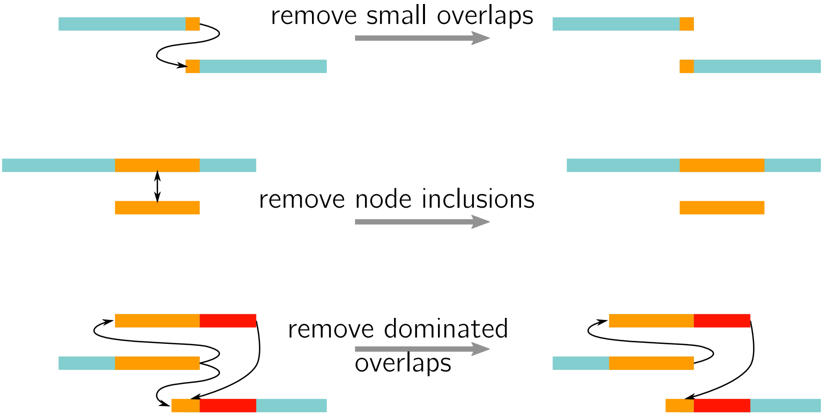


- Read length matters

Read size=2000



Overlap graph simplifications



remove small overlaps

remove node inclusions

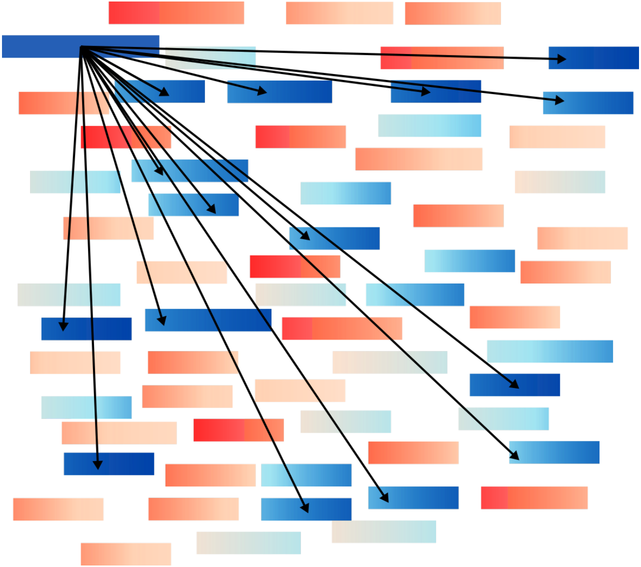
remove dominated overlaps

First (and most important) checkpoint

- Assembly orders reads using overlaps; longer overlaps are **generally** better.
- Multiple possible overlaps necessitate graphs for structuring information.
- Repeats longer than reads result in fragmented assembly (contigs).

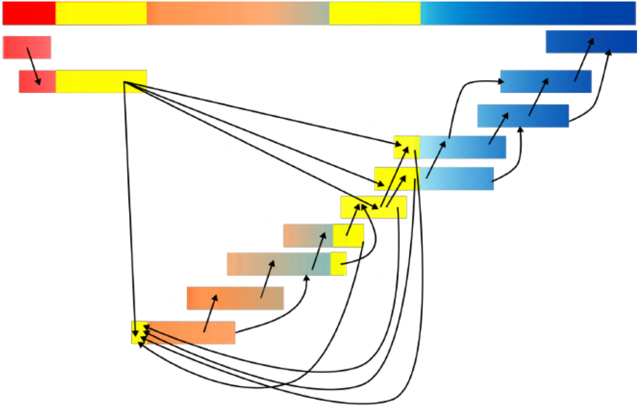
•Compute overlaps

Detecting overlaps means a lot of comparisons



•Compute overlaps

Even considering only the long overlaps means a lot of comparisons



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...

We have to be efficient and focus on "relevant" overlaps

Overlap graph burden: number of overlaps

For each base of the genome:

Read depth	Overlaps depth
10	100
20	400
50	2,500
100	10,000

The amount of overlaps is not linear

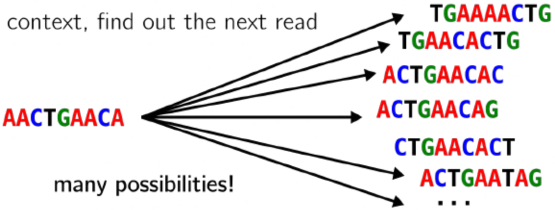
Linear: 2X data 2X time

Quadratic: 2X data 4X time

Another idea

what we have done:

from the context, find out the next read



AACTGAACA

TGAAAACTG

TGAACACTG

ACTGAACAC

ACTGAACAG

CTGAACACT

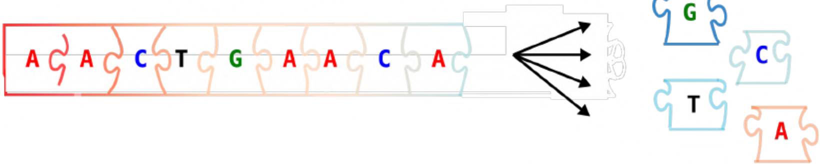
ACTGAATAG

...

many possibilities!

NEW SOLUTION !

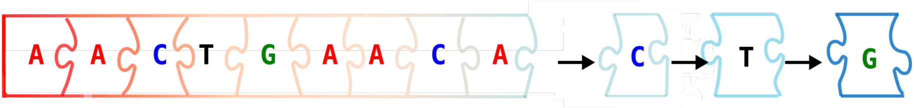
instead, from the context, find out the next nucleotide



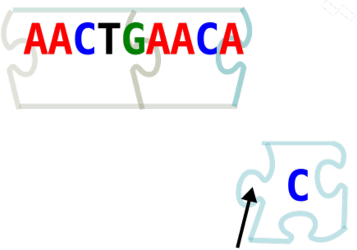
only 4 possibilities!

- Another idea

add the nucleotides one by one

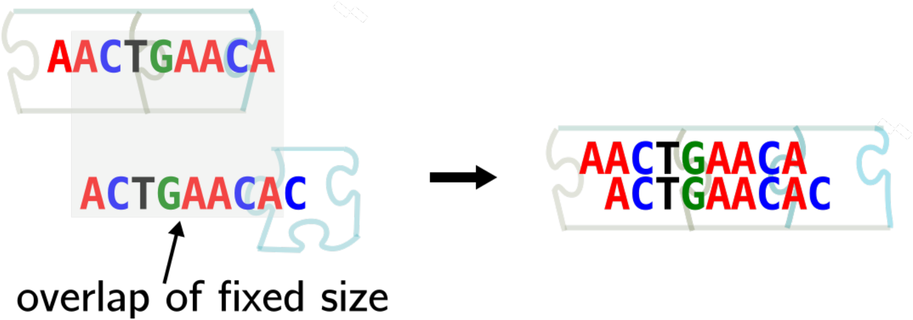


- Context



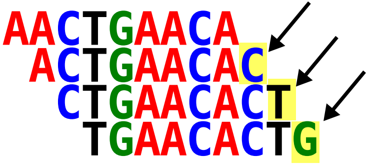
context to join the next base?

Context



AACTGAACA
ACTGAACAC
overlap of fixed size
AACTGAACA
ACTGAACAC

Assembly



```
AACTGAACA
 ACTGAACAC
  CTGAACACT
   TGAACACTG
```

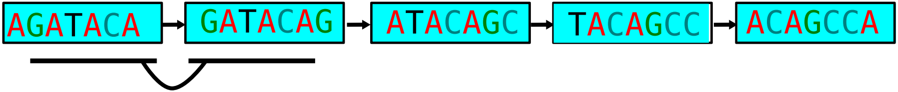
- The de Bruijn graph

Read

AGATACAGCCA

De Bruijn graph

Kmer=node

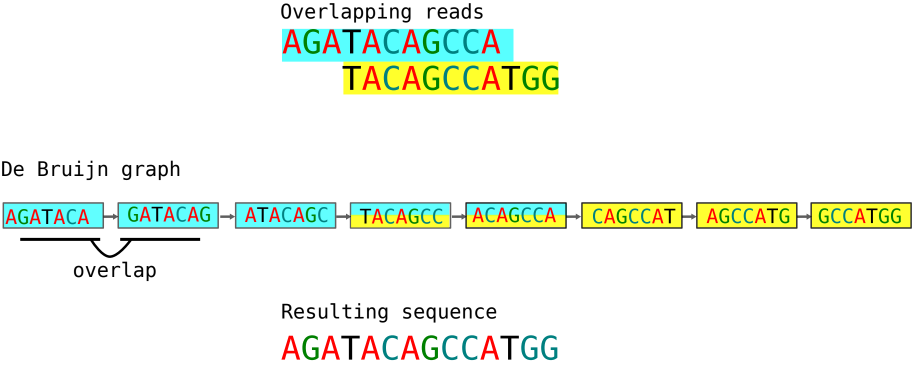

AGATACA → GATACAG → ATACAGC → TACAGCC → ACAGCCA

k-1 overlap=edge

AGATACA + G + C + C + A

=AGATACAGCCA

de Bruijn graph assembly



Overlapping reads

AGATACAGCCA

TACAGCCATGG

De Bruijn graph

AGATACA → GATACAG → ATACAGC → TACAGCC → ACAGCCA → CAGCCAT → AGCCATG → GCCATGG

overlap

Resulting sequence

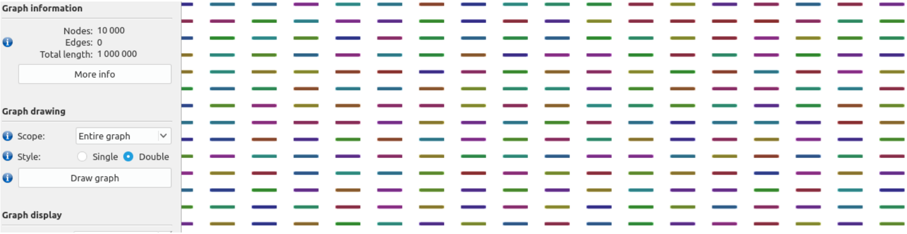
AGATACAGCCATGG

Why bother with k-mers?

in my graph, k-mer size = read size

Why bother with k-mers?

in my graph, k-mer size = read size



Graph information

Nodes: 10 000
Edges: 0
Total length: 1 000 000

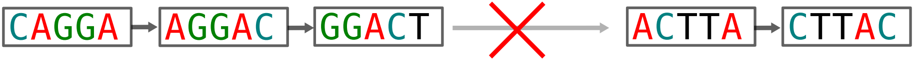
Graph drawing

Scope: Entire graph

Style: ○ Single ● Double

Graph display

- de Bruijn graphs limitation 1: Fixed overlaps



GACTT missing
not k-1 overlap

GGACT and ACTTA overlap is only of size 3 !

Exercise 1: de Bruijn graph time!

Reads

GCCATGGGTTT

TACAGCCATGG

AGCCATGGGTT

GCCATGGGTTT

AGCCATGGGTT

ACAGCCATGGG

GATACAGCCAT

ATACAGCCATG

CATGGGTTTAA

CAGCCATGGGT

GATACAGCCAT

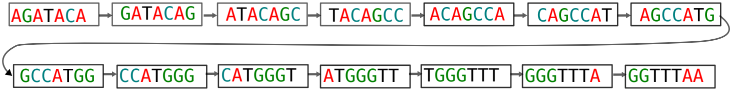
Hint: Use 7-mers

Exercise 1: Solution

read overlaps

```
AGATACAGCCA
 GATACAGCCAT
 GATACAGCCAT
  ATACAGCCATG
   TACAGCCATGG
    ACAGCCATGGG
    ACAGCCATGGG
     CAGCCATGGGT
      AGCCATGGGTT
       GCCATGGGTTT
       GCCATGGGTTT
        CCATGGGTTTA
         CATGGGTTTAA
```

de Bruijn graph



AGATACA → GATACAG → ATACAGC → TACAGCC → ACAGCCA → CAGCCAT → AGCCATG → GCCATGG → CCATGGG → CATGGGT → ATGGGTT → TGGGTTT → GGGTTTA → GGTTTAA

resulting sequence

AGATACAGCCATGGGTTTAA

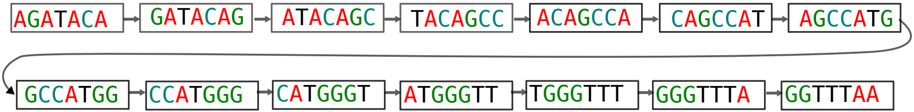
de Bruijn graphs abstract redundancy

read overlaps

```
AGATACAGCCA
 GATACAGCCAT
 GATACAGCCAT
  ATACAGCCATG
   TACAGCCATGG
    ACAGCCATGGG
    ACAGCCATGGG
     CAGCCATGGGT
      AGCCATGGGTT
       GCCATGGGTTT
       GCCATGGGTTT
        CCATGGGTTTA
         CATGGGTTTAA
```

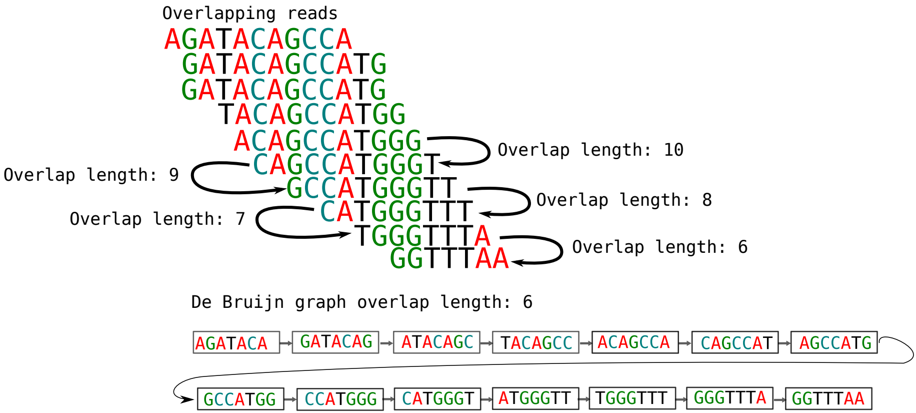
65 non distinct 7-mers in reads

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



AGATACA → GATACAG → ATACAGC → TACAGCC → ACAGCCA → CAGCCAT → AGCCATG → GCCATGG → CCATGGG → CATGGGT → ATGGGTT → TGGGTTT → GGGTTTA → GGTTTAA

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



Overlapping reads

AGATACAGCCA
GATACAGCCATG
GATACAGCCATG
TACAGCCATGG
ACAGCCATGGG
CAGCCATGGGT
GCCATGGGTT
CATGGGTTT
TGGGTTTA
GGTTTAA

Overlap length: 10

Overlap length: 9

Overlap length: 8

Overlap length: 7

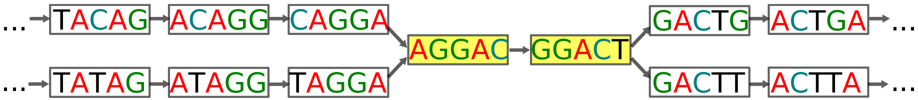
Overlap length: 6

De Bruijn graph overlap length: 6

AGATACA → GATACAG → ATACAGC → TACAGCC → ACAGCCA → CAGCCAT → AGCCATG → GCCATGG → CCATGGG → CATGGGT → ATGGGTT → TGGGTTT → GGGTTTA → GGTTTAA

- Repeats in a de Bruijn graphs

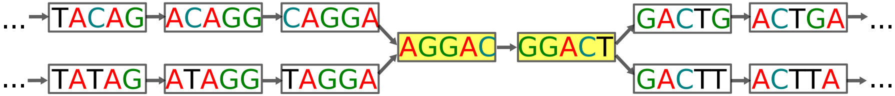
...TACAGGACTTA... ...TATAGGACTGA...



