Short-read metagenomics assembly methods

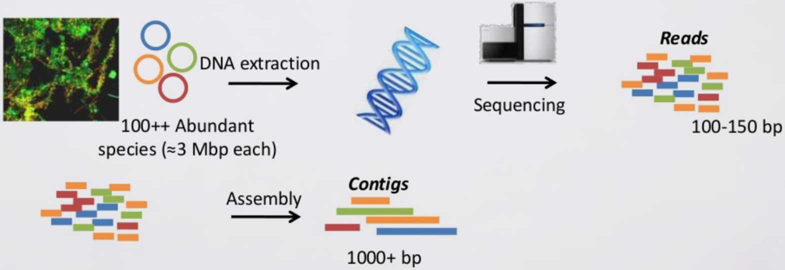
Rayan Chikhi

Institut Pasteur

EBAME6, Oct 2021

Metagenome assembly

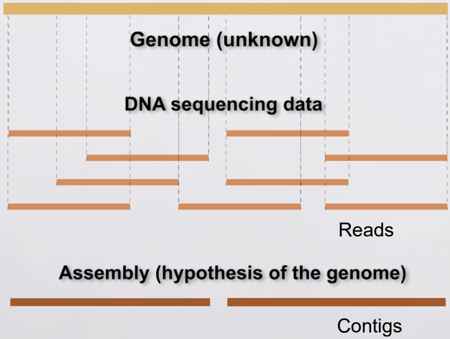
Reconstruct genomes of species, possibly even strains, from short read sequencing data of an environment



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

44 years of genome assembly

- **1977**: First complete genome assembled (phi X 174)
- **2003**: Human Genome Project completed
- **2014**: First $1,000 genome
- **2021**: Truly completed (Telomere-2-Telomere)



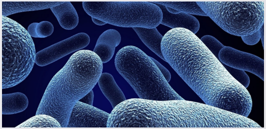
Genome (unknown)

DNA sequencing data

Reads

Assembly (hypothesis of the genome)

Contigs



Additional 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)

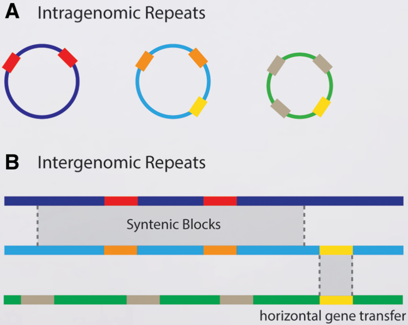
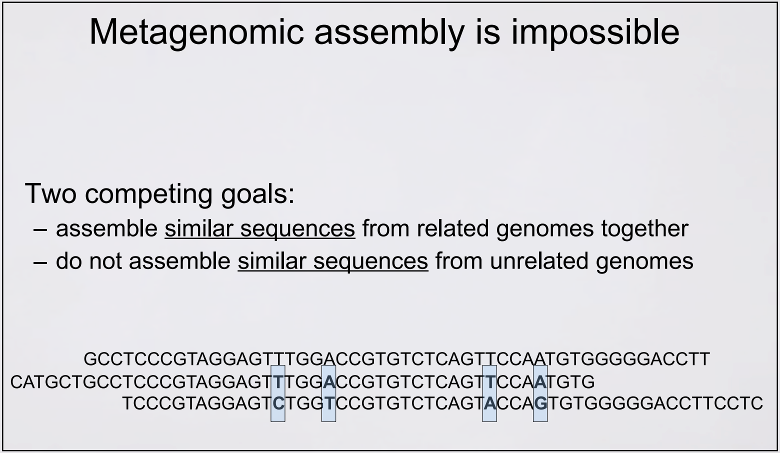


Fig: Olsen *et al, 2017*



Metagenomic assembly is impossible

Two competing goals:

- assemble similar sequences from related genomes together
- do not assemble similar sequences from unrelated genomes

```
      GCCTCCCGTAGGAGTTTGGACCGTGTCTCAGTTCCAATGTGGGGGACCTT
CATGCTGCCTCCCGTAGGAGTTTGGACCGTGTCTCAGTTCCAATGTG
         TCCCGTAGGAGTCTGGTCCGTGTCTCAGTACCAGTGTGGGGGACCTTCCTC
```

