de novo assembly

Rayan Chikhi

CNRS

Workshop on Genomics - Cesky Krumlov
January 2017

Your instructor is..

- Junior CNRS researcher in Lille, France
- Postdoc at Penn State, PhD at ENS Rennes, France
- CompSci background

Research:

- Software and methods for *de novo* assembly:
  - Minia
  - DSK
  - Bcalm
  - KmerGenie
- Collab. on large-genomes assembly projects (giraffe, gorilla Y)

@RayanChikhi on Twitter

`http://rayan.chikhi.name`

Questions to the audience

- Already have data to assemble?
- Plans to sequence *de novo*?
- RNA-Seq?
- Metagenome?
- PacBio/Nanopore reference-free?

Course structure

- Short intro
- Basic definitions
- Fundamentals: why assemblies are as they are
- Metrics: methods for evaluation
- Visualization: see pretty assembly graphs
- RNA-Seq: how Trinity works
- In practice: best practices ; multi-k ; scaffolding ; various

genome
not known

r e a d s
overlapping substrings that cover the genome redundantly

assembly
what we think the genome is

Definition of an **assembly** (a trickier question than it seems)

Set of sequences which best approximate the original sequenced material.

Why assemble?

- **Create/update** reference genome / transcriptome
- **Gene** contents
- Novel **insertions**
- understand **un-mapped** reads
- **SNPs** in non-model organisms
- Find **SV**'s (Evan's talk)
- Specific **regions** of interest
- Metagenomics
- ..

Plan

- What is a de novo assembly
  - Basics
  - Short Exercise
- Some useful assembly theory
  - Graphs
  - Contigs construction
  - Exercise
- Visualizing and evaluating assemblies
  - Bandage
  - Reference-free metrics
  - Exercise
- Assembly software
  - DNA-seq assembly
  - RNA-seq assembly
  - Multi-k and scaffolding
  - Other data types
  - Exercise

Basic expectations

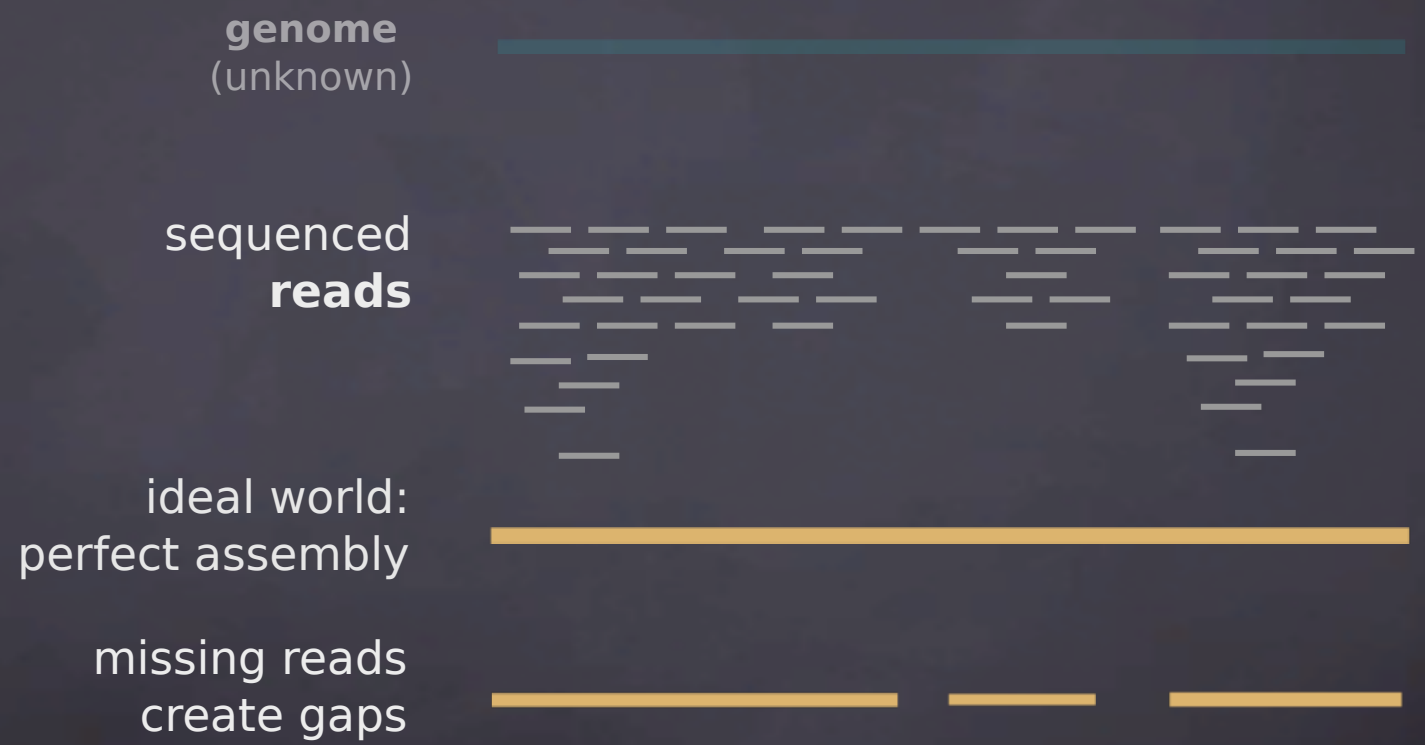
An assembly generally is:

- smaller than the reference,
- fragmented

Basic expectations

An assembly generally is:

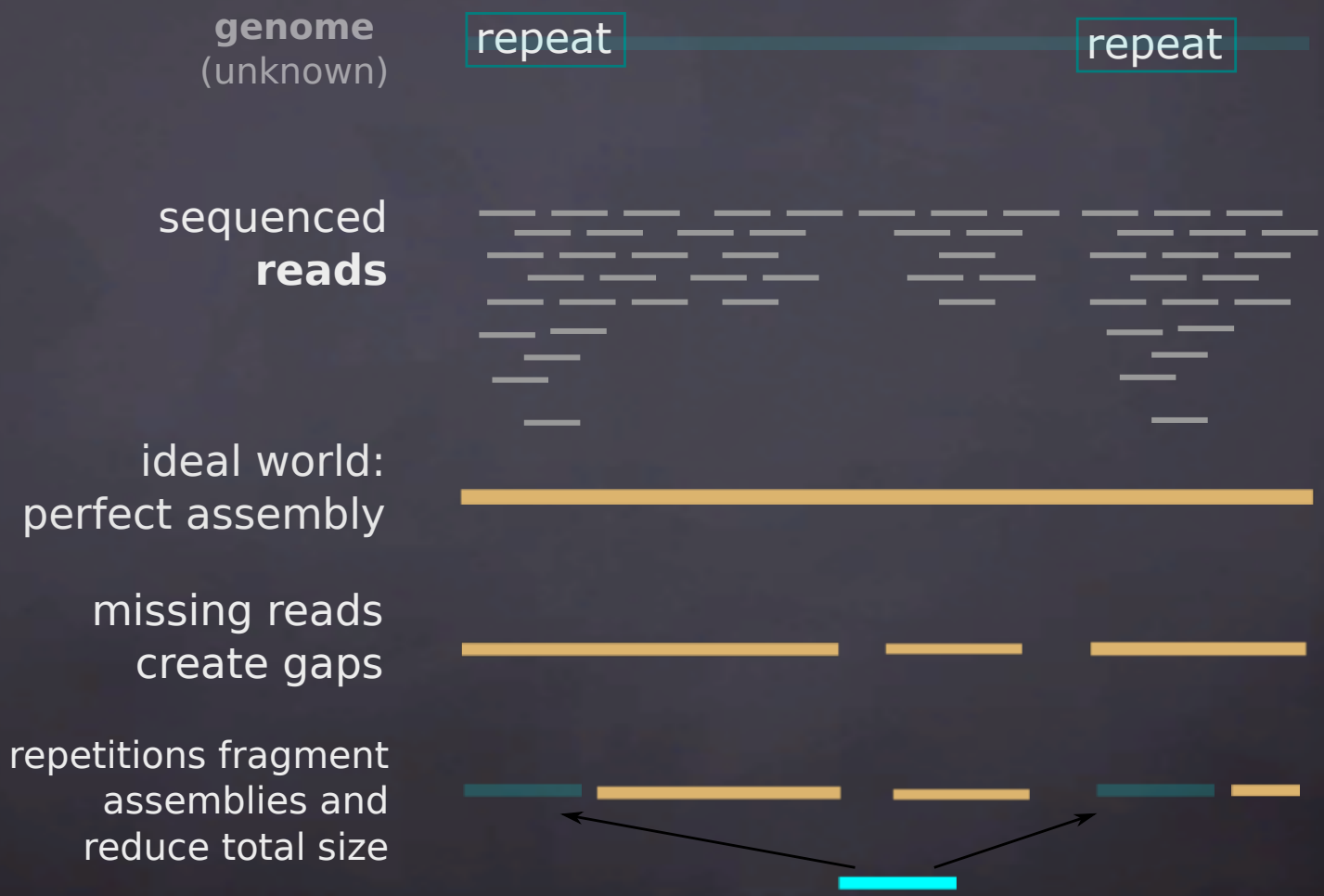
- smaller than the reference,
- fragmented



Basic expectations

An assembly generally is:

- smaller than the reference,
- fragmented



Some vocabulary:

Read Any sequence that comes out of the sequencer

Paired read forward $read_1$, gap $\leq$ 500 bp, reverse $read_2$

Mate-pair reverse $read_1$, gap $\geq$ 1 kbp, forward $read_2$

Single read Unpaired read

***k*-mer** Any sequence of length k

Contig gap-less assembled sequence

Scaffold sequence which may contain gaps (N)

Overlaps

What does it mean for two strings to overlap?
$\rightarrow$ a suffix of the first string equals (or is close to) a prefix of the other string.

Exact overlaps:

```
1: ACTGCT
2:  CTGCT
3:    GCTAA
```

read 1 overlaps with read 2 and also with read 3.

read 2 overlaps with read 3.

Inexact overlaps (here, allowing for ≤ 1 mismatch):

```
1: ACTGCT
2:  CTACT
3:    ACGAA
```

read 1 overlaps with read 2 with 1 mismatch.
read 1 would overlap with read 3 but with 2 mismatches.

read 2 overlaps with read 3 with 1 mismatch.

Graphs

A graph is a set of nodes and a set of edges (directed or not).

Exercise

Here is a set of reads:

```
TACAGT
  CAGTC
   AGTCA
      CAGA
```

1. How many k-mers are in these reads (including duplicates), for $k = 3$?
2. How many *distinct* k-mers are in these reads?
   - (i) for $k = 2$
   - (ii) for $k = 3$
   - (iii) for $k = 5$
3. How many distinct pair-wise overlaps of length ≥ 3 are there between the reads?
4. Pretend these reads come from the genome TACAGTCAGA. What is the largest k such that the set of distinct k-mers in the genome is exactly the set of distinct k-mers in the reads above?

Exercise (solution)

Here is a set of reads:

```
TACAGT
  CAGTC
   AGTCA
      CAGA
```

1. How many k-mers are in these reads (including duplicates), for $k = 3$? 12
2. How many *distinct* k-mers are in these reads?
   - (i) for $k = 2$: 7
   - (ii) for $k = 3$: 7
   - (iii) for $k = 5$: 4
3. How many distinct pair-wise overlaps of length ≥ 3 are there between the reads? : 3
4. Pretend these reads come from the genome TACAGTCAGA. What is the largest k such that the set of distinct k-mers in the genome is exactly the set of distinct k-mers in the reads above? 3; for k=4, TCAG does not appear in the reads

(Exercices 1,2,4 count as "kmer tasks")

Plan

- What is a de novo assembly
  - Basics
  - Short Exercise
- Some useful assembly theory
  - Graphs
  - Contigs construction
  - Exercise
- Visualizing and evaluating assemblies
  - Bandage
  - Reference-free metrics
  - Exercise
- Assembly software
  - DNA-seq assembly
  - RNA-seq assembly
  - Multi-k and scaffolding
  - Other data types
  - Exercise

Graphs for sequencing data

Overlaps between reads is the fundamental information used to assemble.
Graphs represent these overlaps.

Two different types of graphs for sequencing data:

- de Bruijn graphs — for Illumina data
- string graphs — for PacBio/Nanopore data

A bioinformatician familiar with these graphs will know:

- how to set the **parameters** of an assembler
- the type of **errors** that assemblers make
- why **variants** are hidden
- why some repetitions are **over-collapsed**
- why **heterozygous** regions may appear **twice**

Overlap graphs

*This is going to be fundamental for **PacBio/Nanopore** data.*

1. **Nodes** = reads
2. **Edges** = overlaps between two reads

In this example, let's say that an overlap needs to be:

- exact
- over at least 3 characters,

Reads:

```
ACTGCT
 CTGCT (overlap of length 5)
  GCTAA (overlap of length 3)
```

Graph:

String graphs

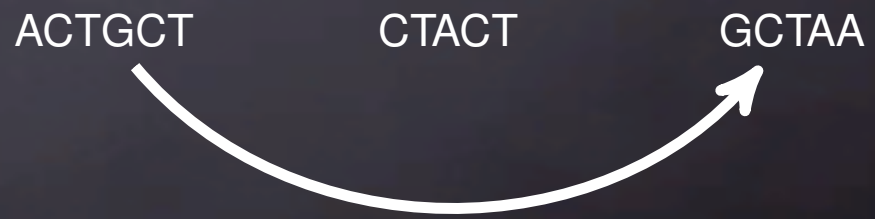
A **string graph** is obtained from an overlap graph by removing redundancy:

- redundant reads (those fully contained in another read)
- transitively redundant edges (if $a \to c$ and $a \to b \to c$, then remove $a \to c$)

Two examples:

```
ACTGCT
 CTGCT (overlap length 5)
   GCTAA (overlap length 3)
```


```
ACTGCT
 CTACT
   GCTAA
```



String graphs

A **string graph** is obtained from an overlap graph by removing redundancy:

- redundant reads (those fully contained in another read)
- transitively redundant edges (if $a \to c$ and $a \to b \to c$, then remove $a \to c$)

Two examples:

```
ACTGCT
 CTGCT (overlap length 5)
   GCTAA (overlap length 3)
```

Let's have inexact overlaps now

```
ACTGCT
 CTACT
   GCTAA
```

From Overlap graphs to String graphs

Overlap graph with exact overlaps ≥ 3,

String graph with exact overlaps ≥ 3,

The read CTGCT is contained in ACTGCT, so it is redundant

de Bruijn graphs

*This is going to be fundamental for **Illumina** data.*

A **de Bruijn** graph for a fixed integer k:

1. **Nodes** = all *k-mers* (substrings of length k) in the reads.
2. **Edges** = all exact overlaps of length exactly $(k-1)$ between k-mers

Example for $k = 3$ and a single read:

`ACTG`

ACT → CTG

de Bruijn graphs

Example for many reads and still $k = 3$.

```
ACTG
 CTGC
  TGCC
```

de Bruijn graphs: redundancy

What happens if we add redundancy?

```
ACTG
ACTG
 CTGC
 CTGC
 CTGC
  TGCC
  TGCC
```

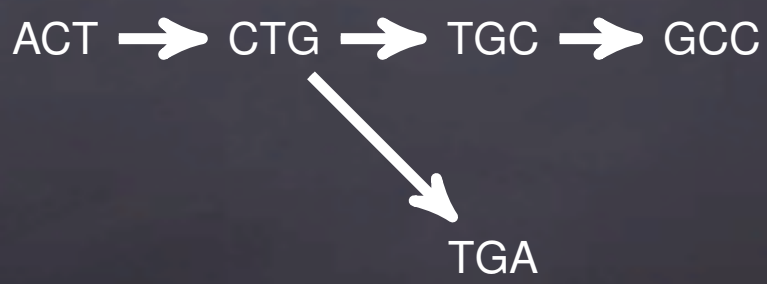
dBG, $k = 3$:

de Bruijn graphs: errors

How is a sequencing error (at the end of a read) impacting the de Bruijn graph?

```
ACTG
 CTGC
 CTGA
  TGCC
```

dBG, $k = 3$:



de Bruijn graphs: SNPs

What is the effect of a SNP (or a sequencing error inside a read) on the graph?

