Efficient indexing of k-mer presence and abundance in sequencing datasets

Camille Marchet[1], Zamin Iqbal[2], Daniel Gautheret[3], Mikaël Salson[1] and Rayan Chikhi[4]

[1] Univ. Lille, CRIStAL, France
[2] European Bioinformatics Institute, Cambridge, UK
[3] Paris-Saclay University, I2BC, France
[4] Institut Pasteur, CNRS, France

VanBUG seminar, Vancouver, 2020

You may have seen this talk before..

A shorter version was made by Camille Marchet for

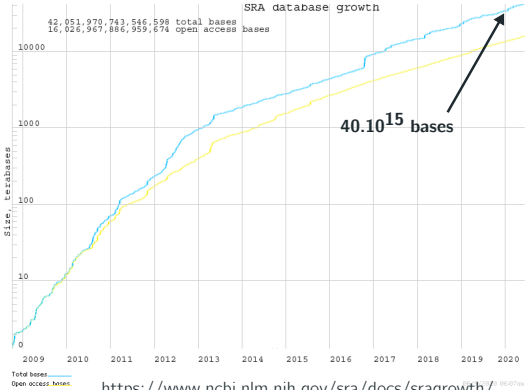
- ISMB 2020
- BiATA 2020
- Jobim 2020

The '`diff`' with today's talk is:

- Expanded intro (why we do this)
- More context around the techniques
- Expanded outro (applications, Serratus)

Not so many slides. Feel free to ask/write questions during.

Analyzing all available raw sequences is a challenge



https://www.ncbi.nlm.nih.gov/sra/docs/sragrowth/

- Petabases of raw reads stored at EBI/NCBI
- Cannot download them all, nor perform sequence search

YouTube: 100-1000 PB

NCBI SRA database : 30 PB

Institut Pasteur: 8 PB

Your laptop: 0.001 PB

Cold Spring Harbor Laboratory
bioRχiv
THE PREPRINT SERVER FOR BIOLOGY

New Results

Petabase-scale sequence alignment catalyses viral discovery

Robert C. Edgar, Jeff Taylor, Tomer Altman, Pierre Barbera, Dmitry Meleshko, Victor Lin, Dan Lohr, Gherman Novakovsky, Basem Al-Shayeb, Jillian F. Banfield, Anton Korobeynikov, Rayan Chikhi, Artem Babaian

doi: https://doi.org/10.1101/2020.08.07.241729

Cold Spring Harbor Laboratory

bioRχiv

THE PREPRINT SERVER FOR BIOLOGY

New Results

Petabase-scale sequence alignment catalyses viral discovery

Robert C. Edgar, Jeff Taylor, Tomer Altman, Pierre Barbera, Dmitry Meleshko, Victor Lin, Dan Lohr, Gherman Novakovsky, Basem Al-Shayeb, Jillian F. Banfield, Anton Korobeynikov, Rayan Chikhi, Artem Babaian

doi: https://doi.org/10.1101/2020.08.07.241729

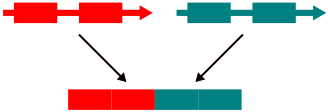
* But also those projects: Recount2, Toil, ARCHS4, MetaSRA, ...

A Tera increase in sequencing production in the past 25 years		
Genes & Operons	1990	**Kilo** = 1,000
Bacterial genomes	1995	**Mega** = 1,000,000
Human genome	2000	**Giga** = 1,000,000,000
Human microbiome	2005	**Tera** = 1,000,000,000,000
50K Microbiomes	2015	**Peta** = 1,000,000,000,000,000
what is expected for the next 15 years ? (a Giga?)		
200K Microbiomes	2020	**Exa** = 1,000,000,000,000,000,000
1M Microbiomes	2025	**Zetta** = 1,000,000,000,000,000,000,000
Earth Microbiome	2030	**Yotta** = 1,000,000,000,000,000,000,000,000

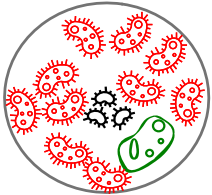
Source: @kyrpides

Sequence databases: a vast and blooming application range

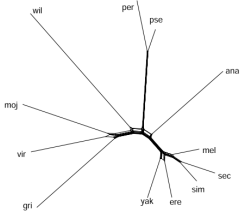
fusion events
in 10,000 RNA TGCA datasets
[Yu et al. '18]



antimicrobial resistance genes
in >400,000 bacterial and viral genomes
[Bradley et al. '19]



alignment and reference-free
phylogenetic network inference
[Wittler '20]