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

- de Bruijn graphs limitation 2: Repeats

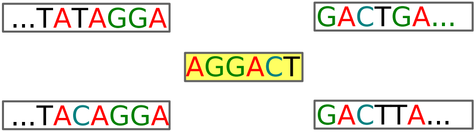
...TACAGGACTTA... ...TATAGGACTGA...



...TACAG ACAGG CAGGA AGGAC GGACT GACTG ACTGA ...

...TATAG ATAGG TAGGA GACTT ACTTA ...

genome pieces



...TATAGGA

GACTGA...

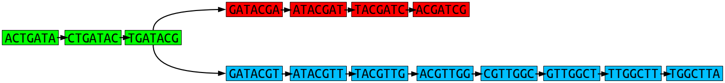
AGGACT

...TACAGGA

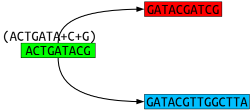
GACTTA...

On the representation of de Bruijn graphs

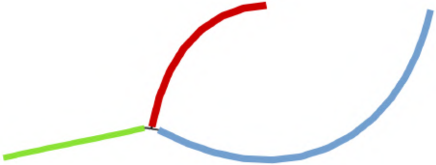
De Bruijn graph:



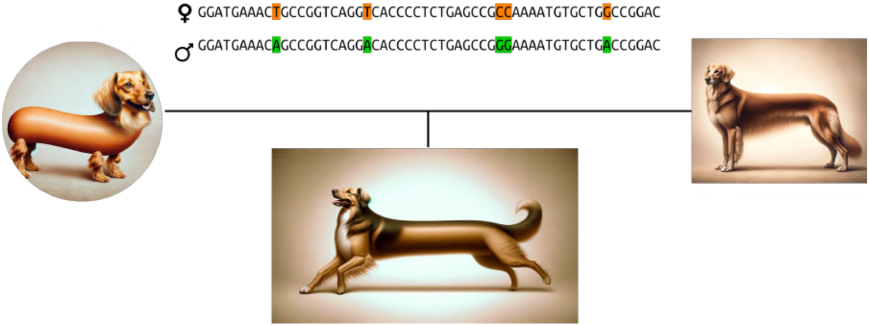
Compacted De Bruijn graph:



Graphical representation
(.gfa plot using Bandage):



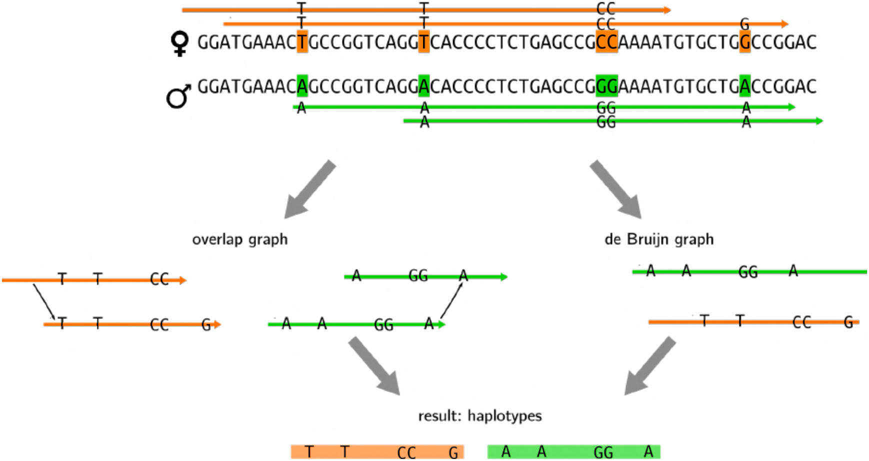
- The boy is diploid!



♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

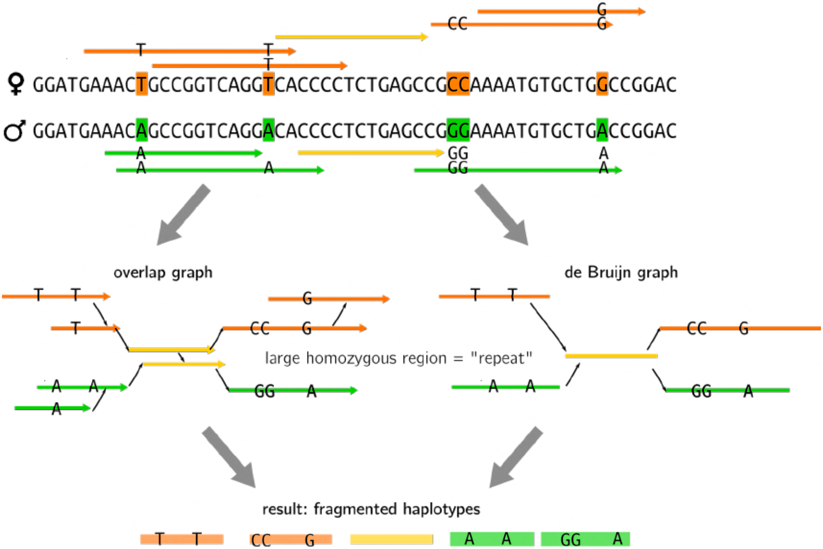
♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC

Ploidy and very long reads



Haplotypes can be "phased"

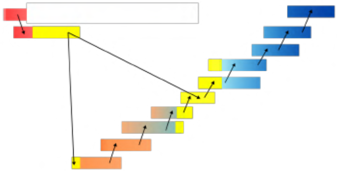
- Homozygous vs heterozygous regions


G
CC G
T T
T
♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC
♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC
A GG A
A A GG A
overlap graph
de Bruijn graph
T T
T
G
CC G
large homozygous region = "repeat"
A A
A
GG A
T T
CC G
A A
GG A
result: fragmented haplotypes
T T
CC G
A A
GG A

Assembly concession: assembly can be fragmented due to ploidy

Method checkpoint: de Bruijn graph versus overlap graph

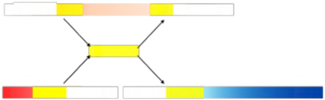
Overlap graph



quadratic growth with coverage

issue with repeats larger than reads

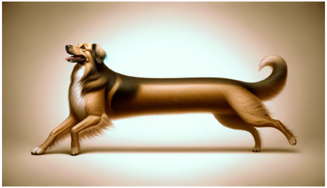
(compacted) de Bruijn graph

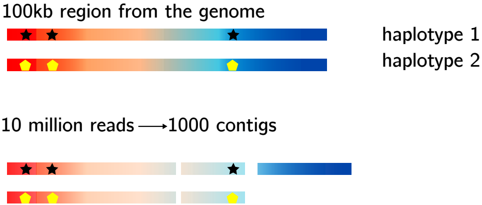


abstracts coverage

issue with repeats larger than *k*

Data checkpoint: results for the *long, perfect boy*

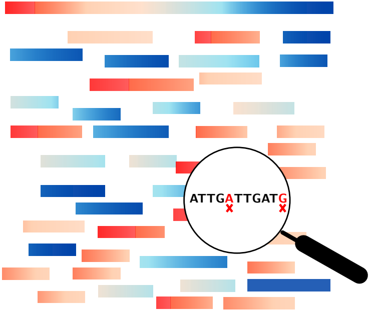




- Contigs can reach the chromosome's order of magnitude in length (megabases)
- Breaks due to large repeats
- Haplotypes can be partially reconstructed

•Second experiment: *noisy, super long boy*'s genome

100kb region from the genome
(only for the record, we actually don't have it)



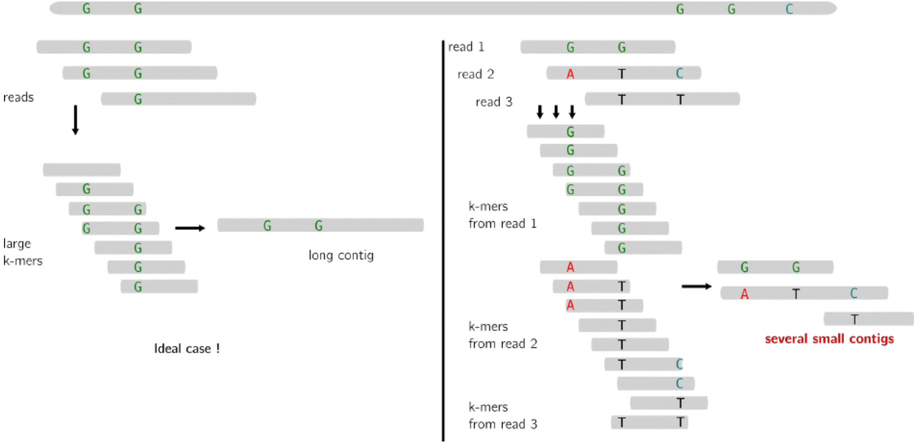
Genome size
1 billion bases

Reads
1 million
mean size 100kb
sequencing errors: 5-10%

- de Bruijn graph or overlap graph?

Sequencing errors and *k*-mers

Issue with large *k*-mers



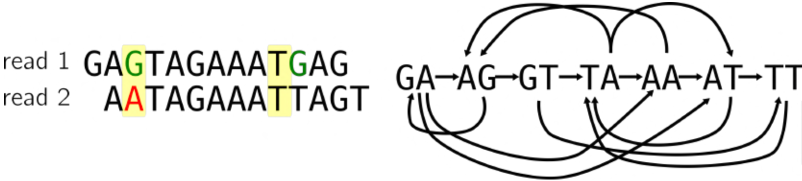
- large *k*-mers won't connect
- many spurious *k*-mers

Sequencing errors and *k*-mers

Let's try small *k*-mers

Sequencing errors and *k*-mers

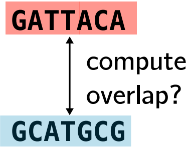
Let's try small *k*-mers



- the graph becomes too complicated: everything is a "repeat"
- we lose the advantage of having **long** reads

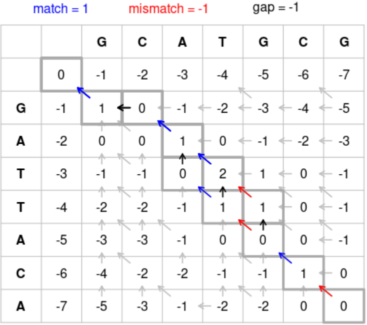
→ **de Bruijn graph out!**

Overlap graph: inexact matches


GATTACA

compute overlap?

GCATGCG

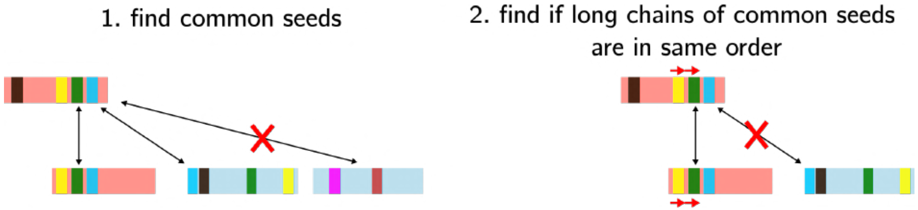


		G	C	A	T	G	C	G
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

Quadratic alignment for each pair of reads

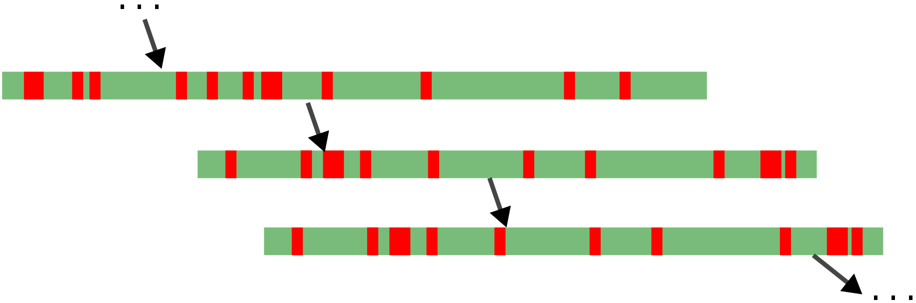
Quadratic number of comparisons to perform . . .

Overlap graph: drop alignment



Procedure called *anchor chaining*.

- How to get accurate contigs from noisy reads?

