What can be done without a reference genome

Rayan Chikhi

Institut Pasteur

Workshop on Genomics - Cesky Krumlov
January 2019

Your instructor is..

- PI in bioinformatics at Institut Pasteur, Paris (since 2019)
- Was junior CNRS researcher in Lille, France
- Postdoc: Penn State, USA; PhD: ENS Rennes, France

Research:

- Software and methods for *de novo* assembly:
  - Minia
  - KmerGenie
  - BCALM
- Some “real” assemblies


@RayanChikhi on Twitter

`http://rayan.chikhi.name`

Course structure

- Short intro
- Fundamentals: objects for reference-free analysis
- Software for making assemblies
- Metrics: methods for assembly evaluation
- Visualization: see pretty assembly graphs
- In practice: multi-k ; scaffolding ; RNA-seq ; metagenomics

Questions to the audience

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

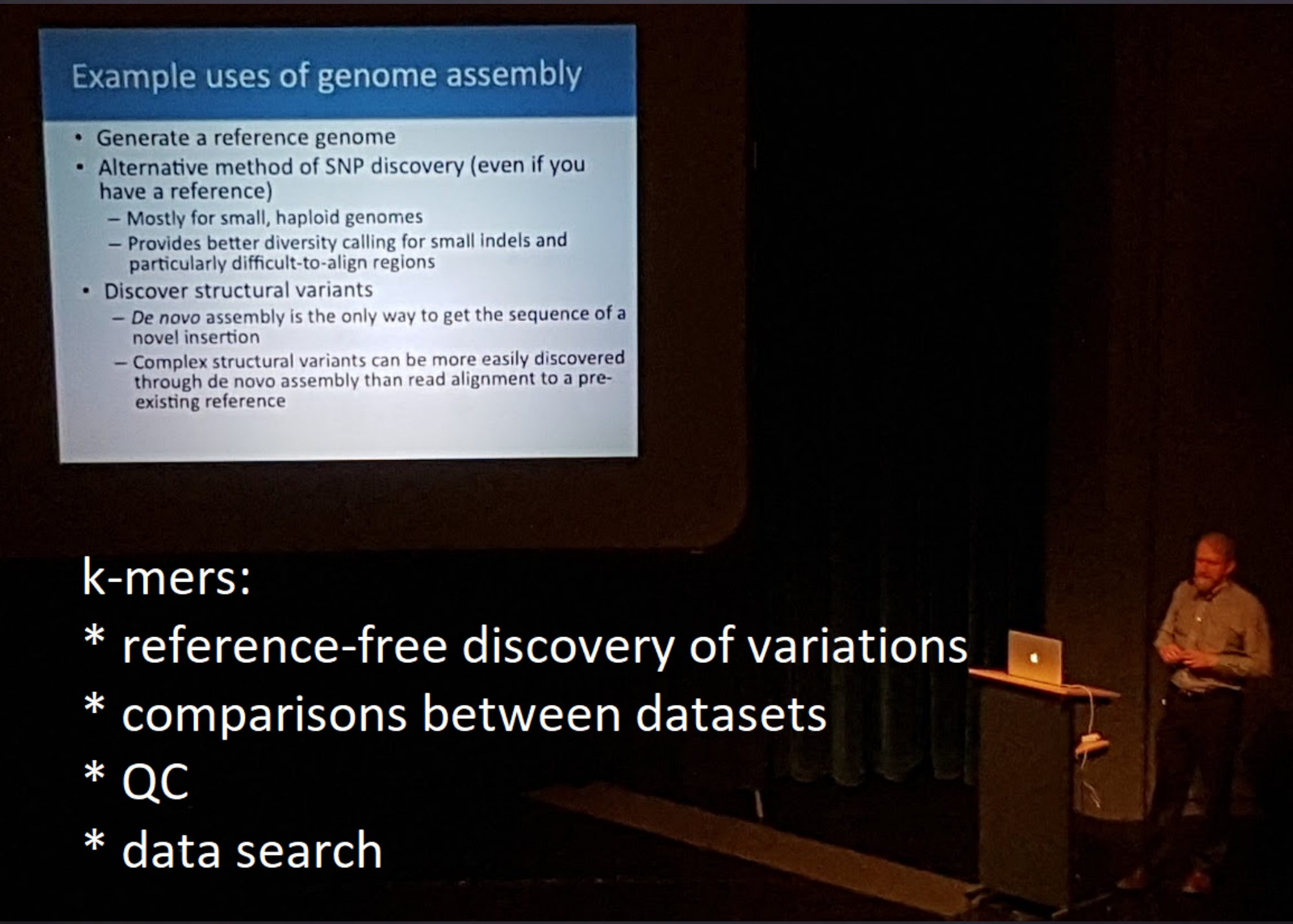
What's assembly?

genome
not known

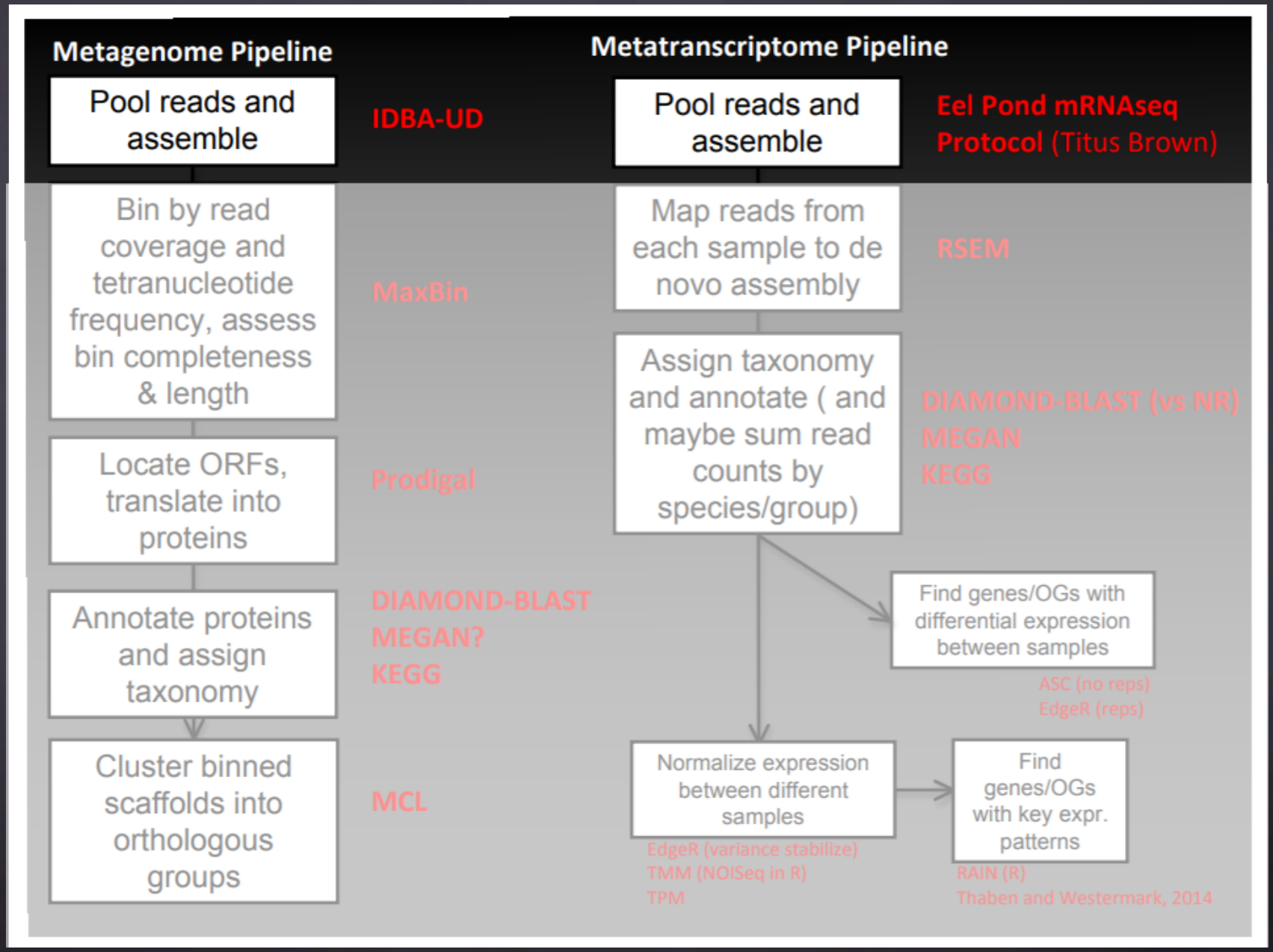
reads
overlapping substrings that cover the genome redundantly

assembly
what we think the genome is

"A set of sequences which best approximate the original sequenced material."


Example uses of genome assembly
• Generate a reference genome
• Alternative method of SNP discovery (even if you have a reference)
– Mostly for small, haploid genomes
– Provides better diversity calling for small indels and particularly difficult-to-align regions
• Discover structural variants
– De novo assembly is the only way to get the sequence of a novel insertion
– Complex structural variants can be more easily discovered through de novo assembly than read alignment to a pre-existing reference
k-mers:
* reference-free discovery of variations
* comparisons between datasets
* QC
* data search

From Sonya's talk



Plan

- Fundamentals
  - Basics
  - Short Exercise
- Some useful assembly theory
  - Graphs
  - Contigs construction
  - Exercise
- RNA-seq and metagenomes
- Visualizing and evaluating assemblies
  - Bandage
  - Reference-free metrics
  - Exercise
- One small other thing you can do with *k*-mers
- Assembly in practice
  - Exercise

Basic expectations

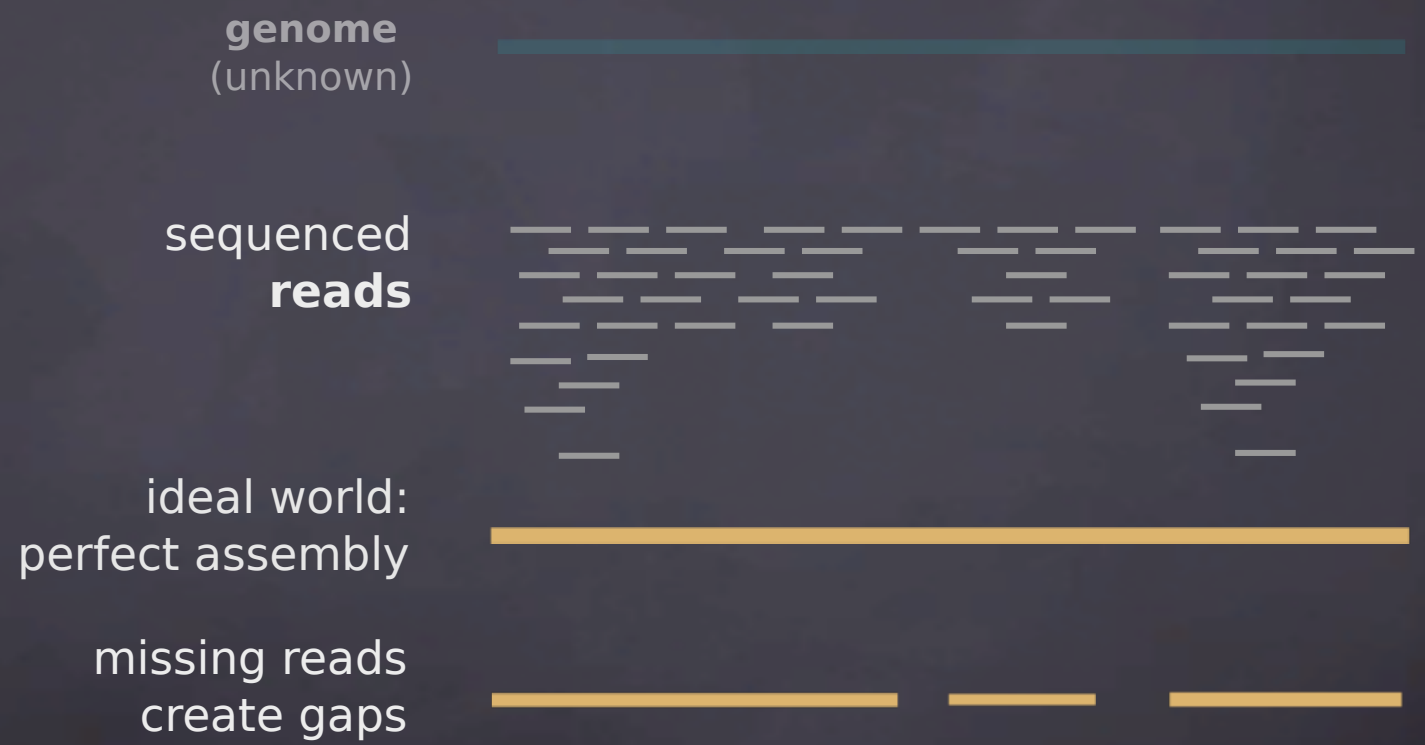
An assembly generally is:

- smaller than the reference,
- fragmented

Basic expectations

An assembly generally is:

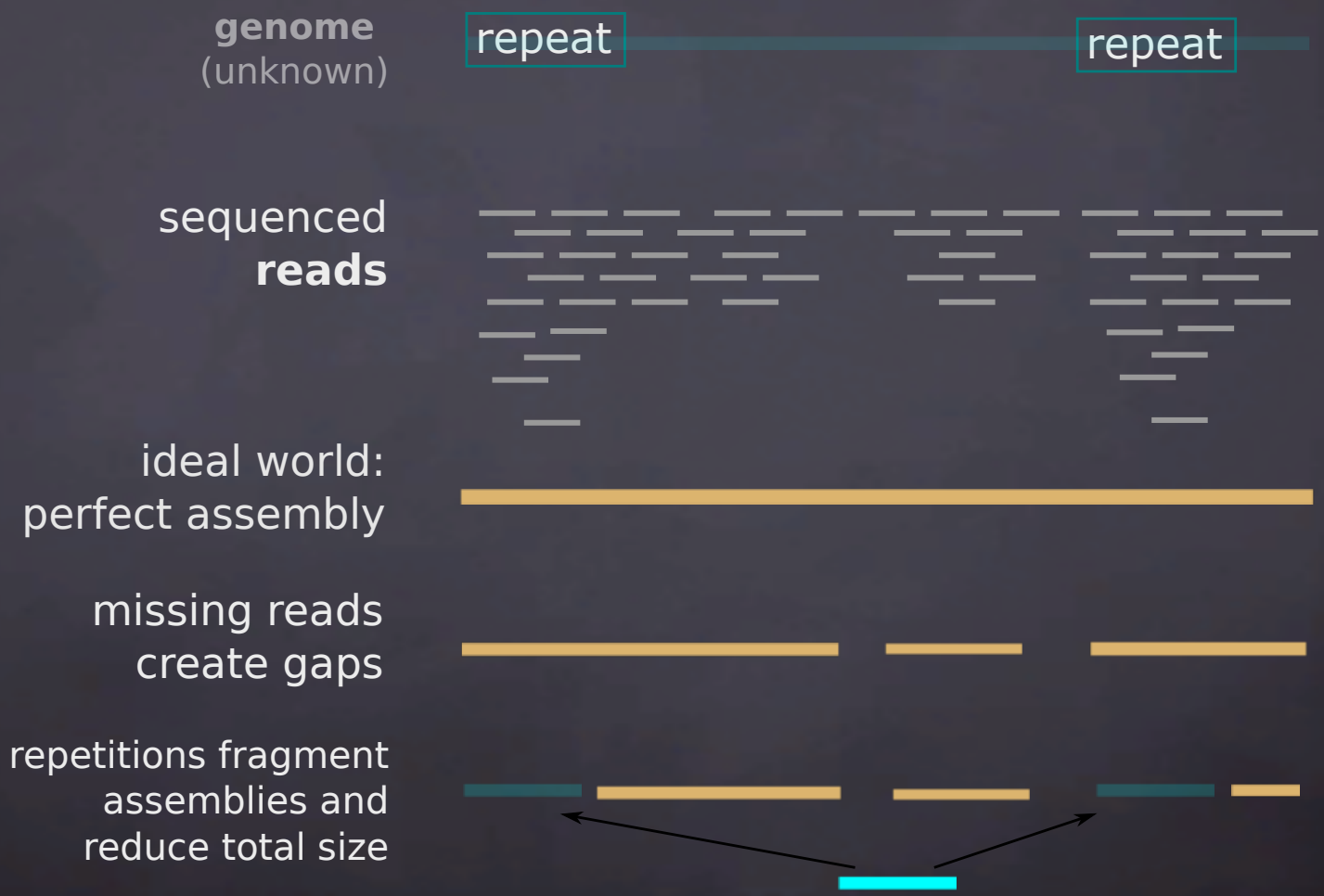
- smaller than the reference,
- fragmented



Basic expectations

An assembly generally is:

- smaller than the reference,
- fragmented



Some vocabulary:

String What computer scientists mean for a sequence or a part of it

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

Contig gap-less assembled sequence

Scaffold sequence which may contain gaps (N)

OLDSCHOOL ASSEMBLY

Won’t be covered here (known as *greedy* algorithms).

Overlaps

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

Exact overlaps:

```
1:  ACTGCT
2:   CTGCT
3:    TGCTAA
```

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.

The fine print

```
 CGTT
ACGT
```

We don't consider prefix->suffix overlaps, only suffix->prefix.

```
TACGTG
 GCGTCA
```

This isn't an exact overlap, even if `CGT` is common.

```
TACT
 ACTG
```

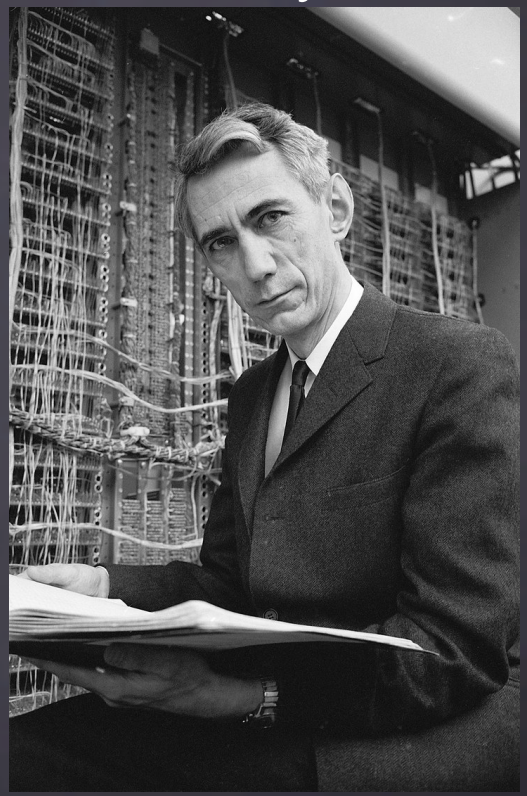
`AC` is not considered an overlap, only `ACT` is.

k-MERS

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

N.G. de Bruijn (1946), de Bruijn sequences [1]

C. Shannon (1948), information theory [2]



[1] construct a short text that contains all *k*-mers exactly once

[2] improve communication by predicting what character is likely to follow a given *k*-mer

k-MER HISTOGRAMS

- x axis: *abundance*
- y axis: number of k-mers having abundance x (i.e. seen x times)

Example reads dataset:

```
ACTCA
GTCA
```

3-mers:

```
ACT
CTC
TCA
GTC
TCA
```

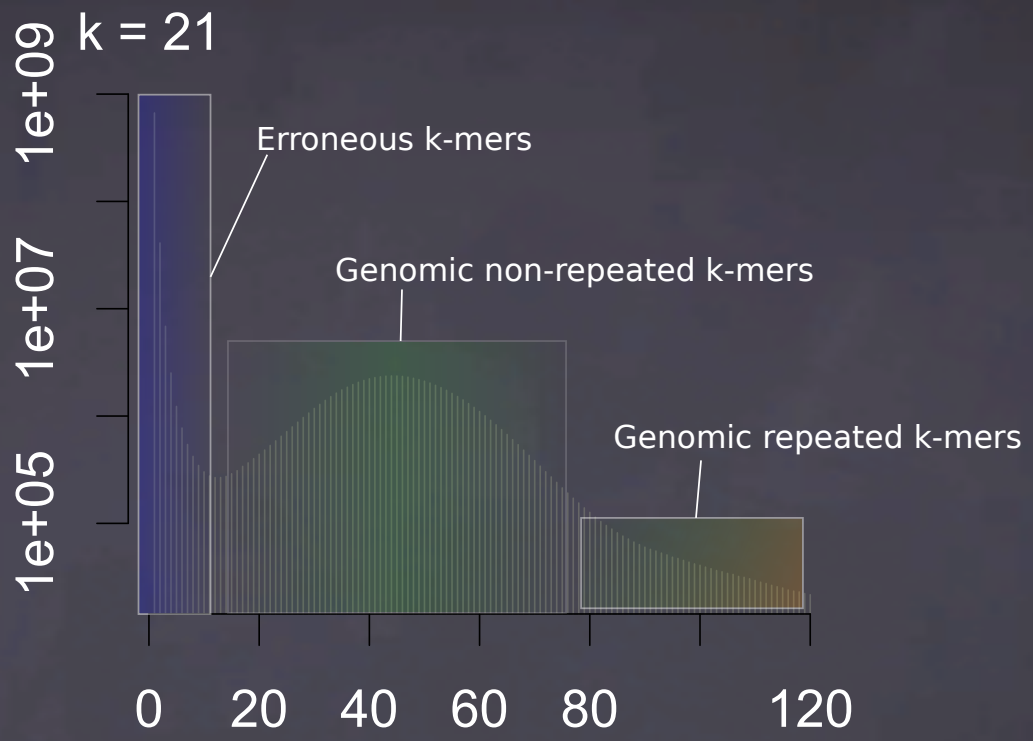
Abundance of distinct 3-mers:

```
ACT: 1
CTC: 1
TCA: 2
GTC: 1
```

3-mer histogram:

x	y
1	3
2	1
3	0
4	0

Tools: DSK, Jellyfish, KMC, KmerGenie



Genomic area

$\approx$

number of distinct k-mers covering the genome

$\approx$

size of the assembly

Chr 14 ($\approx$ 88 Mbp) GAGE dataset; histogram $k = 21$

The choice of k

Choice of k is critical:

- k-mers with sequencing errors are noise
- only k-mers devoid of sequencing errors matter
- $k = 1$: GC-content
- $k < \log_4(|\text{genome}|)$: uninformative graph (all possible nodes/edges present)
- This is related to the "mappability" of short reads.