AGCCTGA
AGCATGA

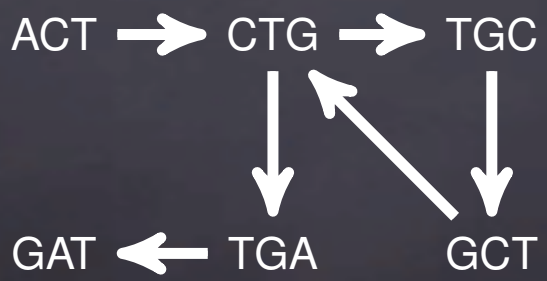
dBG, $k = 3$:

de Bruijn graphs: repeats

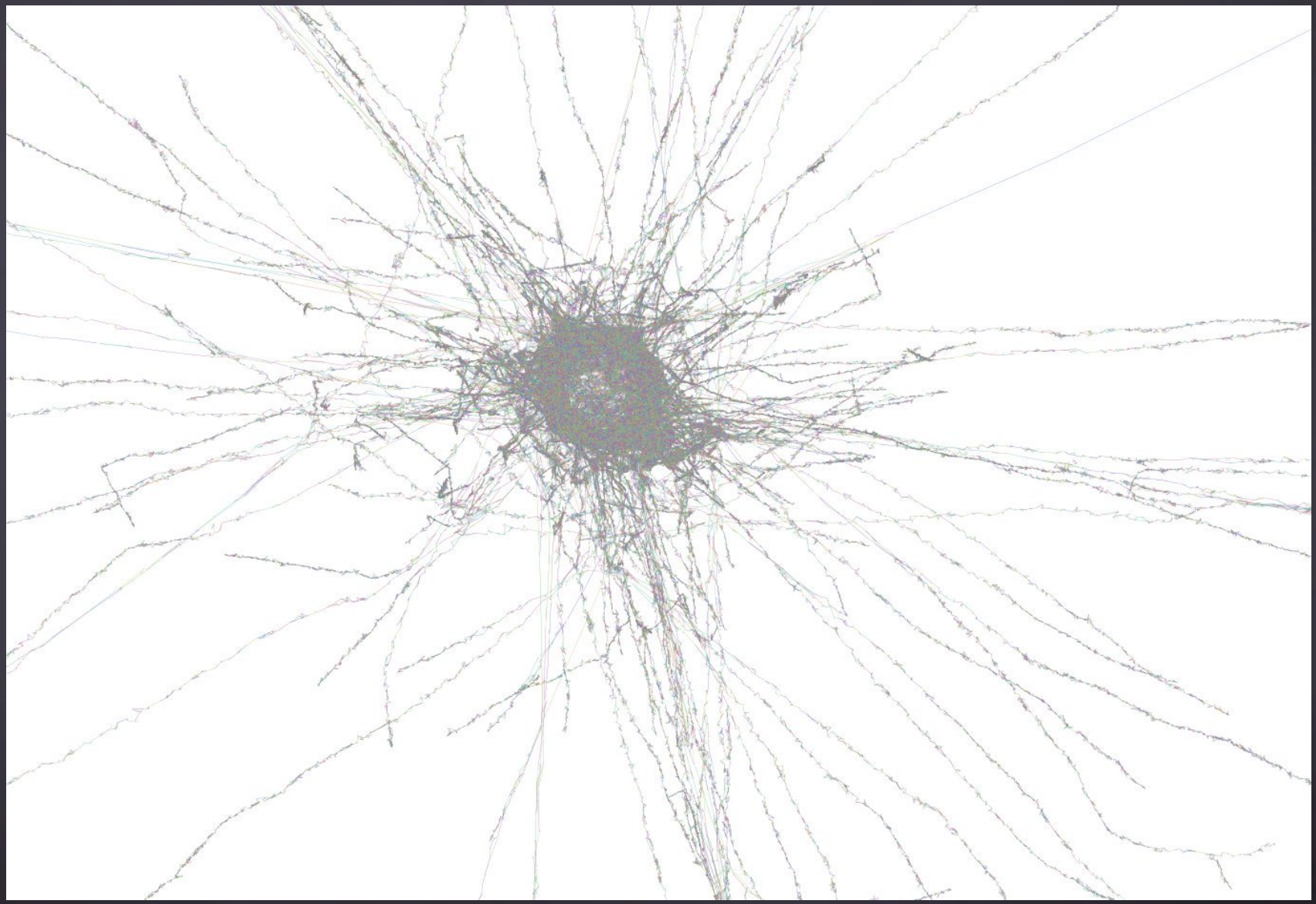
What is the effect of a small repeat on the graph?

```
ACTG
 CTGC
  TGCT
   GCTG
    CTGA
     TGAT
```

dBG, $k = 3$:



REAL DE BRUIJN GRAPHS



S. aureus reads dBG, uncleaned (SRR022865)

Comparison string graph / de Bruijn graph

On the same example, compare the de Bruijn graph with the string graph:

```
AGTGCT
 GTGCTA
   GCTAA
```

String graph with exact overlaps ≥ 3:

de Bruijn graph, $k = 3$:

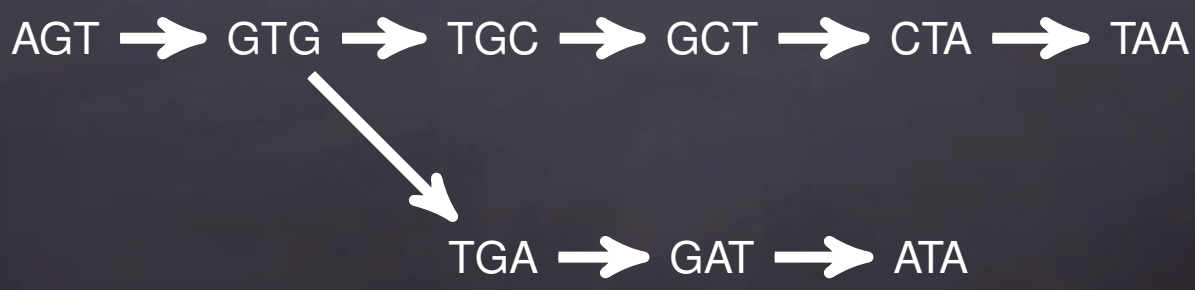
String graph / de Bruijn graph (2)

Let's add an error:

```
AGTGCT
 GTGATA
   GCTAA
```

String graph where overlaps ≥ 3 may ignore up to 1 error:

de Bruijn graph, $k = 3$:



String graph / de Bruijn graph (4)

So, which is better?

- String graphs capture whole read information
- de Bruijn graphs are conceptually simpler:
  - single node length
  - single overlap definition

Historically, string graphs were used for long reads and de Bruijn graphs for short reads.

String graphs are also known as the **Overlap Layout Consensus** (OLC) method.

How does one assemble using a graph?

Assembly in theory

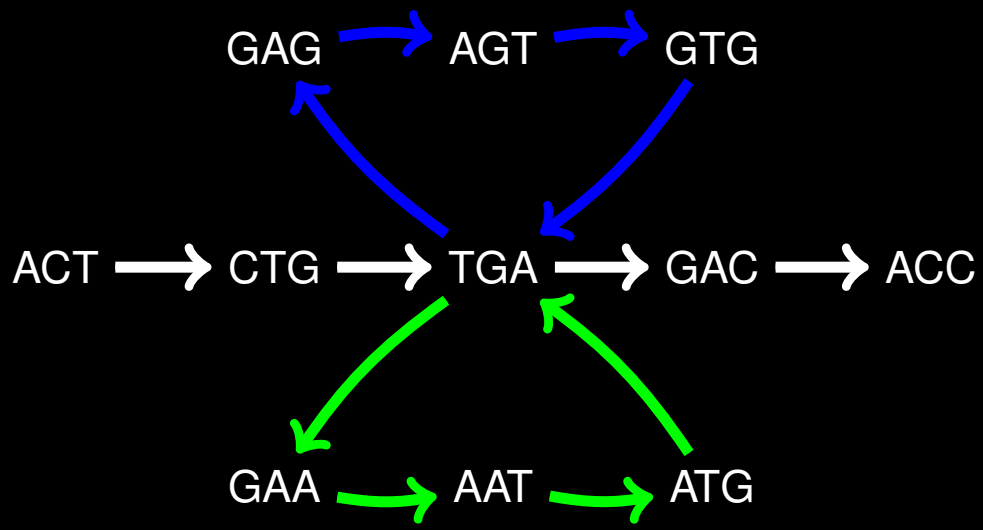
[Nagarajan 09]

Return a path of *minimal length* that traverses **each node at least once**.

Illustration

The only solution is `GATTACATTACAA`.

An ambiguous assembly graph



Because of ambiguities and low-coverage regions, a single path is almost never found in theory, and is really never found in practice.

Assembly in practice

Return a **set of paths** covering the graph, such that *all possible assemblies* contain these paths.

Assembly of the above graph

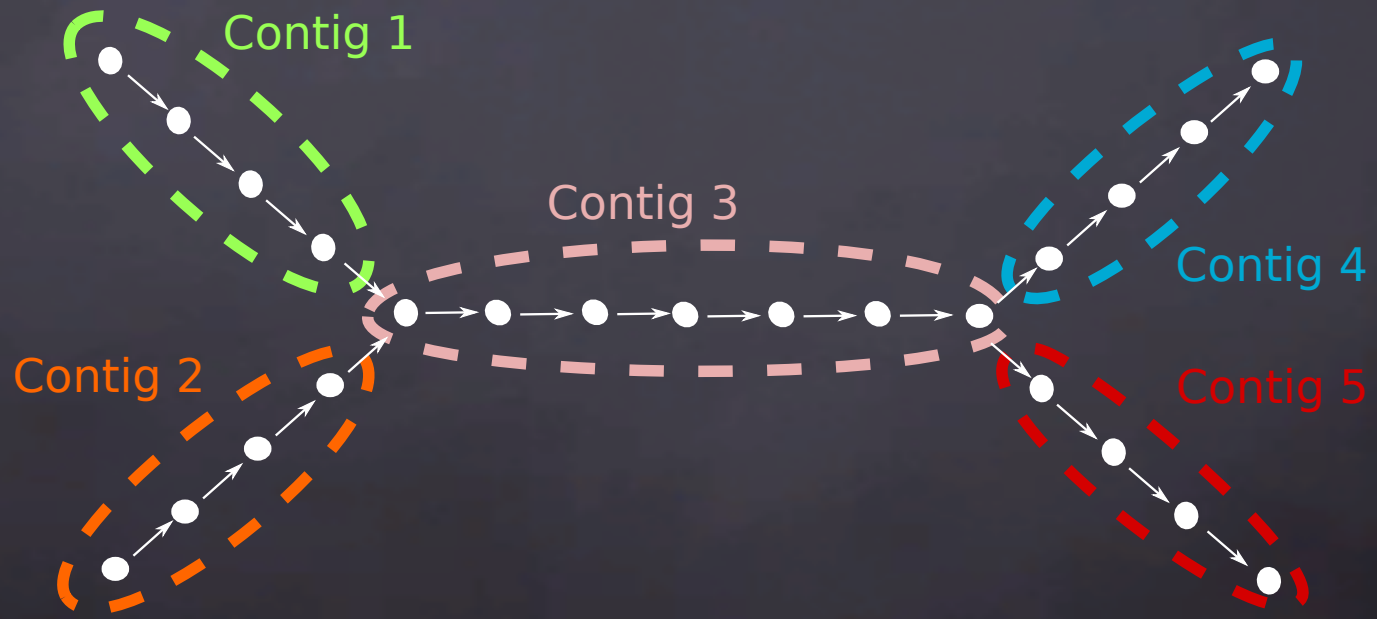
An assembly is the following set of paths:

$$\{\texttt{ACTGA}, \texttt{GACC}, \texttt{GAGTG}, \texttt{GAATG}\}$$

CONTIGS CONSTRUCTION

Contigs are *node-disjoint* **simple paths**.

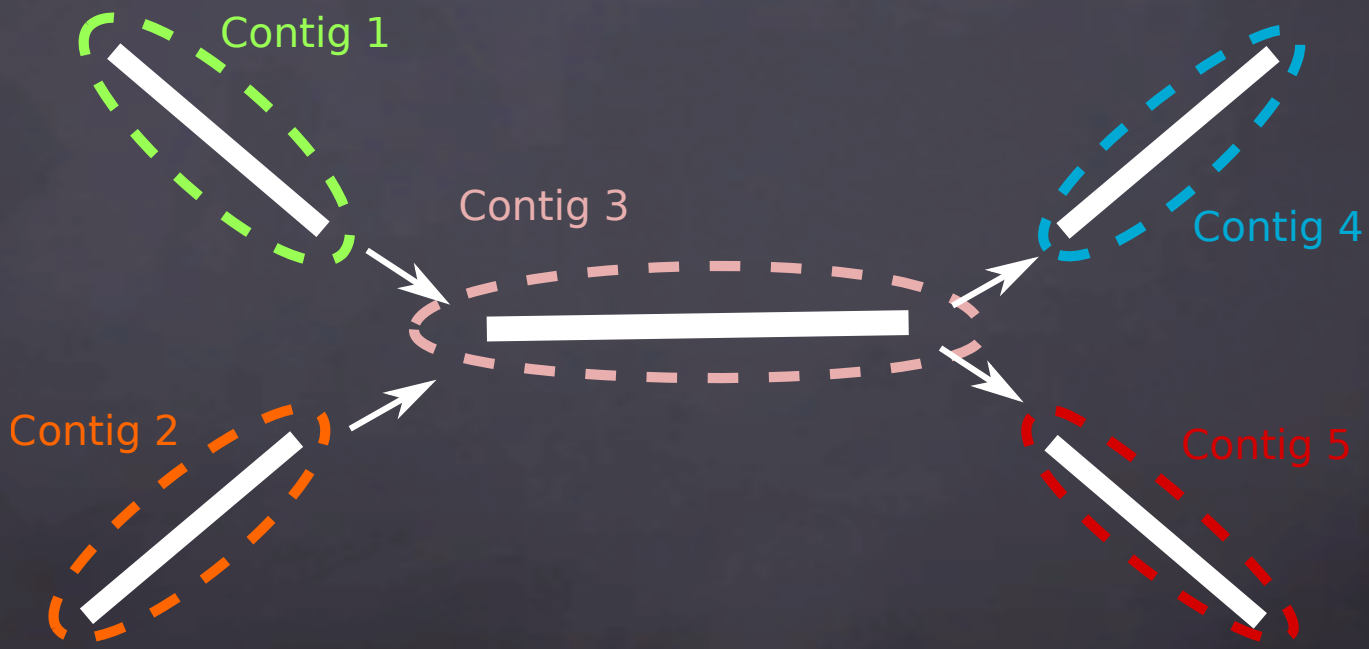
simple path: a path that does not branch.
node-disjoint: two different paths cannot share a node.



CONTIGS GRAPH

Contigs graph:

- nodes = contigs
- edges = overlaps between contigs

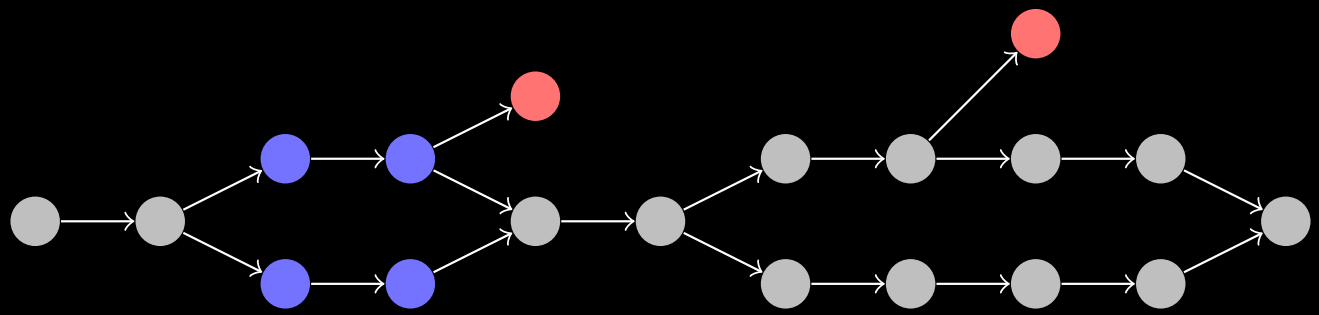


How an assembler works

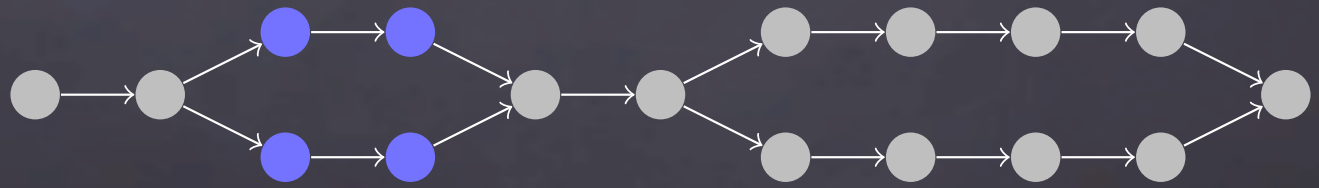
[SPAdes, Velvet, ABySS, SOAPdenovo, SGA, Megahit, Minia, FALCON, Canu, ..]

1) Maybe correct the reads. (SPAdes, HGAP, SGA, FALCON, Canu)
2) Construct a graph from the reads.

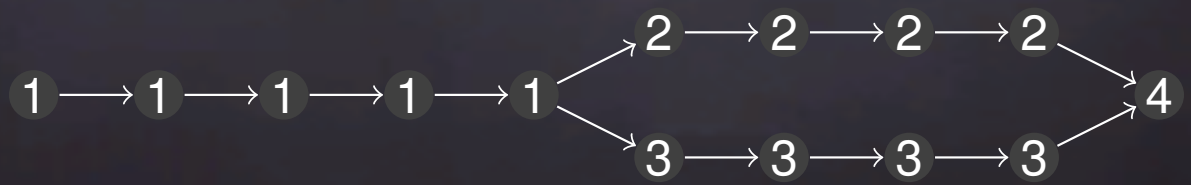
Assembly graph with variants & errors



3) Likely sequencing errors are removed. (not in FALCON)



3) Known biological events are removed. (not in FALCON)
4) Finally, **simple paths** (i.e. contigs) are returned.



Short note on reverse complements

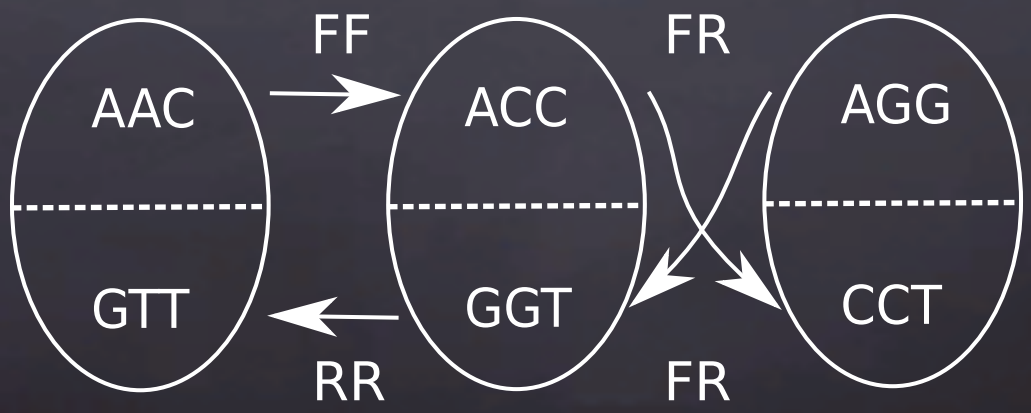
Because sequencing is generally not strand-specific:

> *In assembly, we always consider reads (and k-mers) are equal to their reverse complements.*

E.g:
AAA = TTT
ATG = CAT

In de Bruijn graphs, nodes implicitly represent both strands.

