Genome assembly with either short reads or long reads

Rayan Chikhi

Institut Pasteur & CNRS

Helsinki Bioinformatics Day 2019

Bio

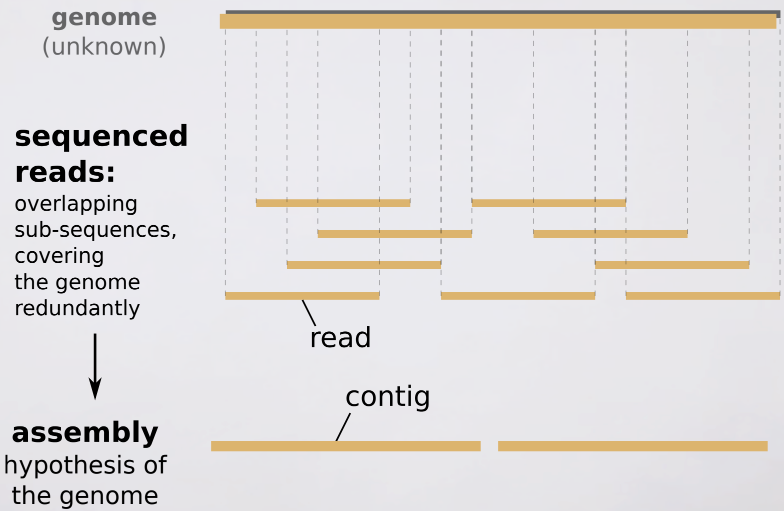
@RayanChikhi
`http://rayan.chikhi.name`

- Compsci/math background
- **Algorithms and data structures for what comes out of DNA sequencers**
- Software:
  - Minia, DSK, Bcalm2, KmerGenie, GATB
- Real assemblies:
  - some bacterias, giraffe, gorilla Y, mountain goat, water buffalo

This talk

- state of short reads assemblers
- state of long reads assemblers
- helping long reads assemblers

Genome assembly

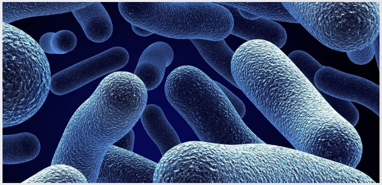


Why assemble

- **Reconstruct** a genome
- a transcriptome
- a pangenome
- novel **insertions**
- **SNPs** in non-model organisms

Also used in:

- DNA variants detection
- Transcript quantification
- Alternative splicing detection



Happy b-day genome assembly

(Staden 1979) *"With modern fast sequencing techniques and suitable computer programs it is now possible to sequence whole genomes without the need of restriction maps."*

(Adapted from A. Phillippy's talk, RECOMB-Seq'19)

Genome assembly software is complex

- Coding: PhD (3 years), or team of engineers (1-2 years)
- Several not-always-independant components
- Heuristics everywhere

A good genome assembler is like a good sausage,
you'd rather not know how it was made.

’

(S. Gnerre, ALLPATHS assembler)

Short-read assemblers

de Bruijn graphs

A **de Bruijn** 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

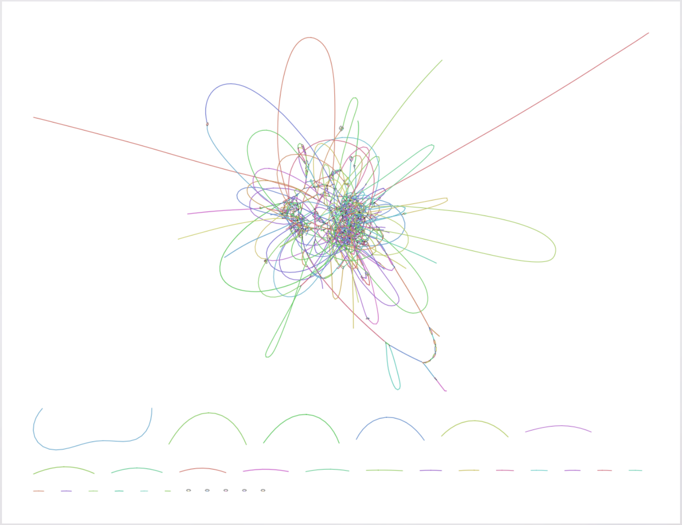
Reads:

```
AGCCTGA
AGCATGA
```

dBG, $k = 3$:

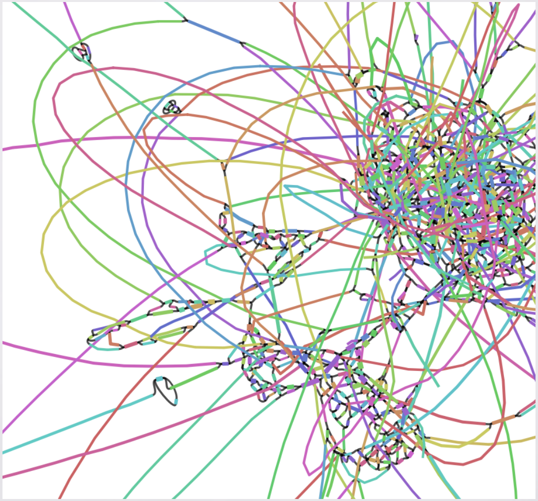
$$\begin{tikzcd}
 & \text{GCC} \arrow[r] & \text{CCT} \arrow[r] & \text{CTG} \arrow[dr] & \\
\text{AGC} \arrow[ur] \arrow[dr] & & & & \text{TGA} \\
 & \text{GCA} \arrow[r] & \text{CAT} \arrow[r] & \text{ATG} \arrow[ur] &
\end{tikzcd}$$

Actual compacted de Bruijn graph



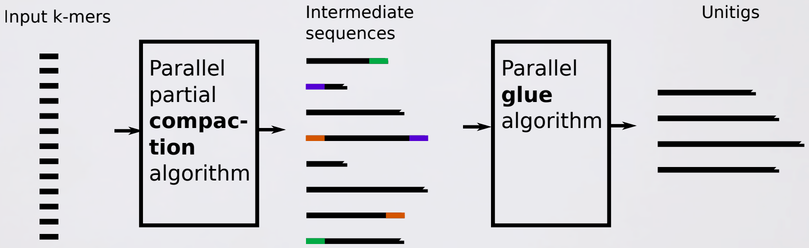
chr14:20Mbp-20.5Mbp GAGE PE reads, SPAdes 3.8 k=31, 1k nodes

Actual compacted de Bruijn graph



same as previous slide, detail

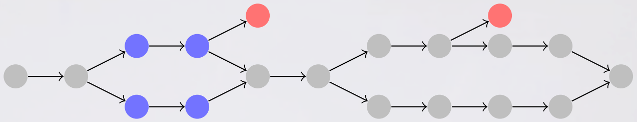
BCALM2 [ISMB'16]: construction of compacted de Bruijn graphs


Input k-mers
Parallel partial **compaction** algorithm
Intermediate sequences
Parallel **glue** algorithm
Unitigs

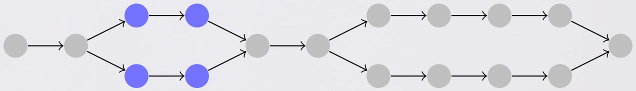
Algorithmic ingredients: minimizer partitioning, fast Malfoy-made compaction algorithm, concurrent union-find, minimal perfect hashing

Short read assemblers

1) de Bruijn **graph** construction



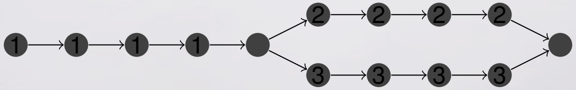
2) Likely sequencing errors are removed.



3) Variations (e.g. SNPs, similar repetitions) are removed.

→ **Collapses strains**

4) **Simple paths** (i.e. contigs) are returned.



5) Extra steps: repeat-resolving, scaffolding

Short read assemblers

- have matured
- now tend to converge towards similar ideas
- mostly useful for metagenomics, transcriptomics
- also large genomes (ABySS2)

→ Careful recovery of low-abundance k-mers, graph simplifications, **multi-k**, heuristic scaffolding

Exhibit 1: MEGAHIT < v1.0

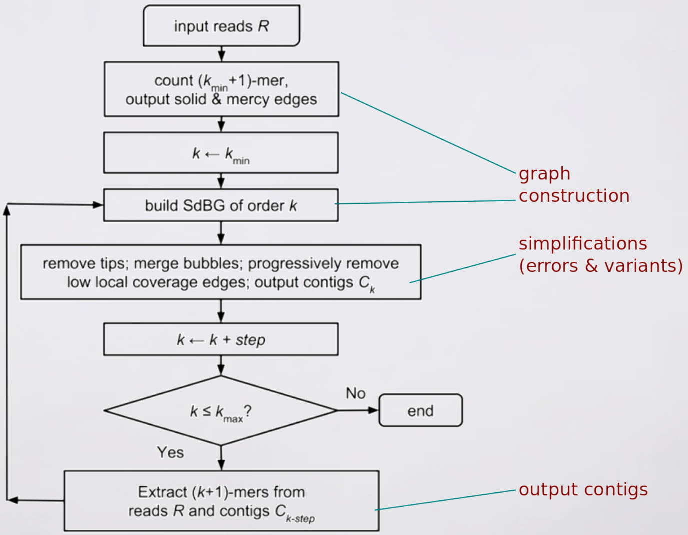
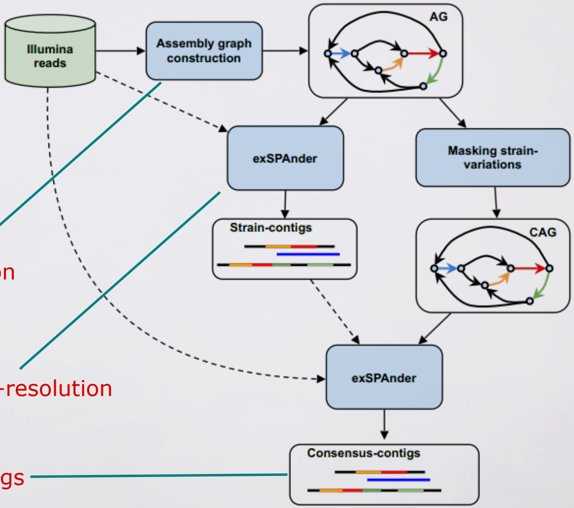

