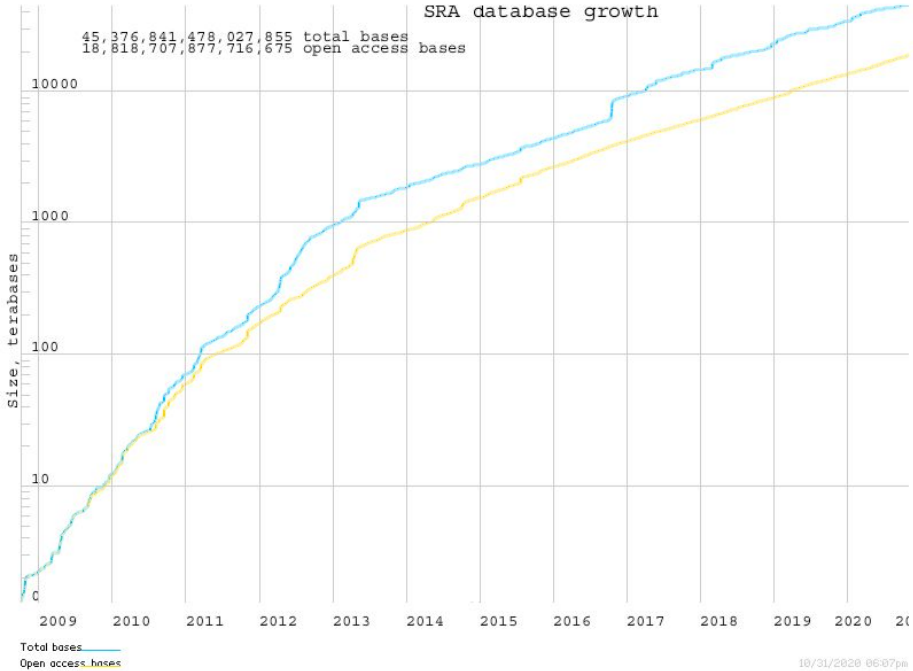
kmtricks: modular k-mer count matrix and Bloom filter construction for large read collections

Téo Lemane, Rayan Chikhi, Pierre Peterlongo

-

SeqBIM 2020

Database growth



- Tara Ocean: **250 billions** metaG reads
- 100000 genome project: **~19 PB**
- SRA: **> 30 PB**

https://trace.ncbi.nlm.nih.gov/Traces/sra/

Indexing: Motivation & Applications

Sequencing data → Assembly/Mapping → Analyses →

Data sleeps in rarely opened drawers

Indexing: Motivation & Applications

Sequencing data → Assembly/Mapping → Analyses →

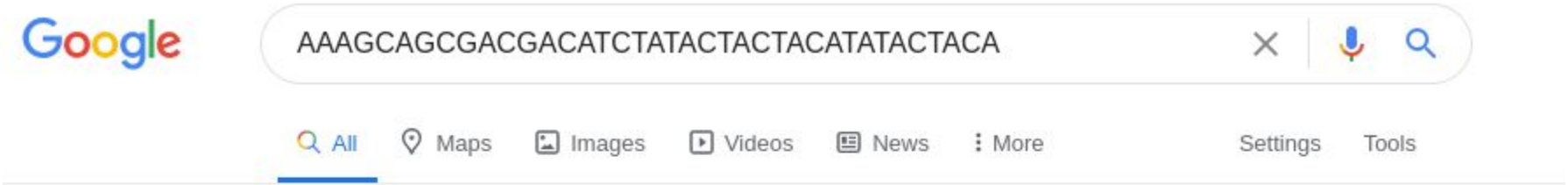
Data sleeps in rarely opened drawers

Querying this data could help answer some questions:

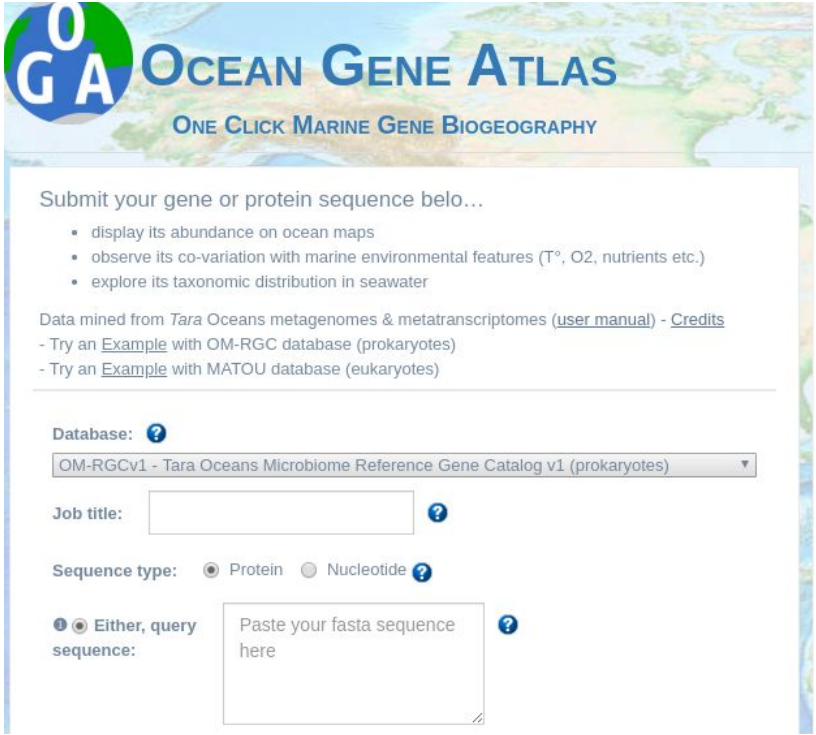
- RNA-seq
  - Expressed isoform according to tissues [1]
  - Gene fusion [2]
- Microbial genomics
  - Antimicrobial resistance [3]
- Genome dynamics
  - Phylogeny [4]
- …

[1] B. Solomon and C. Kingsford. Fast search of thousands of short-read sequencing experiments. Nature Biotechnology, 2016.
[2] Y. Yu, et al. Seqothello: querying rna-seq experiments at scale. Genome Biology, 2018.
[3] N .Luhmann, et al. Blastfrost: Fast querying of 100,000 s of bacterial genomes in bifrost graphs. BioRxiv, 2020.
[4] R. Wittler. Alignment-and reference-free phylogenomics with colored de bruijn graphs. Algorithms for Molecular Biology, 2020.

