A little tour of assembly methods

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

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

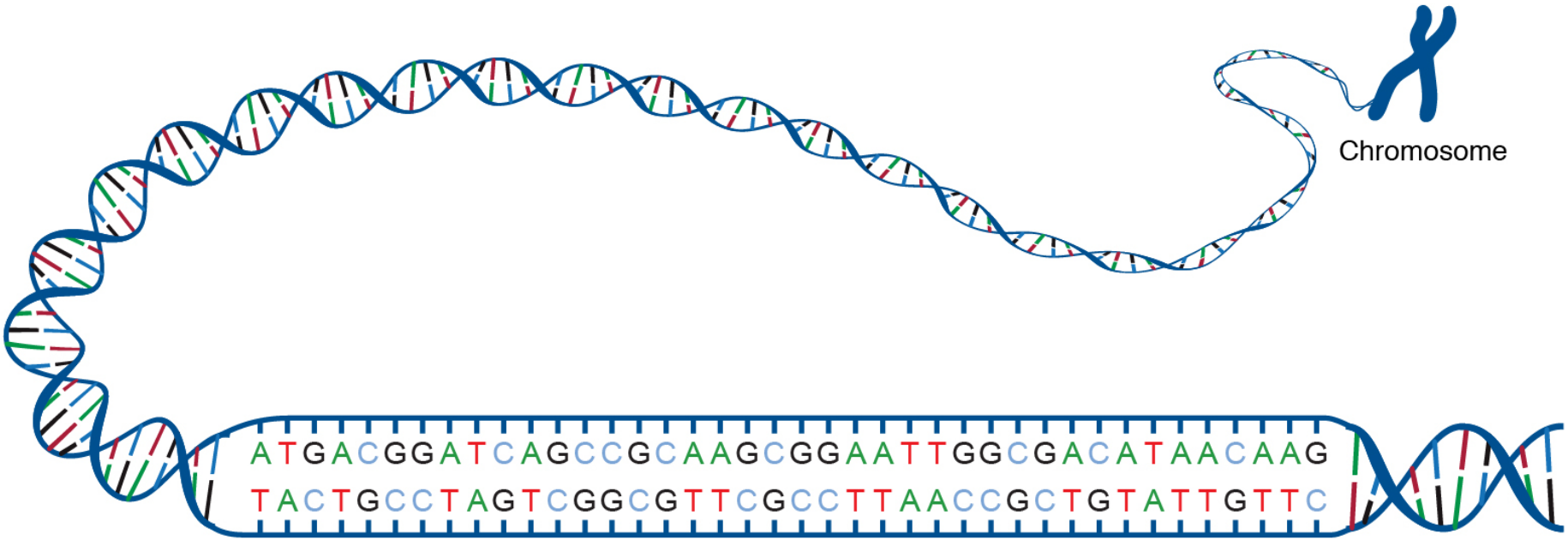
Content of this course

- 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

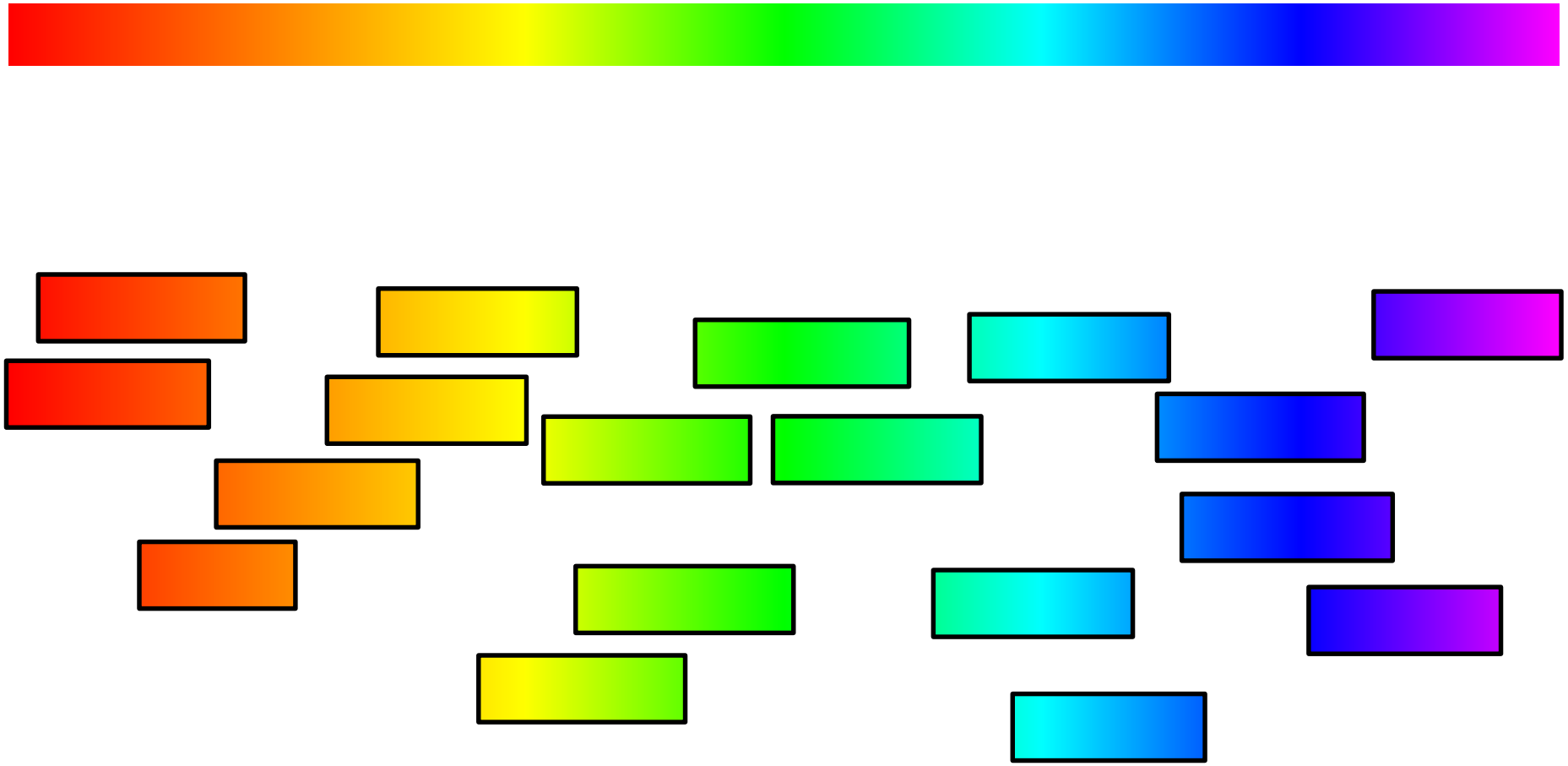
genome size: $\sim$ 32 gigabases

- Accessing a genome

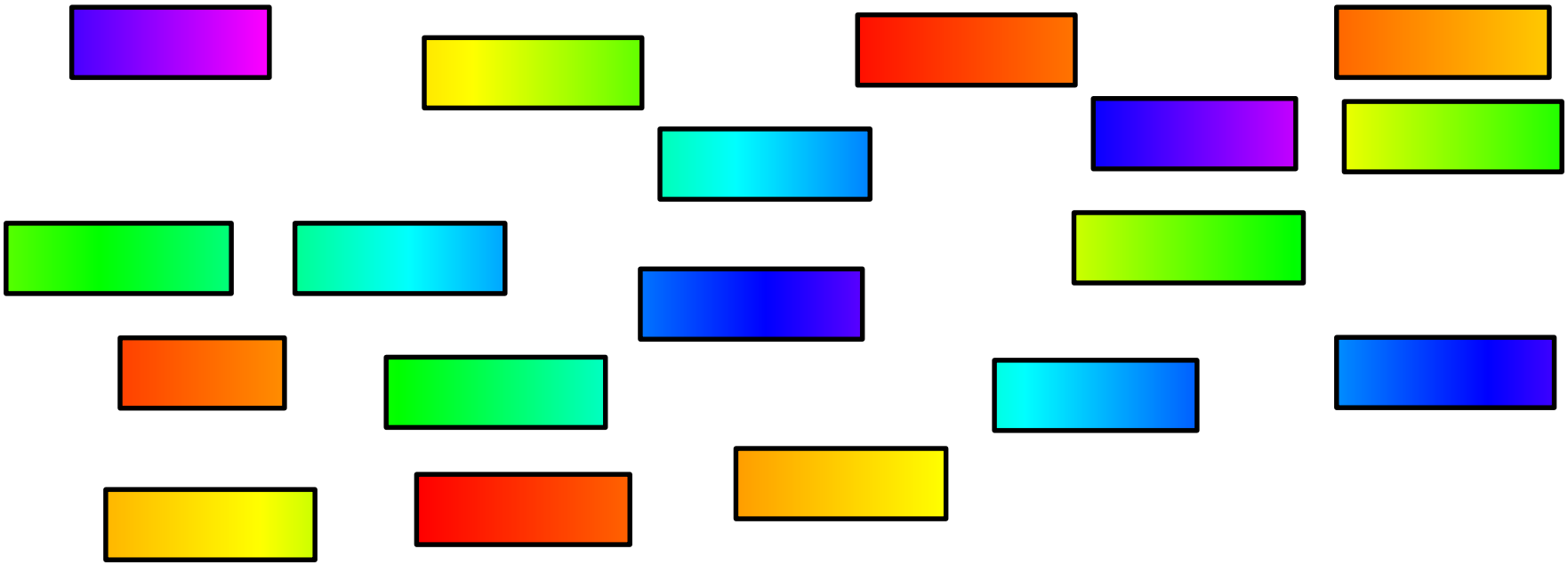


From `www.genome.gov/genetics-glossary/acgt`

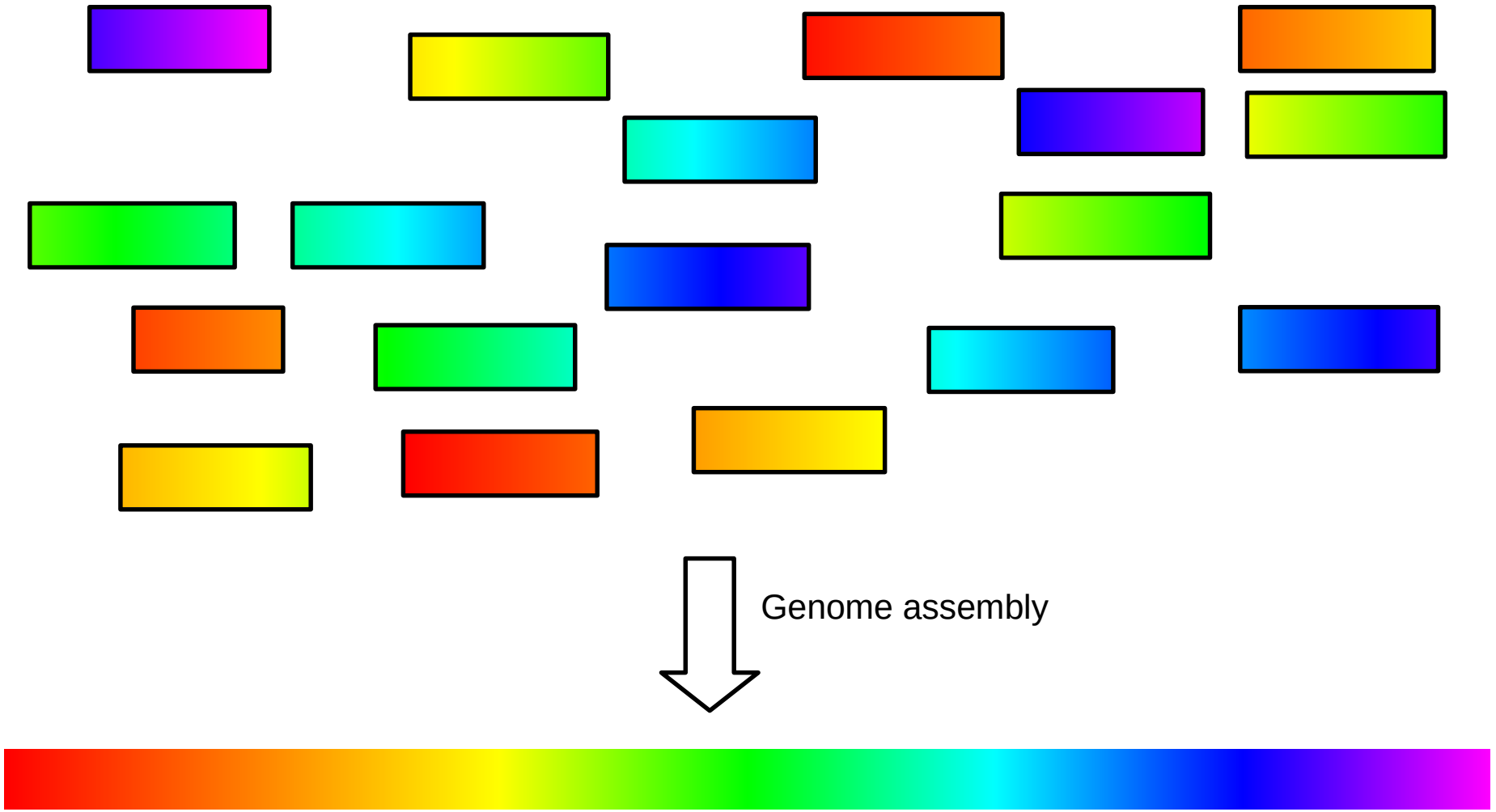
- Reads are words from the genome



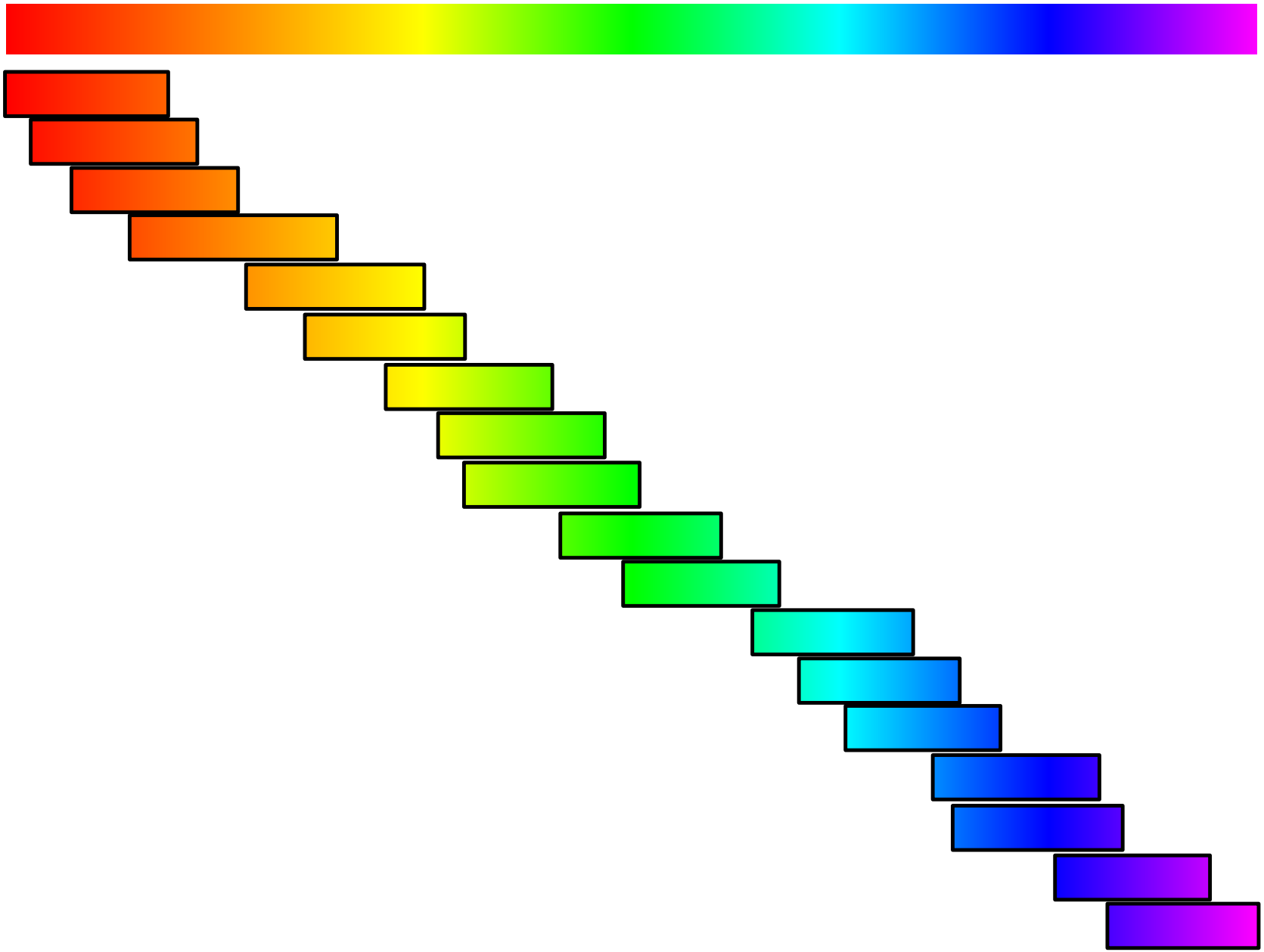
- Reads are **shuffled** words from the genome



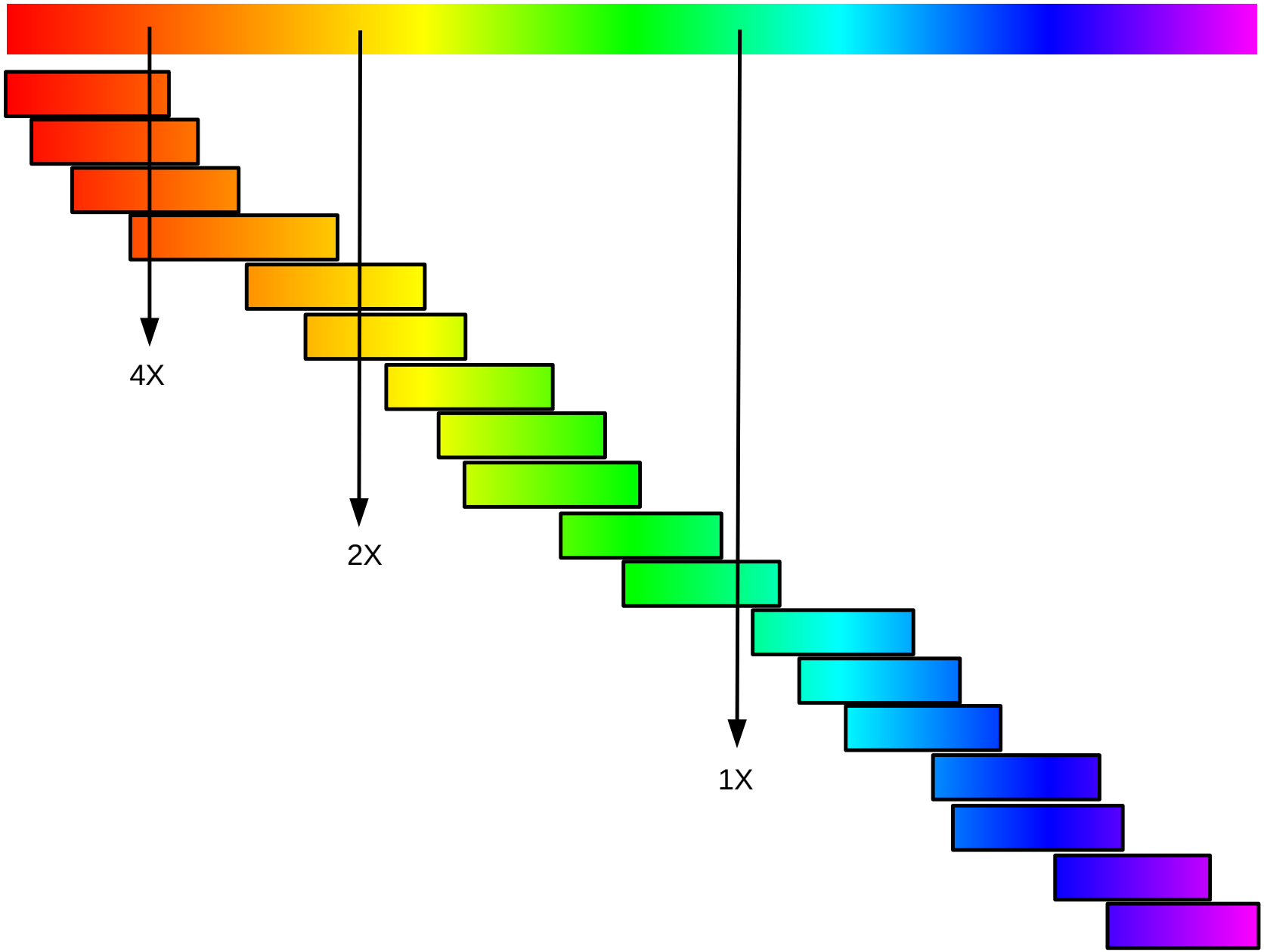
- Genome assembly task



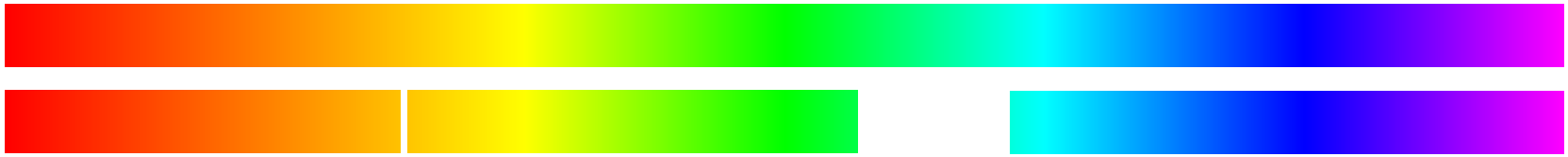
Using read overlaps



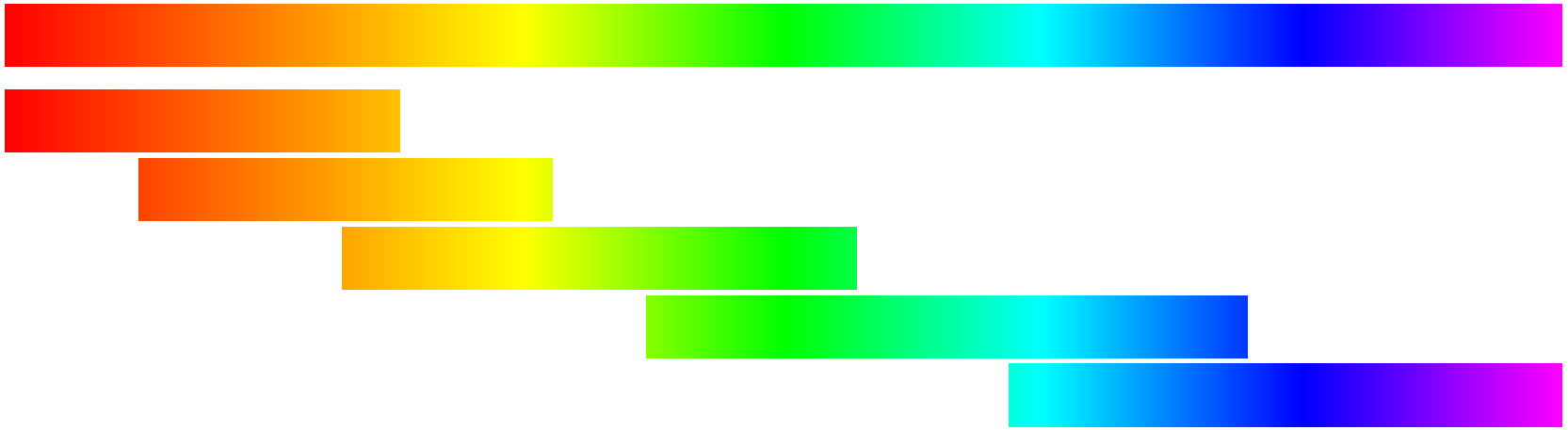
Genome sequencing: coverage



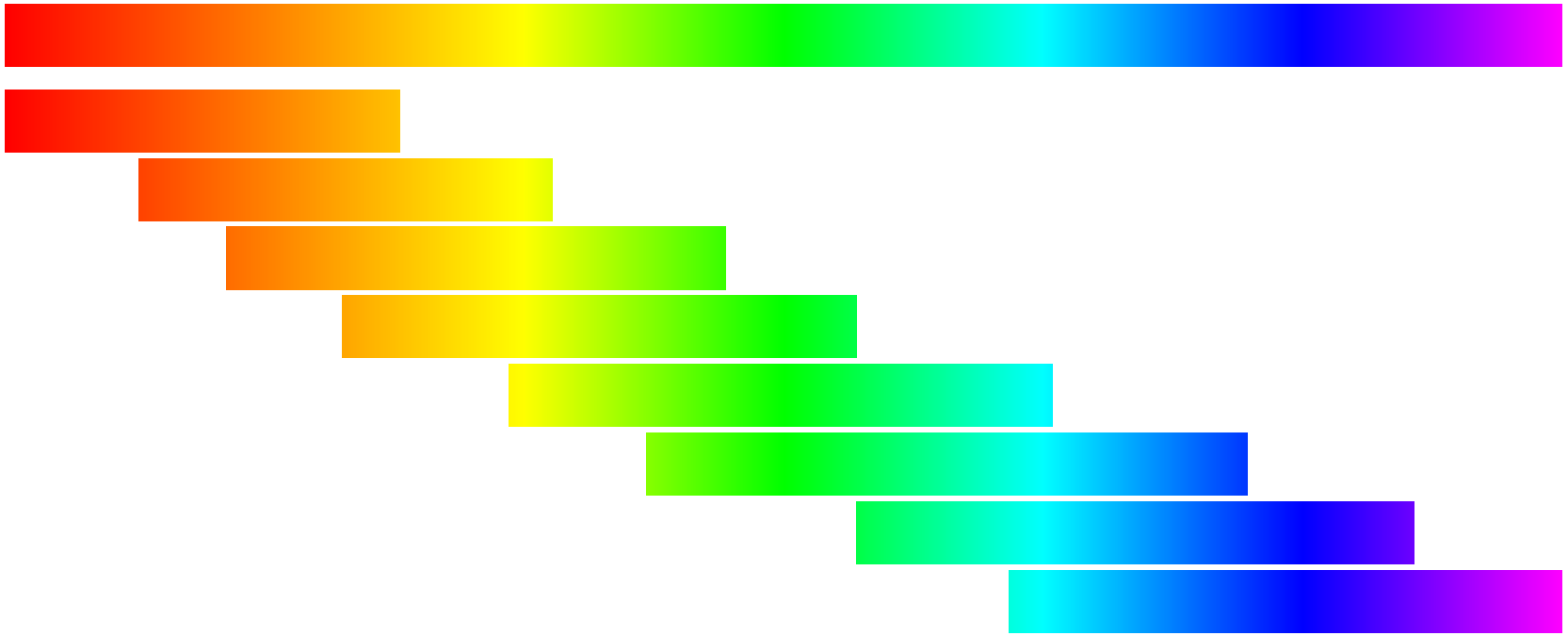
- Genome sequencing: coverage



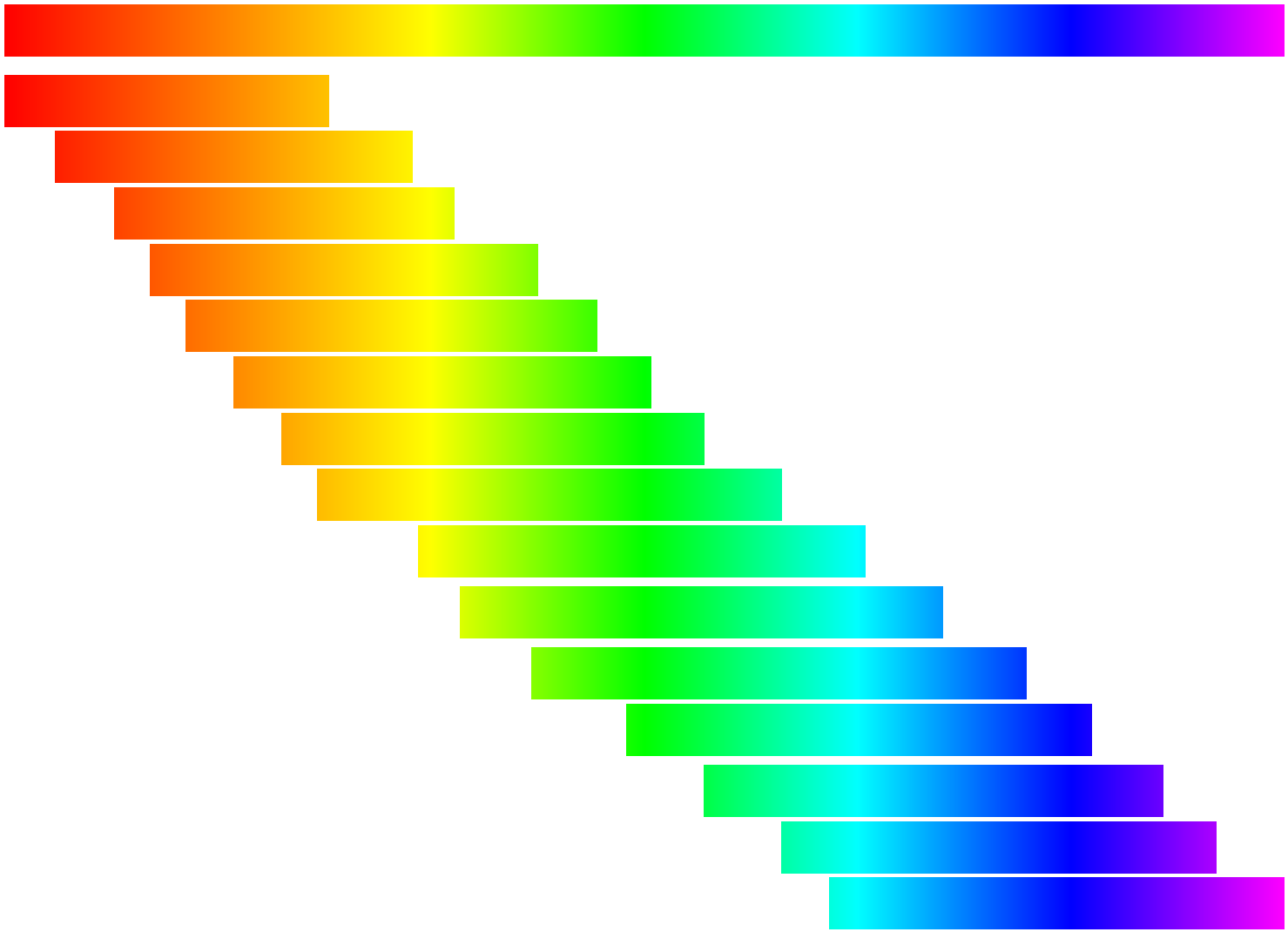
- Genome sequencing: coverage



- 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                  Overlap length:7
2:     GGTATCGTTA

1:  ATCGGTATCG                  Overlap length:4
3:        ATCGTTACGG

1:  ATCGGTATCG                  Overlap length:1
4:             GTTACGGTAT

1:  ATCGGTATCG                  No Overlap
5:               ACGGTATACC
```

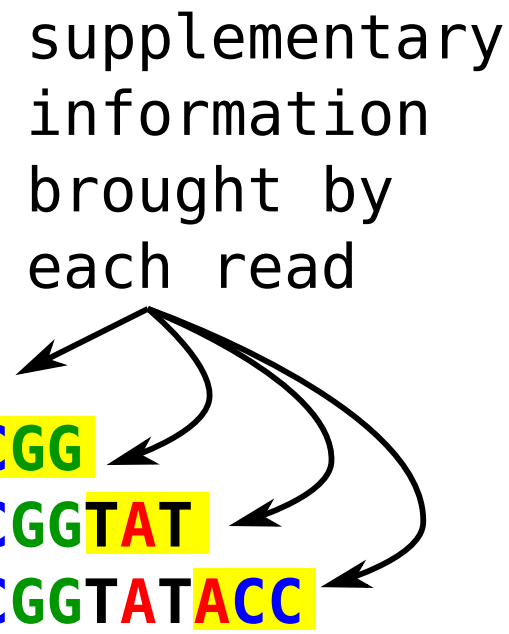
- Assembly idea number 1: assemble the longest overlaps

Reads:

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

Best overlaps:

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



Output"genome":

ATCGGTATCG + TTA + ATCGTTACGG + TAT + ACC

- Your time to shine!

Let assemble this genome!

Your read set:

```
1: ATTTACGGGT
2: TTACGGGTGG
3: ACGGGTCCTT
4: GTCCTTTCTT
5: TTTCTTACGG
```

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
```

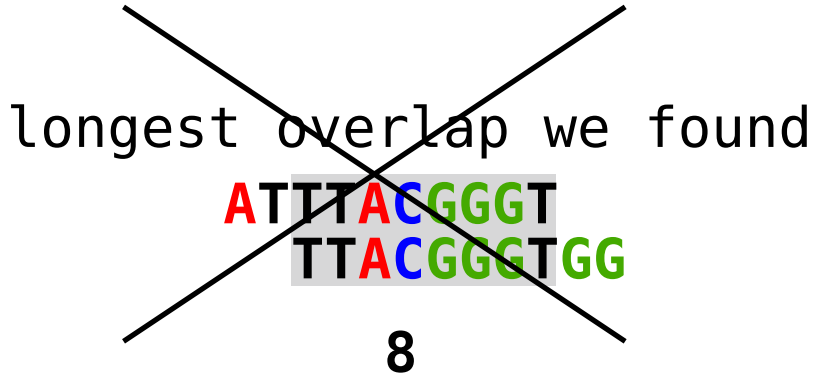
- The actual solution

The actual genome:

ATTTACGGGTCCTTTCTTACGGGTGG

How the reads should be ordered:

```
ATTTACGGGT
    ACGGGTCCTT
    6   GTCCTTTCTT
            TTTCTTACGG
                TTACGGGTGG
```



- What happened?