Small k-mers: repeated across genome, often non-erroneous
Long k-mers: more specific, more likely to contain a seq error

Generally, $k \geq 20$. (Compare 4^k to the genome size.)
Higher sequencing coverage $\rightarrow$ use longer k's.

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

Plan

- Fundamentals
  - Basics
  - Short Exercise
- Some useful assembly theory
  - Graphs
  - Contigs construction
  - Exercise
- RNA-seq and metagenomes
- Visualizing and evaluating assemblies
  - Bandage
  - Reference-free metrics
  - Exercise
- One small other thing you can do with *k*-mers
- Assembly in practice
  - Exercise

SOME ASSEMBLY THEORY

Graphs

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

Graphs for sequencing data

Overlaps between reads: fundamental information to assemble.
Graphs represent overlaps.

Two different types of graphs for sequencing data:

- de Bruijn (DB) graphs — for Illumina, 10X data
- string graphs — for PacBio/Nanopore data

Knowledge of these graphs, useful for:

- **reference-free** analysis
- setting software **parameters**
- type of regions **well or badly** assembled
- how **repetitions** are handled
- how **heterozygosity** is handled

Overlap graphs

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

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

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

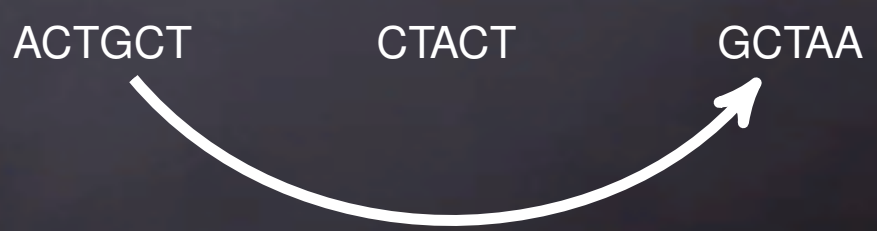
A **string graph** is obtained from an overlap graph by removing redundancy:

- 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 (DB) graphs

*This is going to be fundamental for **Illumina** & 10X data.*

A **DB** graph for a fixed integer k:

1. **Nodes** = all *k-mers* 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

DB GRAPHS

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

```
ACTG
 CTGC
  TGCC
```

DB GRAPHS: REDUNDANCY

What happens if we add redundancy?

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

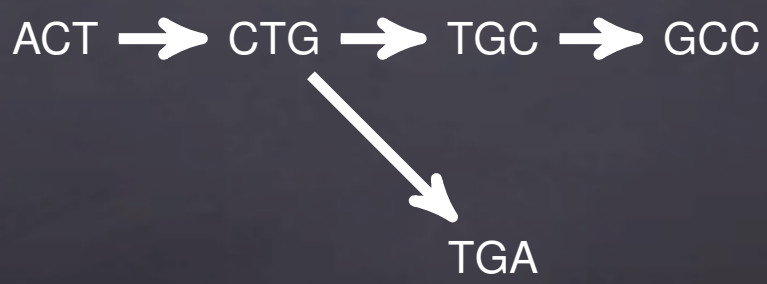
dBG, $k = 3$:

DB GRAPHS: ERRORS

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

```
ACTG
 CTGC
 CTGA
  TGCC
```

dBG, $k = 3$:

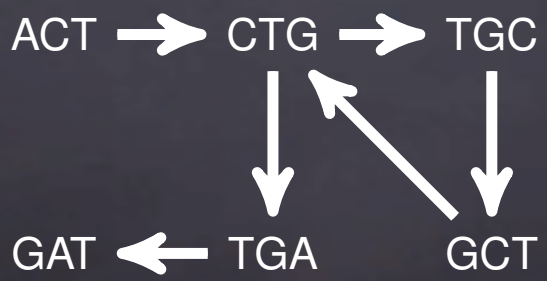


DB GRAPHS: REPEATS

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

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

dBG, $k = 3$:

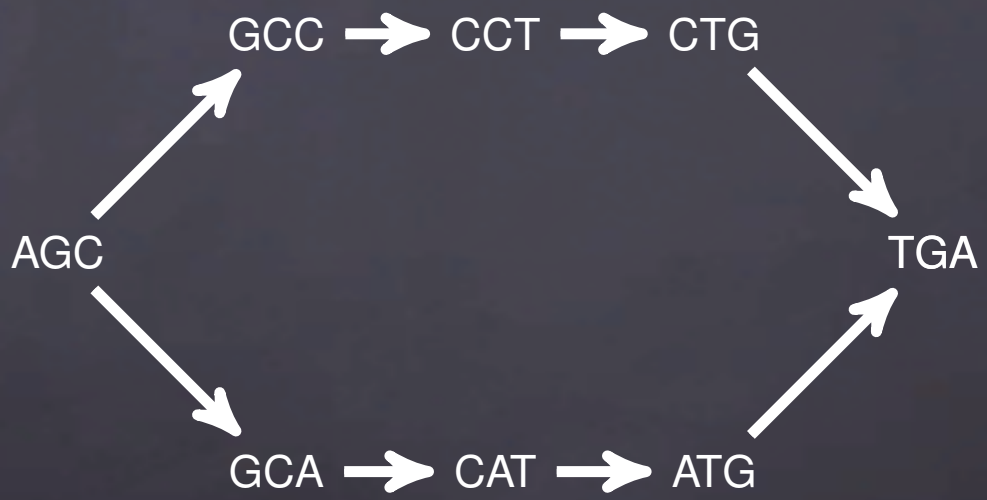


DB GRAPHS: SNPS

SNPs can be directly “found” in the graph.

```
AGCCTGA
AGCATGA
```

dBG, $k = 3$:



DiscoSNP: finds SNPs without a reference. [Uricaru et al, 2015]

Comparison string graph / DB graph

On the same example, compare the DB graph with the string graph:

```
AGTGCT
 GTGCTA
   GCTAA
```

String graph with exact overlaps ≥ 3:

DB graph, $k = 3$:

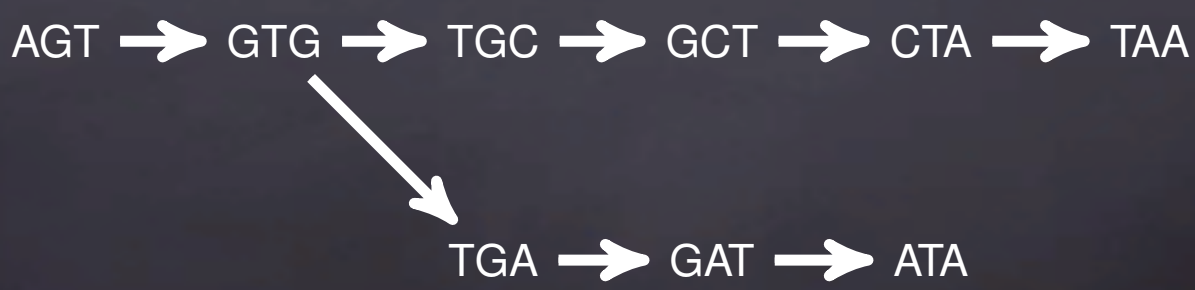
String graph / de Bruijn graph (2)

Let's add an error:

```
AGTGCT
 GTGATA
   GCTAA
```

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

DB graph, $k = 3$:



String graph / DB graph (4)

So, which is better?

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

Historically, string graphs were used for long reads and DB 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

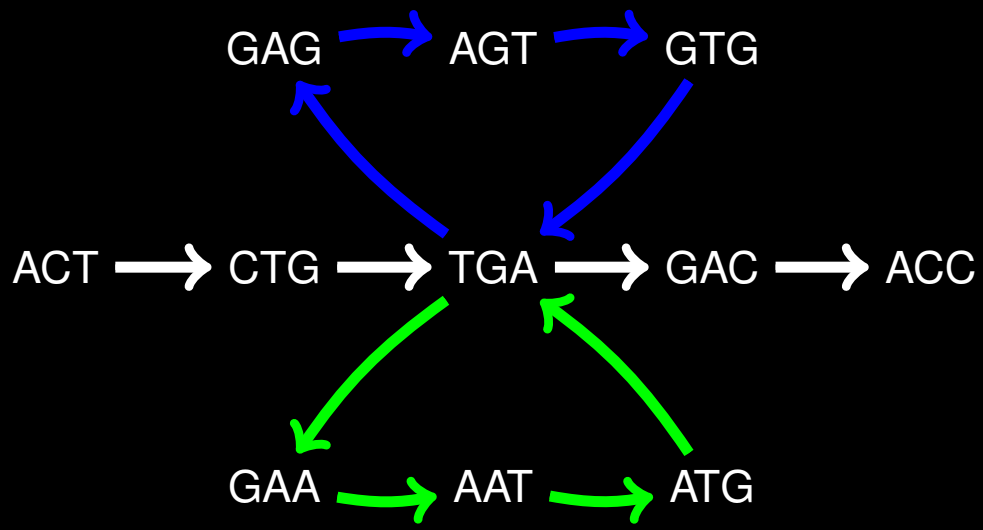
[Nagarajan 09]

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

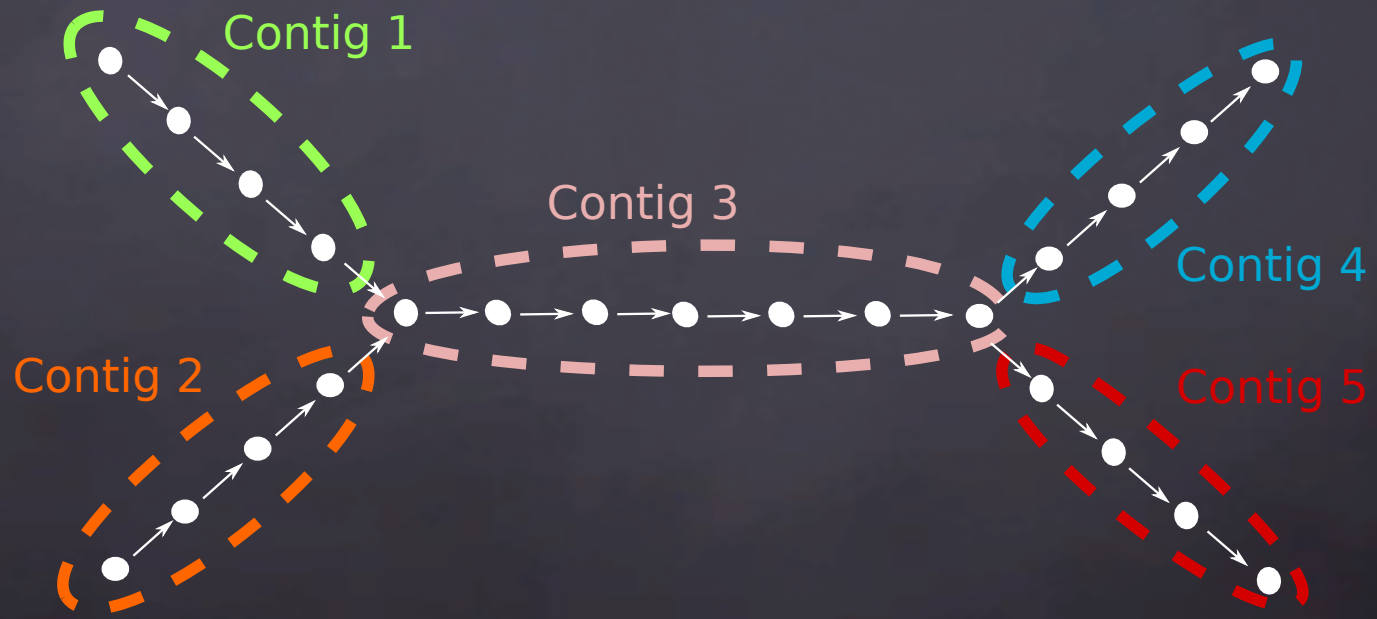
An assembly is the following set of paths:

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

CONTIGS CONSTRUCTION

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

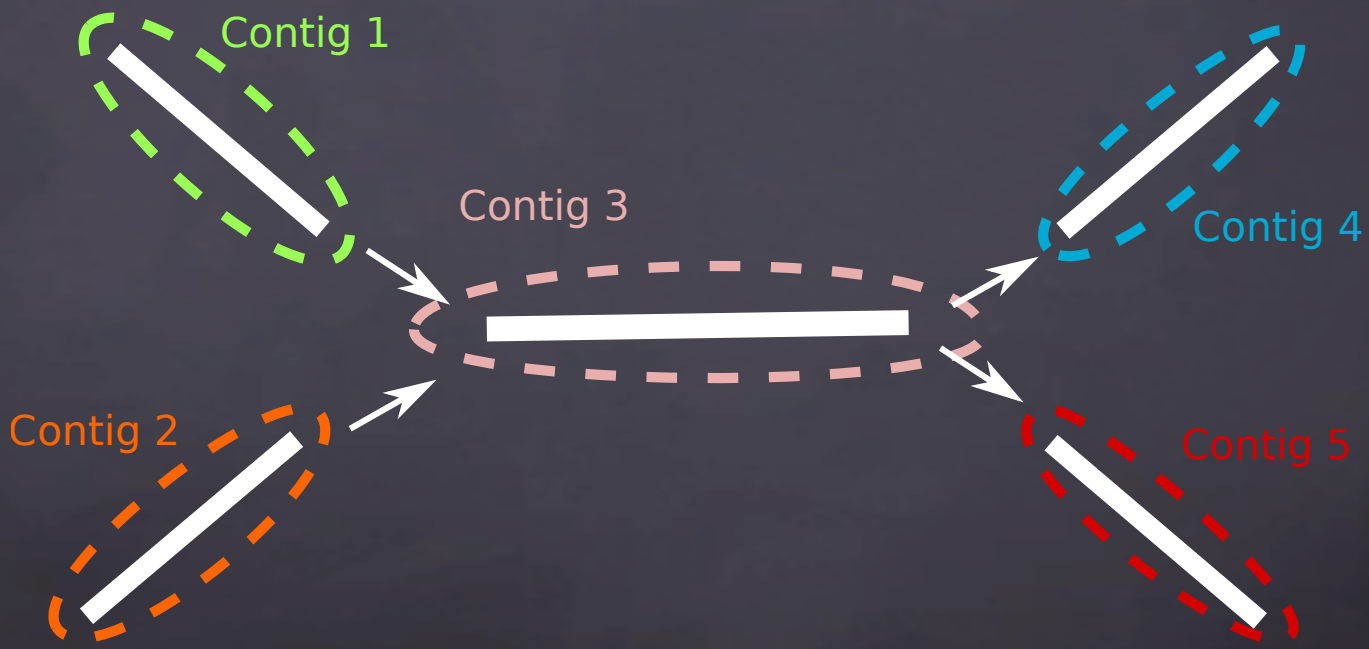
simple path: all internal nodes have a single in/out edge.
node-disjoint: two different paths cannot share a node.



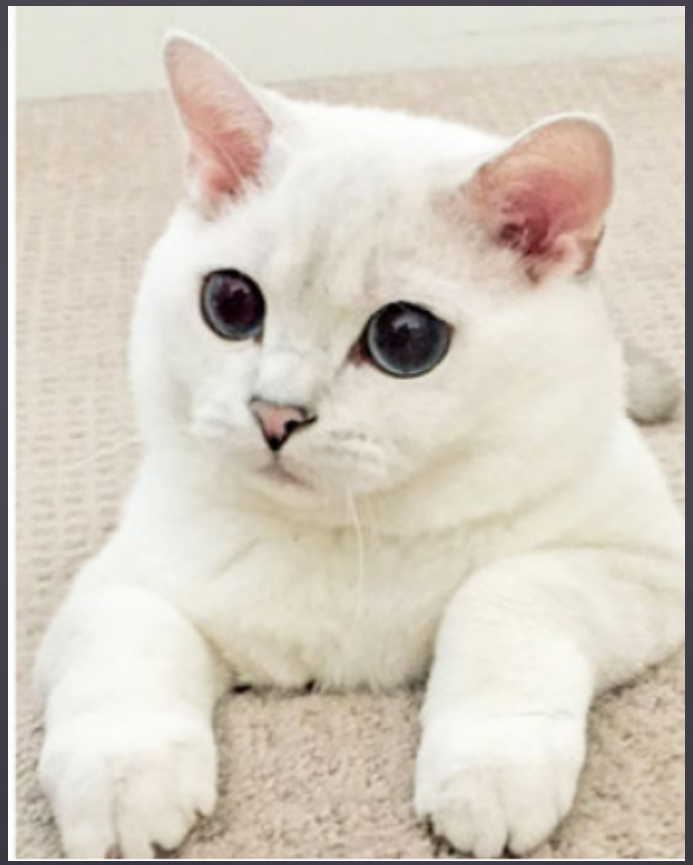
CONTIGS GRAPH

The result of an assembly is a **contig graph**:

- nodes = contigs
- edges = overlaps between contigs



Coffee break?

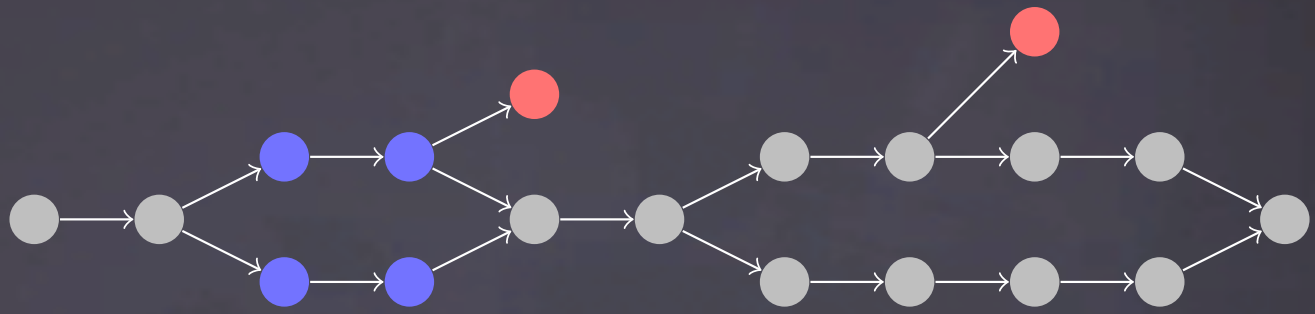


Coffee break

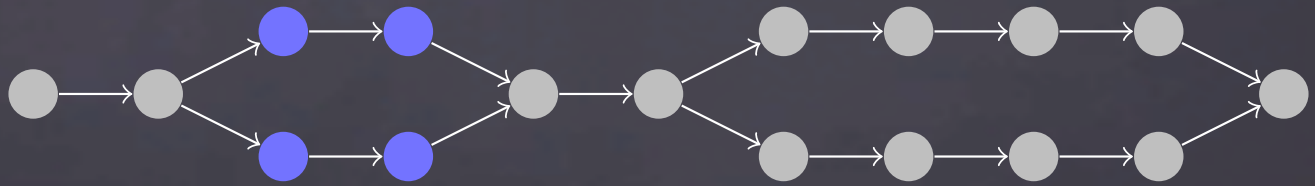
How Illumina assemblers work

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

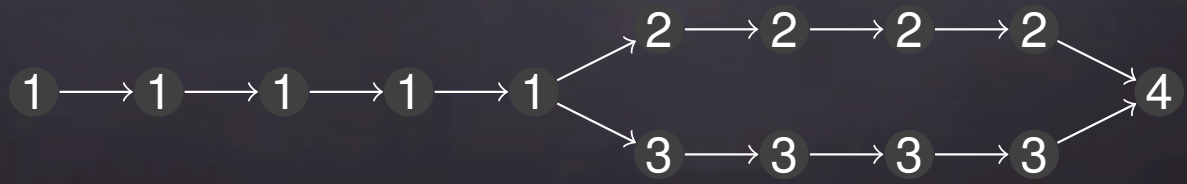
1) Maybe correct the reads. (SPAdes, HGAP, SGA)
2) Construct a graph from the reads. It will contain variants & errors.