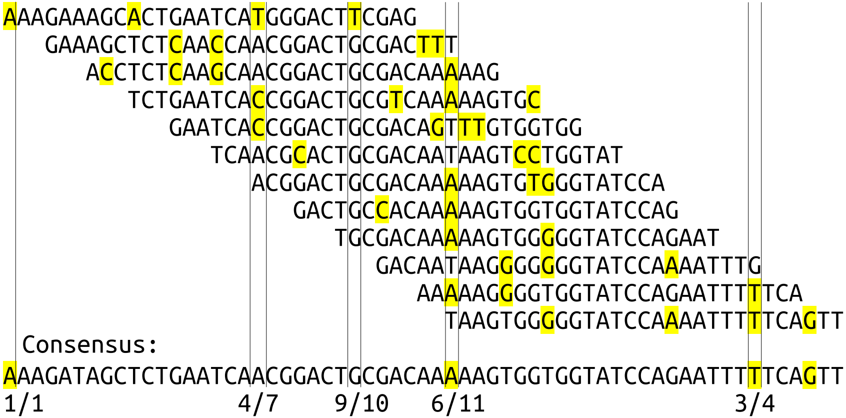


Using coverage to remove noise: consensus

Genome:

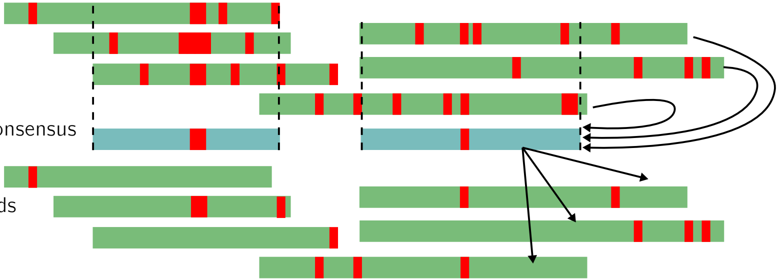
TAAGAAAGCTCTGAATCAACGGACTGCGACAATAAGTGGTGGTATCCAGAATTTGTCACTT

Reads:



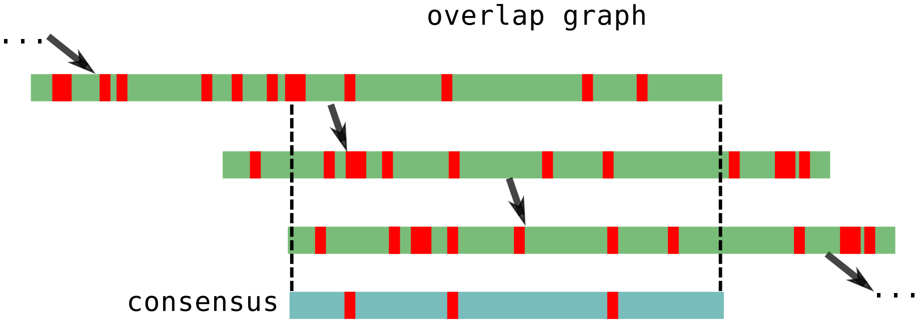
```
AAAGAAAGCACTGAATCATGGGACTTCGAG
   GAAAGCTCTCAACCAACGGACTGCGACTTTT
      ACCTCTCAAGCAACGGACTGCGACAAAAAG
         TCTGAATCACCGGACTGCGTCAAAAAGTGC
            GAATCACCGGACTGCGACAGTTTGTGGTGG
               TCAACGCACTGCGACAATAAGTCCTGGTAT
                  ACGGACTGCGACAAAAAGTGTGGGTATCCA
                     GACTGCCACAAAAAGTGGTGGTATCCAG
                        TGCGACAAAAAGTGGGGGTATCCAGAAT
                           GACAATAAGGGGGGGTATCCAAAATTTG
                              AAAAAGGGGTGGTATCCAGAATTTTTCA
                                TAAGTGGGGGTATCCAAAATTTTTCAGTT
 Consensus:
AAAGATAGCTCTGAATCAACGGACTGCGACAAAAAGTGGTGGTATCCAGAATTTTTCAGTT
1/1              4/7    9/10   6/11                  3/4
```

- Consensus before assembly: correction

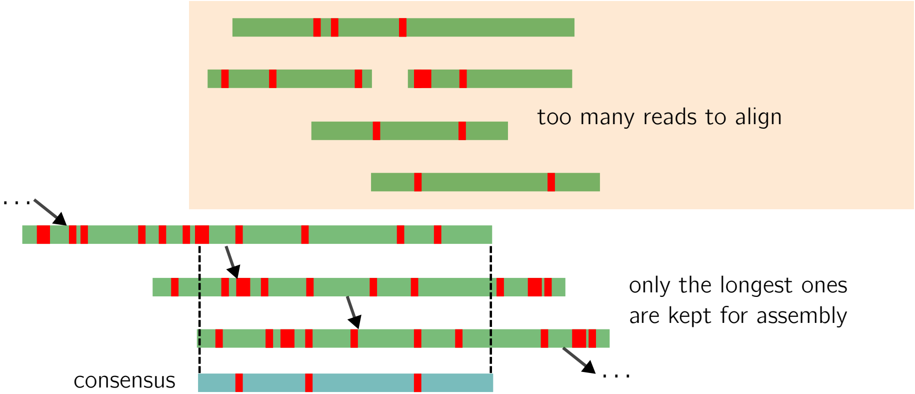


1. align reads
2. compute consensus
3. correct reads

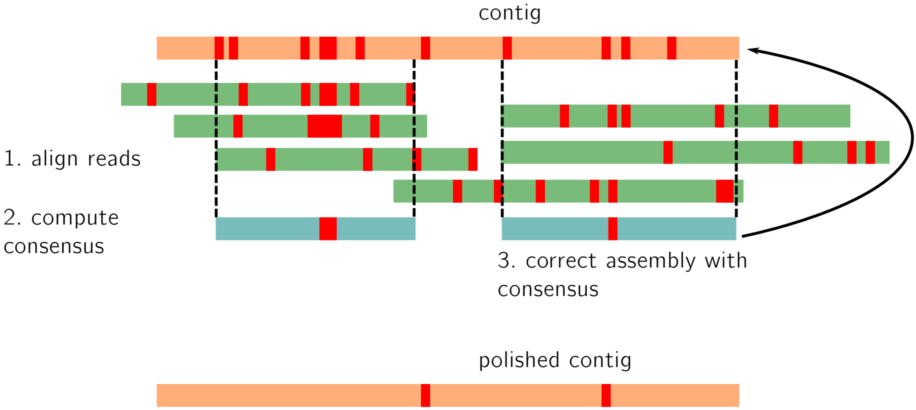
- Consensus during assembly (hence the OL**C**)



- Consensus during assembly. Yes but:



Consensus after assembly: polishing

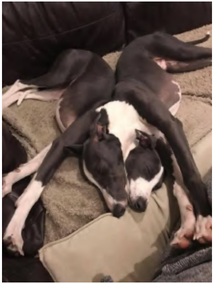


Correction/Consensus during assembly/Polishing

- Correction ×
  - Redundancy: 100X depth → 100X more bases to correct
- Consensus during assembly ≈
  - Do not use all reads
- Polishing ✓
  - Correct each base of the genome once
  - Use all reads

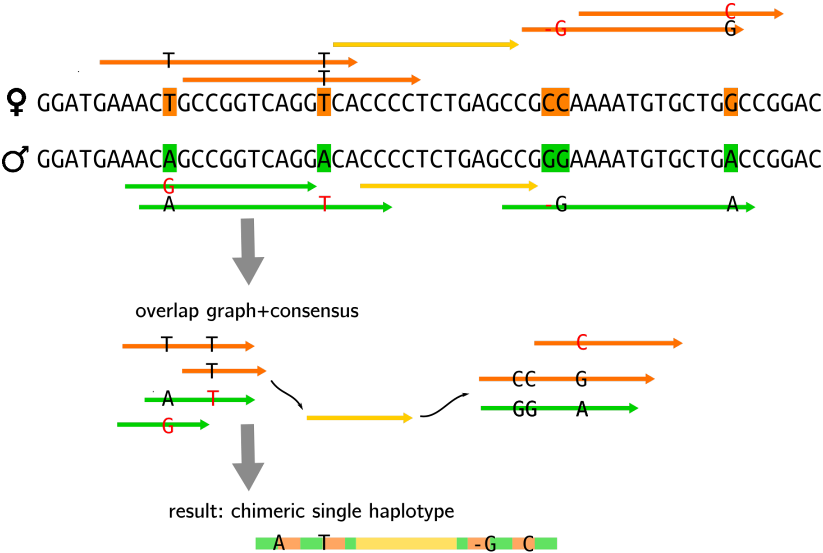
- Consensus destroys heterozygosity

```
reads      AATTGATCCGATACCC-GTAA-A
           AATTGGGCCGATACCC-GTAA-AG
           -ATTGATCCGA-ACCCCGTAA-A
           AATTGATCCGATACCC-GTAA-A
               GCTCCGAGACCA-GTCA-ATTG
               GCTCC-AGACCA-GTCA-ATTT
                  CCGAGACCA-GTCG-ATTGCAAA-
                  CCGAGACCA-GT-A-ATTGCGAAC
                  CCGACACCA-GTGAAATTGCAAAC
           ------------------------------
consensus  AATTGATCCGAGACCA-GTCA-ATTGCAAAC
```



→ a mix between the two alleles

Consensus destroys heterozygosity


C
-G G
T T
T
♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC
♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC
G
A T -G A
T T
T
A T
G
C
CC G
GG A
A T -G C

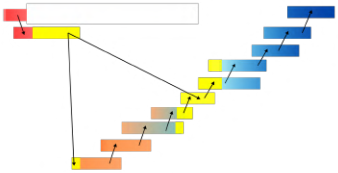
overlap graph+consensus

result: chimeric single haplotype

Assembly concession: "haploid" assembly due to errors

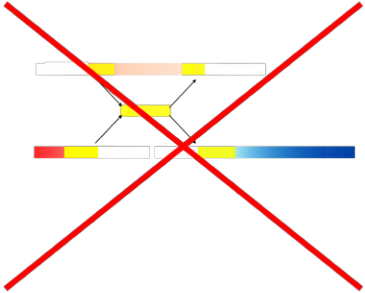
Method checkpoint: de Bruijn graph vs overlap graph

Overlap graph



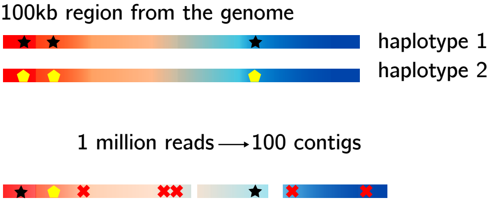
can deal with approximate overlaps
(seed and chain)
polishing

(compacted) de Bruijn graph



too many errors prohibit connectivity

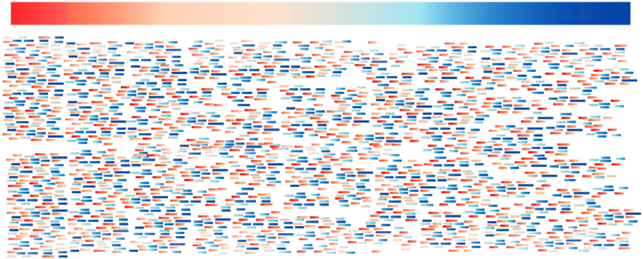
- Data checkpoint: results for the *noisy super long boy*



- Contigs can reach the chromosome's order of magnitude in length (megabases)
- Breaks due to very large repeats
- Contigs are chimeras of haplotypes

• Third experiment: *short boy*'s genome

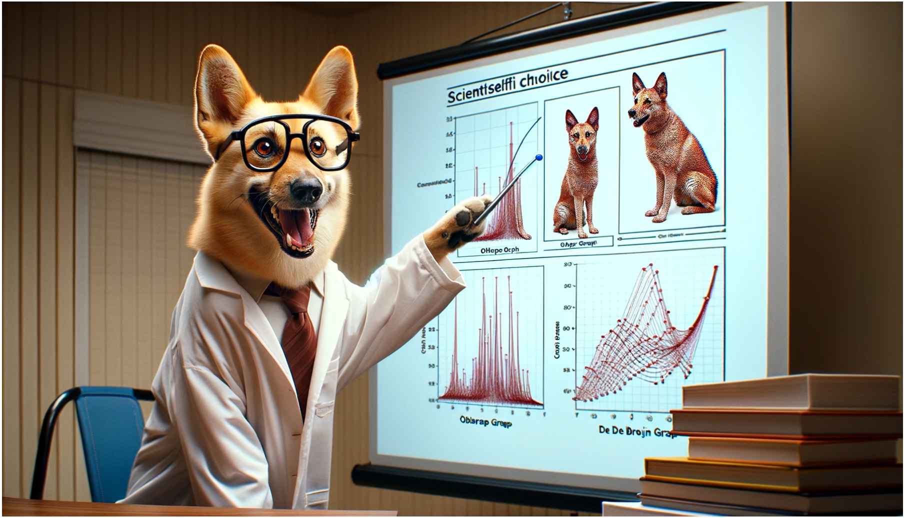
100kb region from the genome
(only for the record, we actually don't have it)



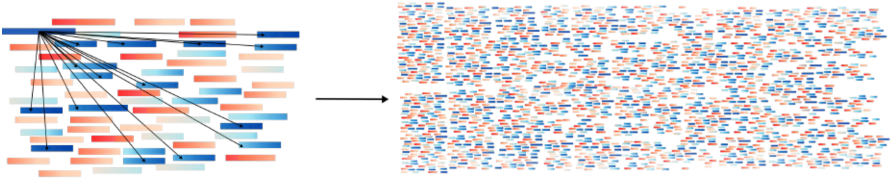
Genome size
1 billion bases

Reads
1 billion
size 100 bases
<1% error

de Bruijn graph or overlap graph?



