REINDEER: efficient indexing of k-mer presence and abundance in sequencing datasets

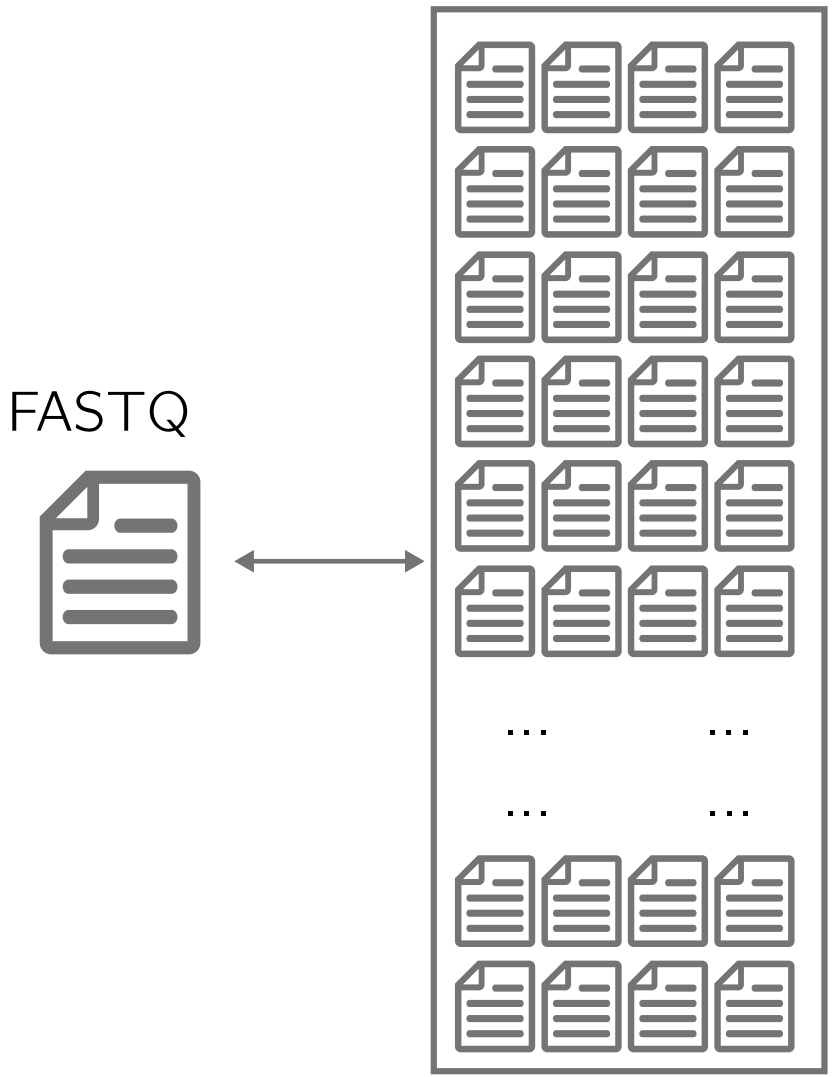
Camille Marchet, Zamin Iqbal, Mikaël Salon, Rayan Chikhi

DSB'20 – Rennes

Context

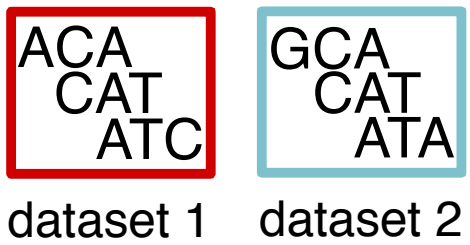
raw reads datasets

FASTQ

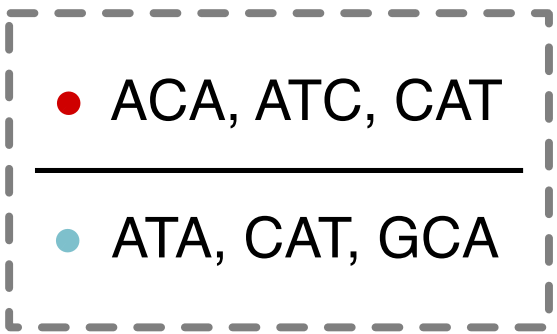


Sets of k-mer sets

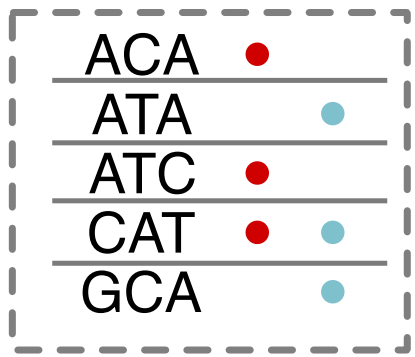
18 related papers and counting since 2016



