Indexing large metagenomic projects with abundances

Téo Lemane, Lucas Robidou, Rayan Chikhi, Pierre Peterlongo

DSB 2023 Delft

Objectives

At the price of
- Approximate answers (FP, overestimations)
- Need fast disk (local SSD preferentially)

Index

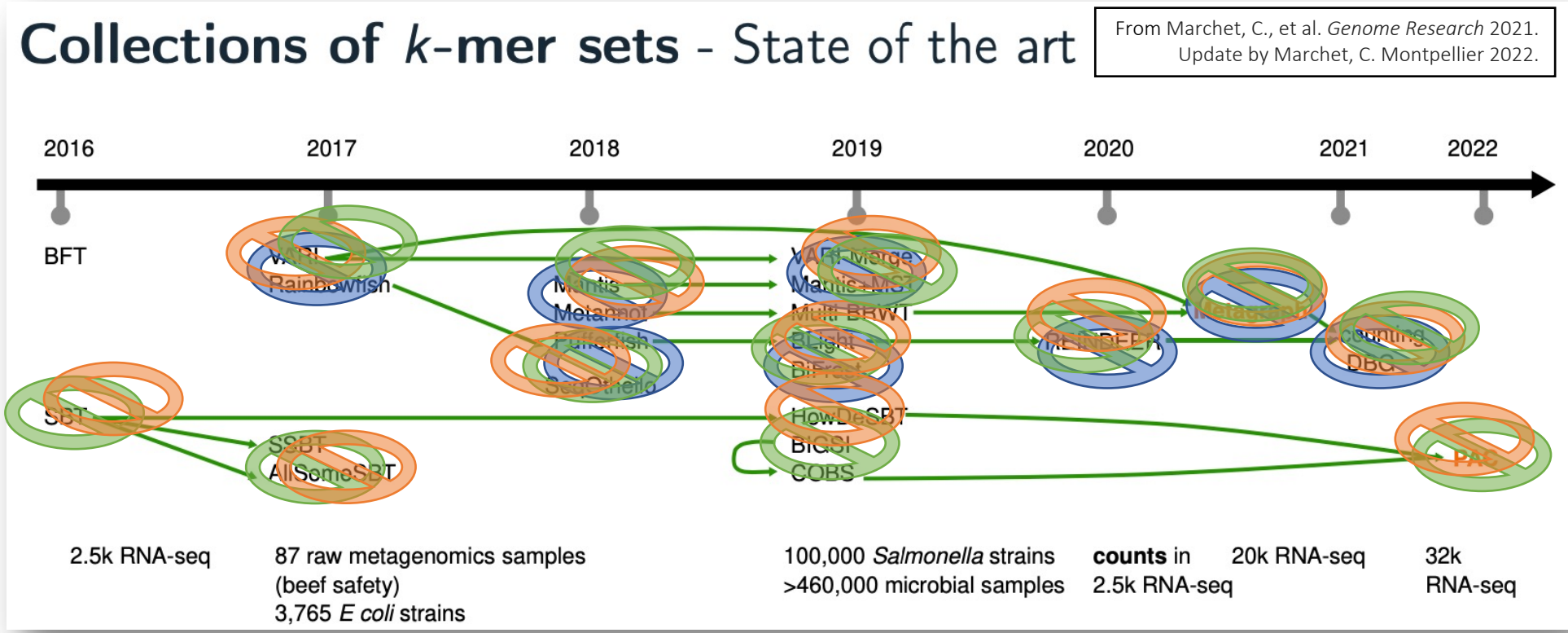
- Genomic datasets:
  - Large
    - > hundreds, thousand samples
    - TB to PB sized
  - **Complex**
    - metagenomes, metatranscriptomes,
    - high variability (sea water, soil, …)
- With:
  - Low RAM usage
    - (max 100 GB)
  - Dynamicity
    - Able to add new samples to the index
  - Fast
    - Indexing hundreds of samples in a few hours

Query

- Short (reads) or long (genomes) sequences
- One sequence (google-like)
  - Real time (milliseconds)
  - No RAM

  Or
- Many sequences (read set)
  - Fast (~hours)
  - RAM limited (max 100 GB)
- With or without abundance

Possible today ?