Problem statement

a set of datasets $\{d_1, d_2, \ldots d_n\}$
(raw reads)

query sequence

return all d_i's where the query occurs

Problem statement

a set of datasets $\{d_1, d_2, \ldots d_n\}$
(raw reads)

query sequences

for each s, return all d_i's where the query occurs

An approximation, using *k*-mers

same datasets, but seen as ***k*-mer sets**

same query, but converted to a **set of *k*-mers**

report d_i's **containing sufficiently enough (≥t) *k*-mers from the query** [Solomon & Kingsford '16]

Presence/absence queries

same datasets, but seen as ***k*-mer sets**

same query, but converted to a **set of *k*-mers**

example, t = 4
return [1 0 1 0 ...]

Abundance queries

same datasets, but seen as ***k*-mer sets**
k-mers are counted (e.g. with KMC)

same query, but converted
to a **set of *k*-mers**

return query abundance in each d_i's
example: return [5 0 4 0 ...]

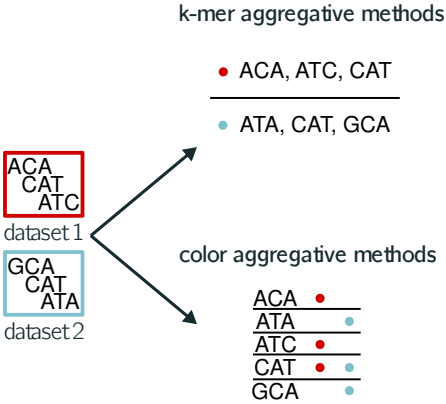
Concretely

- # datasets: $n = 10^3$ to 10^7
- $k = 16$ to 64
- # k-mers per dataset: 10^6 to 10^{10}
- Abundances 1 to $2^{16} - 1$

Elementary query:

- given a k-mer, return whether (/how many times) it occurs in d_i

Literature for presence/absence query

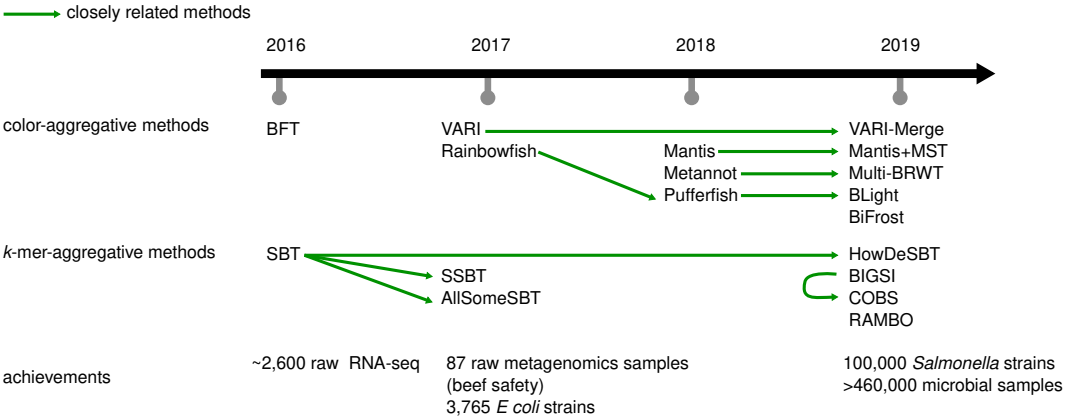


Solomon and Kingsford '16 (SBT)
Sun et al. '18 (SSBT)
Solomon et al. '18 (AllSome SBT)
Bradley et al. '19 (BIGSI)
Harris and Medvedev '19 (HowDeSBT)
Bingmann et al. '19 (COBS)

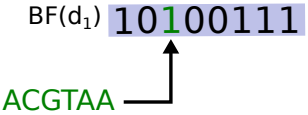
Holley et al. '16 (BFT)
Muggli et al. '17 (VARI)
Almodaresi et al. '17 (Rainbowfish)
Pandey et al. '18 (Mantis)
Yu et al. '18 (SeqOthello)
Mustapha et al. '18 (Metannot)
Almodaresi et al. '19 (Mantis+MST)
Karasikov et al. '19 (Multi-BRWT)
Muggli et al. '19 (VARI-Merge)
Holley and Melsted '19 (BiFrost)

