Metagenome assembly methods

Rayan Chikhi

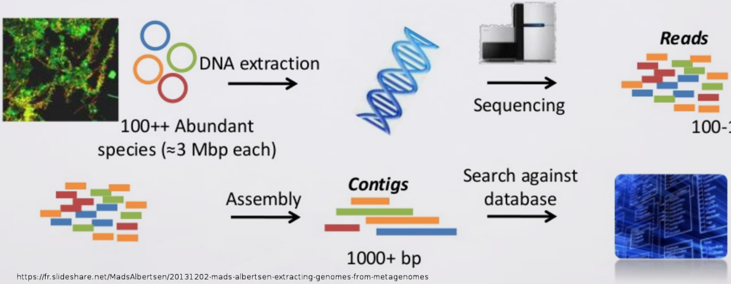
CNRS, Univ. Lille, France

CGSI, July 30th 2018

Slides @

Metagenomic assembly

Reconstruct genomes of species, possibly even strains, from short read sequencing data of an environment



no hits?

https://fr.slideshare.net/MadsAlbertsen/20131202-mads-albertsen-extracting-genomes-from-metagenomes

Challenges

1. closely related strains
2. uneven depths, & low depths
3. inter-species repeats
4. size of datasets
5. lack of long reads

(adapted from A. Korobeynikov's talk)

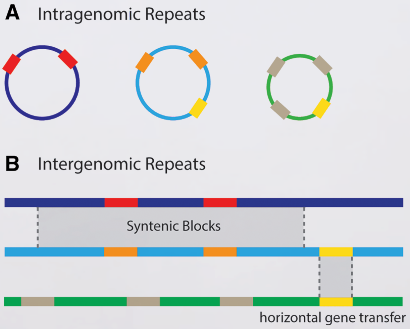


Fig: Olsen *et al, 2017*

What comes after assembly

Contigs binning

- CONCOCT
- MetaBAT
- MaxBin
- MetaWatt

Taxonomic identification

- PhyloPythiaS
- Kraken
- ProPhyle
- Centrifuge

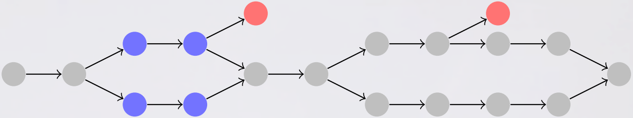
See anvi’o pipeline

Assembly software

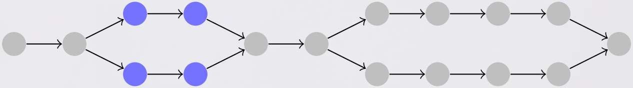
- **IDBA-UD**
- **metaSPAdes** [Nurk *et al, Genome Res., 2017*]
- **MEGAHIT** [Li *et al, Methods, 2016*]
- Minia-pipeline
- Ray-meta
- SOAPdenovo2
- metaVelvet/-SL
- Omega
- InteMAP
- Meraga
- Velour
- A*

How a metagenome assembler generally works

1) de Bruijn **graph** construction



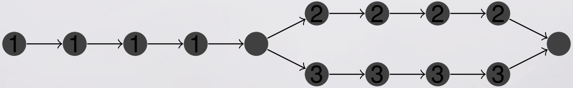
2) Likely sequencing errors are removed.