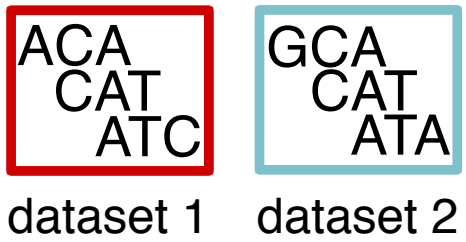
k-mer aggregative method



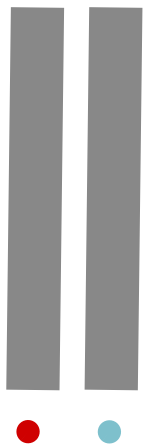
color aggregative method



Sets of k-mer sets

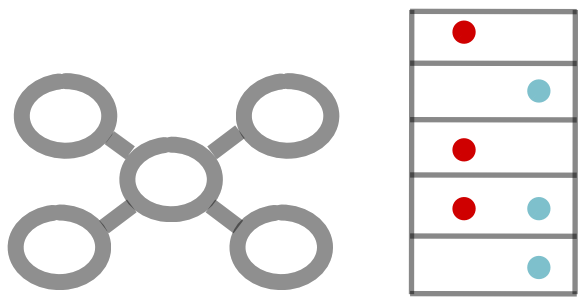


BIGSI



Good performances due to FP tradeoff
Presence/absence

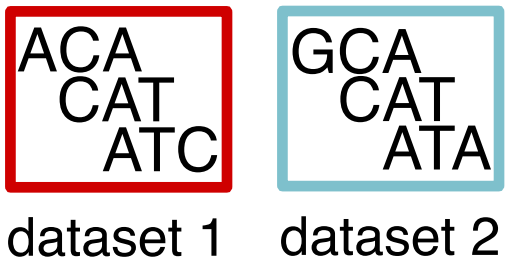
VARI (Muggli et al. 17)