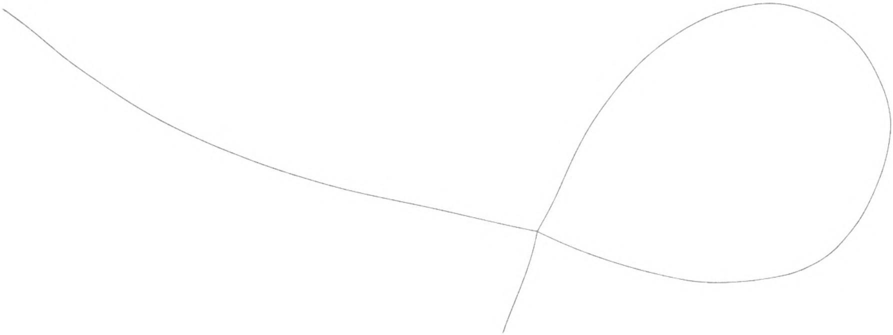
Scalability issue for the overlap graph



At equal coverage we got:
1000 × more reads → 1 million × more overlaps to check!
Overlap graph hardly scales to such a large number of reads/overlaps

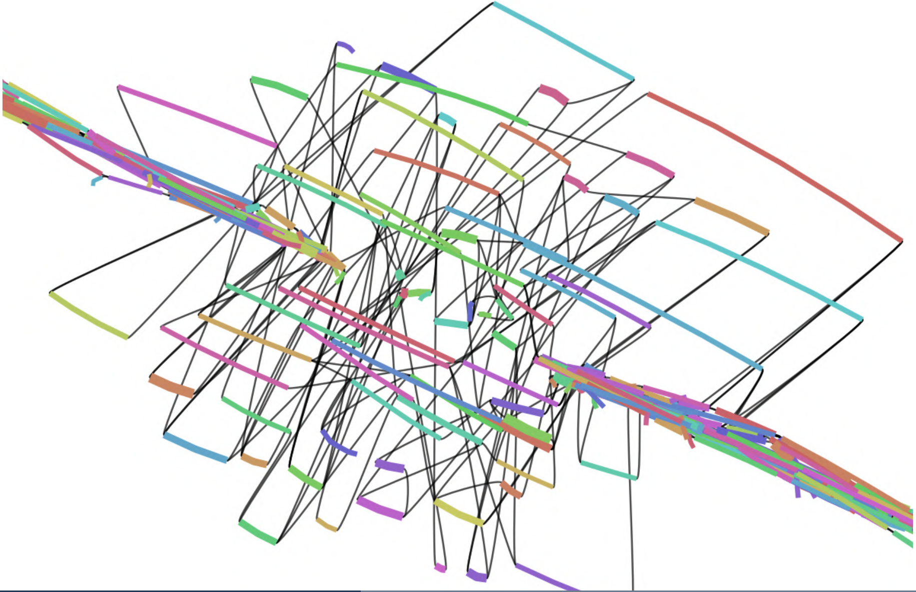
→ **Overlap graph out!**

- de Bruijn graph on a real dataset



- de Bruijn graph on a real dataset ZOOMED IN

- de Bruijn graph on a real dataset ZOOMED IN



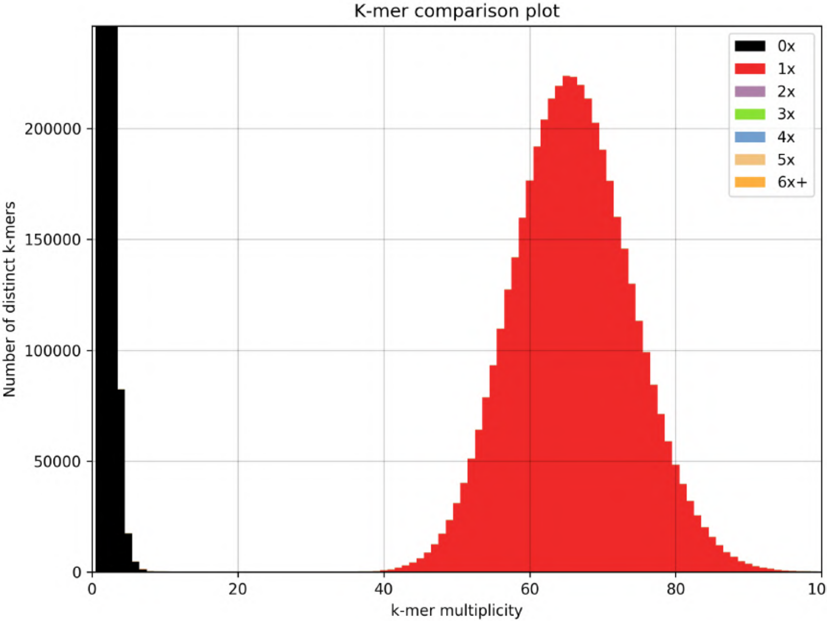
Erroneous *k*-mers vs genomic *k*-mers

```
Genome:
TAAGAAAGCTCTGAATCAACGGACTGCGACA

Reads:
TAAGAAAGCTCTGAATCA
 AAGAAAGCTCTAAATCAAC
  AGAAAGCTCTGAATCAACG
   GAAAGCTCTGAATCAACGGA
    AAAGCTCTGAATCAACGGAC
     AAGCTCTGAATCAACGGACT
      AGCTCTGAATCAACGGACTG
       GCTCTGAATCAACGGTCTGC
        CTCTGAATCAACGGACTGCG
         TCTGAATCAACGGACTGCGA
```

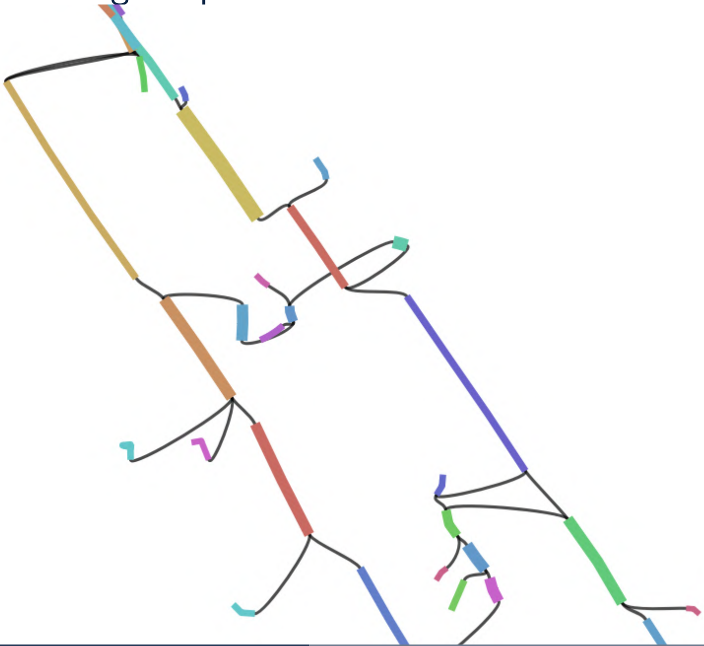
```
9 times   TCTGAAT
1 time    TCTAAAT

6 times          CAACGGA
1 time           CAACGGT
```

Erroneous *k*-mers are seen less than genomic ones

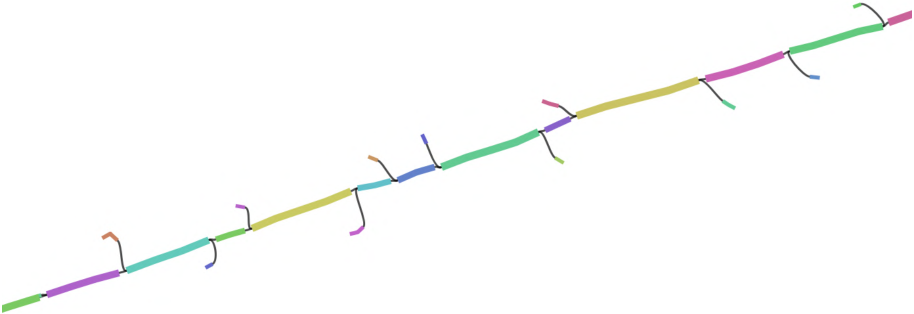
- *K*-mer histogram



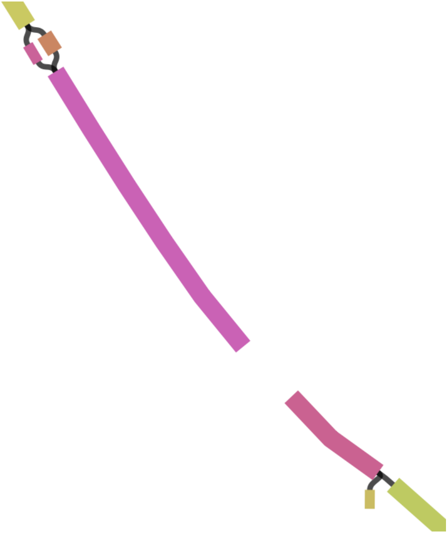
- Removing unique *k*-mers



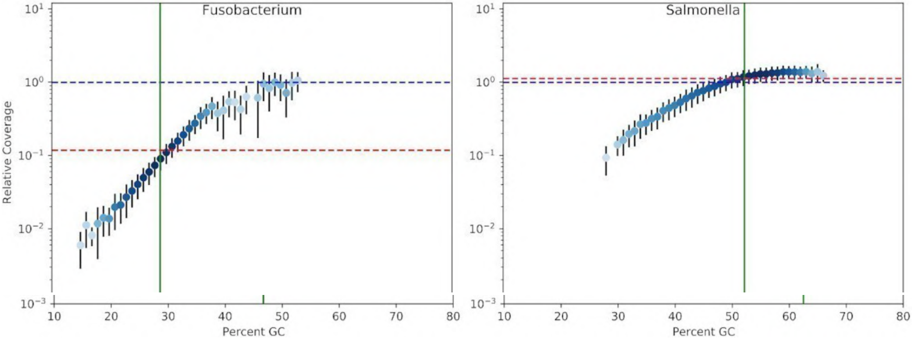
- Removing *k*-mers seen less than 3 times



- Removing *k*-mers seen less than 4 times



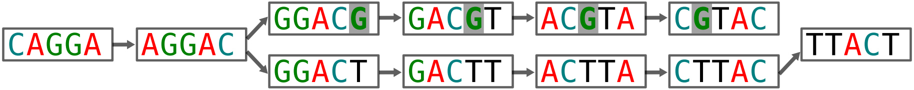
GC bias



Low GC region can be way less sequenced

- Errors in de Bruijn graphs

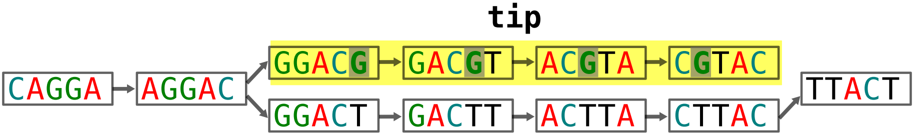
```
...TACAGGACTTACTGA...  genome

        CAGGACTTA
reads    AGGACGTAC ←—— sequencing error
         AGGACTTAC
          GGACTTACT
```



- Errors in de Bruijn graphs

...TACAGGACTTACTGA... genome

reads
CAGGACTTA
AGGACGTAC ← sequencing error
AGGACTTAC
GGACTTACT



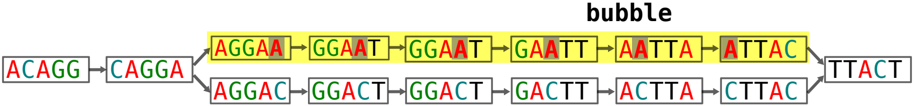
tip

CAGGA → AGGAC → GGACG → GACGT → ACGTA → CGTAC

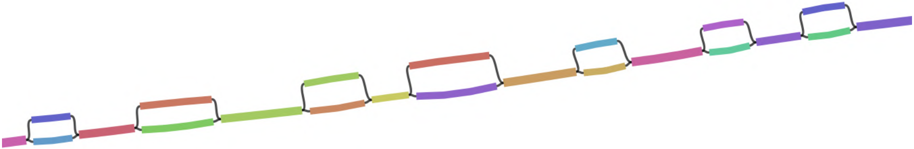
AGGAC → GGACT → GACTT → ACTTA → CTTAC → TTACT

Errors in de Bruijn graphs

```
...TACAGGACTTACTGA...  genome

        ACAGGACTTA
reads    CAGGAATTAC ←—— sequencing error
         CAGGACTTAC
          AGGACTTACT
```



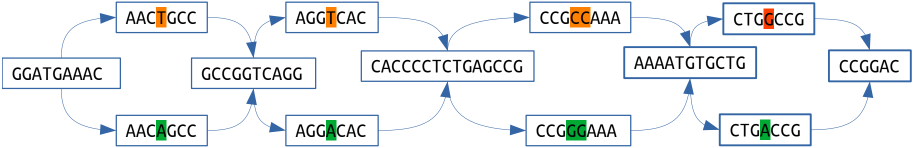
- de Bruijn graph on my diploid genome



- Ploidy and de Bruijn graph

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC



GGATGAAAC

AACTGCC

AACAGCC

GCCGGTCAGG

AGGTCAC

AGGACAC

CACCCCTCTGAGCCG

CCGCCAAA

CCGGGAAA

AAAATGTGCTG

CTGGCCG

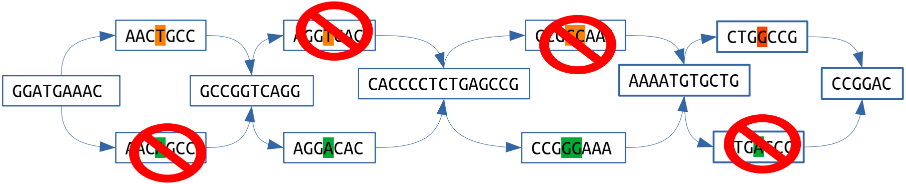
CTGACCG

CCGGAC

Bubble crushing

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC



Assembly:

GGATGAAACTGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGGCCGGAC

- Variants are not "lost"

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC

Assembly:

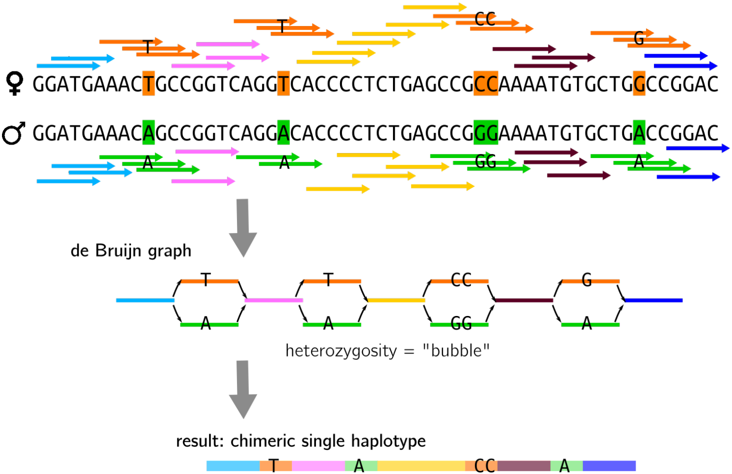
GGATGAAACTGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGGCCGGAC

Reads:

```
GATGAAACTG
 ATGAAACAGC
  TGAAACAGCCG
   GAAACTGCCGG
    AAACTGCCGGT
     AACAGCCGGTC
      ACAGCCGGTCA
       CTGCCGGTCAG
```

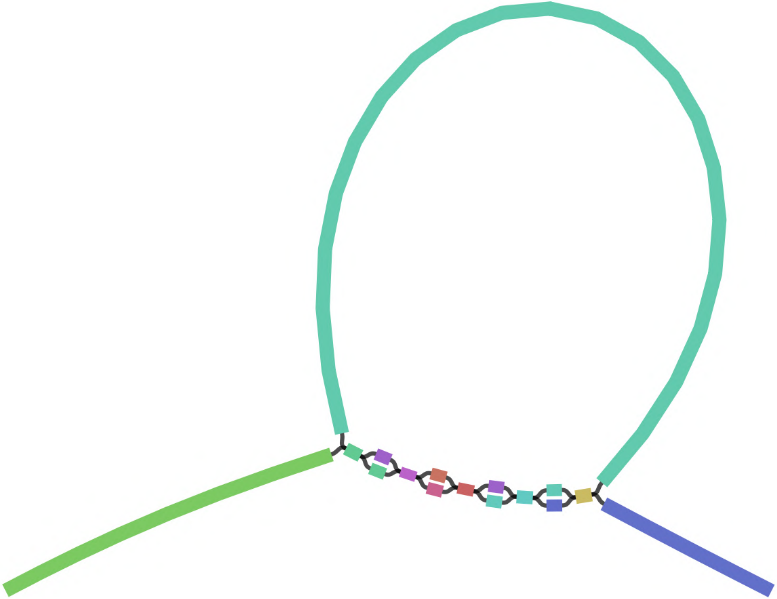
We can align the reads to the assembly and do variant calling

- Haploid assembly


T
T
CC
G
♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC
♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC
A
A
GG
A
de Bruijn graph
T
T
CC
G
A
A
GG
A
heterozygosity = "bubble"
result: chimeric single haplotype
T
A
CC
A

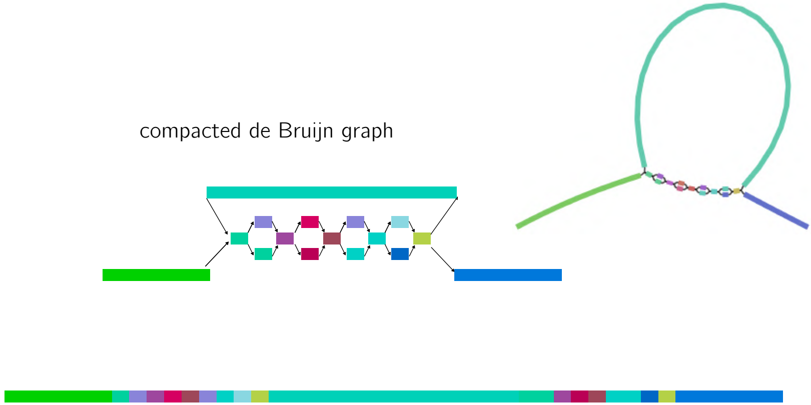
Assembly concession: haplotypes are collapsed when using short reads

- Paralog genes/repeats

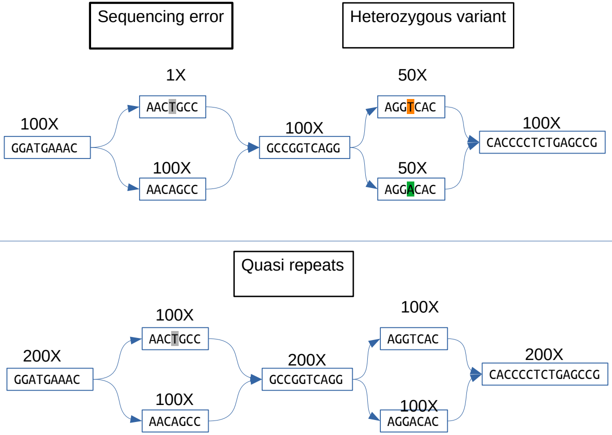


- Paralog genes/repeats

compacted de Bruijn graph



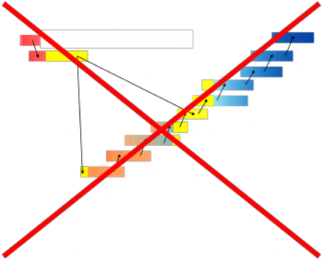
Paralog genes/repeats in graphs



Treating erroneous bubbles is simple, all the others are complex

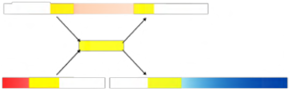
Method checkpoint: de Bruijn graph versus overlap graph

Overlap graph



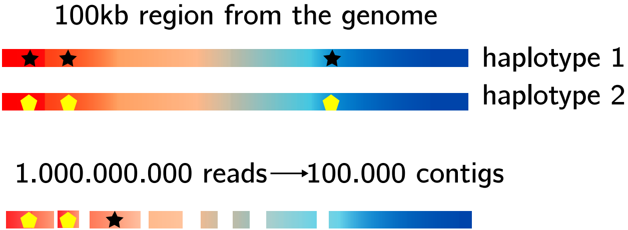
all v all comparisons are not scalable

(compacted) de Bruijn graph



k-mers abstract coverage
errors can be dealt with

Data checkpoint: *short boy* results


100kb region from the genome
haplotype 1
haplotype 2
1.000.000.000 reads ⟶ 100.000 contigs

- Very fragmented assembly of short contigs (mostly below 100kb)
- Very high base accuracy
- Contigs are chimeras of haplotypes
- Can miss low GC content

- Fourth experiment: *golden boy*'s genome

Billion $ project → cancelled

(Time accurate) recap

Sanger

No longer used for assembly (too expensive)

Illumina

De Bruijn graph assembly

Fragmented haploid assembly

Long reads: Oxford Nanopore or PacBio

Overlap graph assembly (+ polishing)

Contiguous haploid assembly

HiFi

Overlap graph or de Bruijn graph assembly

Contiguous diploid assembly

- Back to the present

Challenge 1: Scalability

- Human Genome project (2001)
- 1000 Genomes project (2015)
- 10k Genomes project (2016)
- 100k Genomes project (2018)
- 500K UK genomes (2023)

Many ambitious sequencing projects beyond human: Earth biogenome project, Vertebrate genome project . . .

History

How long to assemble a human genome?

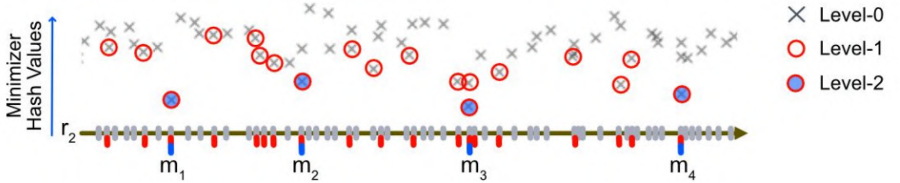
- Sanger: MANY CPU **years**
- Illumina (Overlap graph): 2 CPU **months**
- Illumina (De Bruijn graph): A CPU **day**
- Long reads (Alignment): 2 CPU **years**
- Long reads (Anchors chaining): 20 CPU **days**
- HiFi (Anchors chaining): 2 CPU **days**
- HiFi (De Bruijn graph): A CPU **hour**

Algorithms and data structures matter!

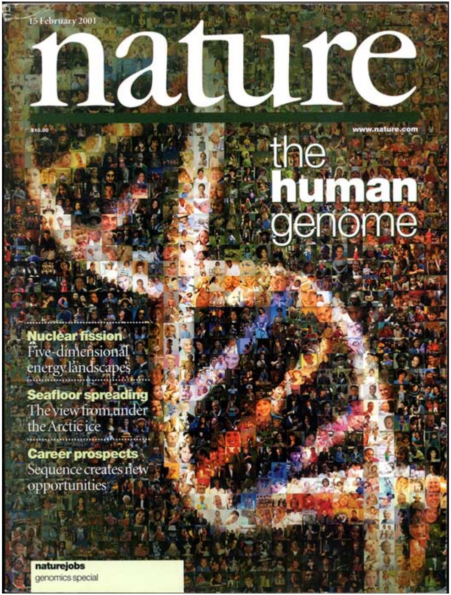
Also long and precise reads are easier to assemble

- Very fast genome assembly with HiFi

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



- Telomere to telomere assembly?



Earth's heart of iron begins to yield its secrets p. 18
Microglia in chronic pain recovery and relapse pp. 33 & 86
Particle acceleration in a nova explosion p. 77
Science
$15
1 APRIL 2022
SPECIAL ISSUE
science.org
AAAS
FILLING THE GAPS
Closing in on a complete human genome p. 42

Challenge 2: Telomere to telomere chromosomes

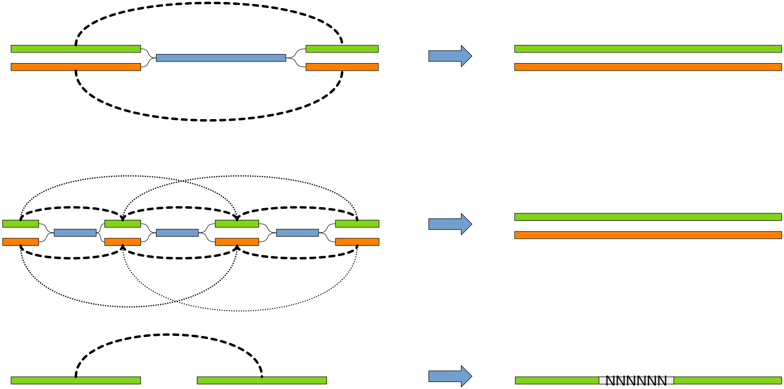
Main problems:

- Very large exact repeats
- Very similar sequences accross the genome
- Low complexity regions
- Mosaic repeats

Need long distance information **AND** high base accuracy

Scaffolding

Use long range information to order contigs into "scaffolds"



Reference-based Scaffolding/assembly

Pros

Do not need high coverage/ long distance information to get contigous assemblies

Cons

Need a related good quality reference
Bias toward reference sequence, for local and structural variants

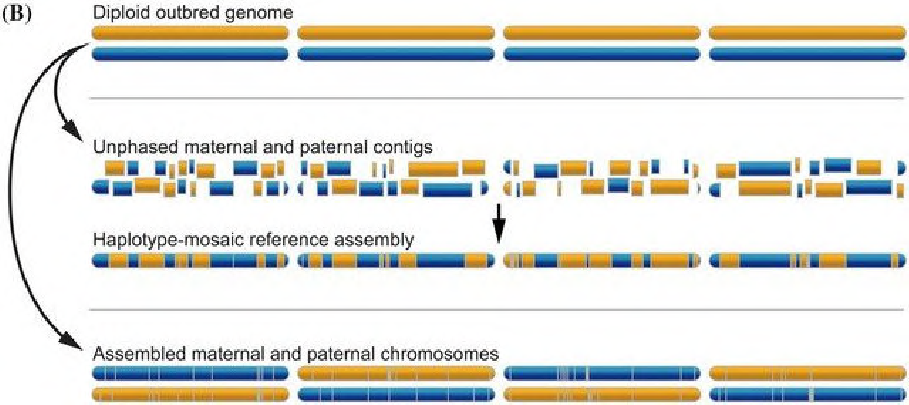
- Map the reads on a reference and compute a consensus (Medaka)
- Use a reference assembly as existing contigs (SPAdes)
- Use one (or several) related references genomes to order contigs (Ragout2)

Telomere-to-Telomere consortium

Has produced in 2022 a complete human genome with one contig per chromosome !

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

- Diploid assembly


(B)
Diploid outbred genome
Unphased maternal and paternal contigs
Haplotype-mosaic reference assembly
Assembled maternal and paternal chromosomes

Telomere-to-Telomere diploid human reference

T2T-YAO released in 2023 a complete human genome with one contig per chromosome !

- 92x PacBio HiFi
- 336x coverage of Oxford Nanopore (ultra long reads)
- 70x PacBio CLR
- 584x Arima Genomics HiC
- BioNano DLS
- Illumina HiSeq 150bp for the son and parents (with 278x and 116x coverage, respectively).