input reads R
count (k_{min}+1)-mer, output solid & mercy edges
$k \leftarrow k_{min}$
build SdBG of order k
remove tips; merge bubbles; progressively remove low local coverage edges; output contigs C_k
$k \leftarrow k + step$
$k \leq k_{max}$?
No
end
Yes
Extract (k+1)-mers from reads R and contigs C_{k-step}
graph construction
simplifications (errors & variants)
output contigs

Exhibit 2: (meta)SPAdes

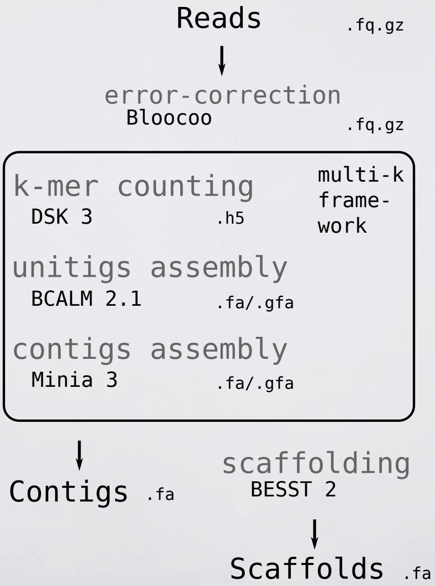


Graph construction & simplifications

Further repeat-resolution

Output contigs

Exhibit 3: the Minia pipeline



Assemblers are now mostly parameter-free

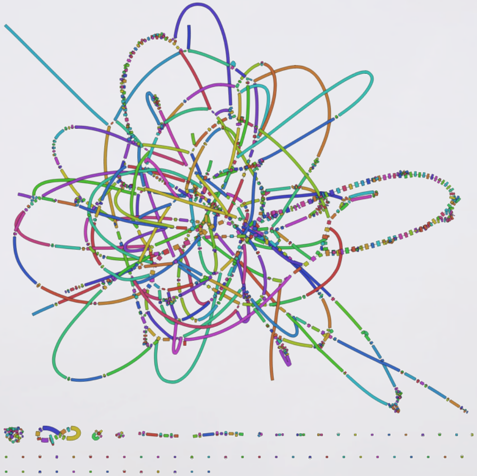
Used to need to choose and set a suitable *k*-mer size.

- **VelvetOptimizer** software
- **KmerGenie** software

.. but not anymore.

Effect of *k*-mer size

Salmonella genome, Velvet assembly, 100 bp Illumina reads.

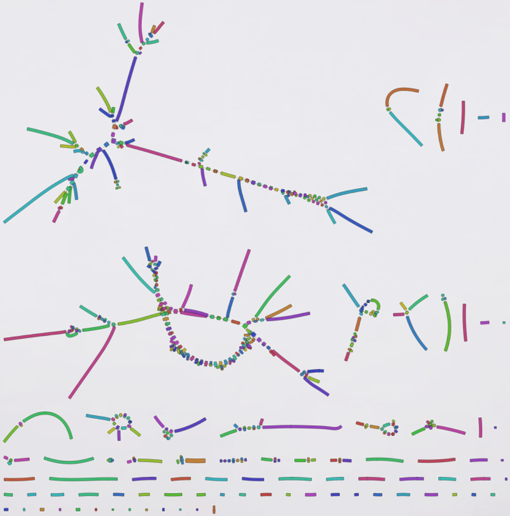


$k = 61$

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

Effect of *k*-mer size

Salmonella genome, Velvet assembly, 100 bp Illumina reads.

