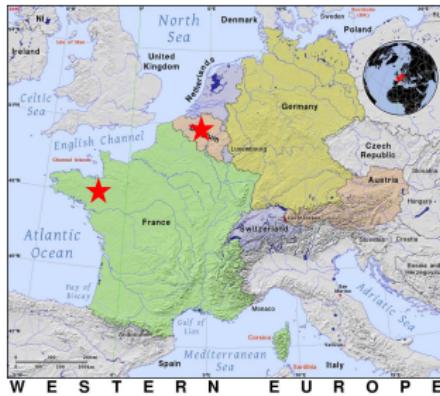


# About me: Ph.D., 2021-2024



**Jean-François Flot**

*Université Libre de Bruxelles*

Focus: assembling wild genomes



**Dominique Laveneir**

*Université de Rennes*

Focus: computational methods

## Metagenome assembly from long reads

# About me: postdoc, since 2025



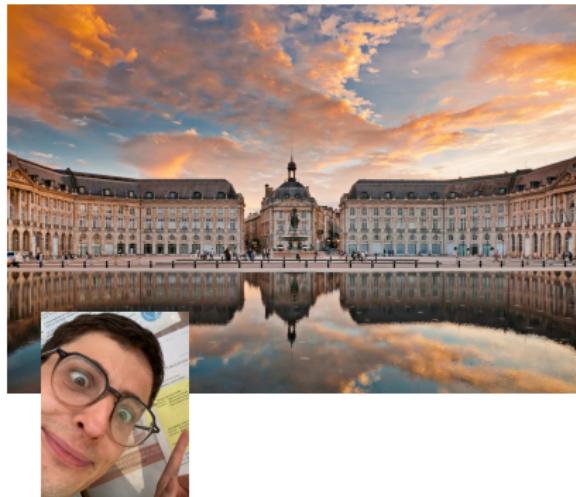
Institut Pasteur, Paris



**Rayan Chikhi**  
*Institut Pasteur, Paris*  
Focus: massive genomics

Index & Search the Logan database

## About me: future in Bordeaux??



*inria??*  
**cnrs??**

# The Logan project: indexing and querying **all** the (meta)genomic data ever published

Roland Faure<sup>1</sup>

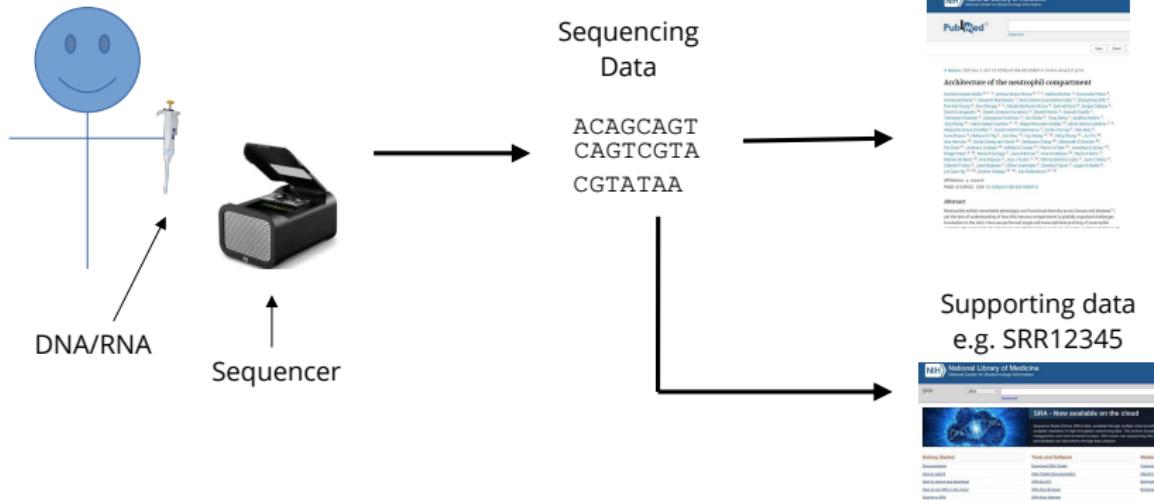
<sup>1</sup>Institut Pasteur

January 2026

Slides available (CC-BY) at: [rolandfaure.github.io](https://rolandfaure.github.io)