The human genome is not THAT hard

Hall of fame of largest assembled genomes of their time:

- Pine (20Gb)

The human genome is not THAT hard

Hall of fame of largest assembled genomes of their time:

- Pine (20Gb)
- Axolotl (32Gb)

The human genome is not THAT hard

Hall of fame of largest assembled genomes of their time:

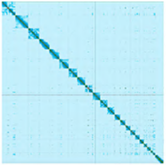
- Pine (20Gb)
- Axolotl (32Gb)
- Lungfish (43Gb)

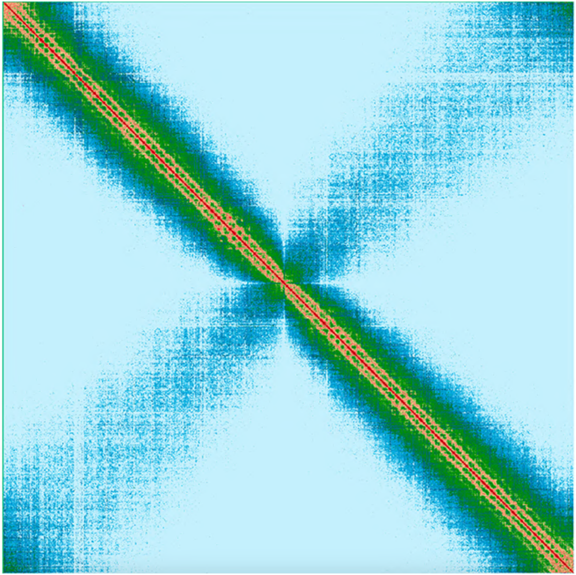
The human genome is not THAT hard

Hall of fame of largest assembled genomes of their time:

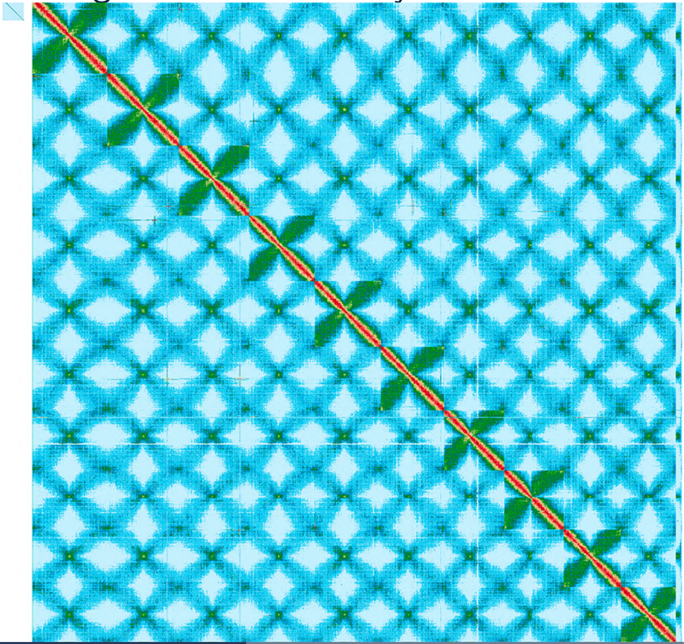
- Pine (20Gb)
- Axolotl (32Gb)
- Lungfish (43Gb)
- Mistletoe (90Gb)
- Metagenomes . . .

The human genome seems small

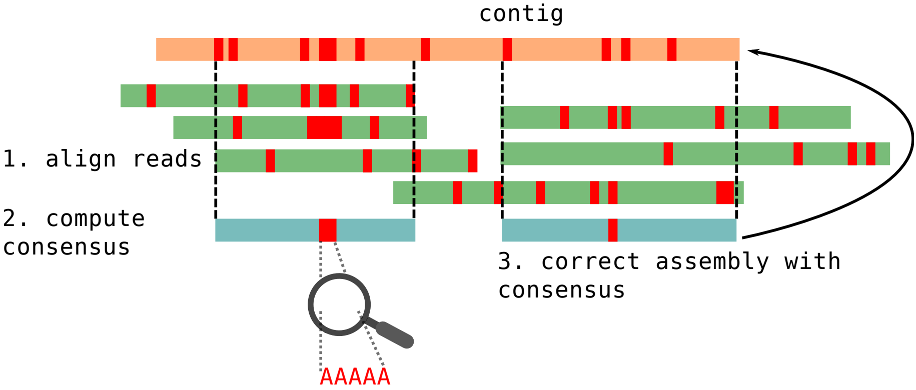




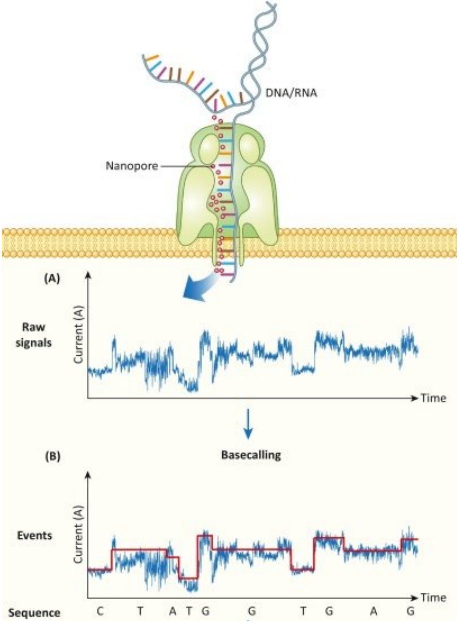
- The human genome seems really small



Challenge 3: Base level accuracy

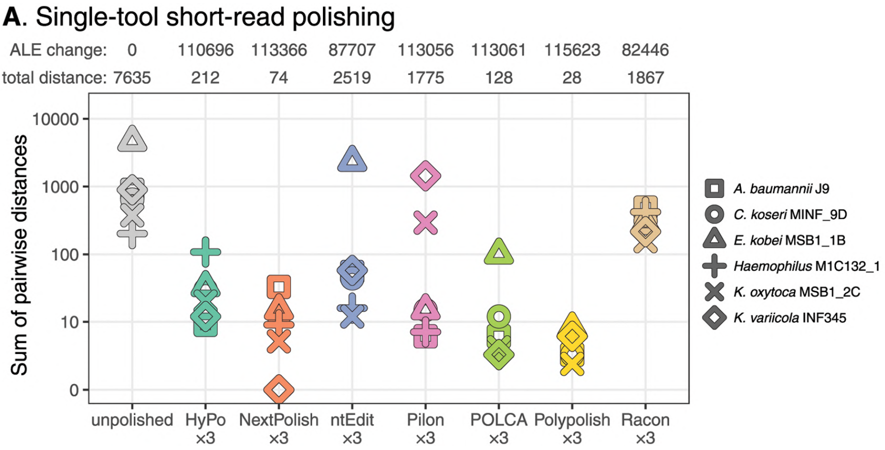


- Homopolymers are hard to read

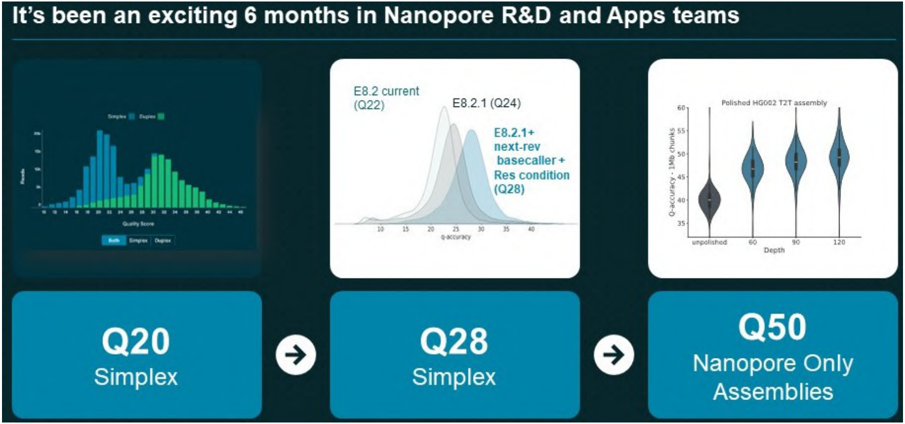


Systematic errors

Polishing with Illumina data can improve the final error rate



- Basecalling progress: Dorado years



It's been an exciting 6 months in Nanopore R&D and Apps teams

Q20 Simplex → Q28 Simplex → Q50 Nanopore Only Assemblies

Replication outside nanopore HQ

Post from Ryan Wick's bioinformatics blog (`rrwick.github.io/`) reports Q20 reads accuracy and Q60 assemblies on 9 bacterial assemblies

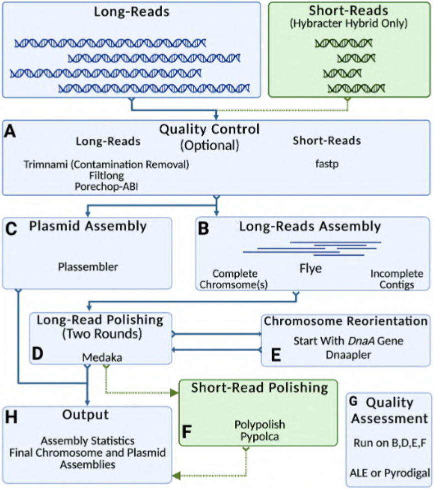
Average	**Read accuracy**	**Assembly accuracy**
mean	97.7%, Q16.4	4 errors, Q60.43
median	99.1%, Q20.5	2 errors, Q60.43
mode	99.4%, Q22.2	NA

HiFi-like Nanopore data ?

(Near) error-less very long reads we have several promising improvements ahead:

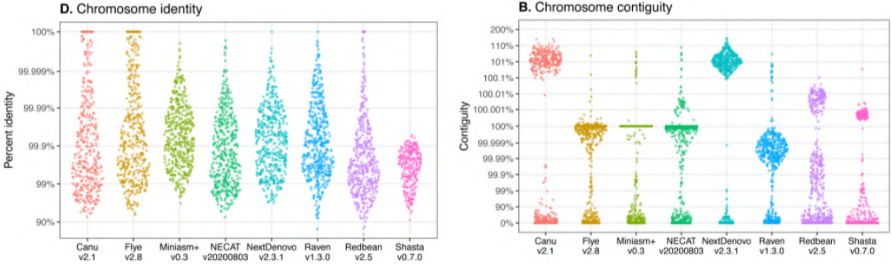
- Very fast assembly
- T2T chromosomes with less data
- Higher consensus accuracy
- Poplyploid assemblies

Challenge 4 : Assembly as a software



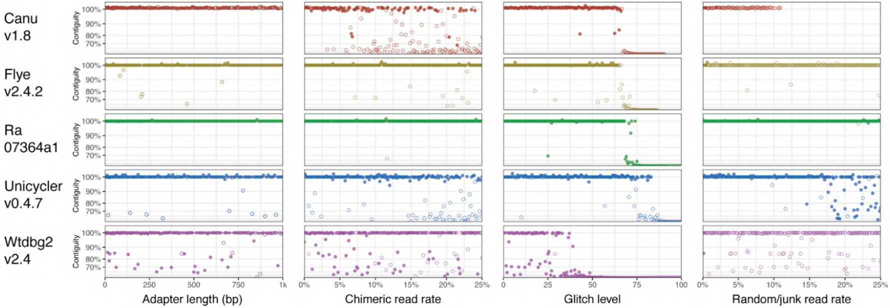
From Hybracter: Enabling Scalable, Automated, Complete and Accurate Bacterial Genome Assemblies

Assemblers behave differently



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

Software robustness



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

- 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(!)

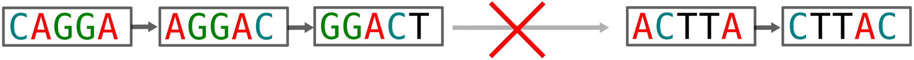
The (first) end

Advanced points

If we have time, we'll review everything (while doing this course, I doubt it . . .)
Else, pick one:

- Multiple k assembly in de Bruijn graphs
- HiFi de Bruijn Assembly
- An overlap graph limitation with noisy long reads (and current fixes)
- The repeat graph

Coming back to a de Bruijn graph limitation: fixed overlaps



CAGGA → AGGAC → GGACT → ACTTA → CTTAC

GACTT missing
not k-1 overlap

GGACT and ACTTA overlap is only of size 3 !

- A too small k is not a solution
- We would like larger k's but miss connections

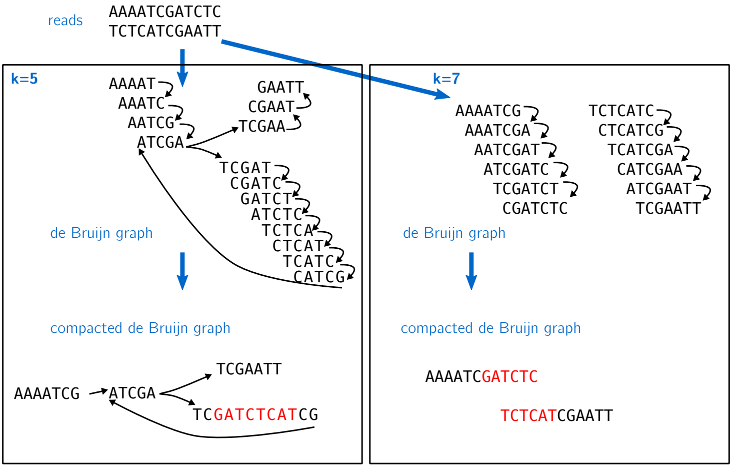
Multiple *k* assembly

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



reads

AAAATCGATCTC
TCTCATCGAATT

k=5

AAAAT
AAATC
AATCG
ATCGA
GAATT
CGAAT
TCGAA
TCGAT
CGATC
GATCT
ATCTC
TCTCA
CTCAT
TCATC
CATCG

de Bruijn graph

compacted de Bruijn graph

AAAATCG
ATCGA
TCGAATT
TCGATCTCATCG

k=7

AAAATCG
AAATCGA
AATCGAT
ATCGATC
TCGATCT
CGATCTC

TCTCATC
CTCATCG
TCATCGA
CATCGAA
ATCGAAT
TCGAATT

de Bruijn graph

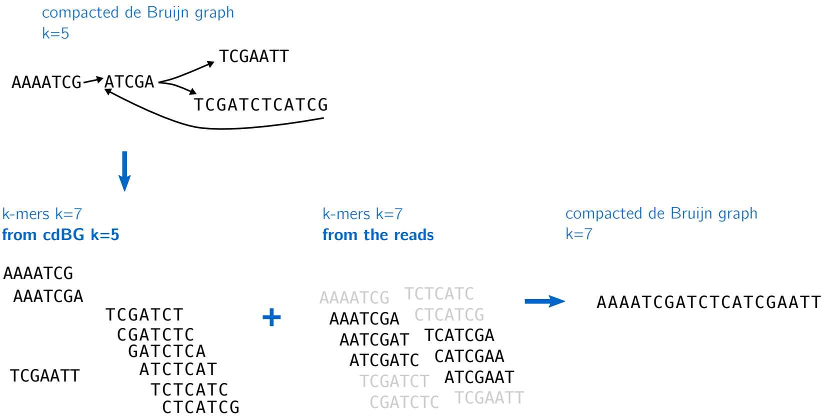
compacted de Bruijn graph

AAAATCGATCTC

TCTCATCGAATT

We are missing GATCTCA and ATCTCAT in the second graph.
But they are present in the first graph!