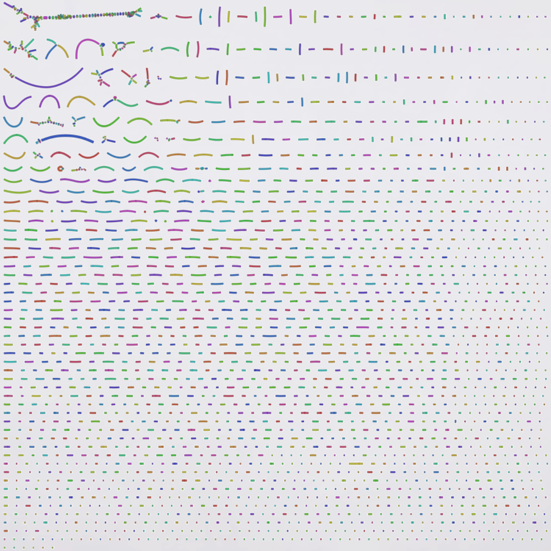


$k = 81$

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

Effect of *k*-mer size

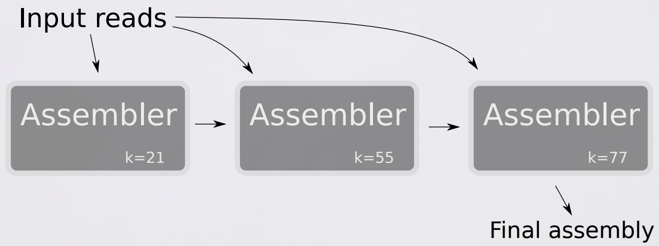
Salmonella genome, Velvet assembly, 100 bp Illumina reads.



k = 91

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

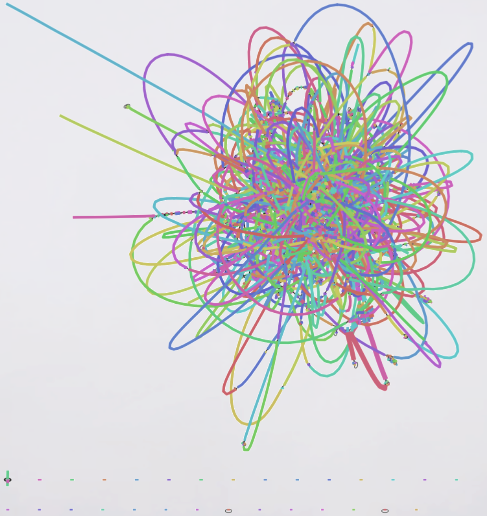
Multi-k



Introduced by [Peng *et al, RECOMB 2010*]

Visualization of multi-k graphs

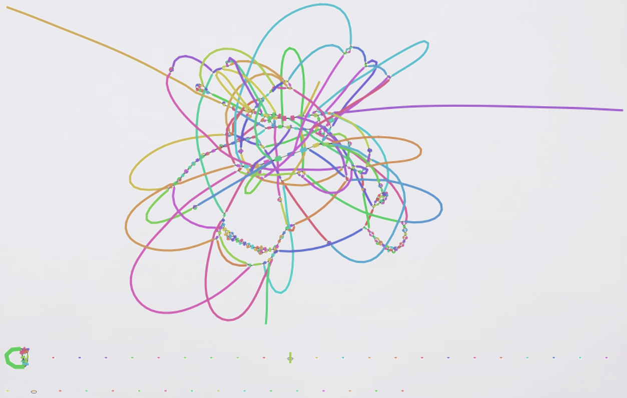
Salmonella genome, SPAdes assembly, MiSeq reads.



k = 21

Visualization of multi-k graphs

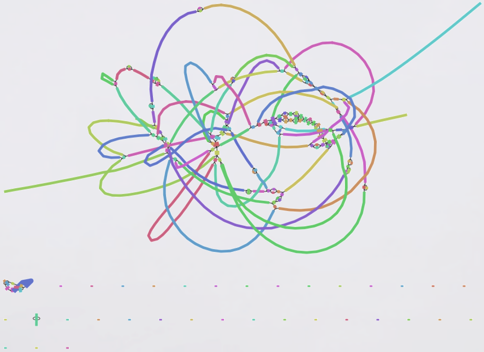
Salmonella genome, SPAdes assembly, MiSeq reads.



k = 55

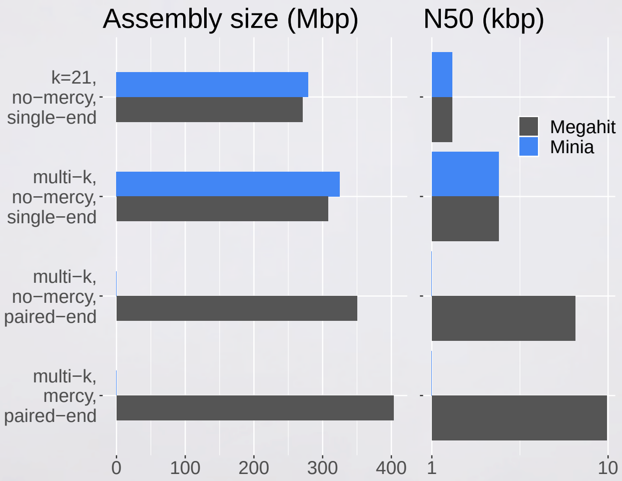
Visualization of multi-k graphs

Salmonella genome, SPAdes assembly, MiSeq reads.



k = 99
→ Still a single component, less repeat-induced complexity

Measuring the impact of multi-k



CAMI, medium dataset, PE data only

What’s next for short reads assembly?