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

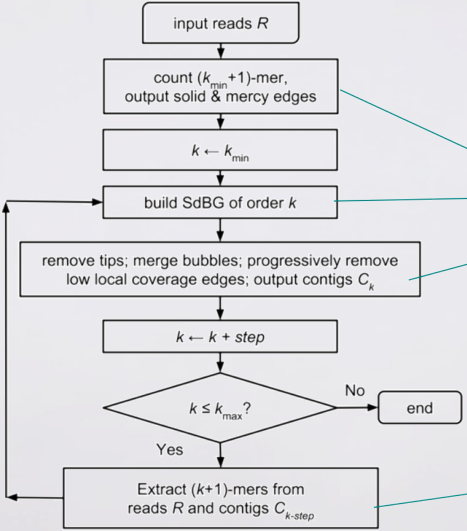
→ **Collapses strains**

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



5) Extra steps: repeat-resolving, scaffolding

MEGAHIT < v1.0

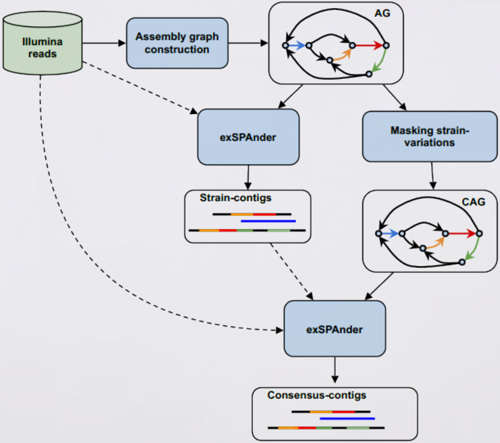

input reads R
count (k_{min}+1)-mer, output solid & mercy edges
$k \leftarrow k_{min}$
build SdBG of order k
remove tips; merge bubbles; progressively remove low local coverage edges; output contigs C_k
$k \leftarrow k + step$
$k \leq k_{max}$?
No
end
Yes
Extract (k+1)-mers from reads R and contigs C_{k-step}

graph construction

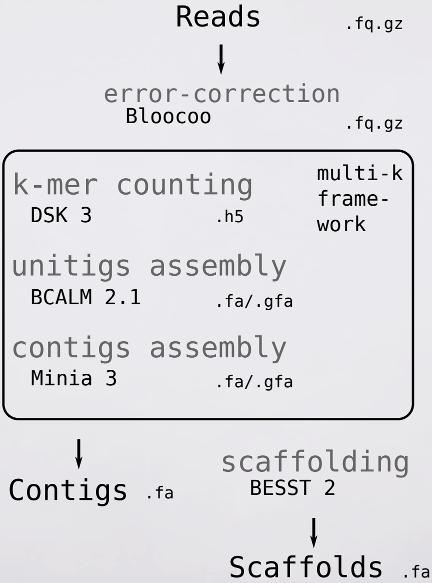
simplifications (errors & variants)

output contigs

metaSPAdes


Illumina reads
Assembly graph construction
AG
exSPAnder
Masking strain-variations
Strain-contigs
CAG
exSPAnder
Consensus-contigs

the Minia pipeline



Reads .fq.gz

error-correction
Bloocoo .fq.gz

k-mer counting
DSK 3 .h5

multi-k
frame-
work

unitigs assembly
BCALM 2.1 .fa/.gfa

contigs assembly
Minia 3 .fa/.gfa

Contigs .fa

scaffolding
BESST 2

Scaffolds .fa

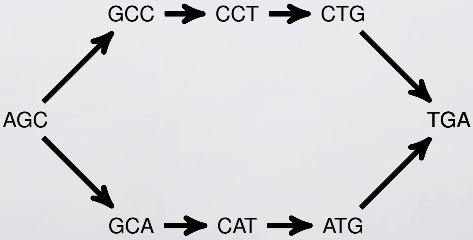
de Bruijn graphs

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

```
AGCCTGA
AGCATGA
```

dBG, $k = 3$:



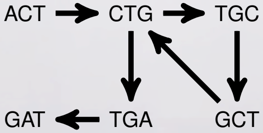
de Bruijn graphs

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

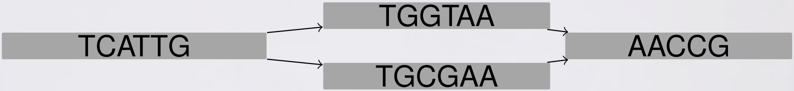
```
ACTG
 CTGC
  TGCT
   GCTG
    CTGA
     TGAT
```

dBG, $k = 3$:



Compacted de Bruijn graph

Compacted de Bruijn graph:

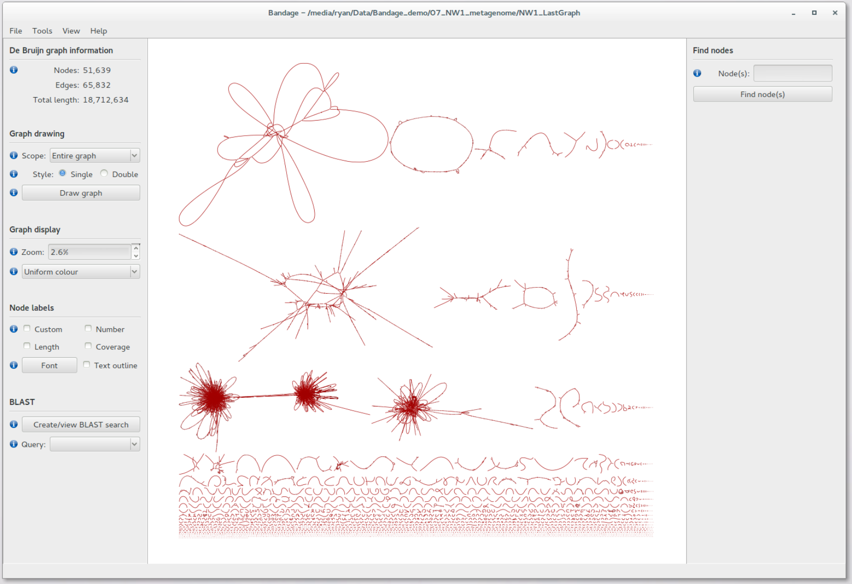


Each non-branching path becomes a single node (*unitig*).

- no loss of information
- less space

Downside: less easy to update

Large metagenome graph



Bandage – /media/ryan/Data/Bandage_demo/07_NW1_metagenome/NW1_LastGraph

De Bruijn graph information

Nodes: 51,639
Edges: 65,832
Total length: 18,712,634

Graph drawing

Scope: Entire graph

Style: ● Single ○ Double

Graph display

Zoom: 2.6%

Uniform colour

Node labels

□ Custom □ Number
□ Length □ Coverage
□ Text outline

BLAST

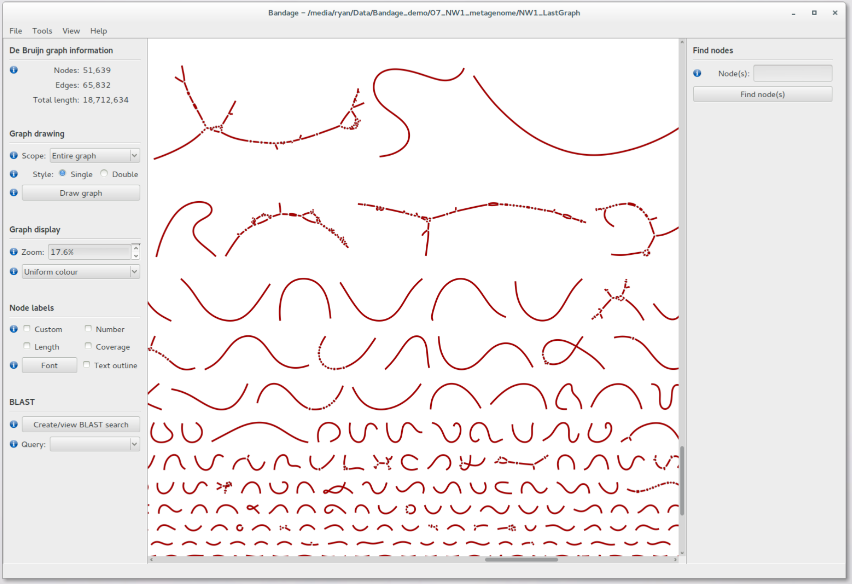
Query:

Find nodes

Node(s):

Source: Bandage wiki

Large metagenome graph (zoom)