See our survey *Data structures based on k-mers for querying large collections of sequencing datasets* [Marchet *et al.*, Gen Res 2020, to appear]

Timeline of methods



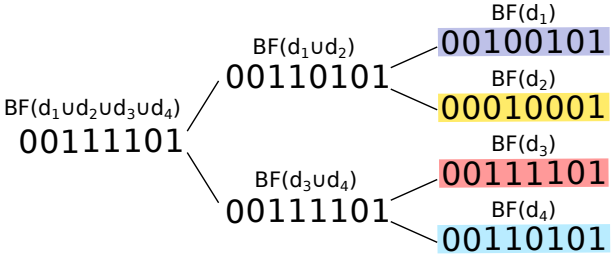
Bloom filters



Determines if a k-mer (here, ACGTAA) is present in d_1 in constant time.

SBT

Sequence Bloom Trees [Solomon & Kingsford, '16]



Determines if a k-mer is present in $d_1, \ldots, d_n$ in time sublinear in n.

BIGSI

Bit-sliced Genomic Signature Index [Bradley *et al*, '19]

BF(d_1)	10100111
BF(d_2)	10010001
BF(d_3)	00111101
BF(d_4)	01110101

Determines if a k-mer is present in $d_1, \ldots, d_n$ by accessing 1 local piece of information.

State of the art of k-mer based indexing

BIGSI achieves very fast queries! Are we done then?

State of the art of k-mer based indexing

BIGSI achieves very fast queries! Are we done then?

No

State of the art of k-mer based indexing

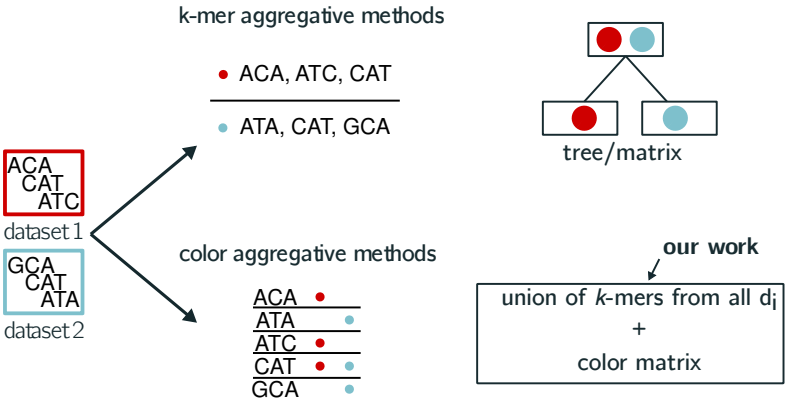
BIGSI achieves very fast queries! Are we done then?

No

Each method has its own drawbacks, and none records abundances.

Also: they all seem limited to $\sim 10^4$ RNA-seqs, or $\sim 10^5$ bacterial WGS.

Literature for presence/absence query



None of these methods handle abundances

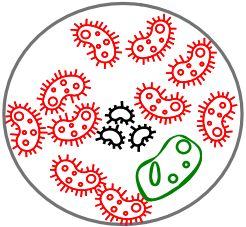
Example of motivation for abundance queries

query, e.g. a mutation in a strain

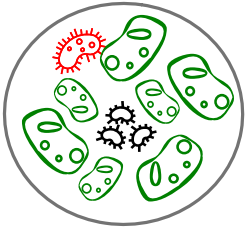
TGATACATCGG

k-mers

...
ATACA
TACAT
ACATC
...



dataset 1: prevalent



dataset2: rare

REINDEER: bird's eye view

- Pre-requisite: count each k-mer in all d_i's
- Then:
  1. Record the union set of all k-mers
  2. Associate each k-mer to a count vector (e.g. ACT to [13 0 4 ...])
  3. Store those vectors (the count matrix)

color aggregative method

	d_1	d_2	d_3	...
ACT	●		●	
ATA		●	●	
ATC	●			
CAT	●	●		
...				

count matrix

	d_1	d_2	d_3	...
ACT	13	0	4	
ATA	0	12	30	
ATC	12	0	0	
CAT	11	7	0	
...				

Building blocks in REINDEER

	d1	d2	d3
1	13	0	4
2	0	12	30
...			