-

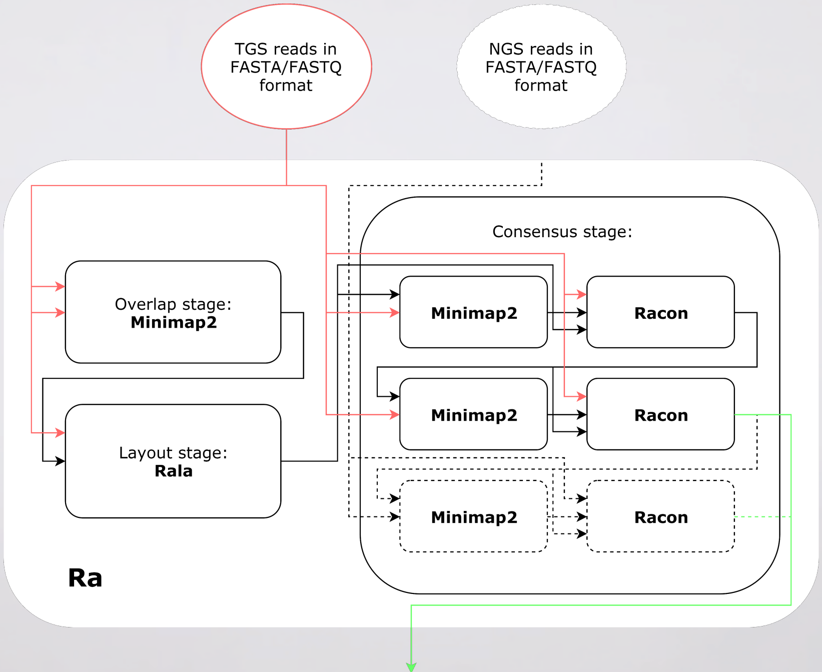
- Can *k*-**mer counting** be done **faster**? (than KMC3)
- Low-memory and even more scalable **dBG compaction**? (Bruno/BCALM2 hybrid)
- Fast **multi-k** (Can we do better than recomputing the whole assembly for each *k*?)
- Graph **simplifications** according to a Bayesian model or even ML.

Third generation assemblers

“First generation” of the 3rd generation

- Canu (Best Overlap Graph)
- Falcon, miniasm, MARVEL (overlap graphs)
- ABruijn
- Hinge
- Flye (2 2-column pages of graph description)

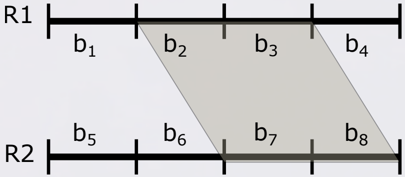
Ra


TGS reads in FASTA/FASTQ format
NGS reads in FASTA/FASTQ format
Overlap stage: **Minimap2**
Layout stage: **Rala**
Consensus stage:
Minimap2
Racon
Minimap2
Racon
Minimap2
Racon
Ra

Wtdbg2

(proposal to rename it to “Wutabaga 2”)

1) Reads are binned into 256bp bins



2) Alignments are found between read bins that share k-mers, using Smith-Waterman

3) Bases are forgotten, a "fuzzy" de Bruijn graph is constructed over the bins

$$b_6 b_7 b_8 \rightarrow b_2 b_3 b_4$$

Shasta (UCSC, LC'19)

- for Oxford Nanopore reads (and maybe also PacBio)
- human genome (60x) in 6 wall-clock hours (64 cores, 2 TB)

Techniques:

- homopolymer compression
- reads summarized as a sequence of "marker" 10-mers

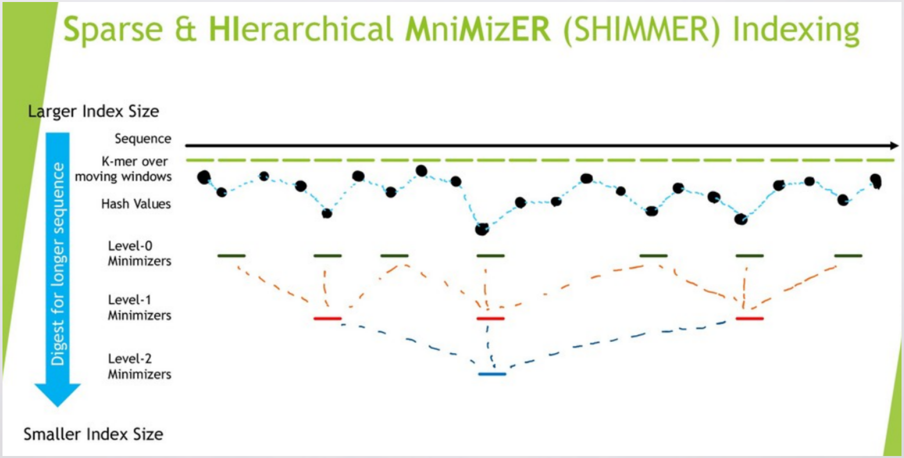
Assign IDs to only a few 10-mers: GCA=0, GAC=1, CGC=2.

```
read:       CGACACGTATGCGCACGCTGCGCTCTGCAGC
markers:     GAC        GCA            GCA
                       CGC      CGC
‘‘summarized’’ read: 1 2 0 2 0
```