Lexicographically minimal *k*-mer is chosen as representative



Exercise

In this exercise, for simplicity, ignore reverse complements.
Reads:

```
TACAGT
  CAGTC
   AGTCAG
     TCAGA
```

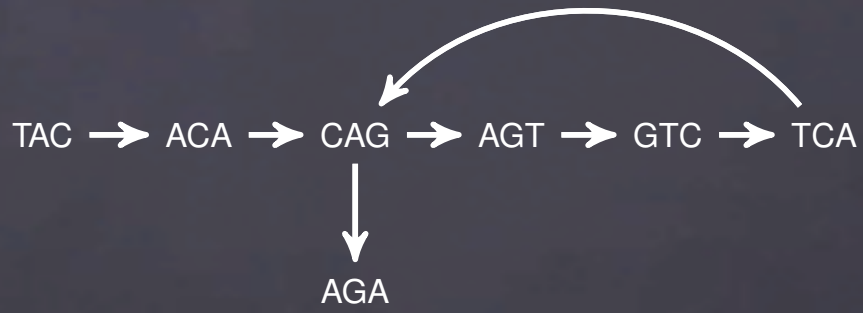
1. Construct the de Bruijn graph for $k = 3$.
   (Reminder: nodes are k-mers and edges correspond to $(k-1)$-overlaps)
2. How many contigs can be created?
3. At which value of k is there a single contig?
4. (optional) Find a mathematical relationship between k_a, the smallest k value with which a genome can be assembled into a single contig (using a de Bruijn graph), and ℓ_r, the length of the longest exactly repeated region in that genome.

Exercise (solution)

In this exercise, for simplicity, ignore reverse complements.
Reads:

```
TACAGT
  CAGTC
   AGTCAG
     TCAGA
```

1. Construct the de Bruijn graph for $k = 3$.
   The 3-mers (nodes) are: TAC, ACA, CAG, AGT, GTC, TCA, AGA

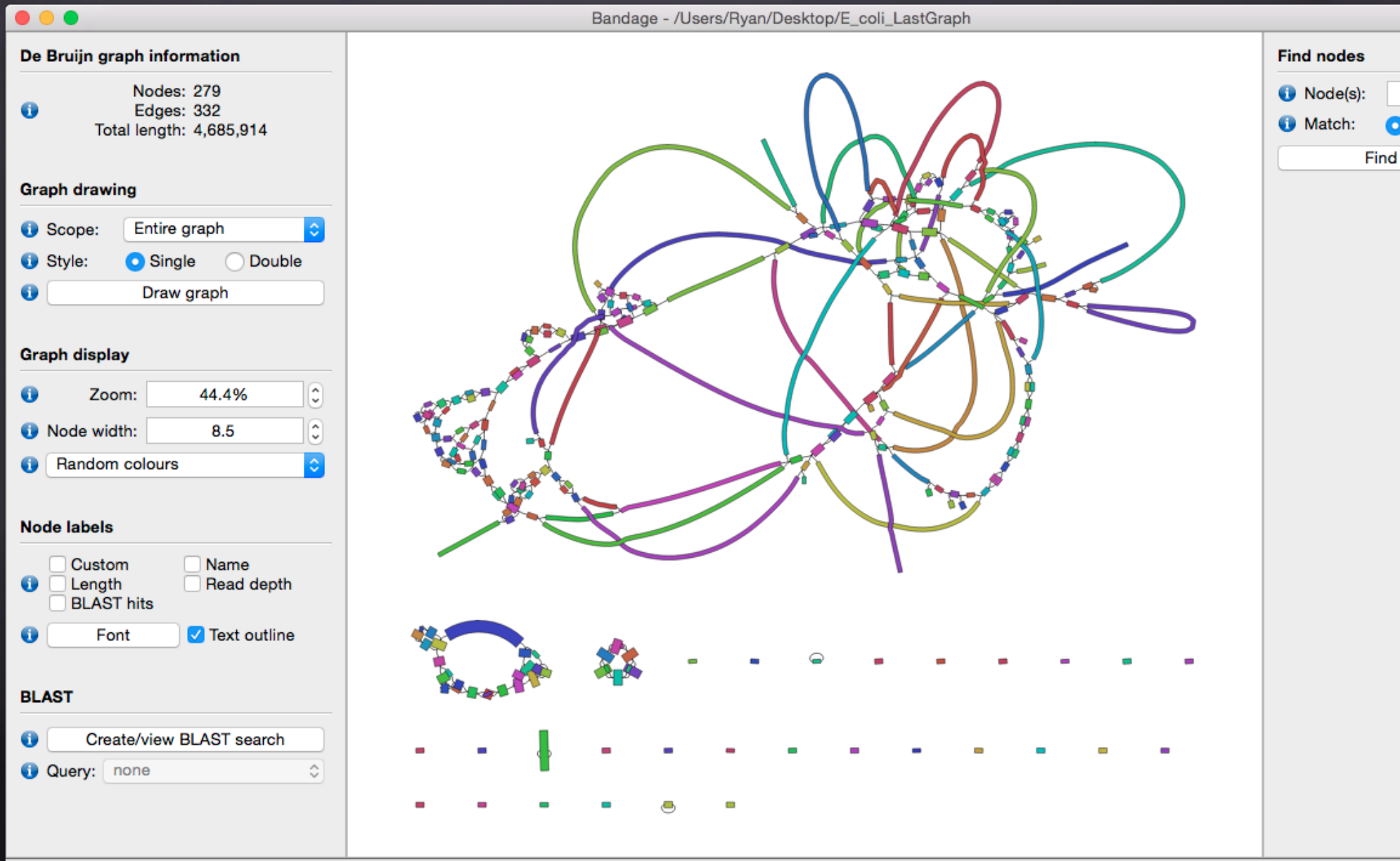


2. How many contigs can be created? 3
3. At which value of k is there a single contig? 5
4. Find a mathematical relationship between k_a, the smallest k value with which a genome can be assembled into a single contig (using a de Bruijn graph), and ℓ_r, the length of the longest exactly repeated region in that genome. $k_a = \ell_r + 2$

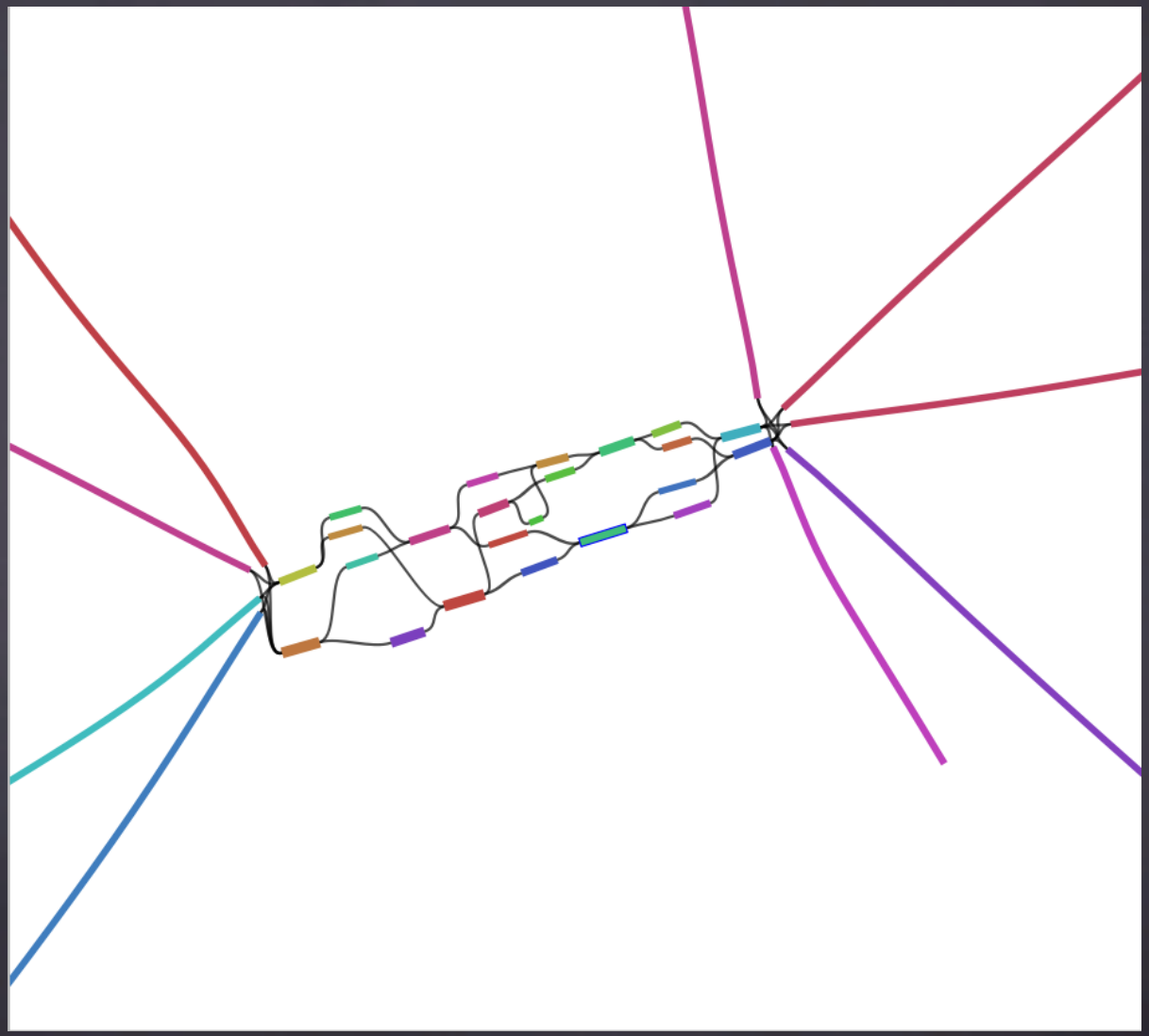
Plan

- What is a de novo assembly
  - Basics
  - Short Exercise
- Some useful assembly theory
  - Graphs
  - Contigs construction
  - Exercise
- Visualizing and evaluating assemblies
  - Bandage
  - Reference-free metrics
  - Exercise
- Assembly software
  - DNA-seq assembly
  - RNA-seq assembly
  - Multi-k and scaffolding
  - Other data types
  - Exercise

Assembly graph visualization: Bandage

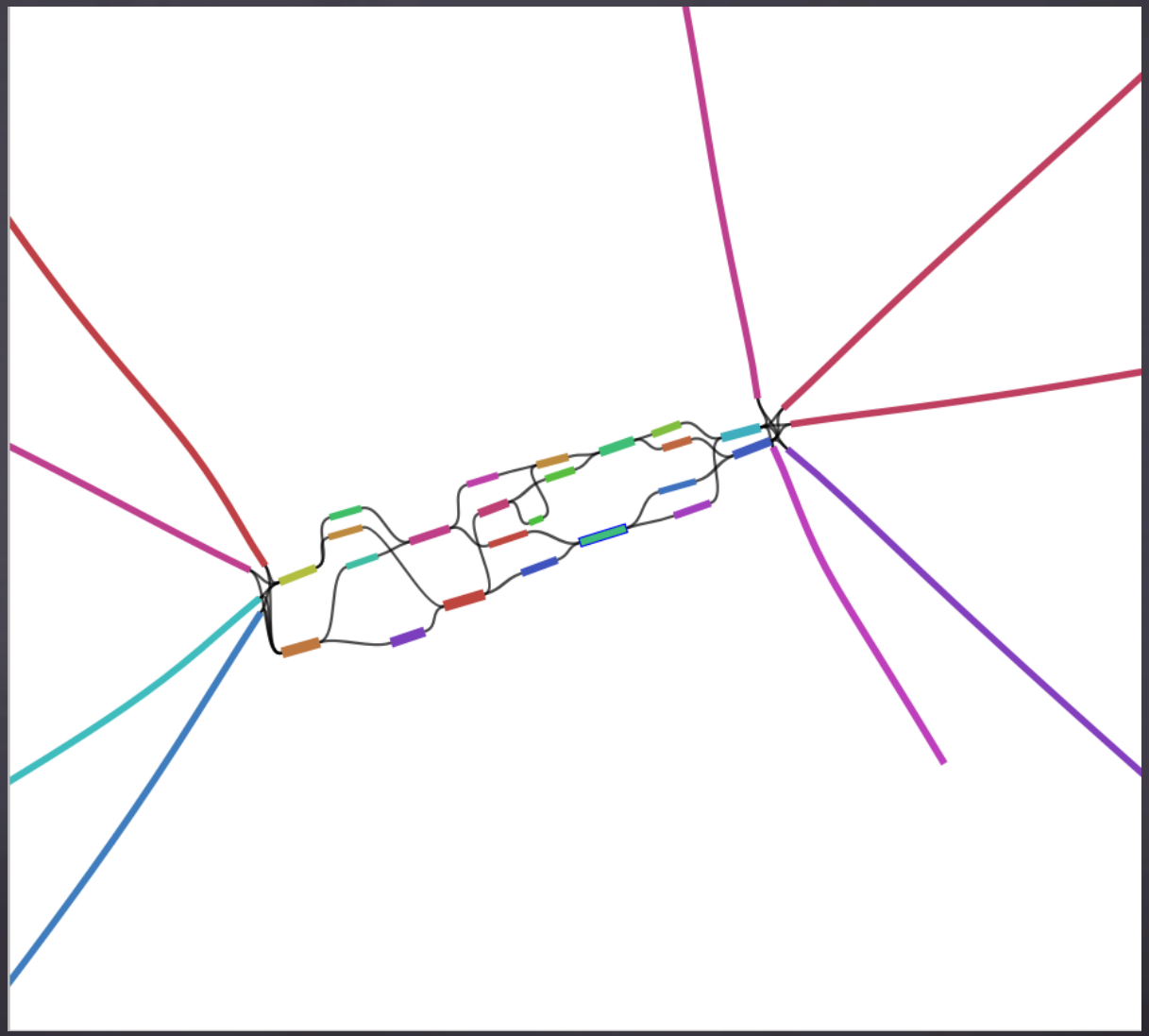


BANDAGE



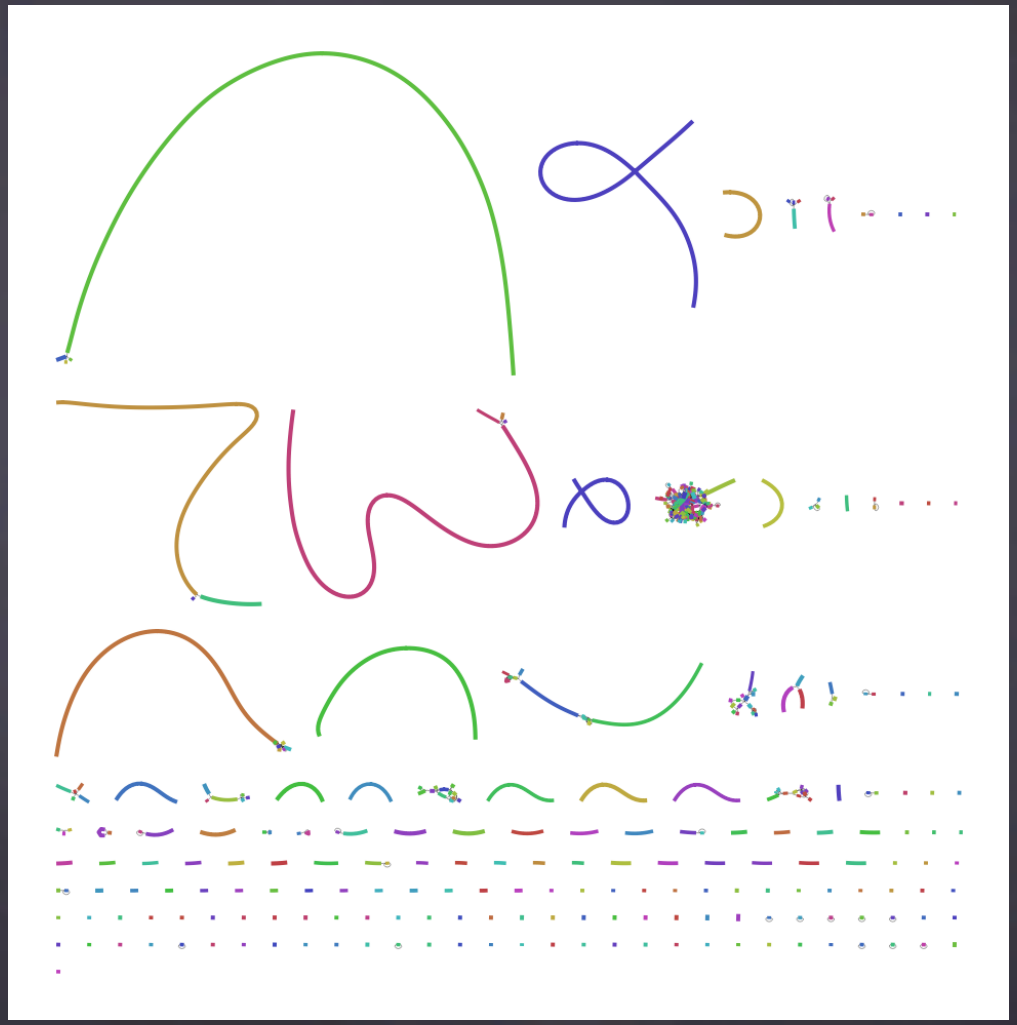
E. coli SPAdes assembly (excerpt). Fig from Lex Nederbragt. What is this knot?