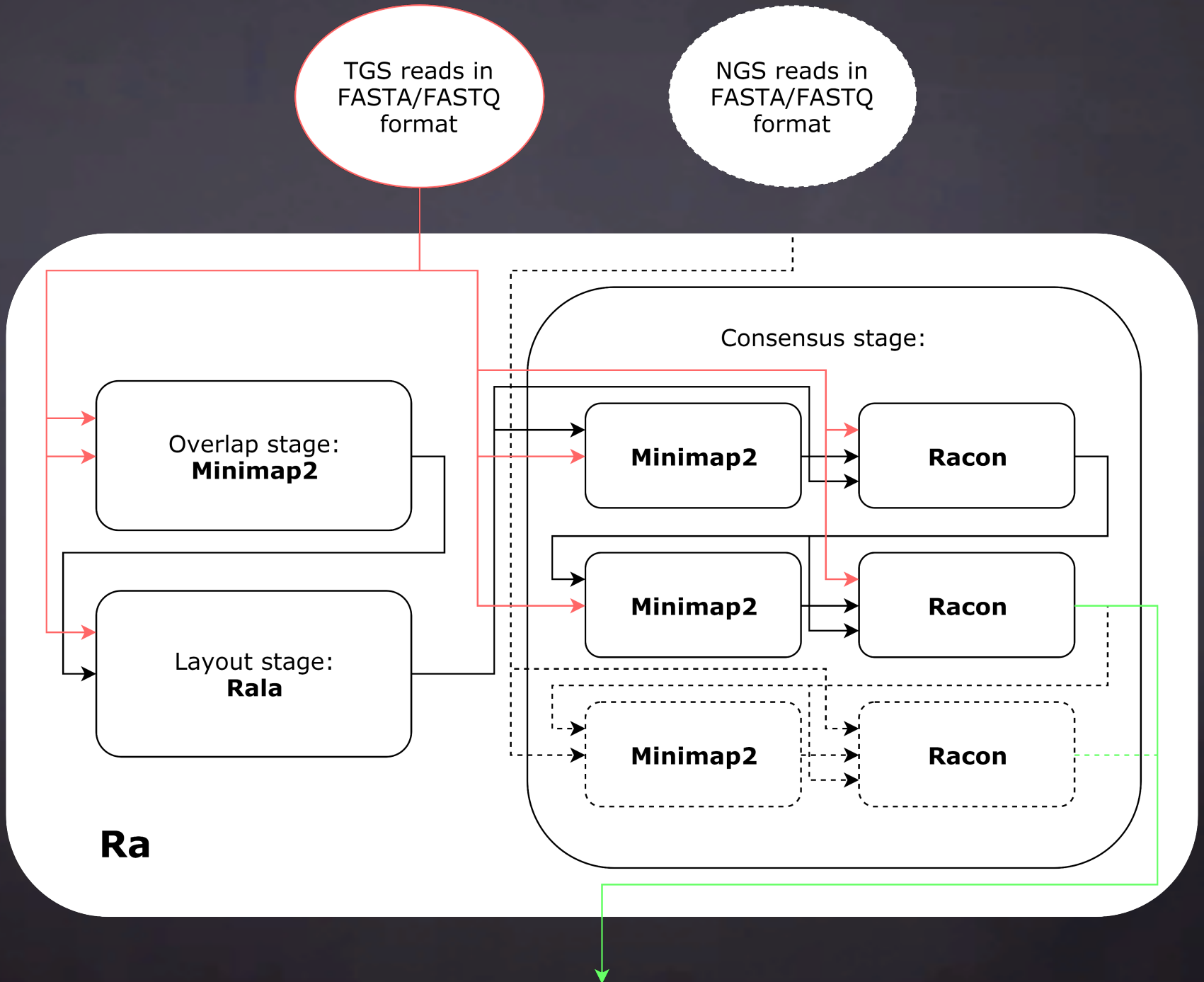
3) Likely sequencing errors are removed.



3) Known biological events are removed.
4) Finally, **simple paths** (i.e. contigs) are returned.



LET'S DISSECT A PACBIO ASSEMBLER



Short note on reverse complements

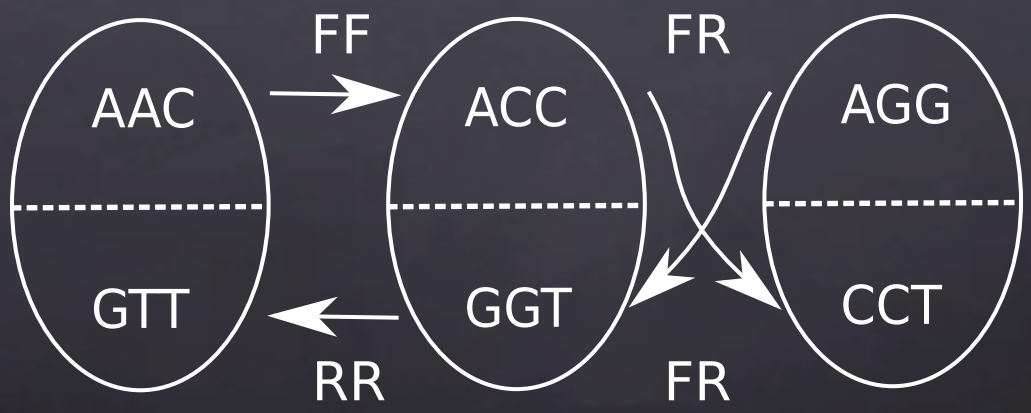
Due to strand ambiguity in sequencing:

> *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
```

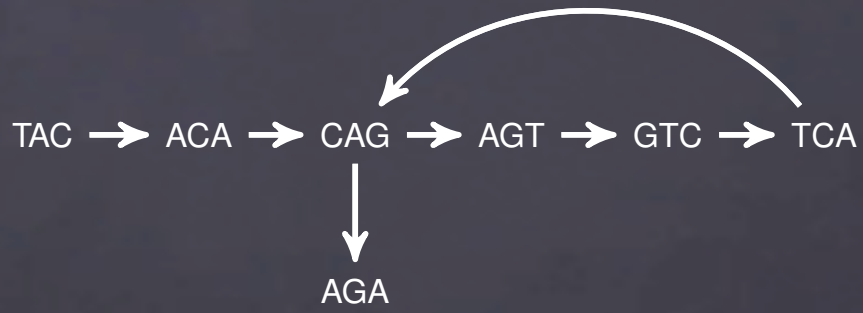
1. Construct the DB graph for $k = 3$.
   (Reminder: nodes are k-mers and edges correspond to $(k-1)$-overlaps)
2. How many contigs can be created?
3. At which value of k is there a single contig?
4. (optional) Find a mathematical relationship between k_a, the smallest k value with which a genome can 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 DB 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

- Fundamentals
  - Basics
  - Short Exercise
- Some useful assembly theory
  - Graphs
  - Contigs construction
  - Exercise
- RNA-seq and metagenomes
- Visualizing and evaluating assemblies
  - Bandage
  - Reference-free metrics
  - Exercise
- One small other thing you can do with *k*-mers
- Assembly in practice
  - Exercise

RNA-SEQ ANALYSIS

Haas & Zody, Nat Biotech 2010

RNA-Seq and metagenome assembly

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

RNA-Seq and metagenome assembly

Despite these differences, DNA-seq assembly methods apply:

- Construct a de Bruijn graph (same as DNA)
- Remove errors and variants (somewhat similar)
- Output contigs (same as DNA)
- Allow to re-use the same contig in many different assembled transcripts or metagenome scaffolds (new part)

RNA-Seq Assembly: Trinity

Quick overview of Trinity steps:

- Inchworm
- Chrysalis
- Butterfly

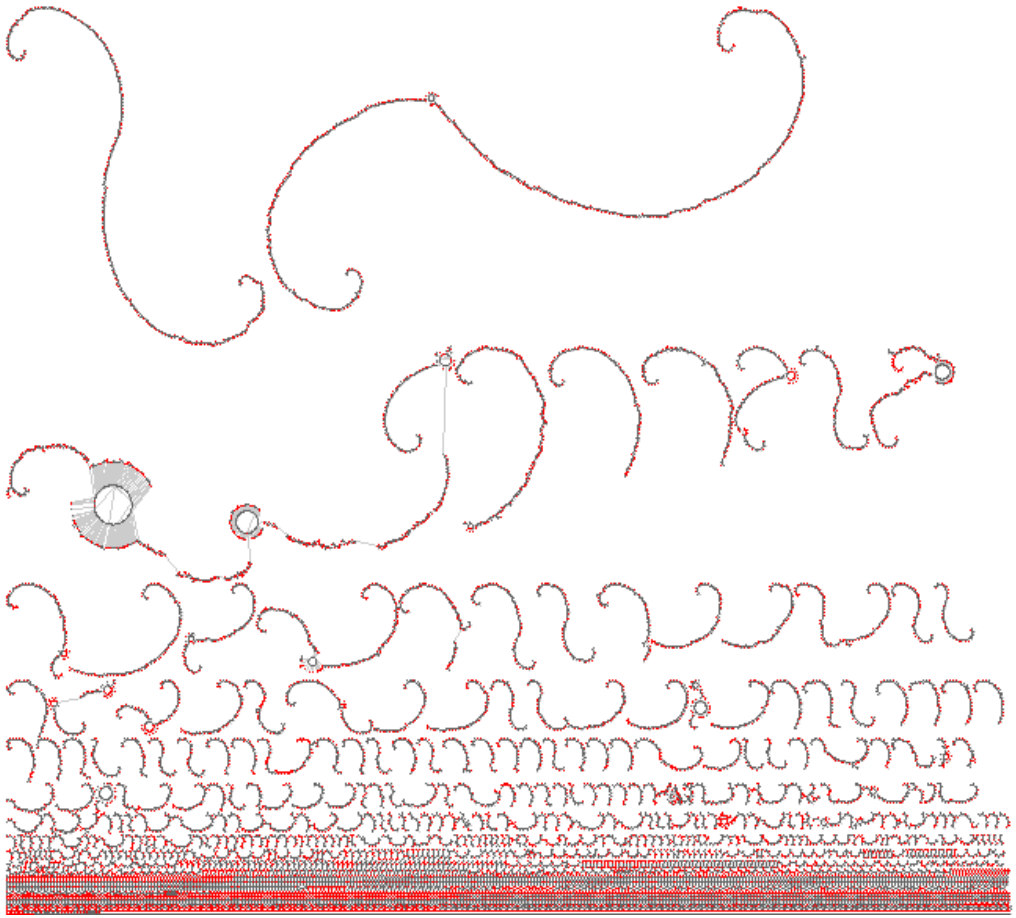
RNA-Seq Assembly: Trinity

- ~~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

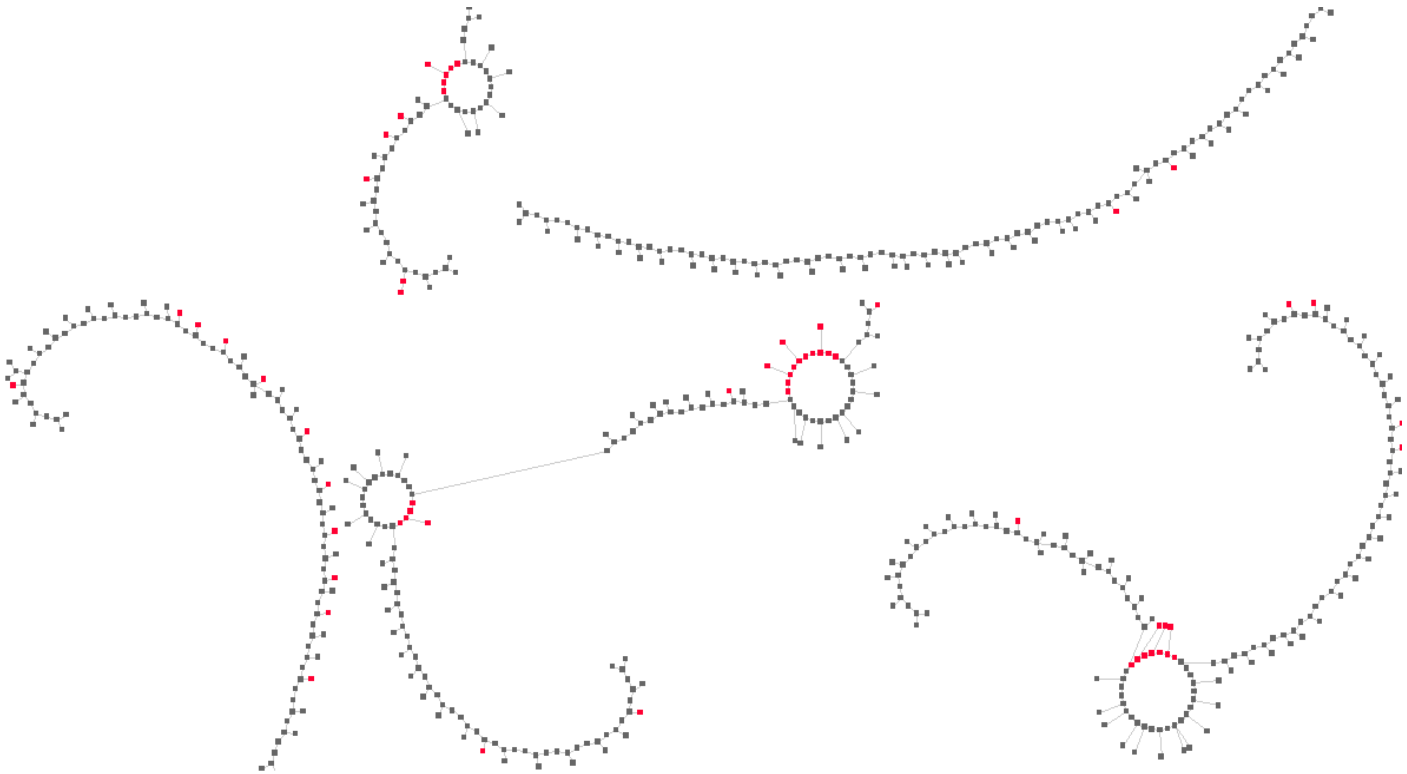
Haas & Zody, Nat Biotech 2010

Example of DBG built from RNA-seq data

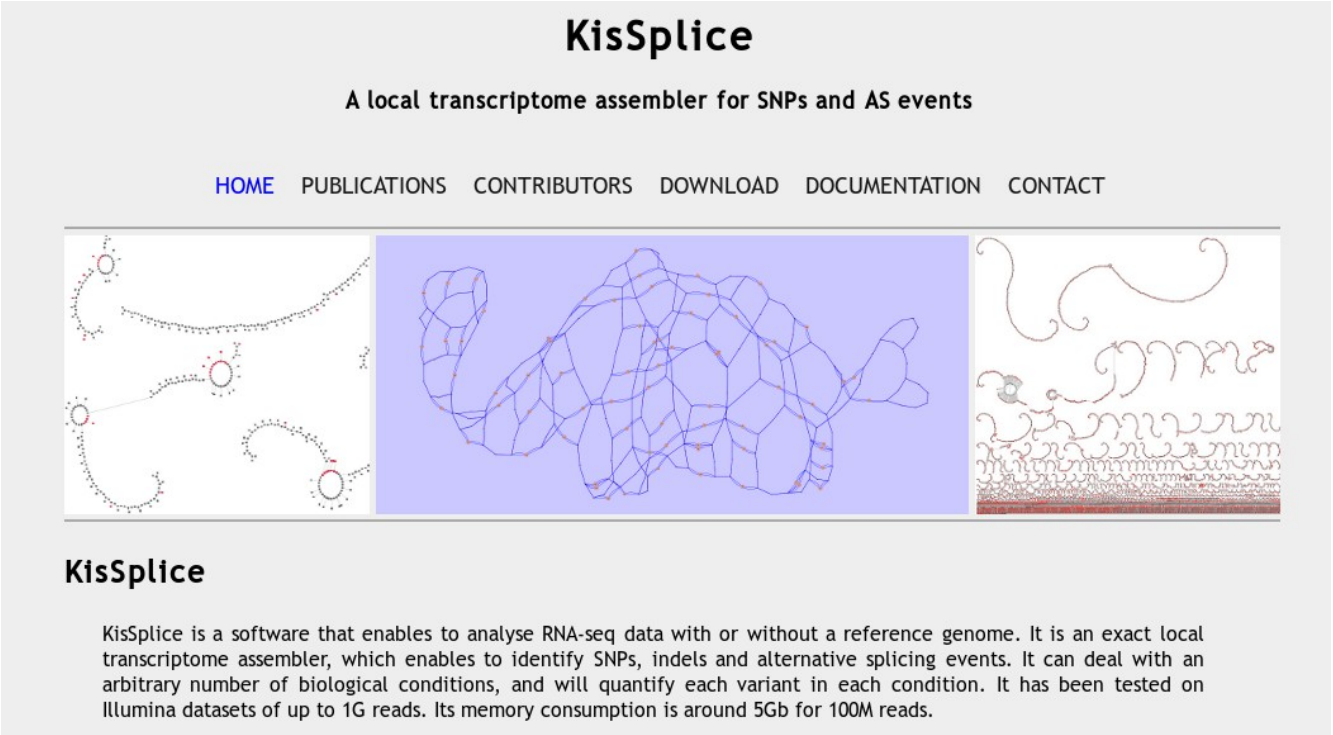


Slides adapted from V. Lacroix / C. Benoit-Pilven

Variants in RNA-seq data



Software



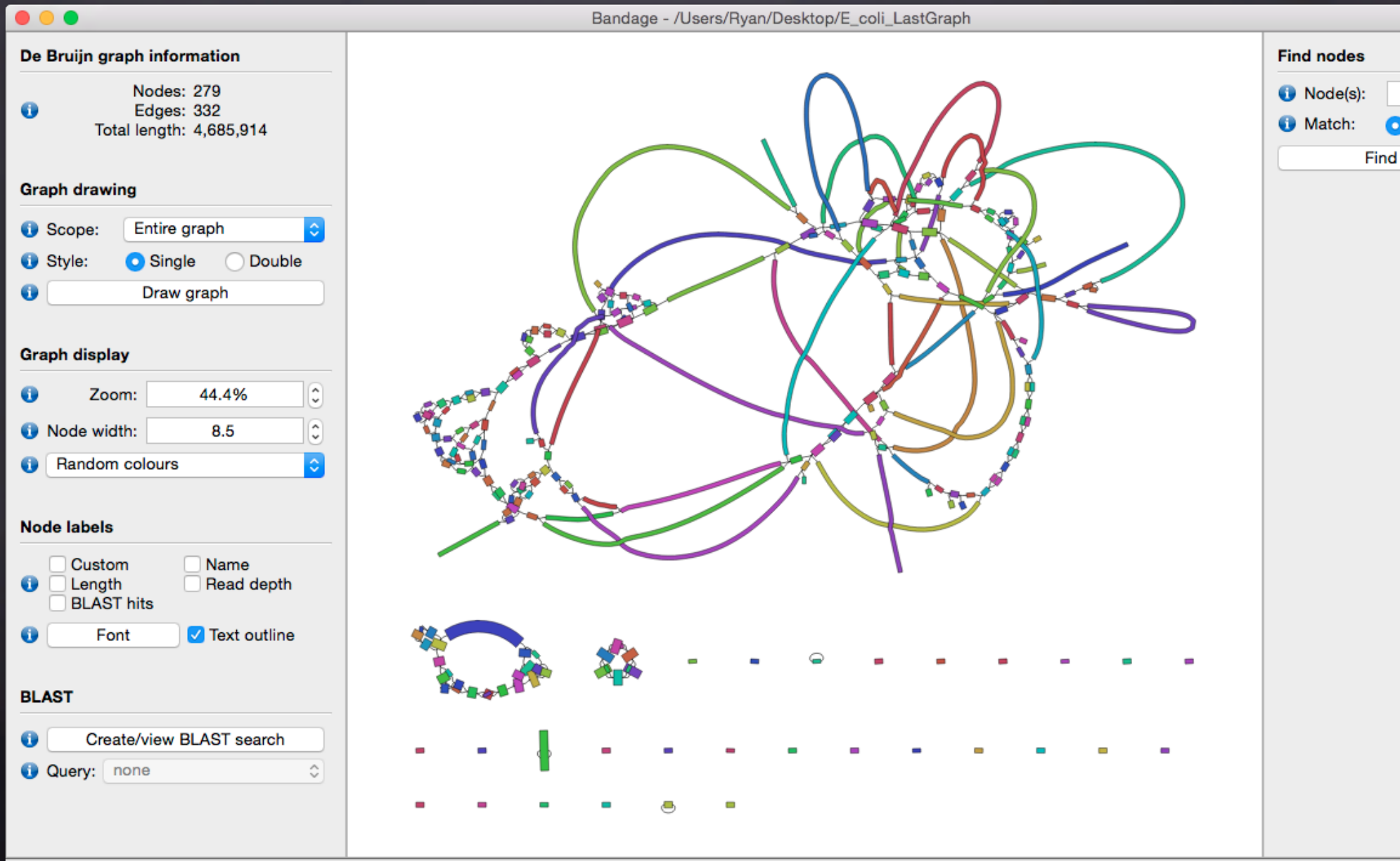
http://kissplice.prabi.fr

Sacomoto et al, Recomb-Seq 2012, Lopez-Maestre et al, NAR 2016

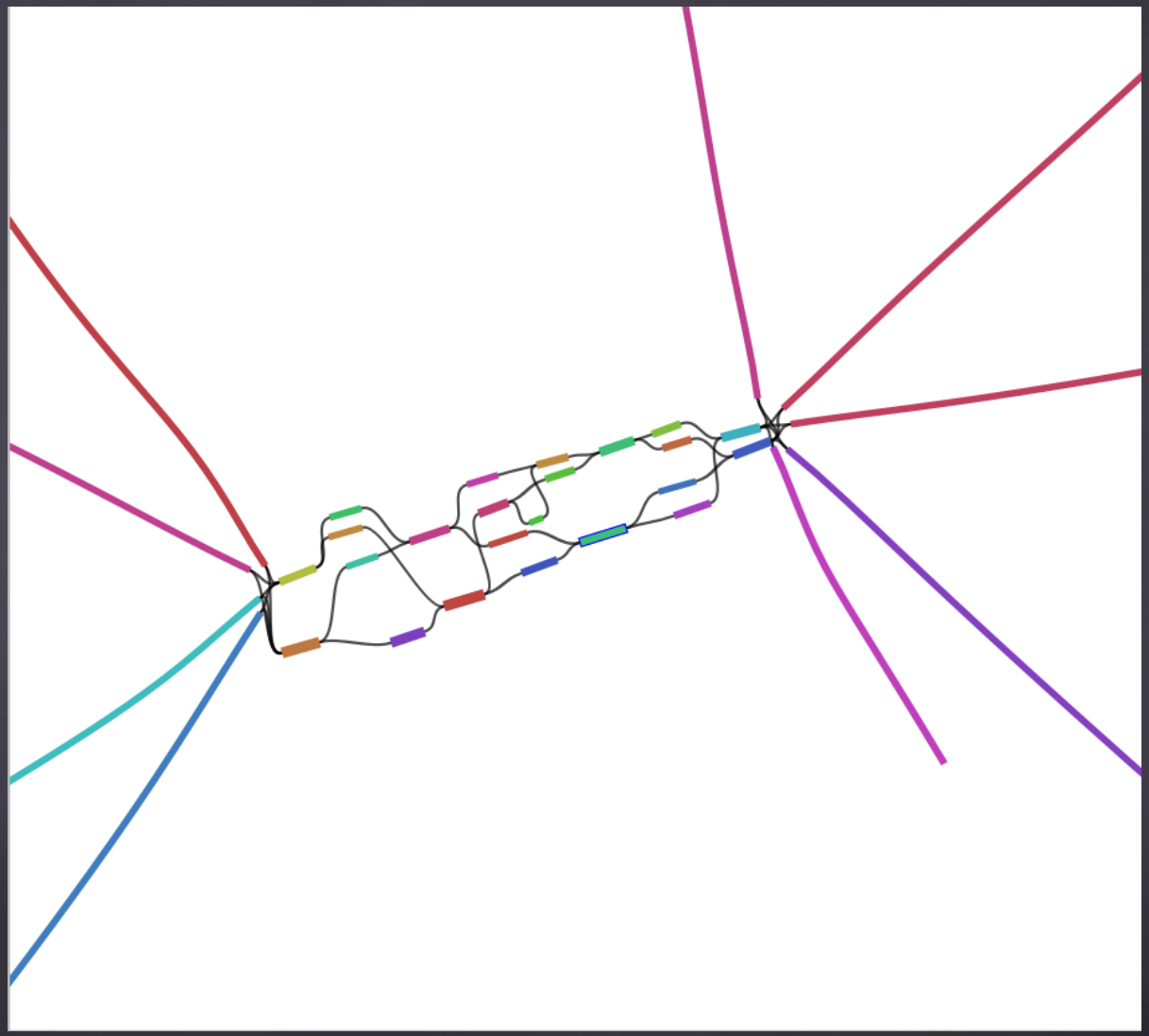
Plan

- Fundamentals
  - Basics
  - Short Exercise
- Some useful assembly theory
  - Graphs
  - Contigs construction
  - Exercise
- RNA-seq and metagenomes
- Visualizing and evaluating assemblies
  - Bandage
  - Reference-free metrics
  - Exercise
- One small other thing you can do with *k*-mers
- Assembly in practice
  - Exercise

Assembly graph visualization: Bandage

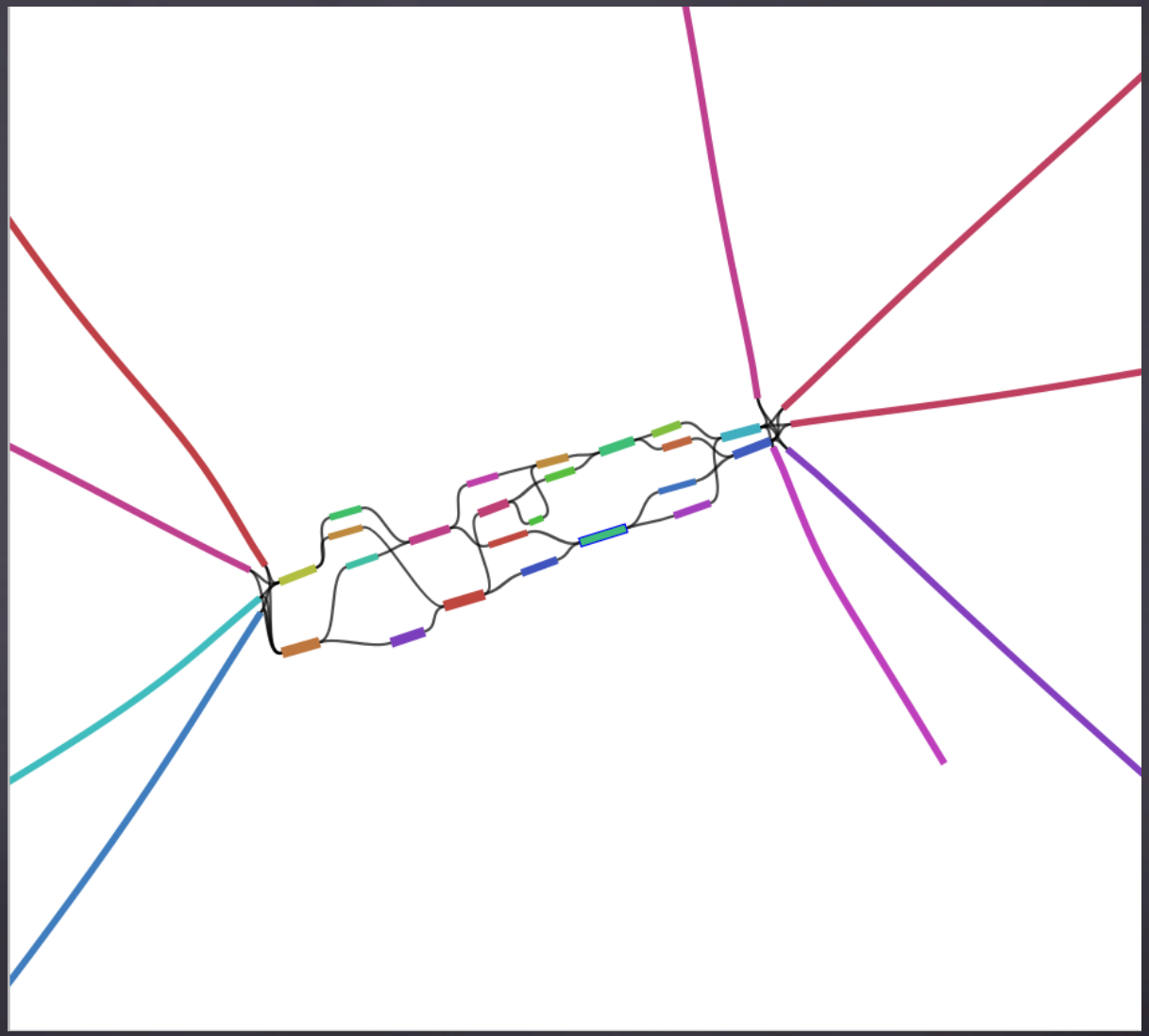


BANDAGE



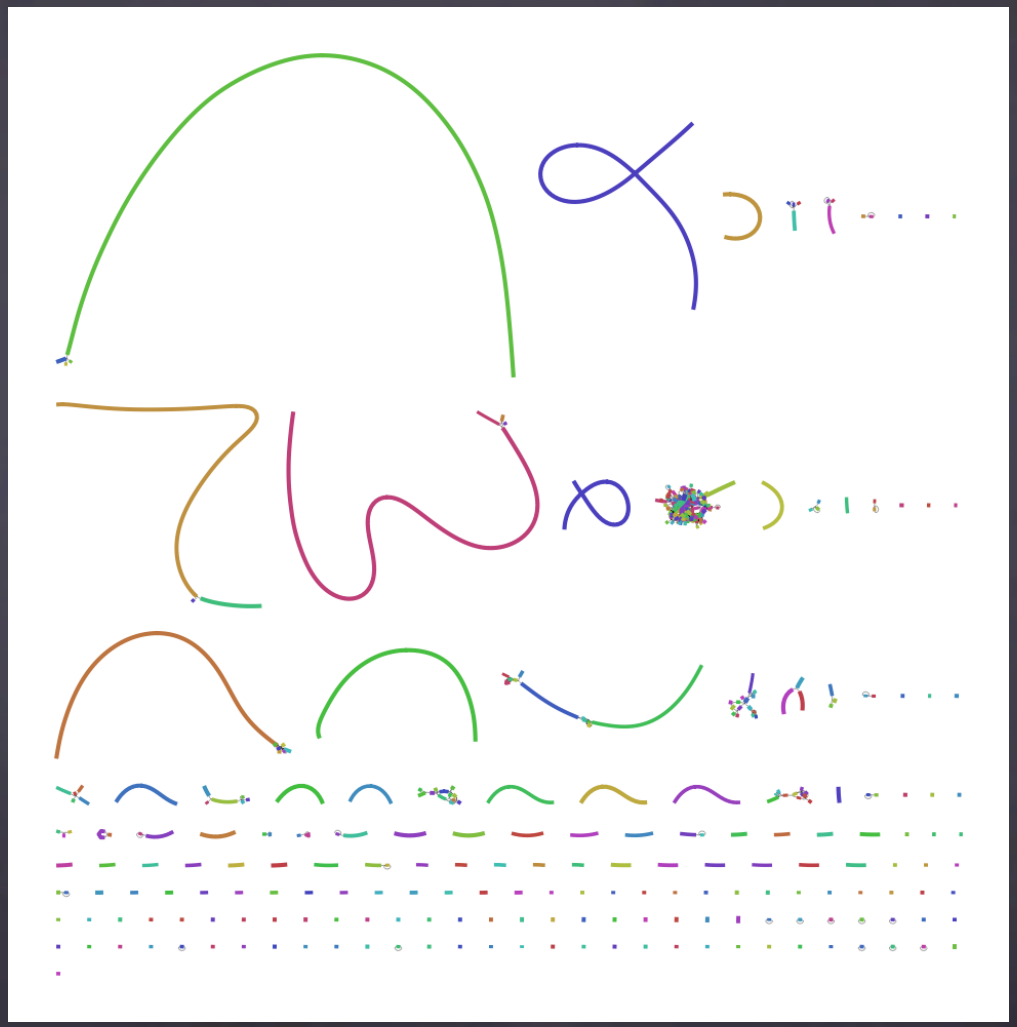
E. coli SPAdes assembly (excerpt). Fig from Lex Nederbragt. What is this knot?

BANDAGE



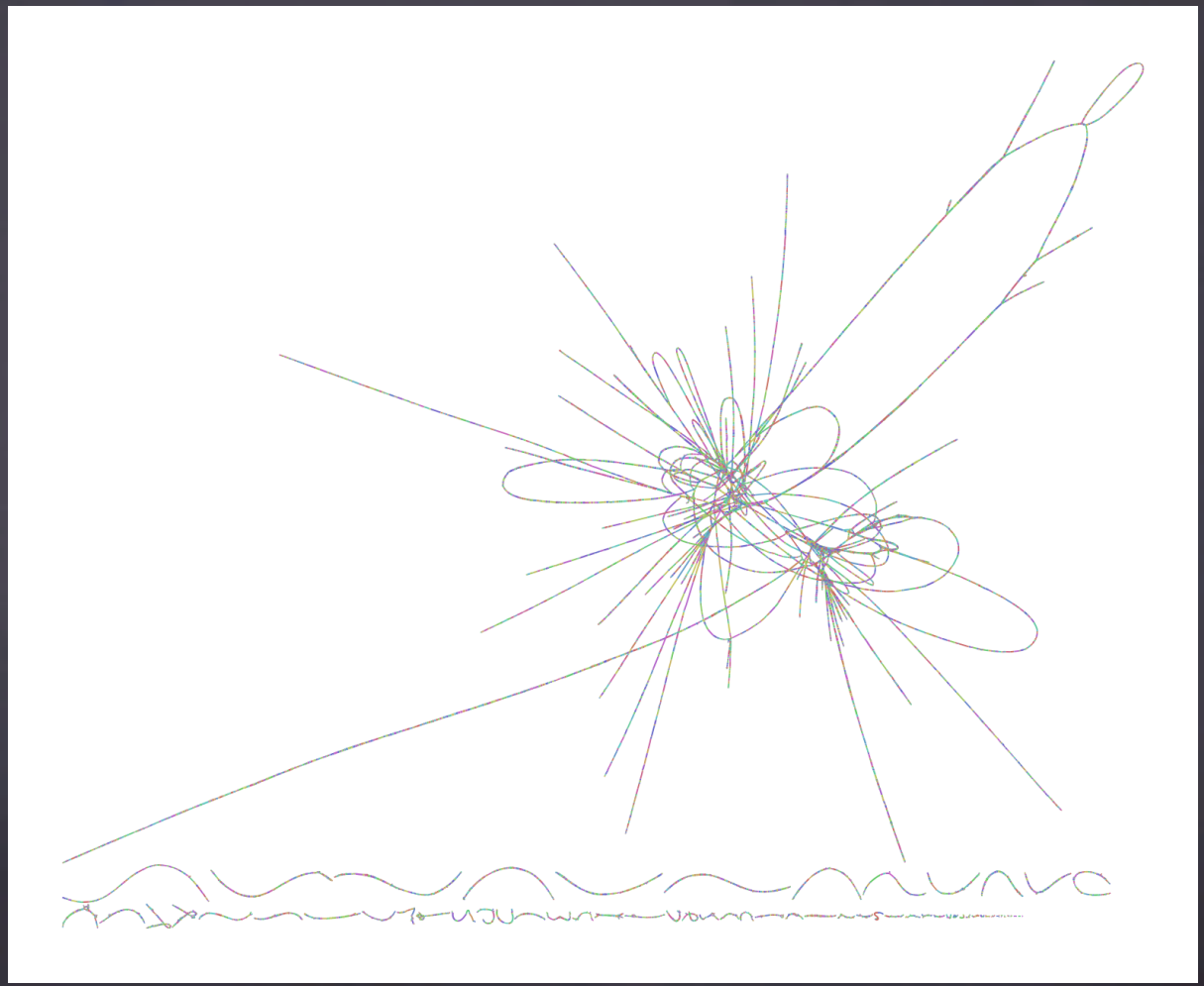
E. coli SPAdes assembly (excerpt). Fig from Lex Nederbragt. What is this knot?
collapsed ribosomal genes (16S, 2S, ..)

PacBio assembly visualization



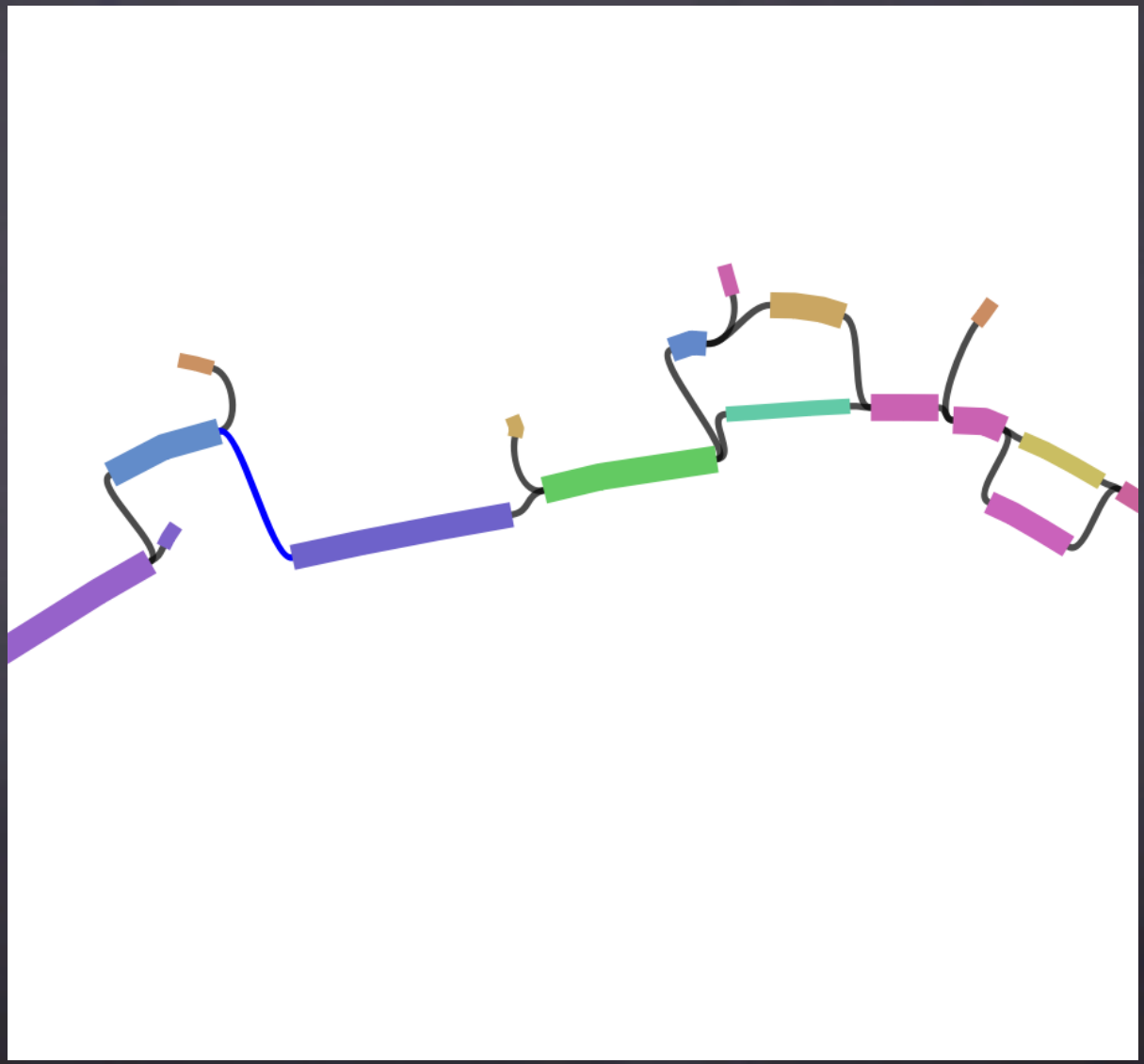
D. melanogaster FALCON assembly. Each node is a contig. (fig. @md5sam)

Effect of simplifications on the graph (Illumina data)



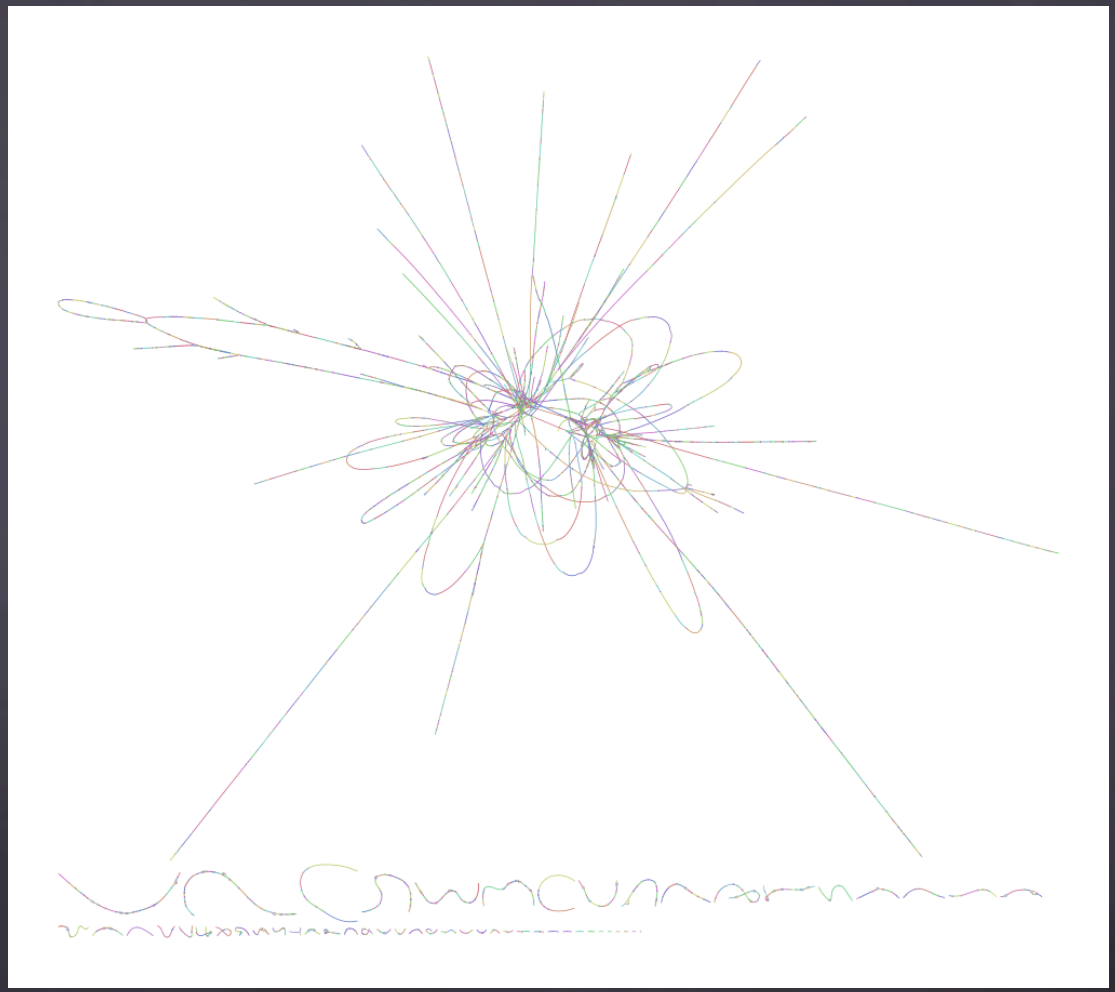
human chr14:20Mbp-20.5Mbp GAGE PE reads, Minia k=31, no graph simplifications at all, around 20k nodes

Effect of simplifications on the graph (Illumina data)



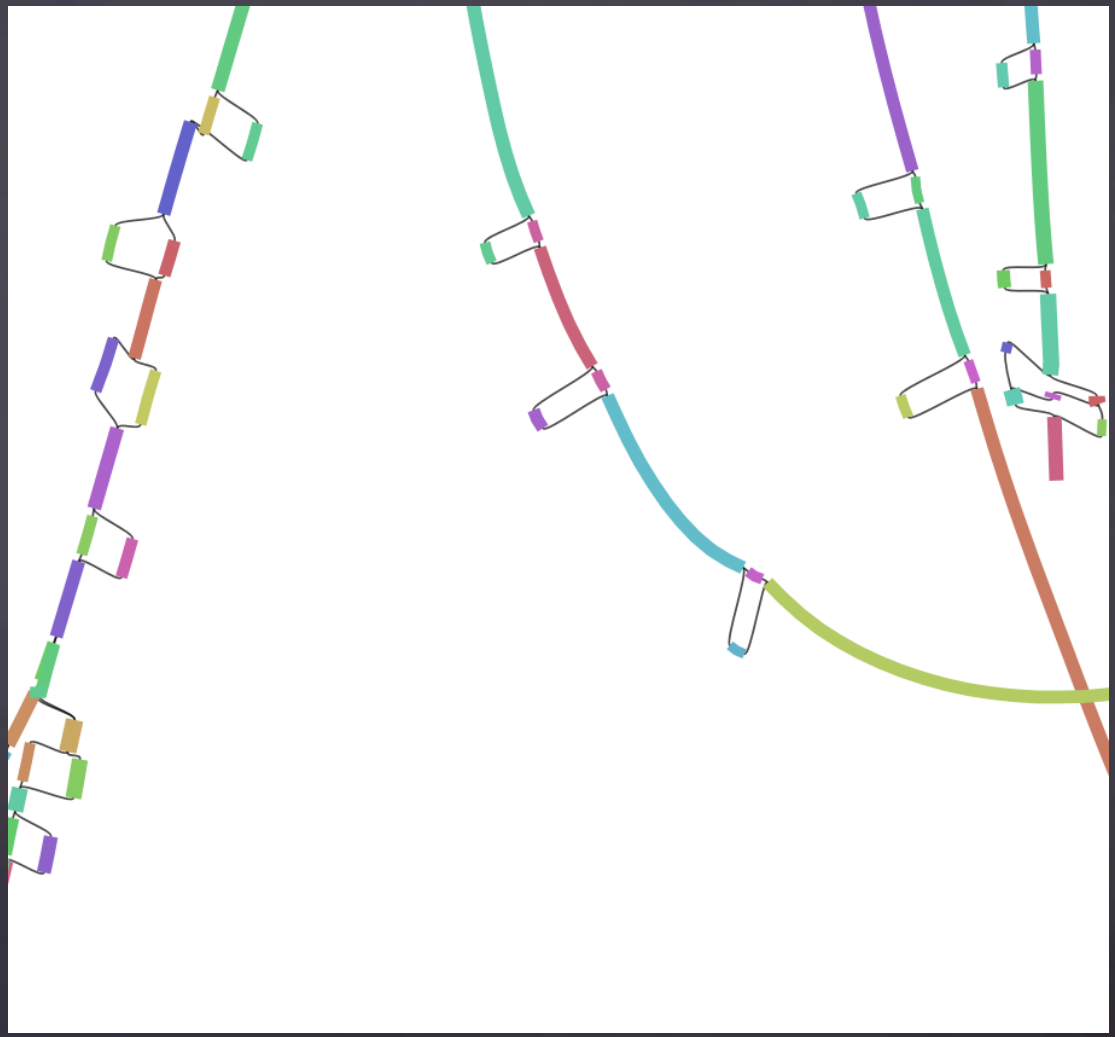
same as previous slide, zoomed in to see tips and bubble

Effect of simplifications on the graph (Illumina data)



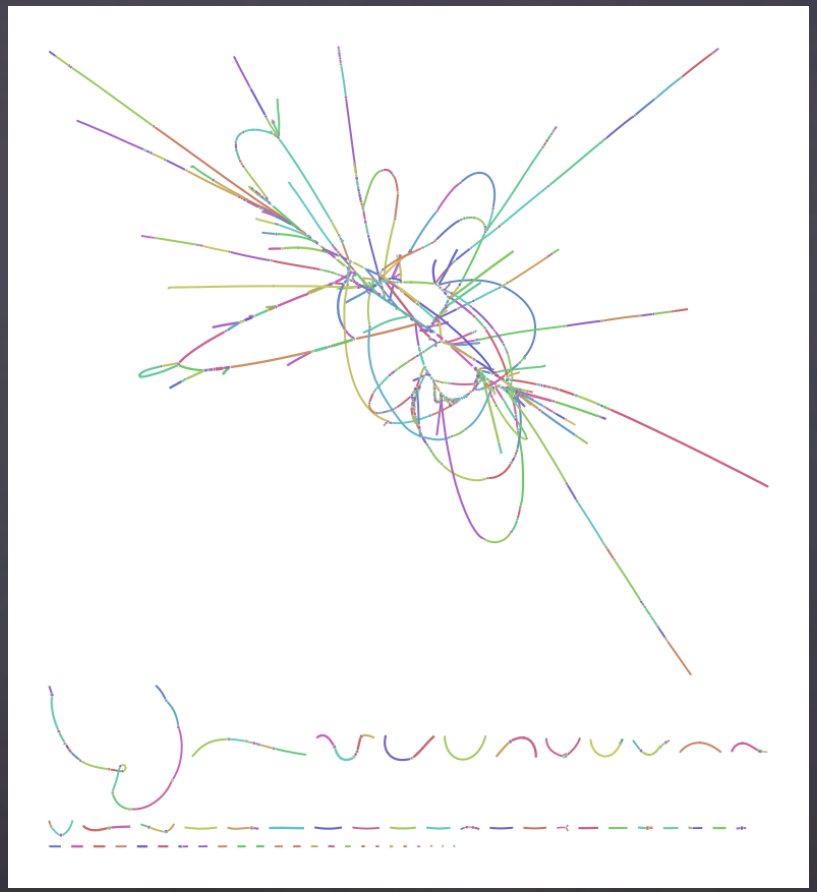
same data & assembler, with tips removed, around 6k nodes

Effect of simplifications on the graph (Illumina data)



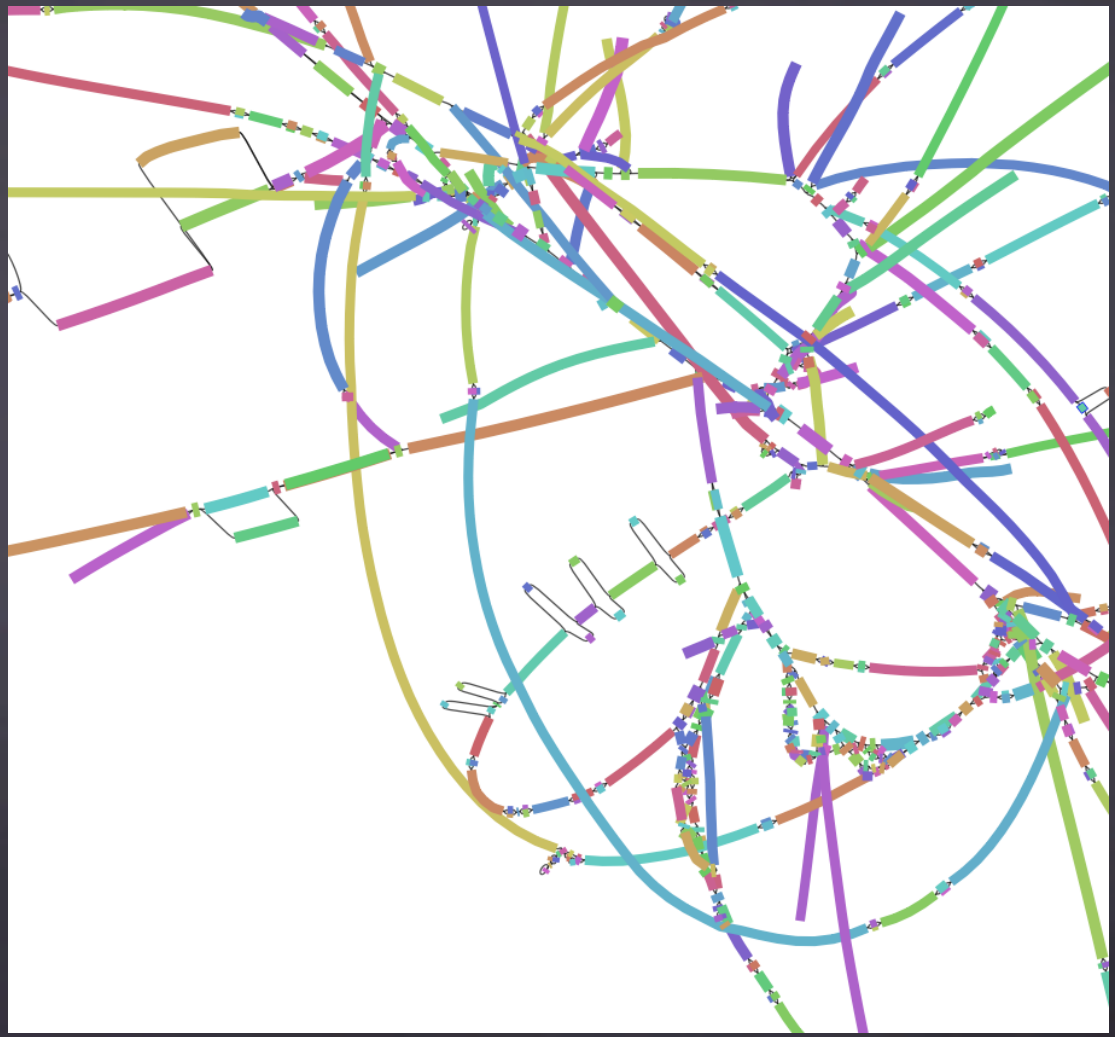
same as previous slide, detail

Effect of simplifications on the graph (Illumina data)



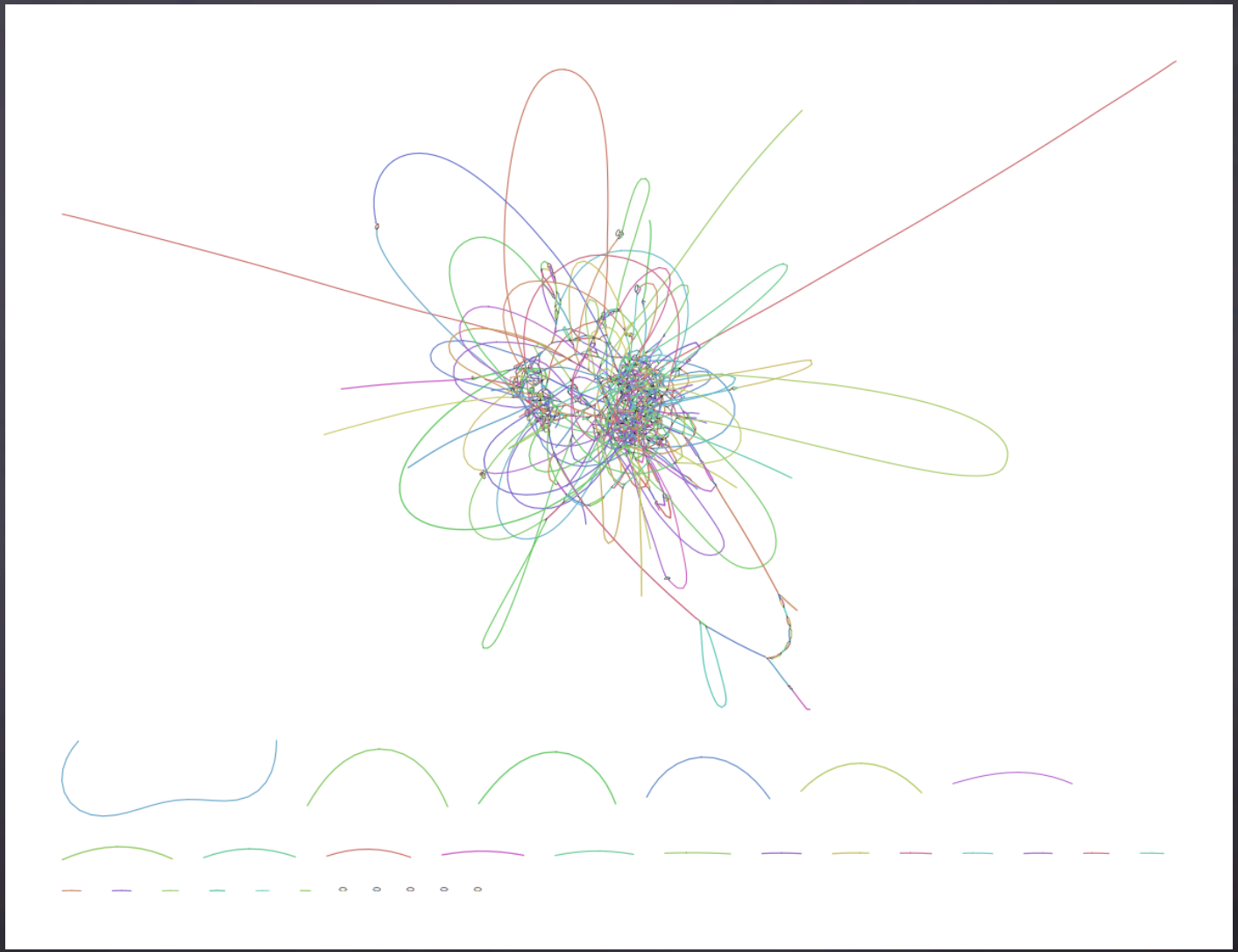
same data & assembler, all simplifications enabled, 1.3k nodes

Effect of simplifications on the graph (Illumina data)



same data as previous slide, detail

Effect of simplifications on the graph (Illumina data)



same data, 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 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

Evaluate a single assembly or compare several assemblies
(from different parameters/assemblers)

Classical metrics using QUAST software:

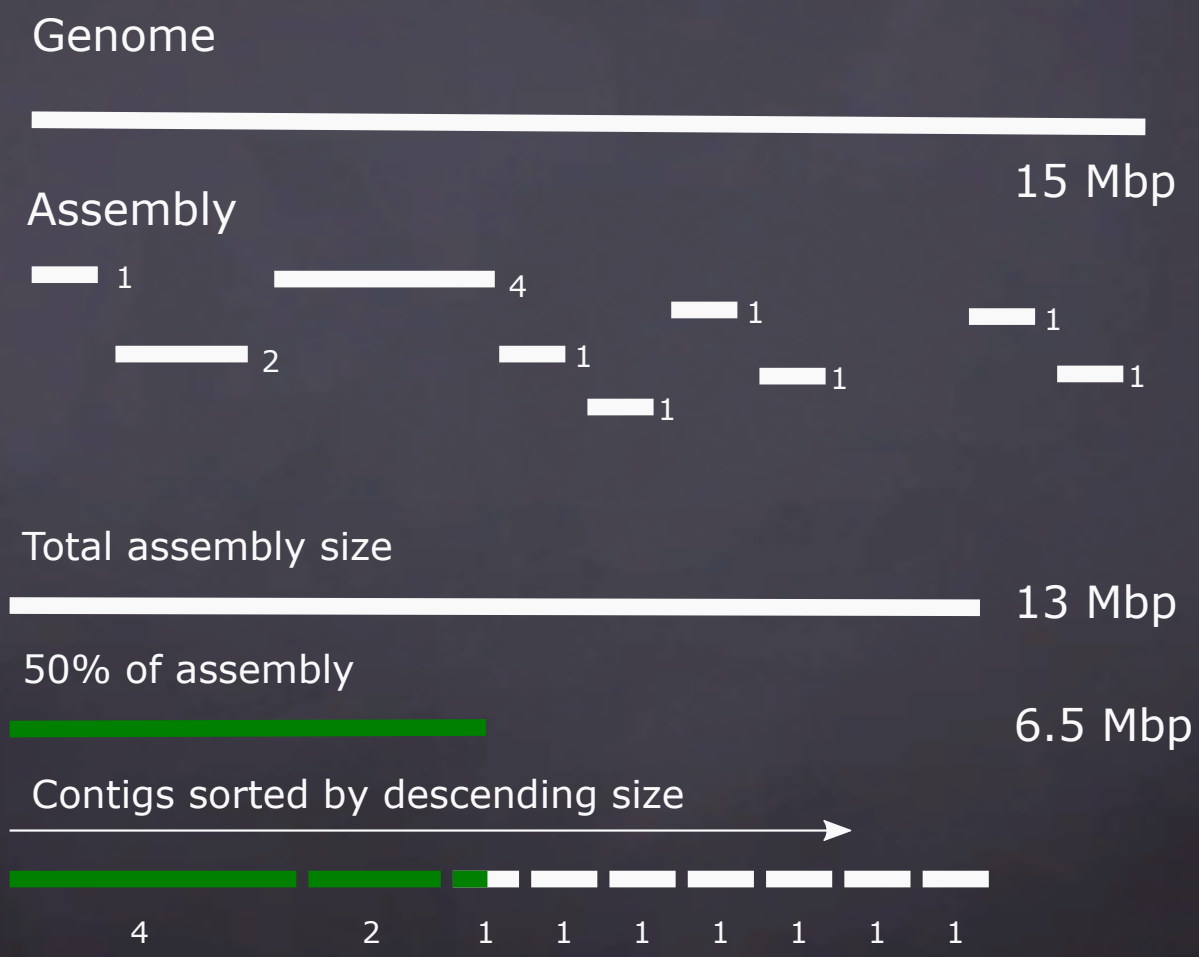
	giraffe
- Number of contigs/scaffolds	2.1M / 2M
- Total length of the assembly	2.93 Gbp
- Length of the largest contig/scaffold	540 kbp / 5.6 Mbp
- Percentage of gaps in scaffolds ('N')	3.2%
- N50/NG50 of contigs/scaffolds	47 kbp / 340 kbp
- Number of predicted genes	17,210
- Number of core single-copy genes (BUSCO software)	