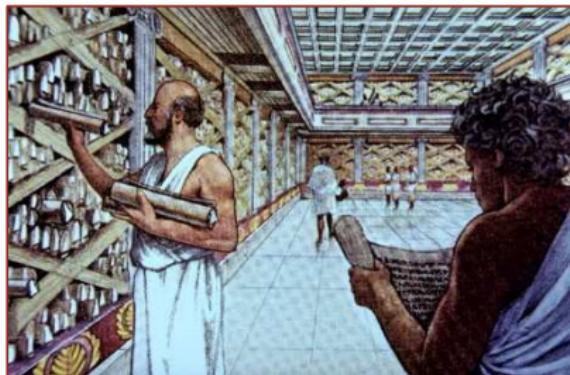
# The Logan database

# J. Doe sequenced something



## The SRA database

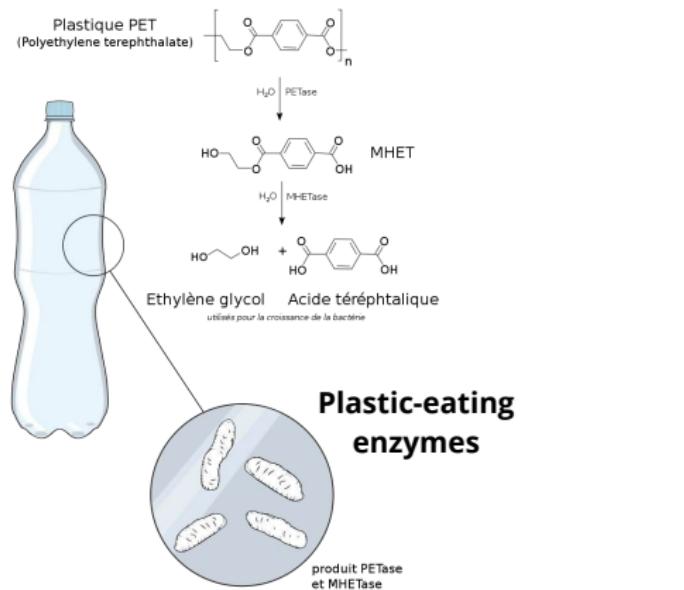
SRA: All public sequencing reads, 80 PB of data



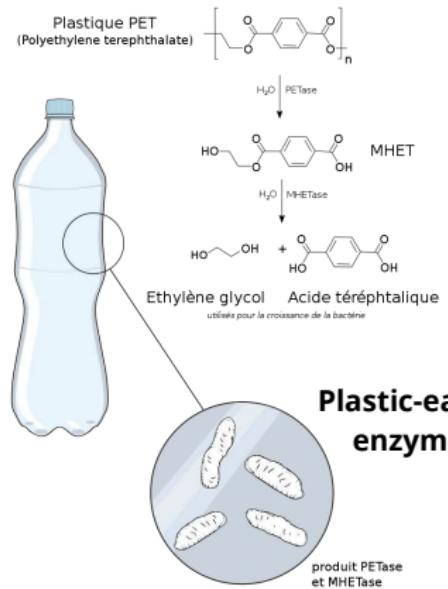
## “Library of Alexandria” for genetics

Slide Credits: Rayan Chikhi

## Plastic-eating enzymes: PETases



# Plastic-eating enzymes: PETases

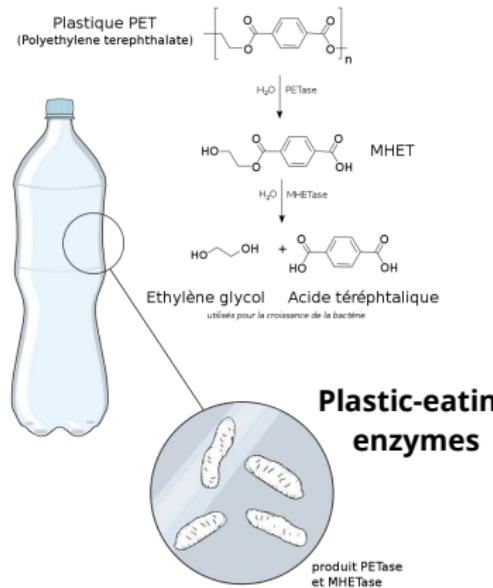


We know of 200 such enzymes but there is much we ignore, and they are hard to find



Artem Babaian

# Plastic-eating enzymes: PETases



We know of 200 such enzymes but there is much we ignore, and they are hard to find



There must be more of them in the SRA!

Artem Babaian

# The SRA is not queryable

The screenshot shows the Logan database search interface. The search term 'CGACTCGTCGCTCGCATG' is entered in the search bar. The results indicate that the term was not found in the SRA database, with a message: 'The following term was not found in SRA: CGACTCGTCGCTCGCATG.' Below this, it says 'No items found.' On the right, a 'Search details' panel shows the search term '(CGACTCGTCGCTCGCATG[All Fields])'.