BANDAGE



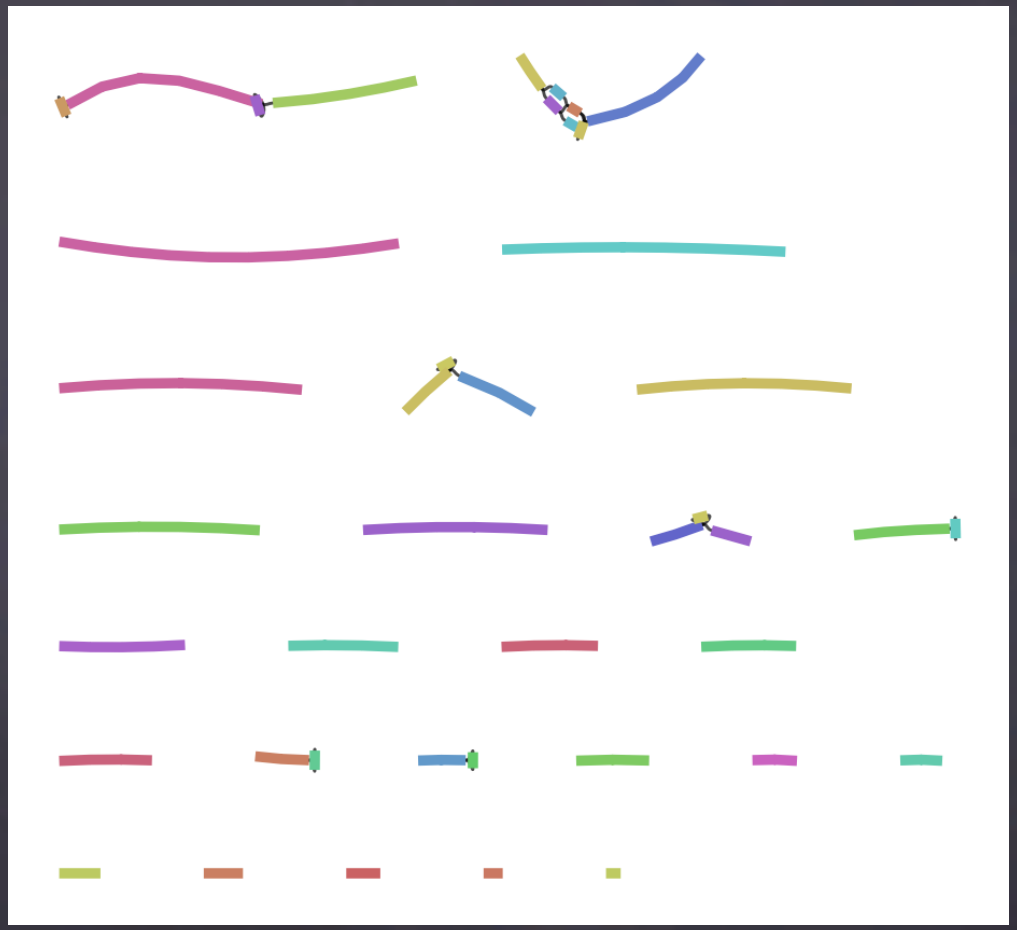
E. coli SPAdes assembly (excerpt). Fig from Lex Nederbragt. What is this knot?
collapsed ribosomal genes (16S, 2S, ..)

PacBio assembly visualization



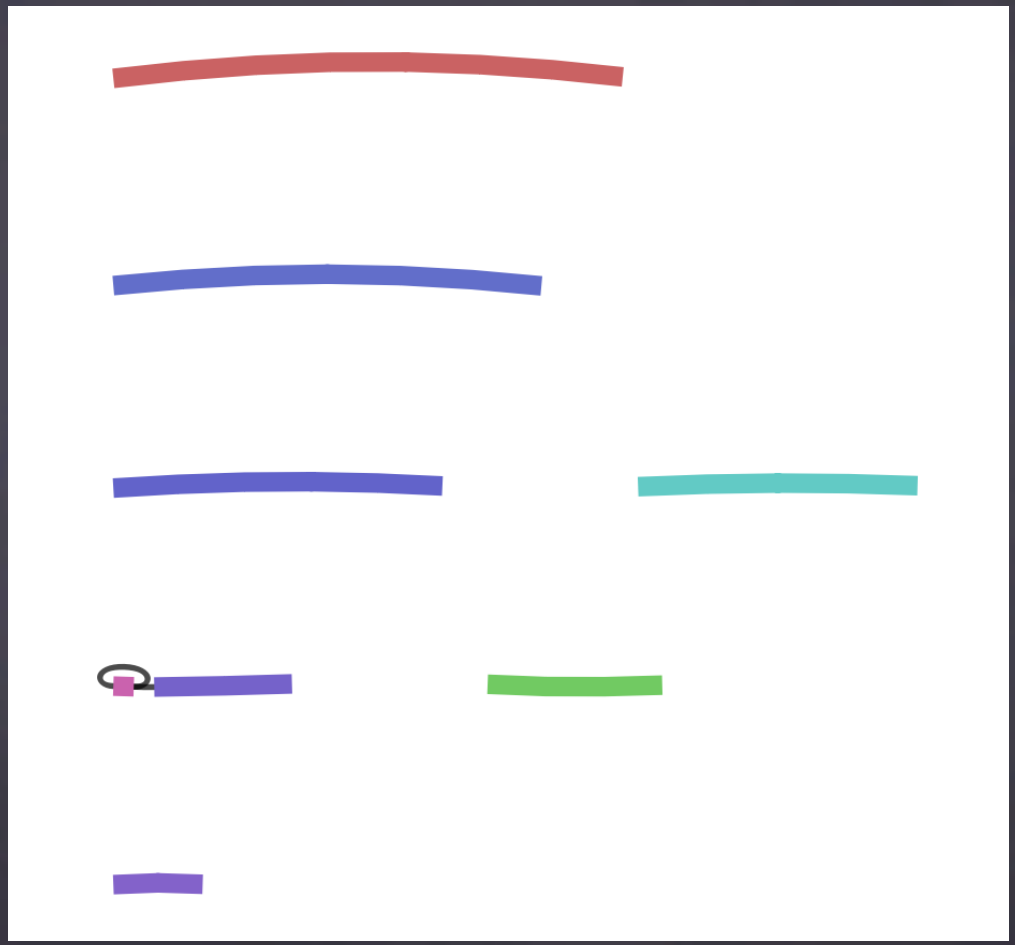
D. melanogaster FALCON assembly. Each node is a contig. (fig. courtesy of @md5sam)

Effect of coverage on the graph (PacBio data)



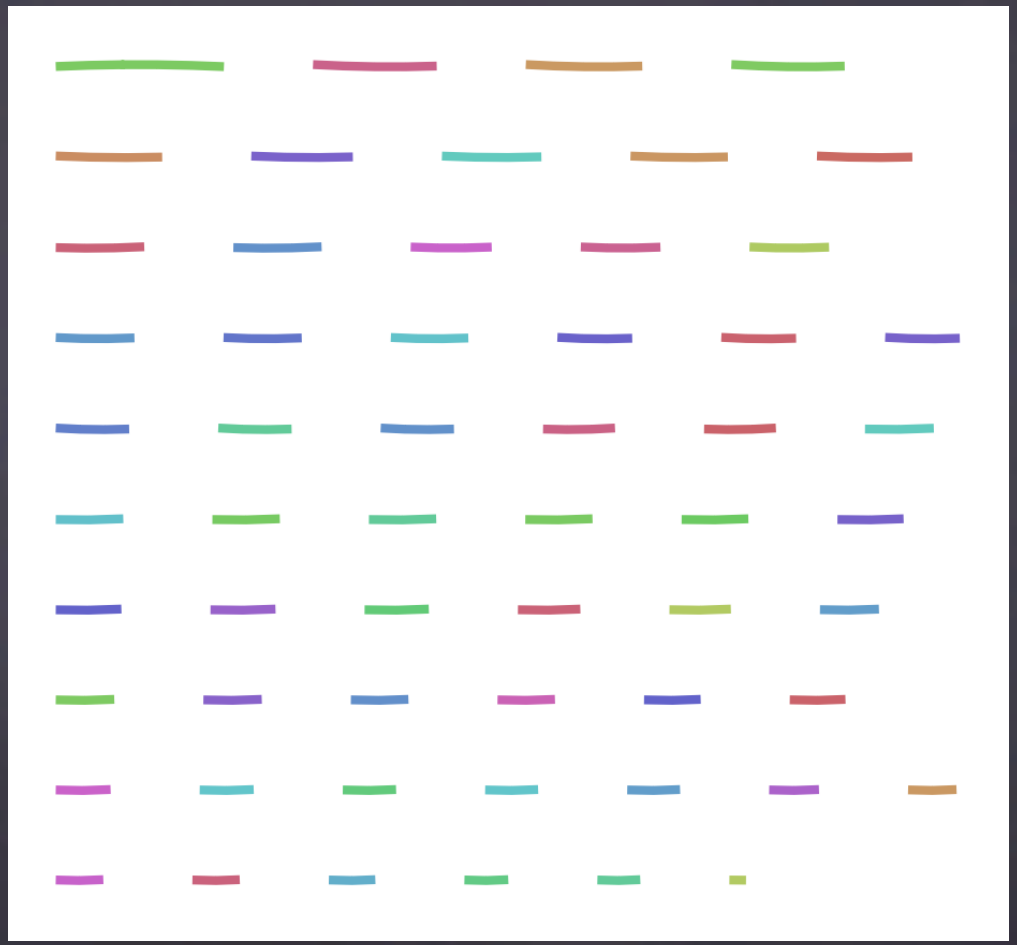
V. Cholerae, Canu assembly (4.2 Mbp total assembly size), 314x coverage

Effect of coverage on the graph (PacBio data)



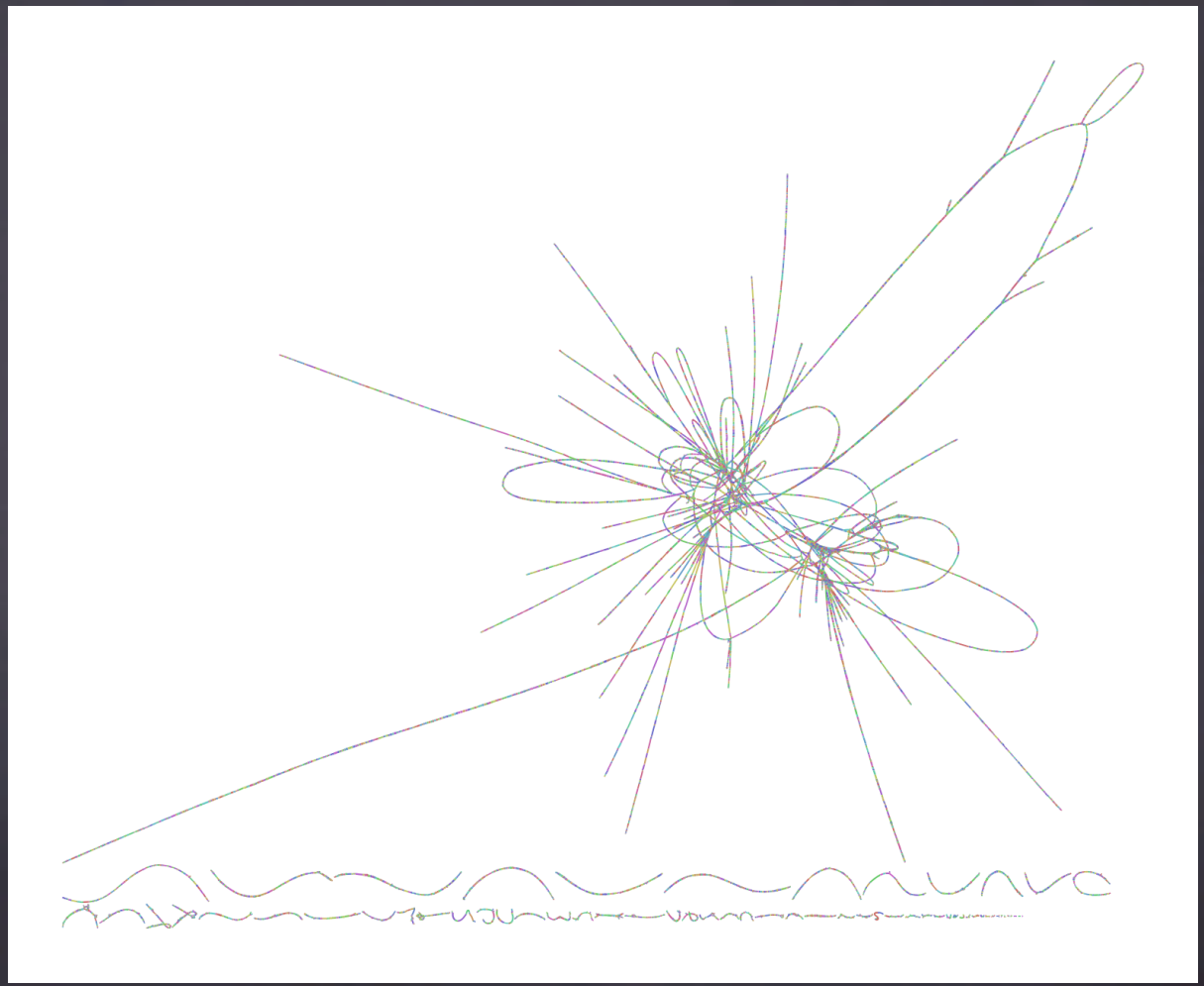
V. Cholerae, Canu assembly (4.2 Mbp total assembly size), subsampled input: 88x coverage

Effect of coverage on the graph (PacBio data)



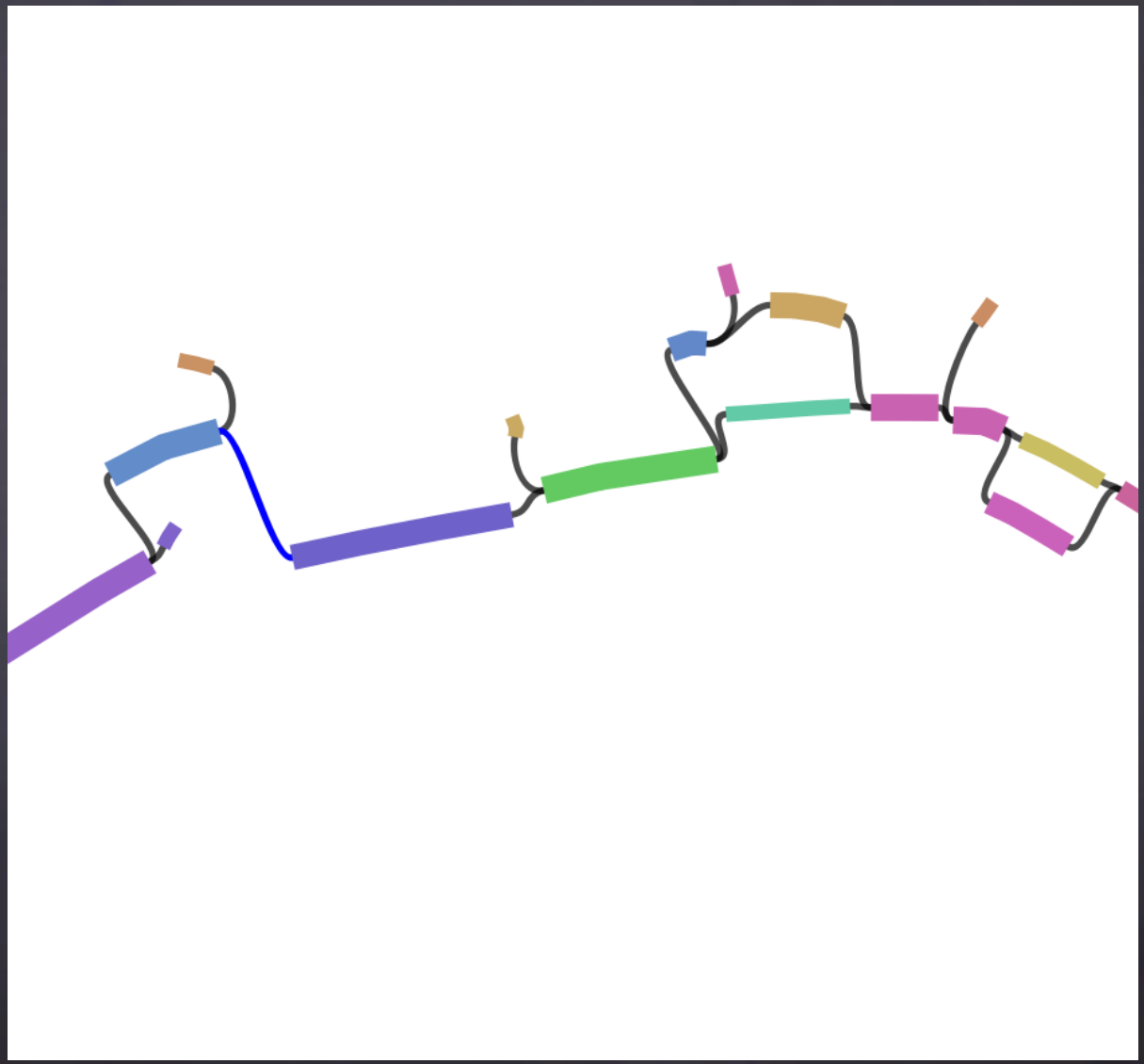
V. Cholerae, Canu assembly (1.3 Mbp total assembly size), subsampled input: 15x coverage

Effect of simplifications on the graph (Illumina data)



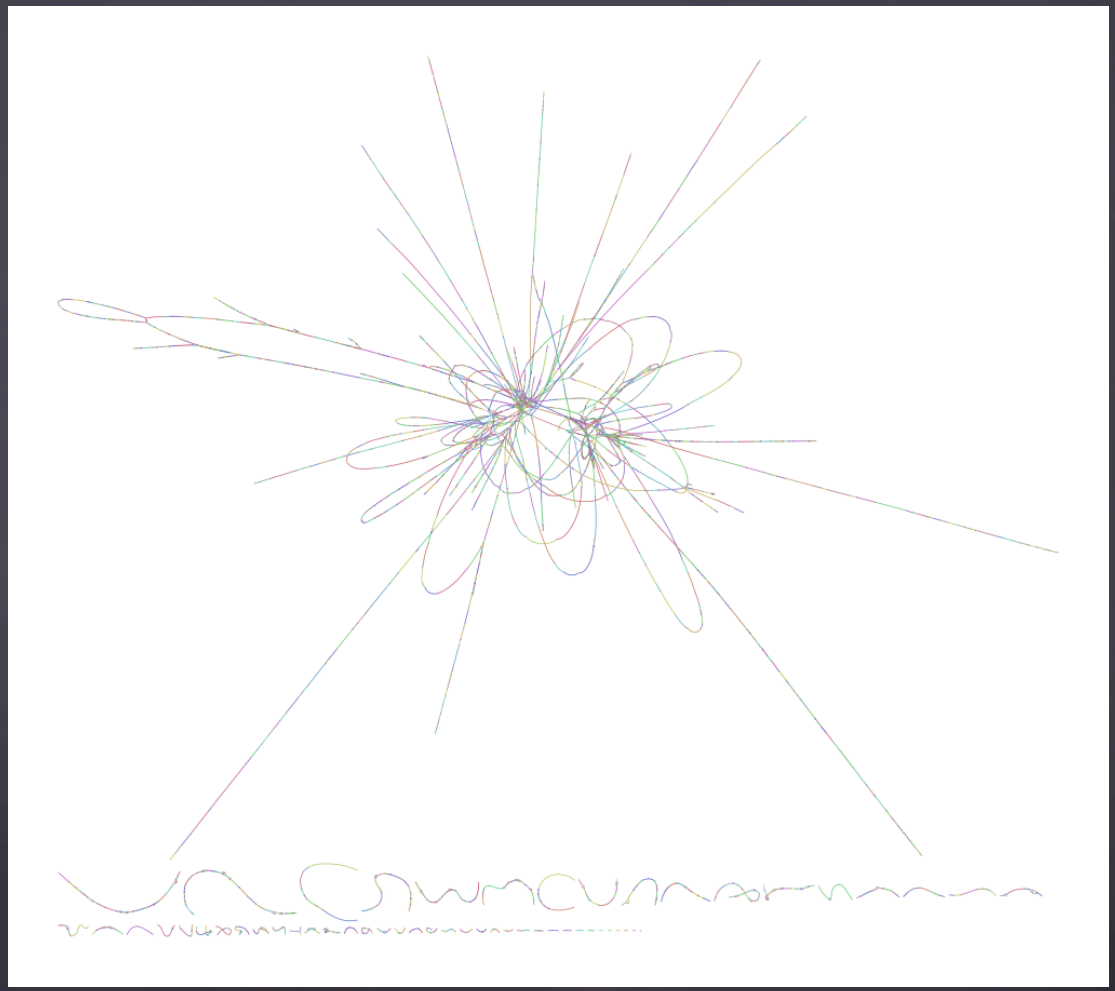
human chr14:20Mbp-20.5Mbp GAGE PE reads, Minia k=31, no graph simplifications at all, around 20k nodes