Bandage – /media/ryan/Data/Bandage_demo/07_NW1_metagenome/NW1_LastGraph

De Bruijn graph information

Nodes: 51,639
Edges: 65,832
Total length: 18,712,634

Graph drawing

Scope: Entire graph

Style: ● Single ○ Double

Graph display

Zoom: 17.6%

Uniform colour

Node labels

□ Custom □ Number
□ Length □ Coverage
□ Text outline

BLAST

Query:

Find nodes

Node(s):

Source: Bandage wiki

Under the hood of metagenome assemblers

Under the hood of metagenome assemblers

Multi-k, variant/error removal, low-abundance rescue

Effect of *k*-mer size

Salmonella genome, Velvet assembly, 100 bp Illumina reads. $k = 51$

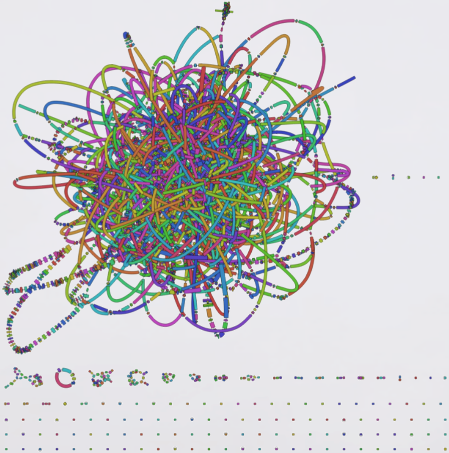
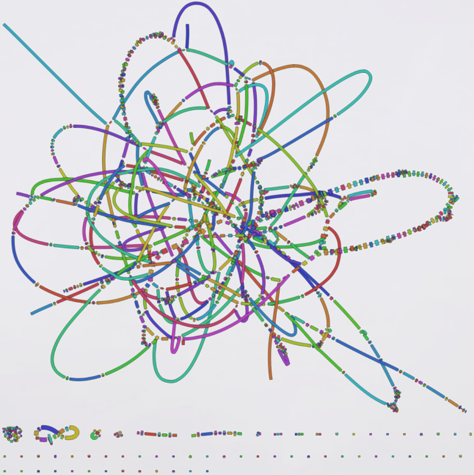


Fig: https://github.com/rrwick/Bandage/wiki/Effect-of-kmer-size

Effect of *k*-mer size

Salmonella genome, Velvet assembly, 100 bp Illumina reads.

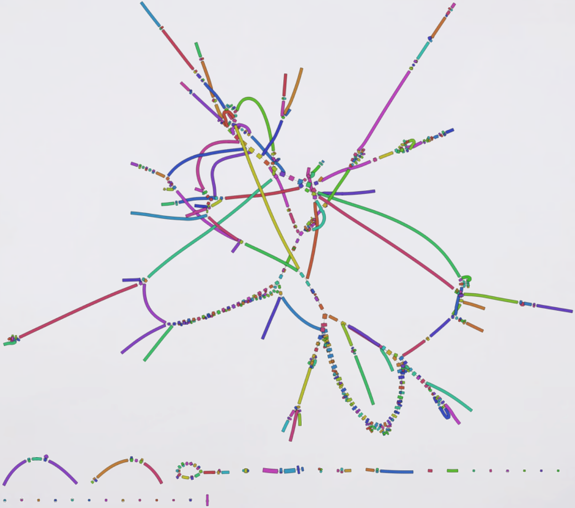


$k = 61$

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

Effect of *k*-mer size

Salmonella genome, Velvet assembly, 100 bp Illumina reads.