The actual genome:

ATTTACGGGTCCTTTCTTACGGGTGG

How the reads should be ordered:

```
ATTTACGGGT
    ACGGGTCCTT
    6   GTCCTTTCTT
            TTTCTTACGG
                TTACGGGTGG
```

longest overlap we found

```
ATTTACGGGT
   TTACGGGTGG
```

8

ATTTACGGGTGG
not in the genome

ACGGGTCCTTTCTTACGG
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:

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

Decimal approximation:

0.99999728949878430238317205542136383671202317193893202410 6...

From `en.wikipedia.org/wiki/Birthday_problem`

- The burden of assembly: genomic repeats

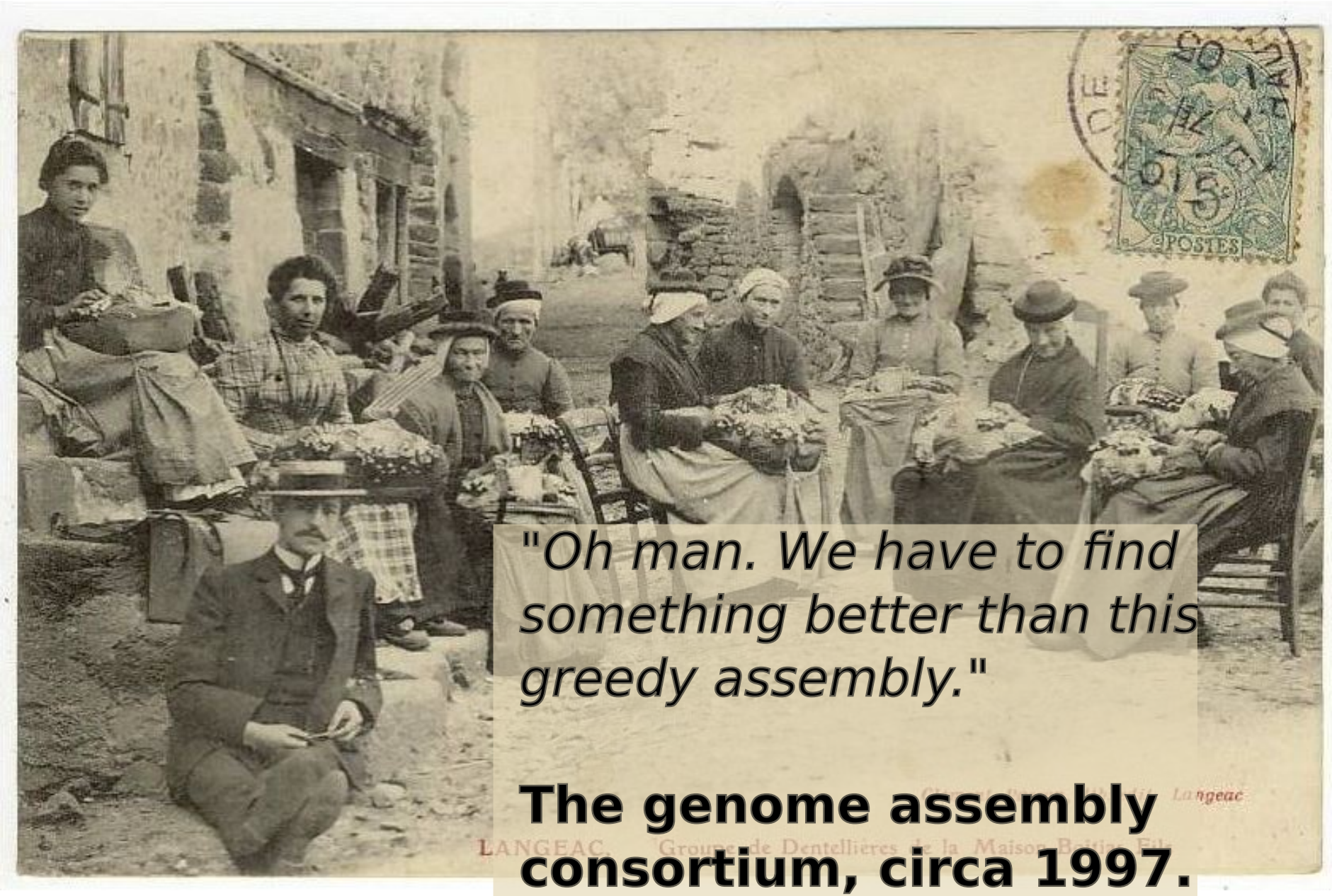
Amount of repeats larger than a given size in *E. coli* genome

- 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

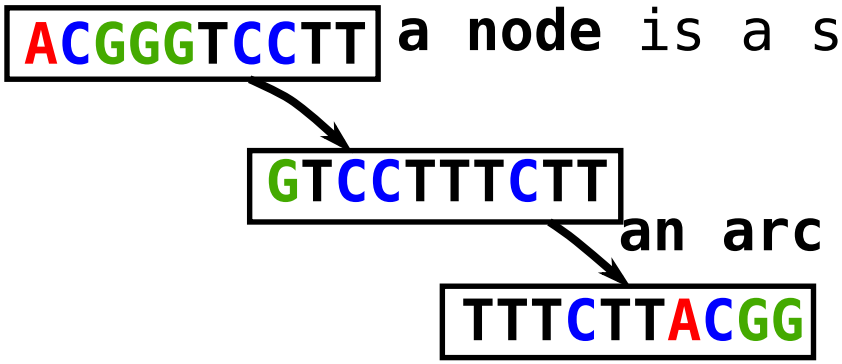
- History from the last century



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

The genome assembly consortium, circa 1997.

- Graph representation



a node is a sequence

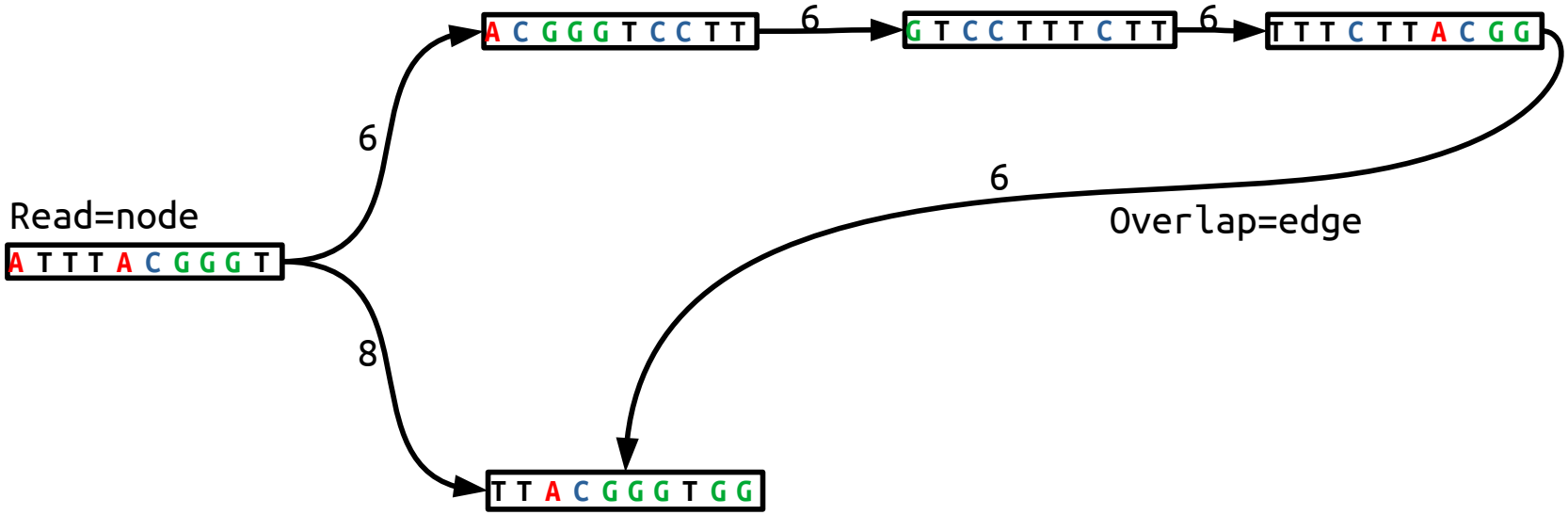
an arc oriented between
a source node and
a sink node

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:
ATTTACGGGTCCTTTCTTACGGGTGG

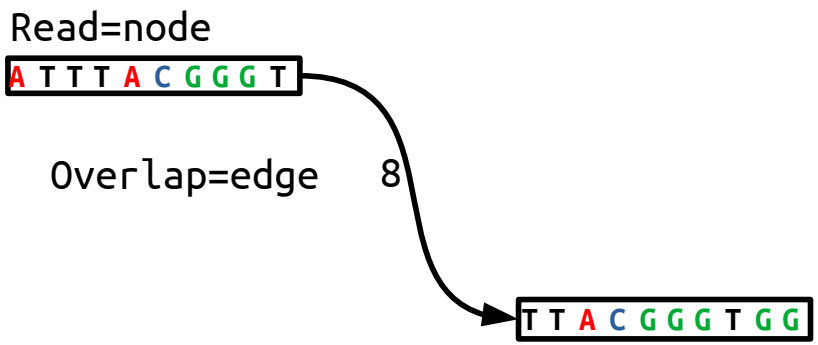
Overlap graph:



Greedy solution

Genome:
ATTTACGGGTCCTTTCTTACGGGTGG

Overlap graph:

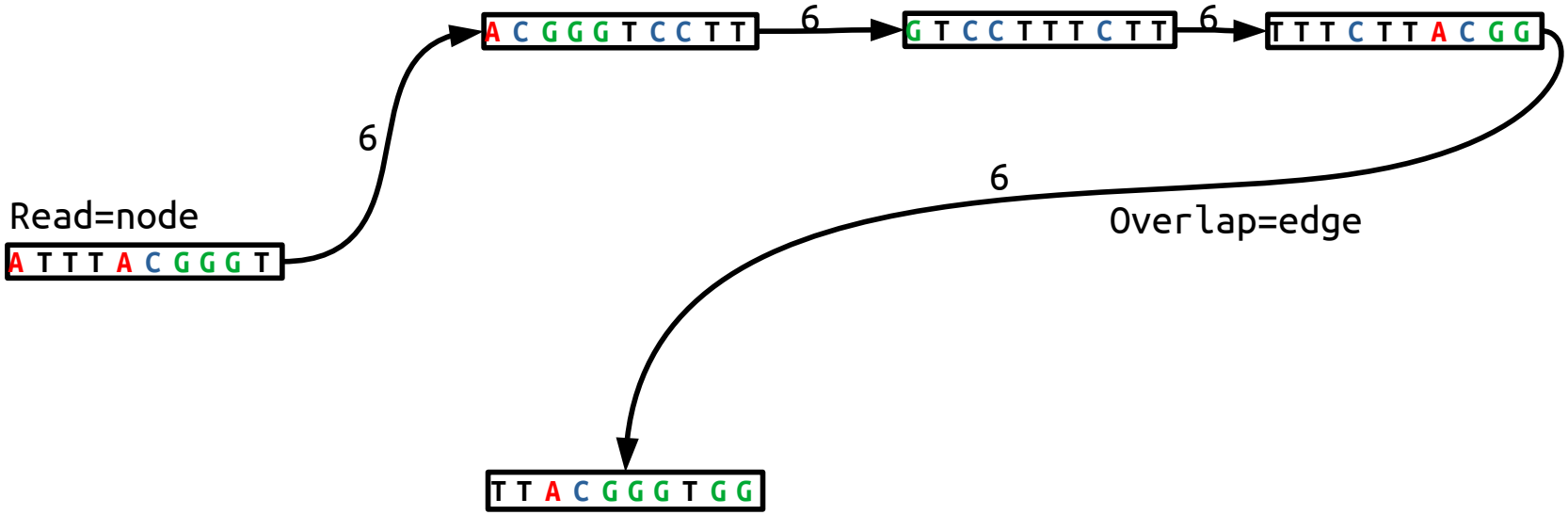


Greedy assembly output:
ATTTACGGGTGG
ACGGGTCCTTTCTTACGG

One piece solution

Genome:
ATTTACGGGTCCTTTCTTACGGGTGG

Overlap graph:

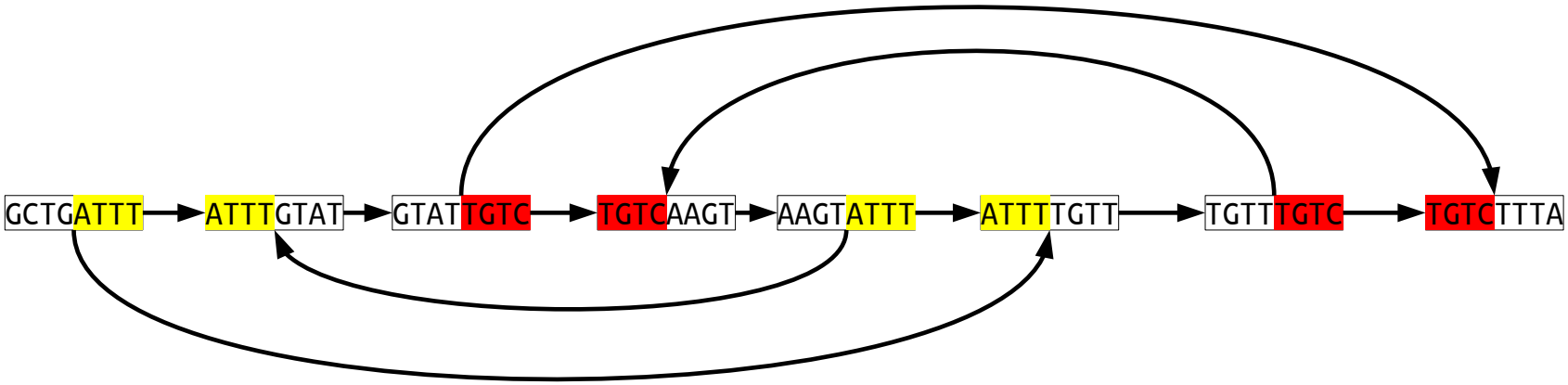


Overlap graph output:
ATTTACGGGTCCTTTCTTACGGGTGG

Multiple repeats

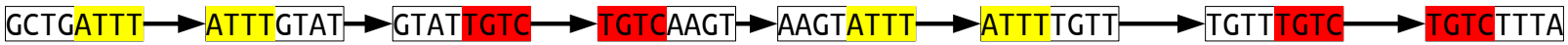
Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA

Overlap graph:



First solution

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

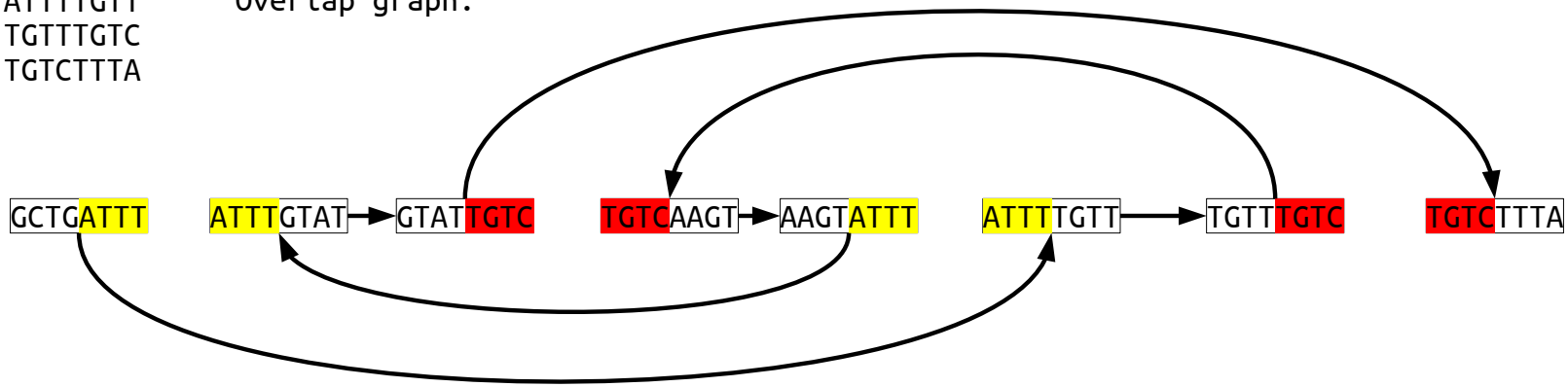



```
Possible assemblies:
GCTGATTTGTATTGTCAAGTATTTTGTTTGTCTTTA
```

Second solution

Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA

Overlap graph:

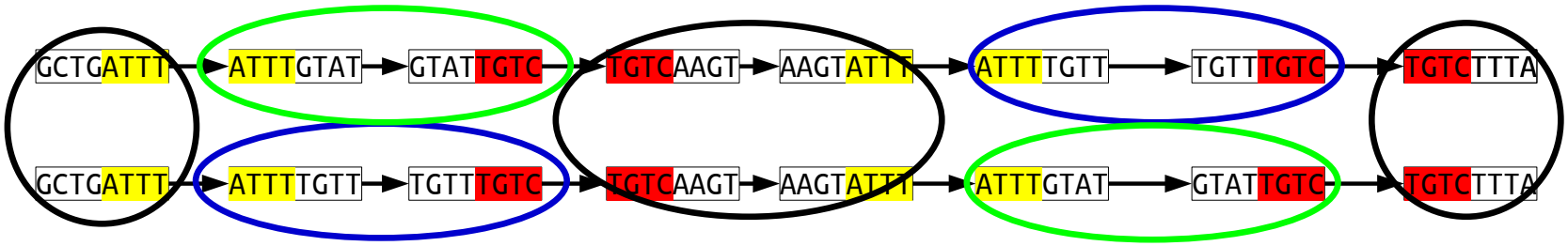


Possible assemblies:
GCTGATTTGTATTGTCAAGTATTTTGTTTGTCTTTA
GCTGATTTTGTTTGTCAAGTATTTGTATTGTCTTTA

Those two solutions are indistinguishable

Parsimonious solution: do not assemble

Possible assemblies:

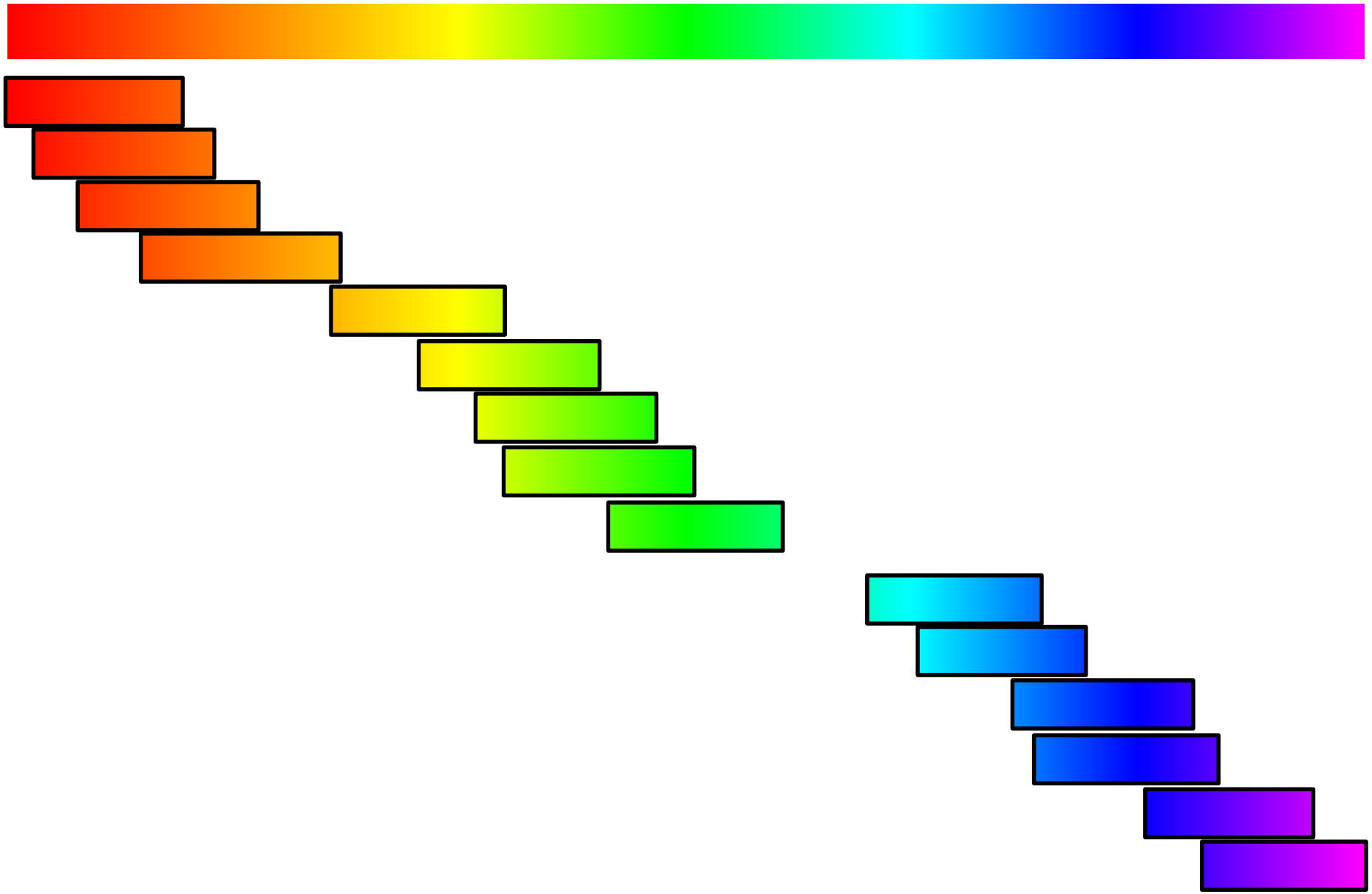


Genome pieces:

GCTGATTT ATTTGTATTGTC TGTCAAGTATTT ATTTTGTTTGTC

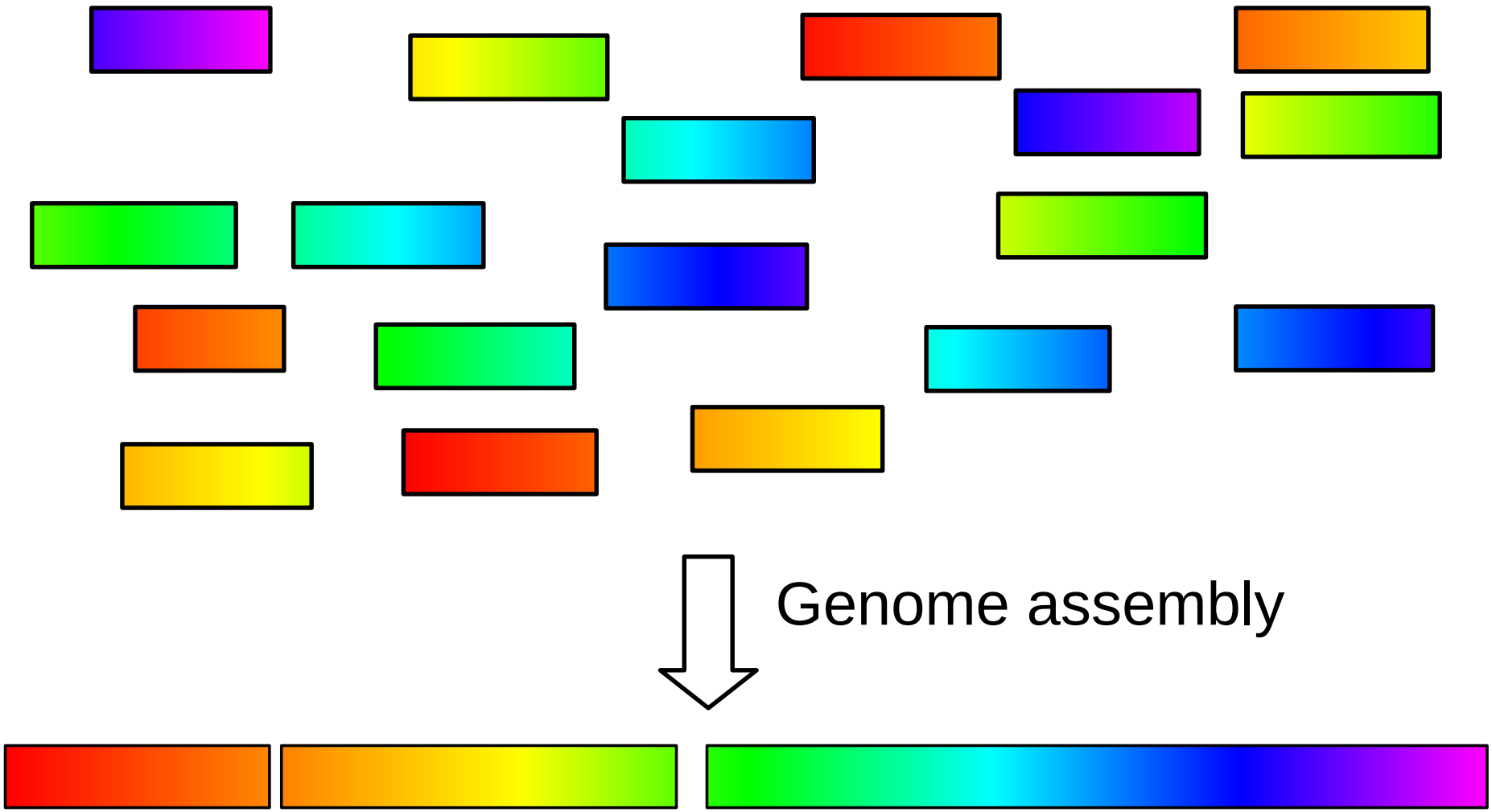
Repeats lead to the fragmentation of the assembly

- Missing information also fragments the assembly

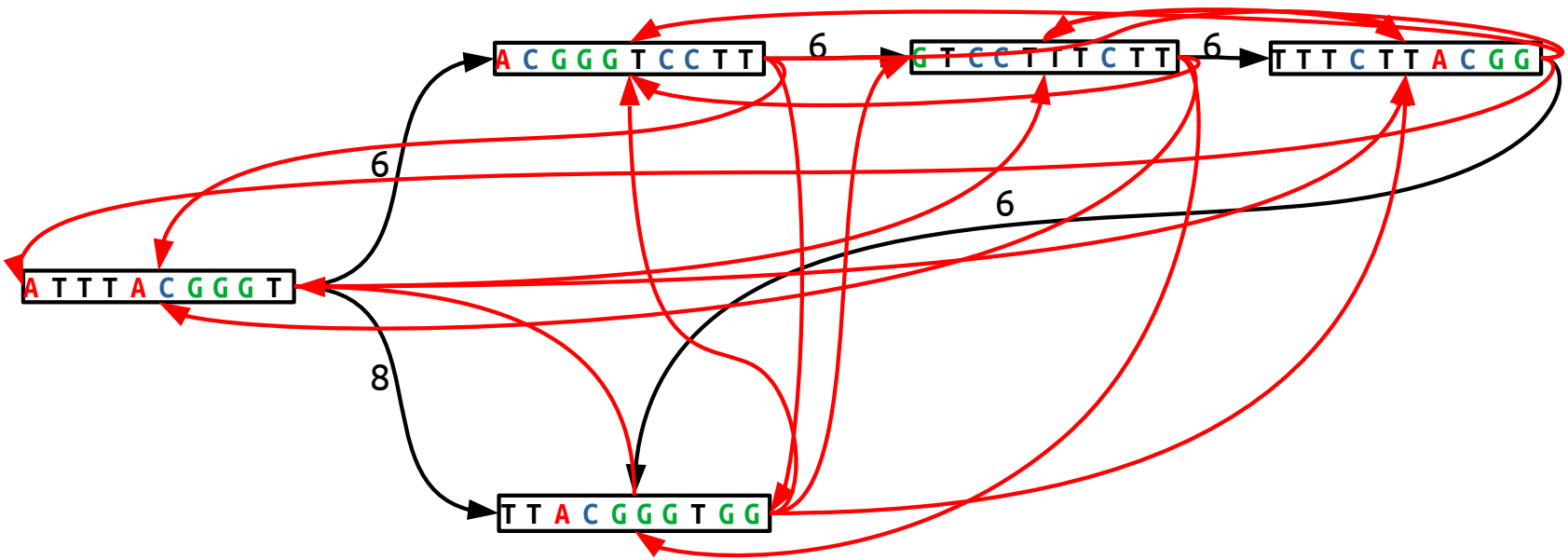


• Assembly **concession number 1: output fragments**

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

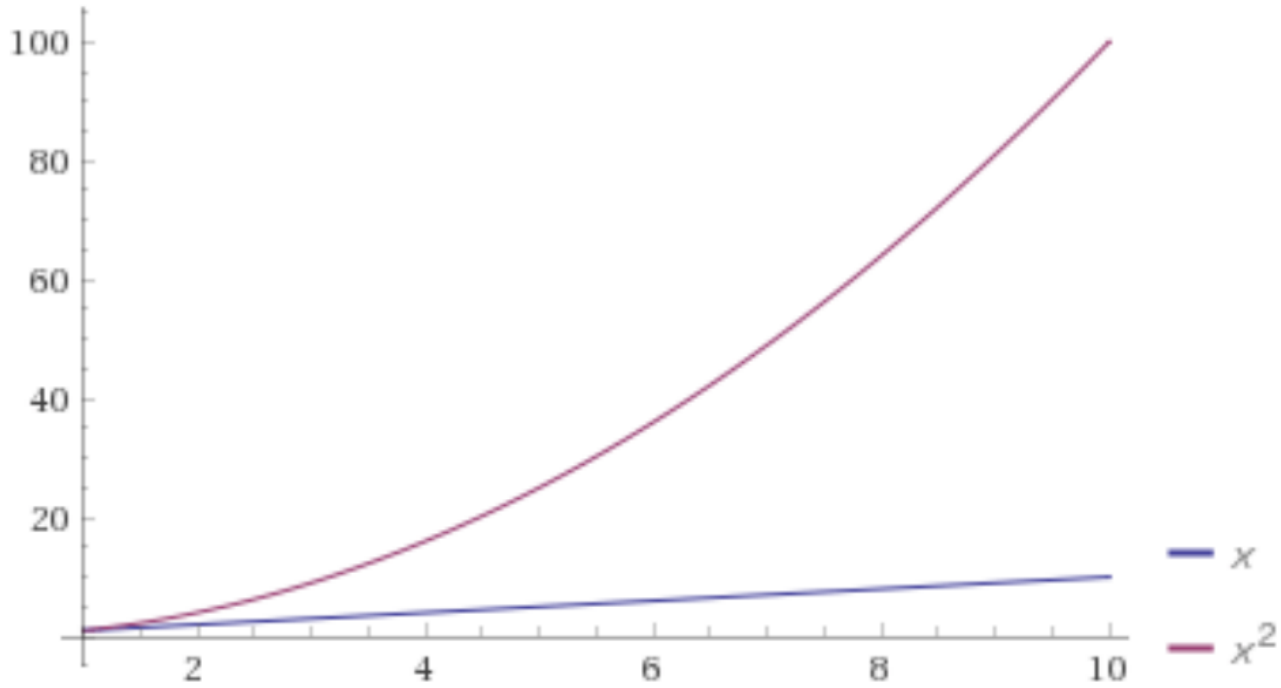


- Overlap graph prerequisite : all overlaps



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

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

Let's select a word from the genome:

AGATACAGCCATGACCG**TAGCATG**CTAACTGTGACGGCATTAC

in the genome, after **TAGCAT**

only **AGCATG** appears

- Genome words / read words

In real cases we don't have the genome

AGATACAGCCATGACCG**TAGCATG**CTAACTGTGACGGCATTAC

in the genome, after **TAGCAT**
only **AGCATG** appears

but we have the reads

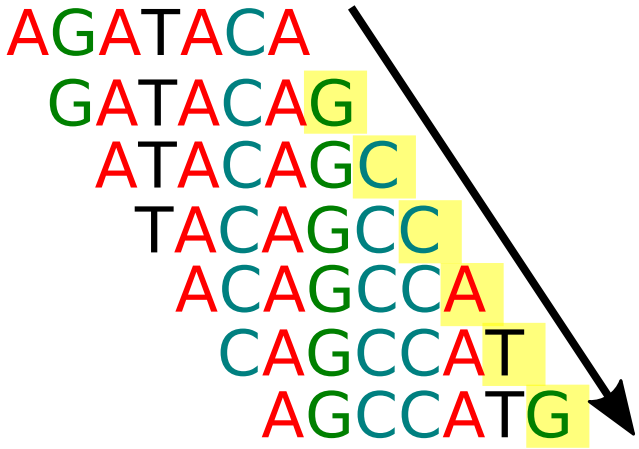
ATGACCG**TAGCATG**CT
ATGACCG**TAGCATG**CT
GACCG**TAGCATG**CTAA