Effect of simplifications on the graph (Illumina data)



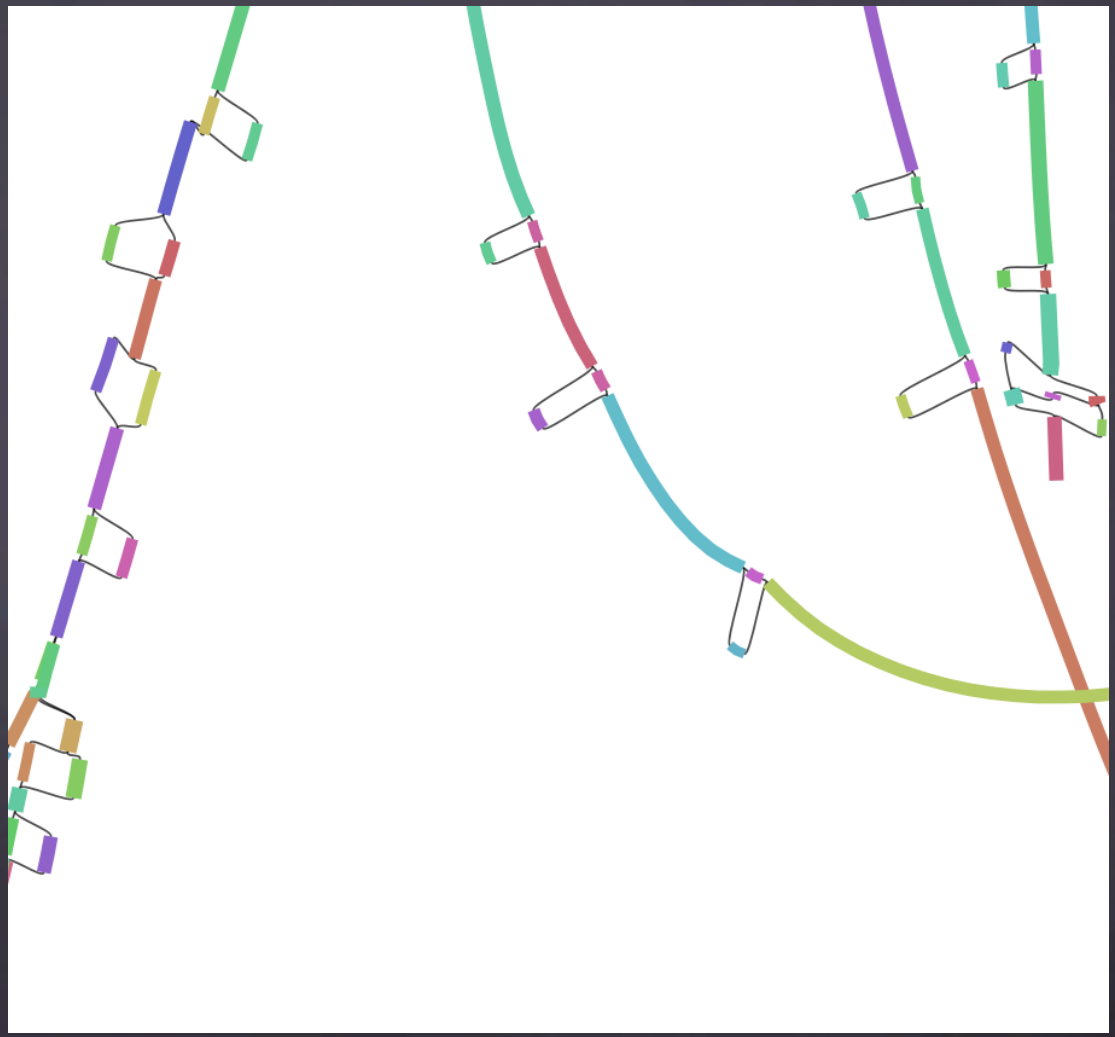
same data as previous slide, zommed in to see tips and bubble

Effect of simplifications on the graph (Illumina data)



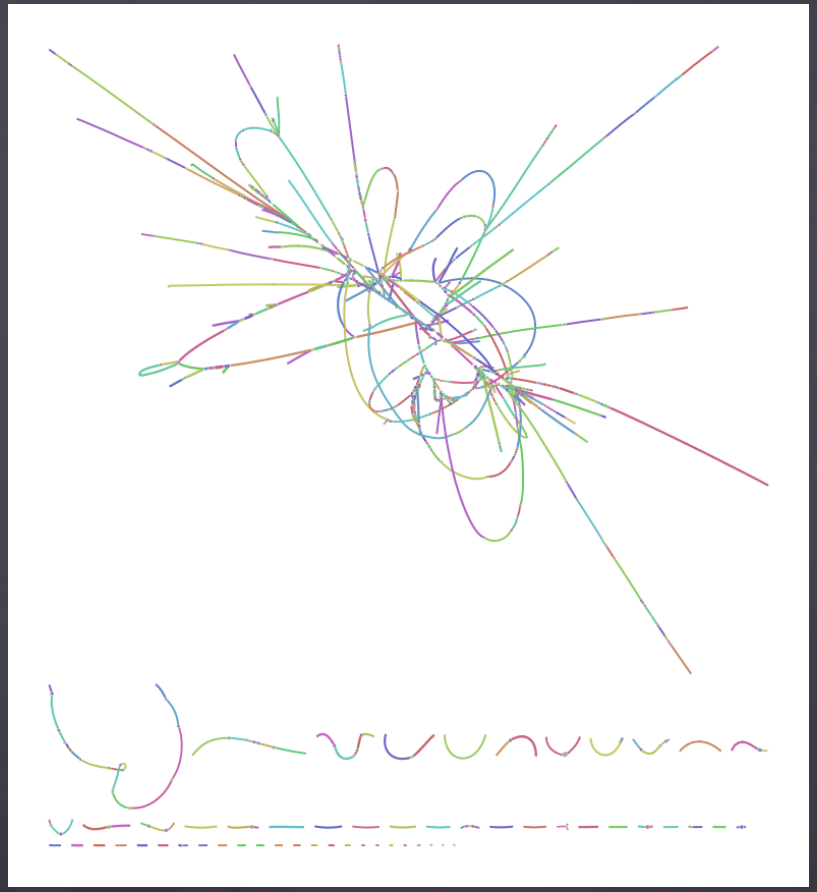
chr14:20Mbp-20.5Mbp GAGE PE reads, Minia k=31, with tips removed, around 6k nodes

Effect of simplifications on the graph (Illumina data)



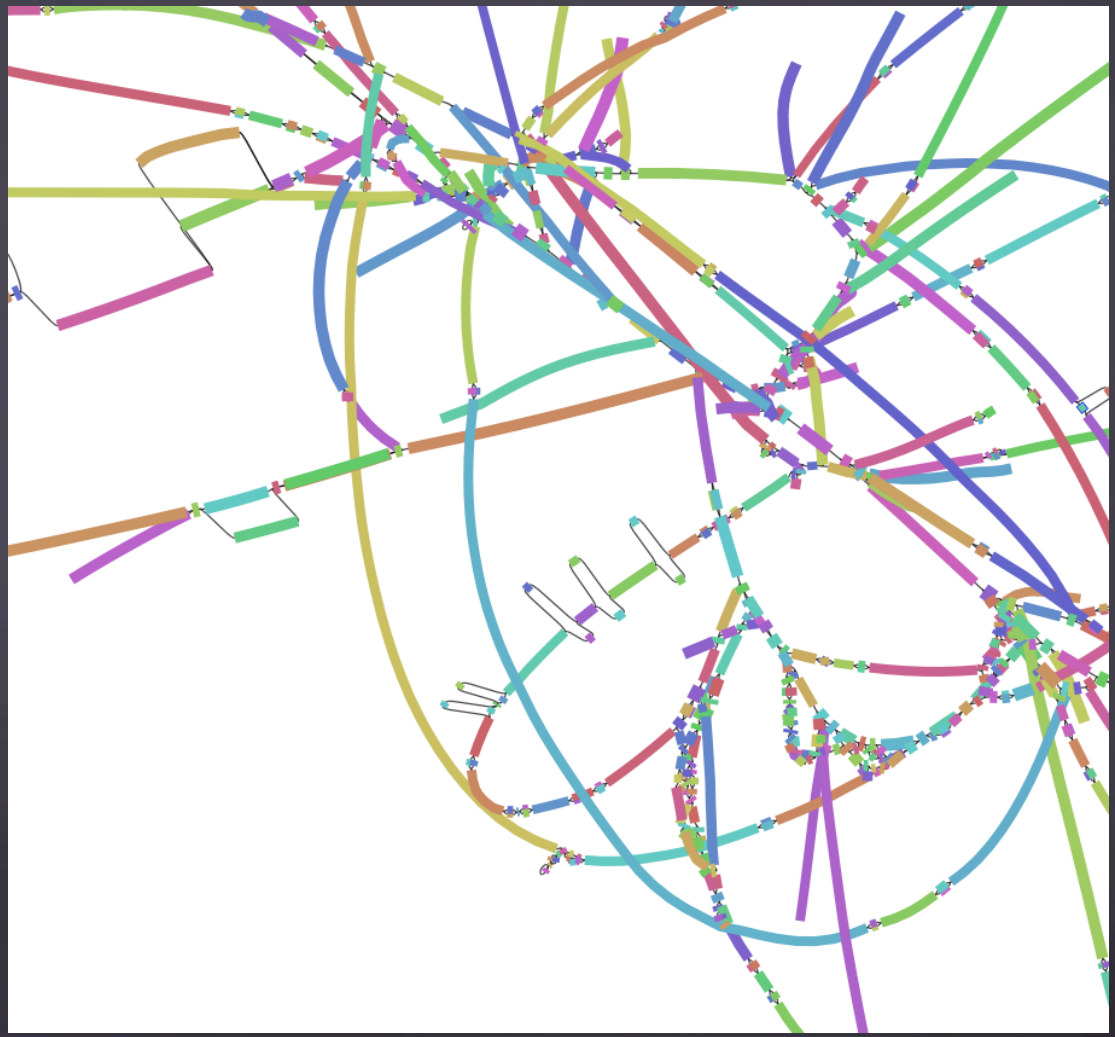
same data as previous slide, detail

Effect of simplifications on the graph (Illumina data)



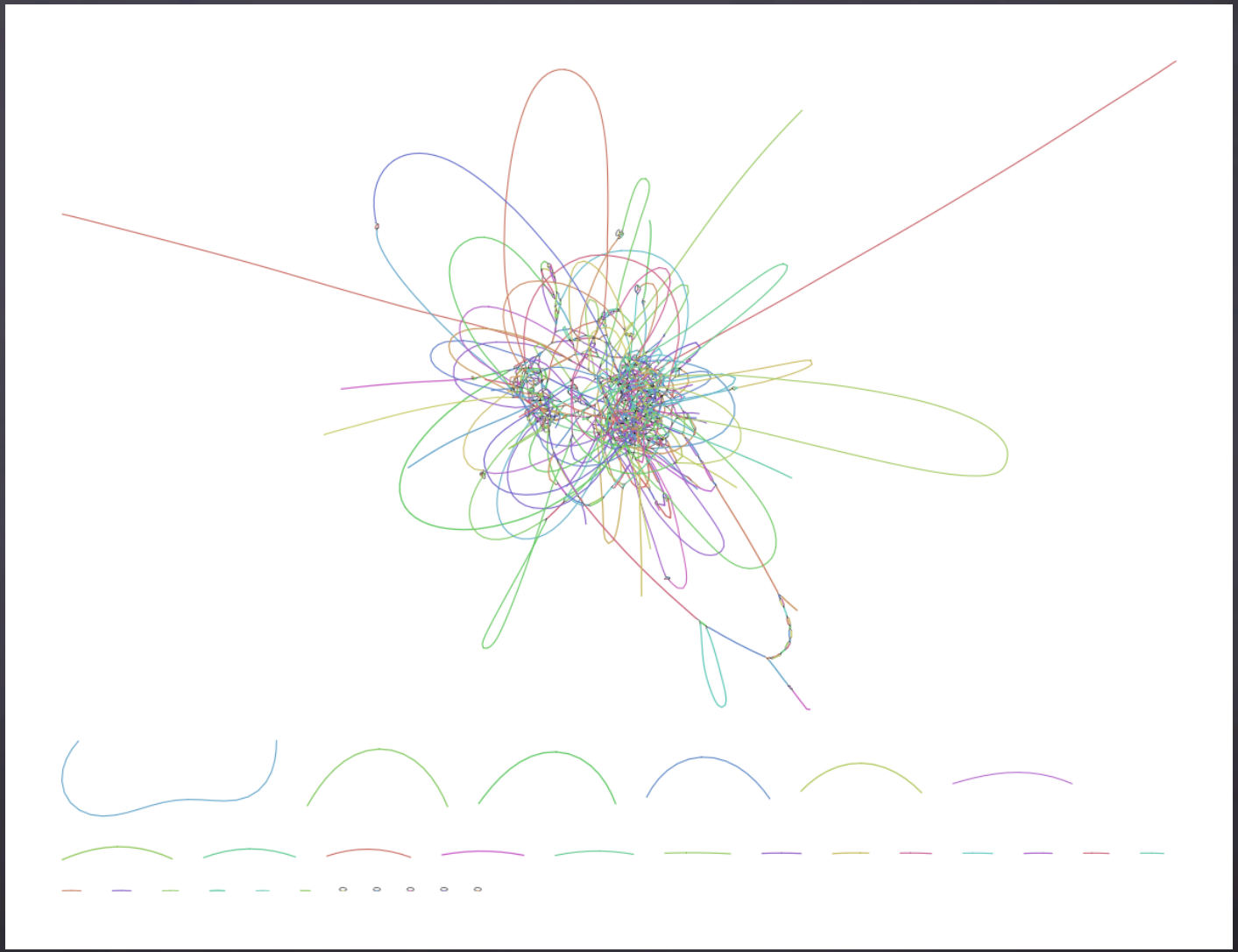
chr14:20Mbp-20.5Mbp GAGE PE reads, Minia k=31, all simplifications enabled, 1.3k nodes

Effect of simplifications on the graph (Illumina data)



same data as previous slide, detail

Effect of simplifications on the graph (Illumina data)



chr14:20Mbp-20.5Mbp GAGE PE reads, SPAdes 3.8 k=31, 1k nodes

Effect of simplifications on the graph (Illumina data)

same as previous slide, detail

Metrics

Preamble: There is no trivial total order (i.e. ranking) between assemblies.

Why? $>$ 2 independent criteria to optimize (e.g., total length, and average size of assembled sequences)

Example Would you rather have an assembly with high coverage and short contigs, or an assembly with low coverage and long contigs?