## The SRA now



Slide Credits: Teo Lemane

- ▶ 27 millions accessions

## Quiz: At 1 Gbit/s, how much time to download the SRA?

A: 20 days

B: 20 weeks

C: 20 months

D: 20 years

## Quiz: At 1 Gbit/s, how much time to download the SRA?

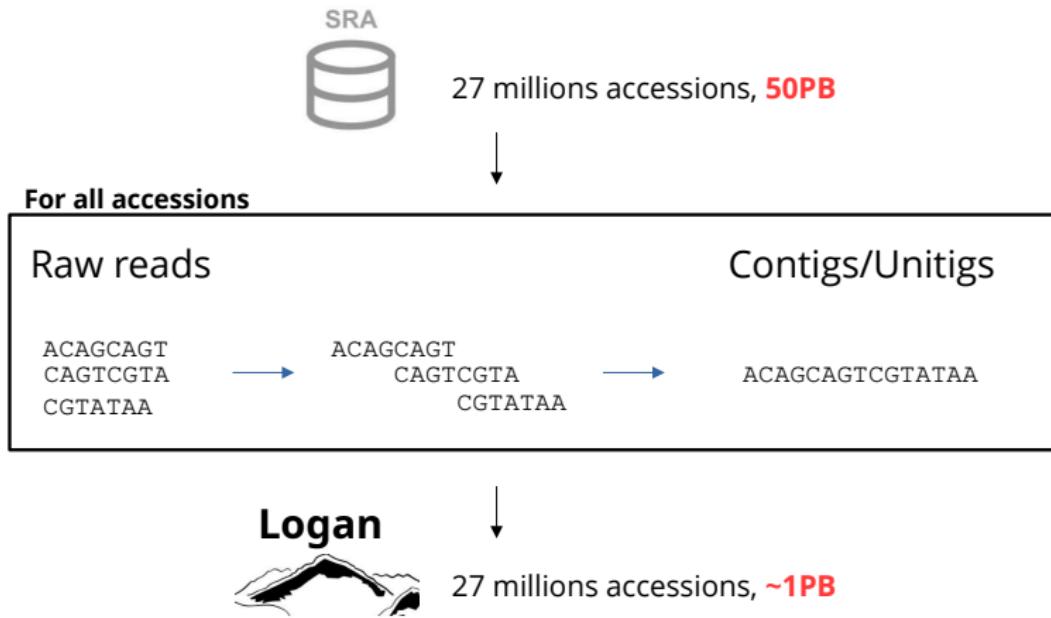
A: 20 days

B: 20 weeks

C: 20 months

D: 20 years

# The Logan database



## How much did it cost to assemble Logan?

A: \$5,000

B: \$50,000

C: \$500,000

D: \$5,000,000

## How much did it cost to assemble Logan?

A: \$5,000

B: \$50,000

C: \$500,000

D: \$5,000,000

## The Logan database

- ▶ 2.18 million parallel CPUs, 30h wall-clock time
- ▶ All the assemblies are available online

# The Logan database

- ▶ 2.18 million parallel CPUs, 30h wall-clock time
- ▶ All the assemblies are available online

## Downloading

To download one accession, type:

```
 wget https://s3.amazonaws.com/logan-pub/c/[accession]/[accession].contigs.fa.zst
```



# The Logan database

- ▶ 2.18 million parallel CPUs, 30h wall-clock time
- ▶ All the assemblies are available online

## Downloading

To download one accessi

```
wget https://s3.amazo
```

Let's look for  
homologs of my  
enzyme in Logan!



Artem Babaian

a.zst

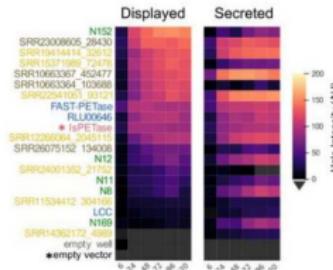


## Searching through Logan

- ▶ Downloaded the 1PB of contigs
- ▶ Aligned the known enzymes against the contigs (DIAMOND)

## Searching through Logan

- ▶ Downloaded the 1PB of contigs
- ▶ Aligned the known enzymes against the contigs (DIAMOND)
- ▶ 1.12 billion hits, 215 million clusters 90% identity
- ▶ Some discovered enzyme have better activity than known ones



## How much did it cost to align on Logan?

A: \$10

B: \$100

C: \$1,000

D: \$10,000

## How much did it cost to align on Logan?

A: \$10

B: \$100

C: \$1,000

D: \$10,000

## Indexing nucleotides

## Let's index nucleotides

- ▶ Reminder: **1PB** of data, 27 million datasets
- ▶ Query: sequence
- ▶ Answer:
  - ▶ Difficulty level 1: datasets containing similar sequences
  - ▶ Difficulty level 2: the actual similar sequences