```
./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

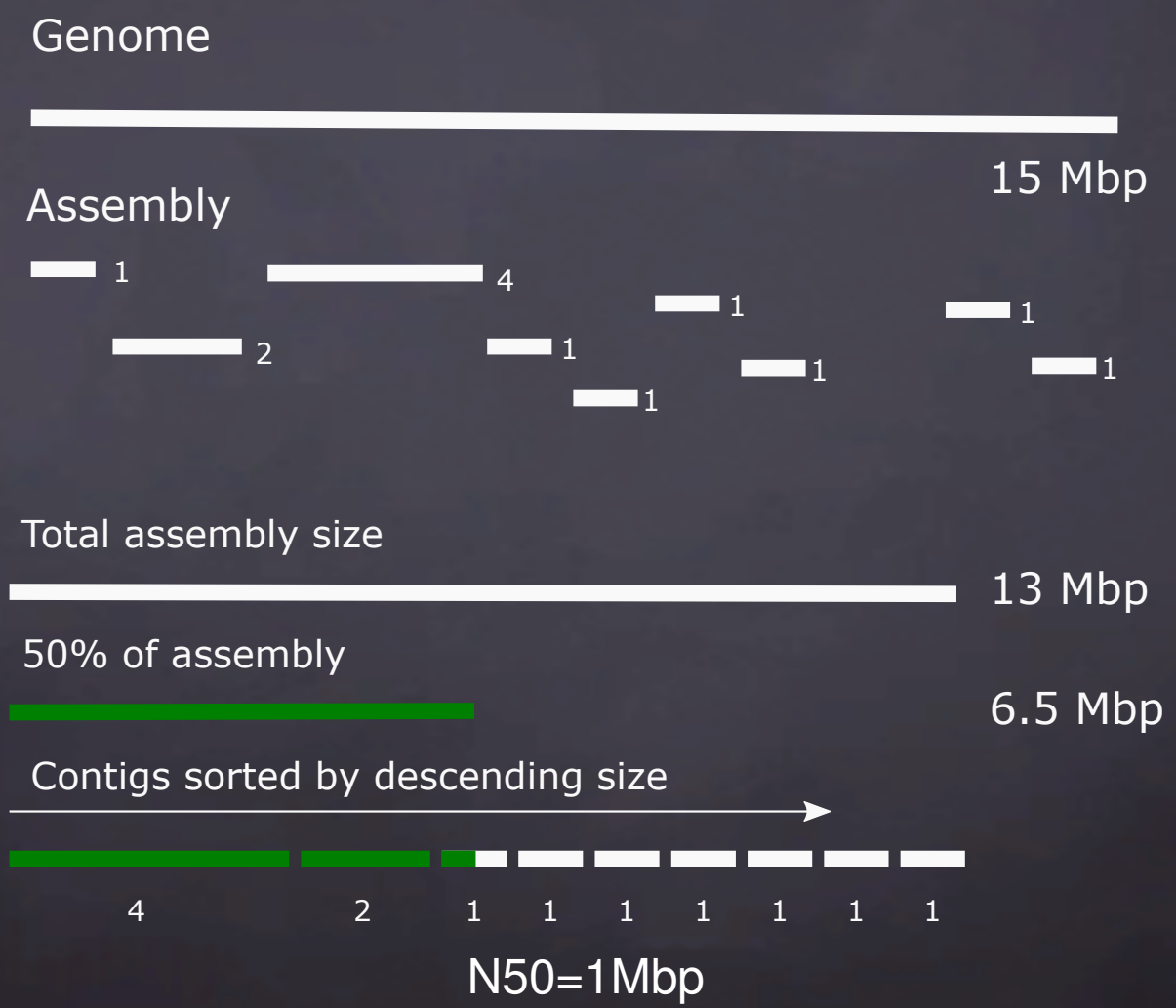
NG50 = Largest contig length at which that contig and longer contigs cover 50% of the total **genome** length



Reference-free metrics: N50

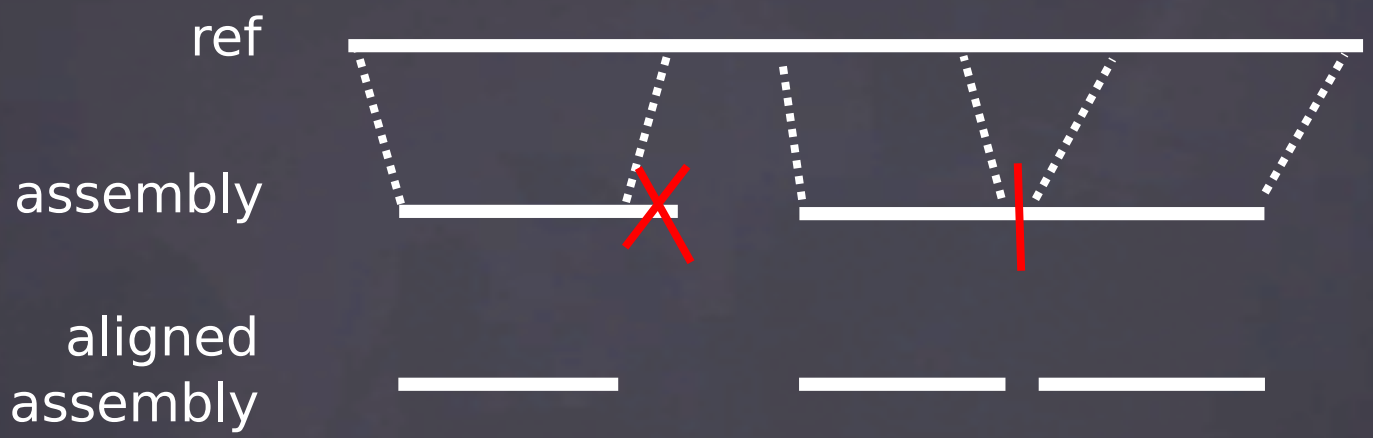
N50 = Largest contig length at which that contig and longer contigs cover 50% of the total **assembly** length

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



Reference-based: NA50

The best metric no-one has heard of.