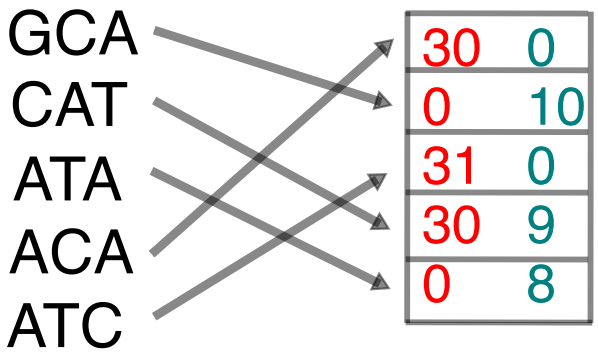


De Bruijn graph representation
Presence/absence + bubble calling

Our goal

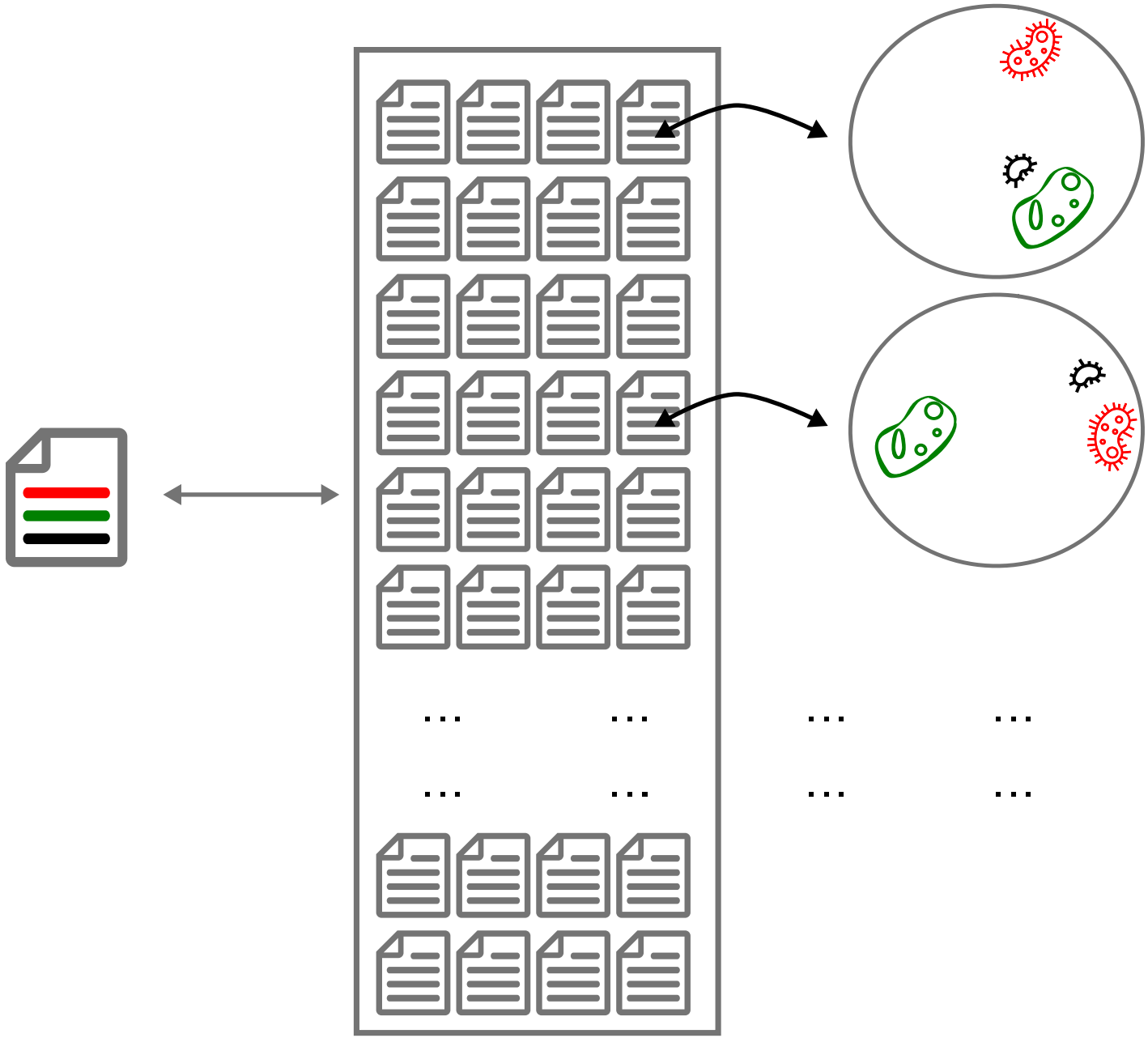
REINDEER method:
Query abundances of sequences in a collection of datasets of raw reads