Fast (indexing + query)
- No random access

→ no tree

Low memory
- Do not raise index in RAM at indexing or query time

High complexity
- Do not require unitigs
- No not associate k-mers to color vector

Note
theoretical complexity != practical speed

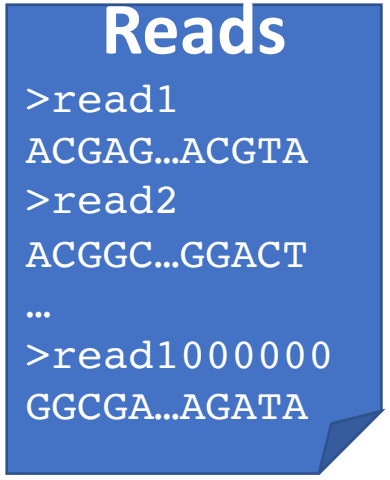
`--march native -03`

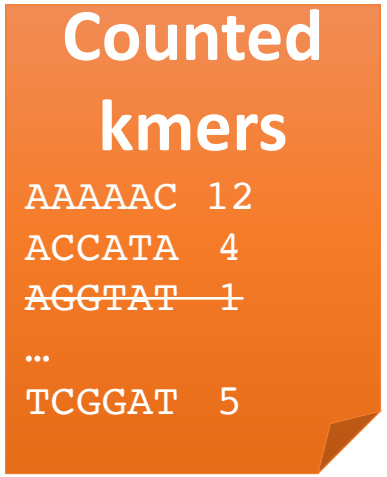
Note (even more knowing that many of the authors of these tools are in the room ☺)
These tools are not adapted to our objectives (prev. slide) but
Adapted to other contexts, and have other features (alignments, variants, …)

Indexing: conceptual view

One read set:

- Extract & count kmers
- Filter kmers
- Generate a [counting] bloom filter





Note: Bloom Filter

1	0	0	1	0	0	1	1

A bit vector B of fixed size
Add one element -> hash(element) -> set B[hash] = 1
Query one element -> hash(element) -> returns B[hash]
0: absent
1: present (**possibly a False Positive**)

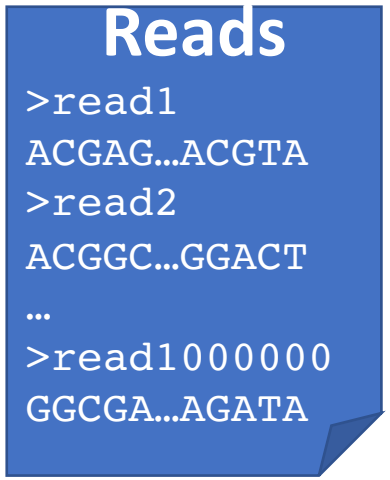
Note: counting Bloom Filter

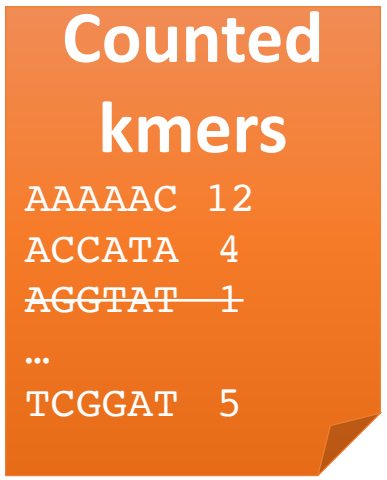
A bit vector B of fixed size, x bits per element
Add one element -> hash(element) -> B[hash] += 1
Query one element -> hash(element) -> returns B[hash]
0: absent
n>0: present with non null False Positive Rate
n: abundance (**possibly overestimated**)

Indexing: conceptual view

One read set:

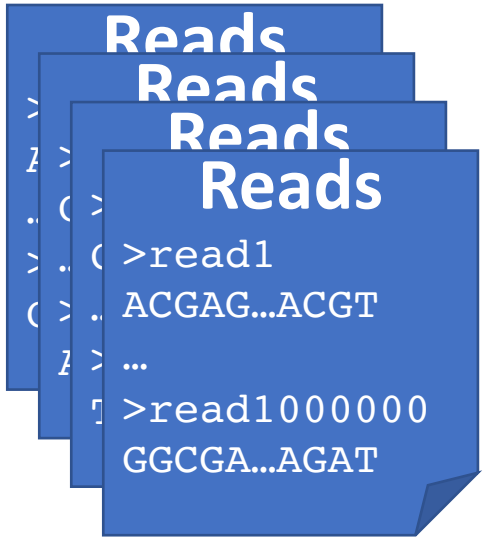
- Extract & count kmers
- Filter kmers
- Generate a [counting] bloom filter





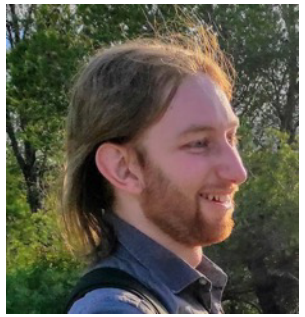
N read sets:

- Create N [counting] bloom filters
- This is the index



Two contributions

[Counting] Bloom Filters



Lucas Robidou

- Exponential decrease of Bloom filter FPrate
- Decrease of counting Bloom Filter overestimations

https://www.biorxiv.org/content/10.1101/2022.06.27.497694v3

From reads to Indexes

Téo Lemane

- Optimized kmer index:
  - Representation
  - Creation
  - Update
  - Query

Lemane T. et. al. “kmindex, user-friendly indexation and real-time query of kmers in terabyte-sized genomic data banks”
Manuscript in prep.

Lucas Robidou https://github.com/lrobidou/fimpera

Fimpera: Counting BF with low disk, low FP, low counting overestimates, no drawback

key idea for presence absence:

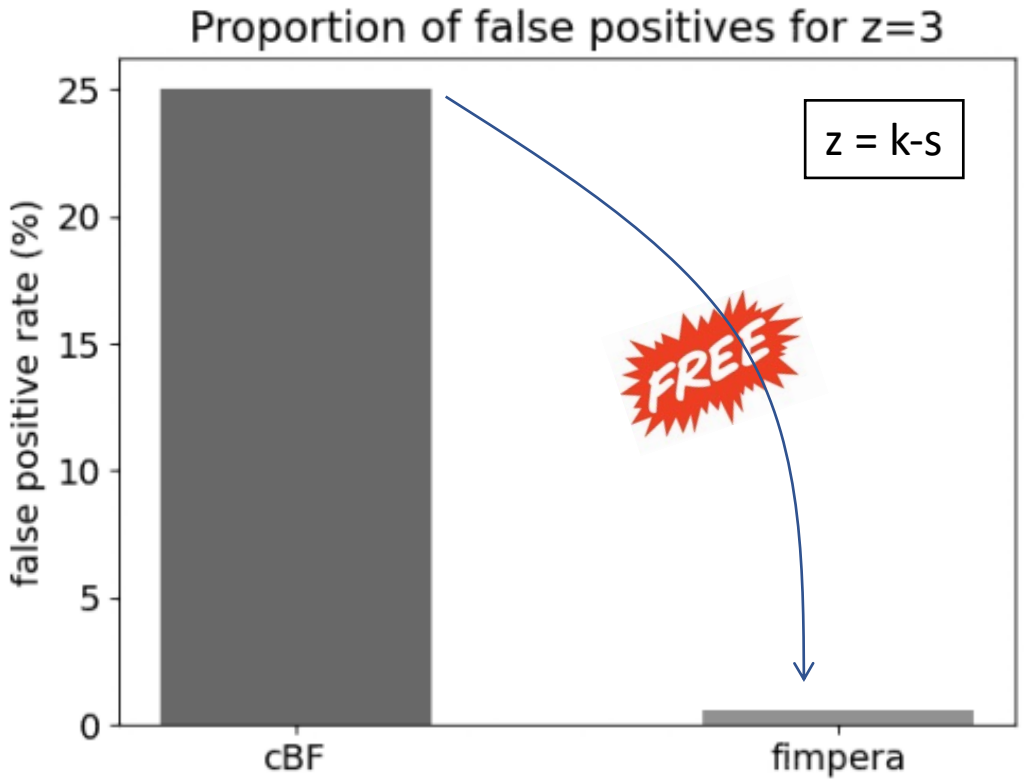
- If a kmer exists all words inside this kmer (smers) exist