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

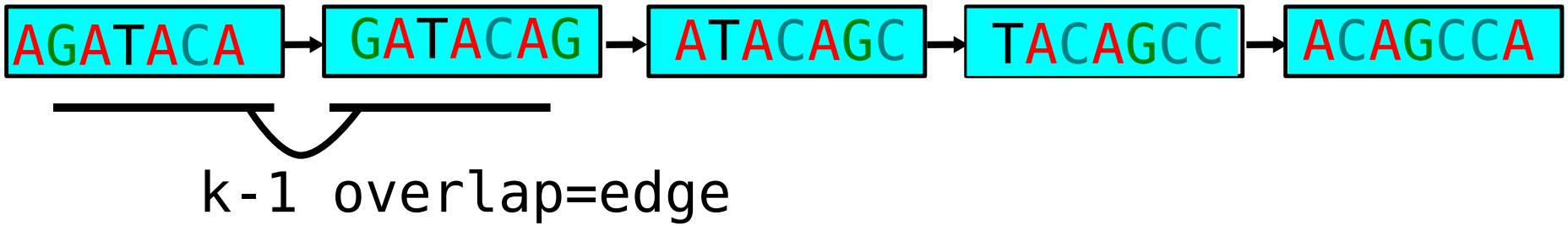
The de Bruijn graph

Read

AGATACAGCCA

De Bruijn graph

Kmer=node



AGATACA + G + C + C + A

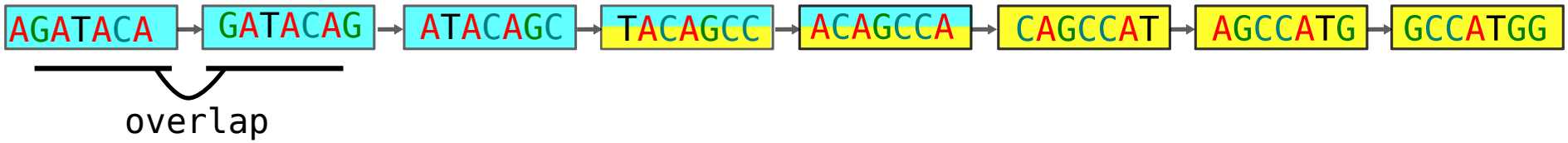
=AGATACAGCCA

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

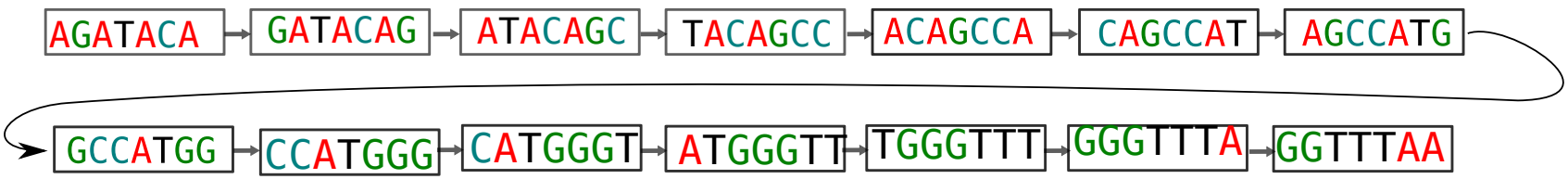
Hint: Use 7-mers

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

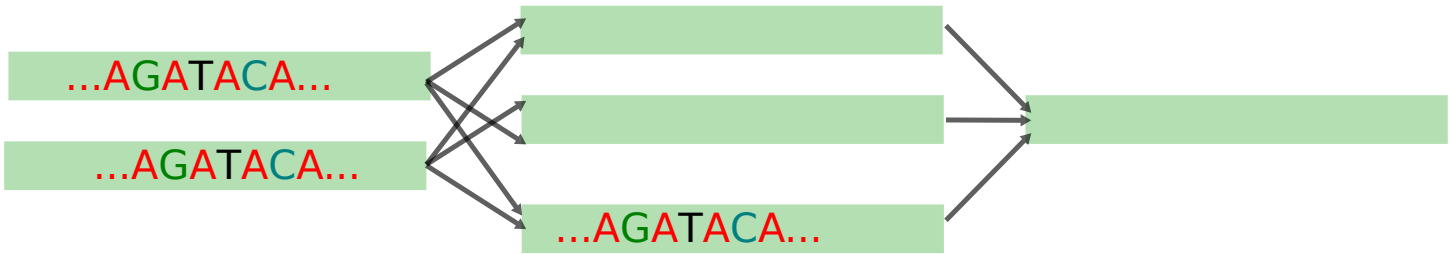
reads

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

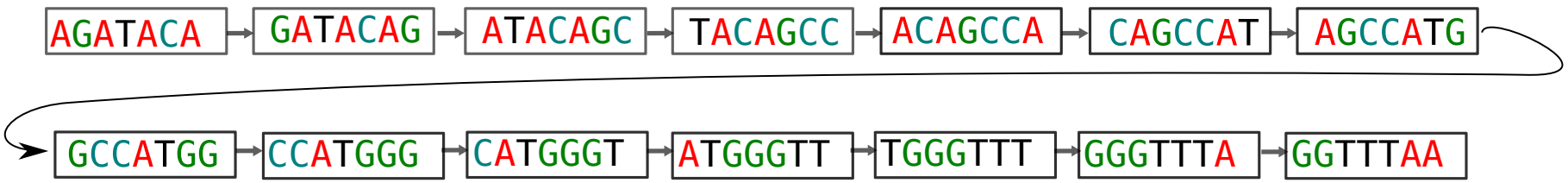
Overlapping reads

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

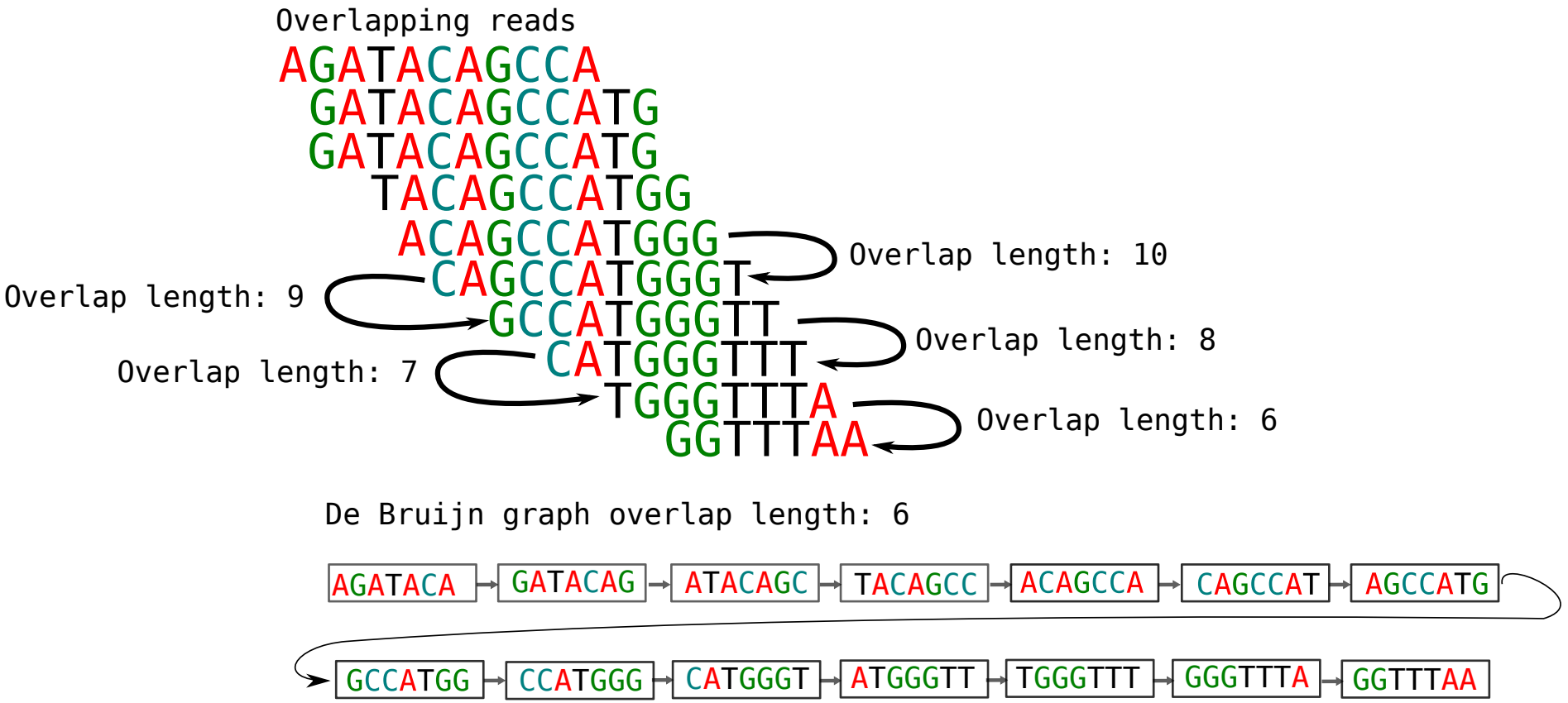
62 **(non distinct)** 7-mers in the reads

De Bruijn graph

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

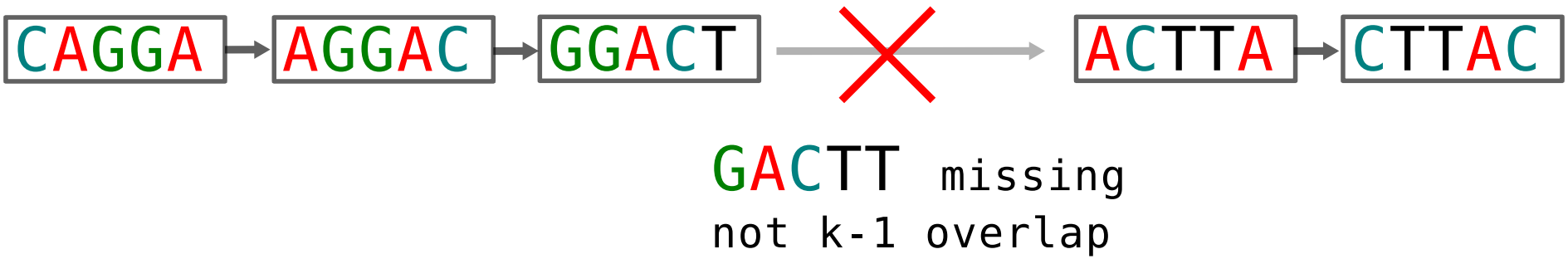


- de Bruijn graphs only rely on $k-1$ overlaps



- de Bruijn graphs limitation

Fixed overlaps



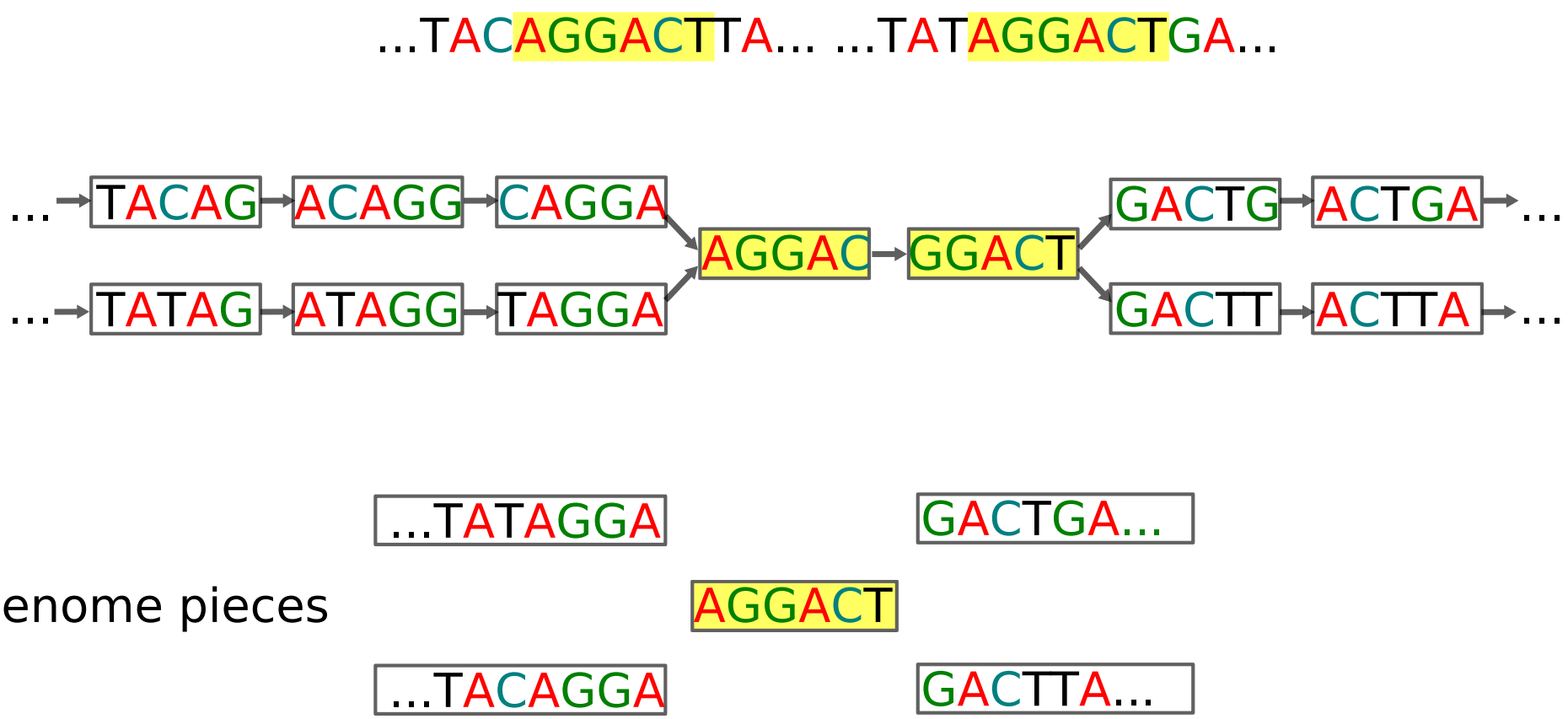
de Bruijn graphs limitation

Repeats...

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

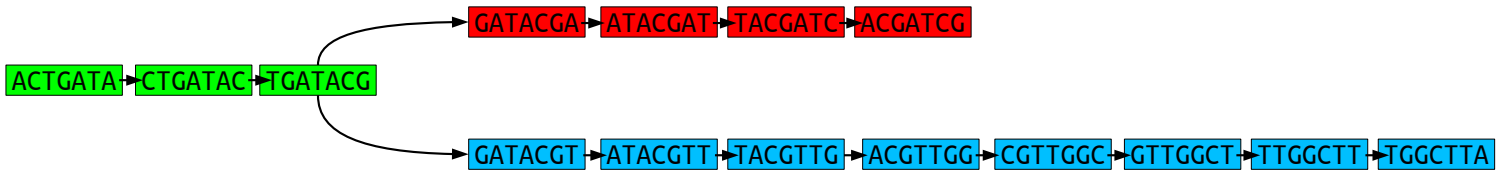
each k-mer appears only once in a de Bruijn graph

- de Bruijn graph limitation

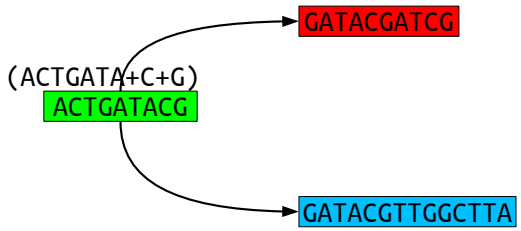


On the representation of de Bruijn graphs

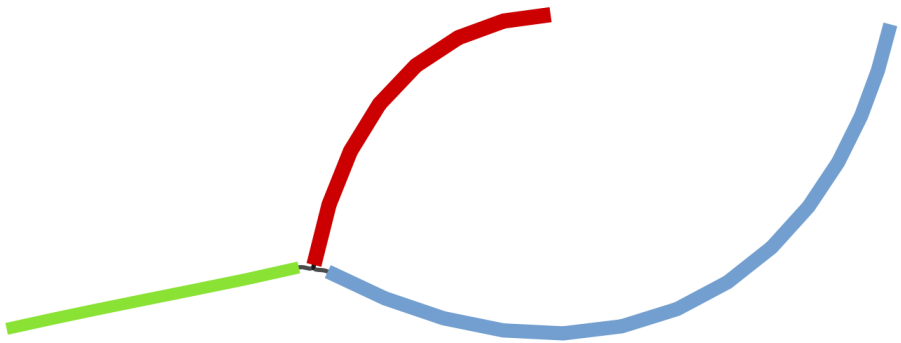
De Bruijn graph:



Compacted De Bruijn graph:

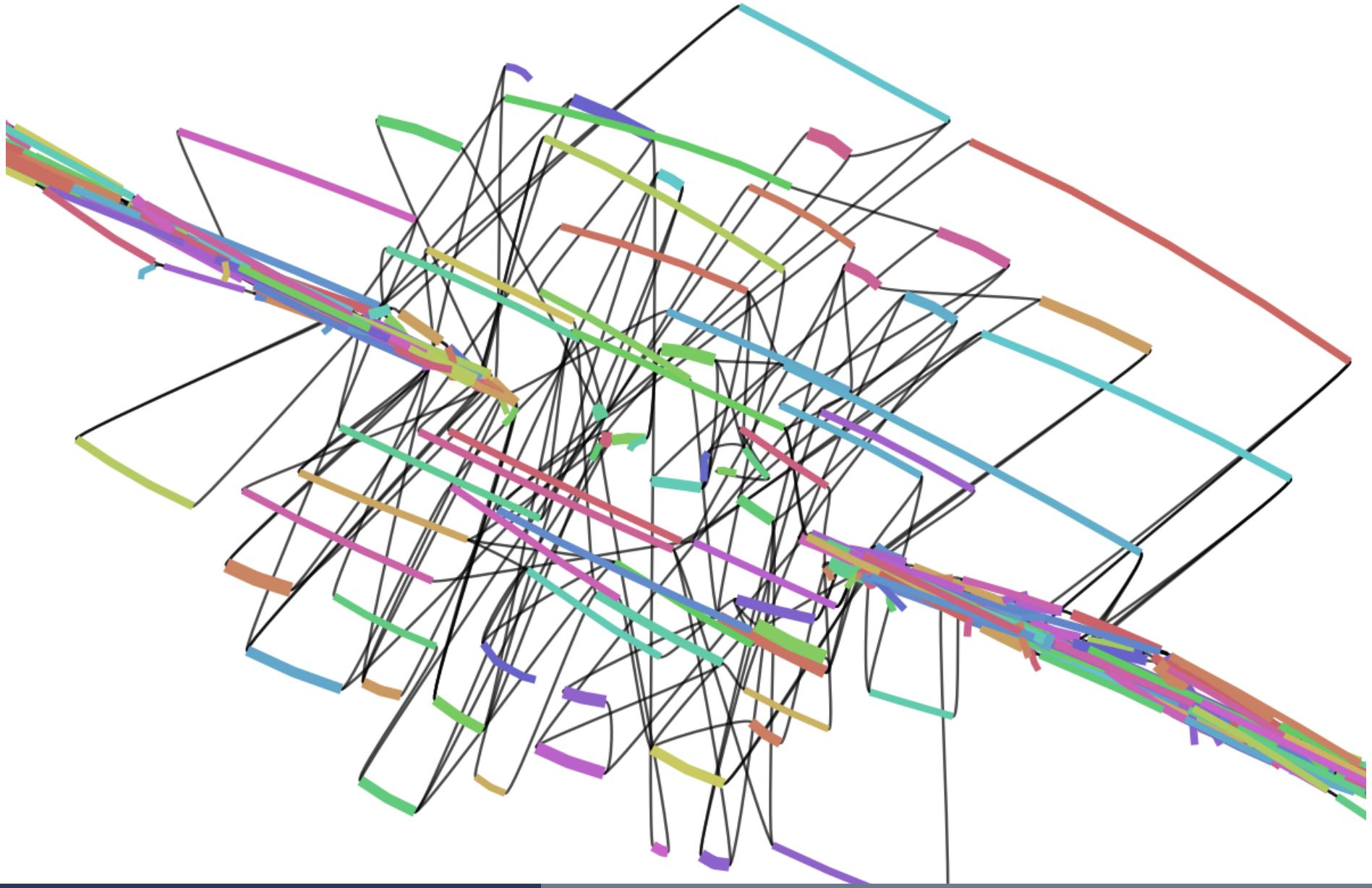


Graphical representation
(.gfa plot using Bandage):



- 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

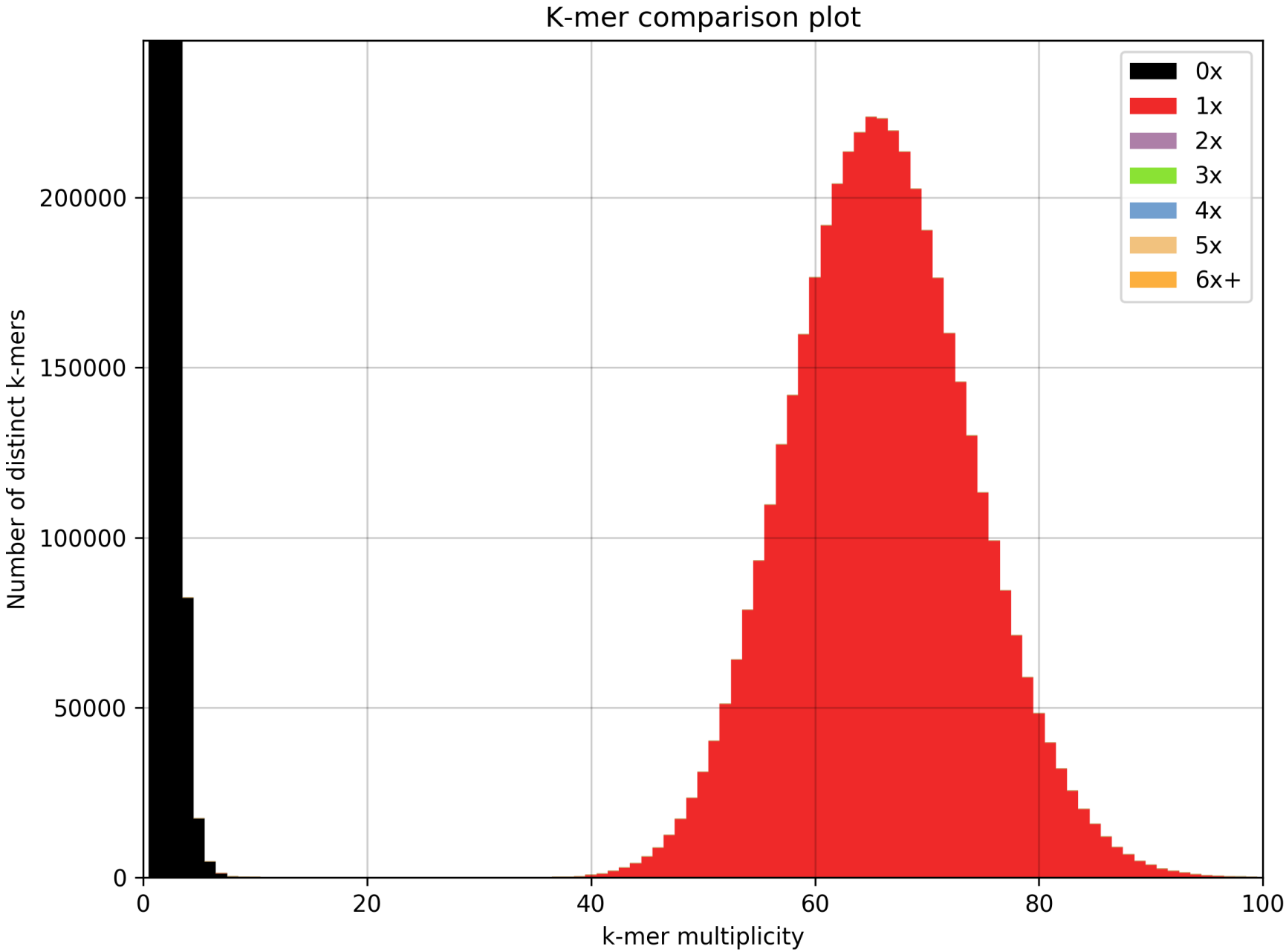
Reads:

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

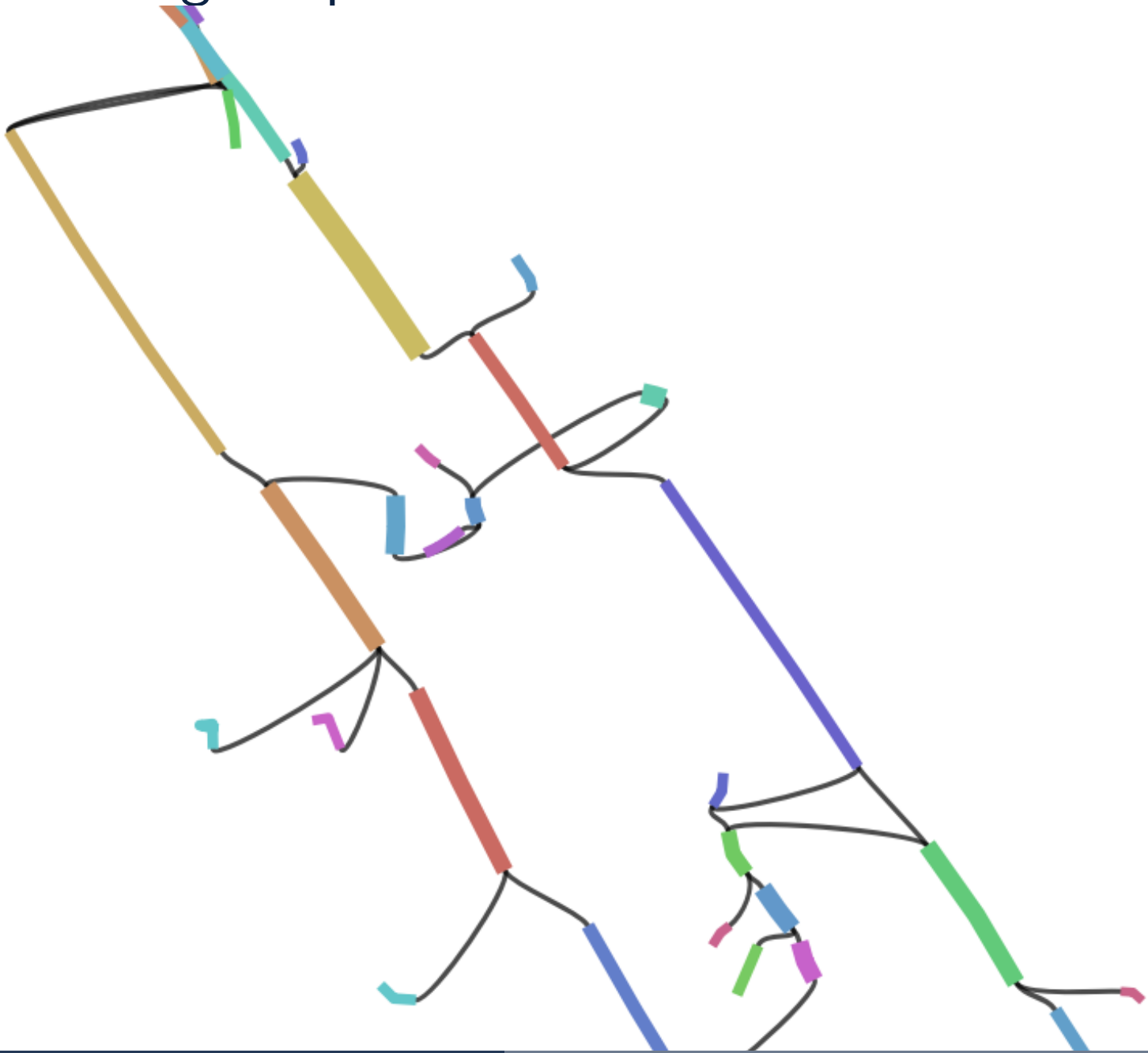
```
9 times   TCTGAAT
1 time    TCTAAAT

6 times           CAACGGA
1 time            CAACGGT
```

Erroneous k-mers are seen less than genomic ones

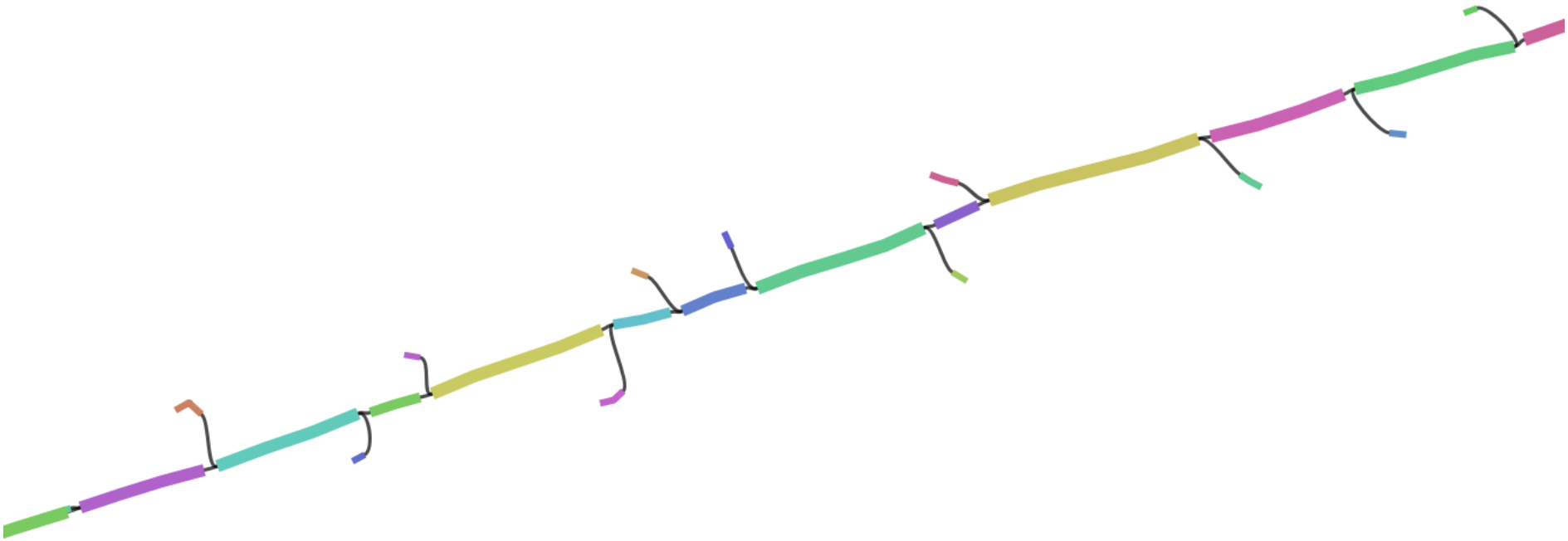
- *K*-mer histogram



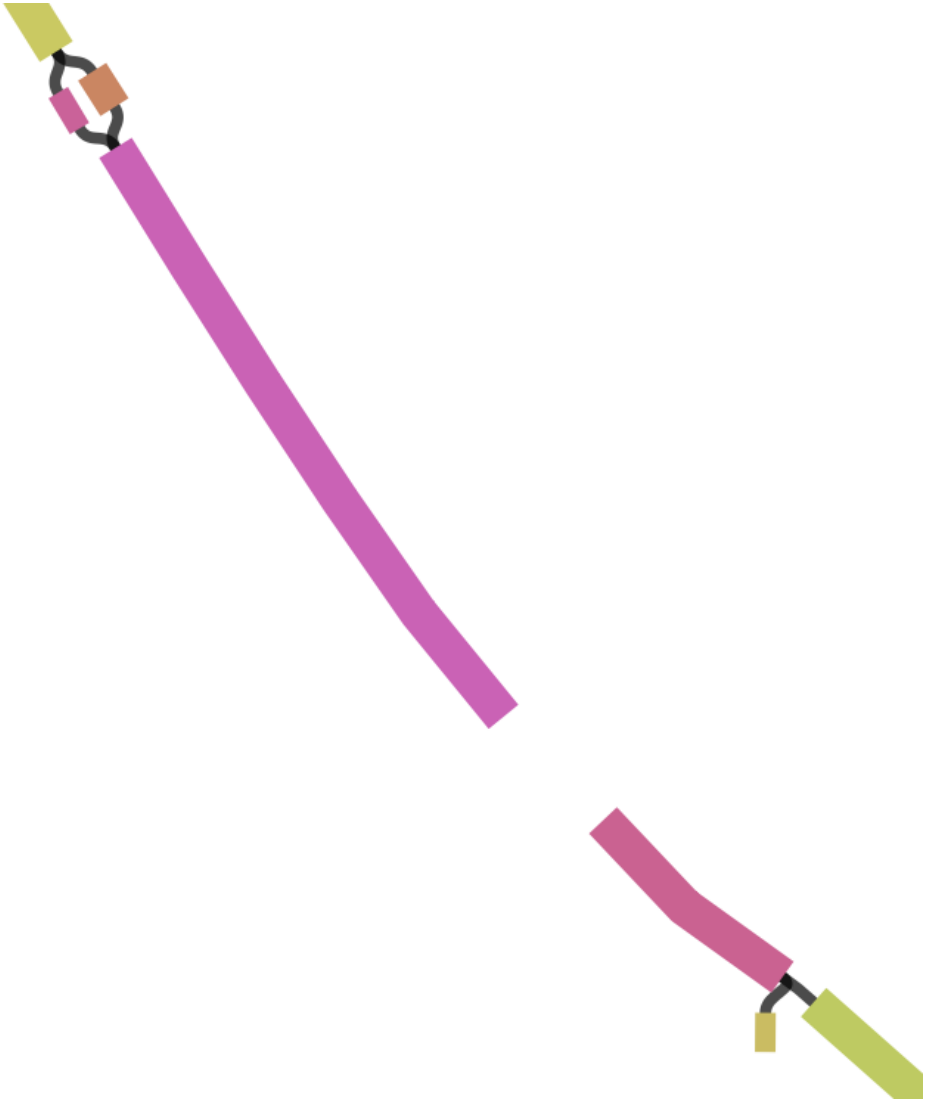
- Removing unique k-mers



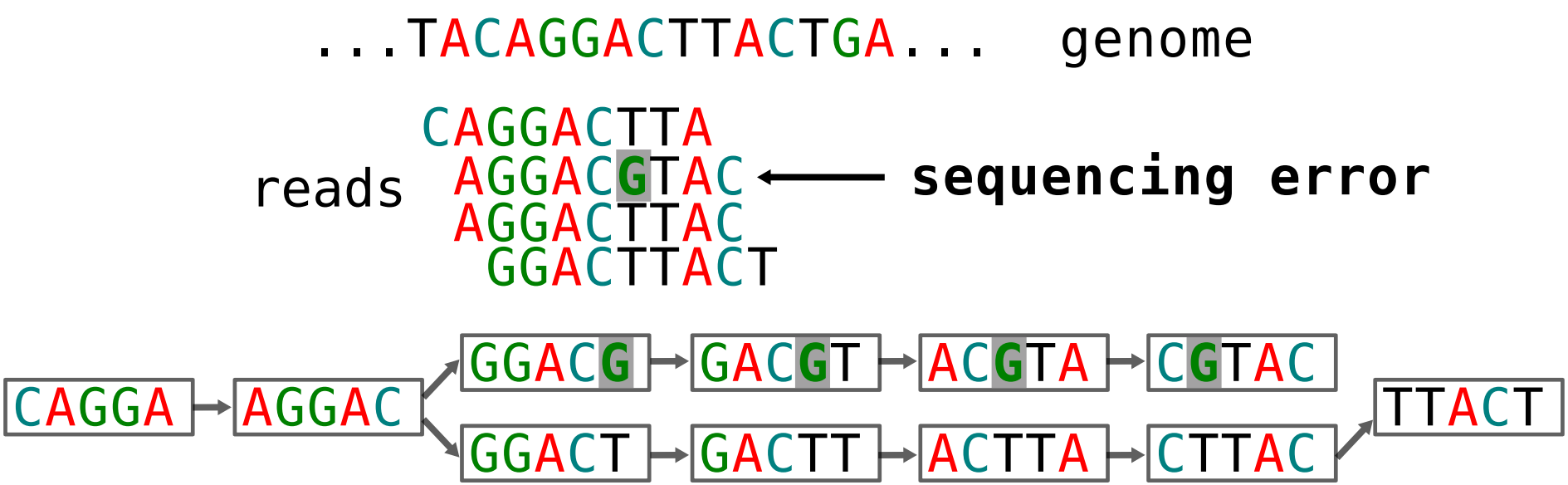
- Removing k-mers seen less than 3 times