How to query these data ?

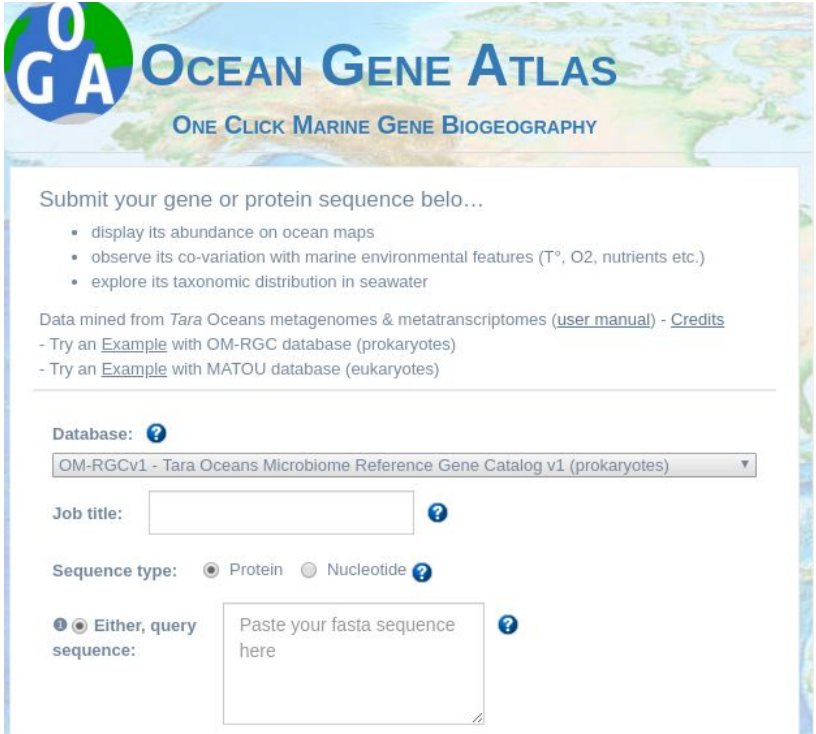


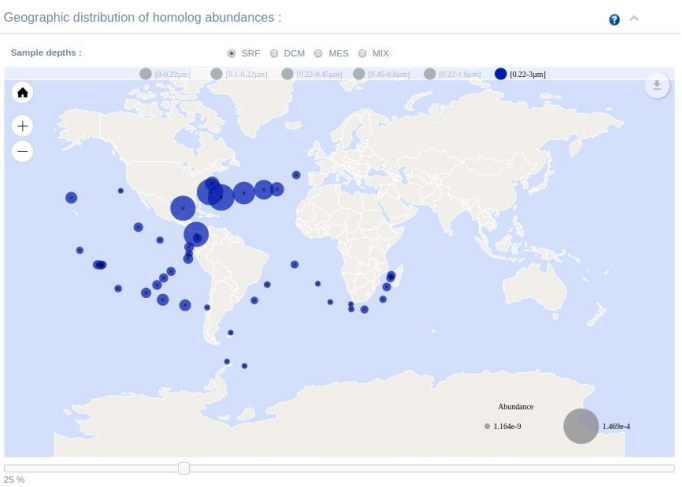
Existing solution: Example of Ocean Gene Atlas



http://tara-oceans.mio.osupytheas.fr/ocean-gene-atlas/

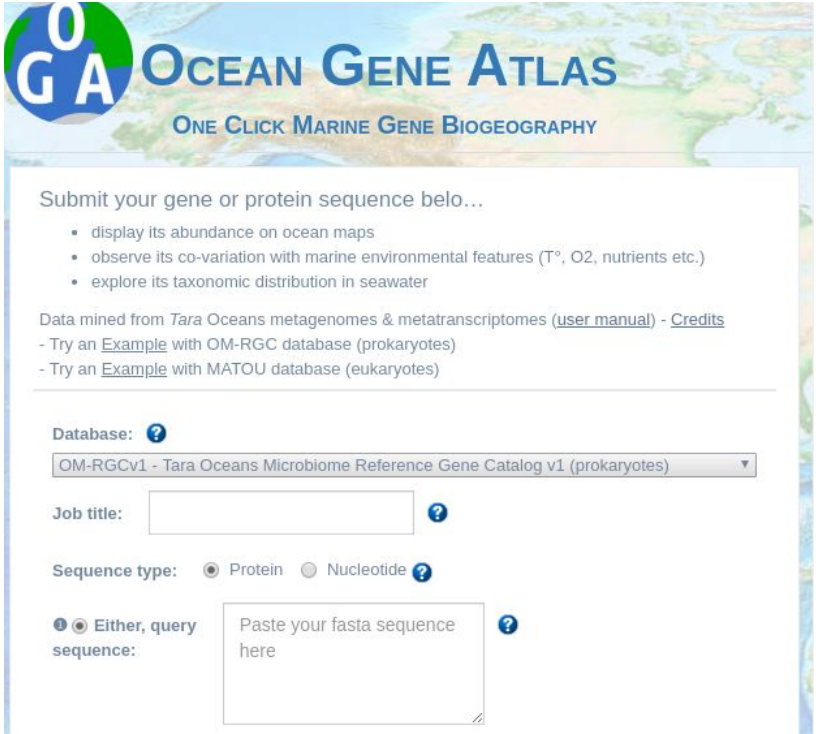
Existing solution: Example of Ocean Gene Atlas