Overview of reference-free metrics

1. Individually evaluate a single assembly
2. Compare several assemblies made from different parameters or assemblers

Classical metrics: [QUAST]

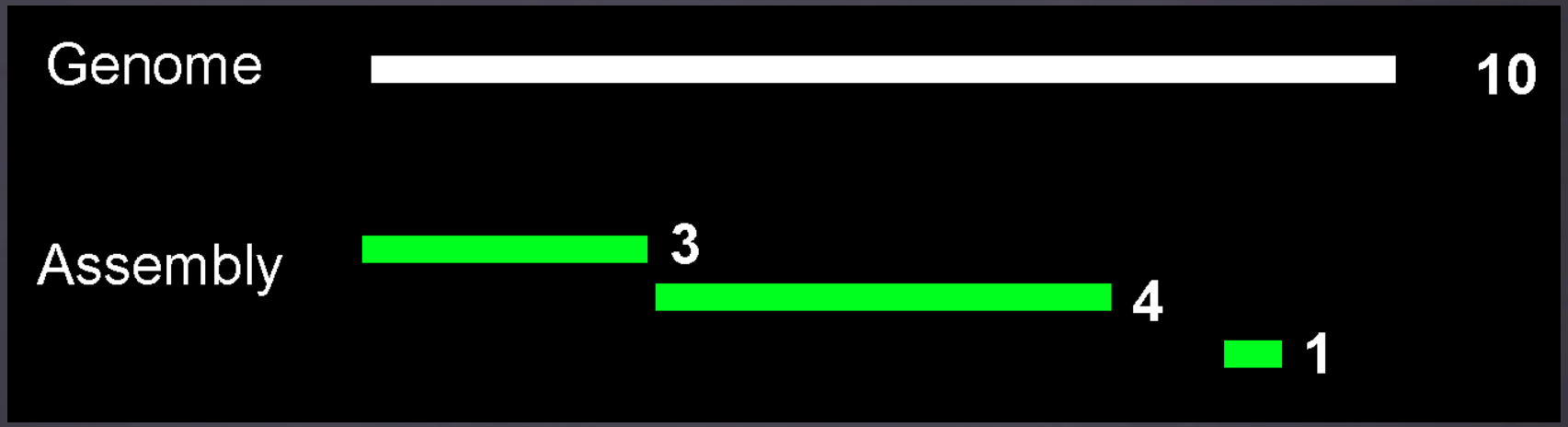
- Number of contigs/scaffolds
- Total length of the assembly
- Length of the largest contig/scaffold
- Percentage of gaps in scaffolds ('N')
- N50/NG50 of contigs/scaffolds
- Number of predicted genes
- Number of core single-copy genes [BUSCO]

```
./quast.py assembly.fa
```

Reference-free metrics: N50

N50 = Largest contig length at which that contig and longer contigs cover 50% of the total **assembly** length

NG50 = Largest contig length at which that contig and longer contigs cover 50% of the total **genome** length



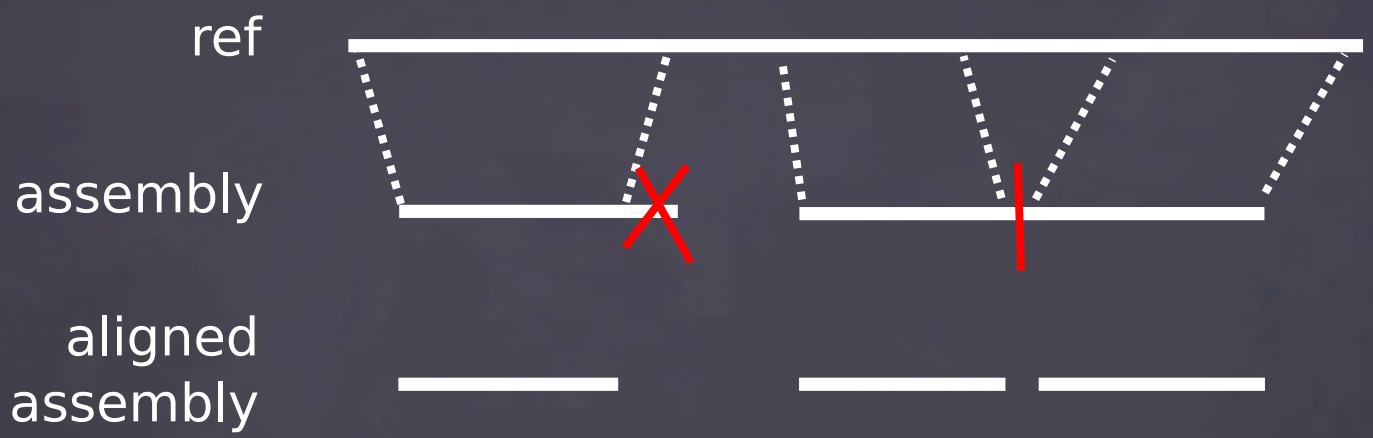
If you didn't know N50, write it down, there will be an exercise !

A practical way to compute N50:

- Sort contigs by decreasing lengths
- Take the first contig (the largest): does it cover 50% of the assembly?
- If yes, its length is the N50 value.
- Else, consider the two largest contigs, do they cover 50%?
- If yes, then the N50 is the length of the second largest contig.
- And so on..

Reference-based: NA50

The best metric no-one has heard of.



- Align contigs to reference genome.
- Break contigs at misassembly events and remove unaligned bases.
- Compute N50/NG50 of the result.

Other metrics of interest

Internal consistency : Percentage of paired reads correctly aligned back to the assembly (*happy* pairs).

Can pinpoint certain misassemblies (mis-joins).

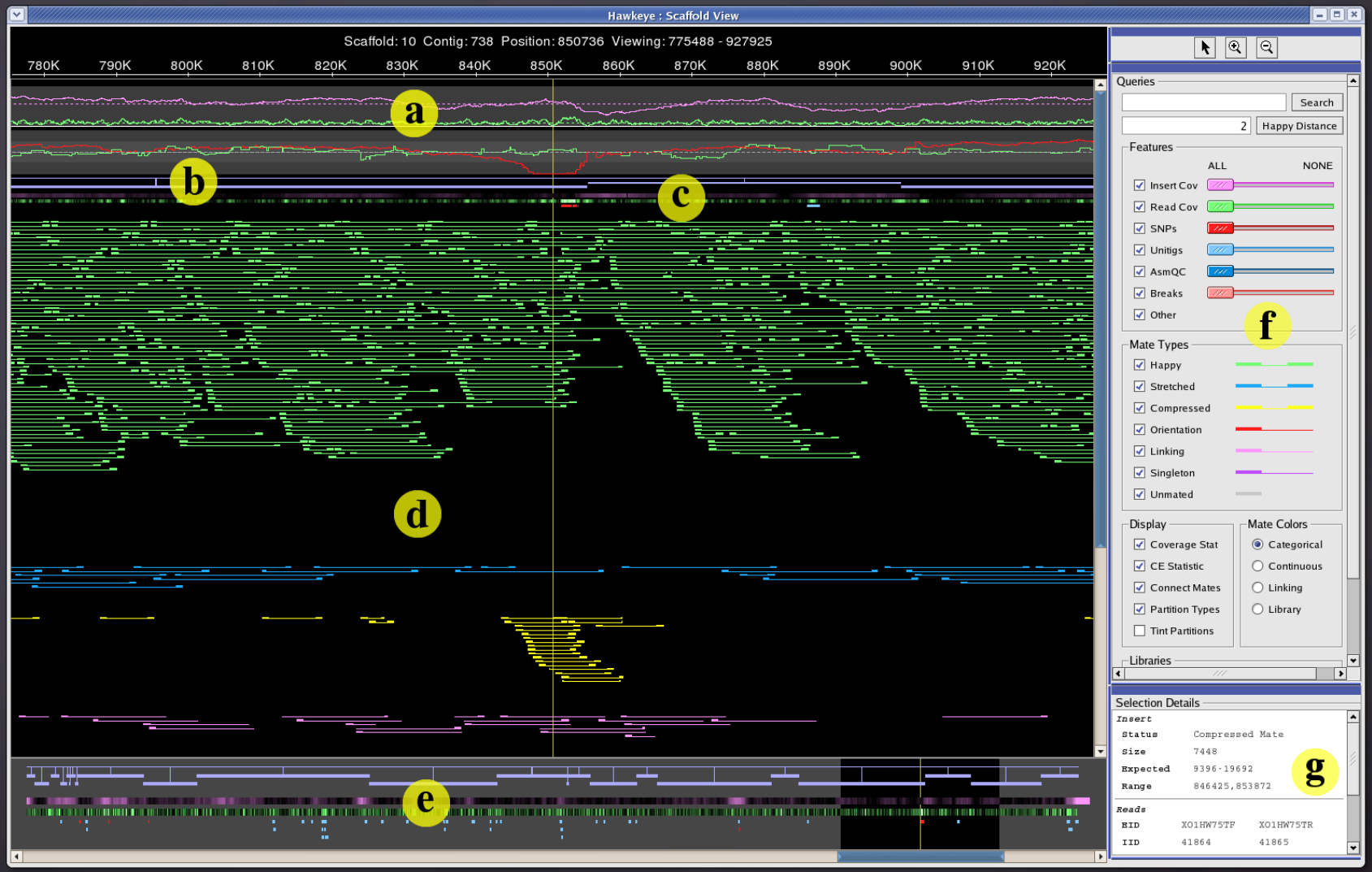
- REAPR [M Hunt, .. (Gen. Biol.) 2013]
- FRCurve [F. Vezzi, .. (Plos One) 2013]

Assembly Likelihood : $\prod_i p(r_i|A)$, where $p(r_i|A)$ is the probability that read r_i is sequenced if the genome was A

In practice, $p(r_i|A)$ is estimated by aligning r_i to the assembly.

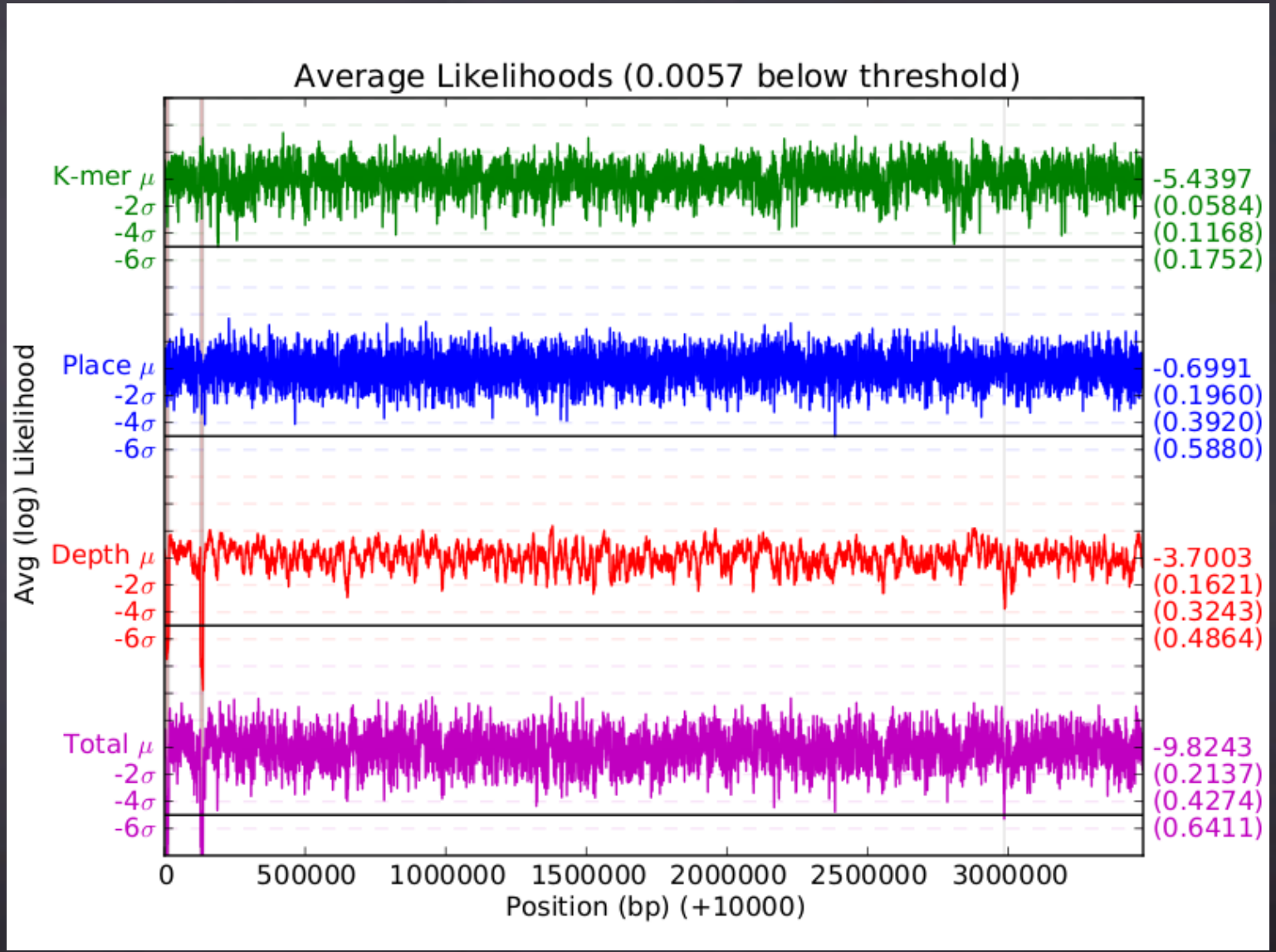
- ALE [Clark, (Bioinf.) 2013]
- CGAL [Rahman, (Gen. Biol.) 2013]
- LAP [Ghodsi, (BMC Res. Notes) 2013]

INTERNAL CONSISTENCY: EXAMPLE



Hawkeye software

ASSEMBLY LIKELIHOOD



ALE plot of likelihood windows over the E. coli genome.

Summary

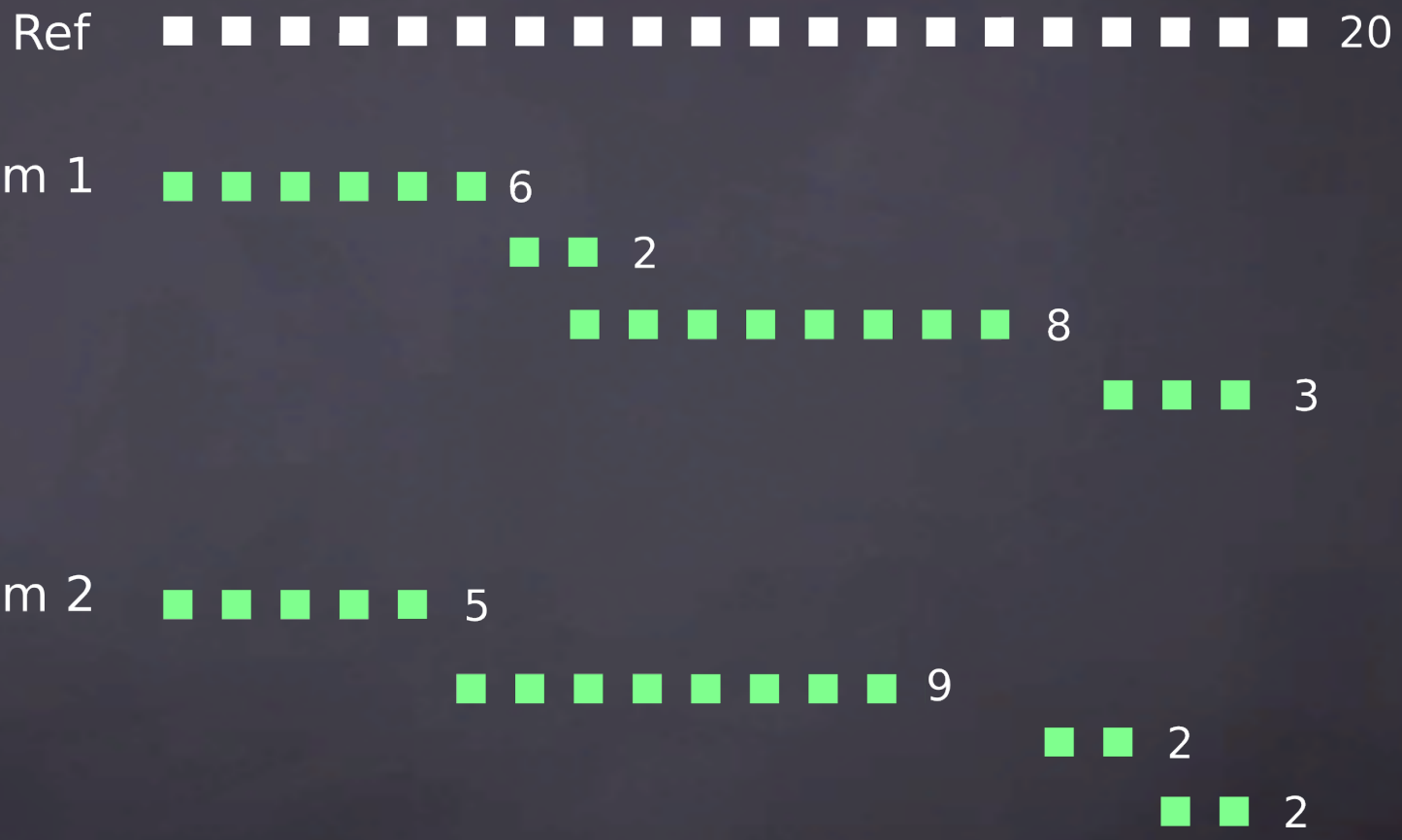
Google 'assembly uncertainty' for a nice summary, blog post by Lex Nederbragt.
In summary:

- No total order for metrics
- Use QUAST
- Use BUSCO

EXERCISE

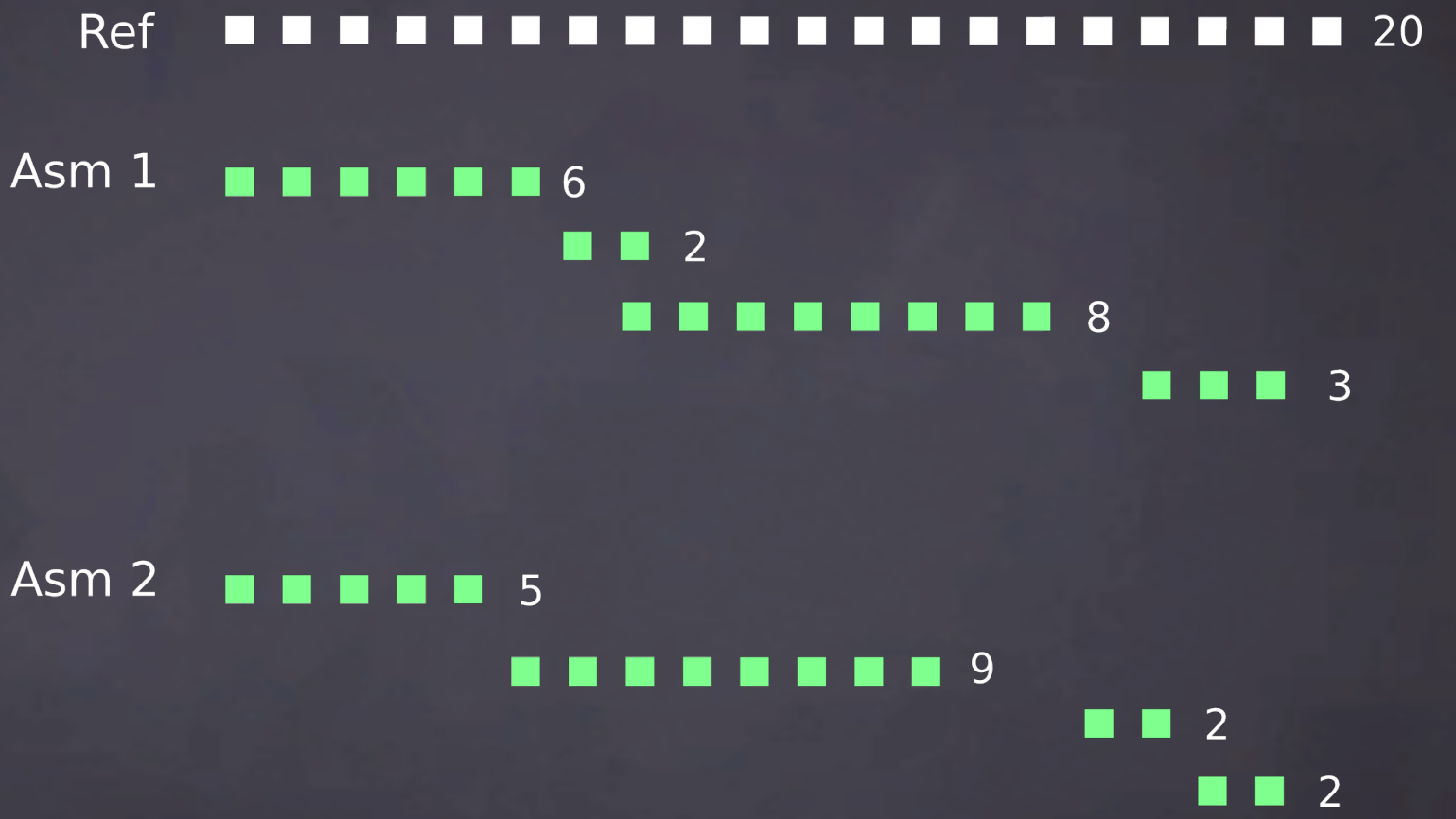
At some point in life, one may need to compare assemblies.