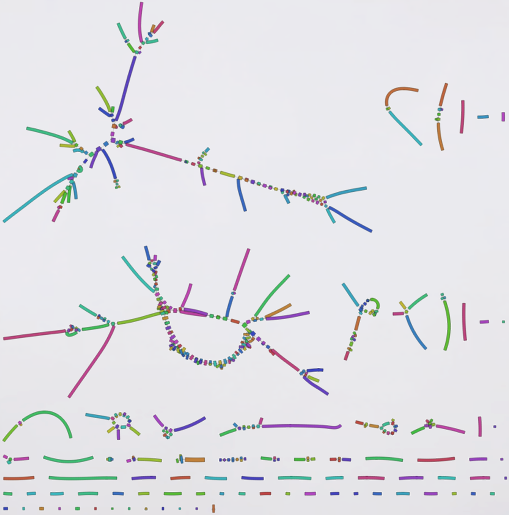


k = 71

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

Effect of *k*-mer size

Salmonella genome, Velvet assembly, 100 bp Illumina reads.

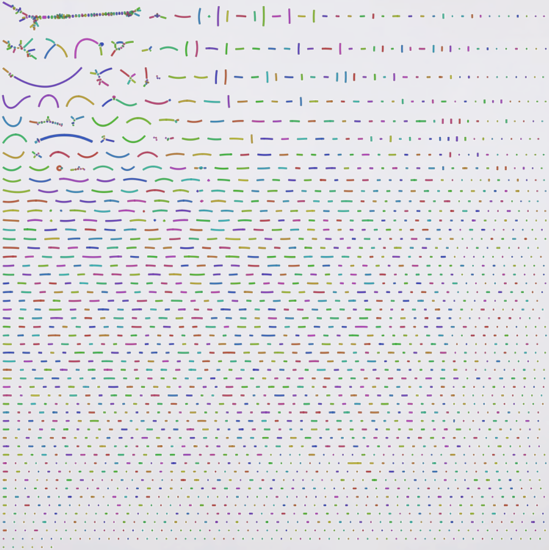


$k = 81$

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

Effect of *k*-mer size

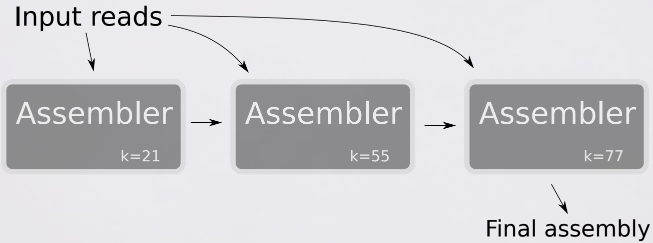
Salmonella genome, Velvet assembly, 100 bp Illumina reads.



k = 91

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

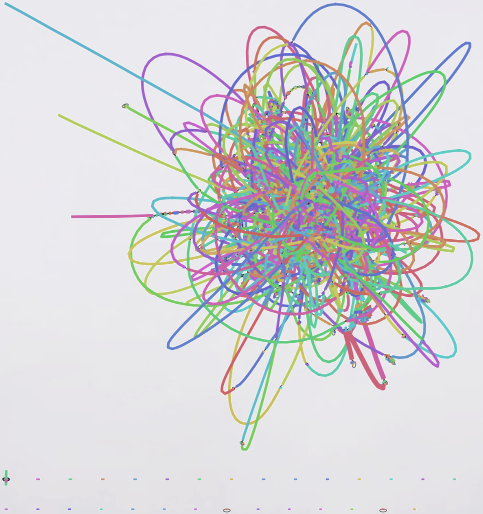
Multi-k



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

Visualization of multi-k graphs

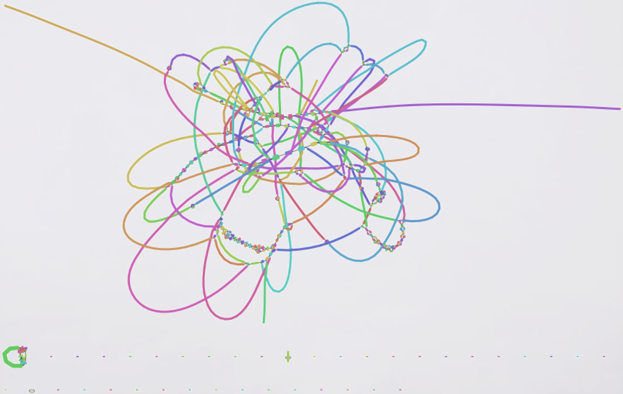
Salmonella genome, SPAdes assembly, MiSeq reads.