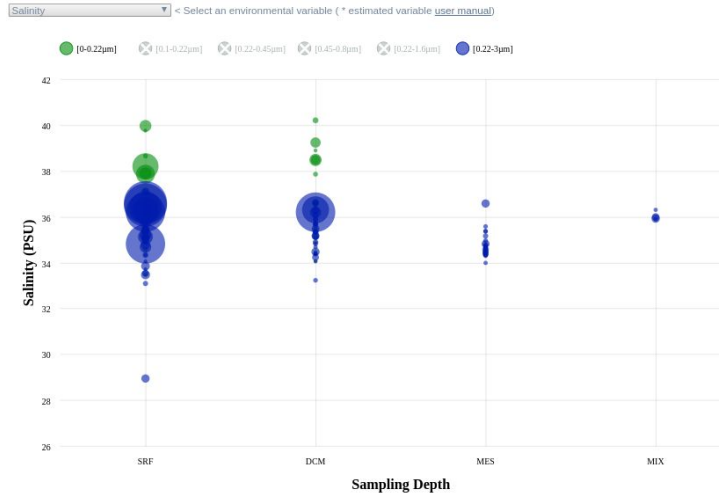




http://tara-oceans.mio.osupytheas.fr/ocean-gene-atlas/

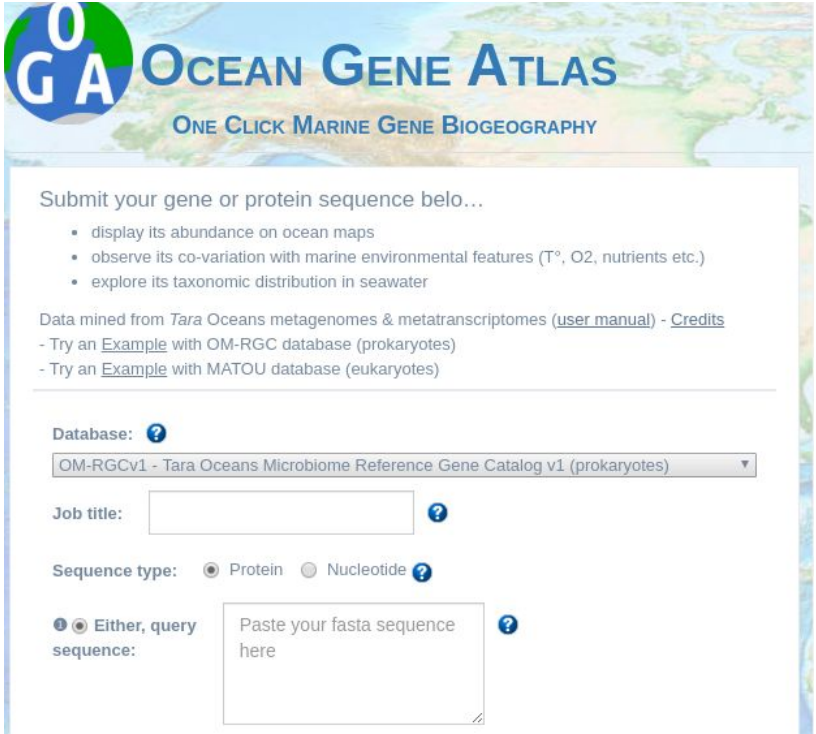
Existing solution: Example of Ocean Gene Atlas

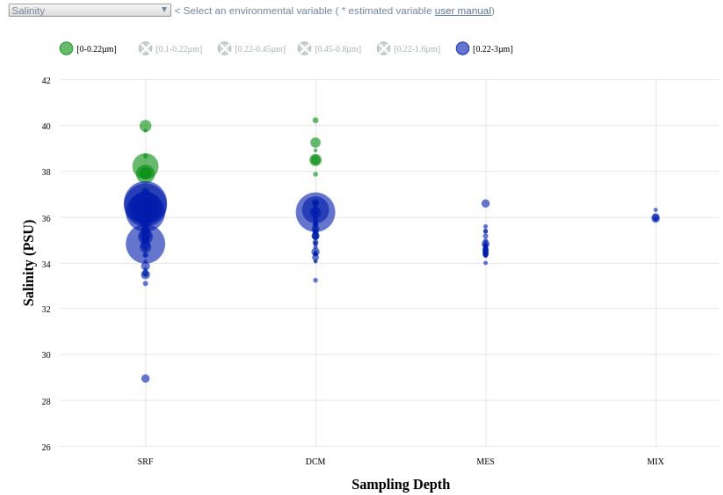




http://tara-oceans.mio.osupytheas.fr/ocean-gene-atlas/

Existing solution: Example of Ocean Gene Atlas





- Limited to Tara assembled Genes
- Usage of Blast, Diamond, HMMER

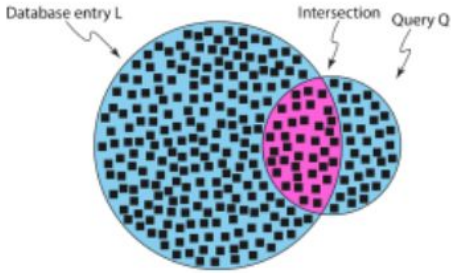
http://tara-oceans.mio.osupytheas.fr/ocean-gene-atlas/

From sequence alignement to k-mers

Problem: Given experiments sets, and a sequence of interest, which dataset contains this sequence ?

In terms of k-mers:

- A query *Q* matches an experiment *L* if at least a fraction θ of *Q*'s k-mers are present in *L*.