Mihai Pop, Sergey Koren, Dan Sommer

Slide credit: H. Touzet

Metagenome assembly software

- **metaSPAdes** [Nurk *et al, Genome Res., 2017*]
- **MEGAHIT** [Li *et al, Methods, 2016*]
- **metaFlye** (LR) [Kolmogorov *et al, bioRxiv, 2019*]
- Minia-pipeline [me!]
- IDBA-UD
- Ray-meta
- SOAPdenovo2
- metaVelvet/-SL
- Omega
- InteMAP
- Meraga
- Velour
- A*

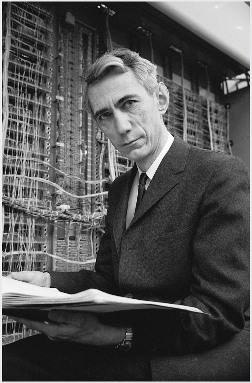
Under the hood of metagenome assemblers

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 shortest sentence containing all *k*-mers exactly once

[2] predict future data given past data, where past = last seen *k*-mer

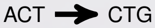
de Bruijn graphs

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

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

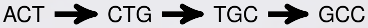
```
ACTG
```



de Bruijn graph

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

```
ACTG
 CTGC
  TGCC
```



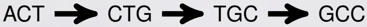
ACT → CTG → TGC → GCC

de Bruijn graph: redundancy

What happens if we add redundancy?

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

dBG, $k = 3$:

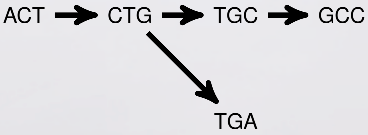


de Bruijn graph: errors

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

```
ACTG
 CTGC
 CTGA
  TGCC
```

dBG, $k = 3$:

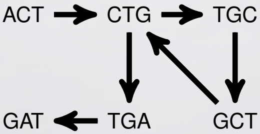


de Bruijn graph: repeats

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

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

dBG, $k = 3$:



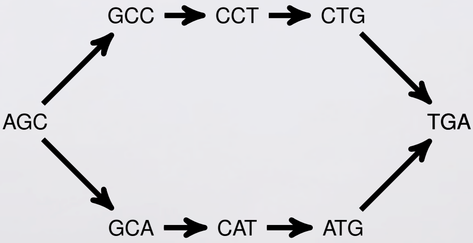
de Bruijn graph: SNPs

SNPs can be directly “found” in the graph.

`AGCCTGA`

`AGCATGA`

dBG, k = 3:



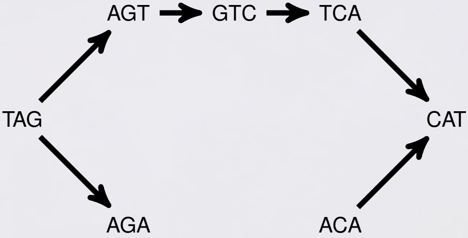
Exercice

Imagine you are a genome assembly software that converted reads into these *k*-mers:

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

They correspond to two strains of a short genome, please assemble those k-mers. Warning: one k-mer could be missing due to low coverage. ignore reverse-complements

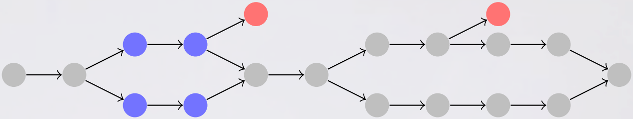
Exercice: solution



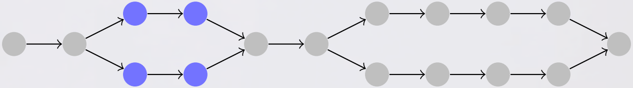
- Discard TTG (connected to nothing)
- Observe a *k*-mer was missing (GAC)
- Two strains: TAGTCAT, TAGACAT

Short read assemblers

1) de Bruijn **graph** construction



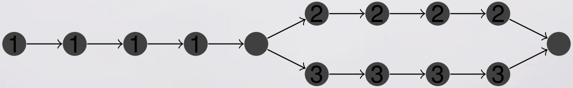
2) Likely sequencing errors are removed.