k = 21

Visualization of multi-k graphs

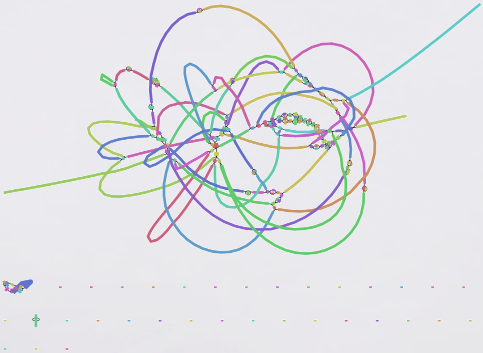
Salmonella genome, SPAdes assembly, MiSeq reads.



k = 55

Visualization of multi-k graphs

Salmonella genome, SPAdes assembly, MiSeq reads.

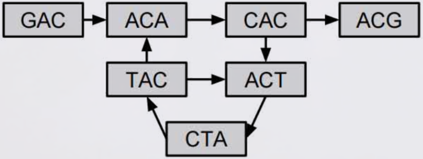


k = 99

→ Still a single component, less repeat-induced complexity

Why is MEGAHIT so fast

- In-memory read indexing, implicit *k*-mer counting
- succinct DBG, carefully engineered construction



An edge-based DBG with *k*=3;

Edges = {GACA, ACAC, CACG, CACT, ACTA, CTAC, TACA, TACT}

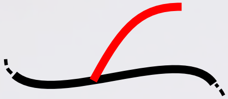
	F
A	0
C	3
G	8
T	9

Tip	Last		W
0	1	ACA	C
1	0	$GA	C
0	1	CTA	C
0	0	CAC	G
0	1	CAC	T
0	1	GAC	A
0	0	TAC	A-
0	1	TAC	T-
0	1	ACG	$
0	1	ACT	A

Dummy Edges = { GAC, ACG}

Fig: Li *et al, 2016*

Graph simplifications (here, SPAdes-inspired)



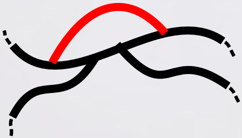
Tip removal:

$\mathrm{len}_{tip} \le 3.5k$

or

$\mathrm{len}_{tip} \le 10k$

$2\mathrm{cov}_{tip} \le \mathrm{cov}_{neighbors}$



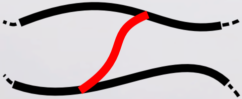
Bulge removal:

$\mathrm{len}_{bulge} \le \max(3k, 100)$

$\mathrm{cov}_{bulge} \le 1.1\mathrm{cov}_{altpath}$

$\mathrm{len}_{altpath} = \mathrm{len}_{bulge} \pm \mathrm{delta}$

$\mathrm{delta} = \max(0.1\mathrm{len}_{bulge}, 3)$



Erroneous connection removal:

$\mathrm{len}_{EC} \le 10k$

$4\mathrm{cov}_{EC} \le \mathrm{cov}_{neighbors}$

Dealing with a flood of erroneous *k*-mers

. . . and keeping low-coverage, good *k*-mers.

The MEGAHIT way: abundance cut-off at 2, *mercy* *k*-mers

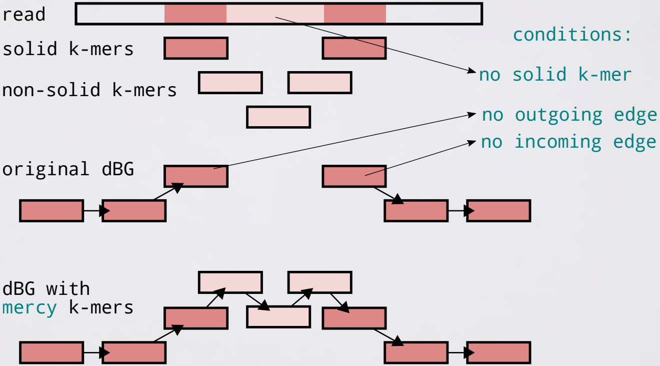
The SPAdes way: abundance cut-off at 1, pre-simplifications prior to graph construction