Source: `https://chanzuckerberg.github.io/shasta/ComputationalMethods.html`

Peregrine (J. Chin, SFAF'19)

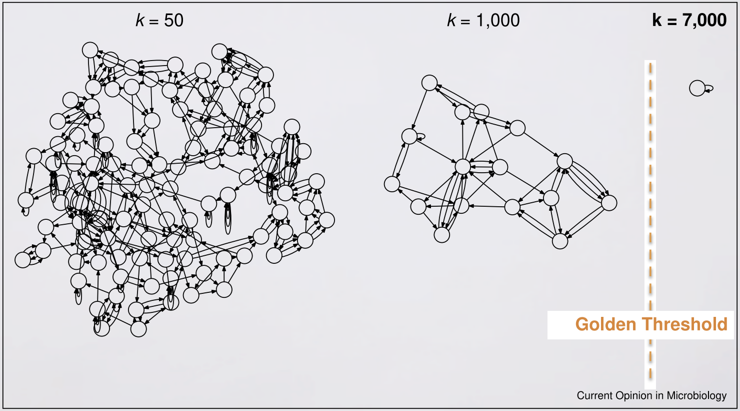
- **only** for accurate long reads: length > 10kb, accuracy > 99%
- read overlaps found by chaining minimizers
- human genome (30x cov) in 20 CPU hours



https://speakerdeck.com/jchin/assembling-human-genome-in-100-minutes

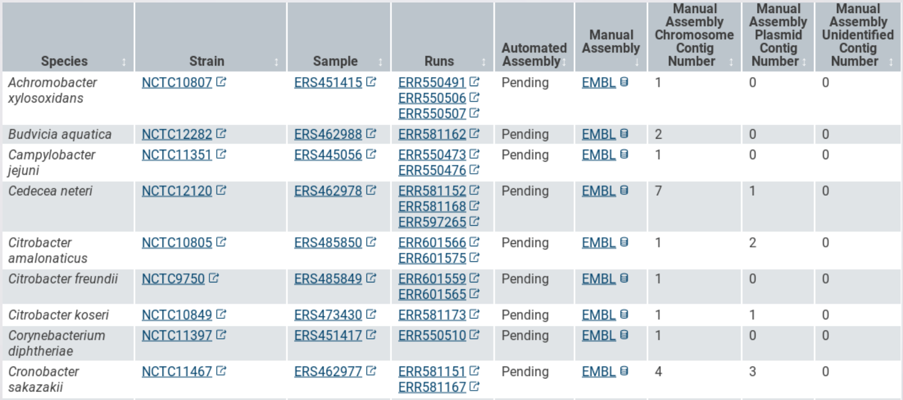
One chromosome = one contig?

Assembly graph of the *E. coli* genome [Koren 2015]:



Slides adapted from P. Marijon, RECOMB-Seq’19

NCTC 3000 database



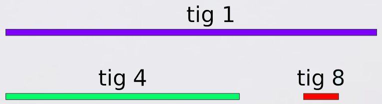
Species	Strain	Sample	Runs	Automated Assembly	Manual Assembly	Manual Assembly Chromosome Contig Number	Manual Assembly Plasmid Contig Number	Manual Assembly Unidentified Contig Number
Achromobacter xylosoxidans	NCTC10807	ERS451415	ERR550491 ERR550506 ERR550507	Pending	EMBL	1	0	0
Budvicia aquatica	NCTC12282	ERS462988	ERR581162	Pending	EMBL	2	0	0
Campylobacter jejuni	NCTC11351	ERS445056	ERR550473 ERR550476	Pending	EMBL	1	0	0
Cedecea neteri	NCTC12120	ERS462978	ERR581152 ERR581168 ERR597265	Pending	EMBL	7	1	0
Citrobacter amalonaticus	NCTC10805	ERS485850	ERR601566 ERR601575	Pending	EMBL	1	2	0
Citrobacter freundii	NCTC9750	ERS485849	ERR601559 ERR601565	Pending	EMBL	1	0	0
Citrobacter koseri	NCTC10849	ERS473430	ERR581173	Pending	EMBL	1	1	0
Corynebacterium diphtheriae	NCTC11397	ERS451417	ERR550510	Pending	EMBL	1	0	0
Cronobacter sakazakii	NCTC11467	ERS462977	ERR581151 ERR581167	Pending	EMBL	4	3	0

599 / 1136 (34 %) assemblies are not single-contig (Feb 2019)

Example (simulated)

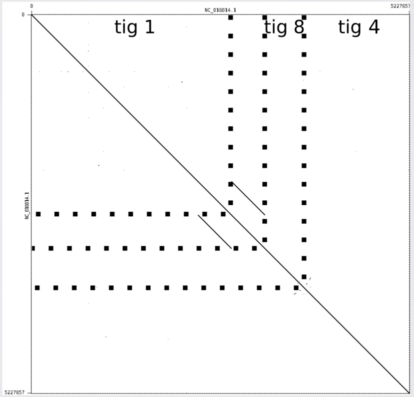
- **Dataset**: *T. roseus* (bacteria), simulated PacBio 20x
- **Assembly tools**: `Canu`

Resulting assembly graph:



Can we recover missing edges between contigs?