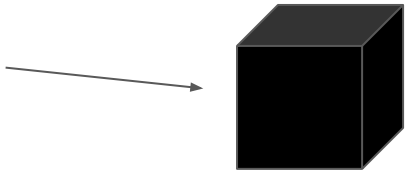


k-mer indexing

query requires membership data structures

Query:
AC**ACTCGCAGAG**GGATTATTTTTAAA

For each k-mer
(e.g. **ACTCGCAGAG**)



dataset_0	False
dataset_1	True
...	...
dataset_n	True

k-mer indexing

k-mer indexing methods (non exhaustive):

- BFT (Holley *et al.*, 2016)*
- Sequence Bloom Tree*:
  - SBT (Solomon & Kingsford, 2016)
  - AllSomeSBT (Sun *et al.*, 2017)
  - SSBT (Solomon & Kingsford, 2018)
  - HowDeSBT (Harris & Medvedev, 2019)
- Mantis (Pandey *et al.*, 2018)
- SeqOthello (Yu *et al.*, 2018)
- BIGSI (Bradley *et al.*, 2019)*
- COBS (Bingmann *et al.*, 2019)*

***Based on Bloom filters**

Review of k-mer indexing methods: Data structure based on k-mers for querying large collections of sequencing datasets (Marchet *et al*. 2019)

k-mer indexing: State of the art

Space and time results on 2585 human RNA-seq sets

Tool	Data Processing Time (days)	Max Ext. Memory (GB)	Time (h, wallclock)	Peak RAM (GB)	Index Size (GB)
SBT	3.5[b]	300[a]	55[b]	25[b]	200[a]
AllSomeSBT	3.5[a]	600[a]	25[a]	35[b]	140[a]
SSBT	3.5[a]	600[a]	55[a]	5[b]	20[a]
HowDeSBT	2.5[a]	30[a]	10[a]	N/A	15[a]
Mantis	130[a]	3,500	20[a]	N/A	30[a]
SeqOthello	3.5[b]	190[b]	2[b]	15[b]	20[b]
BIGSI	N/A	N/A	N/A	N/A	145[c]

Marchet *et al*. 2019

k-mer indexing: State of the art

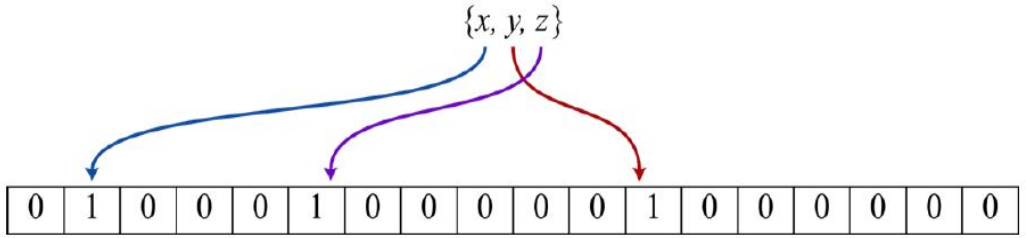
Space and time results on 2585 human RNA-seq sets

Tool	Data Processing Time (days)	Max Ext. Memory (GB)	Time (h, wallclock)	Peak RAM (GB)	Index Size (GB)
SBT	3.5[b]	300[a]	55[b]	25[b]	200[a]
AllSomeSBT	3.5[a]	600[a]	25[a]	35[b]	140[a]
SSBT	3.5[a]	600[a]	55[a]	5[b]	20[a]
HowDeSBT	2.5[a]	30[a]	10[a]	N/A	15[a]
Mantis	130[a]	3,500	20[a]	N/A	30[a]
SeqOthello	3.5[b]	190[b]	2[b]	15[b]	20[b]
BIGSI	N/A	N/A	N/A	N/A	145[c]

Marchet *et al*. 2019

- focuse on improving data processing time in the case of HowDeSBT

Bloom filters

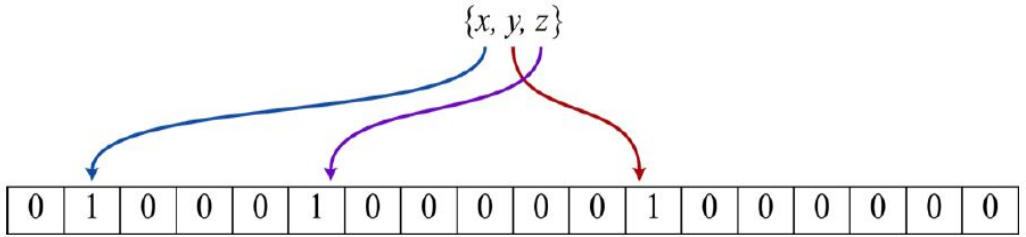


adapted from wikipedia

BF supports two operations:

- **Insertion:** for each key, get *n* positions from *n* hash functions. Set all these positions to 1
- **Query:** check bit value for *n* positions

Bloom filters



adapted from wikipedia

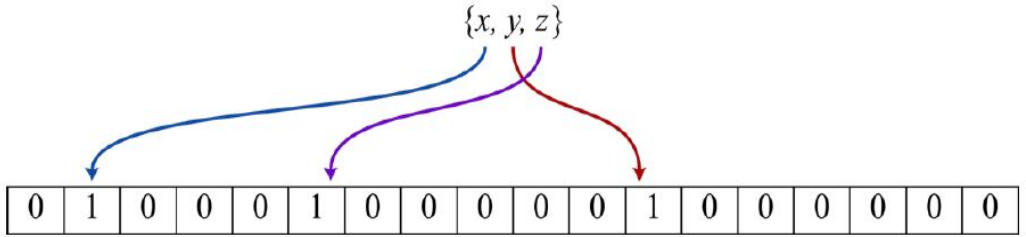
BF supports two operations:

- **Insertion:** for each key, get *n* positions from *n* hash functions. Set all these positions to 1
- **Query:** check bit value for *n* positions