Alternatives:

1. stand-alone fixed-memory tip clipping software @ `github.com/Malfoy/BTRIM`
2. stand-alone mercy *k*-mers module @ `github.com/GATB/minia`
3. pre-tip cleaning in minimizer-partitioned dBG construction

   spoilers: not very effective

Mercy *k*-mers



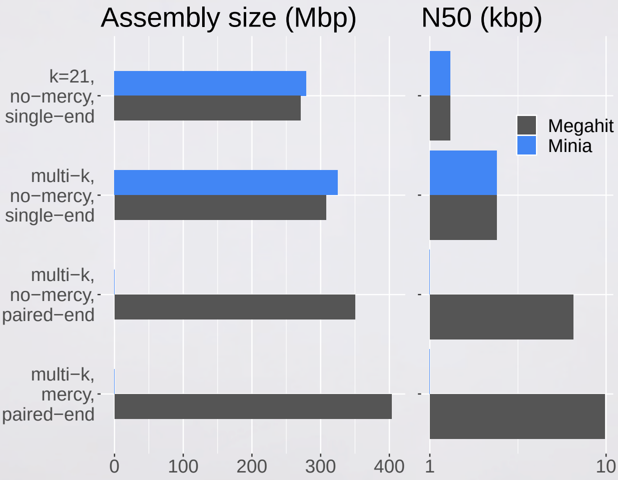
- Recovers filtered-out *k*-mers
- Useful for low-coverage strains.

Metagenomic scaffolding

Same as genome scaffolding, except: contigs may be placed in multiple scaffolds.

- no good stand-alone metagenomic scaffolder
- ‘repeat-resolution’ in metaSPADES
- ‘local assembly’ in MEGAHIT