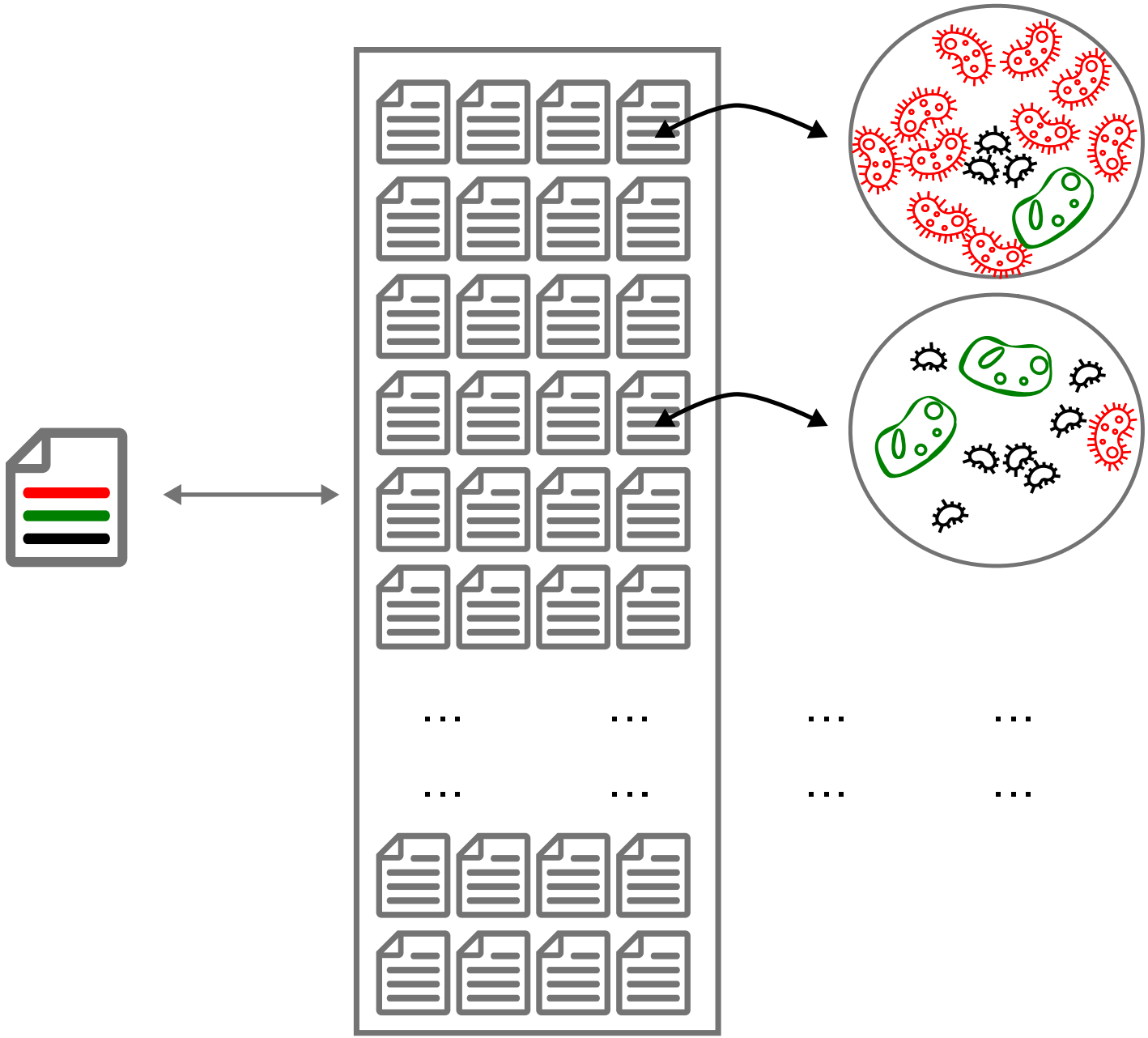


Set of k-mers from all datasets
+
abundance matrix

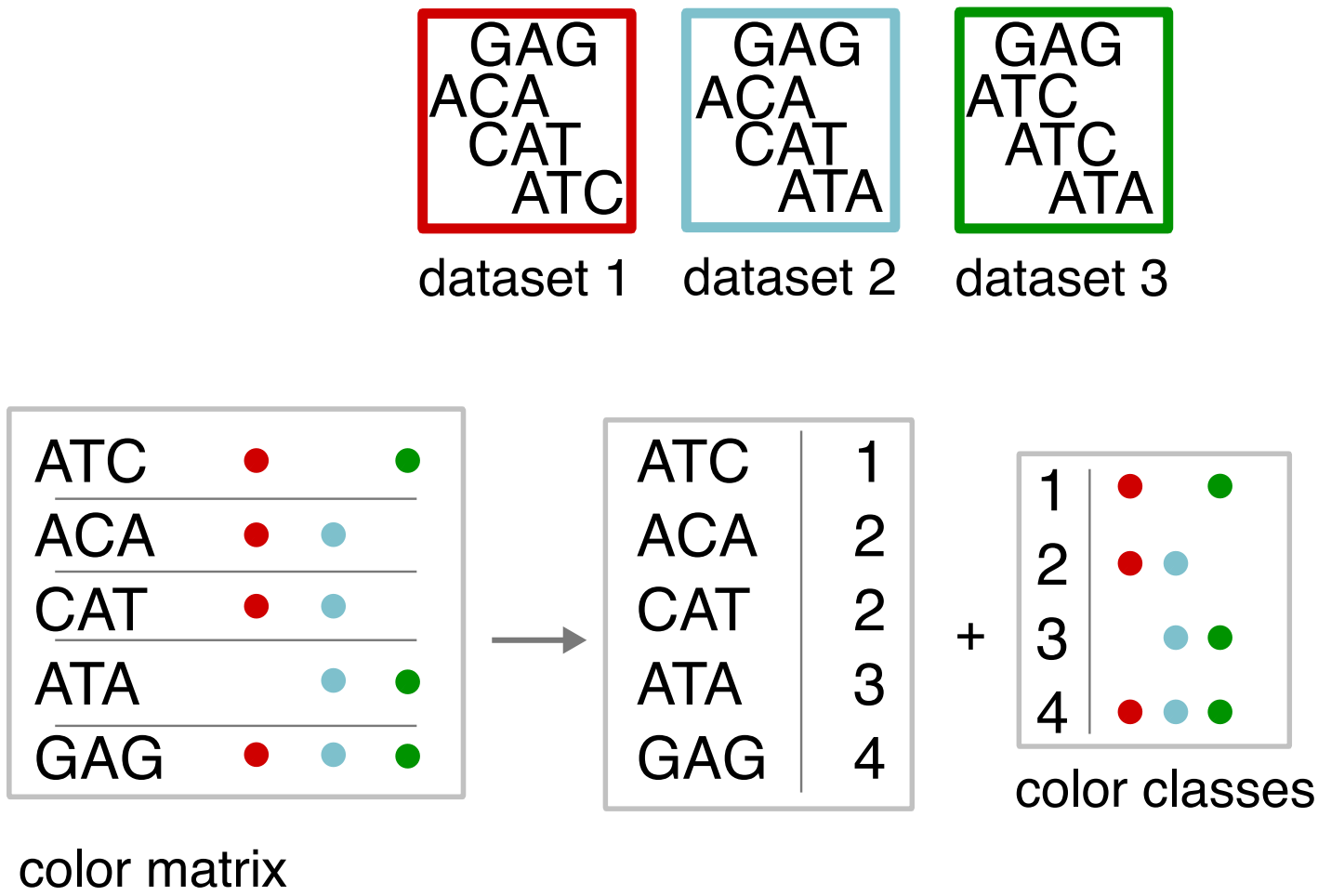
Motivation



Motivation



Color matrix



Abundance matrix

dataset 1 dataset 2 dataset 3

ATC	5		12
ACA	5	20	
CAT	4	21	
ATA		20	15
GAG	5	18	12

count matrix

equivalence classes for counts ?

compression (sparse matrix)

Definitions

dataset

CAGCT AGCTA
ATTTA TATTT
ACTTA

a raw read multiset
we see it as a set of k-mers

count vector

vec[x,i] = count of x in dataset i

abundance matrix

10	0	3	...
2	5	13	...
10	2	3	...
...			