Multiple *k* assembly



compacted de Bruijn graph
k=5

k-mers k=7
from cdBG k=5

AAAATCG
AAATCGA
TCGATCT
CGATCTC
GATCTCA
ATCTCAT
TCTCATC
CTCATCG
TCGAATT

+

k-mers k=7
from the reads

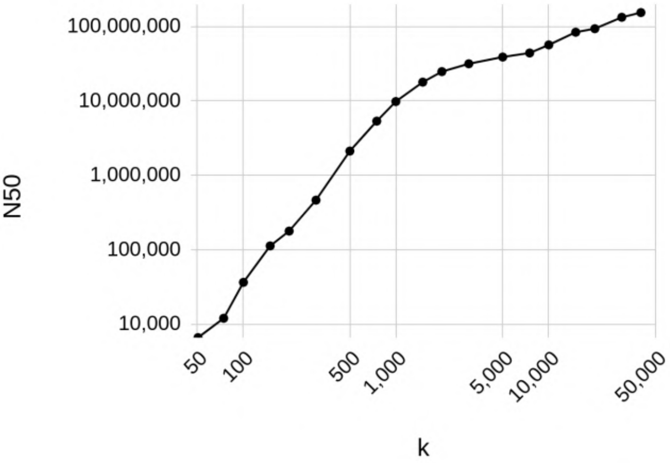
AAAATCG
AAATCGA
AATCGAT
ATCGATC
TCGATCT
CGATCTC
TCTCATC
CTCATCG
TCATCGA
CATCGAA
ATCGAAT
TCGAATT

compacted de Bruijn graph
k=7

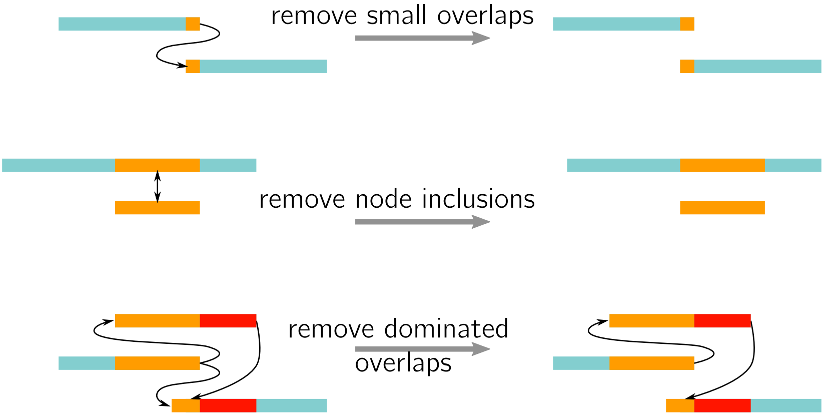
AAAATCGATCTCATCGAATT

HiFi de Bruijn graph Assembly

Using very large K (K=500 to K=5000) de Bruijn graphs to assemble



- Coming back to the overlap graph simplifications

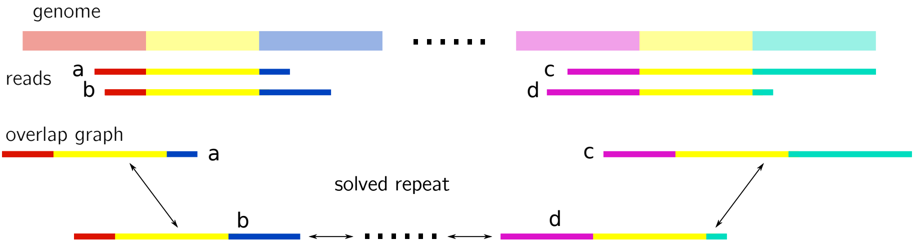


remove small overlaps

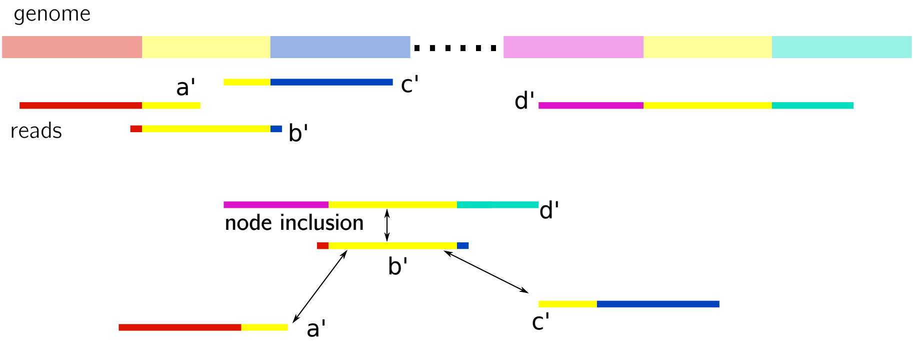
remove node inclusions

remove dominated overlaps

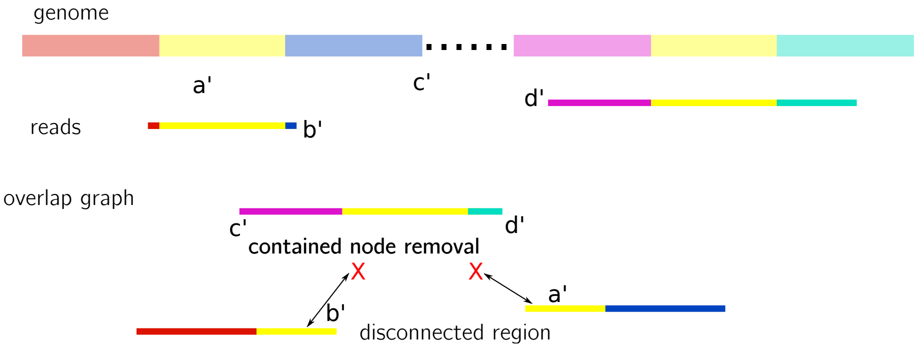
- An overlap graph limitation when using noisy reads



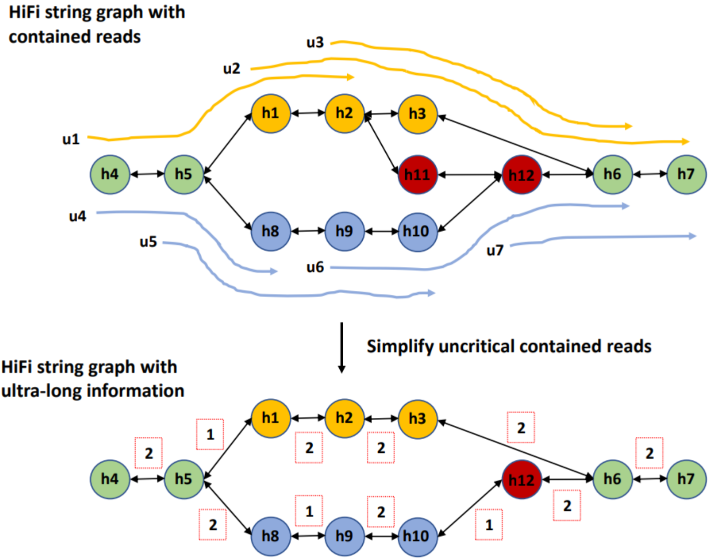
- An overlap graph limitation when using noisy reads



- An overlap graph limitation when using noisy reads

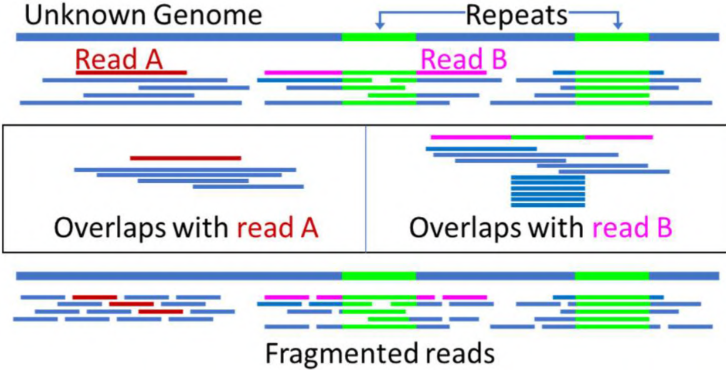


Read threading alternative

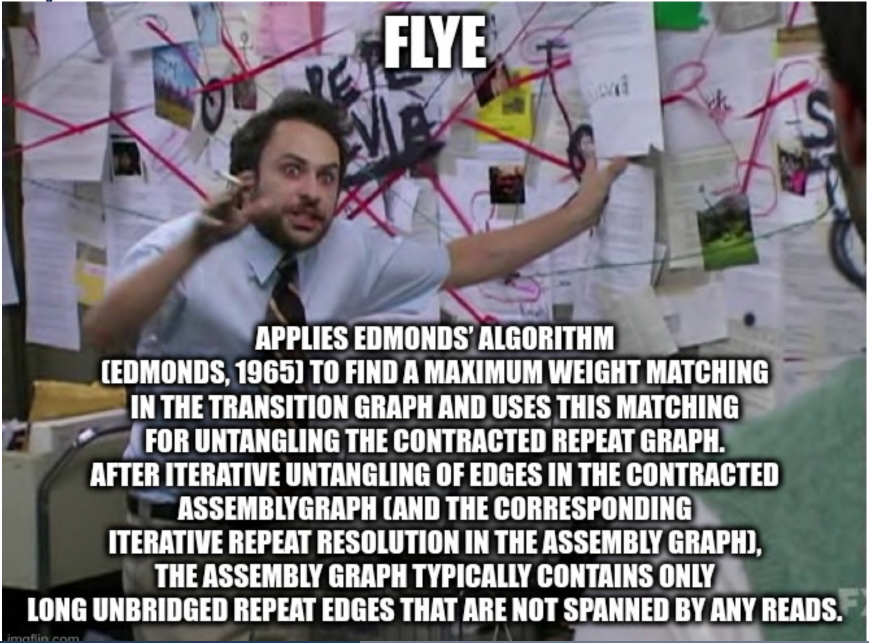

HiFi string graph with contained reads
u1
u2
u3
u4
u5
u6
u7
h1
h2
h3
h4
h5
h6
h7
h8
h9
h10
h11
h12
Simplify uncritical contained reads
HiFi string graph with ultra-long information
h1
h2
h3
h4
h5
h6
h7
h8
h9
h10
h12
2
1
2
2
2
2
2
1
2
1
2

Fragmented read alternative

The RAFT tool fragments the reads that does not cover repeats to avoid read inclusion problems.

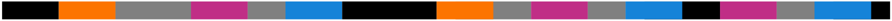


Flye

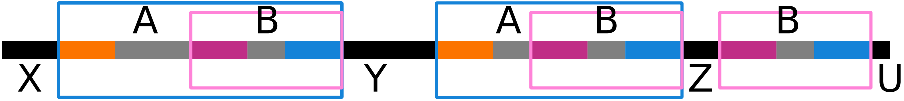

FLYE
APPLIES EDMONDS' ALGORITHM (EDMONDS, 1965) TO FIND A MAXIMUM WEIGHT MATCHING IN THE TRANSITION GRAPH AND USES THIS MATCHING FOR UNTANGLING THE CONTRACTED REPEAT GRAPH. AFTER ITERATIVE UNTANGLING OF EDGES IN THE CONTRACTED ASSEMBLYGRAPH (AND THE CORRESPONDING ITERATIVE REPEAT RESOLUTION IN THE ASSEMBLY GRAPH), THE ASSEMBLY GRAPH TYPICALLY CONTAINS ONLY LONG UNBRIDGED REPEAT EDGES THAT ARE NOT SPANNED BY ANY READS.
imgflip.com

Repeat graph

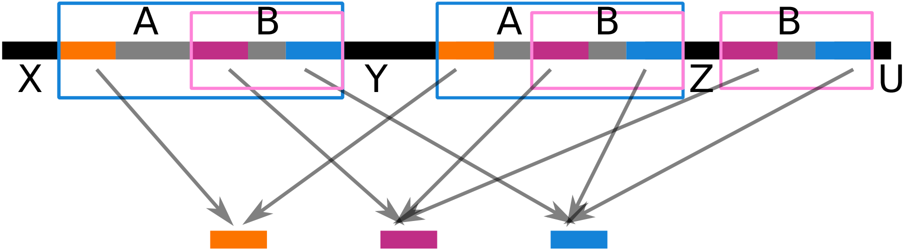
a genome



highlighted repeated regions

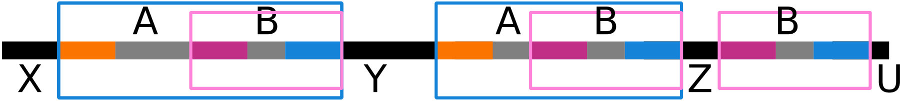


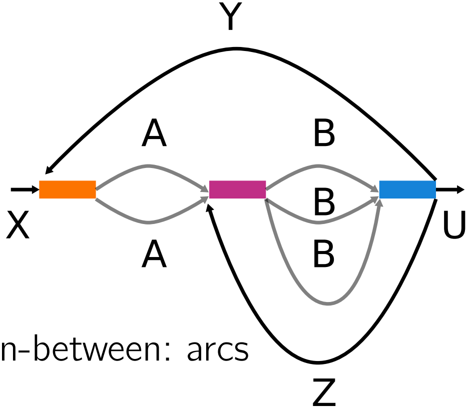
Repeat graph



repeats extremities: graph's nodes

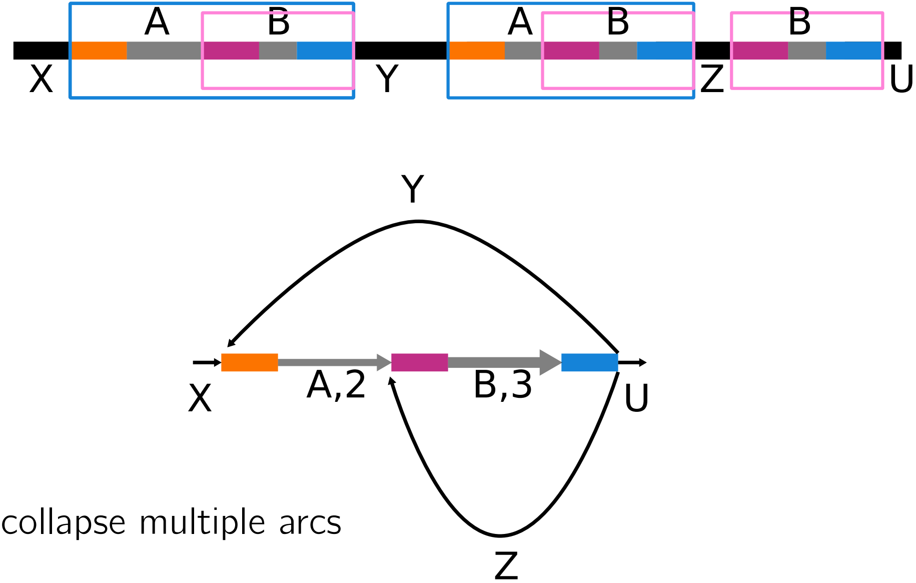
Repeat graph





sequences in-between: arcs

Repeat graph



collapse multiple arcs

The end (Thank you for your attention)