⟷

- If a smer of a kmer does not exists, the kmer does not exist

In practice:

- Index smers
- When querying a kmer, report it as present *iif* all its constituent smers are present



Indexed: Tara Ocean ERR1726642
Queried: Tara Ocean ERR4691696

Lucas Robidou https://github.com/lrobidou/fimpera

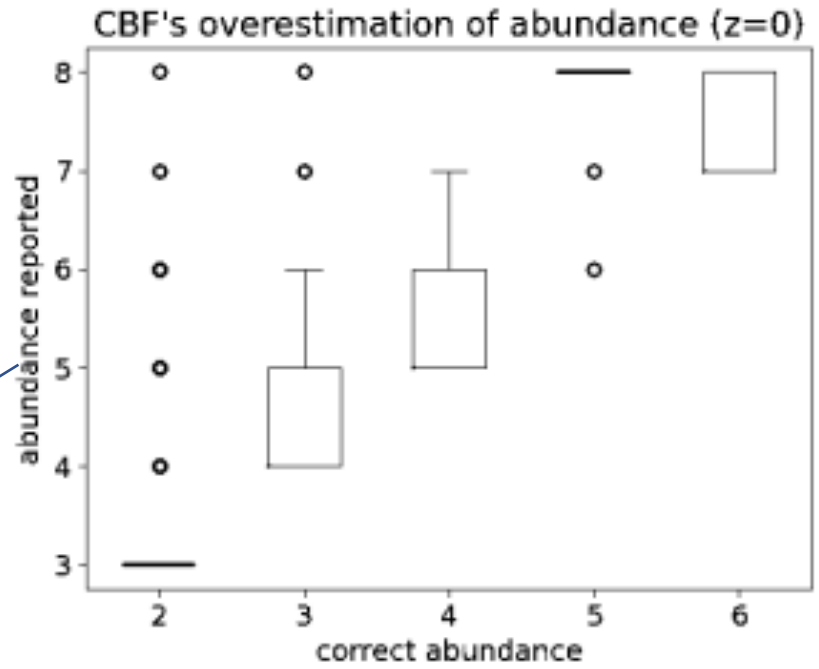
Fimpera: Counting BF with low disk, low FP, low counting overestimates, no drawback

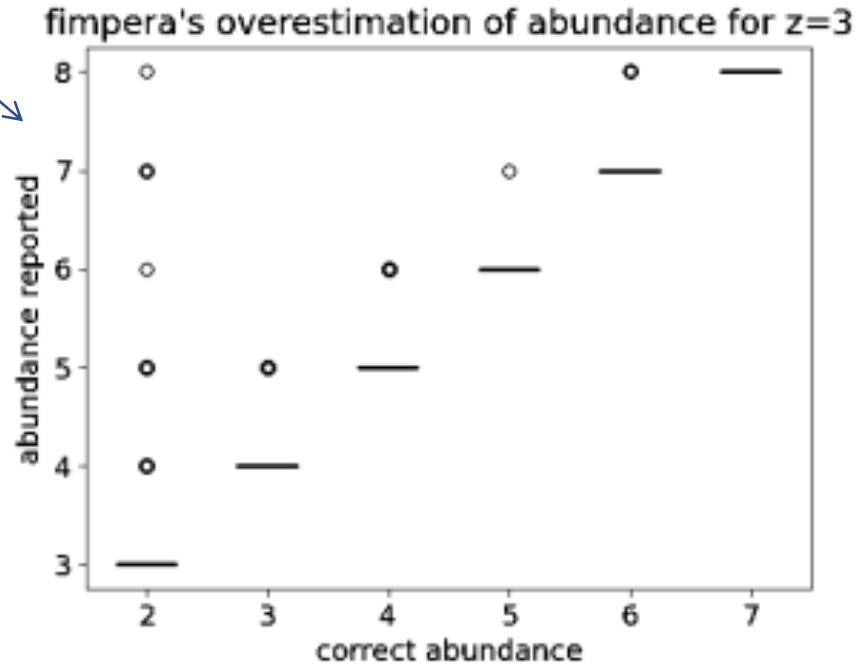
key idea for abundance:

- The abundance of a kmer is at most the abundance of its less abundant constituent smer

In practice:

- Index smers abundances
- When querying a kmer, return the abundance of its least abundant smer





Indexed: Tara Ocean ERR1726642
Queried: Tara Ocean ERR4691696

Lucas Robidou https://github.com/lrobidou/fimpera

Fimpera: Counting BF with low disk, low FP, low counting overestimates, no drawback

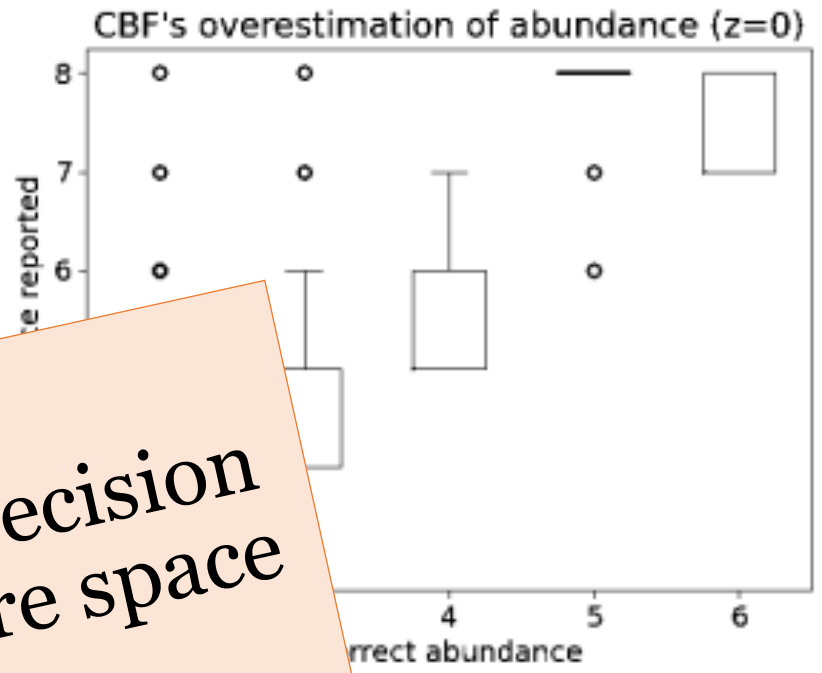
key idea for abundance:

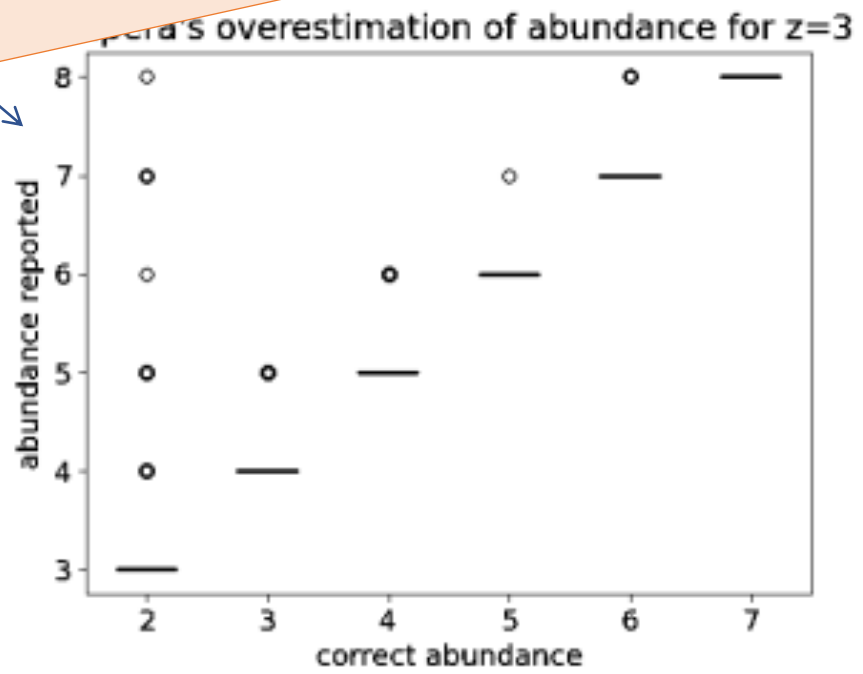
- The abundance of a kmer is
  abundance of its
  constitu

In practice

- Index sme
- When que a kmer, return the abundance of its least abundant smer

Without Fimpera, the same precision would require ~35x times more space





Indexed: Tara Ocean ERR1726642
Queried: Tara Ocean ERR4691696

Téo Lemane

https://github.com/tlemane/kmindex

Kmindex: indexation and real-time query of kmers in terabyte-sized genomic data banks

key features

- Count hashes instead of ascii kmers
- Clever kmer filtration process
- **Once kmers sorted:**
  - **All processes are sequential**
- Kmers -> partition -> parallelization
  - At indexing time
  - At query time

key features

- Avoids HowDeSBT:
  - ☹ looses inter-sample compression (ok for complex datasets)
  - ☺ wins query time (≈ 1300x speed up)
- Instead: simple inverted index
- mmap at query time
  - Clever pages in RAM management
- Integrates Fimpera

Based on kmtricks :

Lemane, T., Medvedev, P., Chikhi, R., & Peterlongo, P. Bioinformatics Advances, 2(1), vbac029.

Result: Index construction

Databank:

- 50 Tara Ocean samples
- Avg 11 billions distinct kmers per sample
- 1.4TB fastq.gz

Indexing: one command line

```
`kmindex files    |smer|    threads    |bloom|`
                   (23)       (32)       (30billions)
```

- Wall clock time: **2h56**
- Peak RAM: 107GB
- Peak disk: 878GB
- Final index size: 164GB

Result: query

Databank:

- 50 Tara Ocean samples
- Avg 11 billions distinct kmers per sample
- 1.4TB fastq.gz

querying: one command line: `` `kmindex query index query.fa` ``

#queries (reads)	1	10k	1 million	10 millions
Max RAM (GB)	0.005	0.05	4.9	46.7
Time (s) – cold RAM	<0.1	20	94	261 (2m21s)
Time (s) – warm RAM	<0.1	10.84	41	227

"rocket mode"
Use as much RAM as available

#queries (reads)	1	10k	1 million	10 millions
Max RAM (GB)	0.005	2.84	133	194
Time (s) – cold RAM	<0.1	17	61	99
Time (s) – warm RAM	<0.05	7	16	64