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

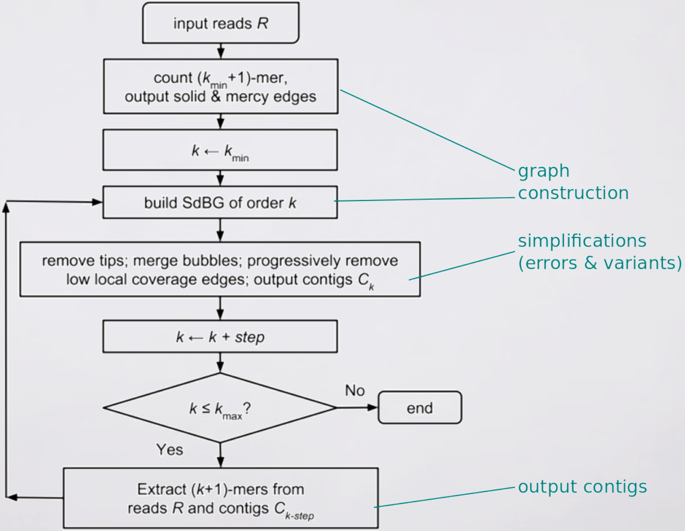
→ **Collapses strains**

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

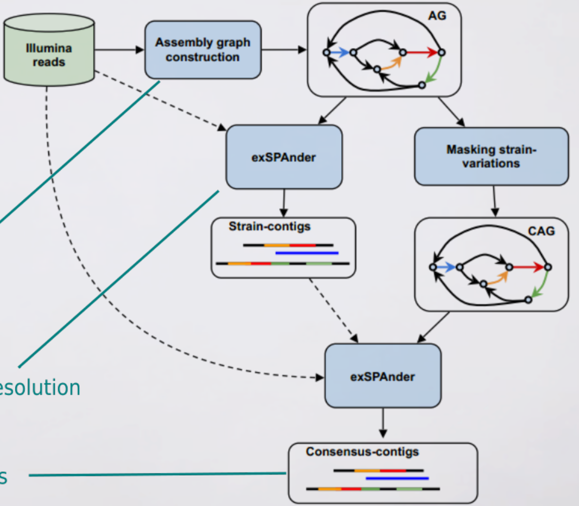


5) Extra steps: repeat-resolving, scaffolding

MEGAHIT


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 $C_{k\text{-}step}$
graph construction
simplifications (errors & variants)
output contigs