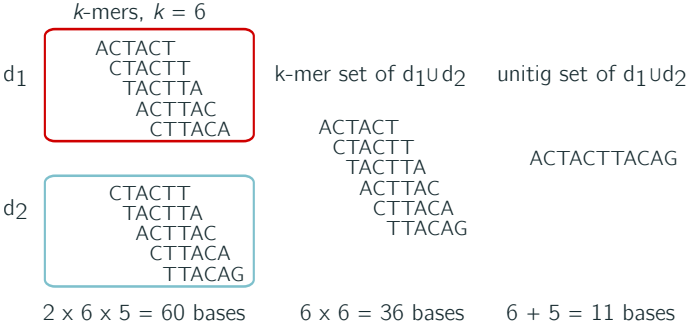
efficient *k*-mer set representation

specialized hash table for *k*-mers

compact count matrix

Block 1: *k*-mer set representation

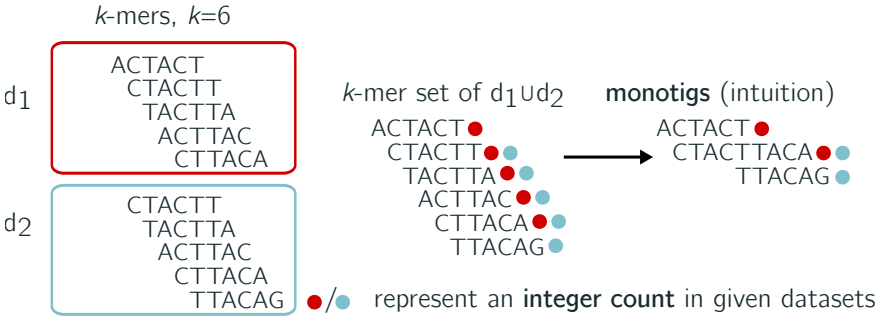
- Unitigs



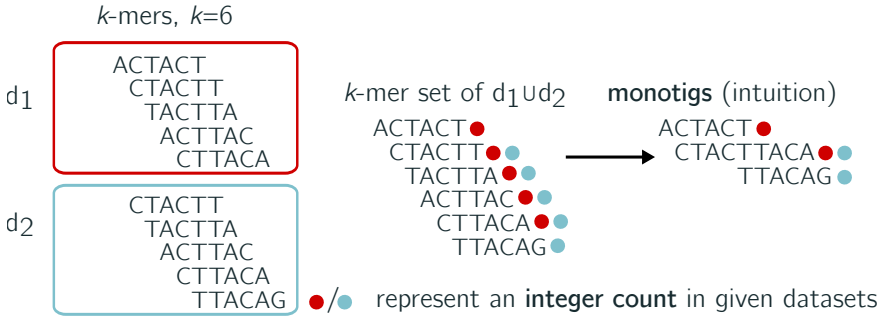
- Simplitigs [Brinda et al. '20], UST [Rahman & Medvedev '20]
  Spectrum preserving string sets (SPSS) are fascinating techniques to compactly *store* *k*-mer sets.

However: not an index (can only list, not search) & doesn't keep **abundances**

Block 1: Monotigs: an SPSS preserving counts info



Block 1: Monotigs: a new SPSS preserving count information



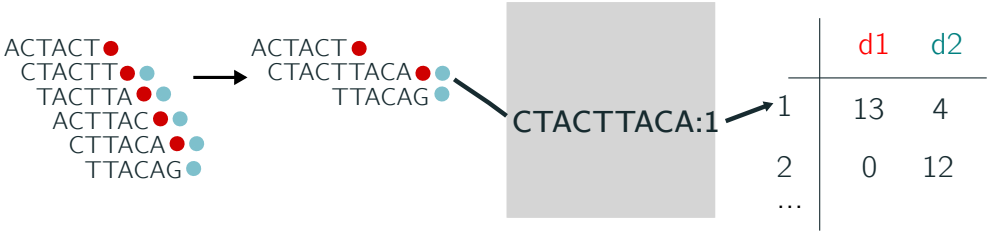
To construct:

- *k*-mers are greedily assembled within minimizer partitions

Block 1: Monotigs in REINDEER

here ●● means $\begin{array}{cc} d_1 & d_2 \\ [13 & 4] \end{array}$

k-mer set from all datasets → **monotigs** → **associate monotig to count vector** → **count matrix**