- Align contigs to reference genome.
- Break contigs at misassemblies and remove unaligned bases.
- Compute N50 of the result. (for NGA50: NG50)

Summary

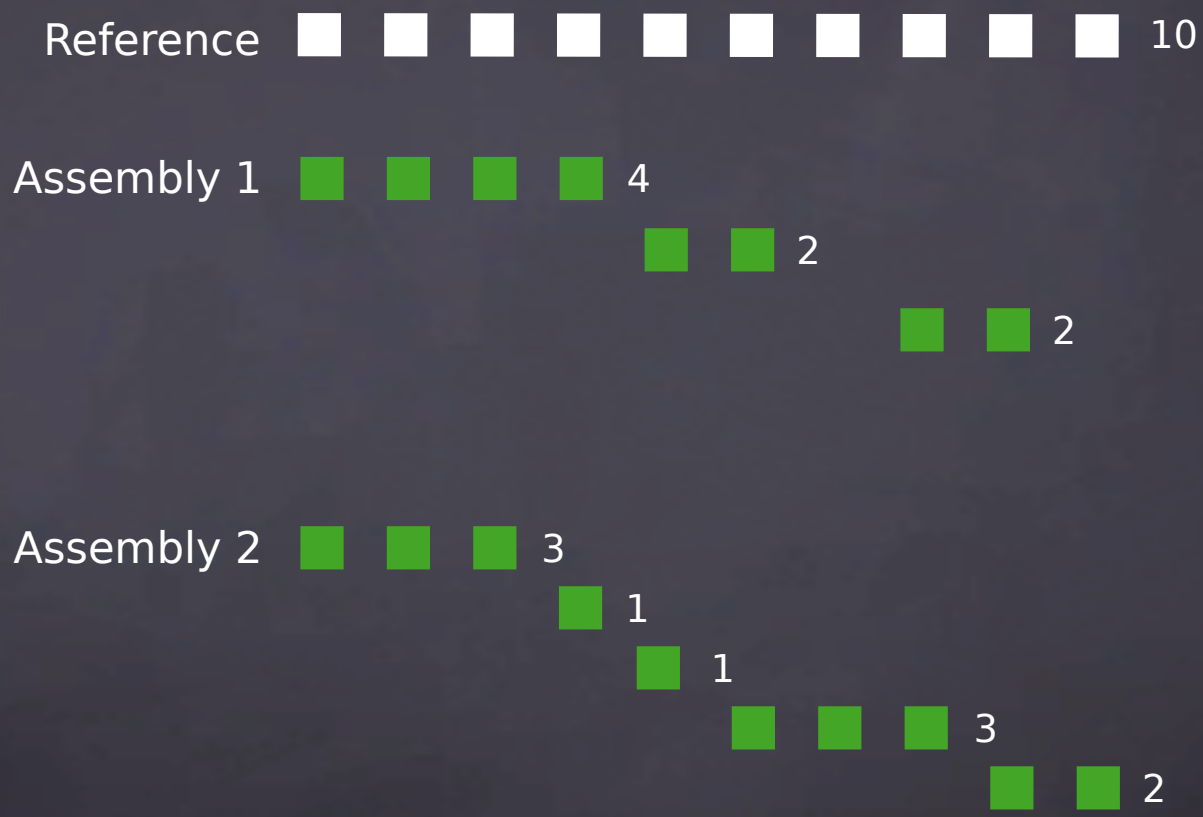
Google 'assembly uncertainty' for a nice summary, blog post by Lex Nederbragt.
In summary:

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

Exercise

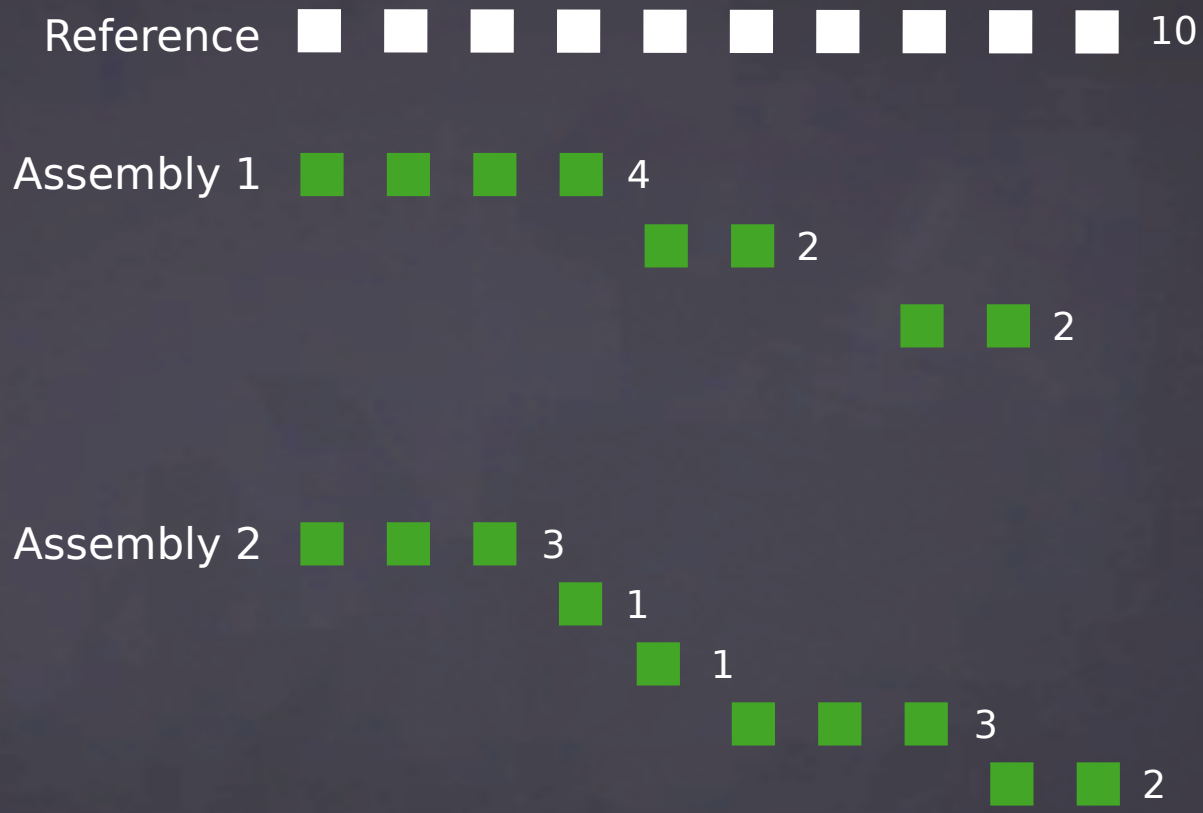
Because at some point in life, one may need to compare genome assemblies.

Here are two assemblies, aligned to the same reference, 1 dot = 1 base pair:



- 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 (8 bp, 10 bp), N50 (4 bp, 3 bp), NG50 (2 bp, 3 bp)
  - Coverage (%) (80, 100)
- Which one is better than the other? (I would say second one: higher NG50, higher coverage. But: more contigs. Mean contig lengths: 2.6 bp versus 2 bp)

Plan

- Fundamentals
  - Basics
  - Short Exercise
- Some useful assembly theory
  - Graphs
  - Contigs construction
  - Exercise
- RNA-seq and metagenomes
- Visualizing and evaluating assemblies
  - Bandage
  - Reference-free metrics
  - Exercise
- One small other thing you can do with *k*-mers
- Assembly in practice
  - Exercise

k-MER MATRIX

k-mer	Abundance per sample				
	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
AAACTG	0	0	0	0	10
ACTGAA	10	12	11	10	9
GTACTG	10	12	11	0	0

Computed for all *k*-mers seen in at least one sample

Finding differences across conditions

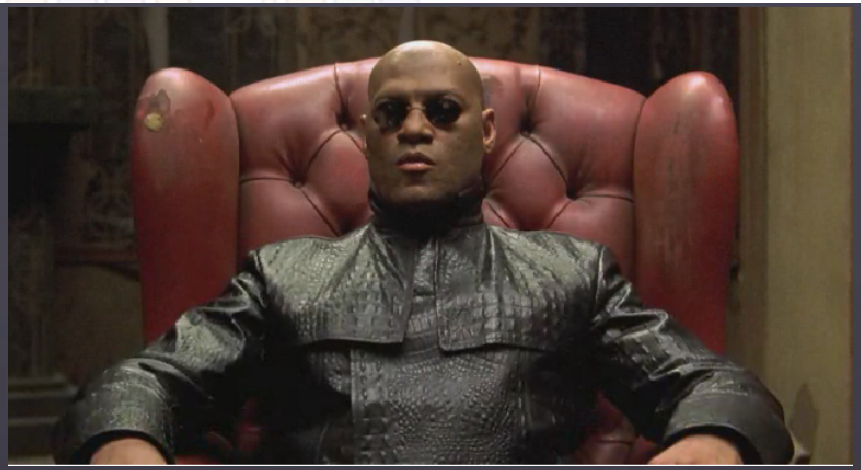
- differential gene expression
- ——”—— transcript expression
- ——”—— usage of exon
- ——”—— alternative splicing
- ——”—— allele-specific expression
- etc..

Tools: DESeq2, DEXSeq, edgeR (see RNAseq lecture)

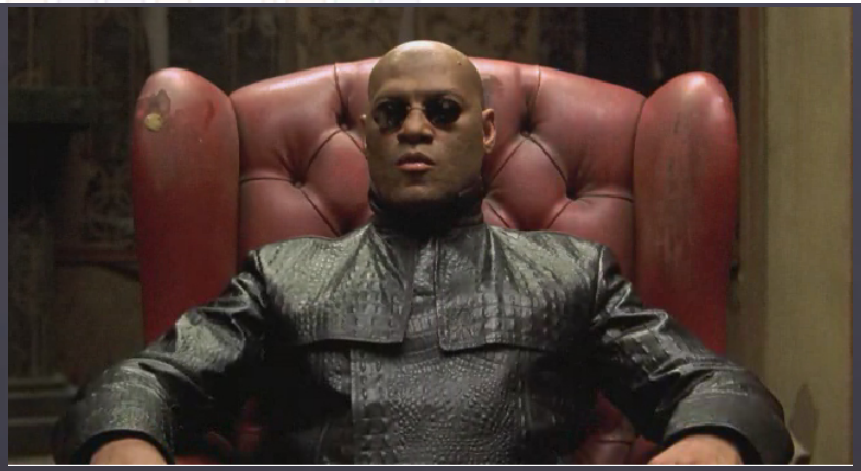
Need GTF or transcriptome.


RNA-Seq reads
Align reads to genome
Assemble transcripts de novo


RNA-Seq reads
Align reads to genome
Assemble transcripts de novo

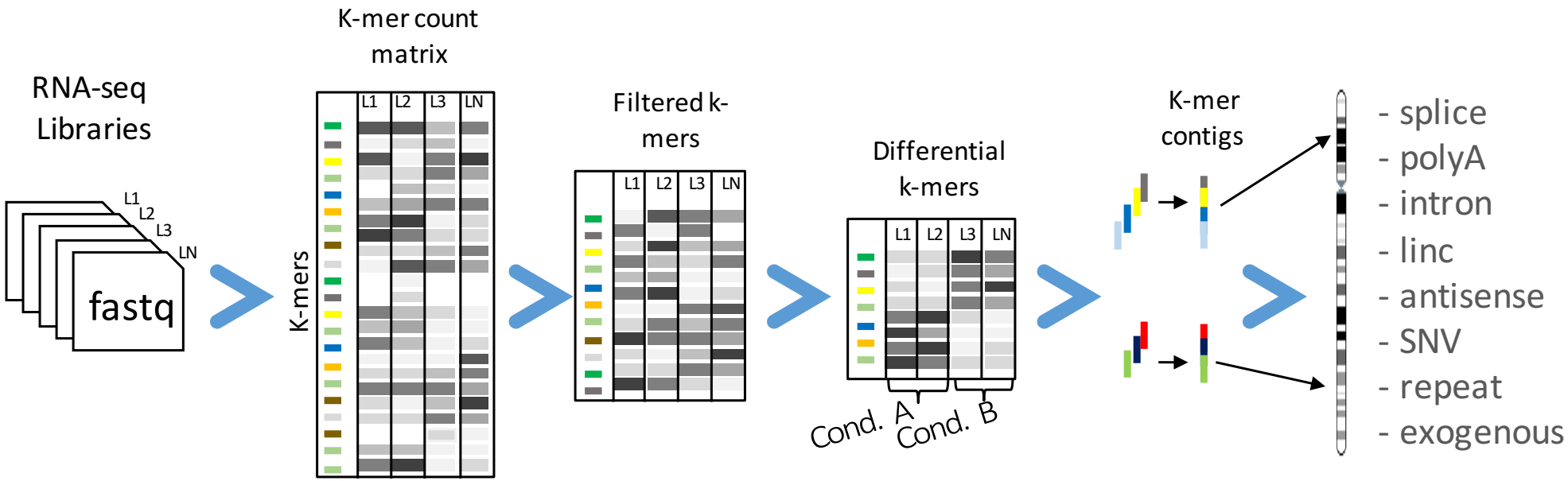



RNA-Seq reads
Align reads to genome
Assemble transcripts de novo



MS

DE-KUPL



Audoux *et al*, Genome Biology (2018)

DE-KUPL RESULTS

condition A: samples 1 & 2

condition B: samples 3 & 4

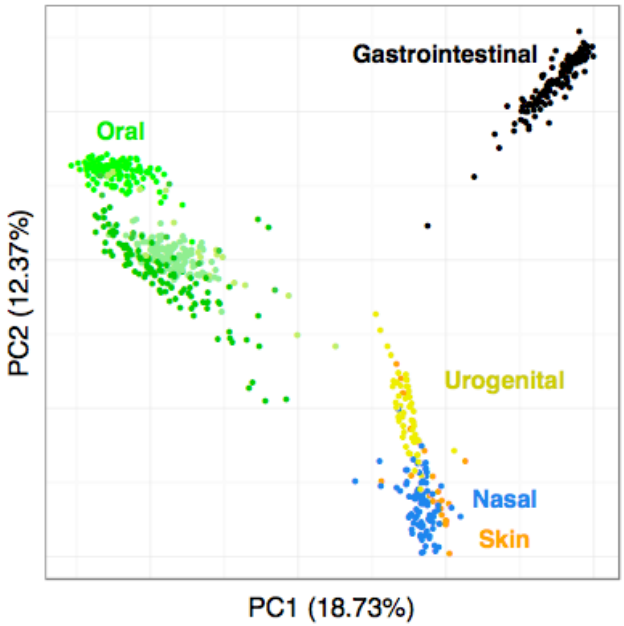
Differential *k*-mer	Abundance per sample				DE-Kupl	
	Sample 1	Sample 2	Sample 3	Sample 4	**logFC**	**p-value**
ACTGAA	10	12	11	10	0.1	0.9
GTACTG	10	12	1	0	3	0.01

In the paper:

- 12 human RNA-Seq, 2 conditions
- 76k DE contigs, 6 hours analysis
- Differential splicing, polyadenylation, lincRNA, antisence RNA, allele-specific expression intron, expressed repeats retention

Simka

[Benoit et al, 2016]



Full HMP data:
n=690 samples,
(32 billion reads)
0.5 day computation

k-mer composition
Abundance table

	A	B	N
ACGAG	2	4	0
CGAGC	2	1	9
GAGCT	0	0	5

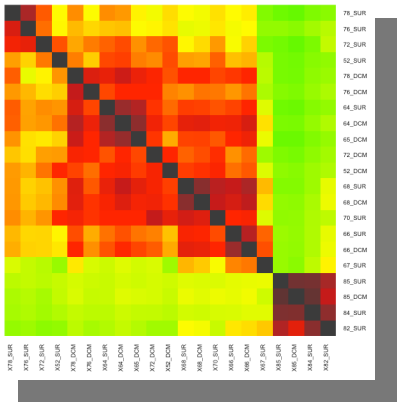
n sequencing datasets

```
>read0
ACACGTCAACGGCTACGGCAGA
CTAACGACTCAGTC
>read1
ACGAGCATCAGGGGACGTATGT
TATGCAGTTCCAGG
```

```
>read0
ACACGTCAACGGCTACGGCAGA
CTAACGACTCAGTC
>read1
ACGAGCATCAGGGGACGTATGT
TATGCAGTTCCAGG
```

```
>read0
ACACGTCAACGGCTACGGCAGA
CTAACGACTCAGTC
>read1
ACGAGCATCAGGGGACGTATGT
TATGCAGTTCCAGG
```

Feature extraction
-
(k-mer counting)



Jaccard, Bray-Curtis, .. similarity matrix

Plan

- Fundamentals
  - Basics
  - Short Exercise
- Some useful assembly theory
  - Graphs
  - Contigs construction
  - Exercise
- RNA-seq and metagenomes
- Visualizing and evaluating assemblies
  - Bandage
  - Reference-free metrics
  - Exercise
- One small other thing you can do with *k*-mers
- Assembly in practice
  - Exercise

Recommended practices (genomes)

- Illumina:
  - 10XGenomics if possible, otherwise mate-pairs
  - Long read lengths, beware of MiSeq 2x300 bp
  - $\geq 50x$ coverage, $\times$ ploidy number
  - Bacteria: no point going $\geq 200x$
- PacBio/Nanopore:
  - $\geq 40x$, for now

For determining minimum coverage, have a look at *Evaluation of hybrid and non-hybrid methods for de novo assembly of nanopore reads* [Sovic et al, 2016], h/t D. Larivière

Assemblers, personal experience, 2019

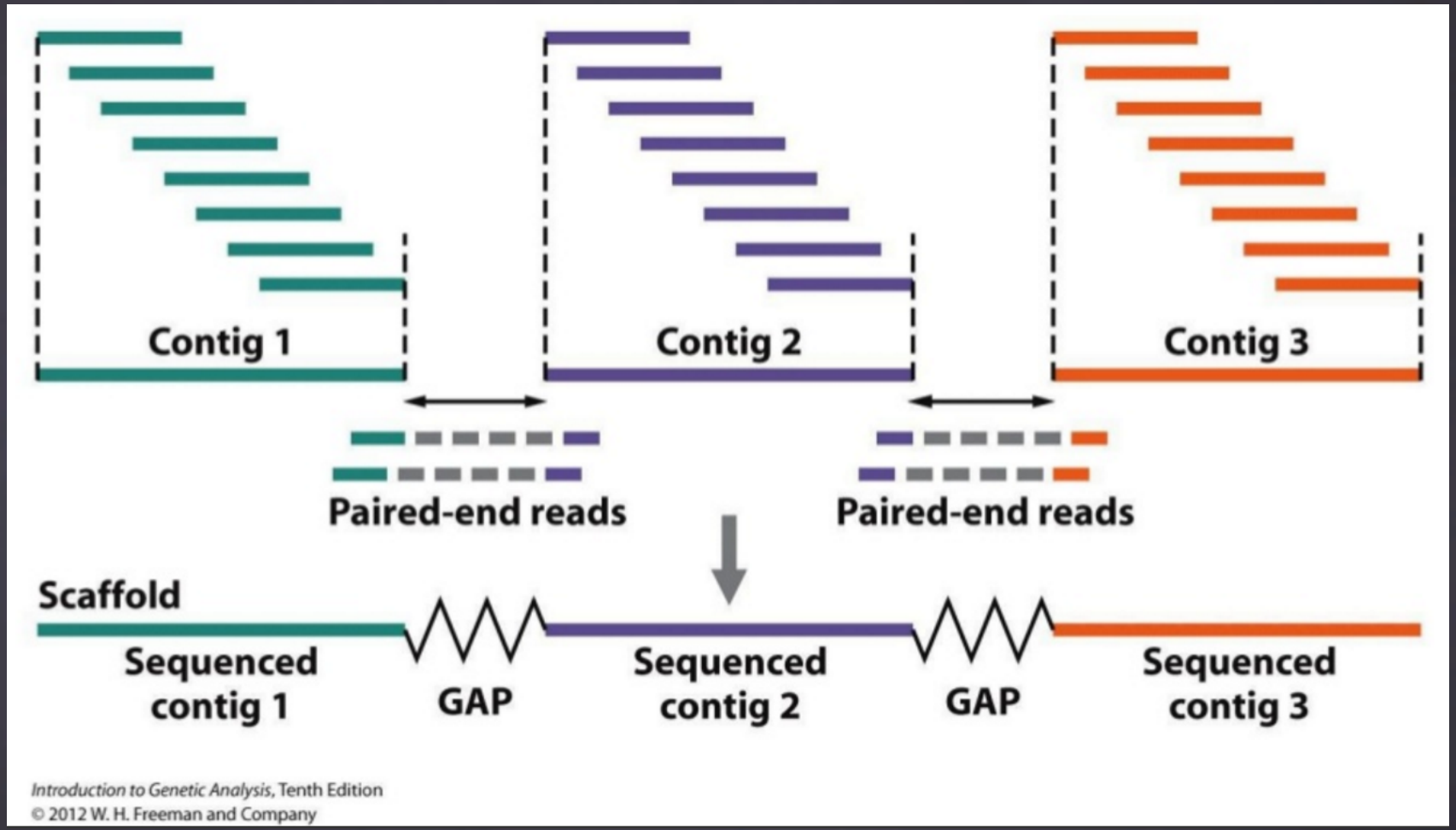
PacBio, Nanopore	Canu
Illumina	SPAdes
10X	Supernova
RNA-Seq	Trinity
Memory issues	Minia pipeline
Large metagenomes	Megahit

Disclaimer: there is a really long list of other amazing assemblers and I wish I had the time to present their benefits and use-cases.