Téo Lemane

## Indexing k-mers efficiently: bloom filters

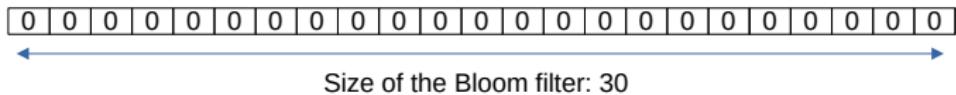
## Dataset

CACTCTGACTGA

cut in k-mers (6-mers here)

CACTCT      CTCTGA      CTGACT      GACTGA  
ACTCTG      TCTGAC      TGACTG

Bloom filter  
(bit vector)



## Indexing k-mers efficiently: bloom filters

Dataset

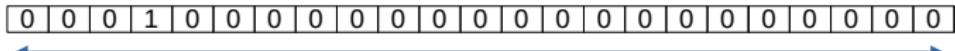
CACTCTGACTGA

cut in k-mers (6-mers here)

CACTCT      CTCTGA      CTGACT      GACTGA  
  ACTCTG      TCTGAC      TGACTG

hash(CACTCT)=4

Bloom filter  
(bit vector)



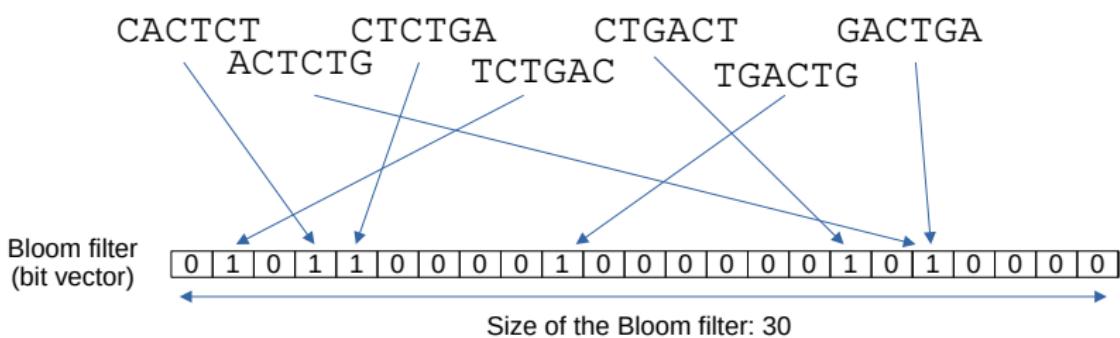
Size of the Bloom filter: 30

## Indexing k-mers efficiently: bloom filters

Dataset

CACTCTGACTGA

cut in k-mers (6-mers here)



# Indexing k-mers efficiently: bloom filters

Is CTCTGA in my dataset ?

Bloom filter  
(bit vector)

0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

## Indexing k-mers efficiently: bloom filters

Is CTCTGA in my dataset ?

hash(CTCTGA)=5

Bloom filter  
(bit vector)



0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

## Indexing k-mers efficiently: bloom filters

Is **AAAAAA** in my dataset ?

hash(AAAA)=14

Bloom filter  
(bit vector)

0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

## Indexing k-mers efficiently: bloom filters

Is GGGGGG in my dataset ?

hash(GGGGGG)=2  
**false positive !**

Bloom filter  
(bit vector)

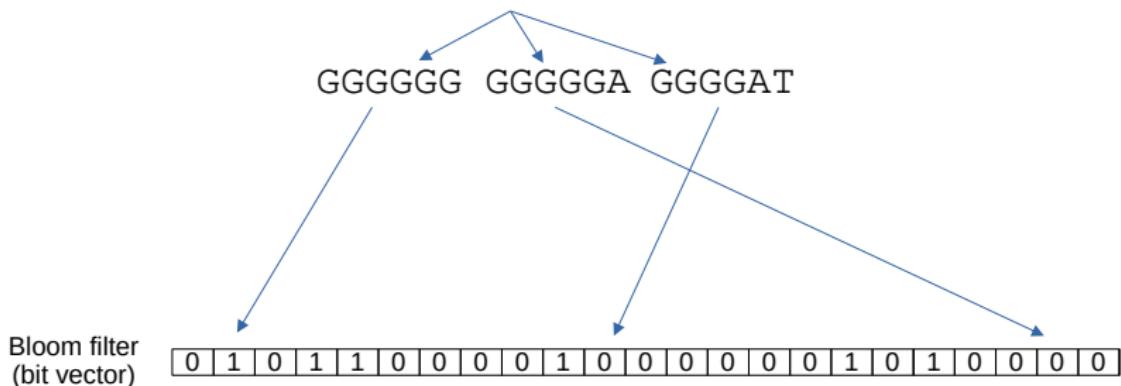


0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

# Indexing k-mers efficiently: bloom filters

## Trick: query (k+s)-mers

Is GGGGGGAT in my dataset ?