a list of count vectors for each x

Definitions

datasets

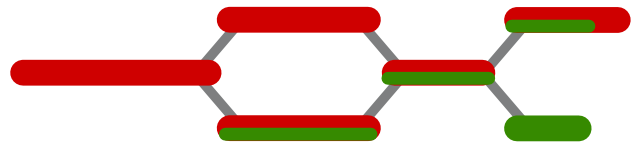
CAGCT AGCTA
TATTT
CTTAT

CAGCT AGCTA
ATTTA TATTT
ACTTA

De Bruijn graph

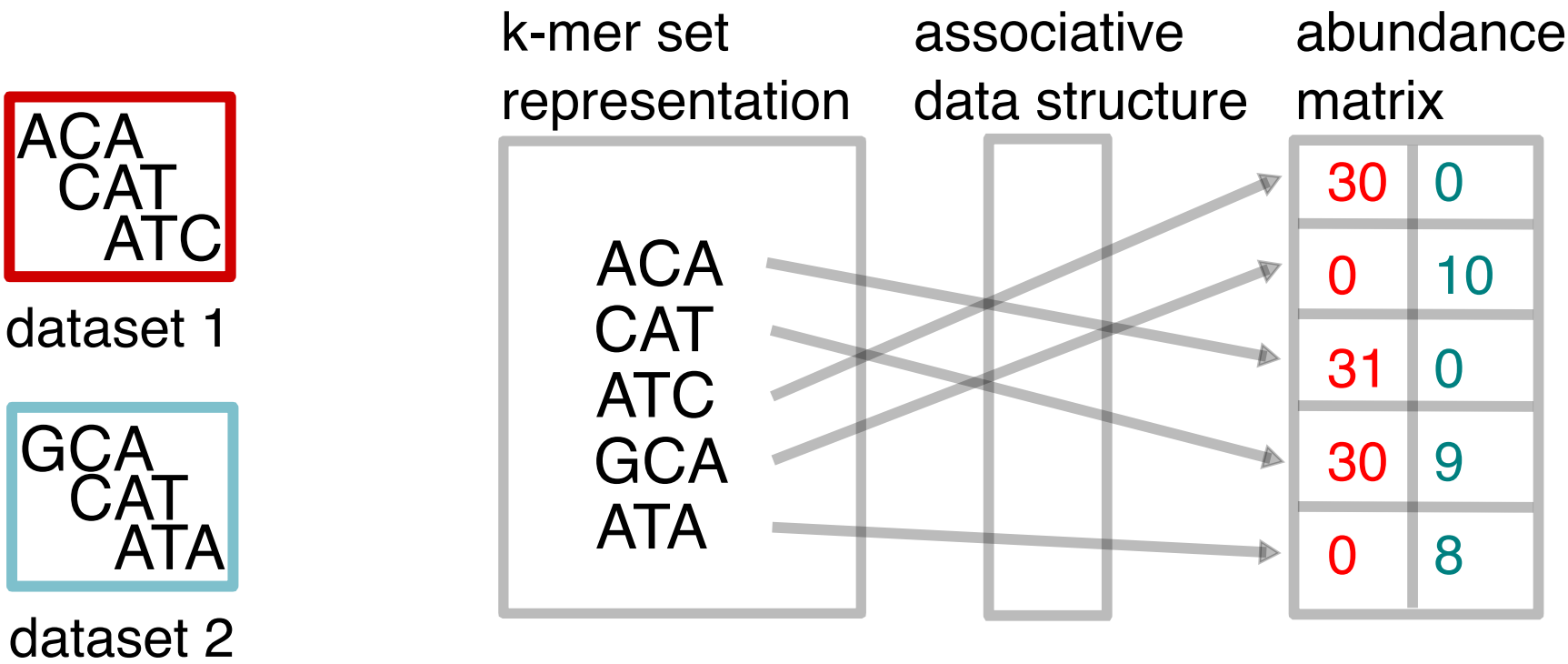
In practice we use a compacted DBG (graph of unitigs)

union De Bruijn graph



represents the set of k-mer sets coming from all datasets

Required building blocks



Associative structure

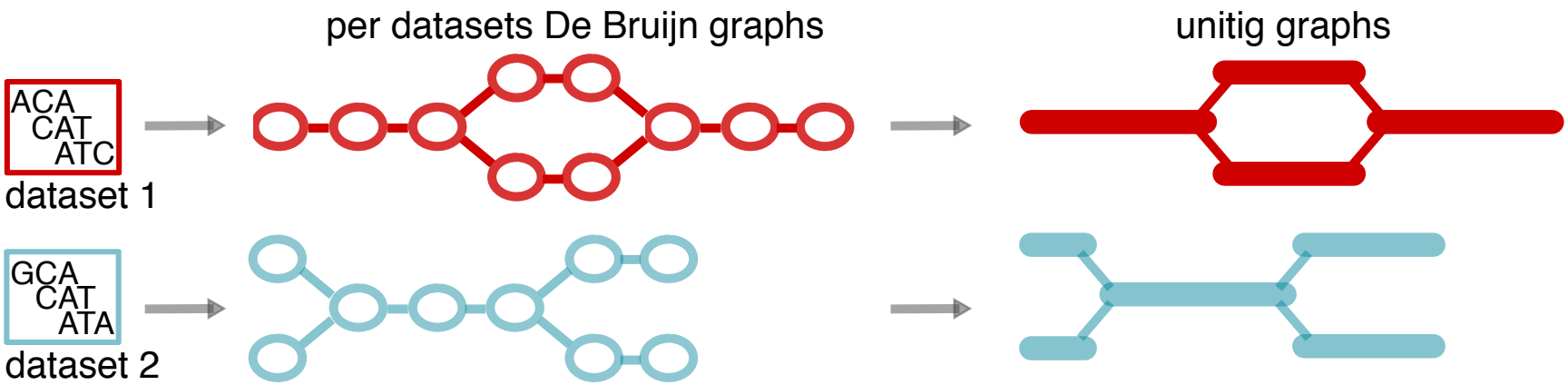
- Compact & compressed indices
  for efficient **exact string matching**
  (FM-index [Ferragina & Manzini '05])

- k-mer indices
  - approximate membership query (**AMQ**)
    (Bloom filters, Othello [Yu et al. '18] ,
    Quasi-dictionary [Marchet et al. '16])
  - **associative indices**
    (Couting Quotient Filters [Pandey et al. '17],
    MPHF [Almodaresi et al '17])