Here are two assemblies, aligned to the same reference:



- For each, compute the following metrics:
  - Total size of the assembly, N50, NG50 (bp)
  - Coverage (%)
- Which one is better than the other?

Exercise (solution)



- For each, compute the following metrics:
  - Total size of the assembly (19 bp, 18 bp), N50 (6 bp, 9 bp), NG50 (6 bp, 5 bp)
  - Coverage (%) (90, 90)
- Which one is better than the other? (I would say first one: higher NG50, less contigs, same coverage as the other. But: has some redundancy.)

Plan

- What is a de novo assembly
  - Basics
  - Short Exercise
- Some useful assembly theory
  - Graphs
  - Contigs construction
  - Exercise
- Visualizing and evaluating assemblies
  - Bandage
  - Reference-free metrics
  - Exercise
- Assembly software
  - DNA-seq assembly
  - RNA-seq assembly
  - Multi-k and scaffolding
  - Other data types
  - Exercise

Recommended practices (genomes)

PacBio whenever you can. Keep an eye on Nanopore.

- Illumina:
  - Longest read lengths
  - $\geq 50x$ coverage, $\times$ ploidy number.
  - For 1 bacterial genome, no point going above $\approx 200x$.
  - Broad recipe: several mate pairs libraries of increasing size
  - SPAdes for small genomes, unclear for large genomes (maybe MaSuRcA, Discovar de novo, Minia).
- PacBio:
  - At least $30x$ too, for now.
  - Assemblers: Canu, SMARTdenovo, FALCON, (miniasm + a correction module)

Assemblers, personal experience, 2017

Most genomes SPAdes

Data following the Broad recipe Discovar de novo

Memory issues Minia

PacBio Canu

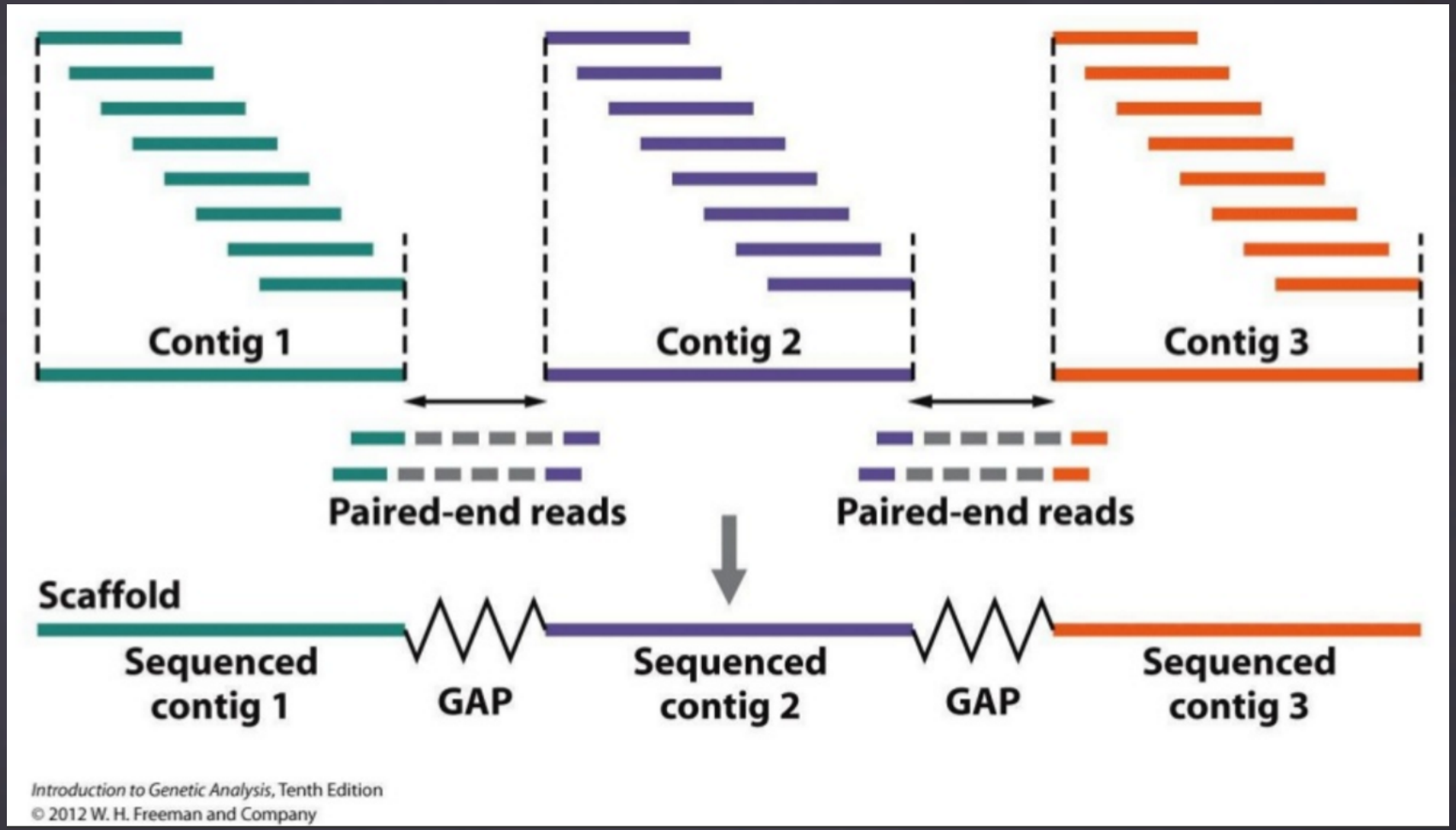
RNA-Seq Trinity

Large metagenomes Megahit

Meta-practices

1. Read Twitter and blogs for PacBio, Nanopore, metagenomes, assembly news.
2. Pick two assemblers
3. Run each assembler at least two times (different parameters set)
4. Compare assemblies
5. If possible, visualize them using Bandage

SCAFFOLDING



- Many scaffolders: SSPACE, BESST, Opera, SWALO
- Best strategy: mate-pairs libraries with many insert sizes
- Note: misjoins are mainly made during scaffolding

Hybrid assembly

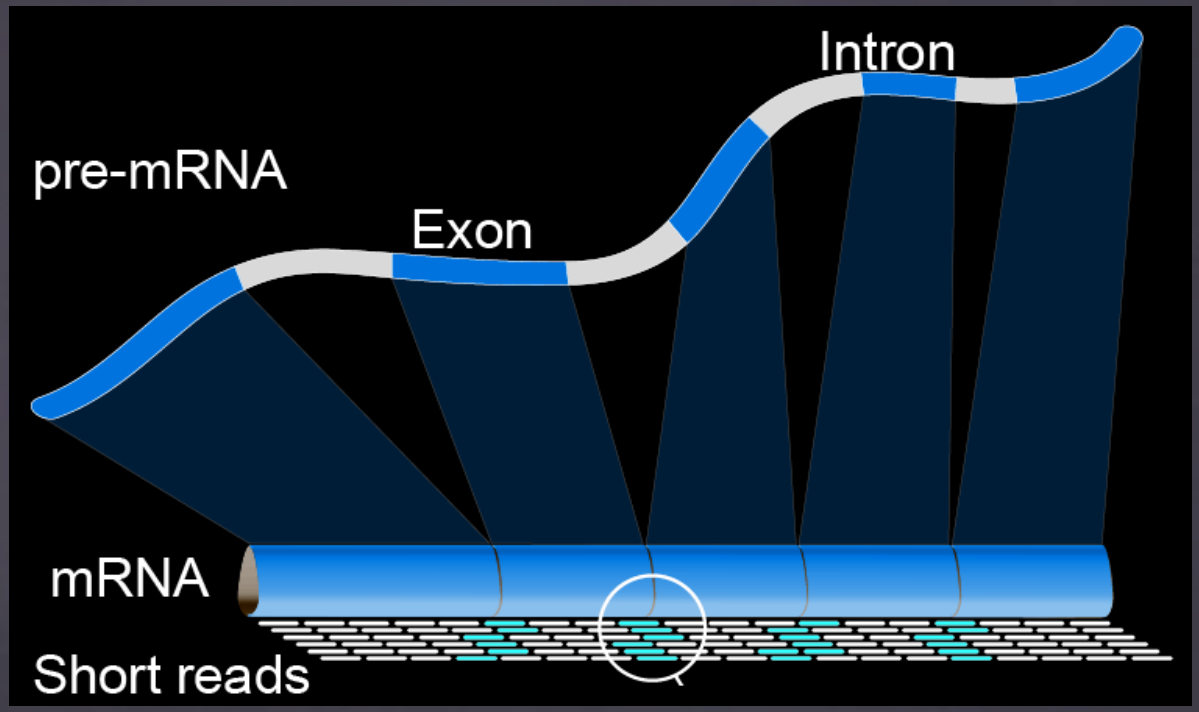
When you have multiple sources of data, e.g.

1. high-coverage Illumina paired-end / mate-pairs
2. low-coverage PacBio

Improve an Illumina assembly using:

- SSPACE-LR (scaffolding using PacBio reads)
- PBJelly (same but also gap-filling)

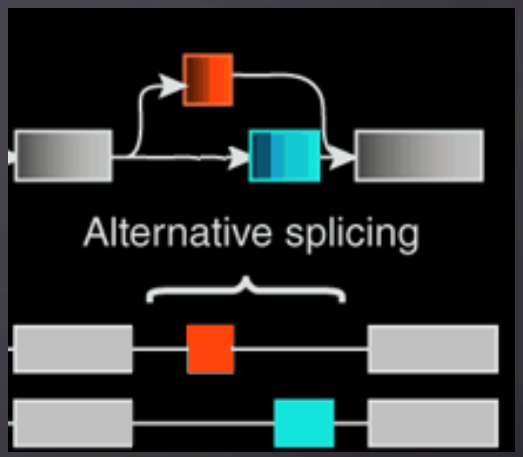
RNA-SEQ AND ASSEMBLY



Goal: reconstruct mRNA sequences

RNA-Seq Assembly

- Short contigs — average mRNA length: 2 kbp
- Uneven coverage — varying expression levels
- Contigs are re-used — alternative splicing



RNA-Seq Assembly

Despite these differences, DNA-seq assembly methods apply:

- Construct a de Bruijn graph (same as DNA)
- Output contigs (same as DNA)
- Allow to re-use the same contig in many different transcripts (new part)

RNA-Seq Assembly: Trinity

Quick overview of Trinity steps:

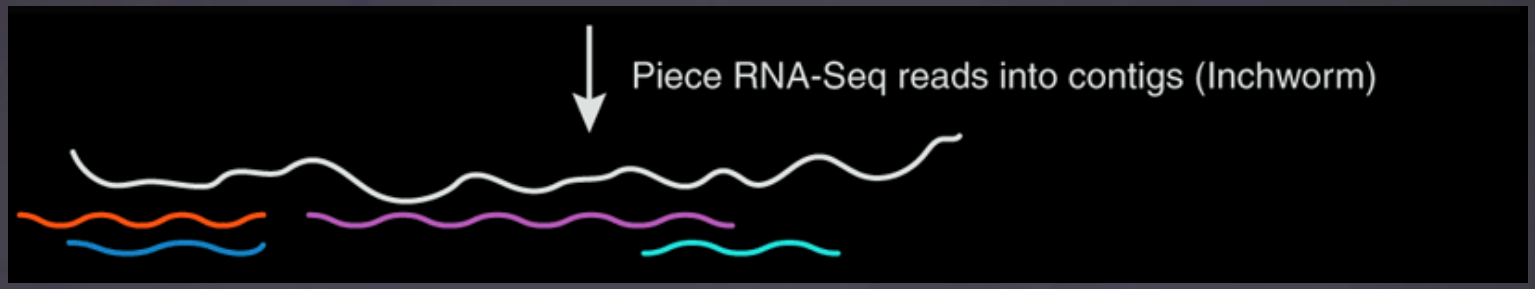
- Inchworm
- Chrysalis
- Butterfly

RNA-Seq Assembly: Trinity

- ~~Inchworm~~ de Bruijn graph construction, part 1
- ~~Chrysalis~~ de Bruijn graph construction, part 2, then partitioning
- ~~Butterfly~~ Graph traversal using reads, isoforms enumeration

RNA-Seq Assembly: Trinity - 1

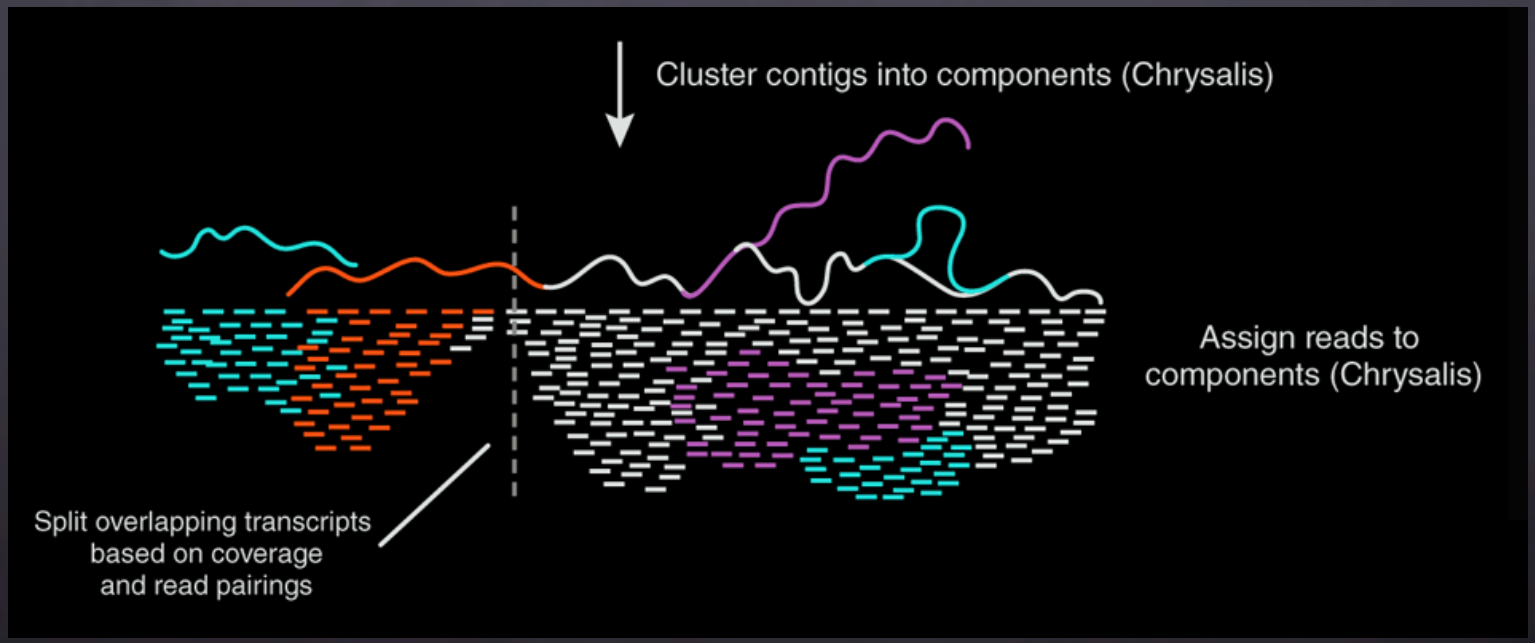
- **Inchworm**



Using *k*-mers, construct pieces of the de Bruijn graph.

Contigs might correspond to the most abundant isoform, but no guarantee.

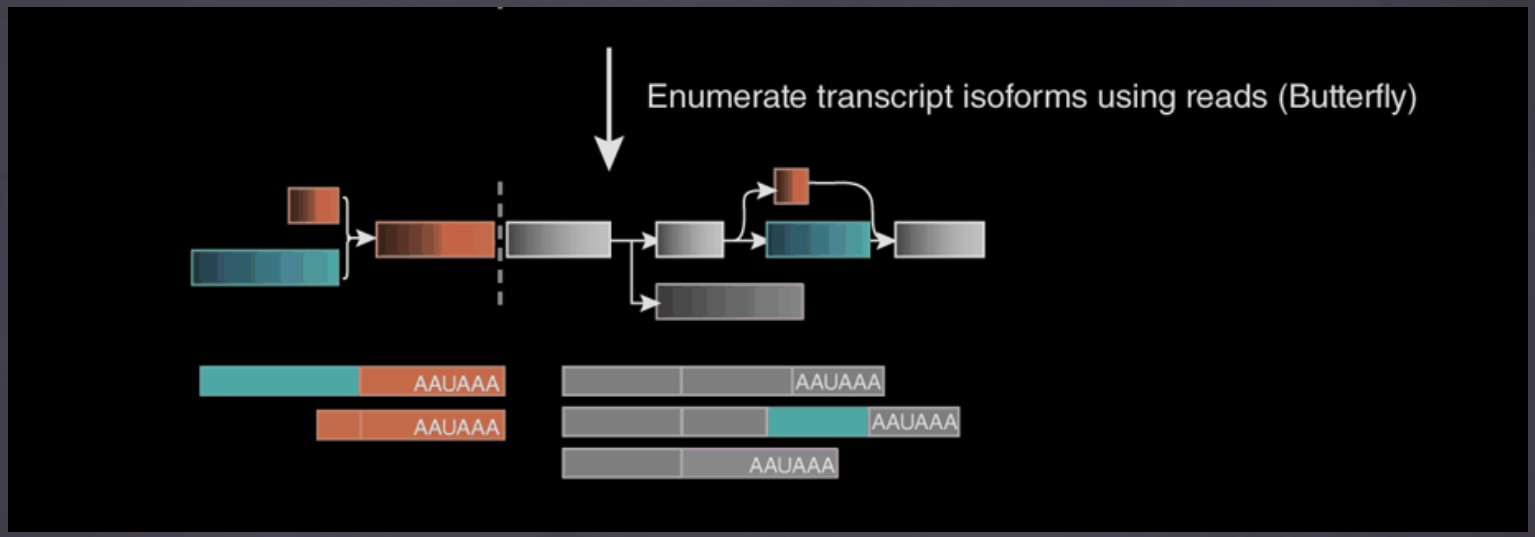
- **Chrysalis**



Construct the true de Bruijn graph. Use it to partition reads.