## Indexing strategy

- ▶ Index 26-mers of all datasets in Bloom filters
- ▶ At query time, query 31-mers

SRR00001	0 1 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0
SRR00002	0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 1 0 1 0 1 0 0 1 1
SRR00003	0 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0
SRR00004	0 1 1 1 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 1 0 0 0 0
SRR00005	1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 1 0 0
SRR00006	1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 0
SRR00007	0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 1

► In total, 1PB



# kmindex

SRR00001	0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0
SRR00002	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	1	0	0	0	1	1
SRR00003	0	0	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0
SRR00004	0	1	1	1	1	0	0	0	0	1	0	1	0	1	0	1	1	0	1	0	0	0	0	0
SRR00005	1	1	1	1	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0	1	0	1	0
SRR00006	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0
SRR00007	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0

## kmindex

SRR00001  
SRR00002  
SRR00003  
SRR00004  
SRR00005  
SRR00006  
SRR00007

0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	0	1	0	0	1	1	0	0	1	1	0	1	0	1	0	1	0	0	1	1
0	0	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
0	1	1	1	1	1	0	0	0	1	0	1	0	1	0	1	1	0	1	0	1	0	0	0	0
1	1	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	1	1	0	0
1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0
0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

In which datasets can we find

GGGGGG

GGGGGA GGGGAT

GGGGGGAT

## kmindex

SRR00001	0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0
SRR00002	0	1	0	0	0	0	1	0	0	1	1	0	0	0	1	1	0	1	0	1	0	1
SRR00003	0	0	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	0
SRR00004	0	1	1	1	1	0	0	0	0	1	0	1	0	1	0	1	1	1	0	1	0	0
SRR00005	1	1	1	1	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0	1	1
SRR00006	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0
SRR00007	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0	0	1

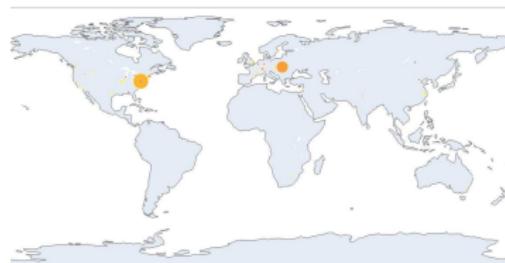
In which datasets can we find

GGGGGA GGGGAT  
GGGGGGAT

## ► Index of 1PB

- ▶ “Find all datasets that share x% of 31-mers with my query”

Table		Map	Plot	BLAST-like alignment	?	Help
InterCoverage > 0.7 AND assay_type IN (WGS, WGA)						
ID	kmers_coverage	bioproject	bioproject	bioproject	bioproject	bioproject
SRB16173100	0.999999	1	PRJNA763258	PRJNA763258	SAMN22020249	SRBAC09
SRB14416987	0.999999	1	PRJNA721998	PRJNA721998	SAMN19658652	SRBAC09
SRB15166088	0.999999	1	PRJNA740342	PRJNA740342	SAMN20037324	SRBAC09
SRB35501501	0.999999	1	PRJEB27179	PRJEB27179	SAMNE539711	SRBAC09
SRB37404047	0.999999	1	PRJNA372431	PRJNA372431	SAMN03135320	SRBAC09
SRB2399873	0.999999	1	PRJEB22684	PRJEB22684	SAMNE78488328	SRBAC09
SRB14605031	0.999999	1	PRJNA732627	PRJNA732627	SAMN22121545	SRBAC09
SRB13309823	0.999999	1	PRJNA697219	PRJNA697219	SAMN17141292	SRBAC09
SRB14614320	0.999999	1	PRJNA727098	PRJNA727098	SAMN19606784	SRBAC09
SRB23795820	0.999999	1	PRJEB22684	PRJEB22684	SAMNE10488221	SRBAC09
SRB13011679	0.999999	1	PRJNA669806	PRJNA669806	SAMN16710096	SRBAC09
SRB23923950	0.999999	1	PRJEB22684	PRJEB22684	SAMNE10488707	SRBAC09
SRB23949336	0.999999	1	PRJEB22684	PRJEB22684	SAMNE10488813	SRBAC09
SRB25617423	0.999999	1	PRJNA100409	PRJNA100409	SAMN06320974	SRBAC09
SRB36591973	0.999999	1	PRJNA100161	PRJNA100161	SAMNE78173649	SRBAC09
SRB3660137	0.999999	1	PRJNA511728	PRJNA511728	SAMN03178007	SRBAC09
SRB11336098	0.999999	1	PRJNA123968	PRJNA123968	SAMN14395256	SRBAC09
SRB25507401	0.999999	1	PRJNA100193	PRJNA100193	SAMN06438017	SRBAC09