metaSPAdes



Graph construction & simplifications

Further repeat-resolution

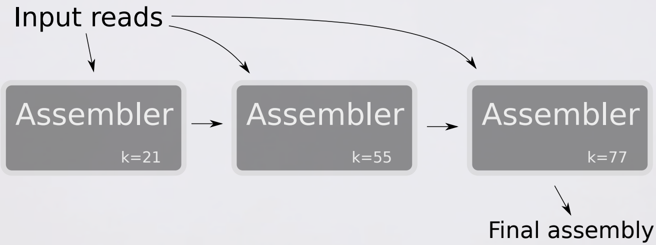
Output contigs

Short read assemblers

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

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

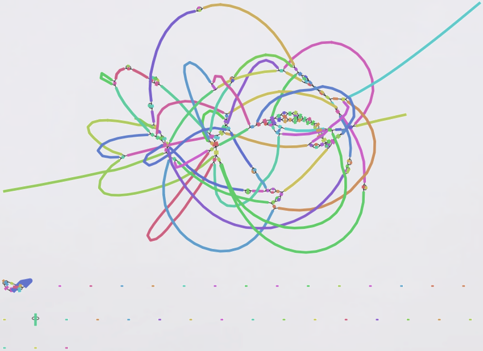
Multi-k



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

Visualization of multi-k graphs

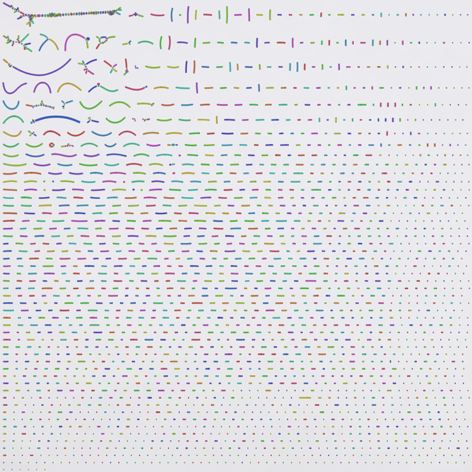
Salmonella genome, SPAdes assembly



k = 99

In contrast, with single-k

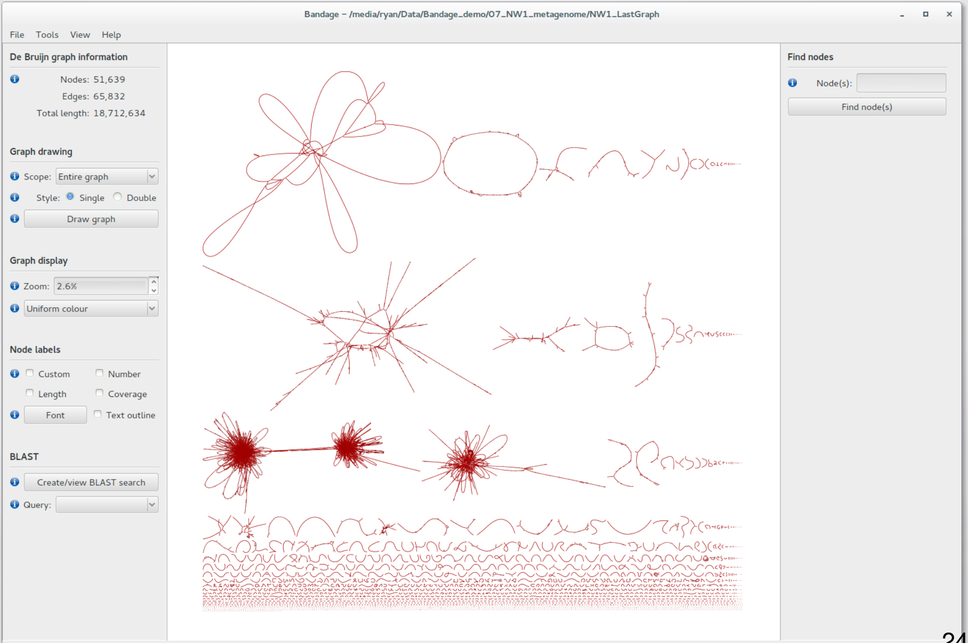
Salmonella genome, Velvet assembly



k = 91 (too high, but shown for comparison)

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

Assembly graph visualization: Bandage



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):

Metagenomics with long reads

Higher contiguity, higher quality. Use whenever possible.

1. metaFlye [Kolmogorov *et al, 2019*]
2. wtdbg2 [Nicholls *et al, GigaScience, 2019*]
3. Canu [see wtdbg2 article]
4. miniasm + Racon

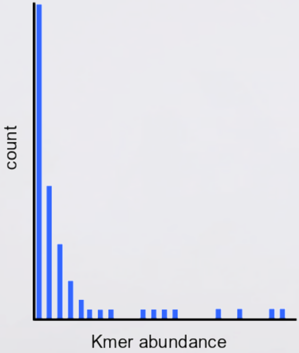
(See the Strainberry talk next week!)

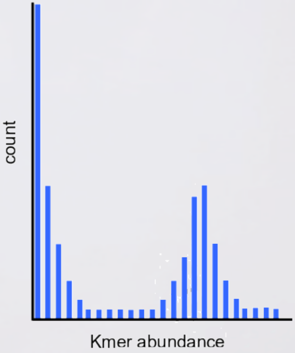
Oxford Nanopore: **needs polishing**

Hi-C

When *can* you assemble

Look at *k*-mer histograms of the reads. (KMC, DSK, Jellyfish)





Credit: `www.cmbi.ru.nl/~dutilh/metagenomics/course_HAN_2014/Speth.pdf`

Why you need ≥ 30x coverage per genome

Probability that a base is not covered: $e^{-coverage}$
(Lander-Waterman)

coverage	probability
5	0.007
10	0.000045
15	3*10-7
20	2*10-9
25	1.4*10-11
30	9.4*10-14
...	
100	3.7*10-44

Dealing with high coverage: Digital Normalization

`https://github.com/dib-lab/khmer`

👍

- Reduces dataset size
- Facilitates assembly

👎

- assembly fragmentation, maybe
- loss of low-coverage variants

Why you shouldn't use digital normalization

`http://ivory.idyll.org/blog/`
`why-you-shouldnt-use-diginorm.html`

Evaluation metrics

Same as regular assembly:

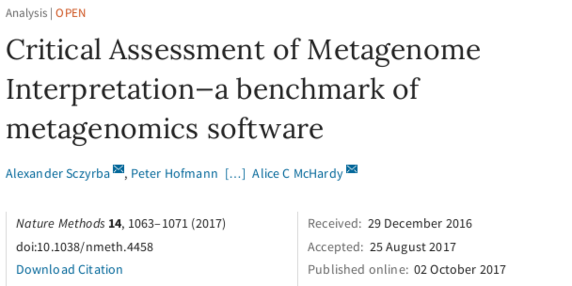
- N50, NG50
- Total size
- % of reads mapping correctly back to the assembly
- Number of predicted genes
- % of contigs matching some known references

Metagenome-specific:

- metaQUAST
- CheckM, marker genes, [Parks *et al, Genome Res. 2015*]
- VALET, internal consistency, [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

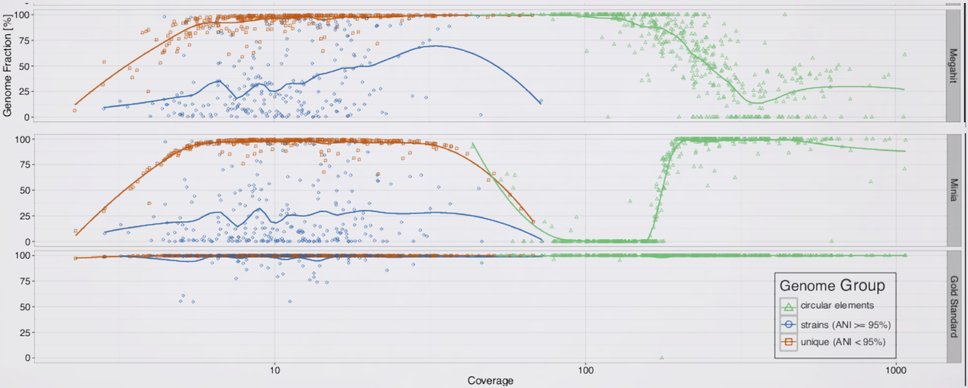
Alexander Sczyrba ✉, Peter Hofmann [...] Alice C McHardy ✉

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

→ CAMI2 paper out recently!

Metagenome assemblies vs coverage

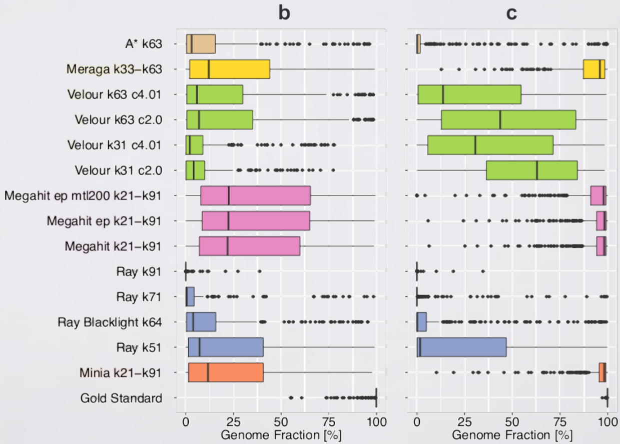


[Sczyrba, Nat Meth 2018]

Too low coverage? won't reconstruct.
Too high coverage? won't reconstruct.
Close strains? won't reconstruct.

Quality of metagenome assembly

b: genomes with **ANI >= 95 % (strains)**,   c: genomes with **ANI < 95**%



[Sczyrba, Nat Meth 2018]

For different species: Meraga, Megahit, Minia did well.
No assembler could reconstruct **close strains**.
metaSPAdes is great but couldn't process this dataset.