Meta-practices

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

SCAFFOLDING



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

Hybrid assembly

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

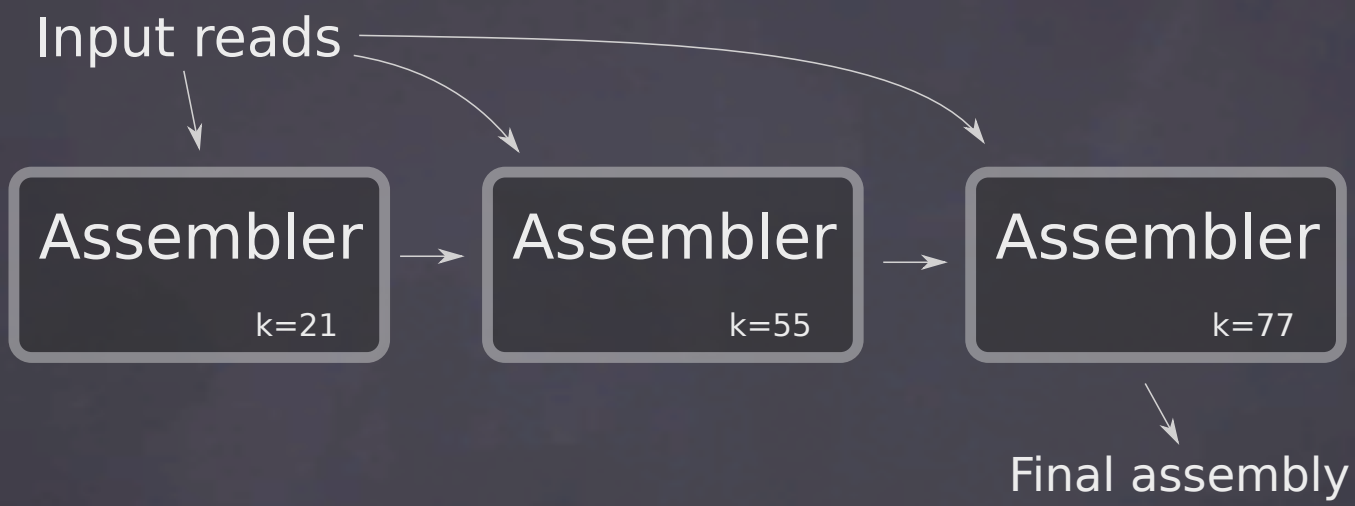
1. high-coverage Illumina paired-end / mate-pairs
2. low-coverage PacBio

Improve an Illumina assembly using:

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

Not aware of any 10X hybrid assembler.

Multi-k assembly



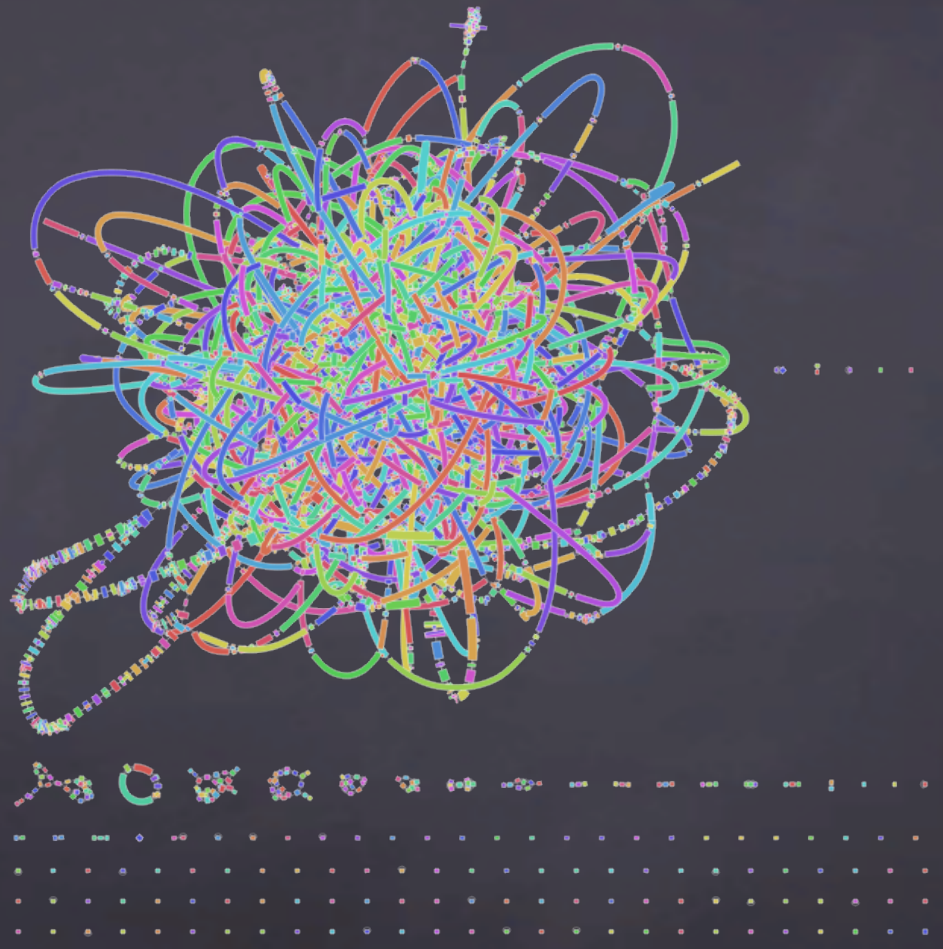
In principle, **better** than single-k assembly.

Notable assemblers that implement multi-k:

- IDBA, SPAdes, Megahit

Effect of k-mer size

Salmonella genome, Velvet assembly, 100 bp Illumina reads.

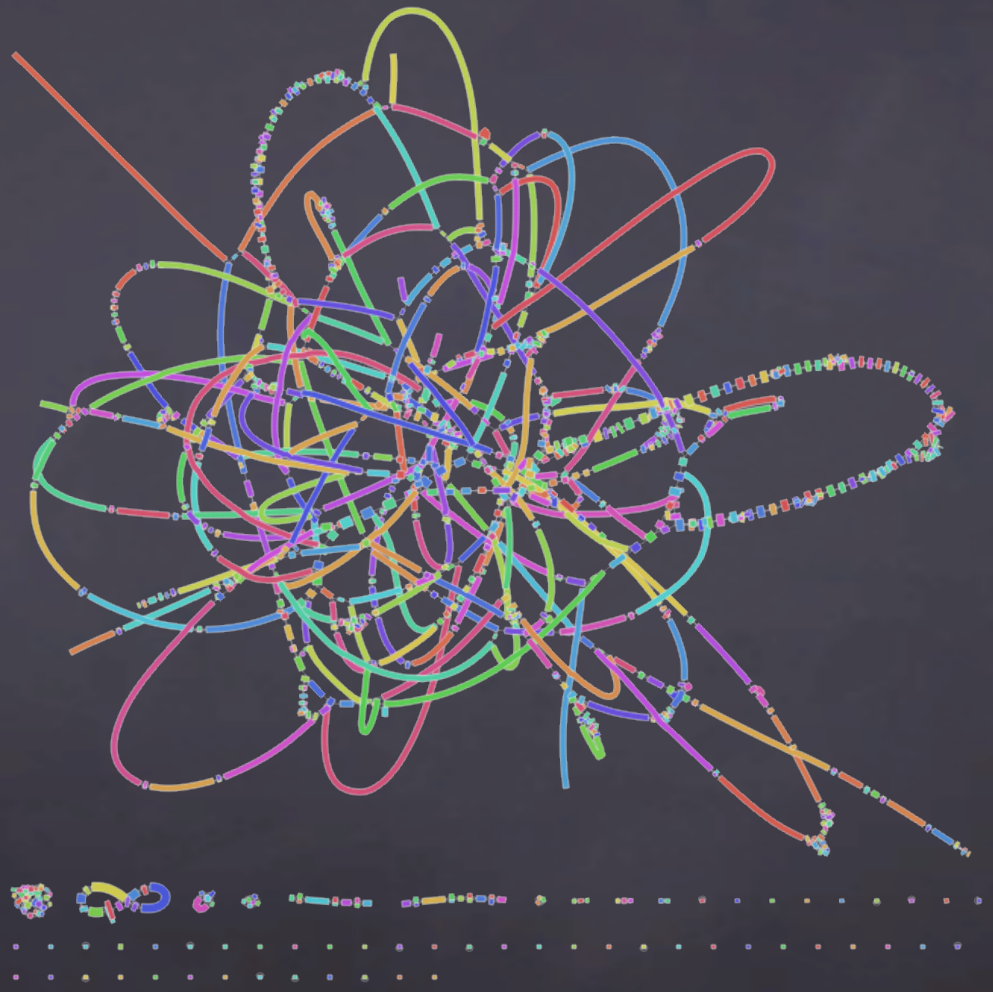


$k = 51$

Fig: https://github.com/rrwick/Bandage/wiki/Effect-of-kmer-size

Effect of k-mer size

Salmonella genome, Velvet assembly, 100 bp Illumina reads.

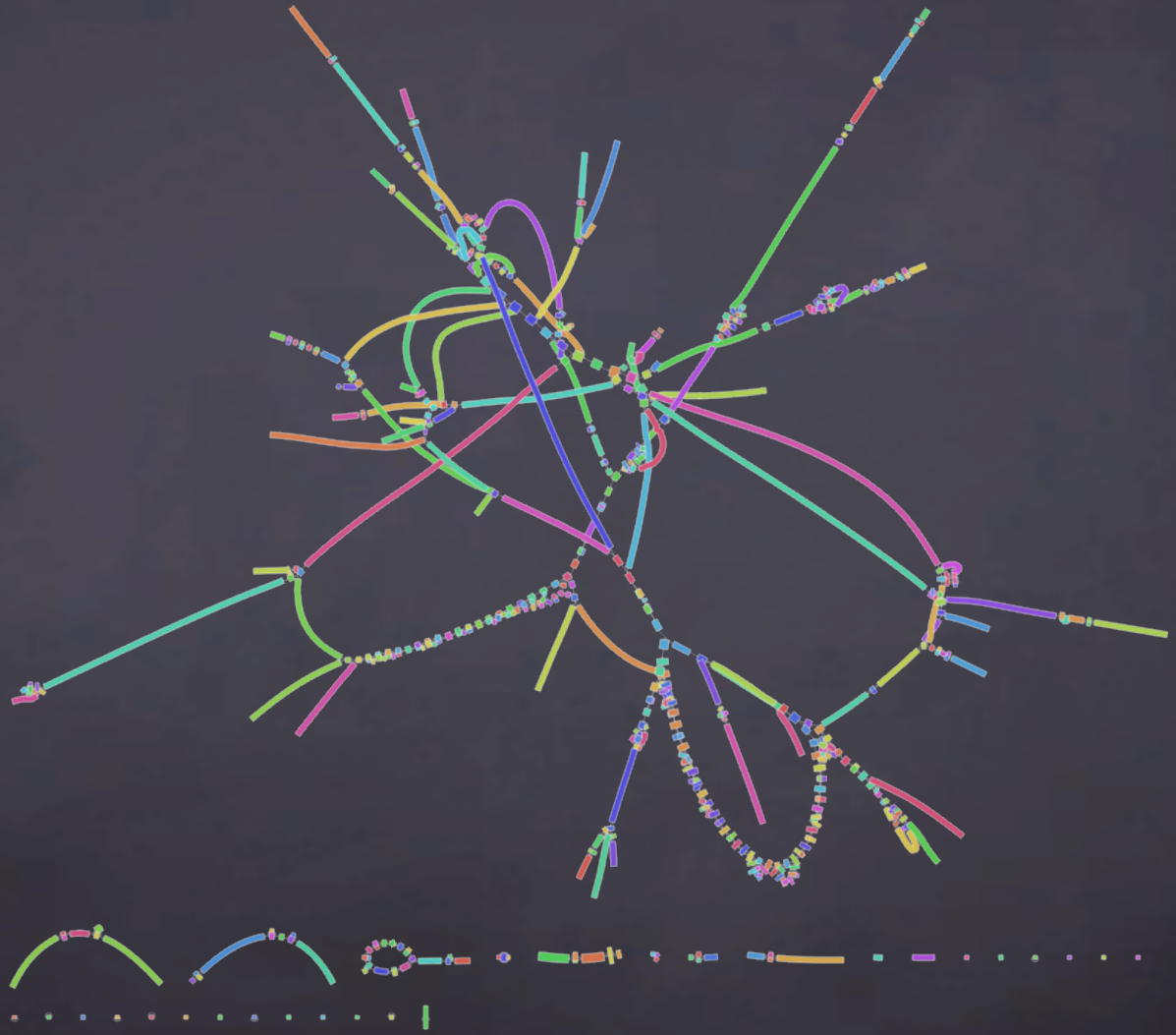


$k = 61$

Fig: https://github.com/rrwick/Bandage/wiki/Effect-of-kmer-size

Effect of k-mer size

Salmonella genome, Velvet assembly, 100 bp Illumina reads.

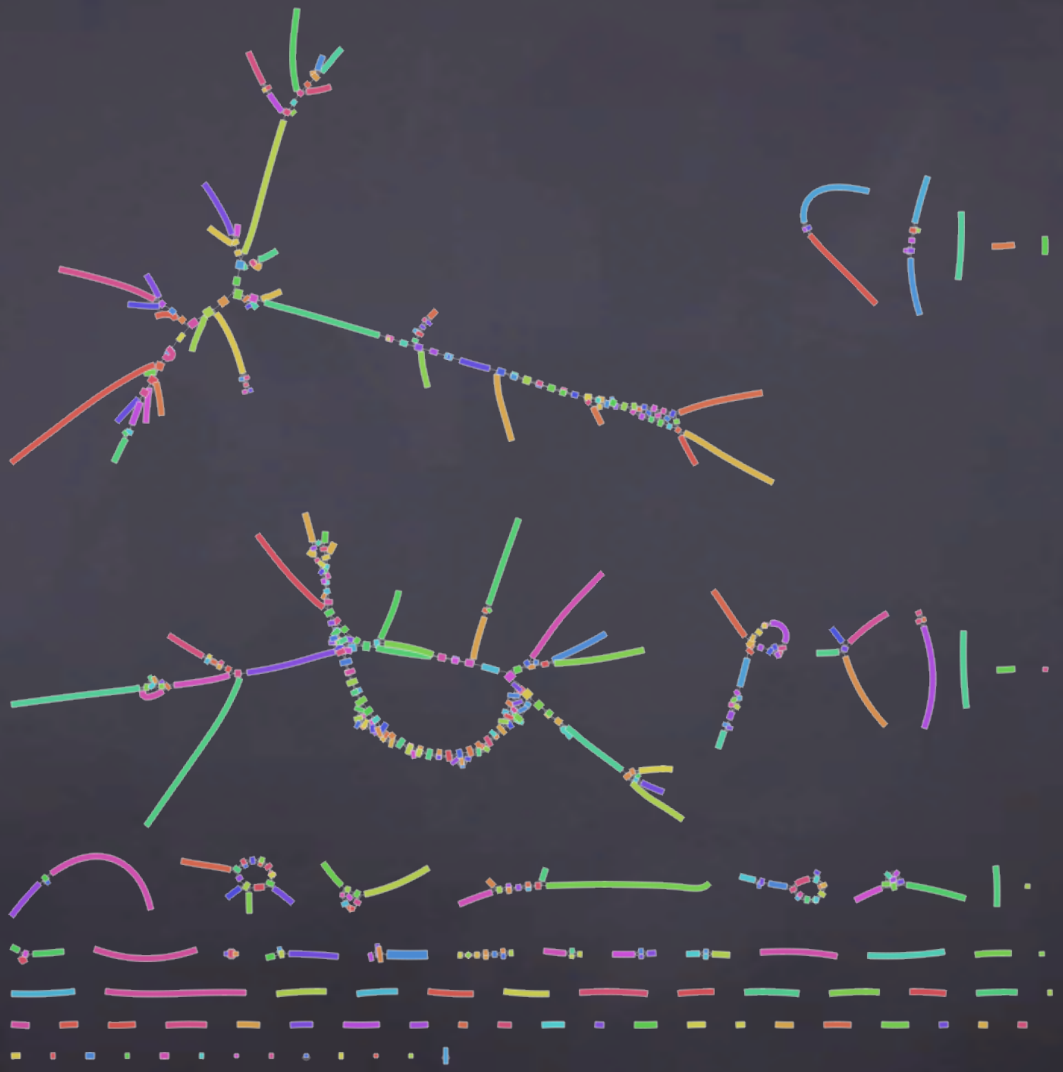


$k = 71$

Fig: https://github.com/rrwick/Bandage/wiki/Effect-of-kmer-size

Effect of k-mer size

Salmonella genome, Velvet assembly, 100 bp Illumina reads.

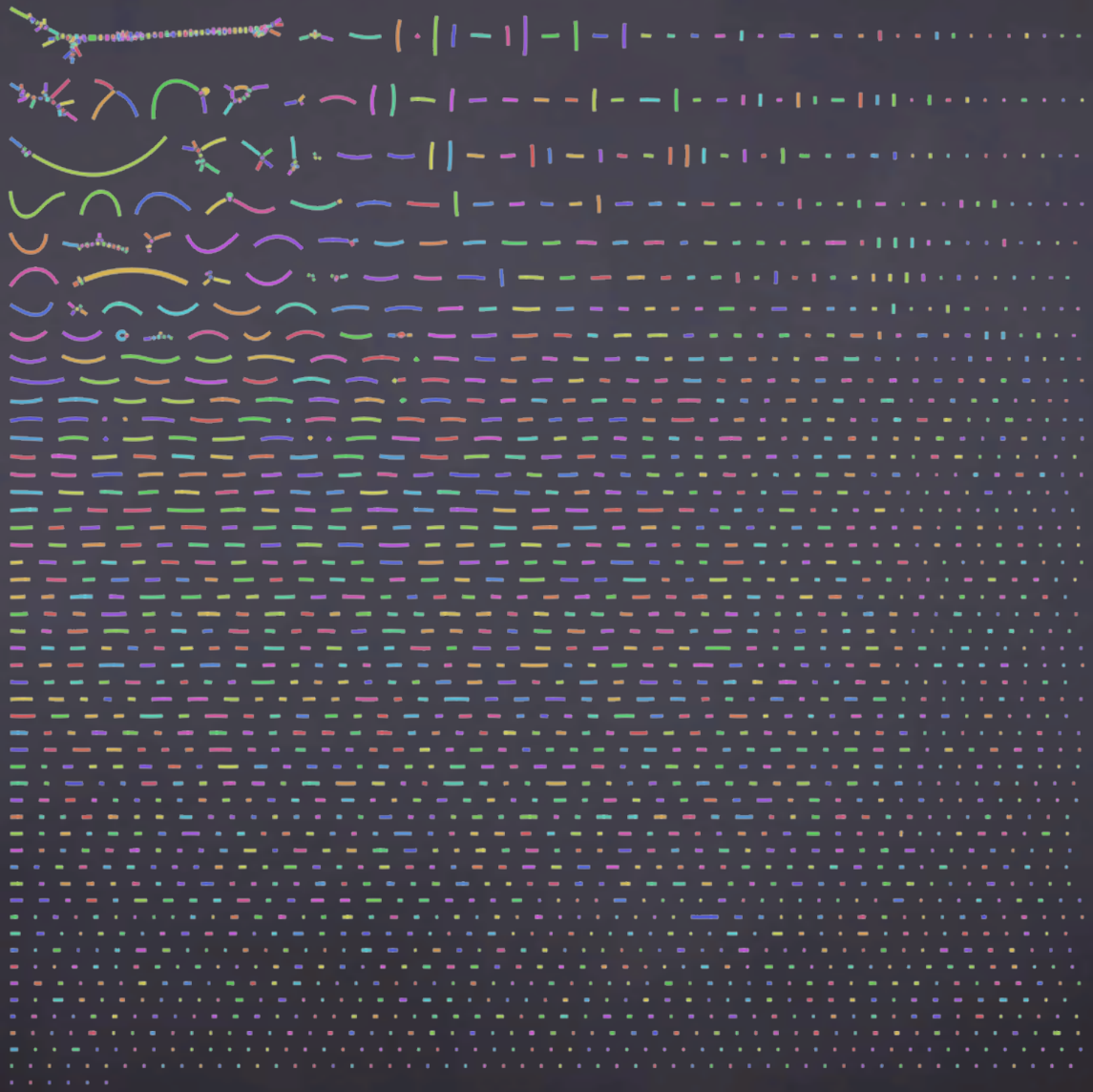


$k = 81$

Fig: https://github.com/rrwick/Bandage/wiki/Effect-of-kmer-size

Effect of k-mer size

Salmonella genome, Velvet assembly, 100 bp Illumina reads.