OGA Server, in progress

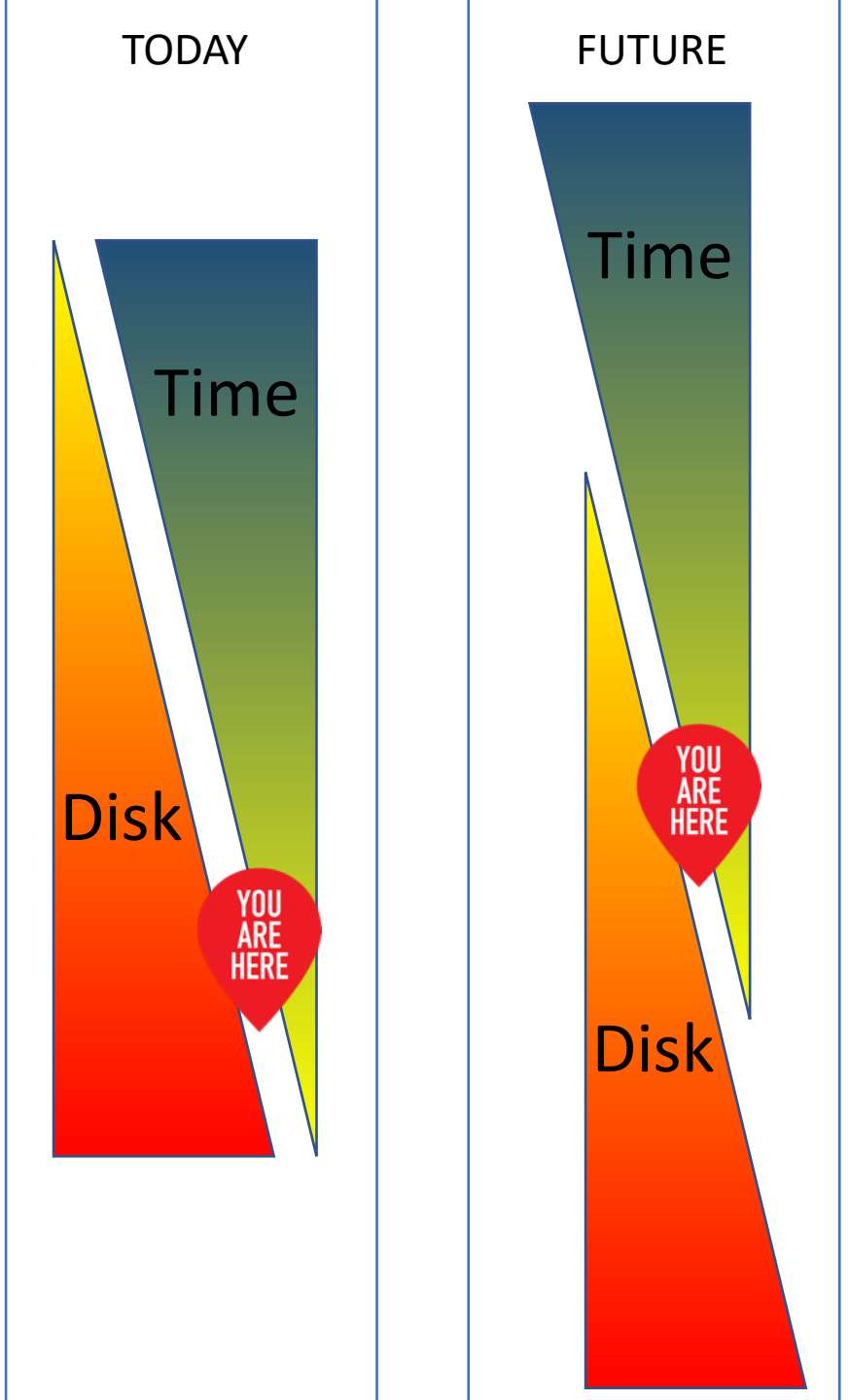
- Index: all Tara Ocean Metagenomic samples
  - Input fastq.gz files: 6TB
    - 241 datasets
    - 6.5 thousand billion nucleotides
    - 266 billion distinct k-mers
      - 6564 billions total k-mers
  - Final index size: 0.6TB

- Included in the Ocean Gene Atlas server

What comes next?

- Deploy new instances
  - MetaSub
  - Tara MetaTranscriptomes
- New usages:
  - Eg. Find Reads containing some indexed kmers

- **Scale even more, from TB to PB.**
  - **Less disk <-> More Time**



https://github.com/tlemane/kmindex

Thanks ☺