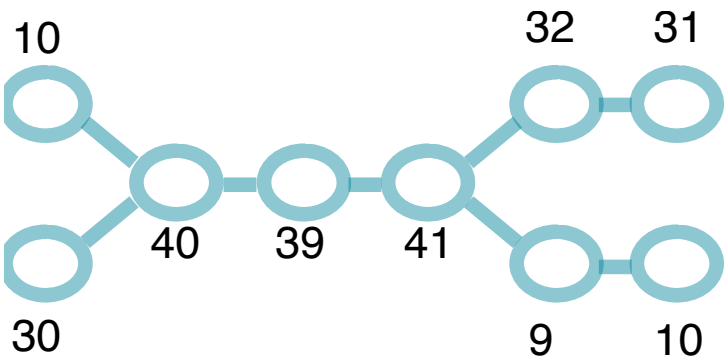
+ minimizers ⟶ **BLight** [Marchet et al. '19]

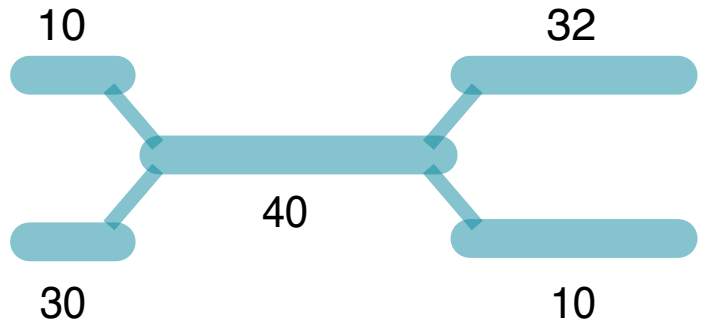
	nb. 31-mers	Pufferfish (time/mem)	BLight (tim/mem)
human	2.5 billions	1 h/20 GB (12.5 GB for the index)	**30 min/8 GB** ($\approx$ 26 bits/k-mer)

K-mer counts


per datasets De Bruijn graphs
unitig graphs
ACA
CAT
ATC
dataset 1
GCA
CAT
ATA
dataset 2