Bloom filters from read set:

- Count k-mers
- For each k-mer: compute hashes and set corresponding bits

Bloom filters



adapted from wikipedia

BF supports two operations:

- **Insertion:** for each key, get *n* positions from *n* hash functions. Set all these positions to 1
- **Query:** check bit value for *n* positions

Bloom filters from read set:

- Count k-mers
- For each k-mer: compute hashes and set corresponding bits

Bloom filters construction issues:

- The largest bottleneck is the k-mer count step
- Bad data locality

} **motivations**

Bloom filters construction

For each dataset:
    Count and dump k-mers on disk

For each dataset:
    For each k-mer:
        Hash and set corresponding bit
    Dump bloom filter

BF are often built one by one because it's not always possible to have several filters in memory (one filter may corresponds to several GB)

Bloom filters construction

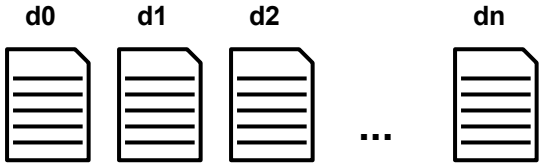
For each dataset:
    Count and dump k-mers on disk

For each dataset:
    For each k-mer:
        Hash and set corresponding bit
    Dump bloom filter

BF are often built one by one because it's not always possible to have several filters in memory (one filter may corresponds to several GB)

Can we build all these filters at the same time and with low memory?