RNA-Seq Assembly: Trinity - 3

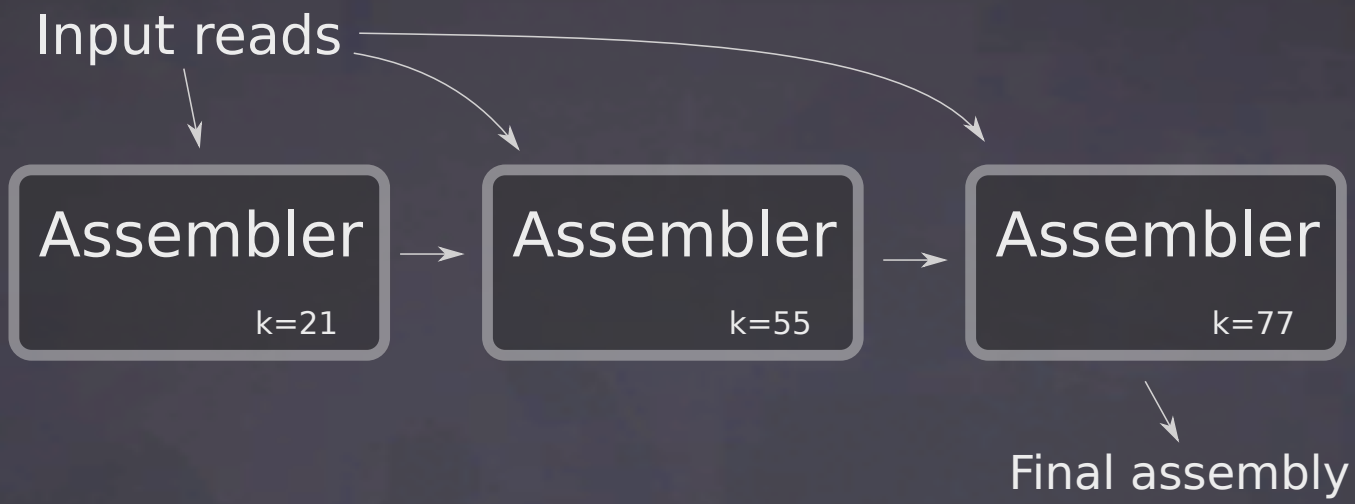
- **Butterfly**



Traverse each de Bruijn graph partition to output isoforms

Difference with DNA-seq assembly: isoforms are, by definition, not *k*-mer-disjoint.

Multi-k assembly



In principle, **better** than single-k assembly.

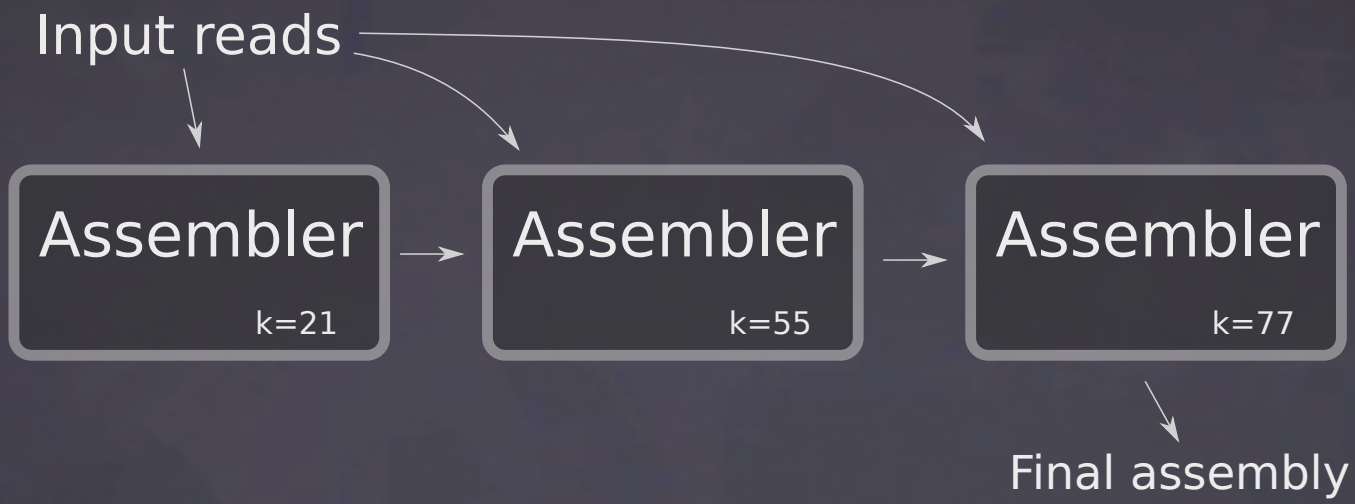
Notable assemblers that implement multi-k:

- IDBA, SPAdes, Megahit

Notable assemblers that don't:

- Velvet, SOAPdenovo, Trinity, ABySS

Multi-k assembly



In principle, **better** than single-k assembly.

Notable assemblers that implement multi-k:

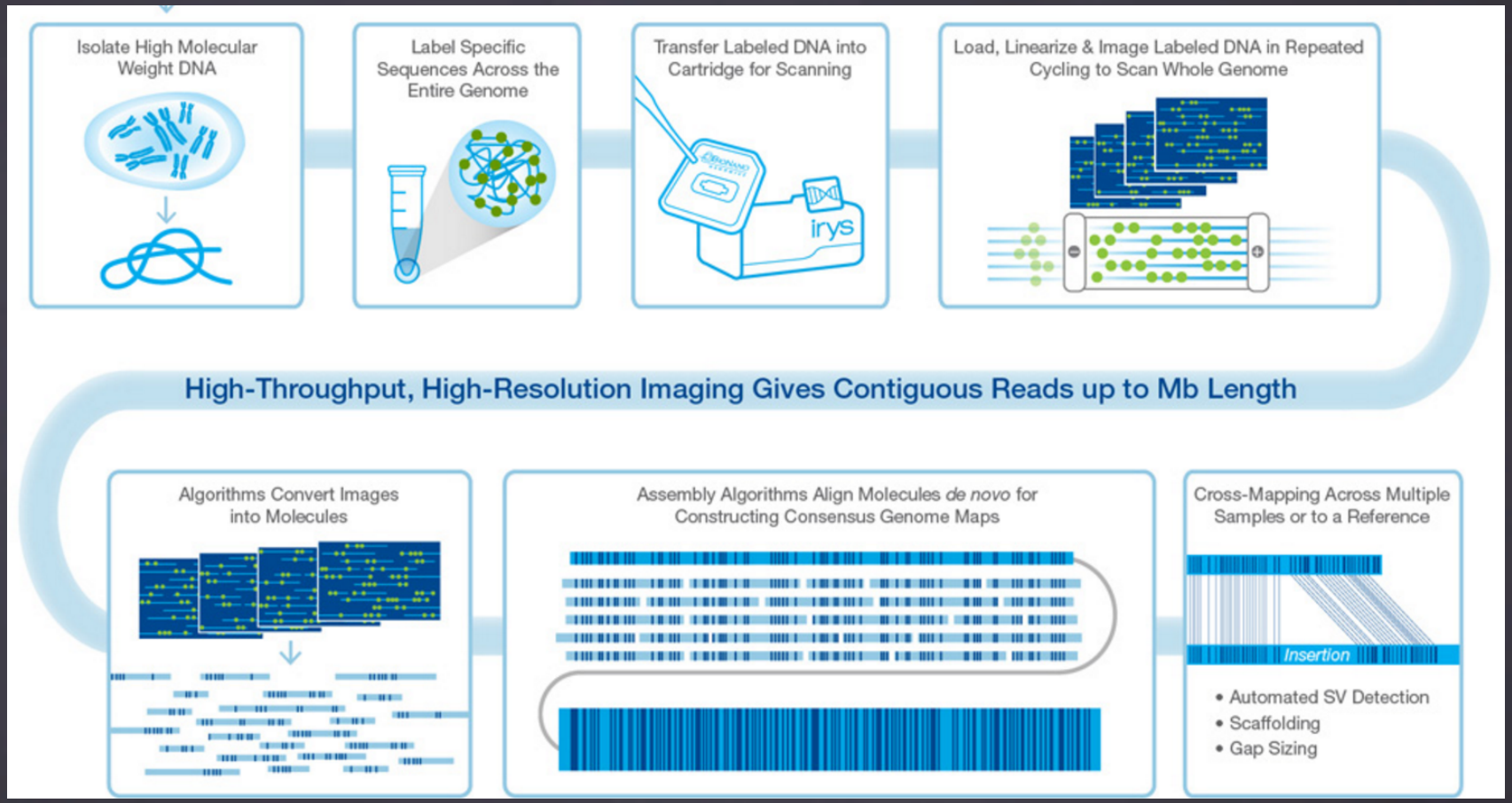
- IDBA, SPAdes, Megahit

Notable assemblers that don't:

- Velvet, SOAPdenovo, Trinity, ABySS

it's 2017, and we are still doing single-k assembly?

GENOME MAPS



Bionano promotional workflow slide

Other technologies: Dovetails, Nabsys, OpGen. Similar principles.

Chromonomer Order an assembly and correct scaffolds using RAD-seq markers, JoinMap, OneMap, r/QTL markers.

Common question: should I trim the reads?

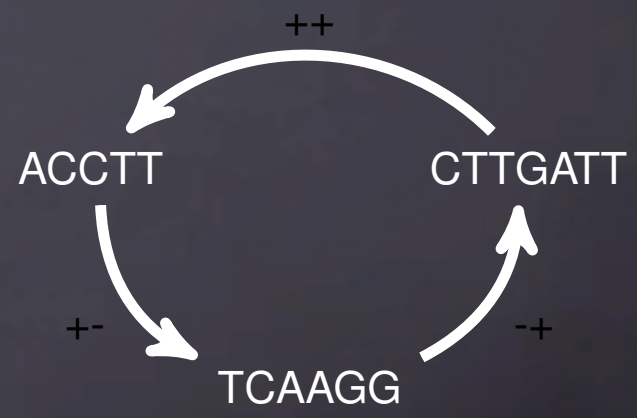
To check and remove adapters: yes absolutely

Quality-trim: I'd say no

Graph formats

- FASTG
- **GFA**
- GFA2

```
H    VN:Z:1.0
S    11   ACCTT
S    12   TCAAGG
S    13   CTTGATT
L    11   +    12   -    4M
L    12   -    13   +    5M
L    11   +    13   +    3M
P    14   11+,12-,13+ 4M,5M
```



Assembly: a solved problem?

Still challenging, even in 2017.

1. PacBio/Nanopore tools are slowly maturing
2. Hard to obtain good assemblies from Illumina data
3. High computational requirements overall

State of the research

1. Data-specific assemblers (PacBio vs. Illumina)
2. Efficient assemblers
3. Long-range data incorporation (e.g. 10x Genomics, Bionano)
4. Assembly-based variant calling

Last Exercise

Reads:

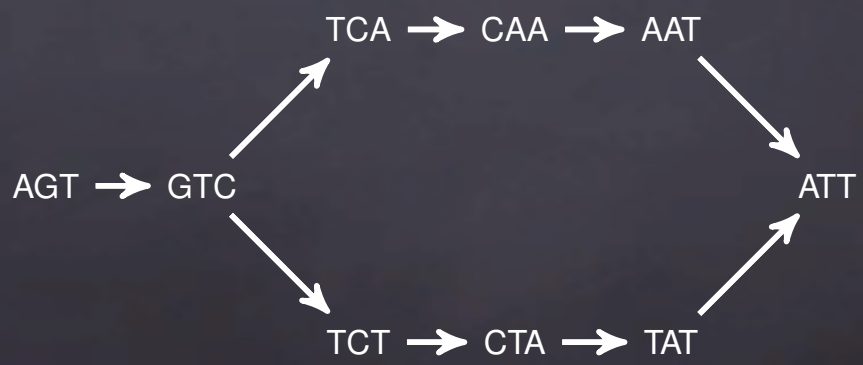
1. `AGTC`
2. `TCAA`
3. `AATT`
4. `GTCT`
5. `TATT`
6. `TCTA`
7. `TCAA`
8. `TCTA`

1. Assemble these reads
2. What was special about this genome?

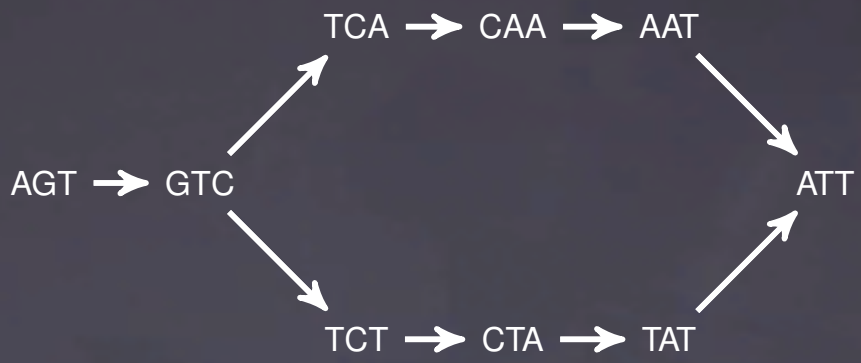
Last Exercise (detailed solution)

Step by step:

- **Choose an assembly model**: de Bruijn graph or string graph
- *The reads are short, let's choose the de Bruijn model*
- **Choose a k-mer size**:
- *Tempting to use $k = 4$, as it is the highest value such that k-mers exist in the reads. However, to obtain a good assembly, all 4-mers from the (unknown) sequenced genome need to be seen in the reads. This is a risky bet. Hence, let's pick a smaller k, $k = 3$.*
- The **nodes** of the graph are all the distinct 3-mers in the reads: AGT, GTC, TCA, CAA, AAT, ATT, TCT, TAT, CTA
- With an appropriate layout, the graph is:



Last Exercise (detailed solution)



- *To assemble this graph, using the contigs construction used before, there would be 4 contigs. Depending on how branching nodes are included in contigs, a possible solution is: AGTC, TCAAT, TCTAT, ATT.*
- But we can actually do better. There are two ways to traverse this graph, yielding an assembly of two "haplotypes":

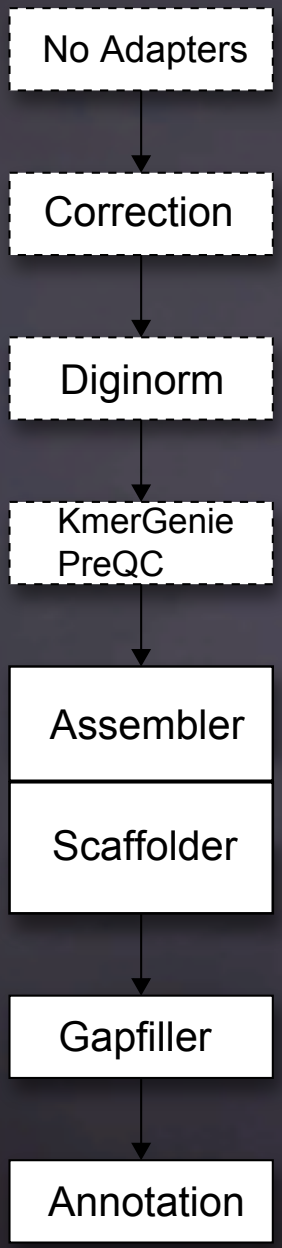
  `AGTCAATT`

  `AGTCTATT`
- This could be a tiny diploid genome with an heterozygous SNP. The bubble is unlikely to be a sequencing error, as I have purposely added reads 7 and 8, which make the *k*-mer coverage of both paths equally high.
- An assembler would collapse this bubble and output only one of the two haplotypes.

Conclusion, what we have seen

- What is a good assembly?
  - No total order
  - Main metrics: N50, coverage, accuracy
  - Use QUAST
- How are assemblies made?
  - Using a de Bruijn graph (Illumina) or a string graph (PacBio)
  - Errors and small variants are removed from the graph.
  - Contigs are just simple paths from the graph.
  - Scaffolds are linked contigs, misassemlies often happen there.
- Assembly software
  - Illumina: SPAdes ($\leq$ 100 Mbp genomes). For larger genomes, it's unclear.
  - PacBio: Canu, Miniasm, SMARTdenovo
- A few tips
  - Try another assembler
  - Try different parameters
  - An assembly is not the **absolute truth**, it is a mostly complete, generally fragmented and mostly accurate **hypothesis**

SUPPLEMENTAL SLIDE: ASSEMBLY PIPELINES



Supplemental slide: the choice of k

Choice of k is critical in dBG applications:

- k-mers with sequencing errors are noise
- only *non-erroneous* k-mers matter
- $k < \log_4(|\text{genome}|)$: nearly complete graph, uninformative
- small k: collapses repeats, more non-erroneous k-mers
- large k: less repeat collapsing, less non-erroneous k-mers (due to error and shortness of reads)

Generally, $k \geq 20$.
(Compare 4^k to the genome size.)
Higher sequencing coverage means larger k values can be used.