## How much does it cost to query Logan-search?

A: \$1

B: \$10

C: \$100

D: \$1,000

## How much does it cost to query Logan-search?

A: \$1

B: \$10

C: \$100

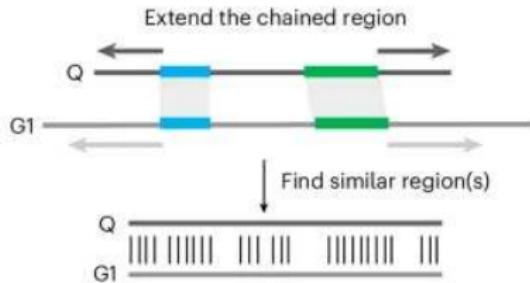
D: \$1,000

## Indexing sequences: limits & future

- ▶ Limit: slow to get the actual sequences
- ▶ Limit: query and target need to share 31-mers

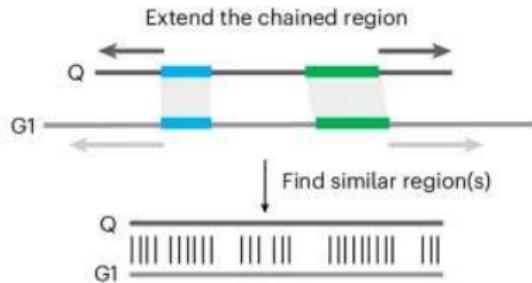
## Another strategy to index sequences: LexicMap

### d Pseudoalignment



## Another strategy to index sequences: LexicMap

### d Pseudoalignment

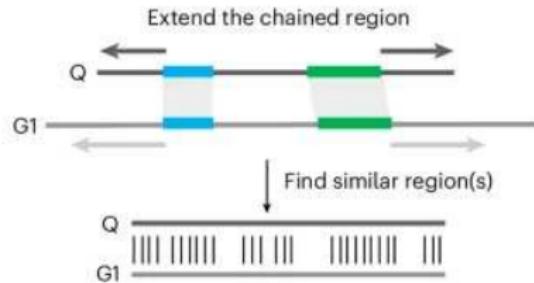


- Available on Galaxy soon



## Another strategy to index sequences: LexicMap

### d Pseudoalignment



- ▶ Available on Galaxy soon



- ▶ More sensitive but slow (several hours)

## Indexing proteins

## Obtaining all the proteins of SRA

- ▶ Ran prodigal on all assemblies: 100 billion proteins
- ▶ Clustered with MMseqs2 at 50% identity: 3 billion representative proteins

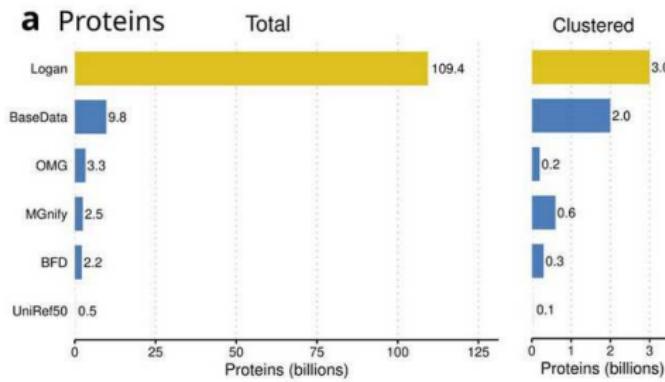
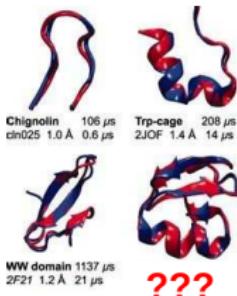


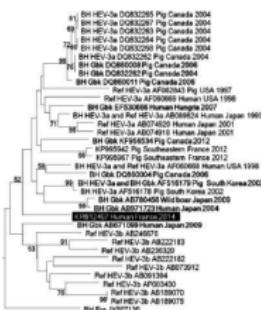
Figure from the Logan preprint

What can we do with these proteins?