Slides for the practical session

Evaluate assembly

Two cases:

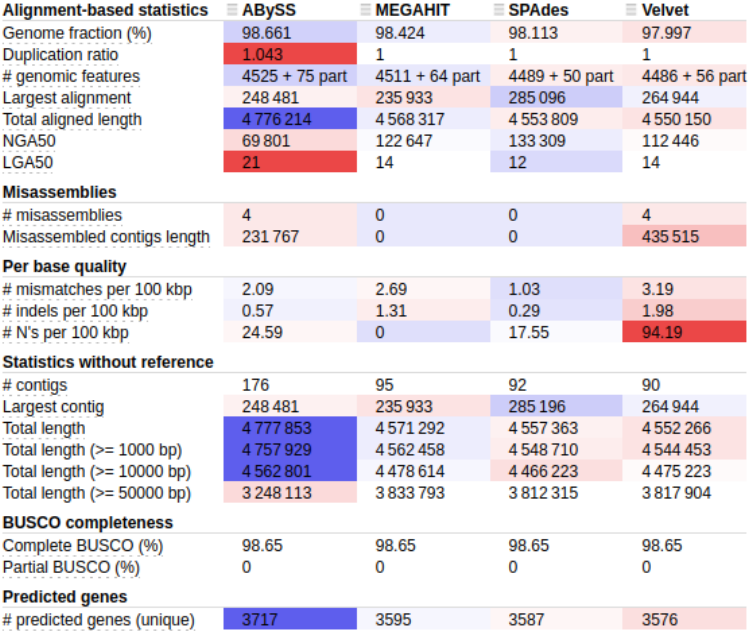
Reference-based

Align contigs to the reference and compare them considering the reference as the ground truth (QUAST).

De novo

- Reads analysis (QUAST)
- Kmer analysis (Merqury)
- Assembly graph analysis (Bandage)

- # QUAST statistics

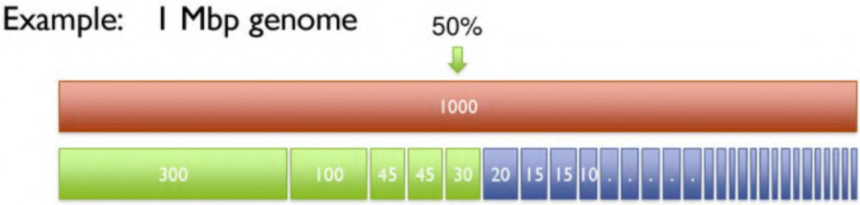


Alignment-based statistics	ABySS	MEGAHIT	SPAdes	Velvet
Genome fraction (%)	98.661	98.424	98.113	97.997
Duplication ratio	1.043	1	1	1
# genomic features	4525 + 75 part	4511 + 64 part	4489 + 50 part	4486 + 56 part
Largest alignment	248 481	235 933	285 096	264 944
Total aligned length	4 776 214	4 568 317	4 553 809	4 550 150
NGA50	69 801	122 647	133 309	112 446
LGA50	21	14	12	14
Misassemblies				
# misassemblies	4	0	0	4
Misassembled contigs length	231 767	0	0	435 515
Per base quality				
# mismatches per 100 kbp	2.09	2.69	1.03	3.19
# indels per 100 kbp	0.57	1.31	0.29	1.98
# N's per 100 kbp	24.59	0	17.55	94.19
Statistics without reference				
# contigs	176	95	92	90
Largest contig	248 481	235 933	285 196	264 944
Total length	4 777 853	4 571 292	4 557 363	4 552 266
Total length (>= 1000 bp)	4 757 929	4 562 458	4 548 710	4 544 453
Total length (>= 10000 bp)	4 562 801	4 478 614	4 466 223	4 475 223
Total length (>= 50000 bp)	3 248 113	3 833 793	3 812 315	3 817 904
BUSCO completeness				
Complete BUSCO (%)	98.65	98.65	98.65	98.65
Partial BUSCO (%)	0	0	0	0
Predicted genes				
# predicted genes (unique)	3717	3595	3587	3576

- 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.



- The catsembler

genome

ACGGATGATAGATTTGATACGA

GATTTGATAC
reads ACGGATGATA
TTTGATACGA

concatenate the reads: super N50!

GATTTGATACACGGATGATATTTGATACGA

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.

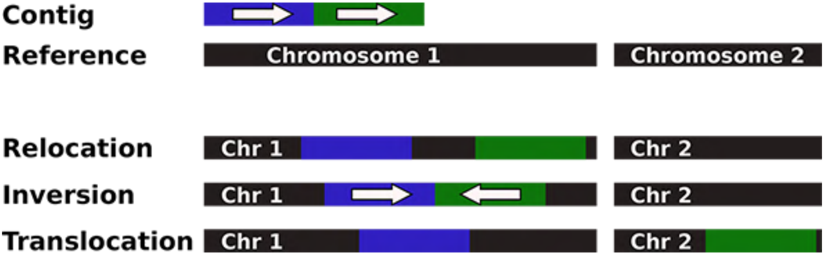
N75

N75 is the same statistic for 75% of the assembly

NGA50

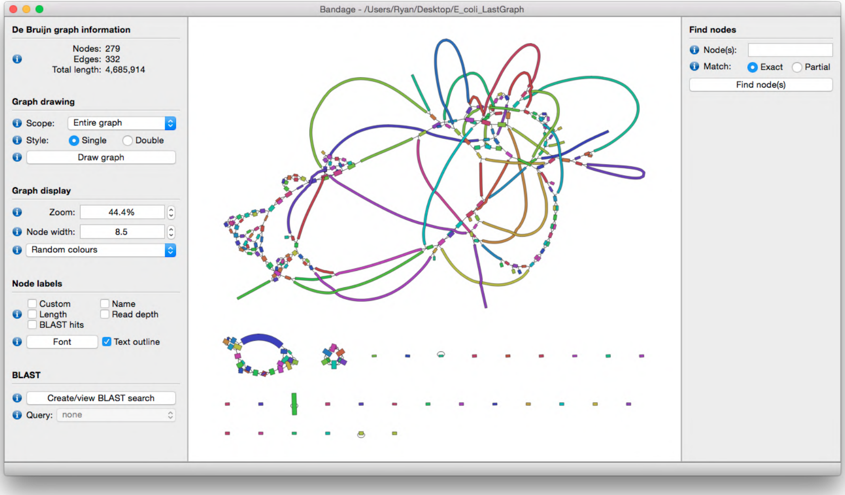
Similar to the N50 but only takes into account contigs/scaffolds that can be **aligned** on the reference genome and consider 50% of the **genome size** instead of the assembly size

- Misassemblies



- Visualize assembly

Bandage tool can visualize assembly graphs (GFA)



Bandage - /Users/Ryan/Desktop/E_coli_LastGraph

De Bruijn graph information

Nodes: 279
Edges: 332
Total length: 4,685,914

Graph drawing

Scope: Entire graph
Style: ● Single ○ Double
Draw graph

Graph display

Zoom: 44.4%
Node width: 8.5
Random colours

Node labels

□ Custom □ Name
□ Length □ Read depth
□ BLAST hits
Font ☑ Text outline

BLAST

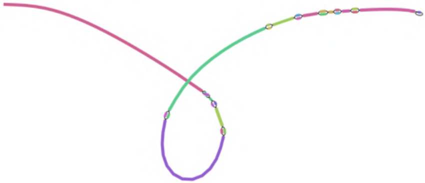
Create/view BLAST search
Query: none

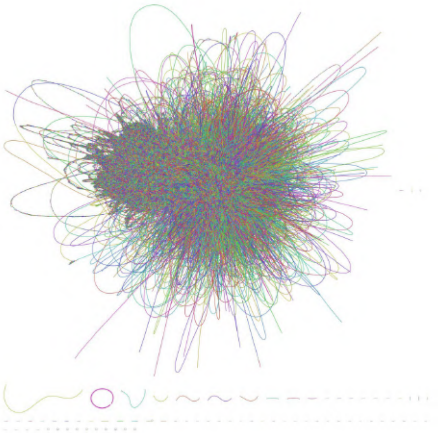
Find nodes

Node(s):
Match: ● Exact ○ Partial
Find node(s)

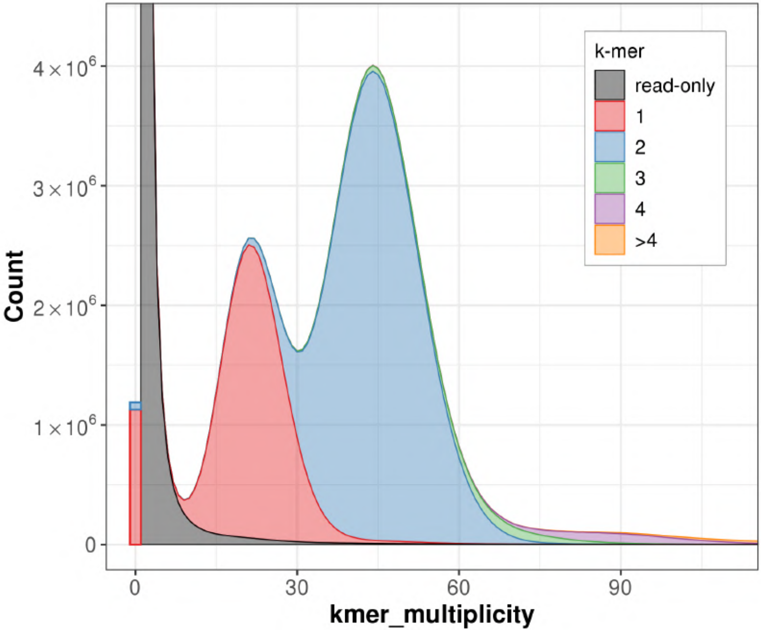
Visualize assembly

Bandage tool can visualize assembly graphs (GFA)

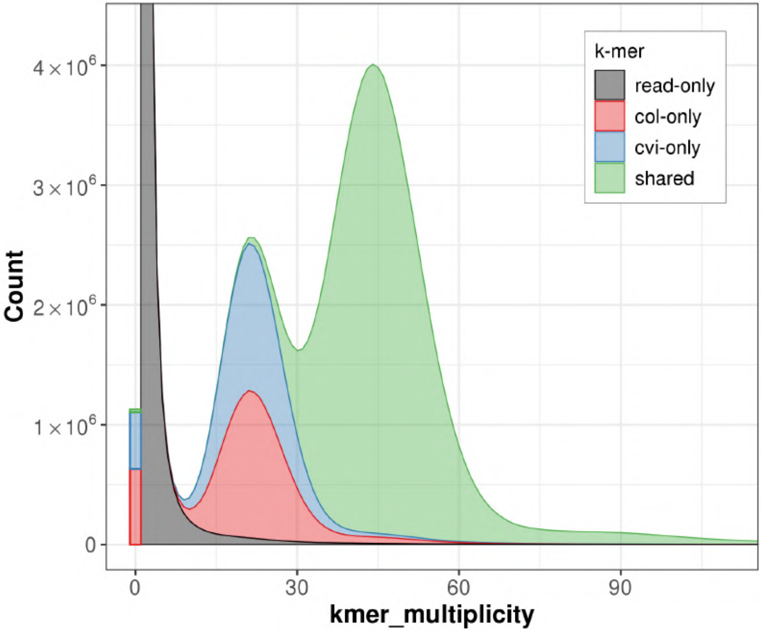




- *K*-mer spectrum visualization with merqury



- Trio *K*-mer spectrum visualization with KAT



SPAdes assembler

- Designed to assemble megabase-sized genomes
- Multiple k de Bruijn graph assembly from short reads
- Can use long reads to solve repeats

Mandatory

Short reads

Optional

Long reads

Hifiasm assembler

- Build an overlap graph from HiFi reads
- Generate both haploid and diploid assemblies
- Can use (very) long reads to solve repeats

Mandatory

HiFi reads

Optional

Long reads

Flye assembler

- Build a repeat graph from long reads
- Can use any kind of long reads
- Can also assemble metagenomes

Mandatory

HiFi/Long reads

Optional

HiFi/Long reads

Unicycler (long read mode)

- Build a overlap graph from long reads
- Polish the assembly
- Also has a short-reads-first similar to SPAdes

Mandatory

Long reads

Optional

Short reads