kmtricks



int vectors

	d0	d1	-	dn
AAAC	7	0	-	1
AACC	12	54	-	1
AACG	0	0	-	32
-	-	-	-	-
TTTT	1	0	-	2

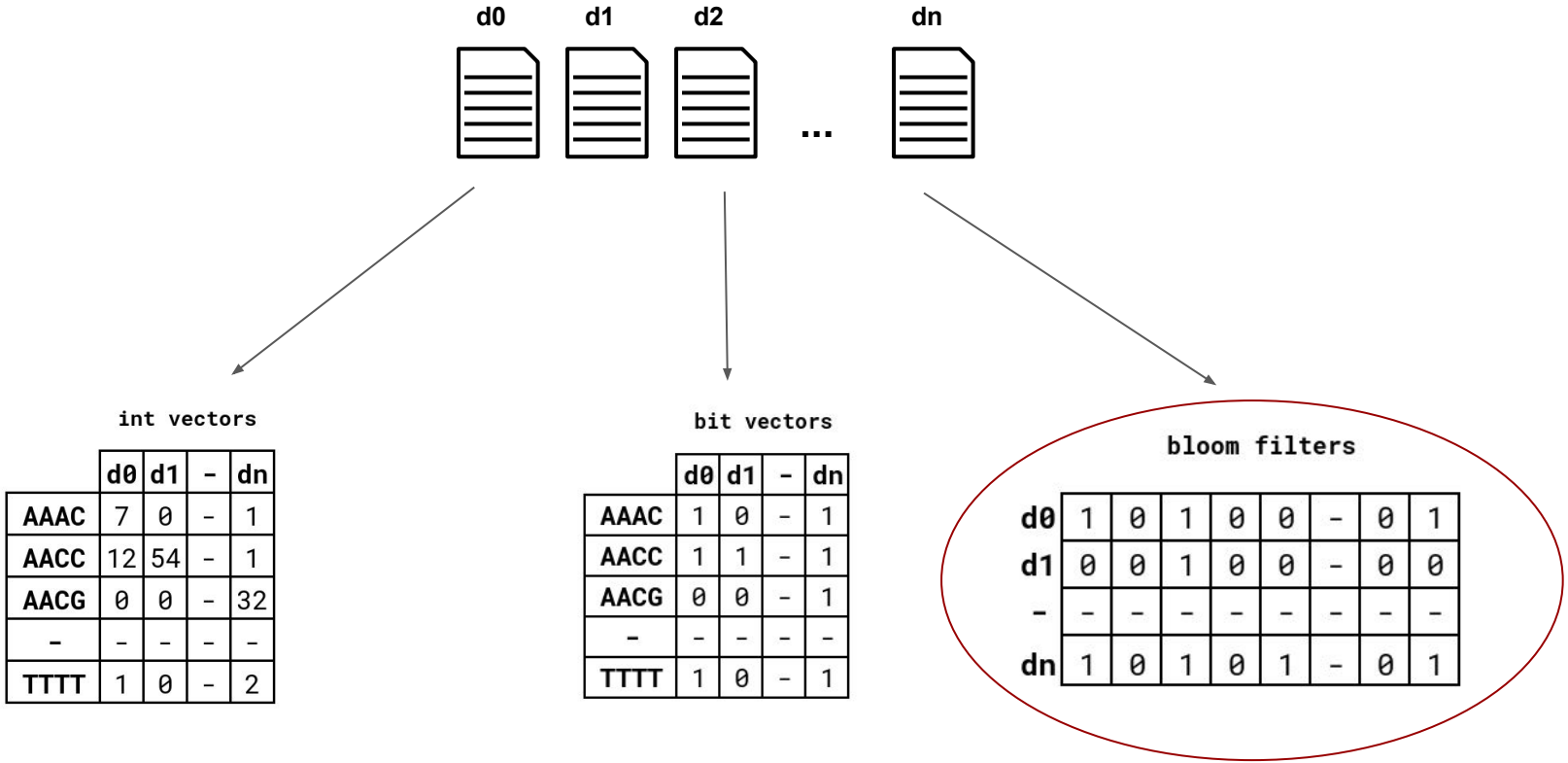
bit vectors

	d0	d1	-	dn
AAAC	1	0	-	1
AACC	1	1	-	1
AACG	0	0	-	1
-	-	-	-	-
TTTT	1	0	-	1

bloom filters

d0	1	0	1	0	0	-	0	1
d1	0	0	1	0	0	-	0	0
-	-	-	-	-	-	-	-	-
dn	1	0	1	0	1	-	0	1

kmtricks



	d0	d1	-	dn
AAAC	7	0	-	1
AACC	12	54	-	1
AACG	0	0	-	32
-	-	-	-	-
TTTT	1	0	-	2

	d0	d1	-	dn
AAAC	1	0	-	1
AACC	1	1	-	1
AACG	0	0	-	1
-	-	-	-	-
TTTT	1	0	-	1

d0	1	0	1	0	0	-	0	1
d1	0	0	1	0	0	-	0	0
-	-	-	-	-	-	-	-	-
dn	1	0	1	0	1	-	0	1

kmtricks: bloom filters construction

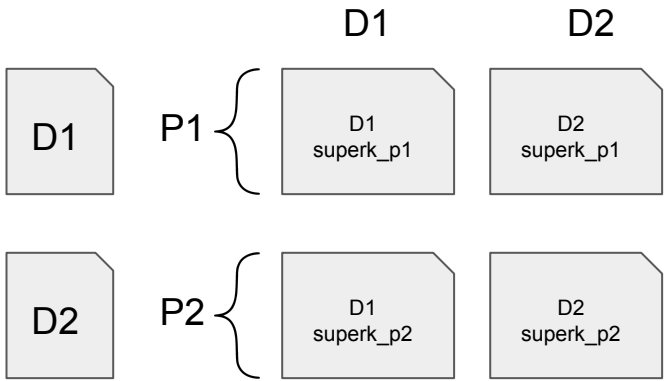
Step 1: Compute minimizers repartition

- Compute minimizers frequency
- Dispatch minimizers in *p* partitions.
- These partitions will contain the k-mers of our data sets. The idea is to have **an equivalent number of k-mers per partition**.

kmtricks: bloom filters construction

Step 2: Compute super-k-mers from reads

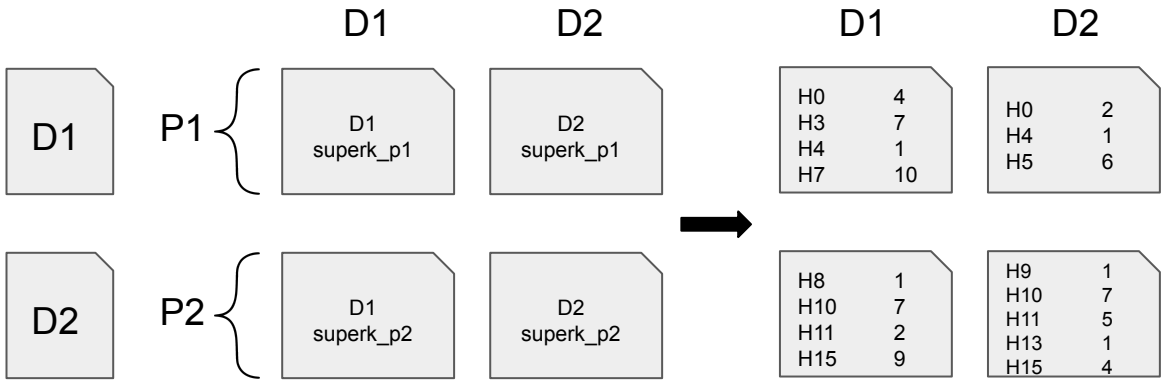
- Dispatch super-k-mers in their partitions according to their minimizers



kmtricks: bloom filters construction

Step 3: Sorting count algorithm

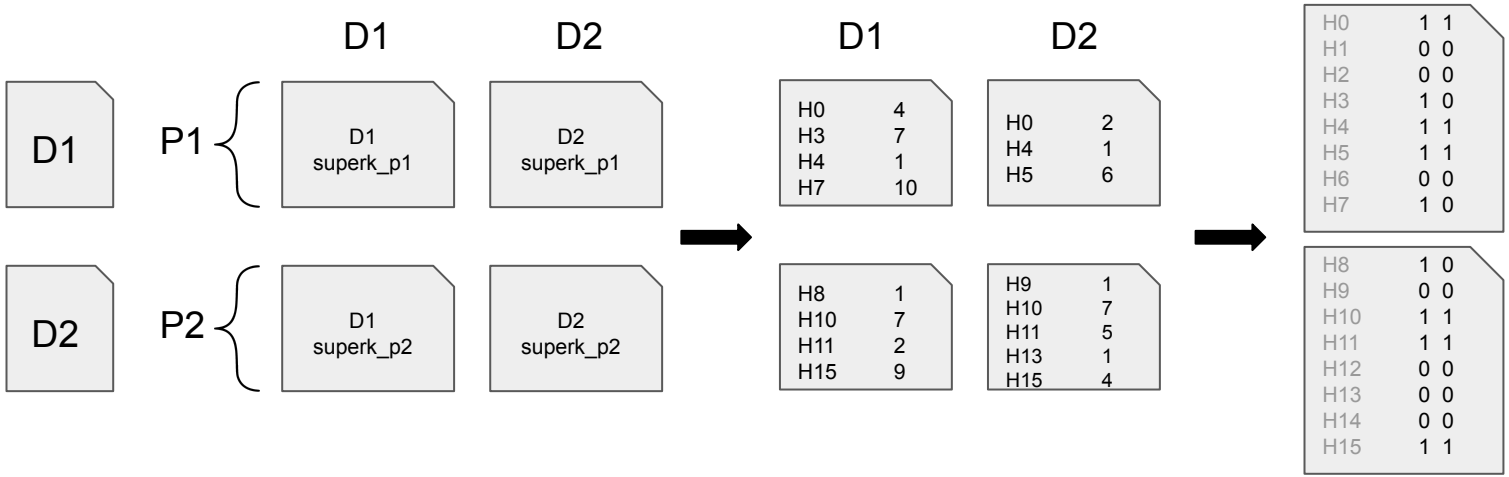
- Split super-k-mers into k-mers and hash them.
- Sort: the count is given by identical consecutive hashes.
- Hash spaces are **specific and consecutive** according to the partitions (== according to a set of minimizers).