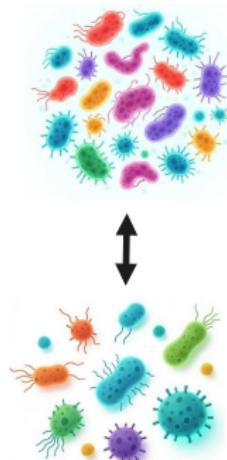
Discover new proteins



Improve protein phylogenies

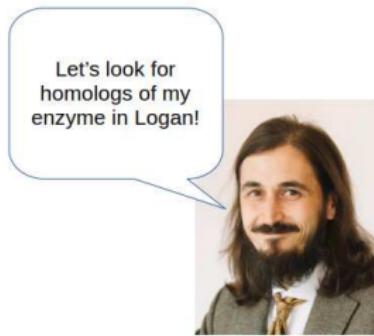


## Compare datasets



## My work: looking for proteins by similarity

- ▶ Query: a protein
- ▶ Answer: all similar proteins in Logan



Artem Babaian

# How to compare (3 billion) proteins?

Strategy 1: sequence  
comparisons

MRIF**GFFITLVAAI**IGQ  
|||||||  
MRIK**GFFITLIAII**IFQ

# How to compare (3 billion) proteins?

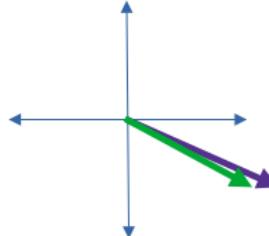
Strategy 1: sequence  
comparisons

MRIF**GFFITLVAII**IQ  
|||||||  
MRIK**GFFITLIAII**IQ

Strategy 2: embedding  
comparisons

**MRIKGFFITLIAII**IQ  
**MRIFGFFITLVAII**IQ

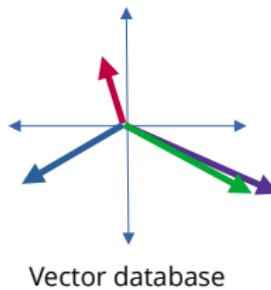
Protein Language  
Model embedding



# Indexing 3 billion proteins

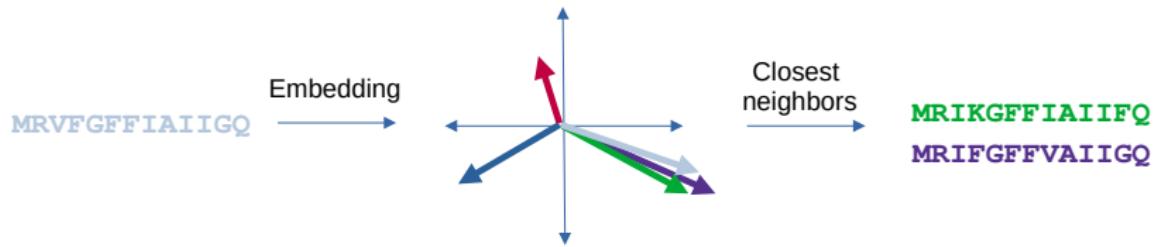
MRIKGFFITLIAIIFQ  
MRIFGFFFITLVAIIGQ  
MSIYHMKVRTITGKDMTLQP  
MTFFFLYIISPMISILIGFK  
.....

Embedding



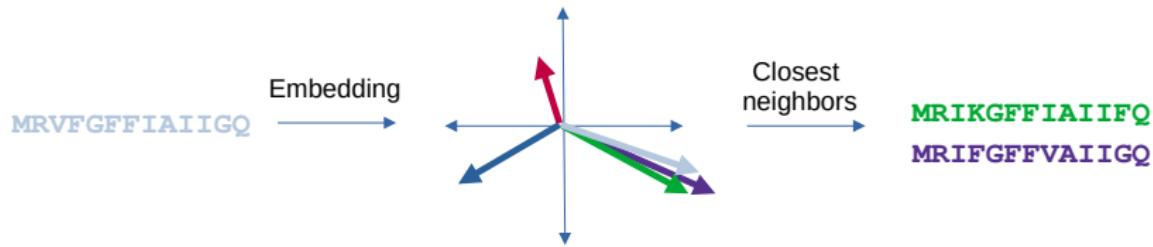
- ▶ Protein Language Model: gLM2
- ▶ 512-dimension vectors
- ▶ 3k GPU.hours
- ▶ Space taken by final database: 1.5TB