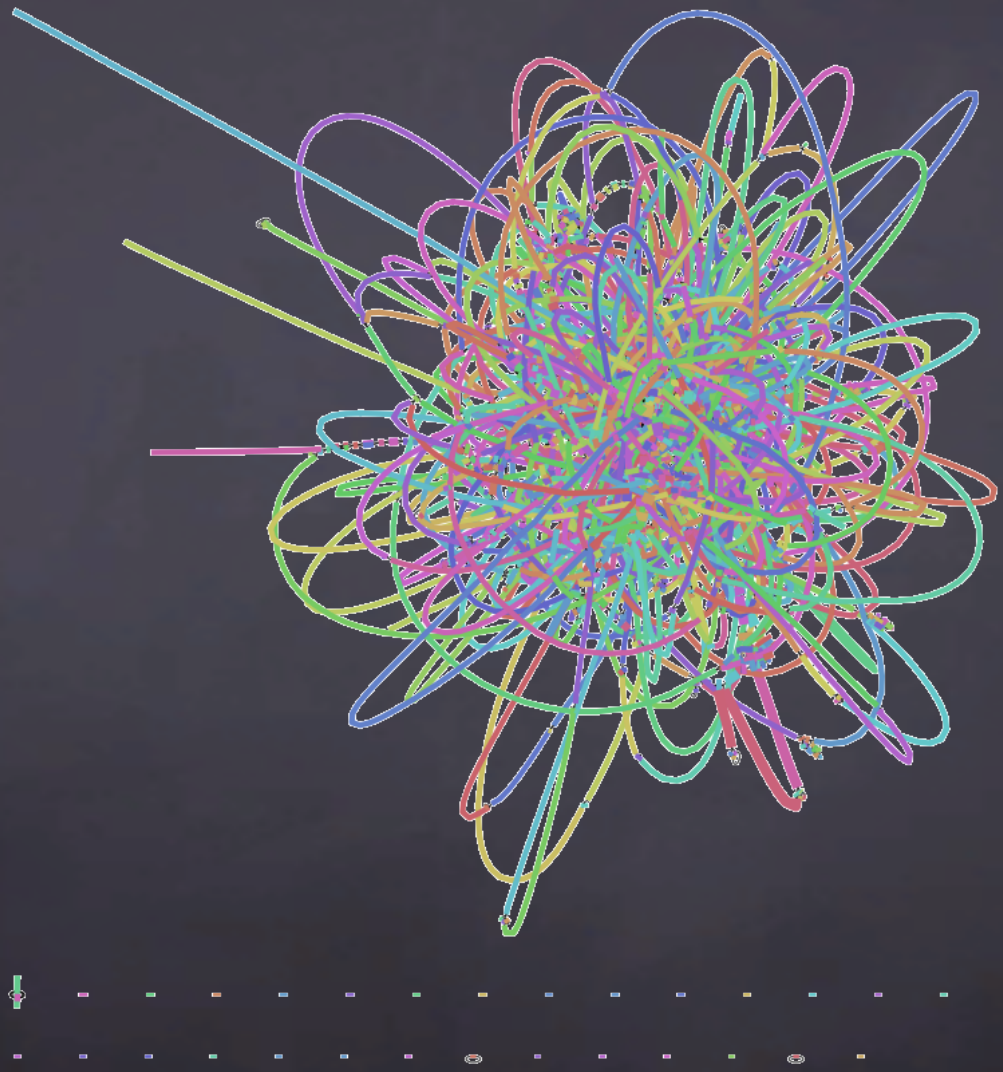


$k = 91$

Fig: https://github.com/rrwick/Bandage/wiki/Effect-of-kmer-size

Visualization of multi-k graphs

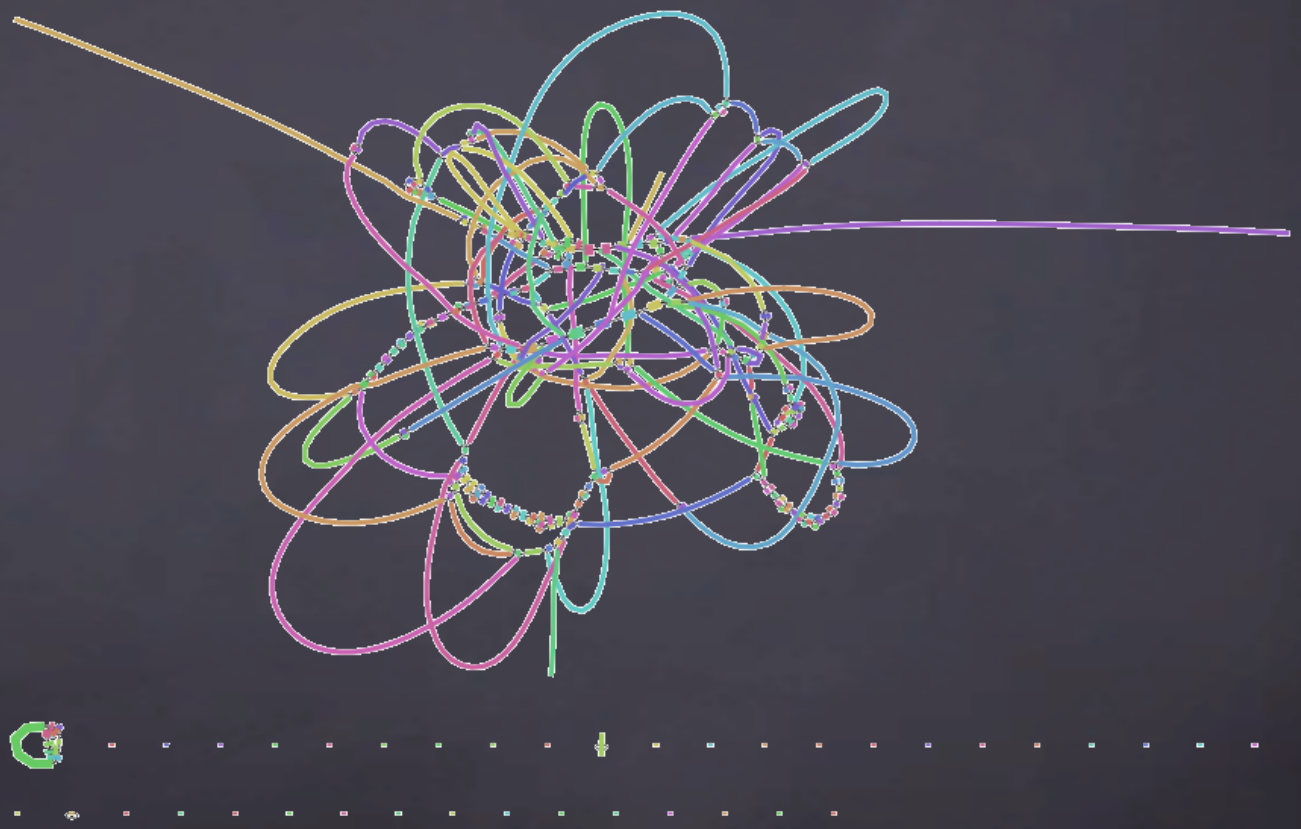
Salmonella genome, SPAdes assembly, MiSeq reads.



$k = 21$

VISUALIZATION OF MULTI-K GRAPHS

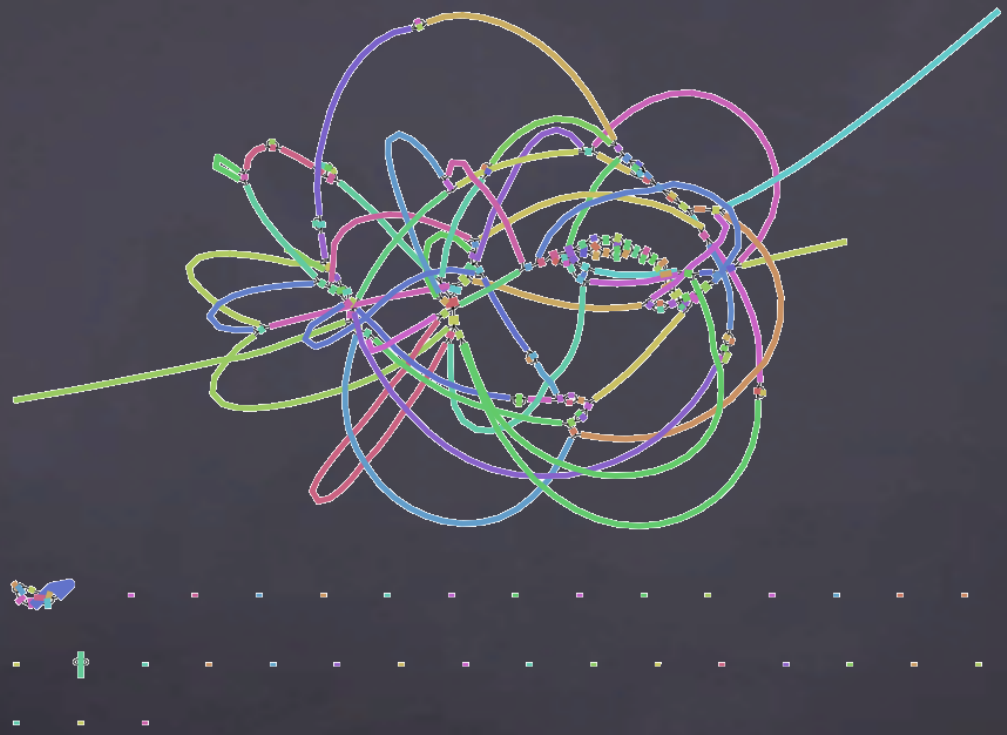
Salmonella genome, SPAdes assembly, MiSeq reads.



$k = 55$

VISUALIZATION OF MULTI-K GRAPHS

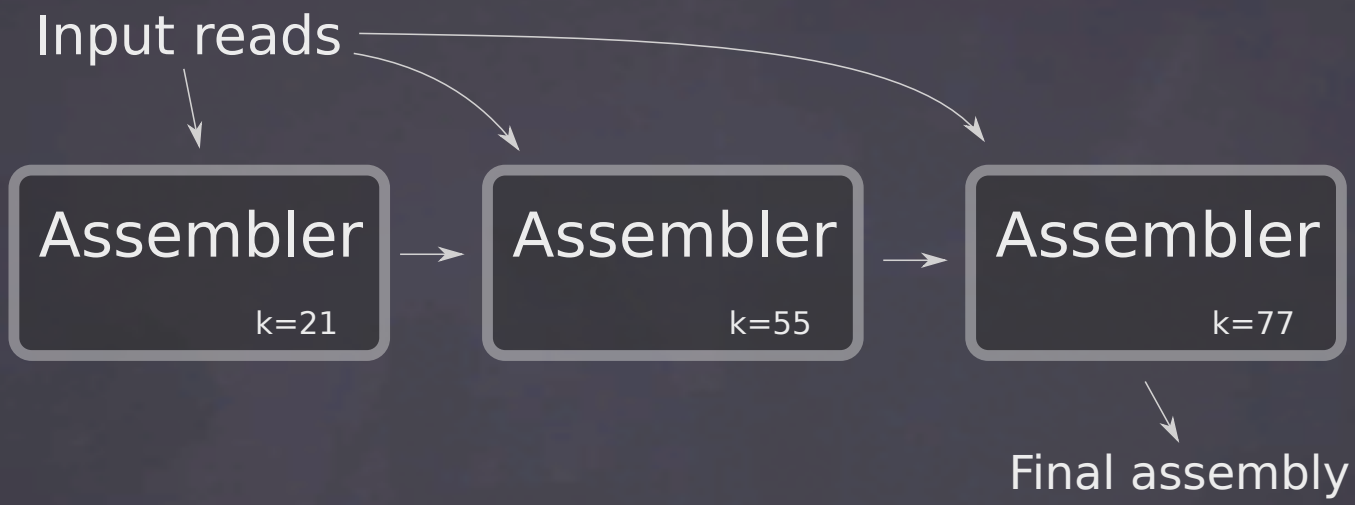
Salmonella genome, SPAdes assembly, MiSeq reads.



$k = 99$

$\rightarrow$ Still a single component, less repeat-induced complexity

Multi-k assembly



Notable assemblers that implement multi-k:

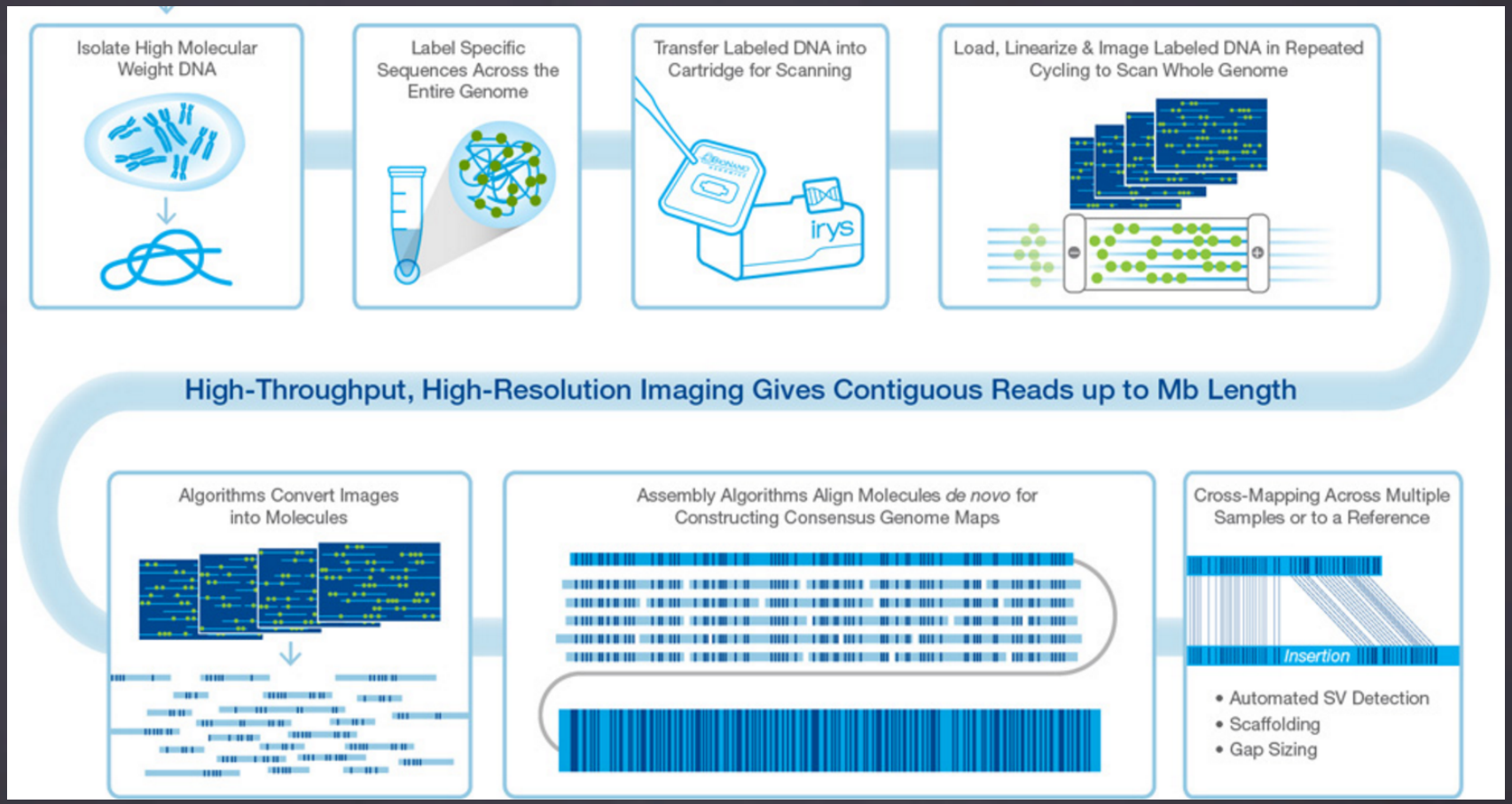
- IDBA, SPAdes, Megahit

Notable assemblers that don't:

- Velvet, SOAPdenovo, Trinity, ABySS


I AM WHY WE CAN'T HAVE NICE THINGS
Also I assemble using a single k value

GENOME MAPS



Bionano workflow

Other techs: Nabsys, OpGen (similar). Dovetail, Hi-C (different)

Chromonomer: Create or correct scaffolds using RAD-seq, or other markers.

Common question: should I trim the reads?

To check and remove adapters: yes absolutely

Quality-trim: I'd say no

Polishing

- Improve base-level accuracy of assemblies
- From <99.9% accuracy to >99.9%
- By re-aligning reads to assembly
- Mainly for PacBio/Nanopore
- Pilon
- Racon
- Nanopolish

(this slide isn't very pretty, I'd need to polish it ;)

Annotation

- Genes
- TEs (e.g. REPET)
- Repeat-masking (e.g. Repeatmasker)
- *ab initio* gene prediction (e.g. Augustus)
- external evidence (e.g. RNAseq, NCBI nr, ..)
- combining everything (e.g. Maker)

Good review: *Ten steps to get started in Genome Assembly and Annotation* [Angel et al, F1000 2017]

Assembly: a solved problem?

Still challenging, even in 2019.

- PacBio/Nanopore tools are slowly maturing
- No competition for 10X assemblers
- ~~Hard~~Hopeless to obtain best-quality assemblies from plain Illumina data
- High computational requirements overall

State of the research

1. More PacBio/Nanopore assemblers
2. Reconstruction of hapotypes (sequence the parents!)
3. *k*-mer methods everywhere
4. us: assembly “debugging”

Last Exercise

Reads:

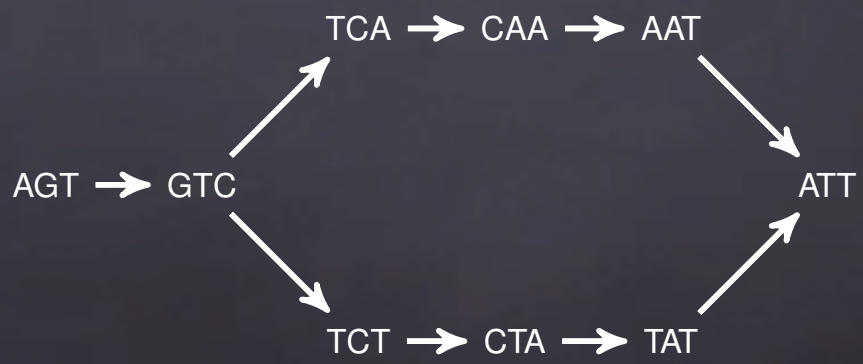
1. AGTC
2. TCAA
3. AATT
4. GTCT
5. TATT
6. TCTA
7. TCAA
8. TCTA

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

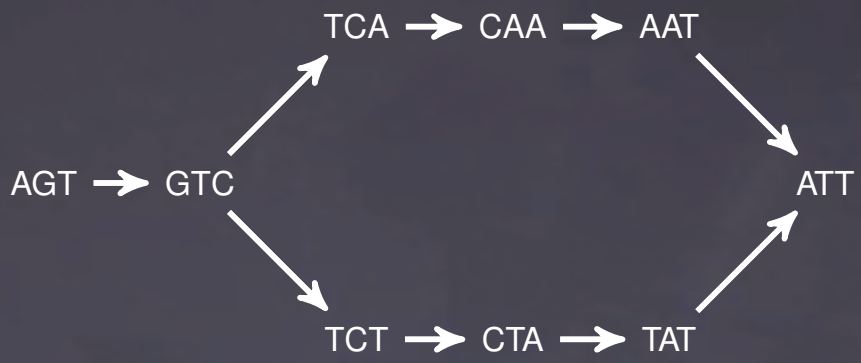
Last Exercise (detailed solution)

Step by step:

- **Choose an assembly model**: de Bruijn graph or string graph
- *The reads are short, let's choose the DB 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":

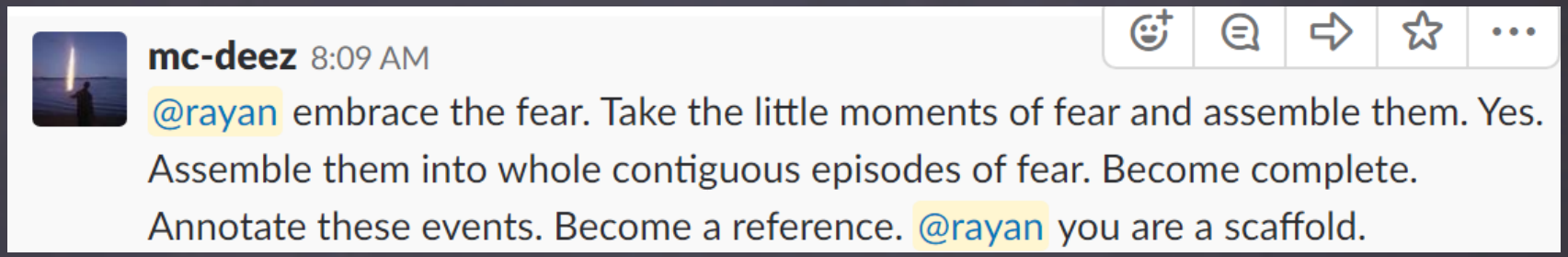
  `AGTCAATT`

  `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

- Assembly evaluation
  - No total order
  - Use QUAST, BUSCO
- Techniques
  - de Bruijn graph (Illumina, 10X), string graph (PacBio, Nanopore)
  - Errors and small variations are removed
  - Contigs are just simple paths from the graph
  - Scaffolds are linked contigs, prone to misassemlies
- *k*-mers
  - Histograms, reference-free variant detection (DNA, RNA)
  - Not covered: taxonomic assignment, digital normalization, error correction, pseudoalignment
- Takeaways
  - Try another assembler
  - Try different parameters
  - An assembly is not the **absolute truth**, it is a mostly complete, generally fragmented and mostly accurate **hypothesis**

Don't fear the assembly



Lex Nederbragt
@lexnederbragt

En réponse à @ctitusbrown

"Finding your way in life is like finding the genome in a De Bruijn graph: it is very easy to find *a* path, very hard to find *the* path".