Dissection of MEGAHIT modules

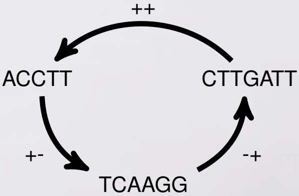


CAMI, medium dataset, PE data only

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
```



Handling reverse complements

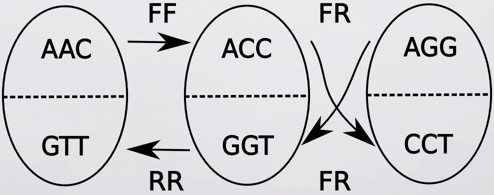
Due to strand ambiguity in sequencing:

> *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



Evaluation of assembly quality

Evaluation metrics

Same as regular assembly:

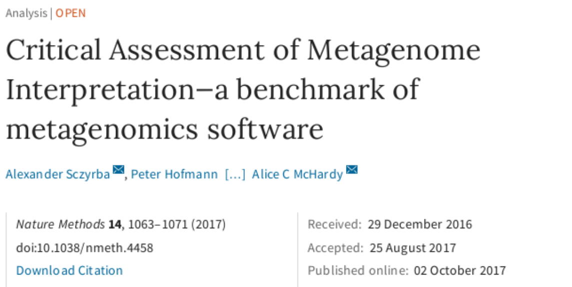
- N50, NG50
- Total size
- % of reads mapping correctly back to the assembly
- Number of predicted genes
- % of contigs matching some known references

Metagenome-specific:

- metaQUAST
- CheckM, marker genes, [Parks *et al, Genome Res. 2015*]
- VALET [Olson *et al, BFB 2017*]

CAMI benchmark

- 3 artificial communities
  - low, medium, high complexity (600 genomes, 5x15 Gbp)
- 6 assemblers evaluated: MEGAHIT, Minia, Ray-meta, ..



Analysis | OPEN

Critical Assessment of Metagenome Interpretation—a benchmark of metagenomics software

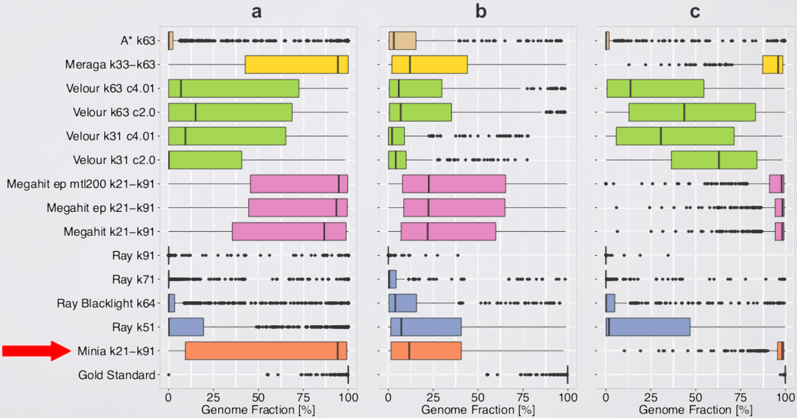
Alexander Sczyrba, Peter Hofmann [...] Alice C McHardy

Nature Methods **14**, 1063–1071 (2017)
doi:10.1038/nmeth.4458
Download Citation

Received: 29 December 2016
Accepted: 25 August 2017
Published online: 02 October 2017

Quality of metagenome assembly

a: all genomes, b: genomes with ANI >= 95%, c: genomes with ANI < 95%



[Sczyrba, Nat Meth 2018]

Minia 6x less mem than MEGAHIT, as fast.
MetaSPAdes: dataset too large.
No assembler could reconstruct **close strains** (ongoing work).

Mosaic DNANexus Challenge 2018

Focus on **strains** assembly

mosaic

Evaluation metrics:

- Genome Fraction
- misassemblies

Mosaic DNANexus Challenge 2018

Focus on **strains** assembly

mosaic

Evaluation metrics:

- Genome Fraction
- misassemblies

Minia’s entry:

Method	**N50**	**Genome Fraction**	**# misassemblies**
Unitigs (BCALM)	0.5 Kbp	95.3%	23
Minia-pipeline only tip clipping	1.3 Kbp	90.8%	286
Minia-pipeline with all simplifications	7.1 Kbp	84.1%	1998

→ **Evaluating** metagenome assemblies is hard

Conclusion

- Metagenome assembly is a hard problem
- Due to strains & low-abundant species, mostly
- Strains: trade-off between contiguity, and genome fraction/misassemblies. Questions on assemblies ranking.
- So far, limited availability of: long reads, Hi-C, 10x Genomics (?)

References:

- `https://github.com/GATB/minia-pipeline`
- CAMI - A Benchmark of Metagenomics Software, 2017
- MEGAHIT & metaSPAdes articles

Acknowledgments: Sergey Nurk, Chris Quince, Aaron Darling, Guillaume Rizk, Claire Lemaitre, Pierre Peterlongo, Charles Deltel, Antoine Limasset, Paul Medvedev, Dominique Lavenier

Postdoc position in France, 2019

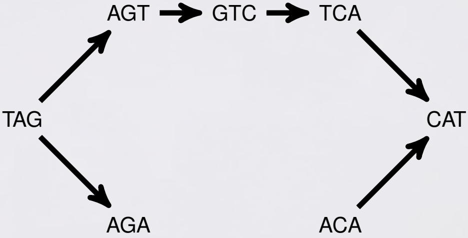
Exercise

k-mers:

1. `ACA`
2. `AGA`
3. `AGT`
4. `CAT`
5. `GTC`
6. `TAG`
7. `TCA`
8. `TTG`

Two strains of a short genome are in this dataset, please assemble them. ignore reverse-complements

Exercice: solution



- Discard TTG (connected to nothing)
- Observe a *k*-mer was missing (GAC)
- Two strains: TAGTCAT, TAGACAT