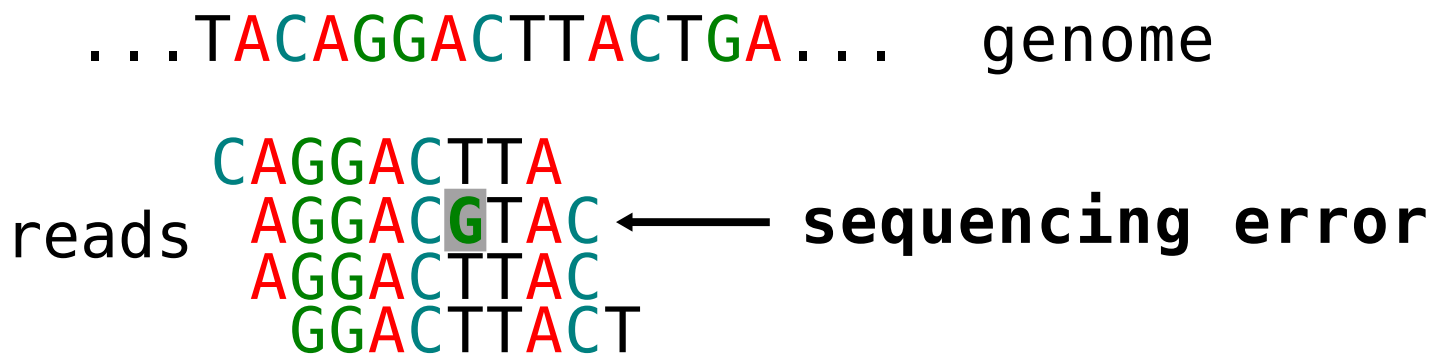
- Removing k-mers seen less than 4 times

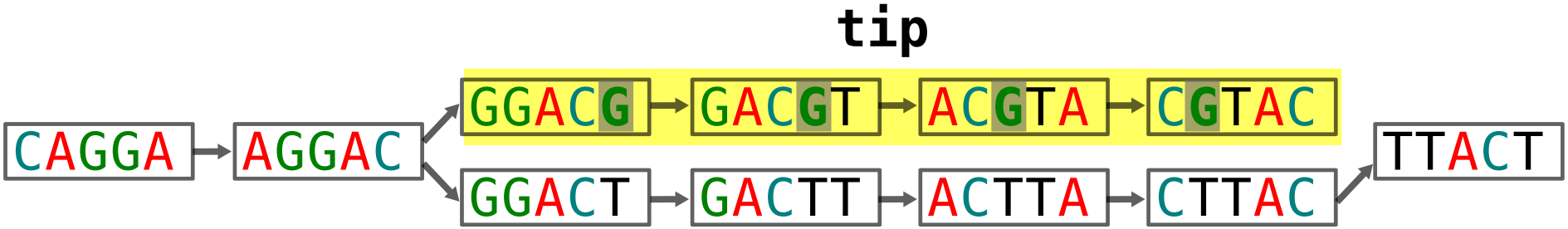


- Errors in de Bruijn graphs



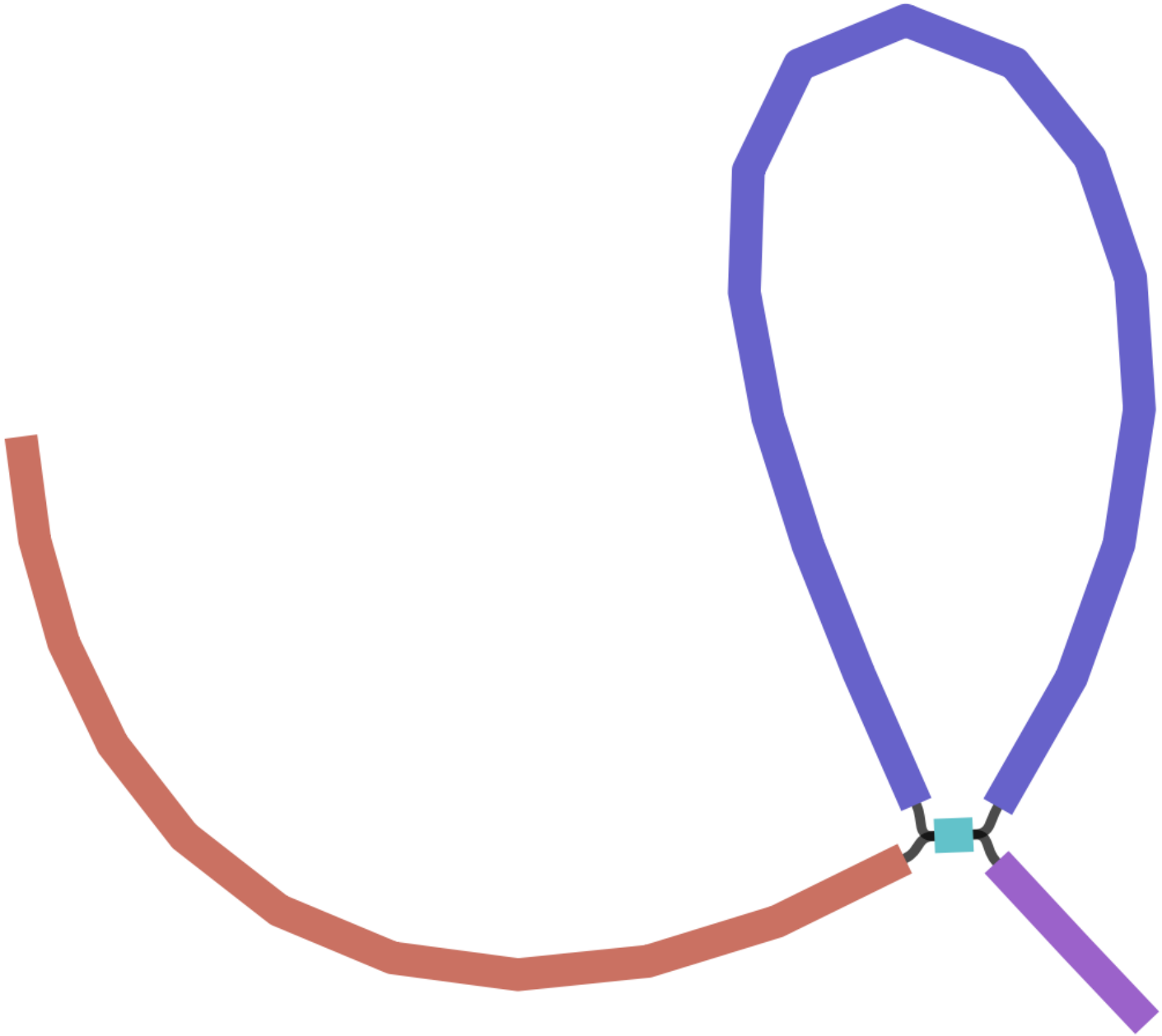
- Errors in de Bruijn graphs





• Errors in de Bruijn graphs

- (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

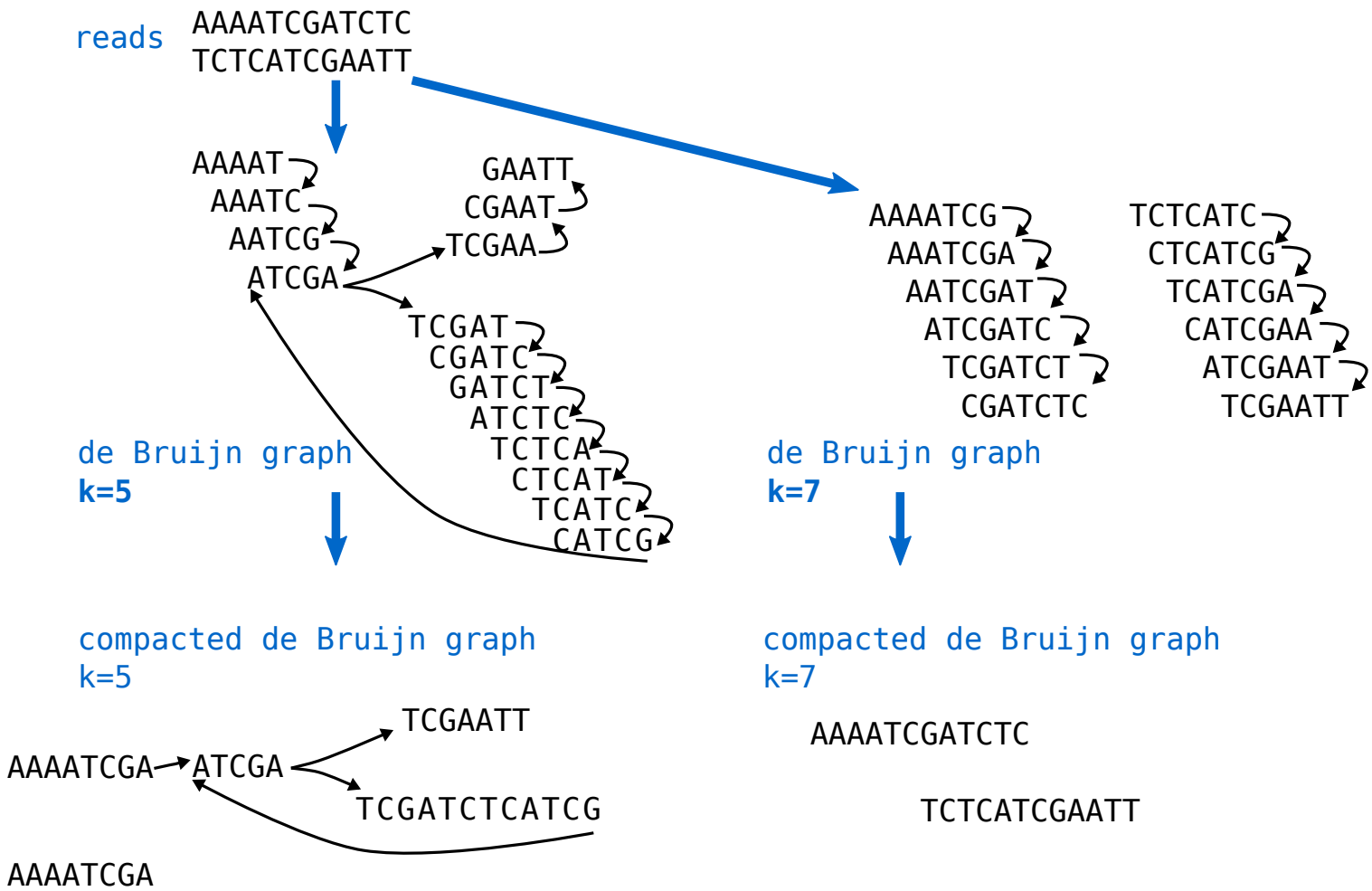
Most de Bruijn graph assemblers can now perform several assemblies with different k-mer sizes to produce an improved "super" assembly

Exercice

Build DBG with k=5 and k=7 from those reads
AAAATCGATCTC
TCTCATCGAATT

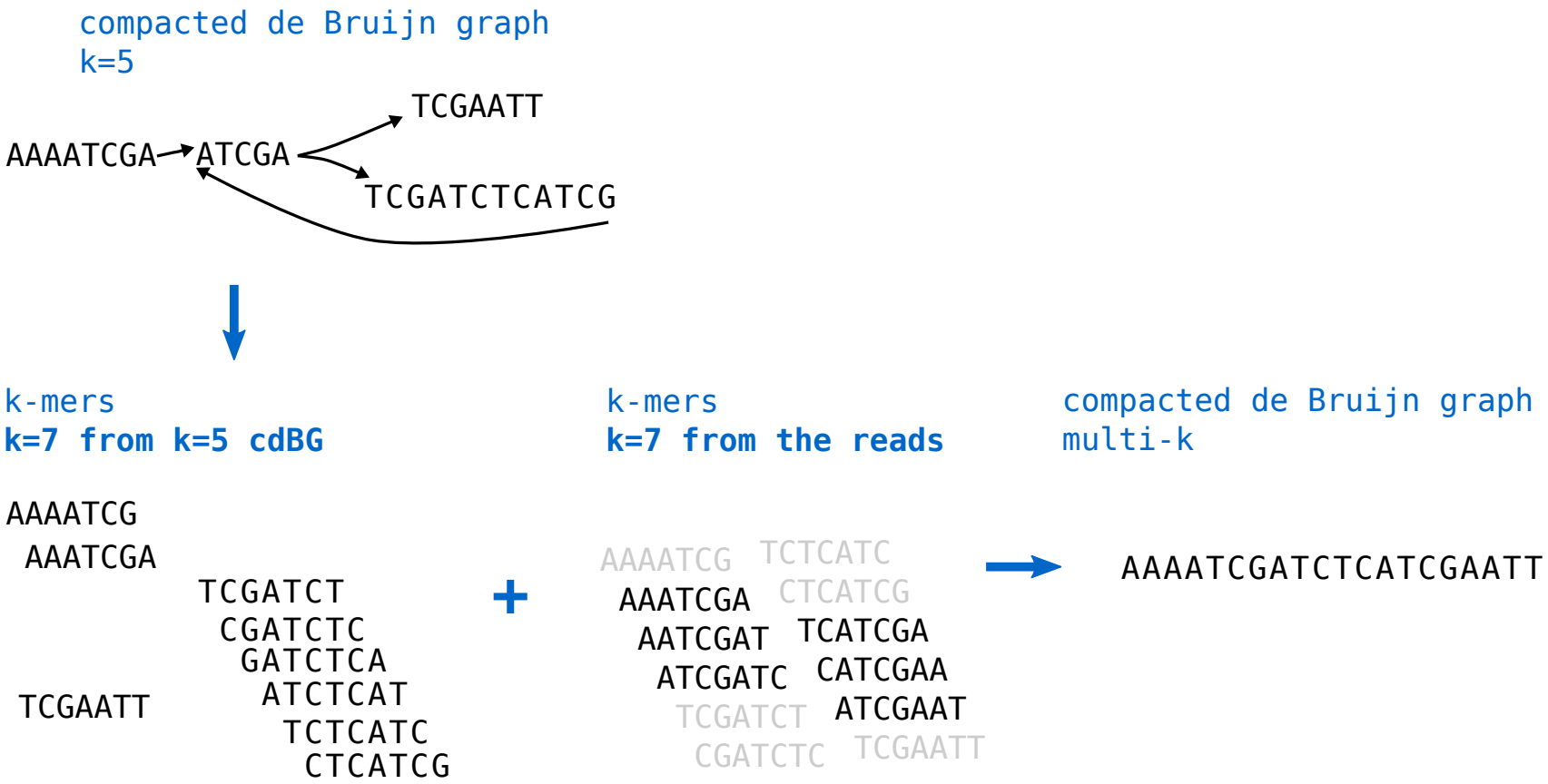
Multiple k assembly

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

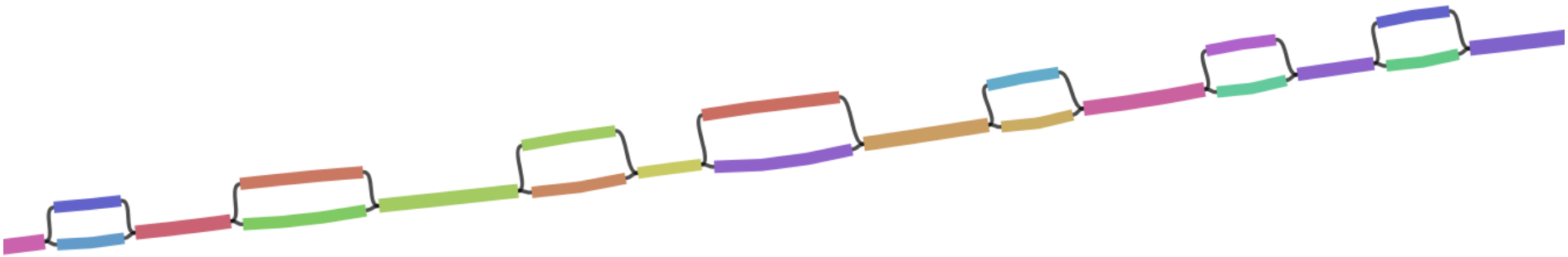


Multiple k assembly

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



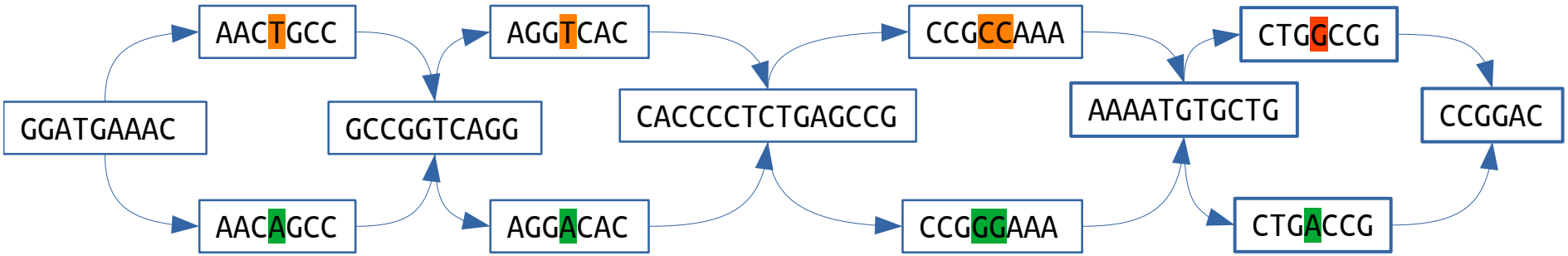
- de Bruijn graph on an eukaryota



- Two or more genomes per individual

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

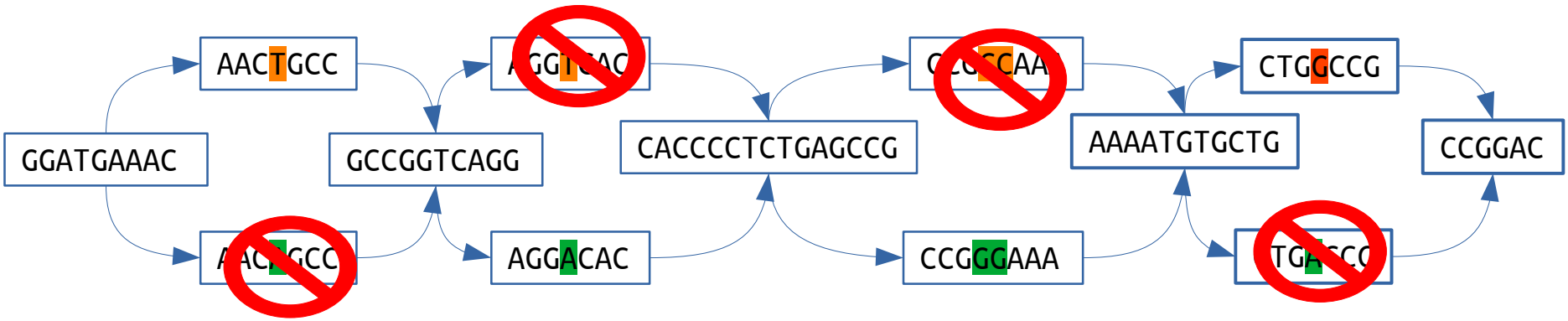
♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC



Two or more genomes per individual

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC



Assembly:

GGATGAAACTGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGGCCGGAC

Assembly **concession number 2**: collapse variability

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC

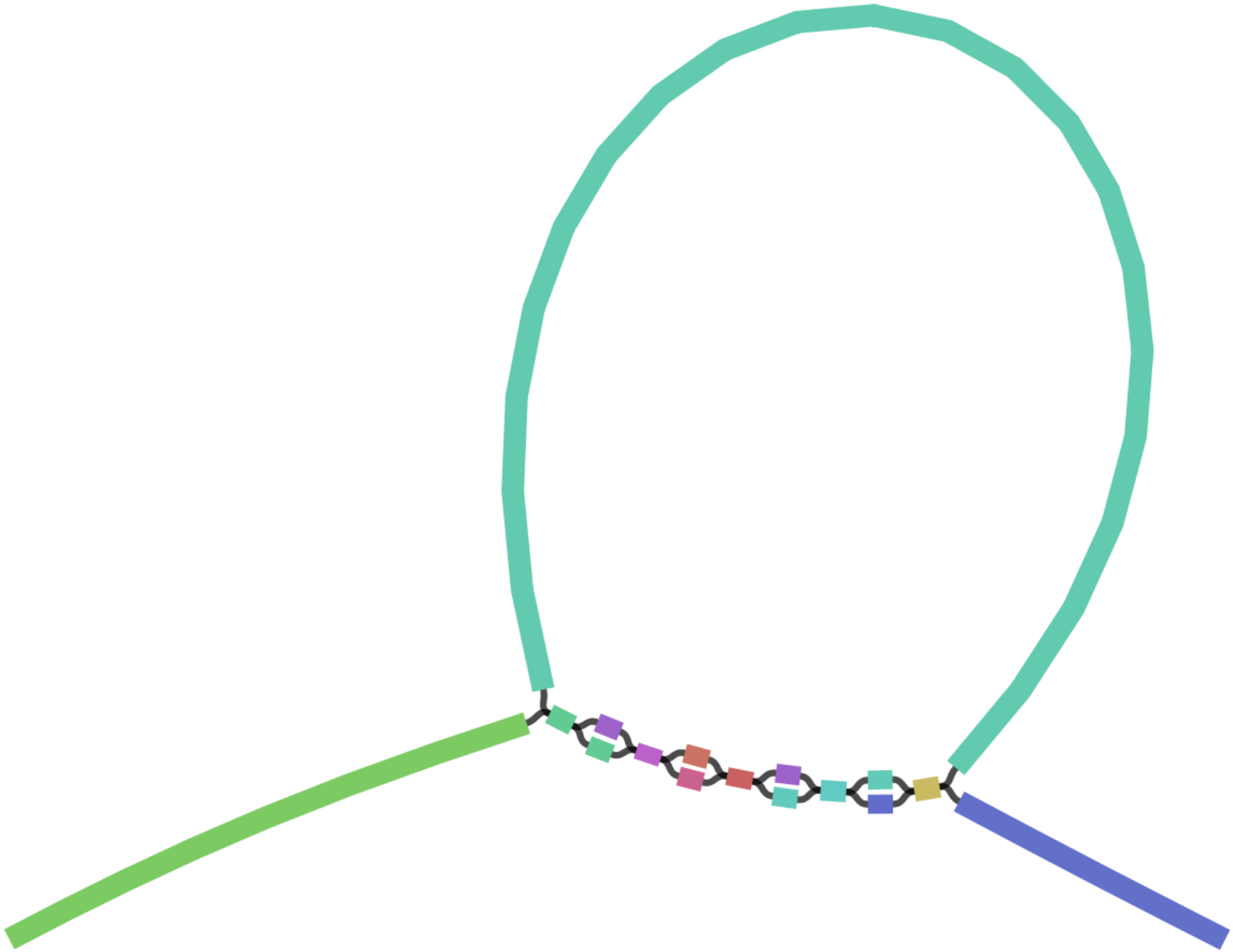
Assembly:

GGATGAAACTGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGGCCGGAC

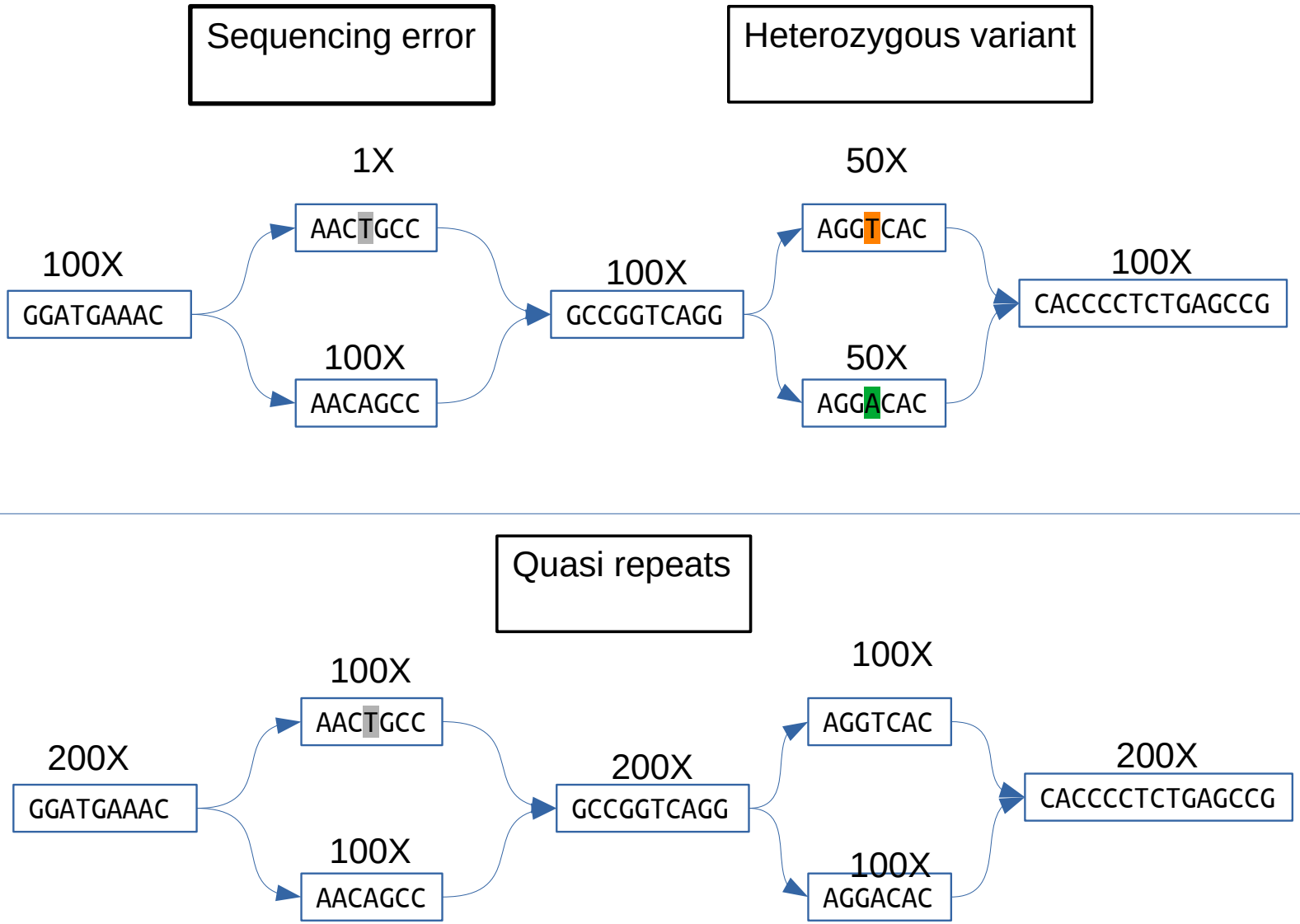
Reads:

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

- Paralog genes/repeats



Paralog genes/repeats in graph



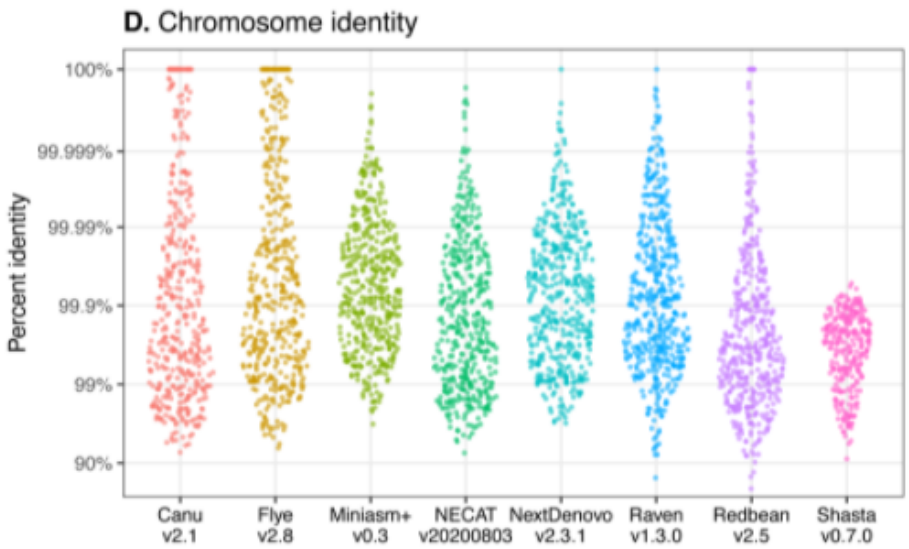
- An assembler is a set of heuristics

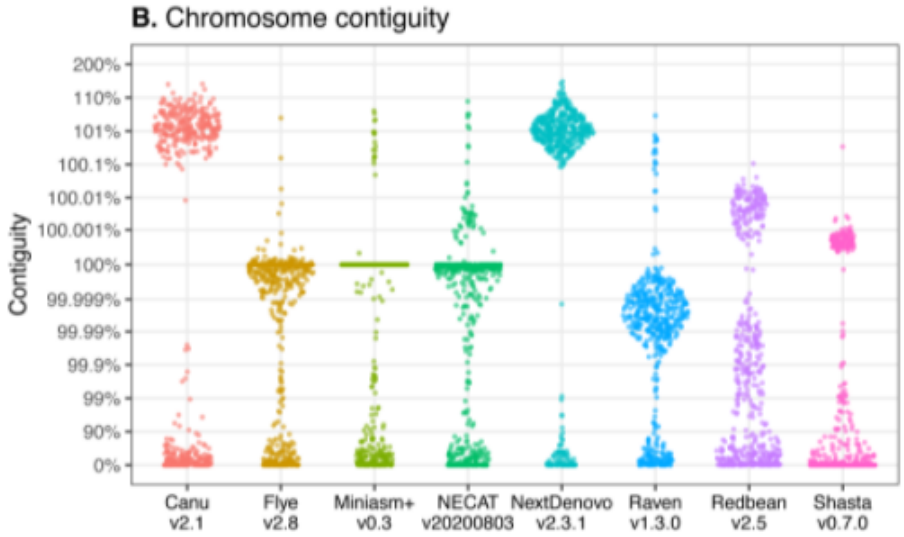
Graph cleaning heuristics

- 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(!)