Not even a repetition problem..



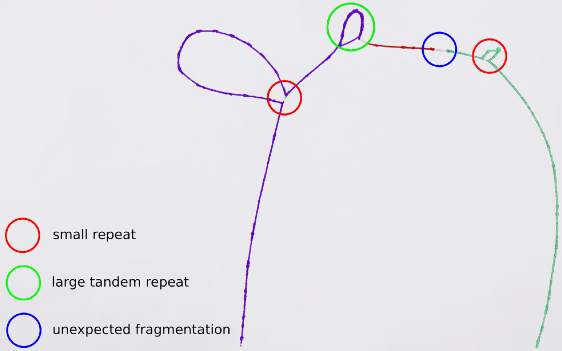
Dotplot of T. roseus genome against itself.

Genome has 460 kbp tandem repeat.
Repetition explains only 1 of the 2 contigs breaks.

Example (simulated)

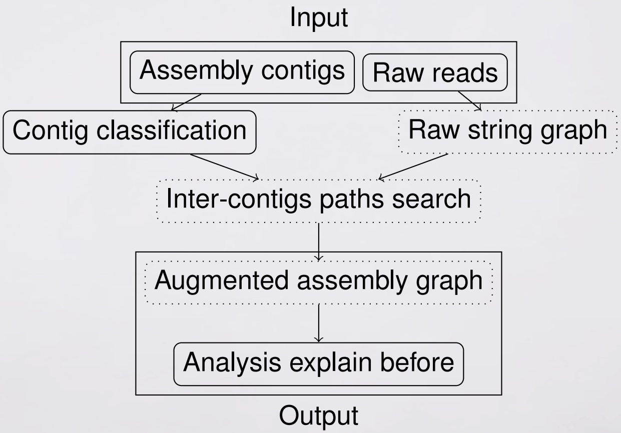
Let's have a look at the original overlap graph:

- nodes → reads
- edges → overlaps



Overlap graph (constructed by `Minimap2`), reads colored by `Canu` contig.

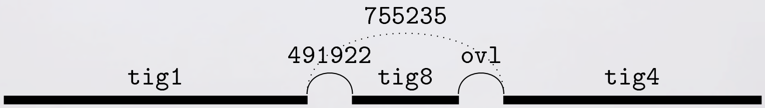
KNOT: Pipeline



The Augmented Assembly Graph

undirected, weighted graph:

- nodes: contigs extremities
- edges:
  - between extremities of a contig (weight = 0)
  - paths found between contigs (weight = path length in bp)

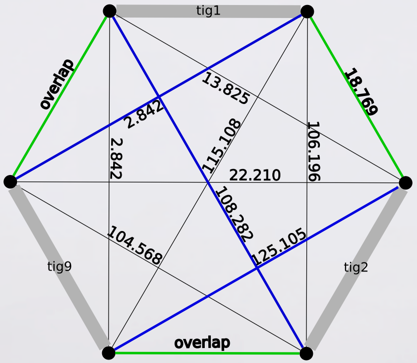


KNOT finds hidden connections between contigs

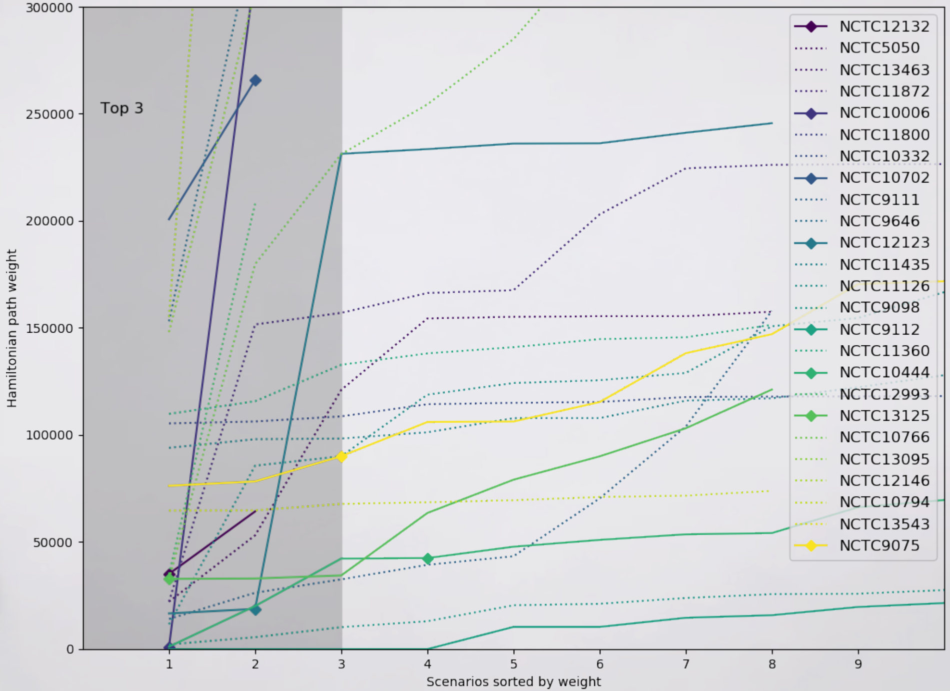
Across 38 datasets:

Mean number of	
`Canu` contigs	4.32
Dead-ends in `Canu` contig graph	4.94
Dead-ends in AAG	2.70

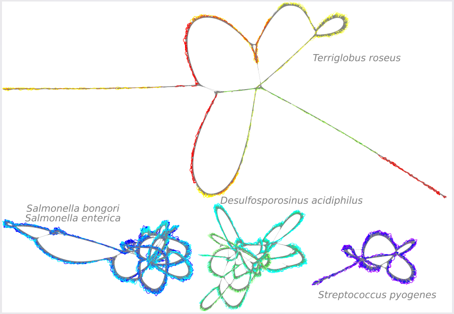
- AAG's are generally complete
- Hamilton walks can be **enumerated**
- Walk weight: sum of edges weights
- **lowest-weight walk** assumed to be the true genome



- 🟩 Green walk weight: 18,769 bases
- 🟦 Blue walk weight: 136,229 bases


Top 3
Hamiltonian path weight
Scenarios sorted by weight
0
50000
100000
150000
200000
250000
300000
1
2
3
4
5
6
7
8
9
NCTC12132
NCTC5050
NCTC13463
NCTC11872
NCTC10006
NCTC11800
NCTC10332
NCTC10702
NCTC9111
NCTC9646
NCTC12123
NCTC11435
NCTC11126
NCTC9098
NCTC9112
NCTC11360
NCTC10444
NCTC12993
NCTC13125
NCTC10766
NCTC13095
NCTC12146
NCTC10794
NCTC13543
NCTC9075

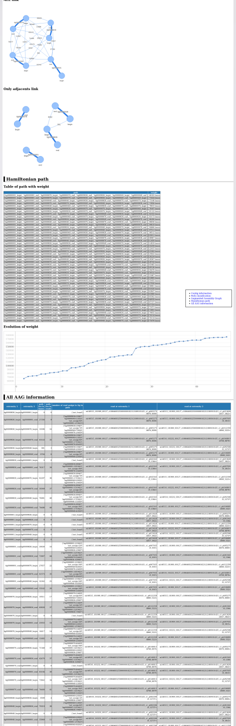
- Bacterial genome assembly **isn't fully solved**
- **Augmented Assembly Graphs** can help



`https://gitlab.inria.fr/pmarijon/knot`

`@pierre_marijon`

- Other analysis tool not based on graphs:
- `https://github.com/johnomics/tapestry`


Only adjacents link

Hamiltonian path

Table of path with weight

Evolution of weight

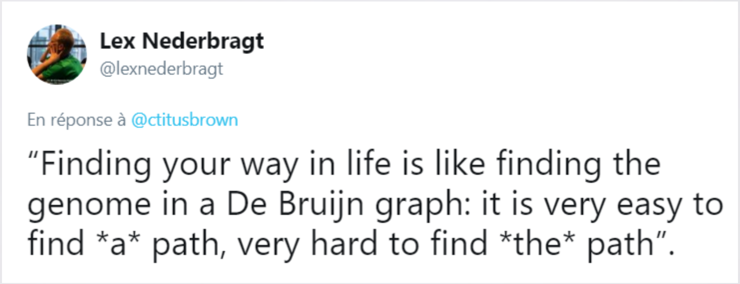
All AAG information

Questions that have been bugging me

-

- Can *k*-**mer counting** be done **faster** (than KMC3), keeping reasonable memory usage?
- Low-memory and scalable **dBG compaction**? (Bruno/BCALM2 hybrid)
- Fast **multi-k** (Can we do better than recomputing the whole assembly for each *k*?)
- High-**performance** & high-**quality** 3rd generation assembler ("fasterFlye", see recent benchmark from R. Wick)
- Can somehow the **marker graph** idea of Shasta be applied.. to *k*-mers?

Acknowledgments: Pierre Marijon, Guillaume Rizk, Antoine Limasset, Paul Medvedev, Claire Lemaitre, Pierre Peterlongo, Charles Deltel, Camille Marchet, Ryan Wick, Sergey Nurk, Kristoffer Sahlin, Lars Arvestad, Aaron Darling, Chris Quince, Dominique Lavenier



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".