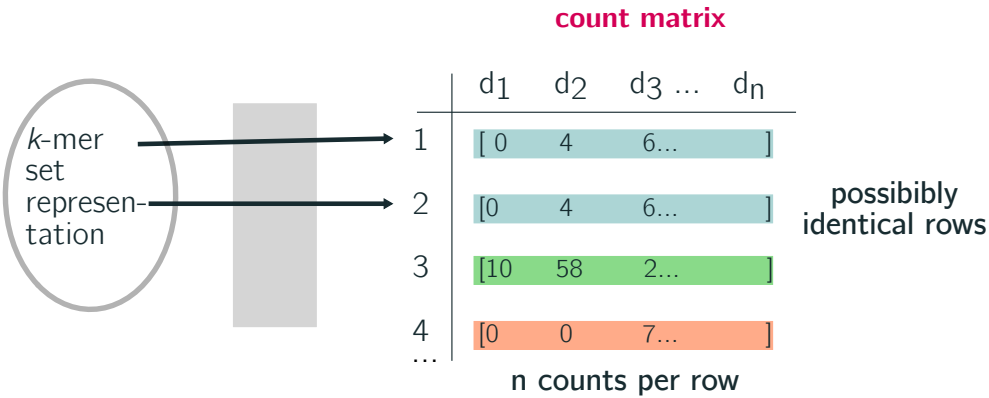


each k-mer accesses its count through its monotig index

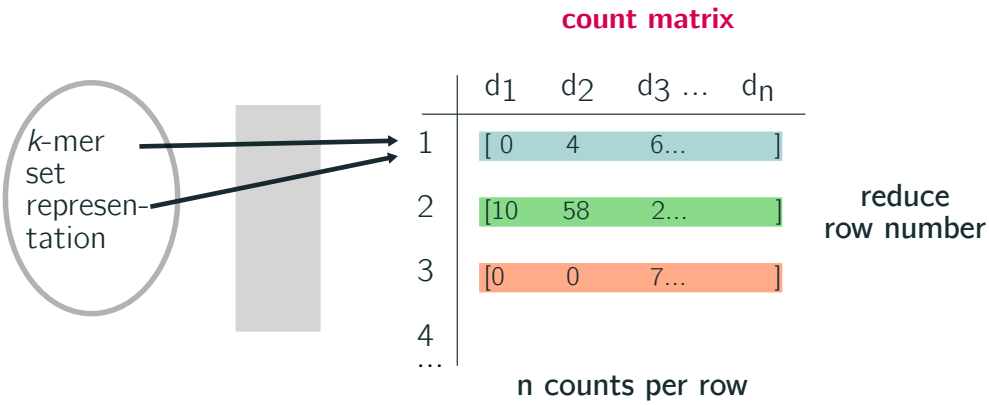
Block 2: Specialized k-mer hash table

- Minimal perfect hashing (MPHF):
  - BBHash [Limasset & Rizk *et al*, '17]
  - A very compact key $\rightarrow$ value store
  - ..which forgets about the keys
  - $\approx$ 3 bits per key
- BLight [Marchet *et al*, '20]
  - Build unitigs (or monotigs)
  - Extract super-k-mers (consecutive k-mers sharing a minimizer)
  - Index k-mer within them using a MPHF