kmtricks: bloom filters construction

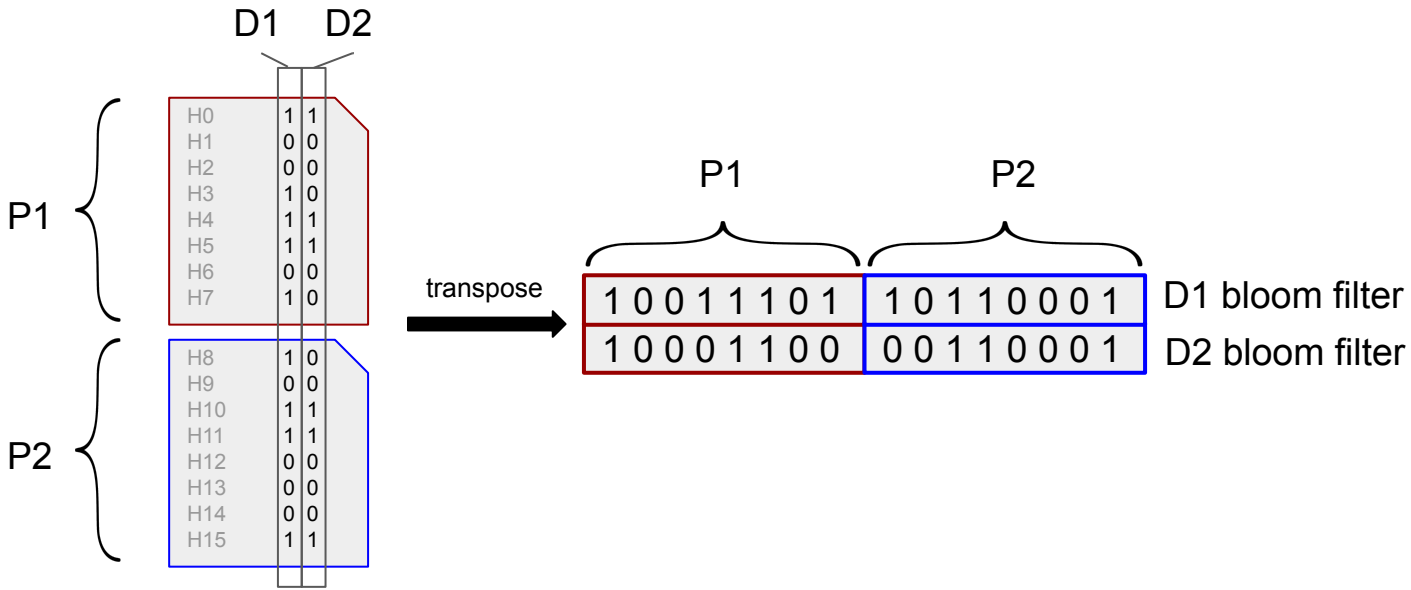
Step 4: Merge equivalent partitions between datasets

- Add **empty lines for missing hashes** (k-mers)
- Hashes are **not stored** but are given by line numbers



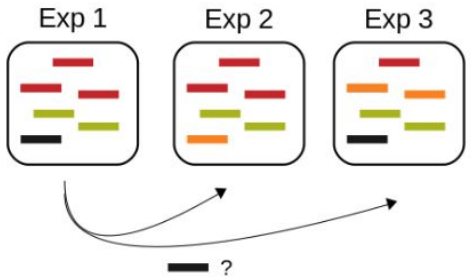
kmtricks: bloom filters construction

Step 5: Transpose each partition to obtains individual bloom filters



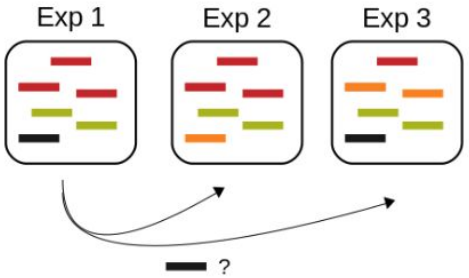
kmtricks: rare k-mers handling

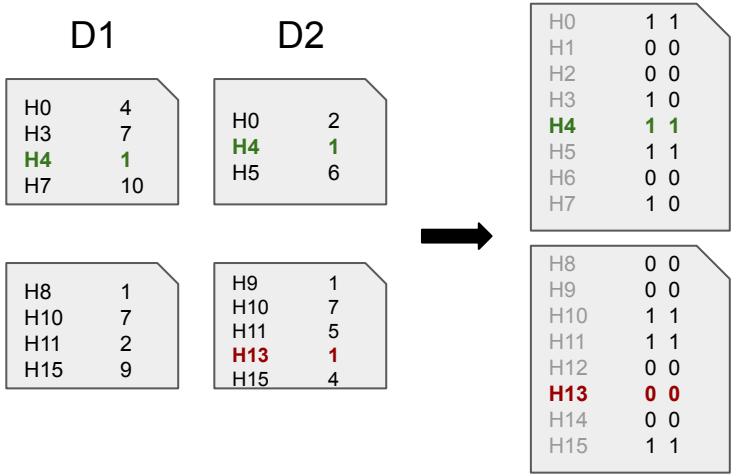
- Leverage information across samples during the merging step.
- **Salvage k-mers seen often but at low counts in datasets**



kmtricks: rare k-mers handling

- Leverage information across samples during the merging step.
- **Salvage k-mers seen often but at low counts in datasets**





kmtricks results

Indexing of 100 human RNA-seq read sets:

- Comparison vs HowDeSBT classical construction

	Time	Max memory	Max disk usage
HowDeSBT makebf	2h27	13.2 GB	55.1 GB
kmtricks	35min48s	3.5 GB	56.6 GB

kmtricks results

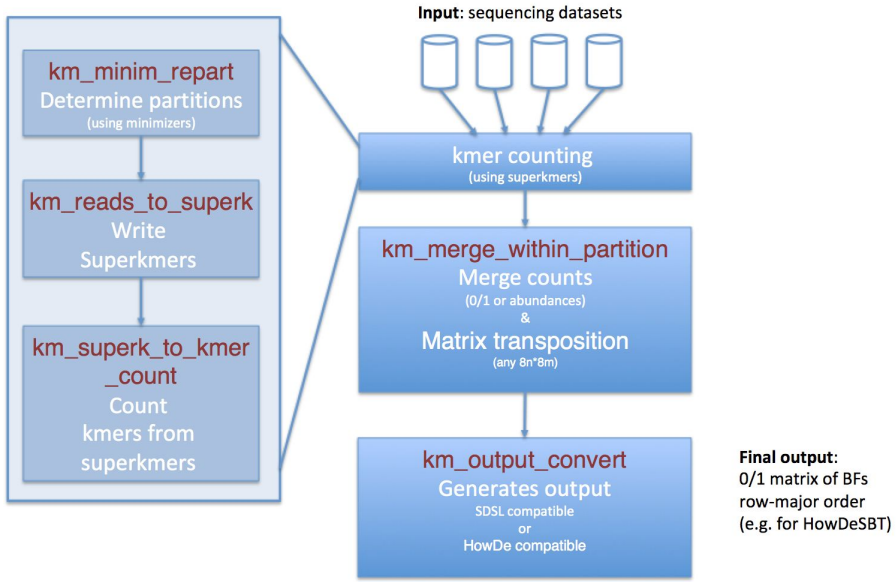
Indexing of 674 human RNA-seq read sets (> 1 TB gzip):

- Comparison vs HowDeSBT classical construction

	Time	Max memory	Max disk usage
HowDeSBT makebf	59h03	13.2 GB	206 GB
kmtricks	22h10	22 GB	1.5 TB
kmtricks w/o merge	17h56	21 GB	238 GB

kmtricks overview

Modular k-mer count matrix and Bloom filter construction for large read collections



Final output:
0/1 matrix of BFs
row-major order
(e.g. for HowDeSBT)

+

kmtricks library:
encoding, sequences representation,
k-way merge algorithm, bit matrix, SSE
transposition, streaming compression etc ...

https://github.com/tlemane/kmtricks

Conclusion & Future work

- Improves bf construction time but **it's still very insufficient to hope to scale up on the very large databases**

- Application on medium/large scale dataset: TARA Ocean (running)

- Take advantage of better data locality:
  - The query can be seen as a set of super-k-mers (corresponding to a **set of minimizers**)
  - For a query, we probably don't need the **whole set of partitions**.

Thank you 

References

N .Luhmann, et al. Blastfrost: Fast querying of 100,000 s of bacterial genomes in bifrost graphs. BioRxiv, 2020.

R. Wittler. Alignment-and reference-free phylogenomics with colored de bruijn graphs. Algorithms for Molecular Biology, 2020.

G. Holley, R. Wittler, and J. Stoye, “Bloom Filter Trie: An alignment-free and reference-free data structure for pan-genome storage,” *Algorithms Mol. Biol.*, vol. 11, no. 1, p. 3, 2016, doi: 10.1186/s13015-016-0066-8.

B. Solomon and C. Kingsford, “Fast search of thousands of short-read sequencing experiments,” *Nat. Biotechnol.*, vol. 34, no. 3, pp. 300–302, Mar. 2016, doi: 10.1038/nbt.3442.

B. Solomon and C. Kingsford, “Improved search of large transcriptomic sequencing databases using split sequence bloom trees,” in *Journal of Computational Biology*, 2018, vol. 25, no. 7, pp. 755–765, doi: 10.1089/cmb.2017.0265.

C. Sun, R. S. Harris, R. Chikhi, and P. Medvedev, “AllSome Sequence Bloom Trees,” *J. Comput. Biol.*, vol. 25, no. 5, pp. 467–479, 2018, doi: 10.1089/cmb.2017.0258.

R. S. Harris and P. Medvedev, “Improved representation of sequence Bloom trees,” *Bioinformatics*, 2019, doi: 10.1093/bioinformatics/btz662.

References

P. Pandey, F. Almodaresi, M. A. Bender, M. Ferdman, R. Johnson, and R. Patro, “Mantis: A Fast, Small, and Exact Large-Scale Sequence-Search Index,” *Cell Syst.*, vol. 7, no. 2, pp. 201-207.e4, Aug. 2018, doi: 10.1016/j.cels.2018.05.021.

Y. Yu *et al.*, “SeqOthello: querying RNA-seq experiments at scale,” *Genome Biol.*, vol. 19, no. 1, p. 167, Oct. 2018, doi: 10.1186/s13059-018-1535-9.

P. Bradley, H. C. den Bakker, E. P. C. Rocha, G. McVean, and Z. Iqbal, “Ultrafast search of all deposited bacterial and viral genomic data,” *Nat. Biotechnol.*, vol. 37, no. 2, pp. 152–159, Feb. 2019, doi: 10.1038/s41587-018-0010-1.

T. Bingmann, P. Bradley, F. Gauger, and Z. Iqbal, “COBS: a Compact Bit-Sliced Signature Index,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11811 LNCS, pp. 285–303, May 2019.

C. Marchet, C. Boucher, S. Puglisi, P. Medvedev, M. Salson, and R. Chikhi, “Data structures based on k -mers for querying large collections of sequencing datasets,” *bioRxiv*, p. 866756, Dec. 2019, doi: 10.1101/866756.