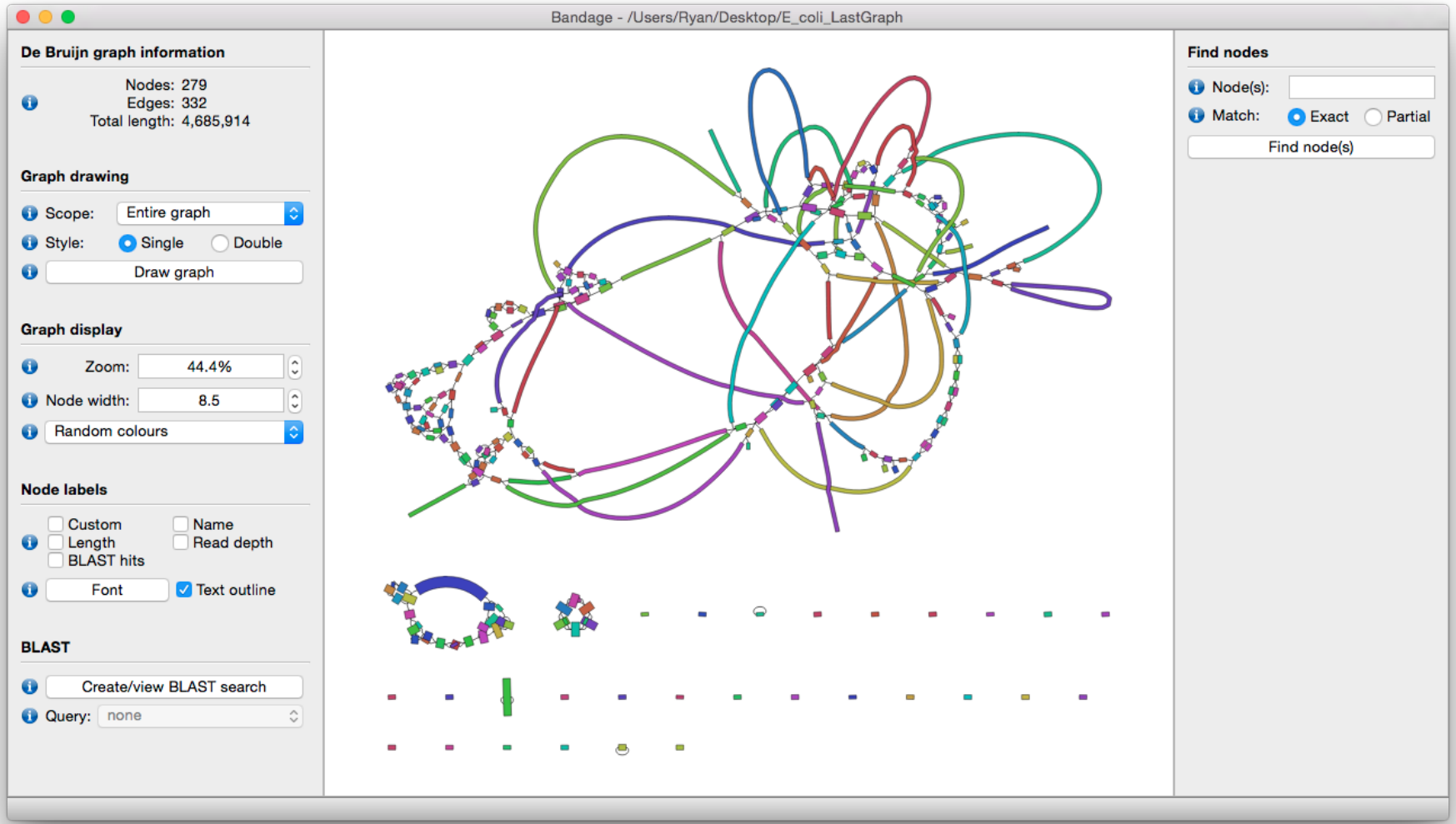
From `github.com/rrwick/Long-read-assembler-comparison`

- What do we do post-assembly?

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

Visualize assembly

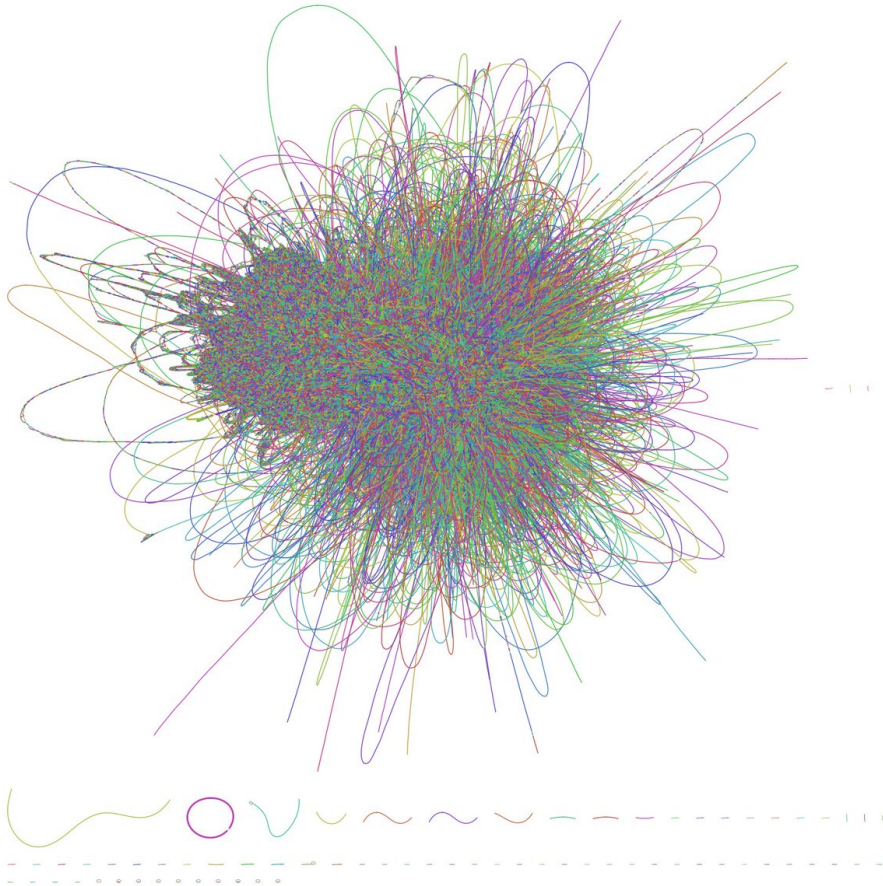
Bandage tool can visualize assembly graphs (GFA)



From `rwick.github.io/Bandage`

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.

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

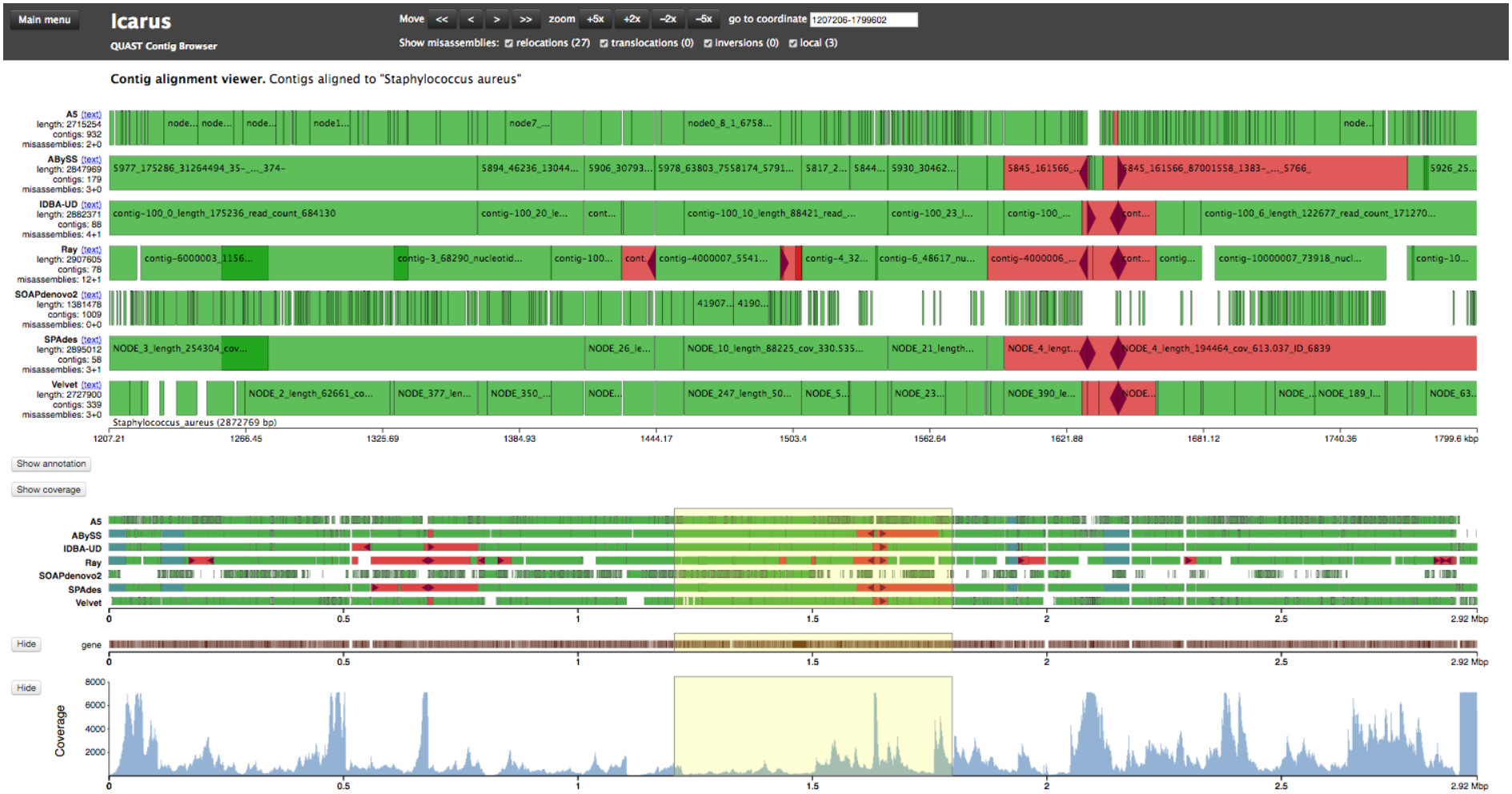
N75 is the same statistic for 75% of the assembly

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

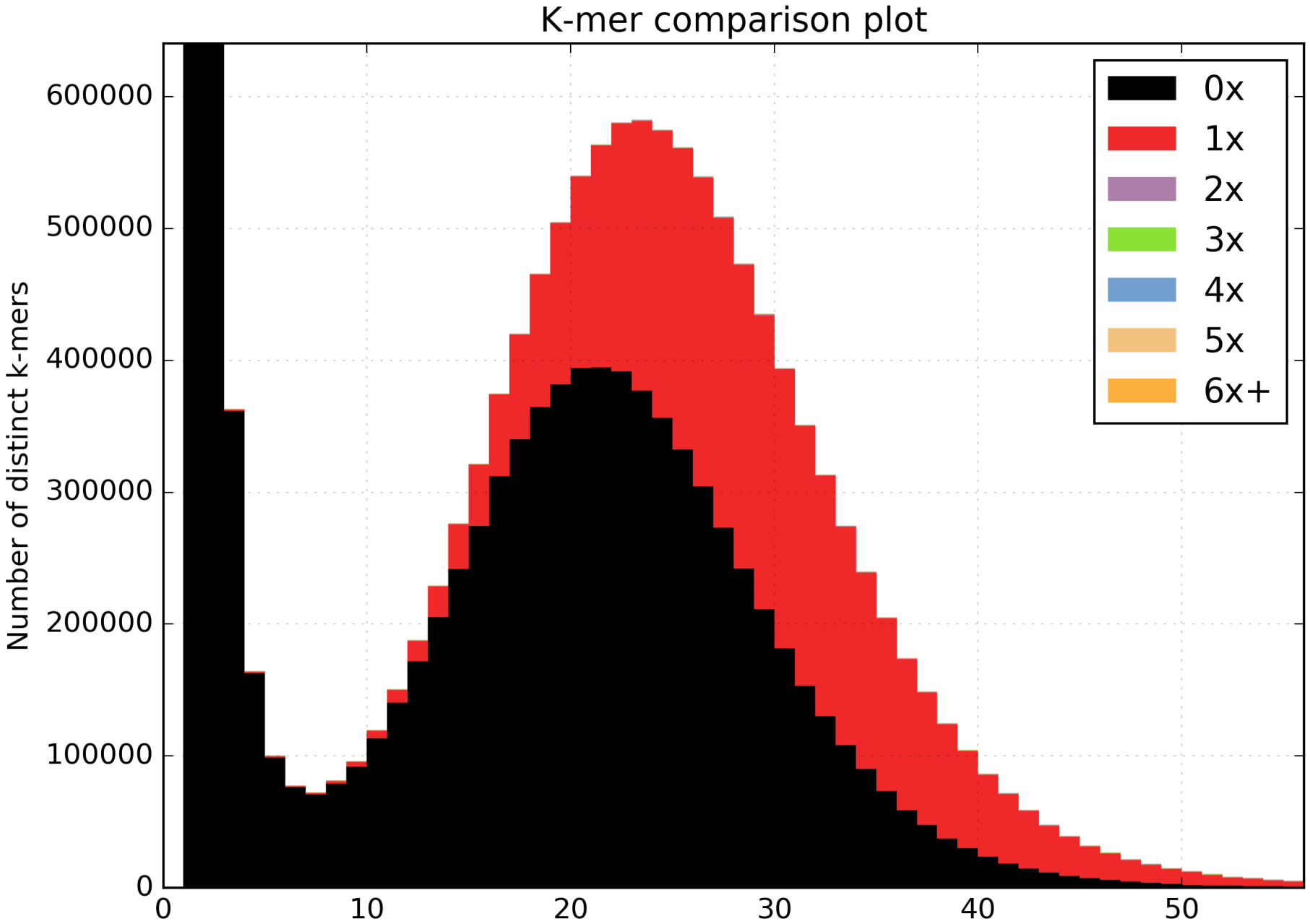
Contigs can be mapped and compared to a reference/closely related genome



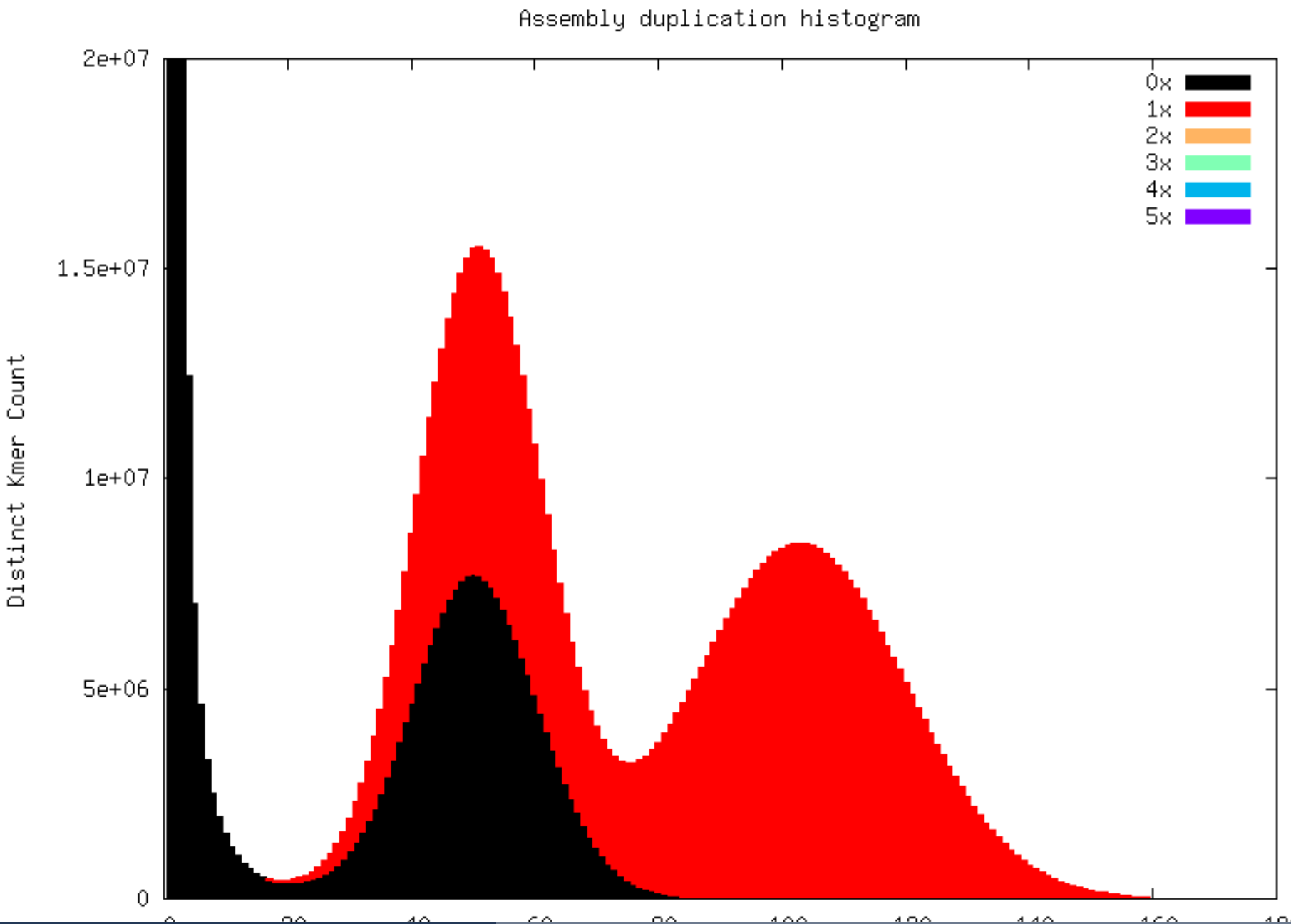
From `quast.bioinf.spbau.ru/manual.html`

- Misassemblies

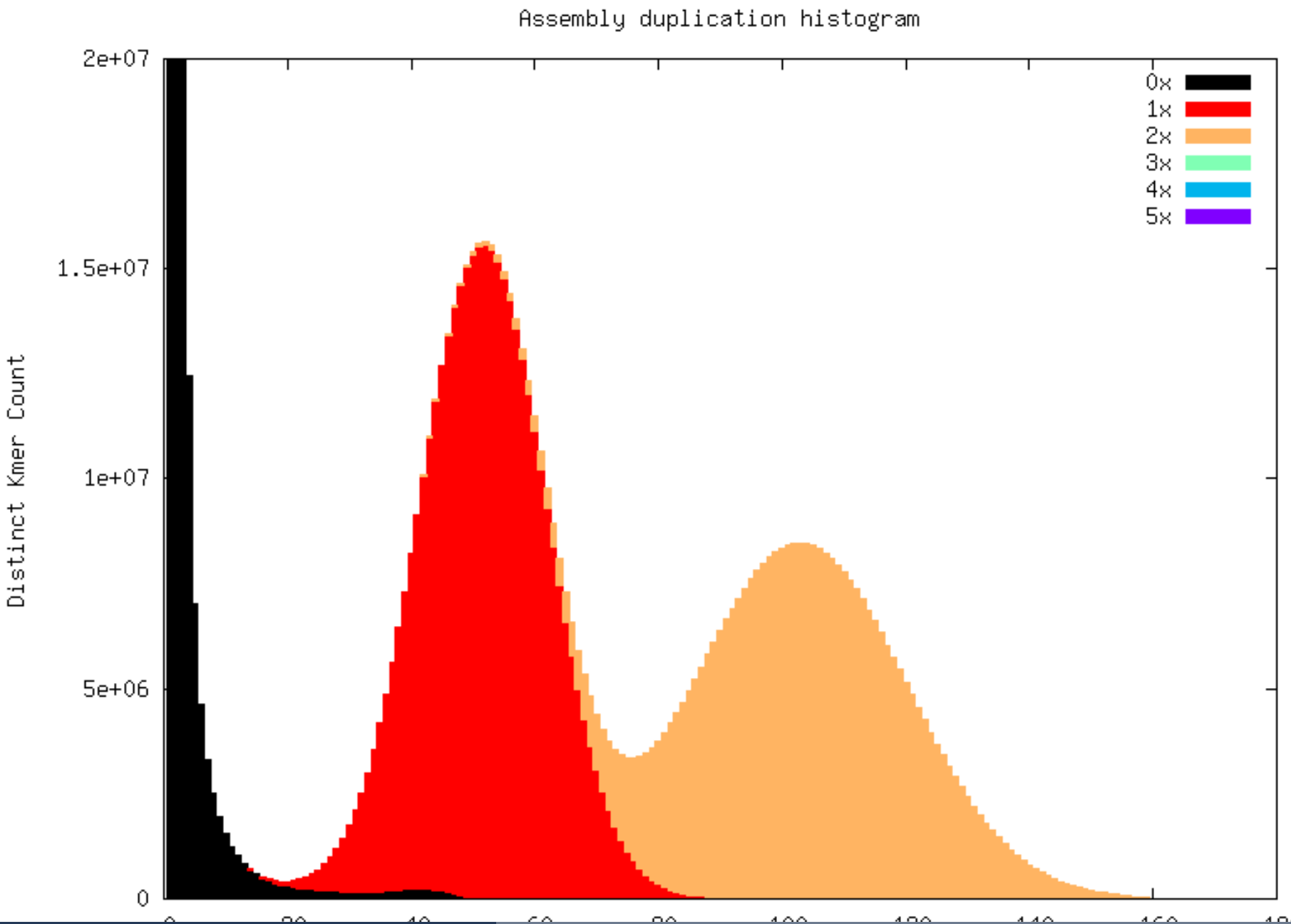
K-mer spectrum visualization with KAT



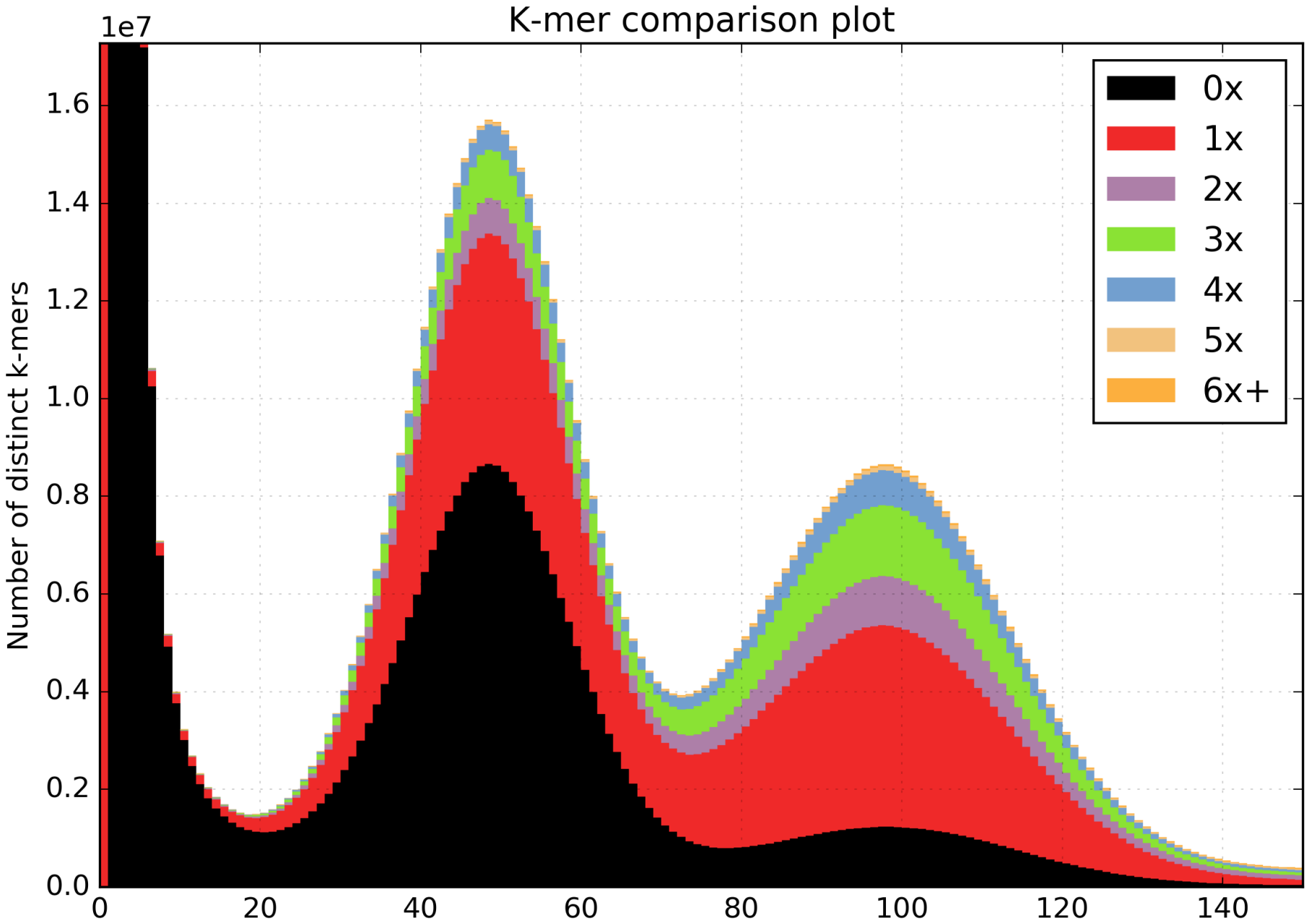
K-mer spectrum visualization with KAT



K-mer spectrum visualization with KAT

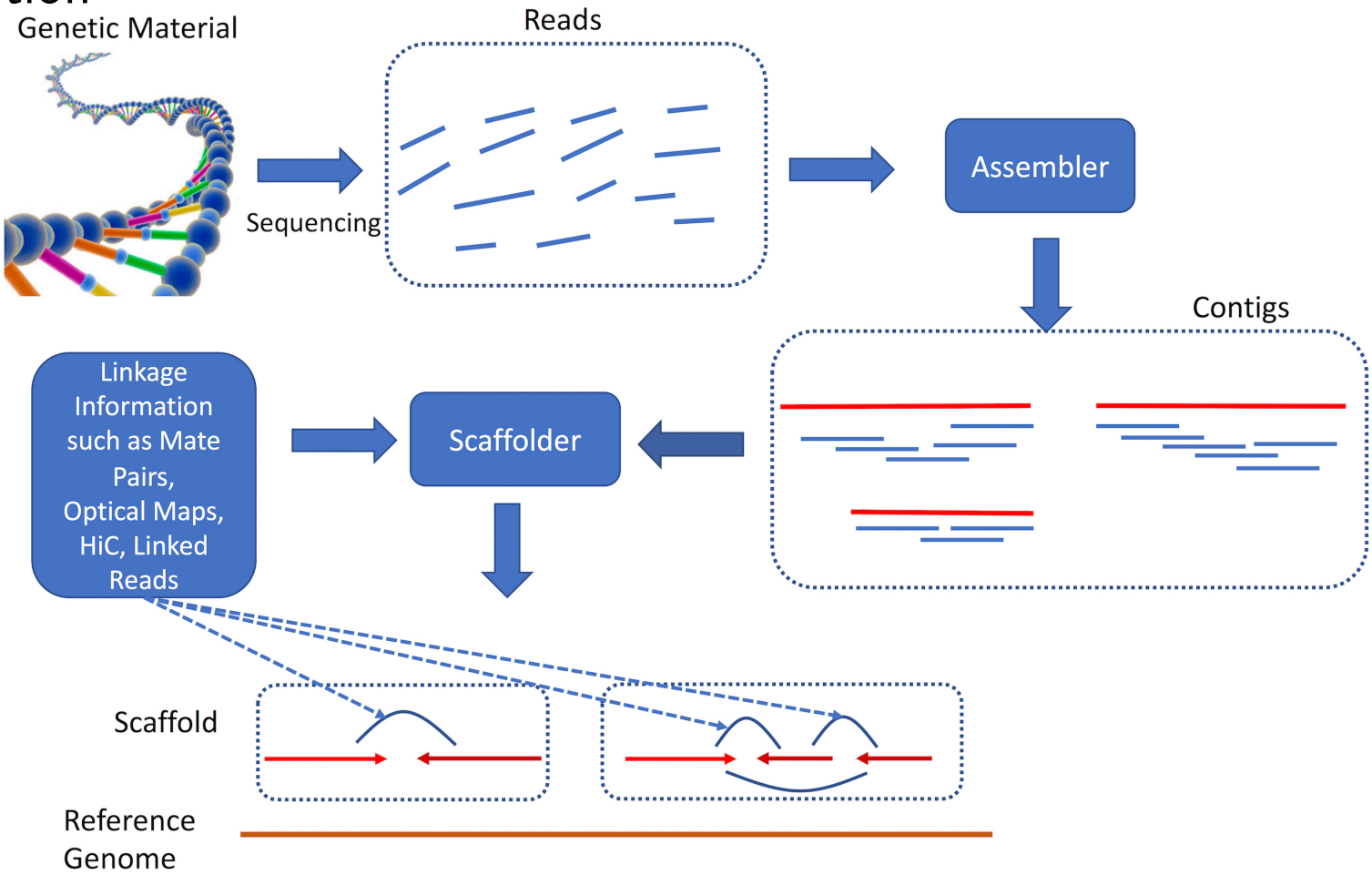


K-mer spectrum visualization with KAT



Scaffolding

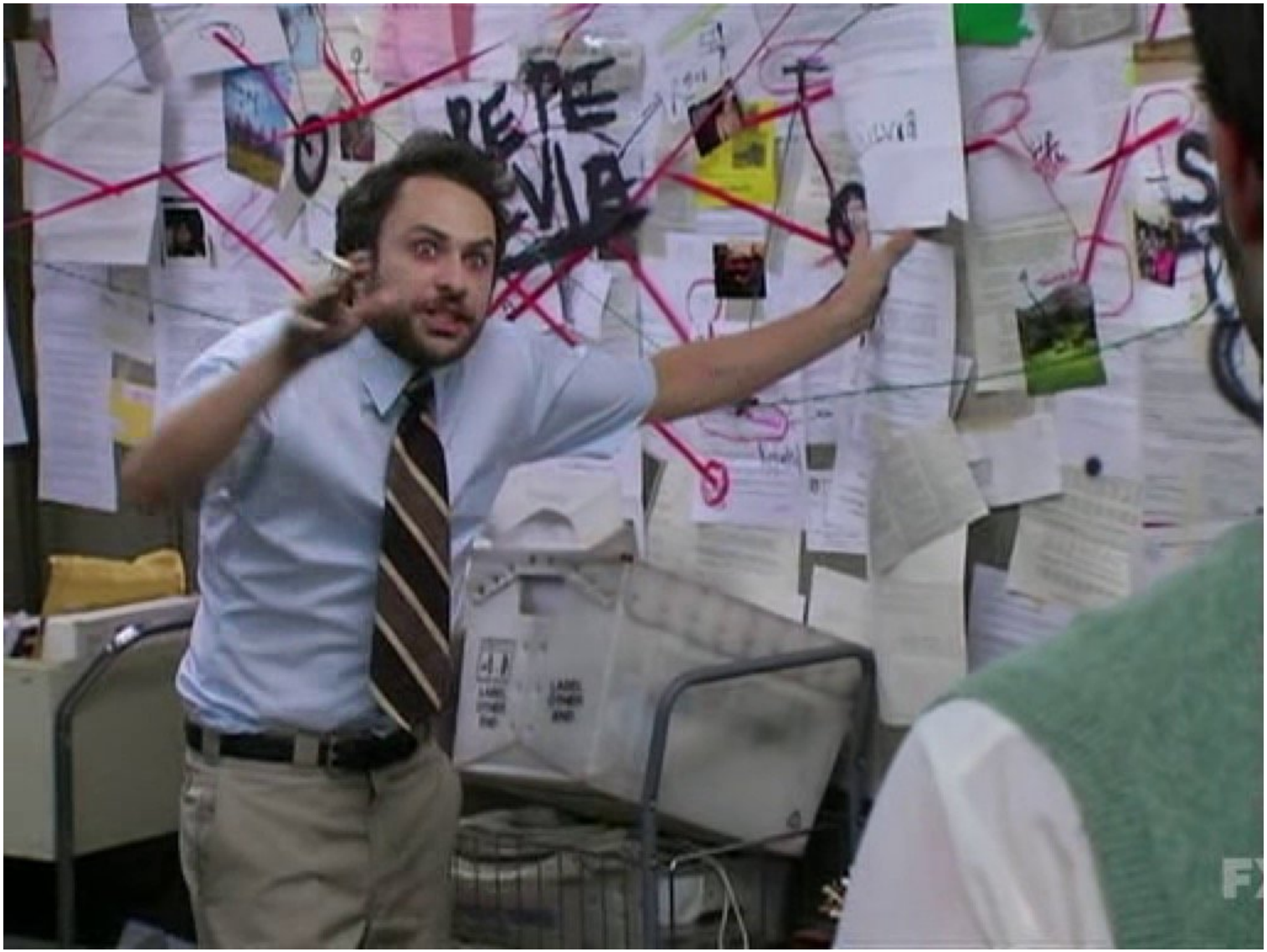
Softwares can improve the assembly continuity by using other kinds of information



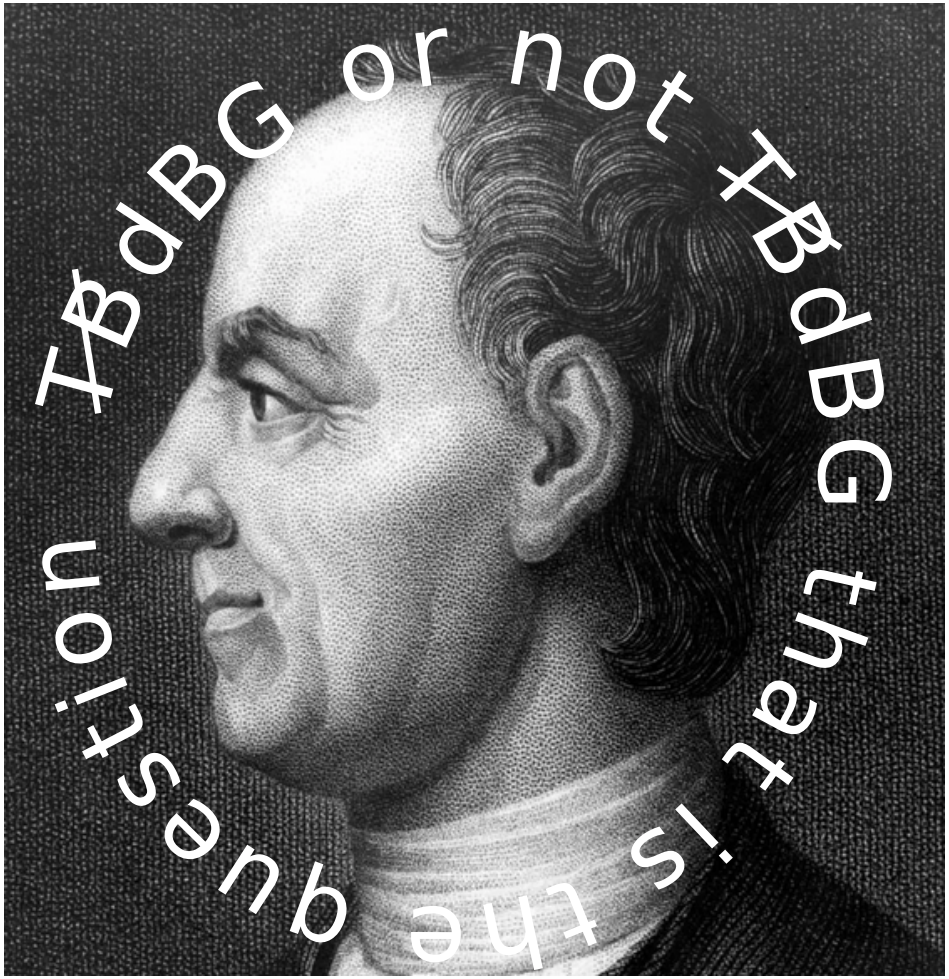
From "Modern technologies and algorithms for scaffolding assembled genomes" Plos Computational Biology