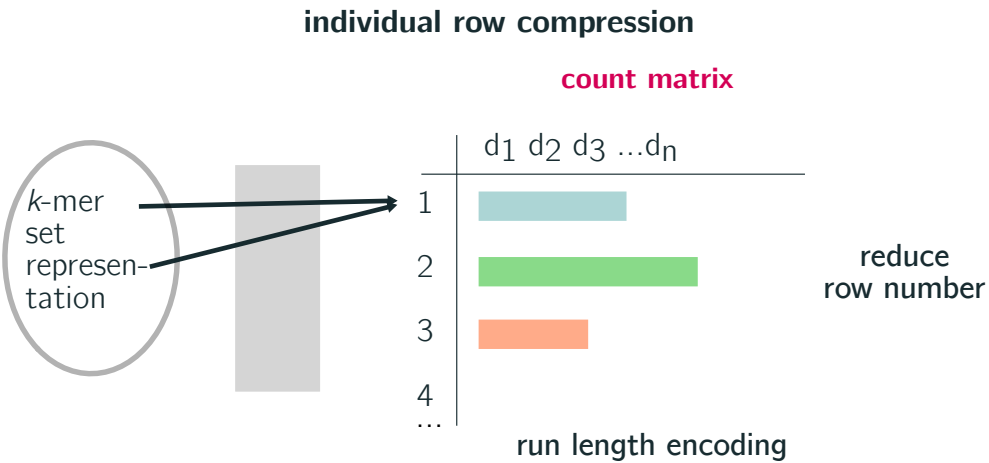
Block 3: Optimizing count matrix space



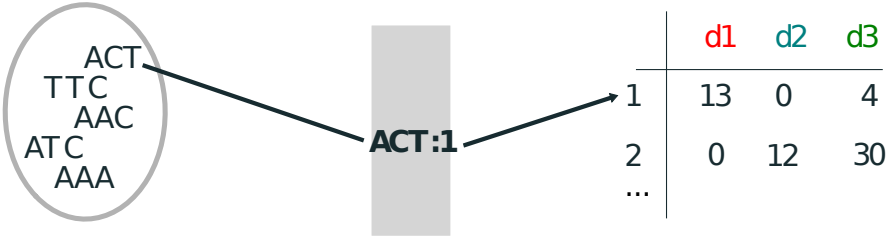
Block 3: Shrinking the count matrix



Block 3: Shrinking the count matrix



REINDEER



efficient *k*-mer set representation

specialized hash table for *k*-mers

compact count matrix

REINDEER's GitHub

github.com/kamimrcht/REINDEER

Index construction

Starting with read files (raw or gzipped fasta/fastq)

Make sure you have installed Bcalm by doing `sh install`. Let's assume you work with two files, `reads_1.fastq` and `reads_2.fastq`. The first thing needed to is to create a file of file (fof) that record the path to the reads. An example file can be found here: `test/fof.txt` Then this file of file is provided for the index construction:

```
./Reindeer --index -f test/fof.txt --bcalm
```

- **Input:** FASTA/Q files (or unitigs from BCALM2)
- **Output:** 1 line per query sequence, with positional counts per dataset
- REINDEER's index can be **serialized on disk** (for re-use)

REINDEER on 2,585 human RNA-seqs

	Tool	Counts	Time (h)
k-mer aggregative	SBT	No	55
	HowDeSBT	No	10
	BIGSI	No	N/A
color aggregative	Mantis	No	20
	SeqOthello	No	2
	REINDEER - presence/absence	No	40
	REINDEER - counts	**Yes**	**45**

REINDEER on 2,585 human RNA-seqs

	Tool	Counts	Time (h)	Peak RAM (GB)
k-mer aggregative	SBT	No	55	25
	HowDeSBT	No	10	N/A
	BIGSI	No	N/A	N/A
color aggregative	Mantis	No	20	N/A
	SeqOthello	No	2	15
REINDEER - presence/absence		No	40	27
REINDEER - counts		**Yes**	**45**	**56**

REINDEER on 2,585 human RNA-seqs

	Tool	Counts	Time (h)	Peak RAM (GB)	Index Size (GB)
k-mer aggregative	SBT	No	55	25	200
	HowDeSBT	No	10	N/A	15
	BIGSI	No	N/A	N/A	145
color aggregative	Mantis	No	20	N/A	30
	SeqOthello	No	2	15	20
	REINDEER - presence/absence	No	40	27	36
	REINDEER - counts	**Yes**	**45**	**56**	**52**

- Final index size is $\sim$ 1% of initial raw data (5TB)
- Few seconds per query

Honest chat about the limitations of REINDEER

Can it index the whole SRA?

- Starts to be impractical at around 10,000 RNA-seqs.

Potential directions:

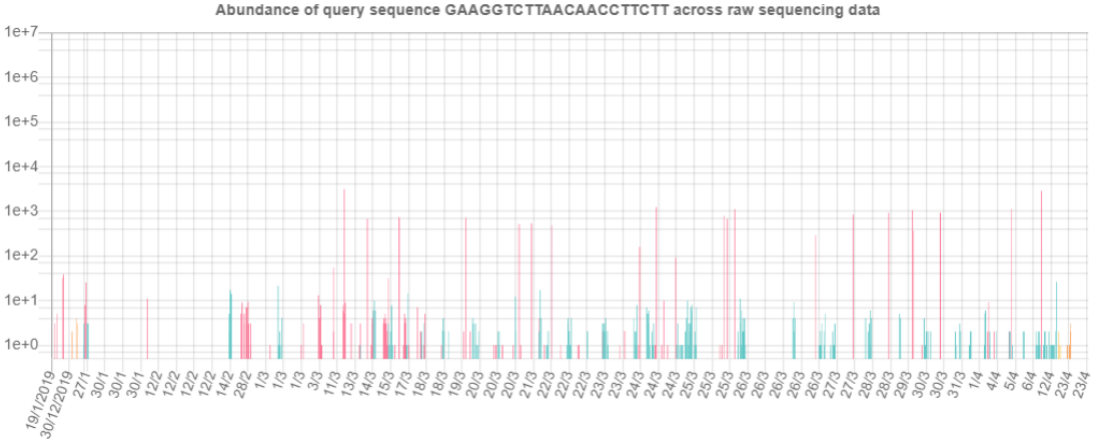
- Index chunks of $\sim$ 5,000 datasets at a time and distribute queries
- Design an even more scalable index
- Somehow compress or pre-process the input data