## Querying 3 billion proteins



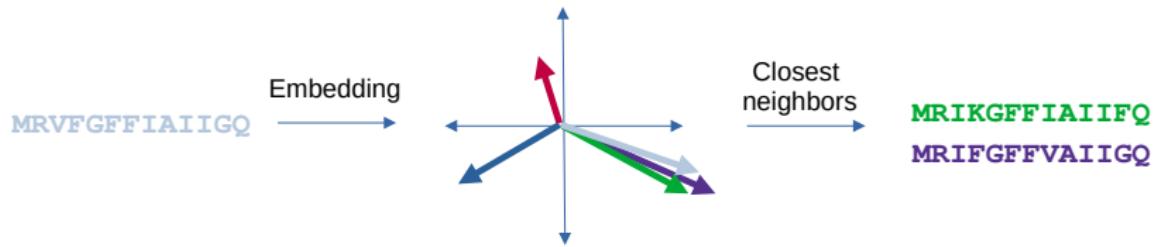
- ▶ Embed query & compare to existing vectors

## Querying 3 billion proteins



- ▶ Embed query & compare to existing vectors
- ▶ Compare to 3 billion vectors using industrial vector databases

## Querying 3 billion proteins



- ▶ Embed query & compare to existing vectors
- ▶ Compare to 3 billion vectors using industrial vector databases
- ▶ Actually just bruteforce comparison

## Protein search: performance

- ▶ Query: 301 known papillomavirus proteins

## Protein search: performance

- ▶ Query: 301 known papillomavirus proteins
- ▶ 4h computation, 120 GB RAM ( $\sim \$10$ )

## Protein search: performance

- ▶ Query: 301 known papillomavirus proteins
- ▶ 4h computation, 120 GB RAM ( $\sim \$10$ )
- ▶ 50k homologs in Logan

## Protein search: performance

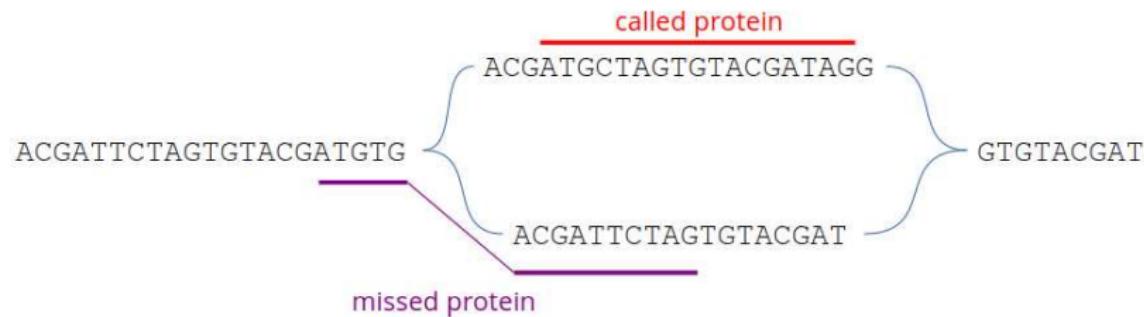
- ▶ Query: 301 known papillomavirus proteins
- ▶ 4h computation, 120 GB RAM ( $\sim \$10$ )
- ▶ 50k homologs in Logan
- ▶ Soon available online on Galaxy and downloadable ( $\sim 5\text{TB}$ )
- ▶ Contact me if you are interested now

## Limits

- ▶ Only full proteins match

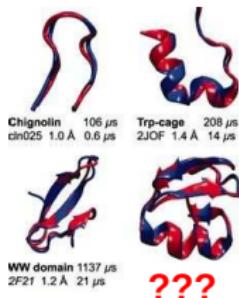
## Limits

- ▶ Only full proteins match
- ▶ 90% proteins missing in the database because of the protein calling

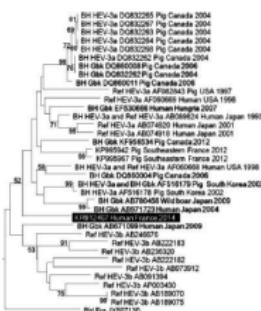


## Future developments of Logan

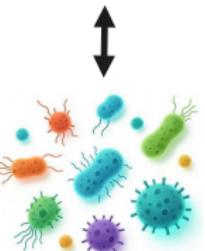
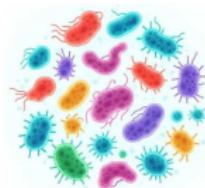
Discover new proteins



Improve protein phylogenies



## Compare datasets



## Take-home message

- ▶ Logan centralizes all public datasets
- ▶ Nucleotide and proteins

## Take-home message

- ▶ Logan centralizes all public datasets
- ▶ Nucleotide and proteins



- ▶ Contact us if you need help using Logan tools
- ▶ Contact us if you want some features
- ▶ You can propose features too