The end

. . . of the theoritical 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]
- Discovar denovo [Weisenfeld 2014]
- Abyss [Simpson 2009]

Third generation sequencing

- Long reads $\approx 10 - 100kbp$
- 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:

GGTAATGGTTTTTTGGTGCTAATGCGTTTTTTCATGGATGTCGTAATTTTTTATCTG

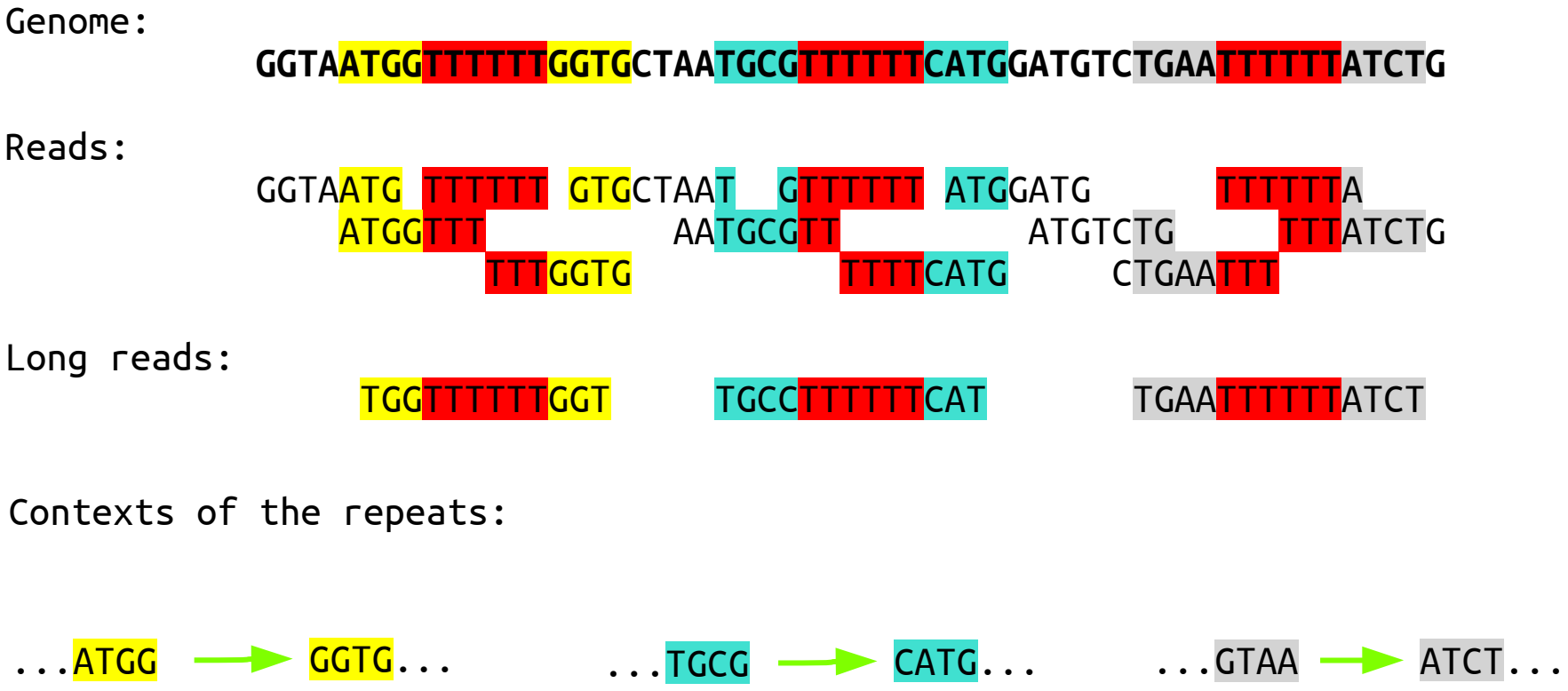
Reads:

```
GGTAATG TTTTTT GTGCTAAT GTTTTTT ATGGATG      TTTTTTA
    ATGGTTT         AATGCGTT         ATGTCGT     TTTATCTG
           TTTGGTG              TTTTCATG     CGTAATTT
```

Contexts of the repeat:

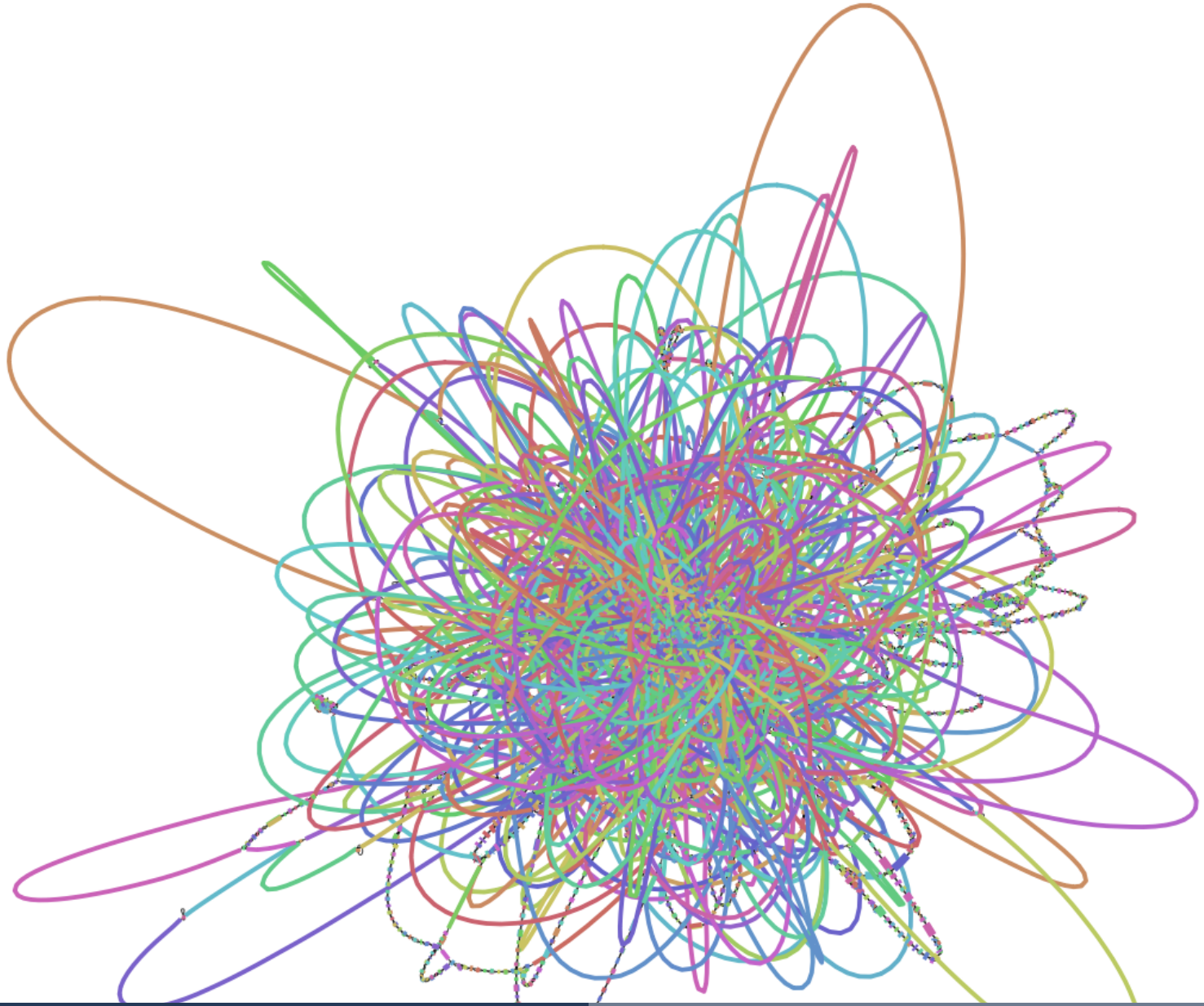
```
...ATGG                                  ATCT...

                  ???TTTTTT???           GGTG...
...TGCG

                                         CATG...
...GTAA
```