CoViDEER

Index of SARS-CoV-2 data
(∼2,000 amplicons + RNAseqs)
hosted at Institut Pasteur

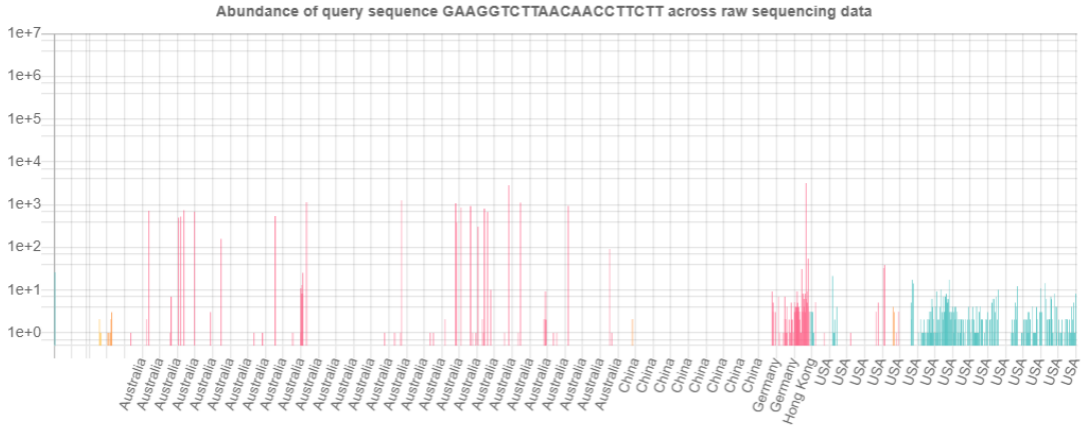
`covid19seqsearch.pasteur.cloud`

CoViDEER, the deletion at NC_045512v2:1605-1607



x axis = datasets, y axis = abundance of query

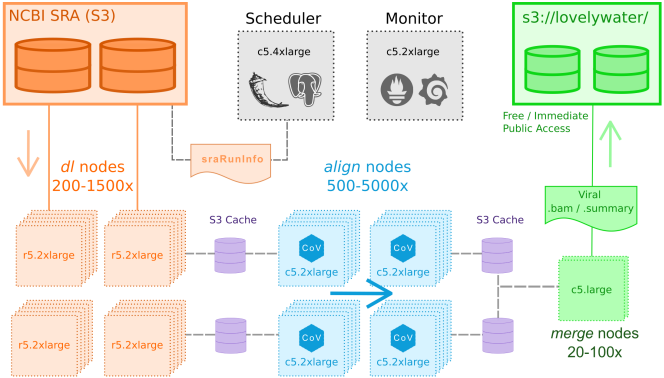
CoViDEER, the deletion at NC_045512v2:1605-1607



x axis = datasets, y axis = abundance of query

Serratus: Petabase-scale alignment to discover novel coronaviruses

No index? Just download and align all of SRA.



More details: or [Edgar *et al*, bioRxiv, 2020]

Perspectives

- Obvious long-term: index everything
- Medium-term: have more scalable indexing techniques

Tools I'd like to have today:

- efficient compression of *collections* of unitigs
- fast compressed random access to huge integer matrix
- fast BWT construction for terabase-scale (redundant) input

Acknowledgments

- Camille Marchet for a majority of these slides (those with bold titles)
- VanBUG organizers, Baraa Orabi, Faraz Hach for the invite
- REINDEER developers (& TRANSIPEDIA members)
- BLight developers
- HowDeSBT team
- Serratus team
- *k*-mer people