Mosaic DNANexus Challenge 2018

Focus on **strains** assembly

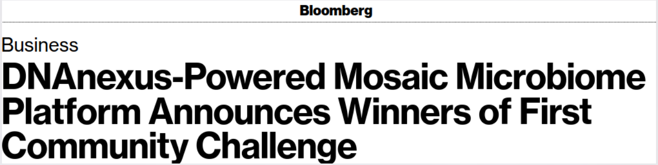
mosaic

Evaluation metrics:

- Genome Fraction
- misassemblies

Method	**N50**	**Genome Fraction**	**# misassemblies**
A regular assembler	7.1 Kbp	84.1%	1998
Initial step (BCALM)	0.5 Kbp	**95.3%**	**23**

(S. Nurk:) don't do it



Bloomberg

Business

DNAnexus-Powered Mosaic Microbiome Platform Announces Winners of First Community Challenge

→ even **evaluating** metagenome assemblies is hard

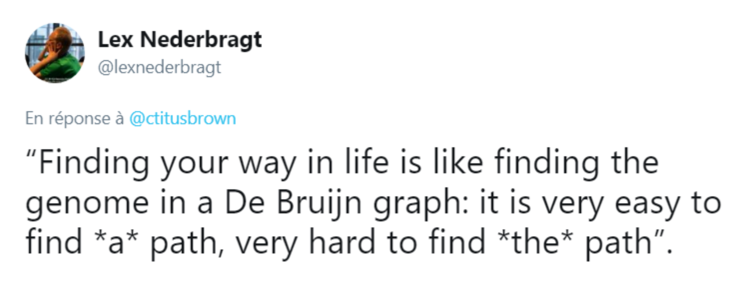
Conclusion

- Metagenome assembly is a hard problem
- Due to strains & low-abundance species, mostly
- Trade-off between contiguity, and genome fraction/misassemblies. Questions on assemblies ranking.
- So far, limited availability of: long reads, Hi-C, linked-reads
- out of RAM? `https://github.com/GATB/minia-pipeline`
- HiFi reads? let's chat about minimizer-space dBG

A reference:

- Ayling *et al*, New approaches for metagenome assembly with short reads, 2019

Acknowledgments: Dag Ahren, Sergey Nurk, Camille Marchet, Antoine Limasset, the fantastic team of the Workshop on Genomics 2020, Chris Quince, Aaron Darling, Guillaume Rizk, Claire Lemaitre, Pierre Peterlongo, Charles Deltel, Paul Medvedev, Dominique Lavenier

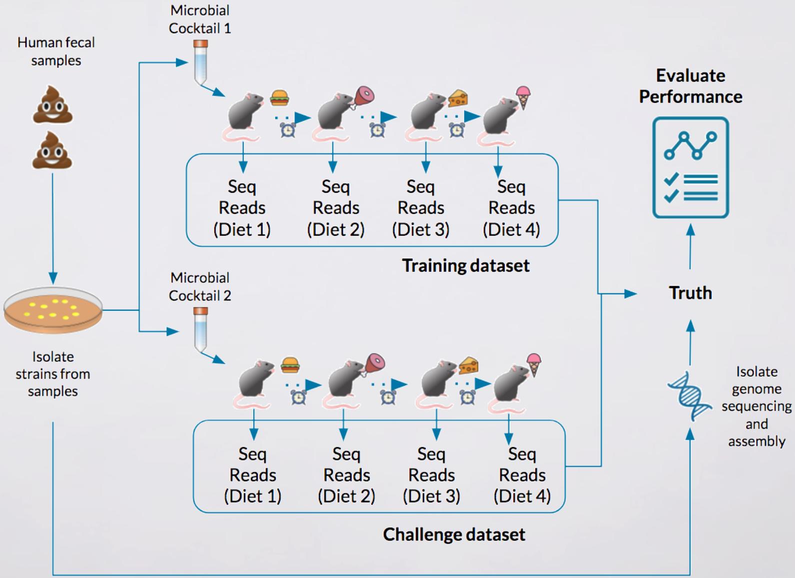


Lex Nederbragt
@lexnederbragt

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”.

Mosaic DNANexus Challenge 2018


Human fecal samples
Microbial Cocktail 1
Seq Reads (Diet 1)
Seq Reads (Diet 2)
Seq Reads (Diet 3)
Seq Reads (Diet 4)
Training dataset
Evaluate Performance
Truth
Isolate strains from samples
Microbial Cocktail 2
Seq Reads (Diet 1)
Seq Reads (Diet 2)
Seq Reads (Diet 3)
Seq Reads (Diet 4)
Challenge dataset
Isolate genome sequencing and assembly