Repeats spanning



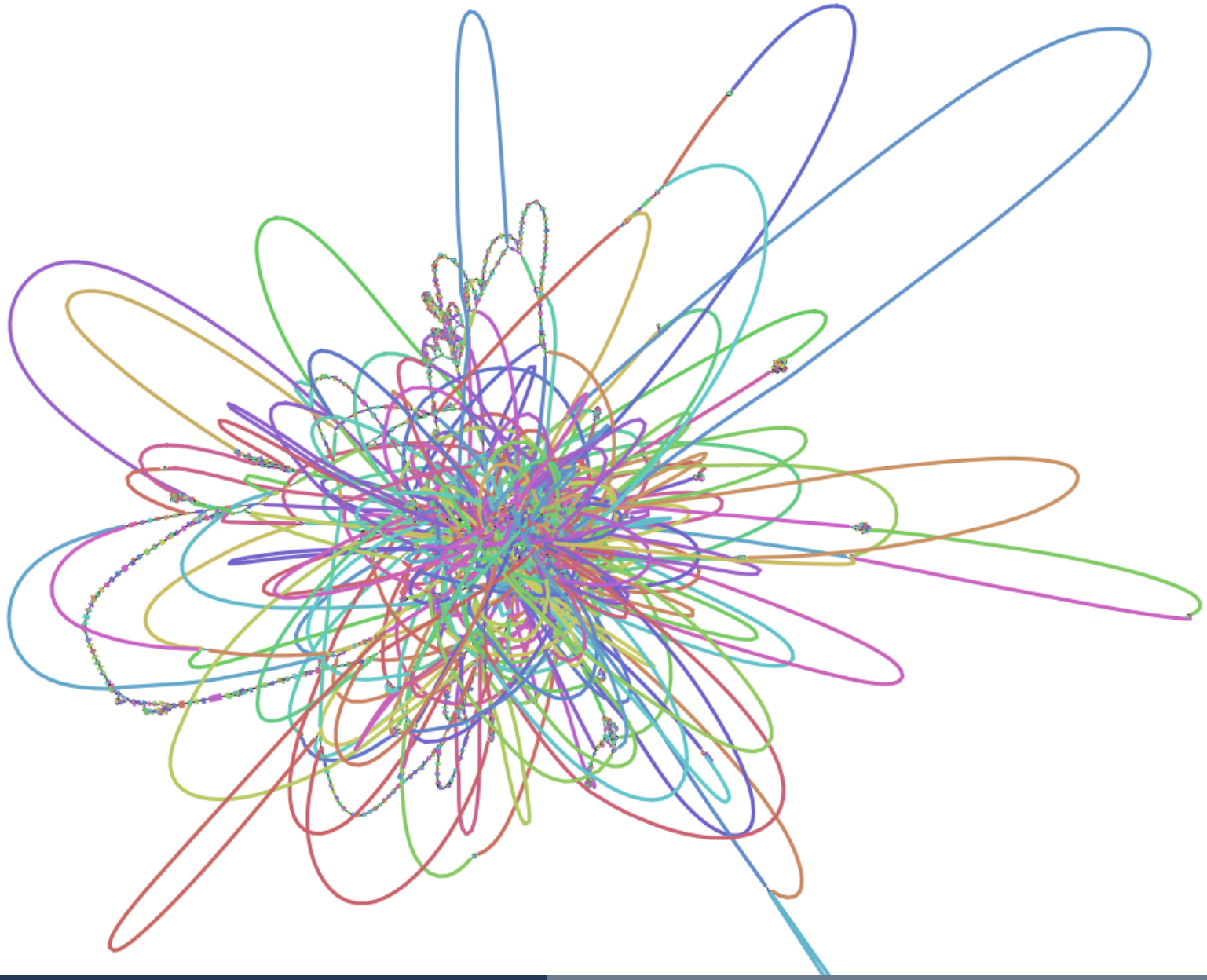
- Read length matters

Read size=21



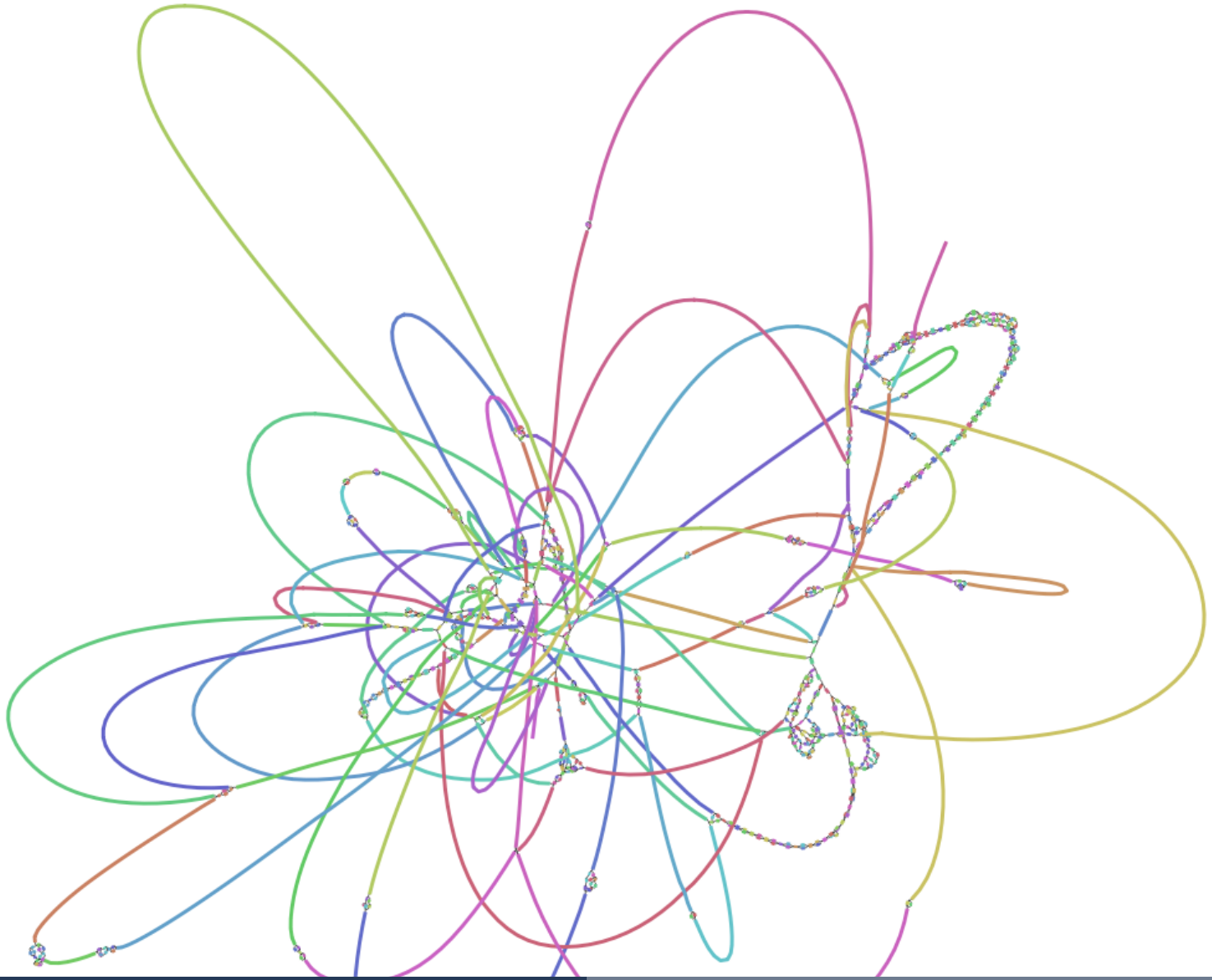
- Read length matters

Read size=31



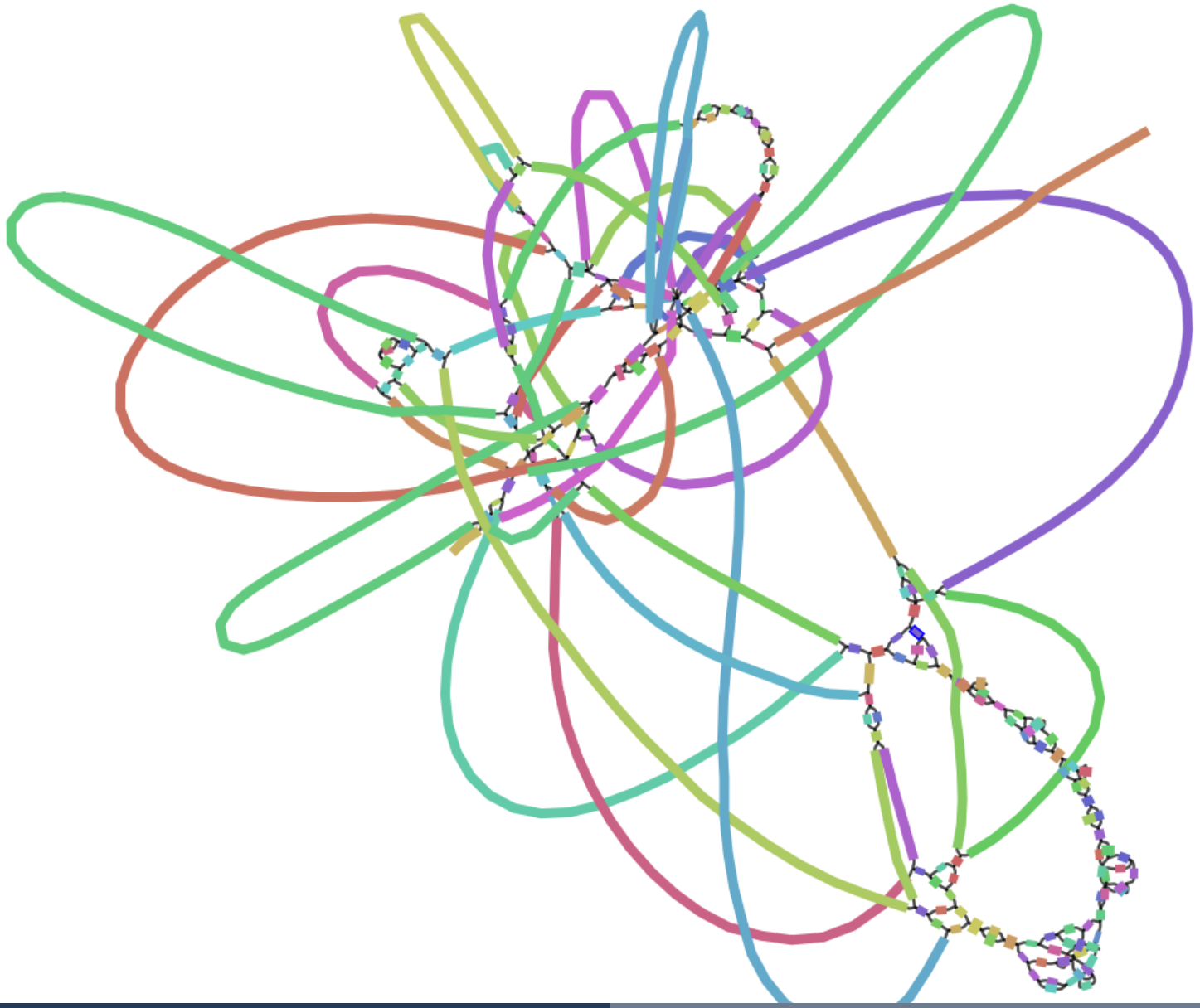
- Read length matters

Read size=63



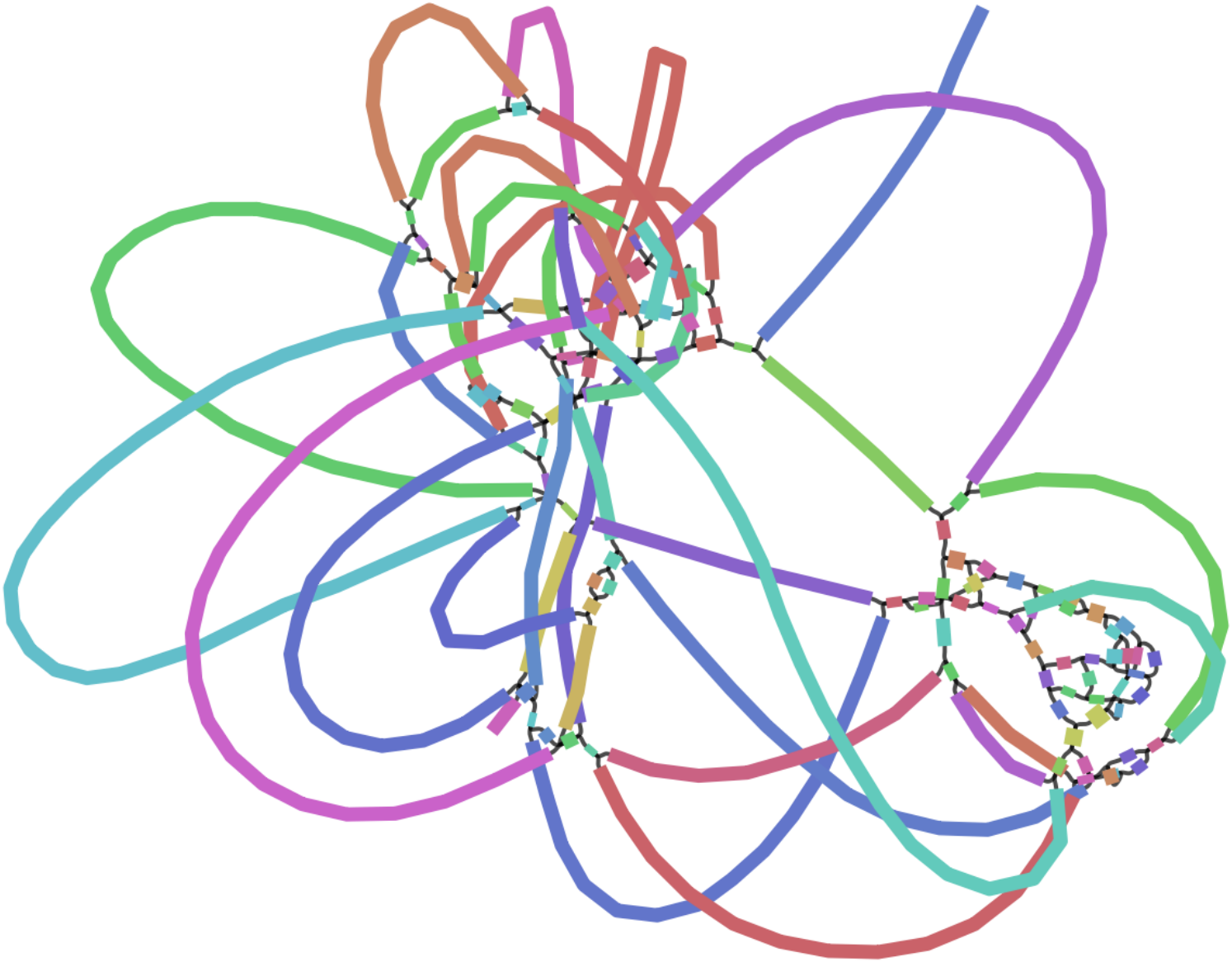
Read length matters

Read size=255



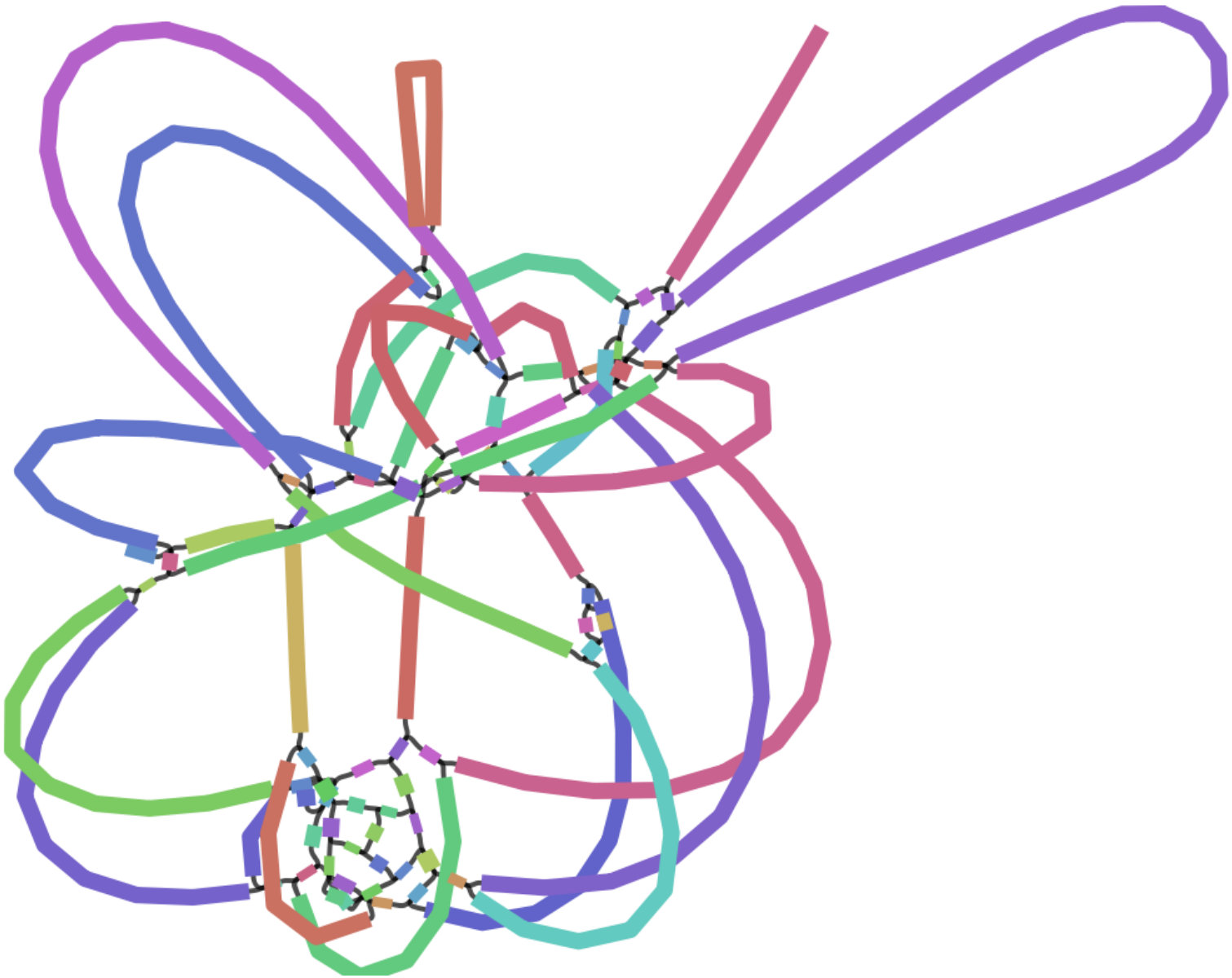
- Read length matters

Read size=500



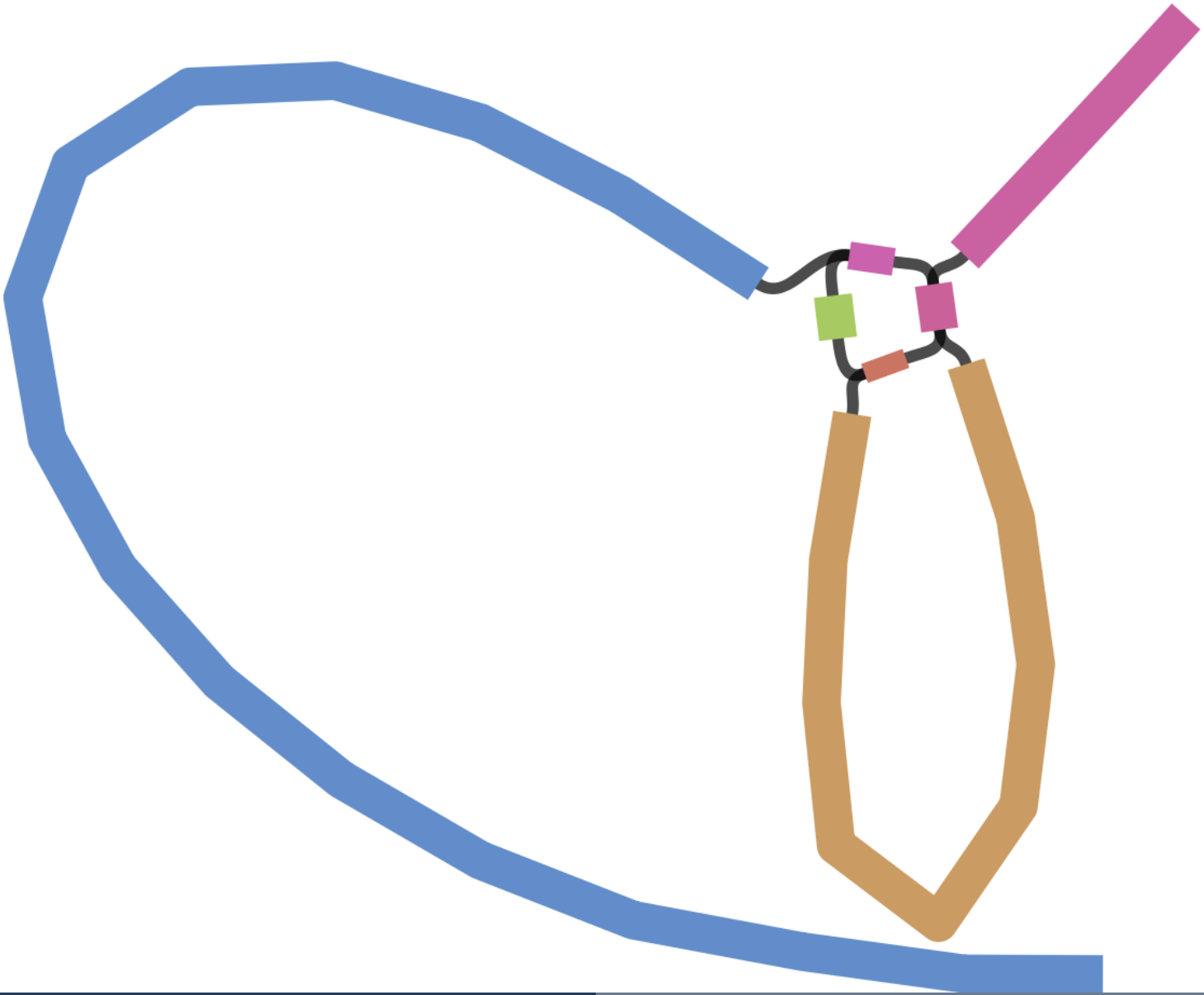
- Read length matters

Read size=1000

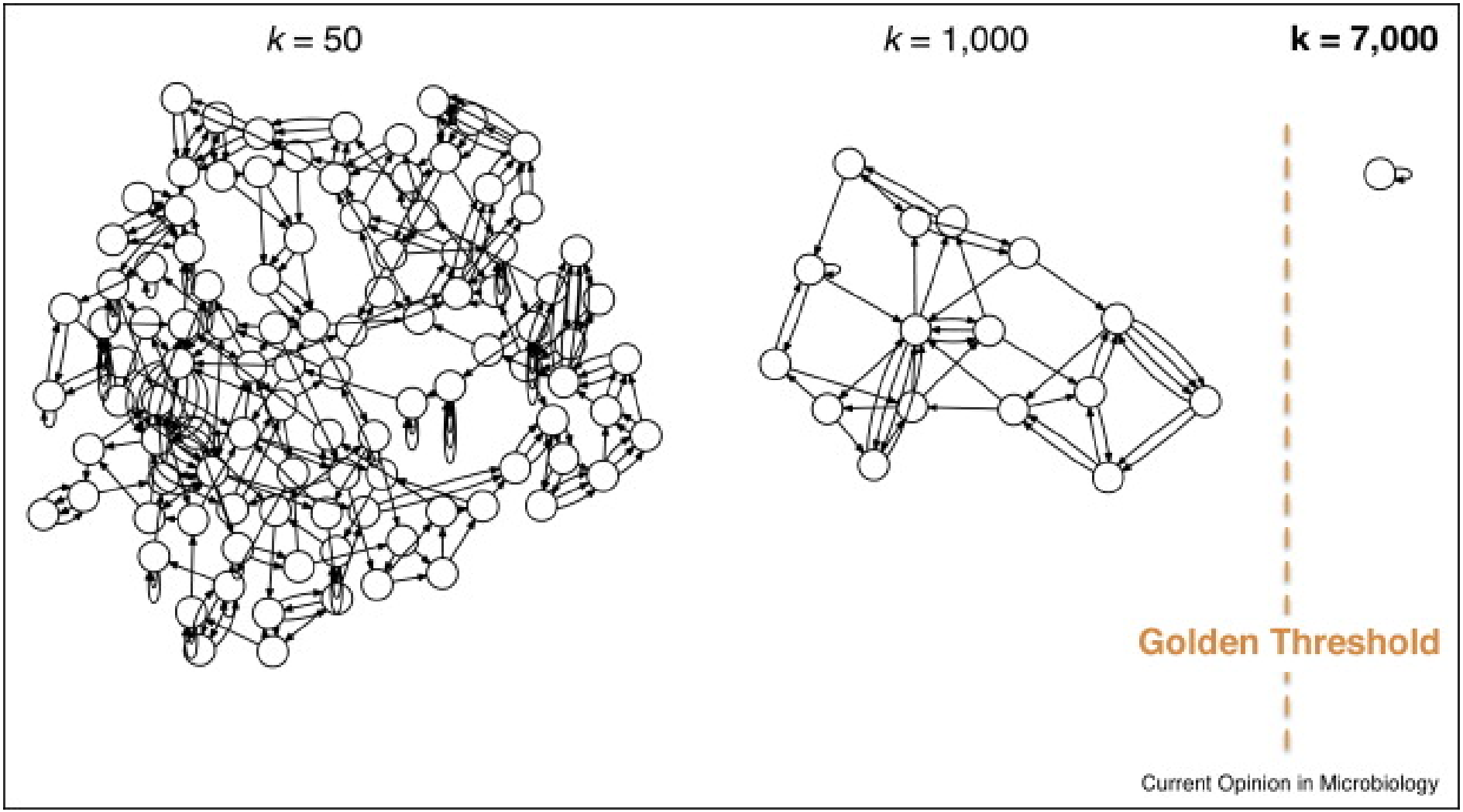


- Read length matters

Read size=2000

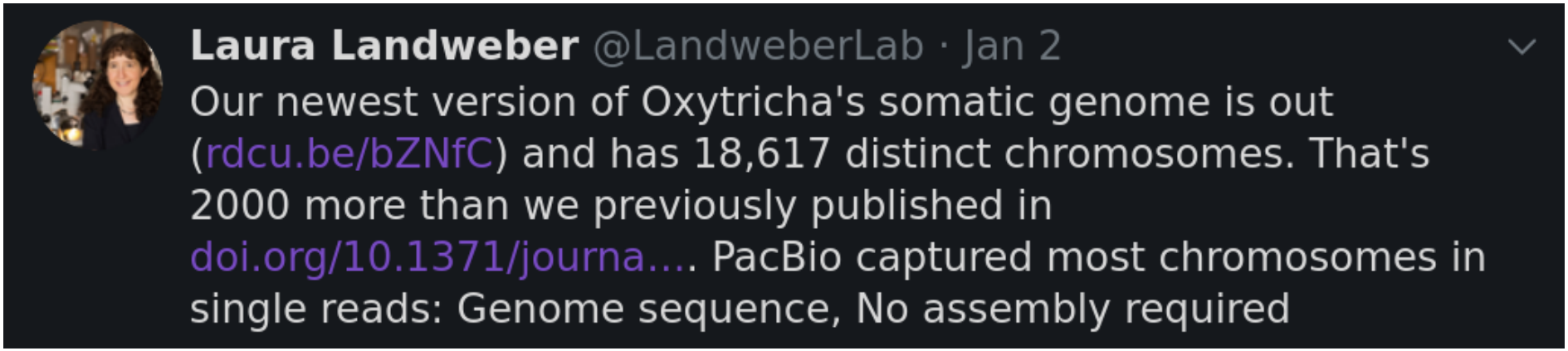


- Great hope for assembly

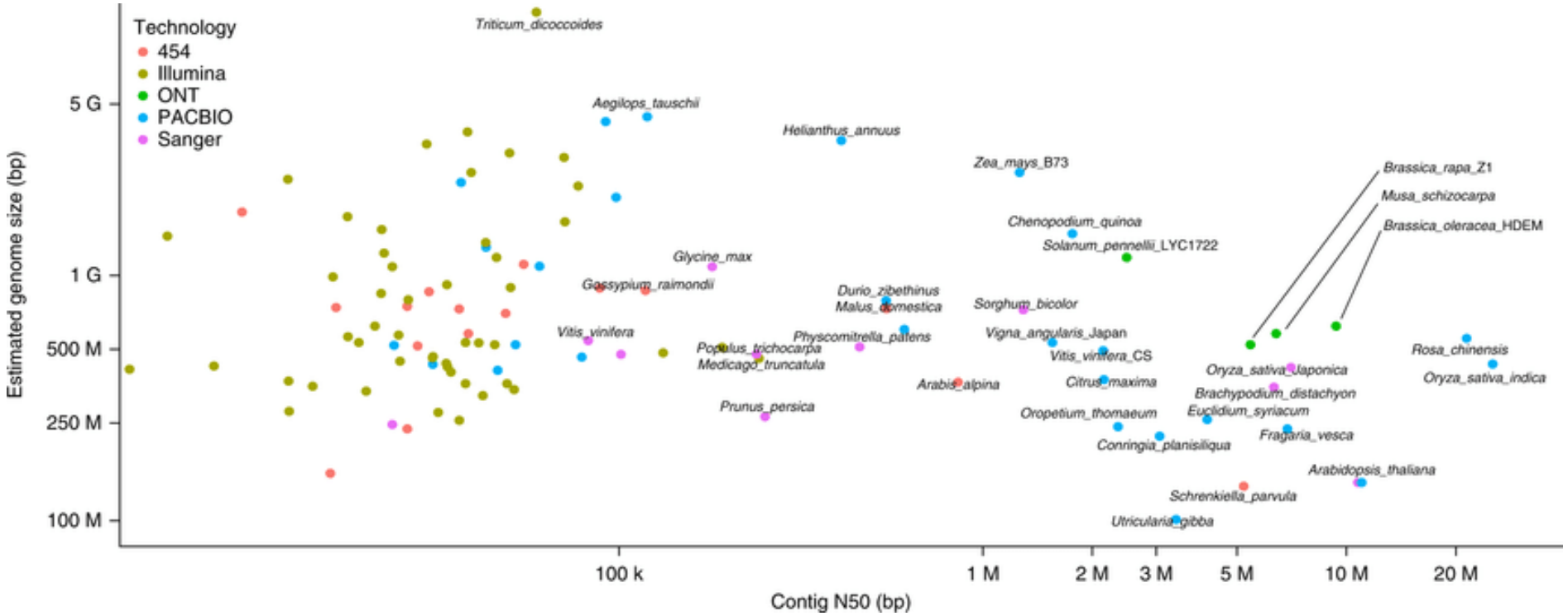


From "One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly" Current Opinion in Microbiology 2015

- Long reads killed the assembly star



- 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?

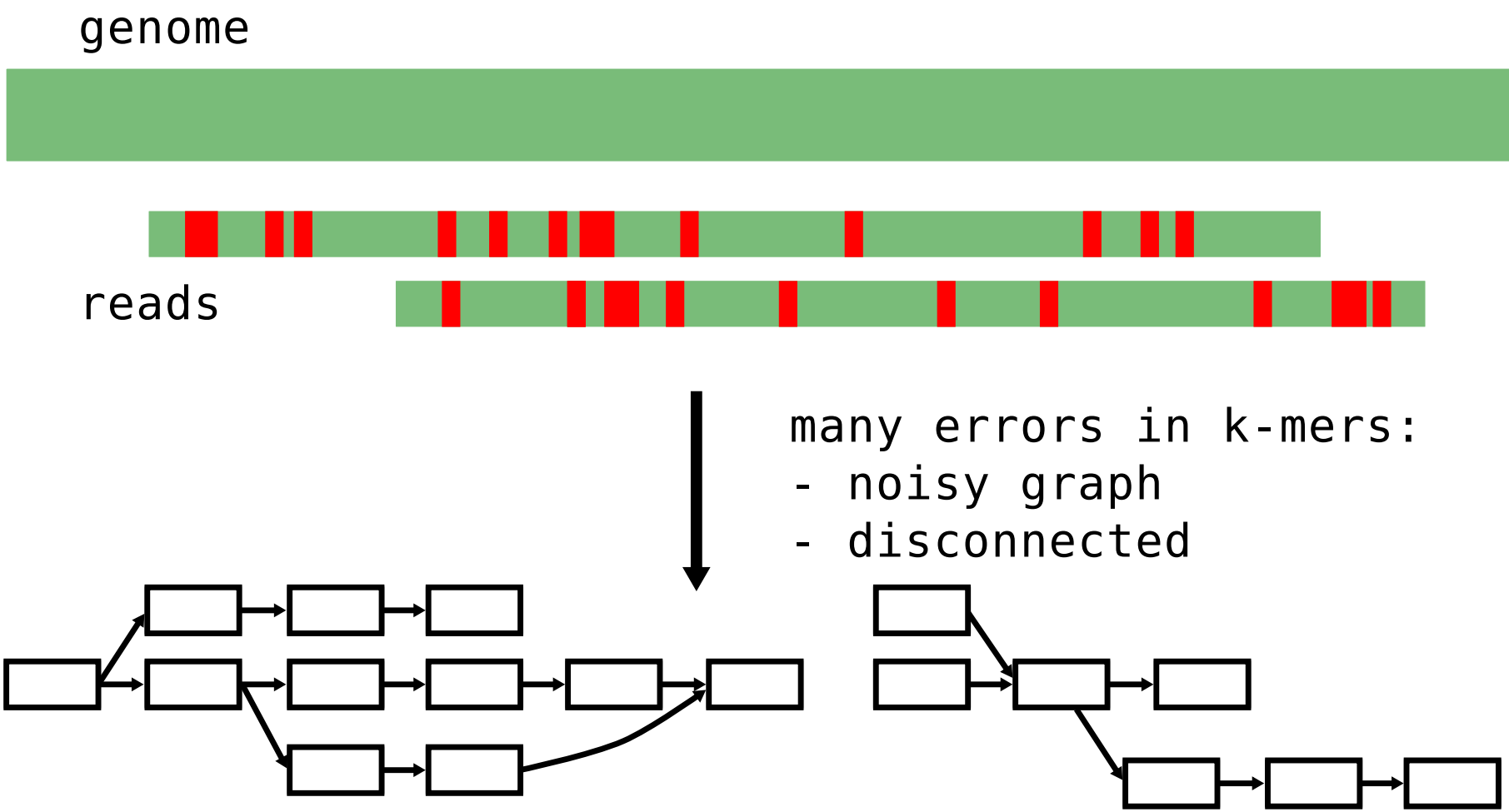
- Long reads $\approx 10kbp$
- High error rate $\approx 10\%$
- High throughput (up to millions of reads per run)

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

Vote!

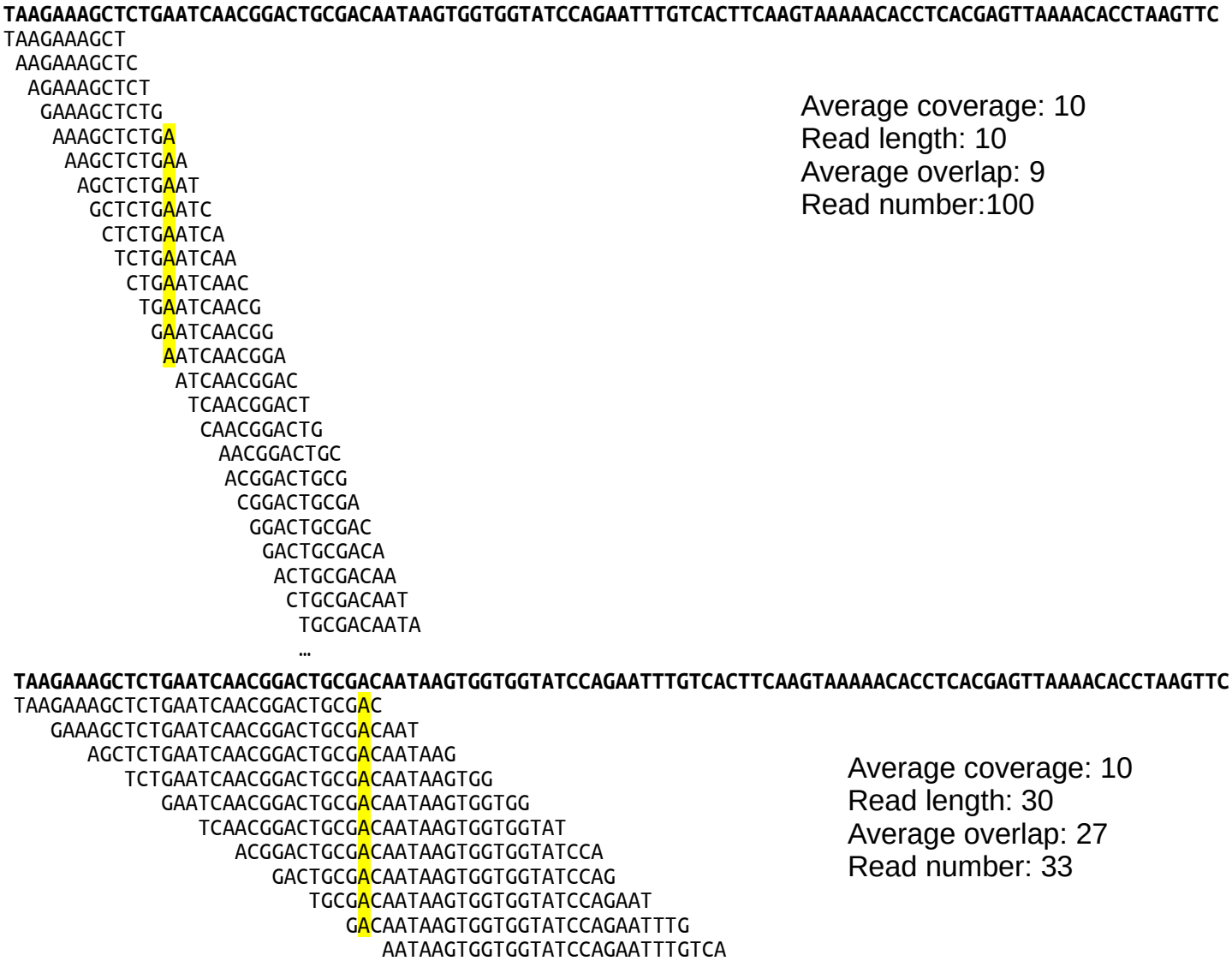
- Greedy
- Overlap graph
- de Bruijn graph

- Long reads for assembly: de Bruijn graph?



• Long reads for assembly: overlap graph?

Supposed to be super expensive!



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

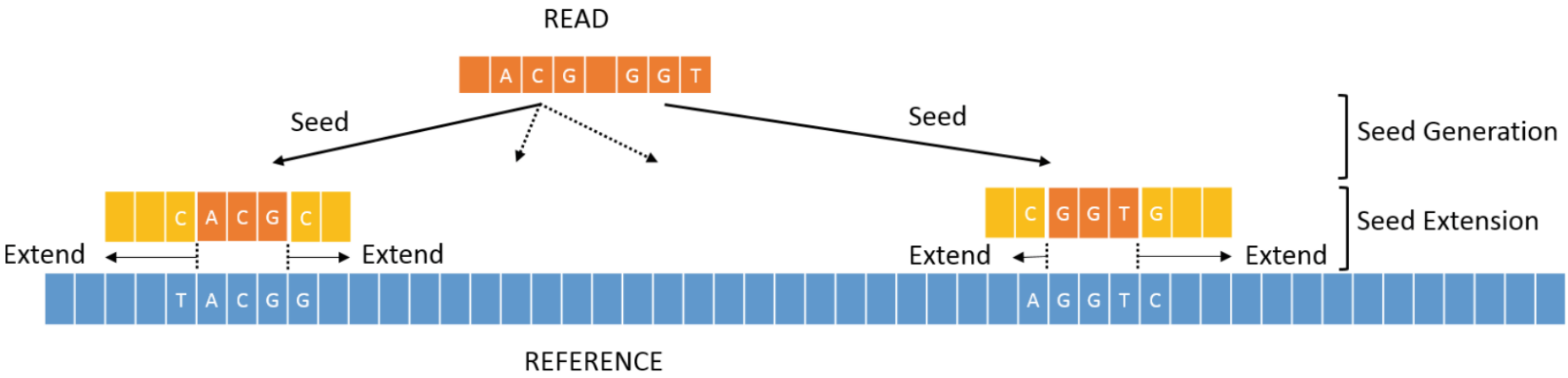
Long reads

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

Very long reads

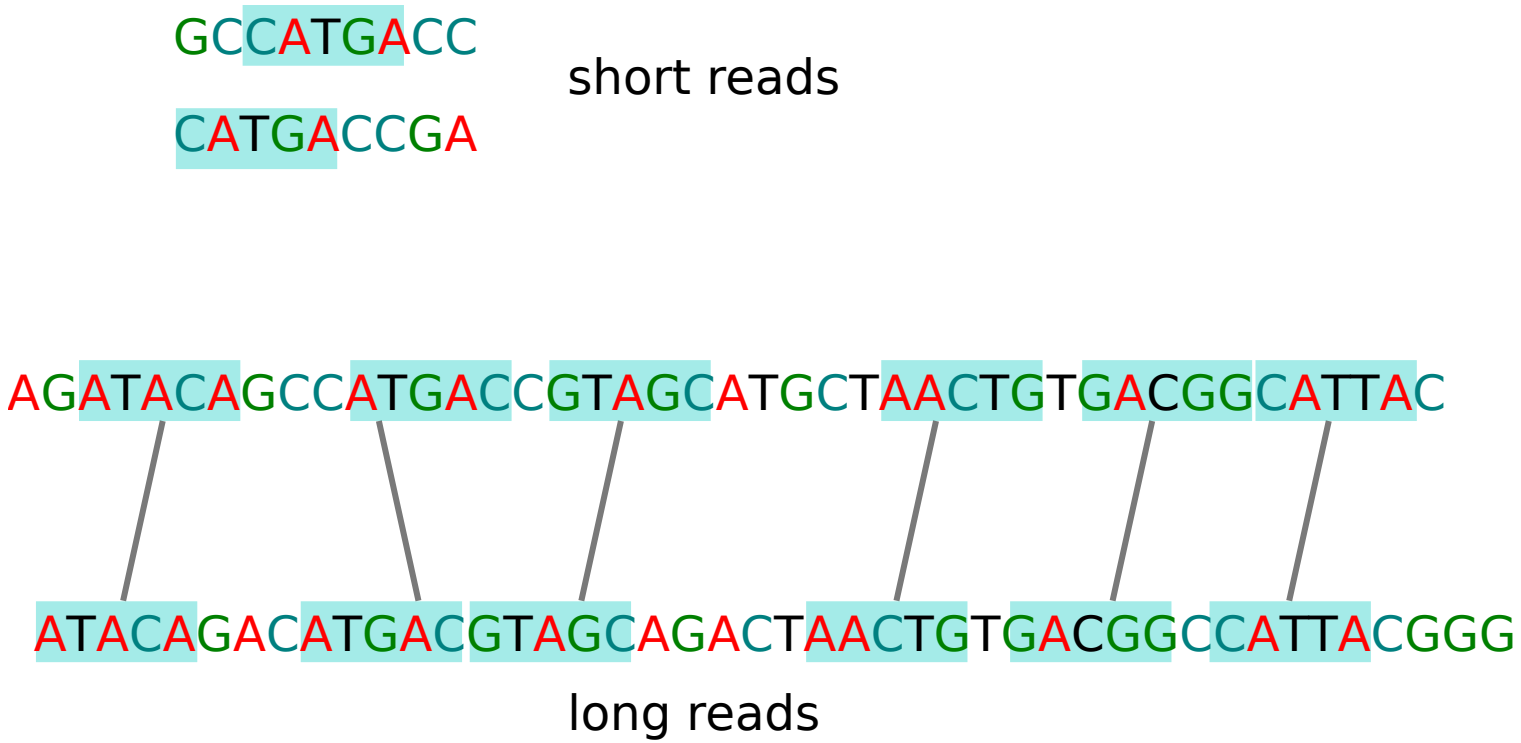
- 5,000 100kbp reads
- 99,000 bp average overlap

- Are large overlaps hard to compute?



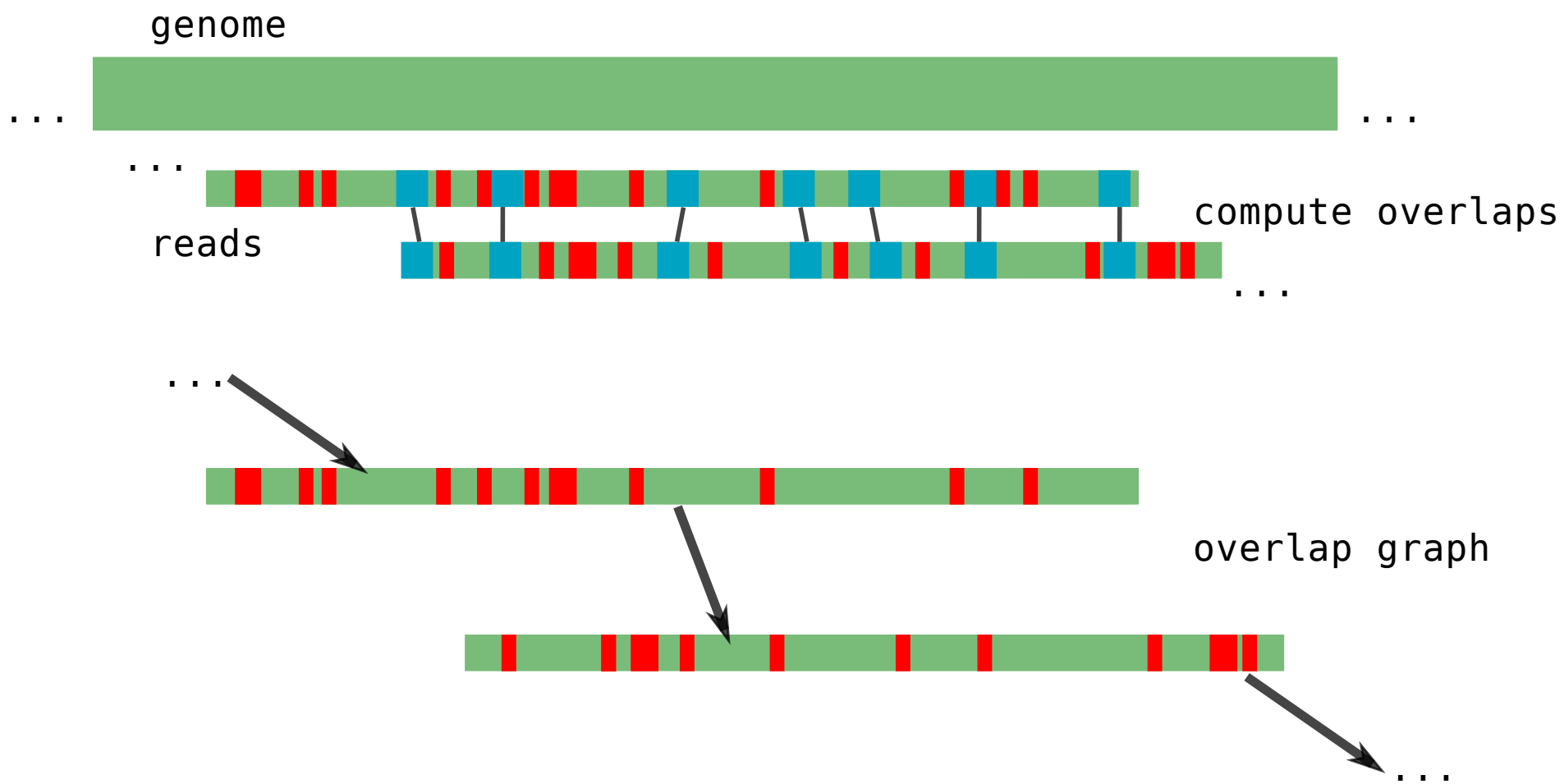
Aligning very long and highly erroneous regions is expected to be expensive, as alignment is quadratic $\approx \mathcal{O}(n^2)$!

"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                    (Substitution)
   GGTATCGTCTA                 (Insertion)
      AT GTTACGG                (Deletion)
```

Insertion and deletion made calling almost impossible

Using coverage to remove noise: Consensus

- Exercise: Perform a consensus

```
Erroneous reads:
TAAGAAAGCTCTGAATCAAACGGTACTGCGA
GAAAGCTTGAATCAAACGGACTGCGACAA
AGCTCTGAATCAACGGACTGCGACAATAA

Contig to polish:
TAAGAAAGCTTGAATCAACGGAATGGCGACAATAA
```

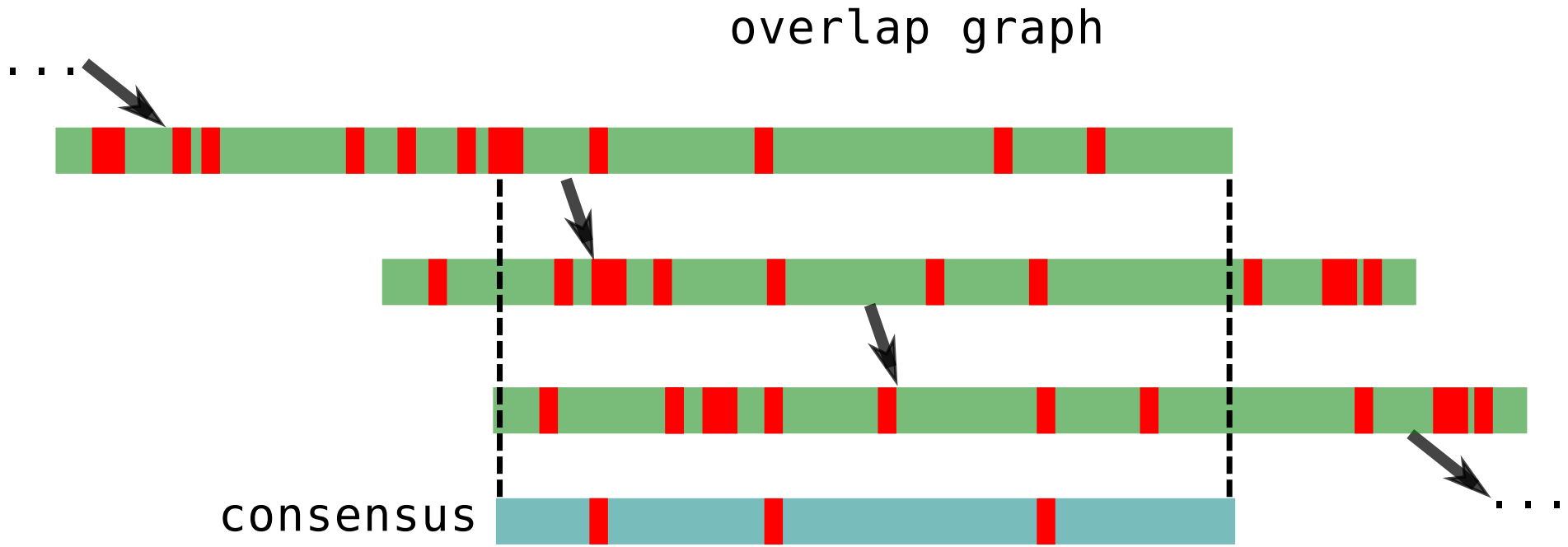
- Exercise: Perform a consensus - solution

Correct contig:
TAAGAAAGCTCTAATCAA-CGGACTG-CGACAATAA

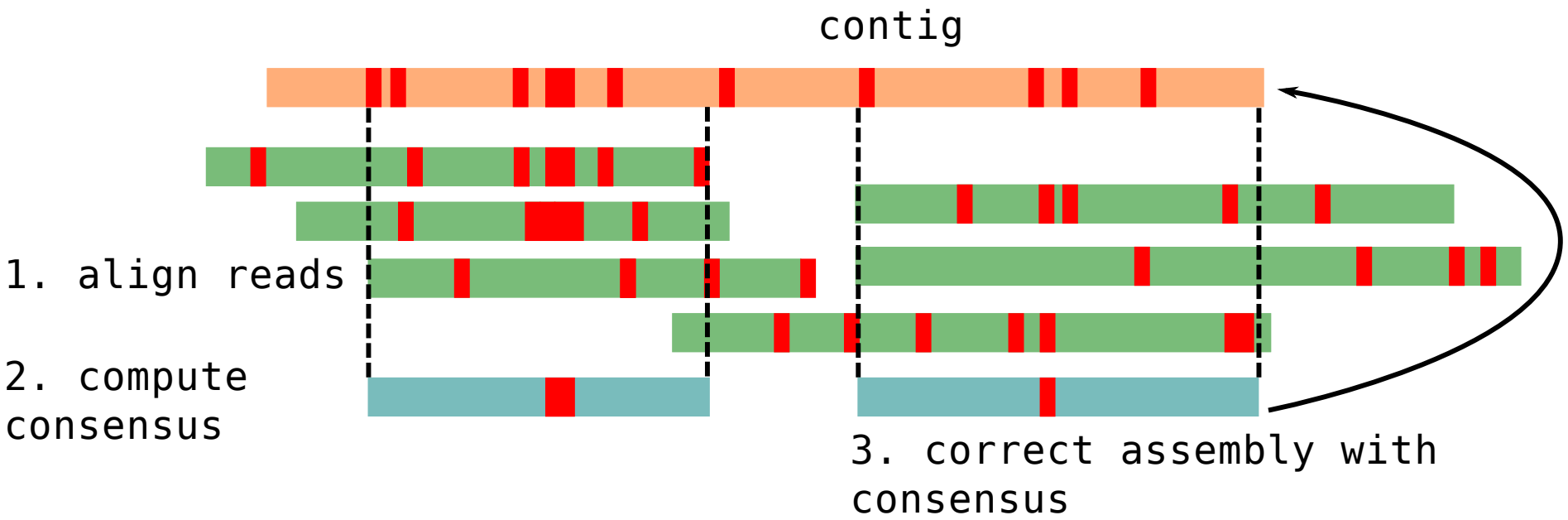
Aligned reads:

```
TAAGAAAGCTCTAATCAAACGGACTGTCGA
    AAAGCT-TAATCAA-CGGACTG-CGACAAT
       GCTCTAATCAA-CGG-CTG-CGACAATAA
```

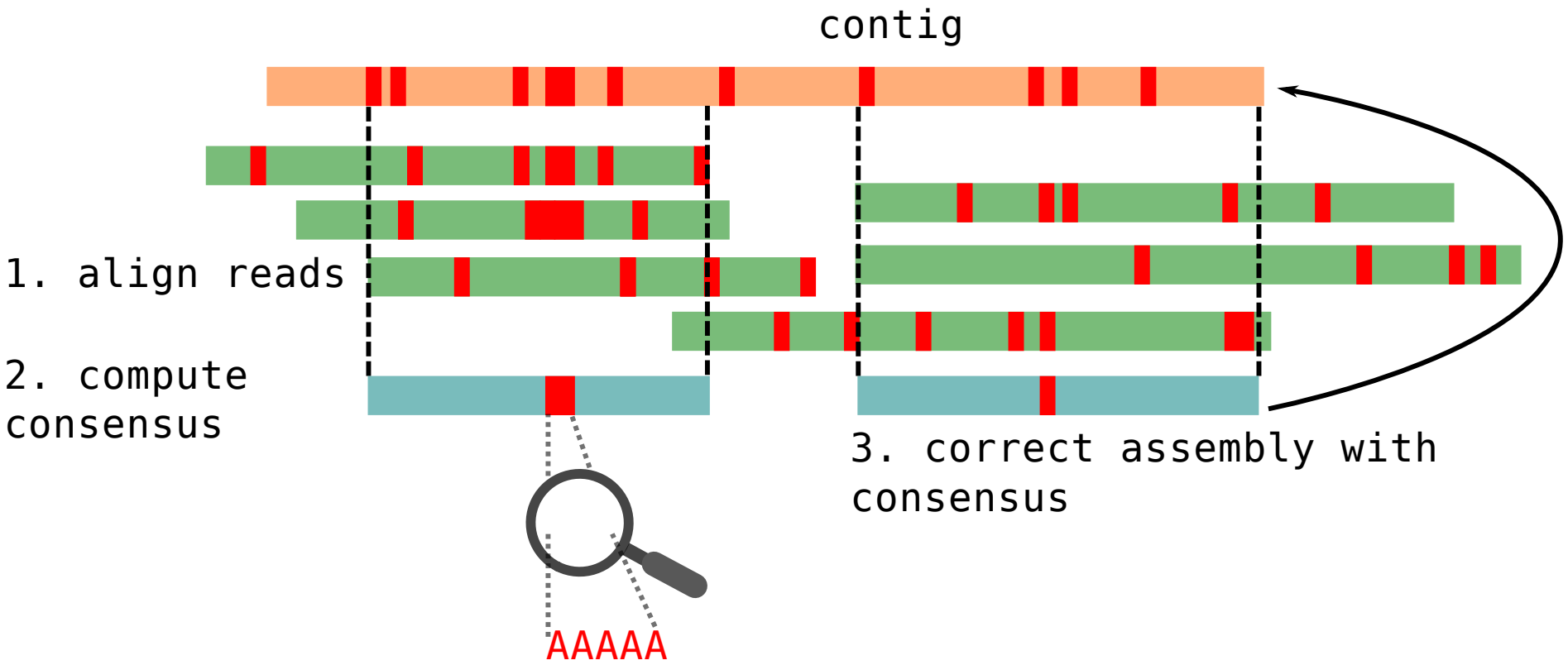
- Consensus during assembly



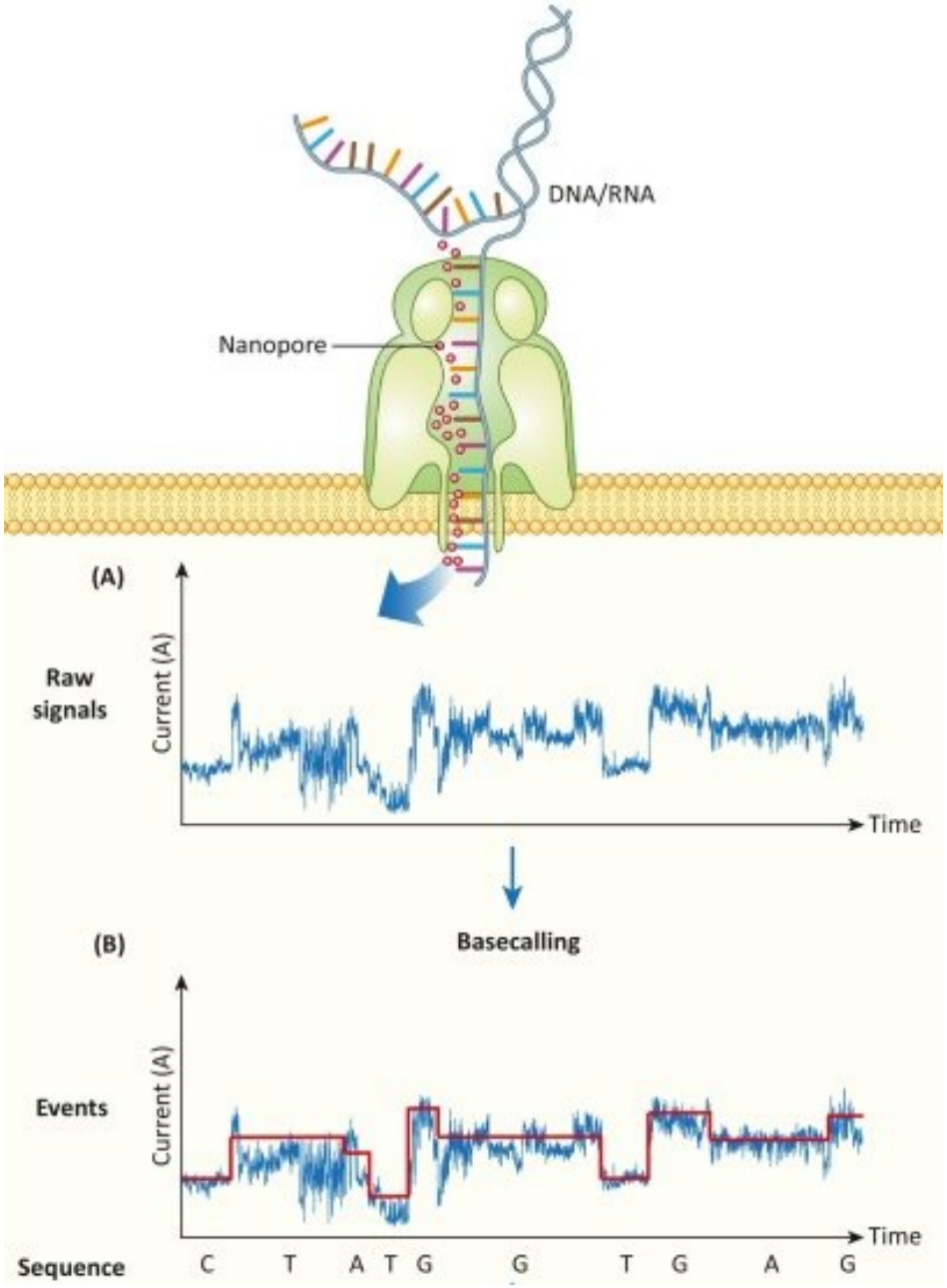
Consensus after assembly: polishing



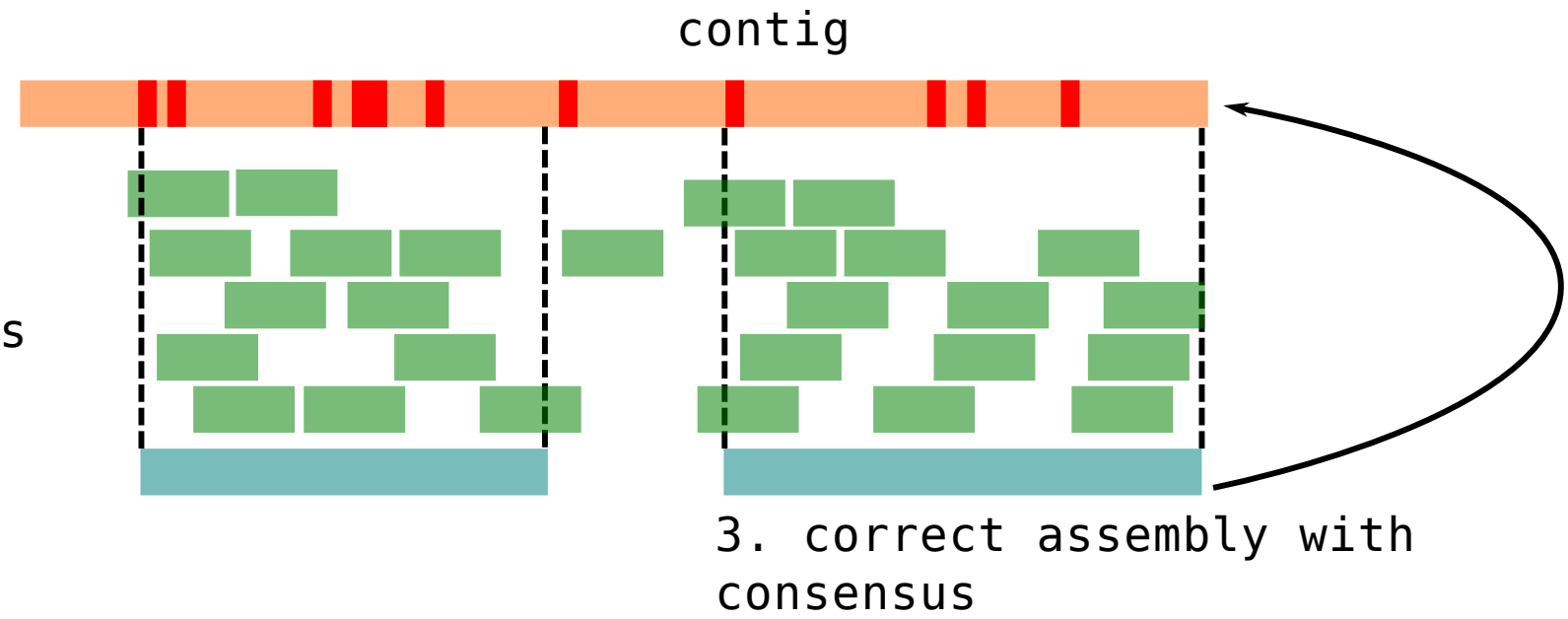
- Consensus after assembly: polishing



- Homopolymers are hard to read

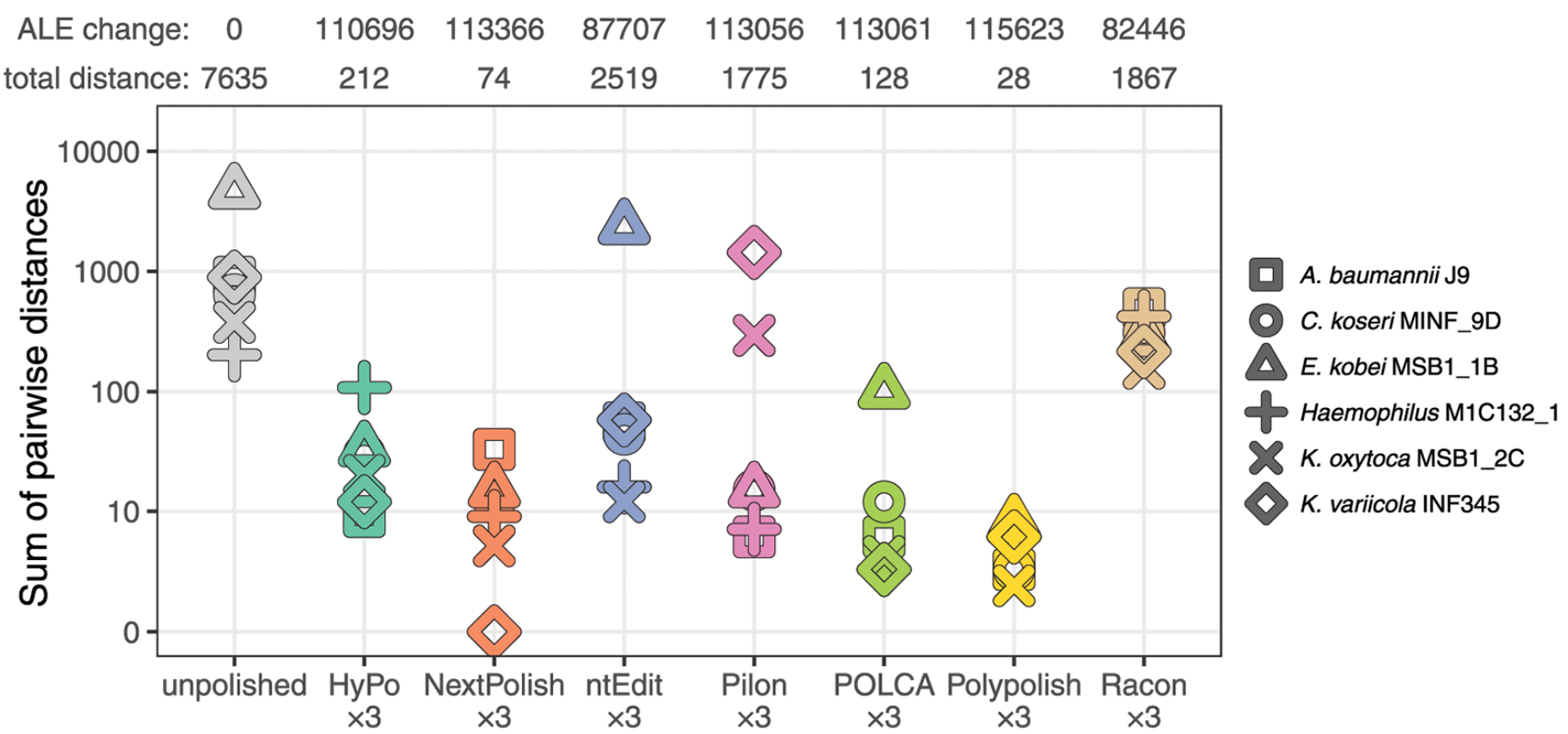


Polishing using accurate reads



Systematics errors

Polishing with Illumina data can improve the final error rate

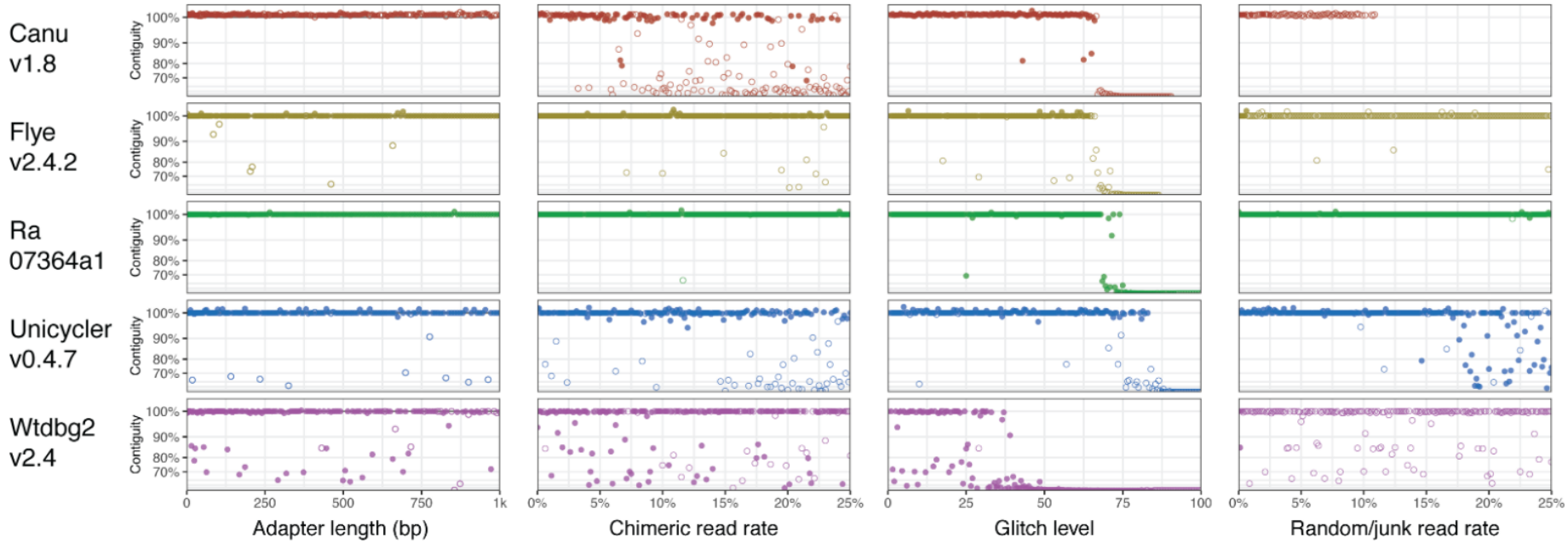


From Polypolish: Short-read polishing of long-read bacterial genome assemblies

- Long reads for assembly: assembly solved?

Assembly is not solved yet

Sometime the software fail



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

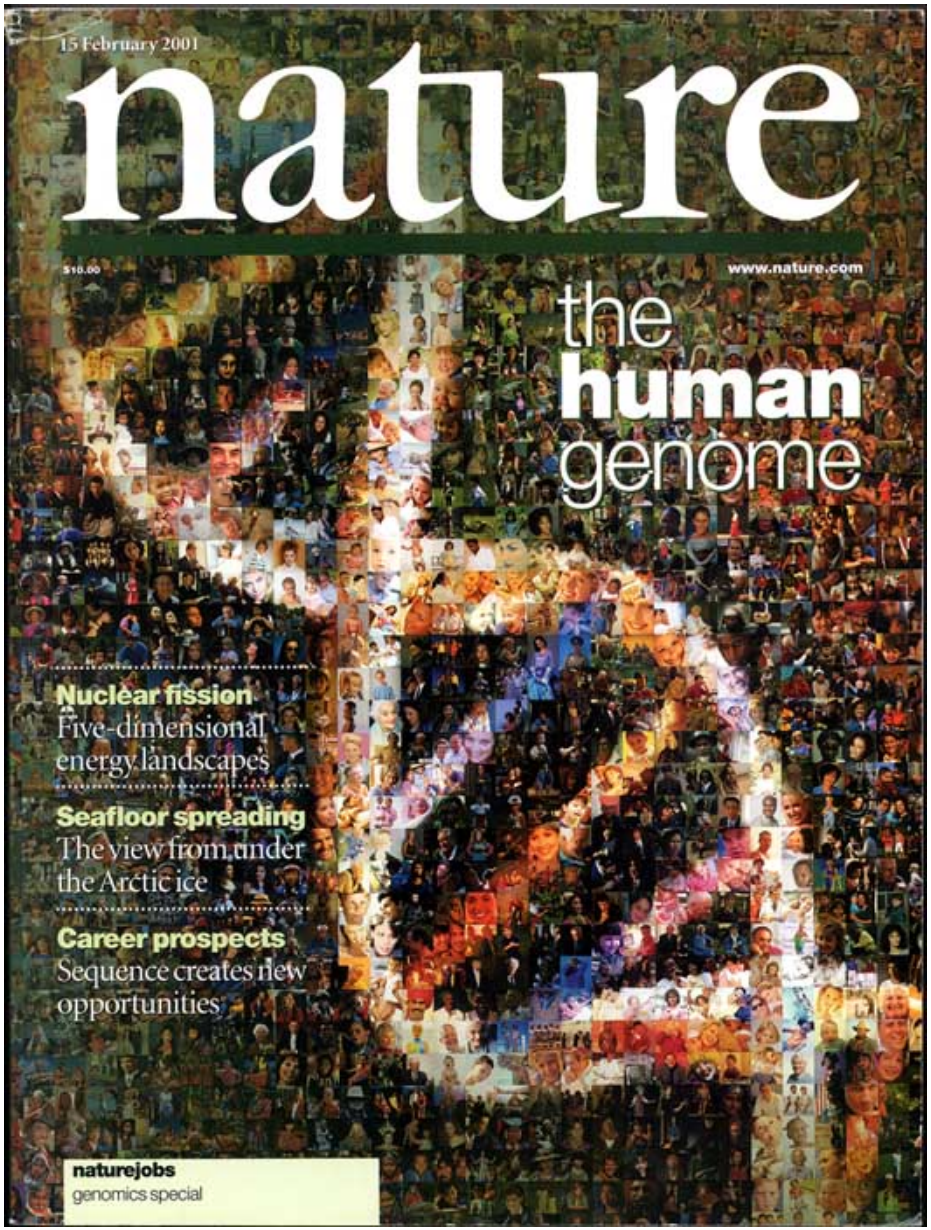
- Long reads for assembly: assembly solved?

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]

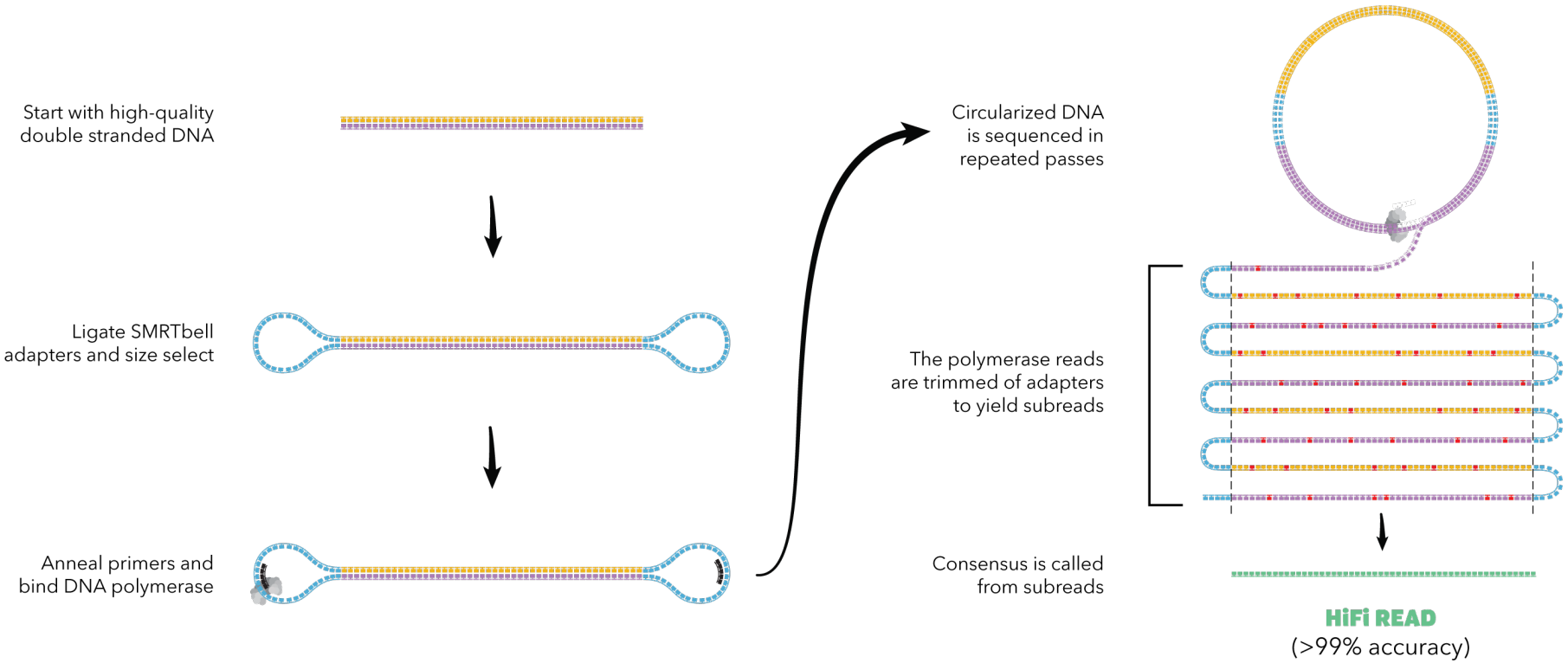
Other notable assemblers

- 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

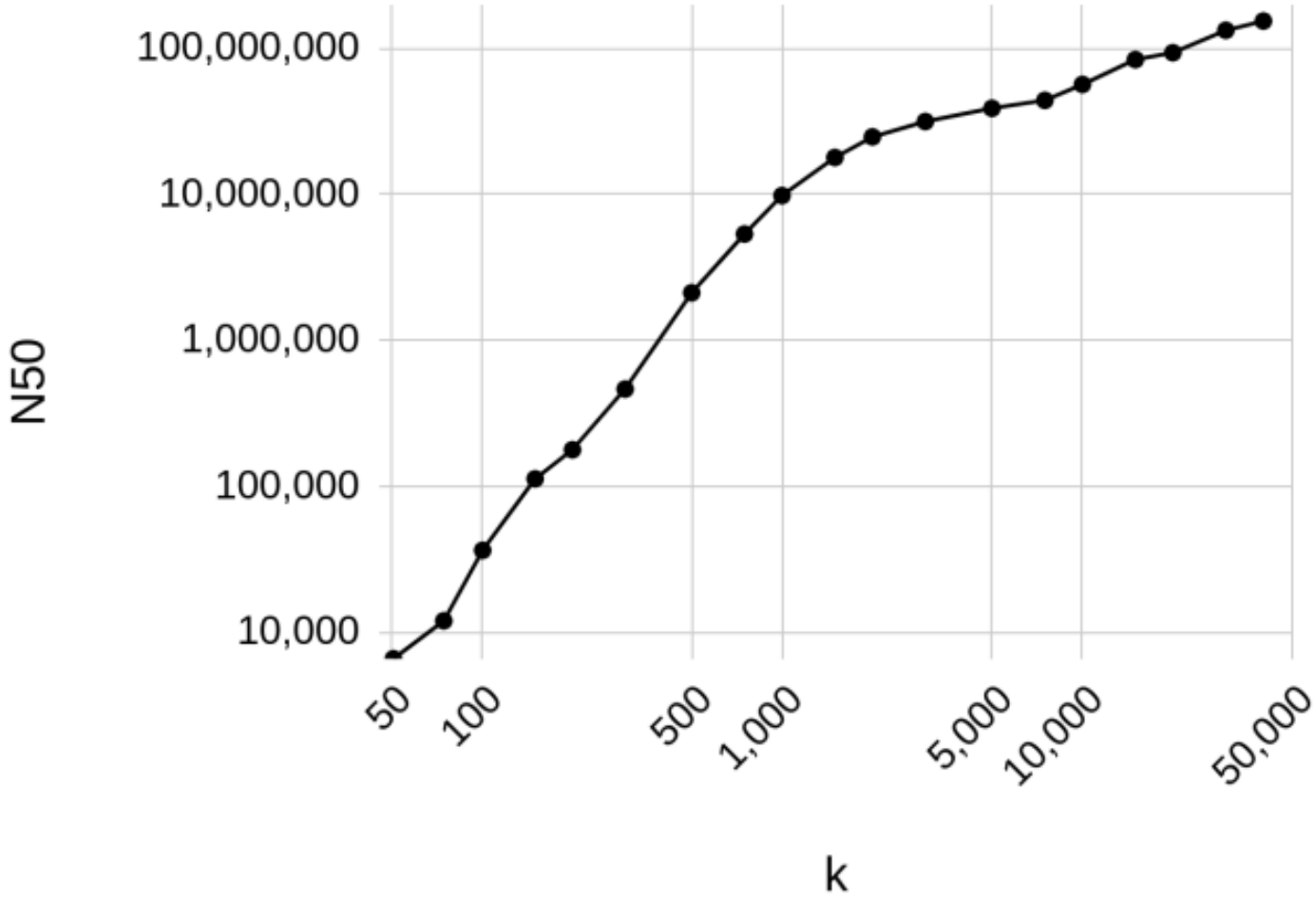
HiFi Assembly

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

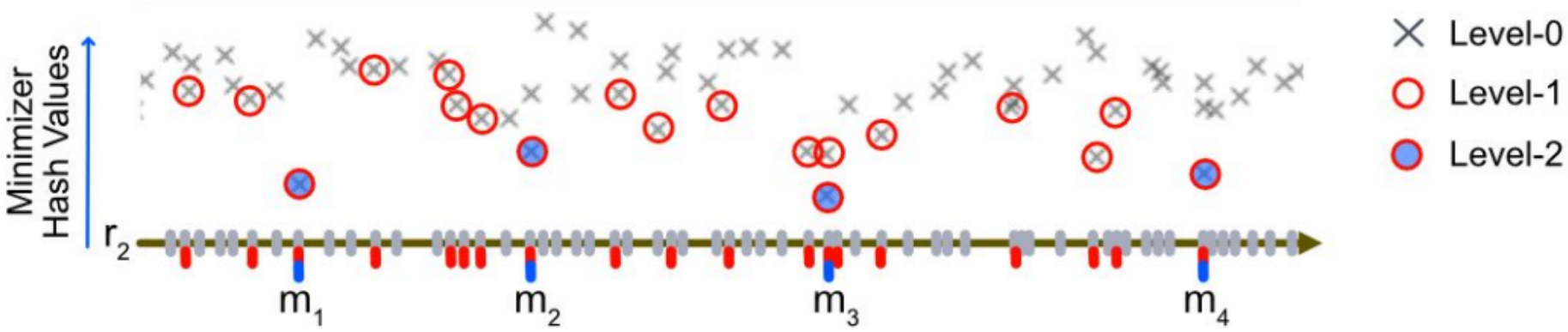
de Bruijn graph Assembly

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

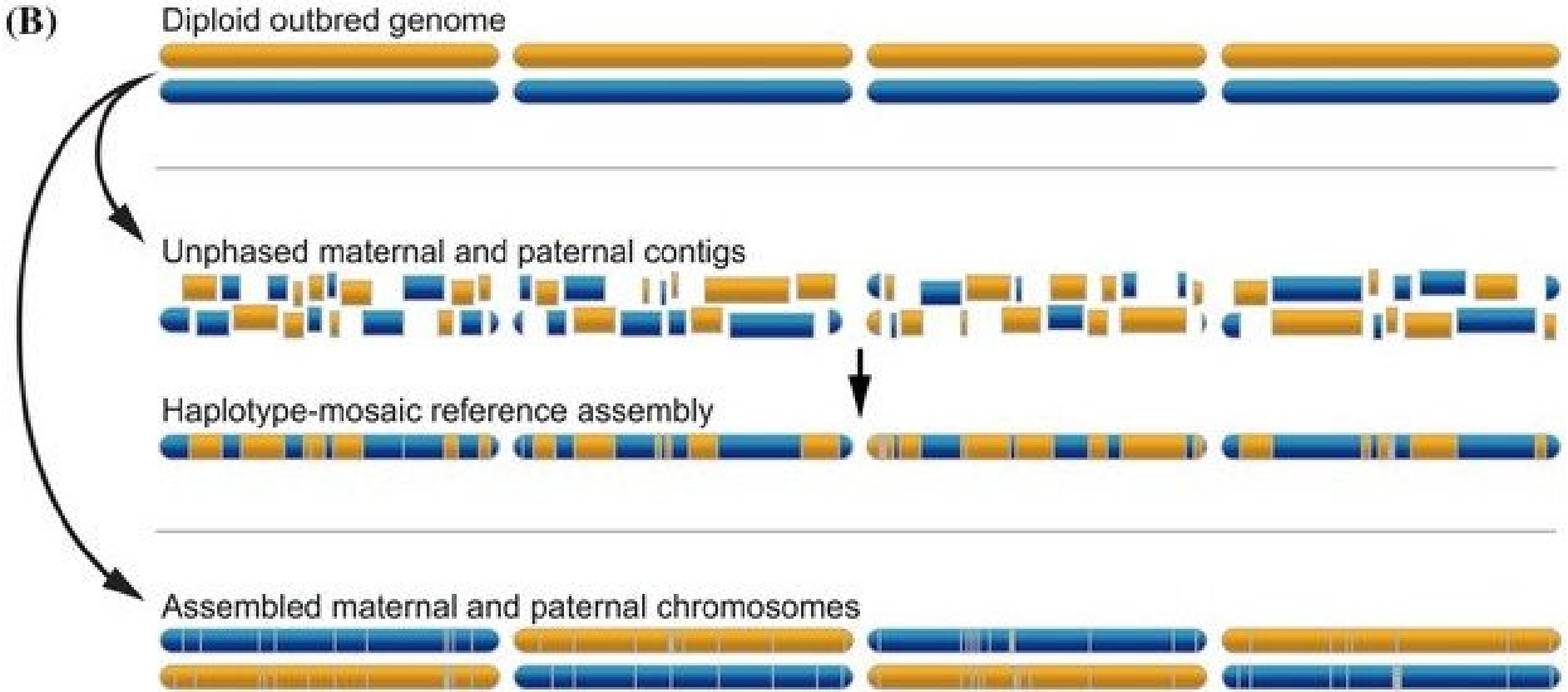


- Very fast genome assembly

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

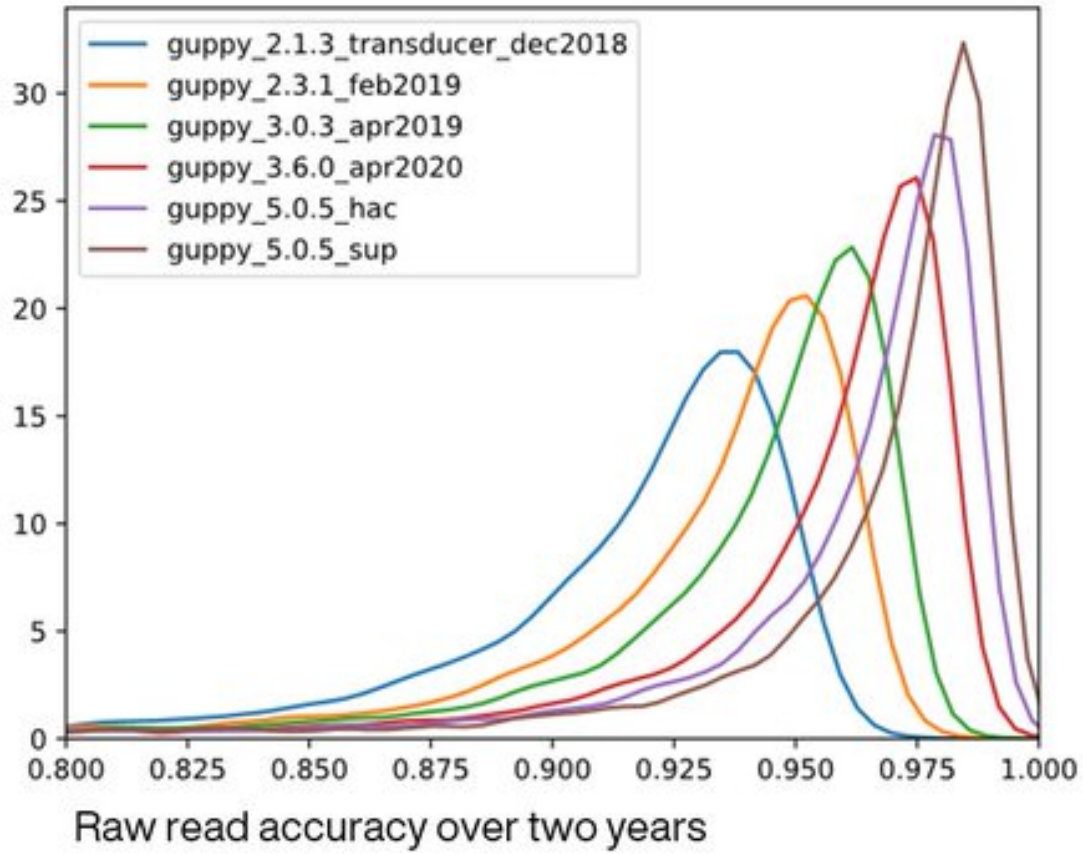


Diploid assembly



Ongoing progress

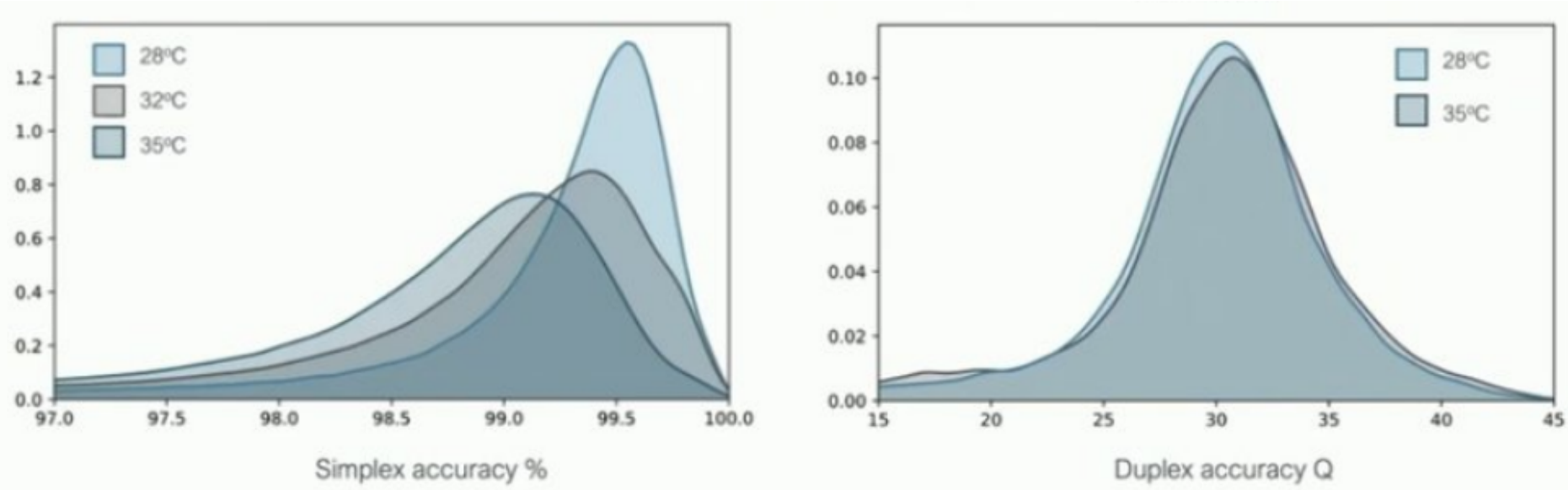
Errors in Nanopore sequencing data are rapidly diminishing



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
@popgengoogling

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