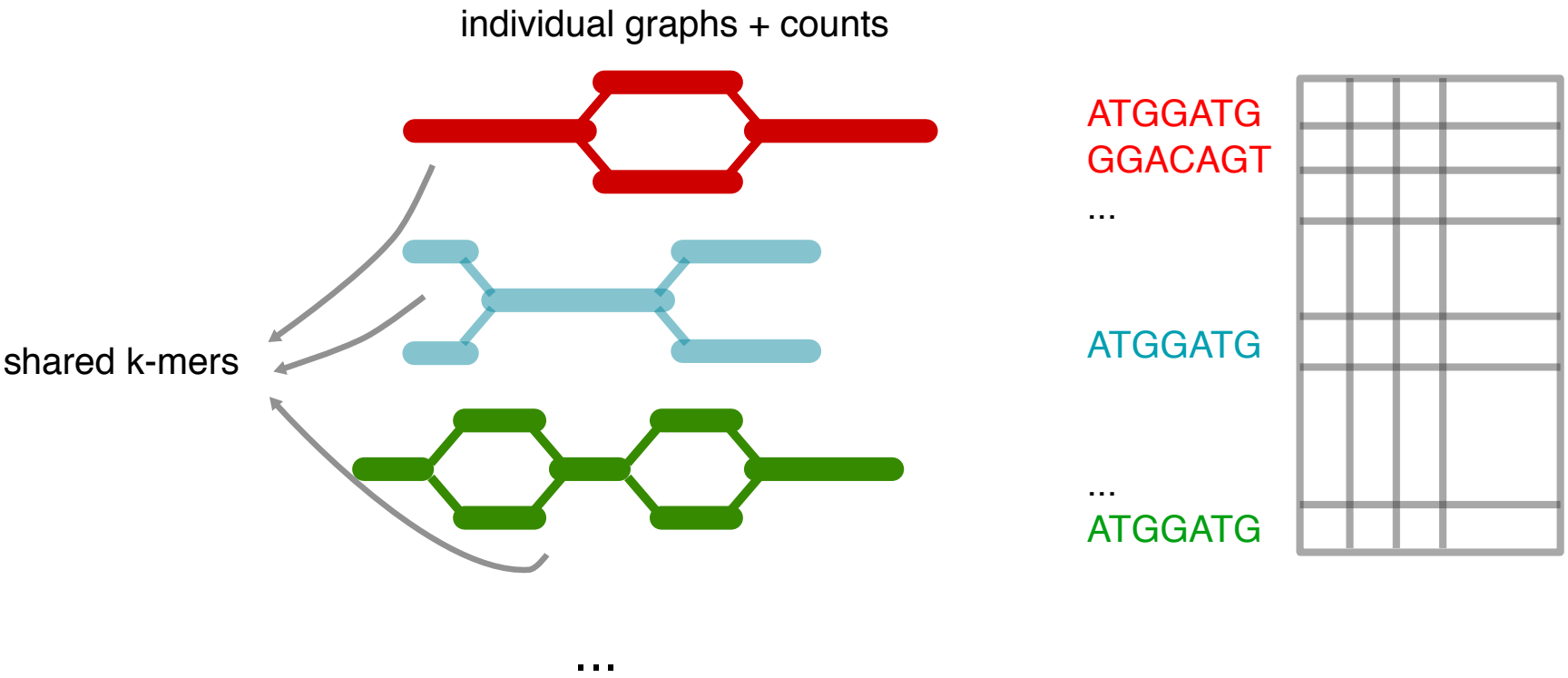
K-mer counts





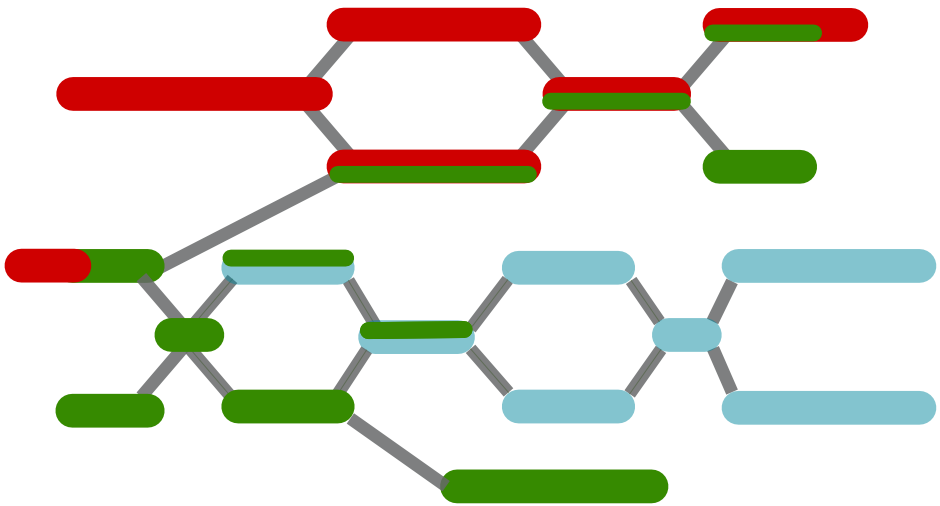
- Good approximation of k-mer counts
- Record more redundant values
- Smooth counts due to sequencing errors

Associate counts to kmers

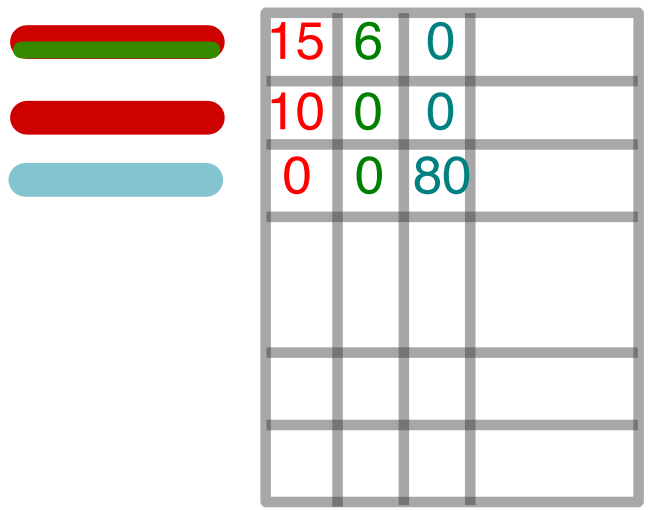


Associate counts to kmers

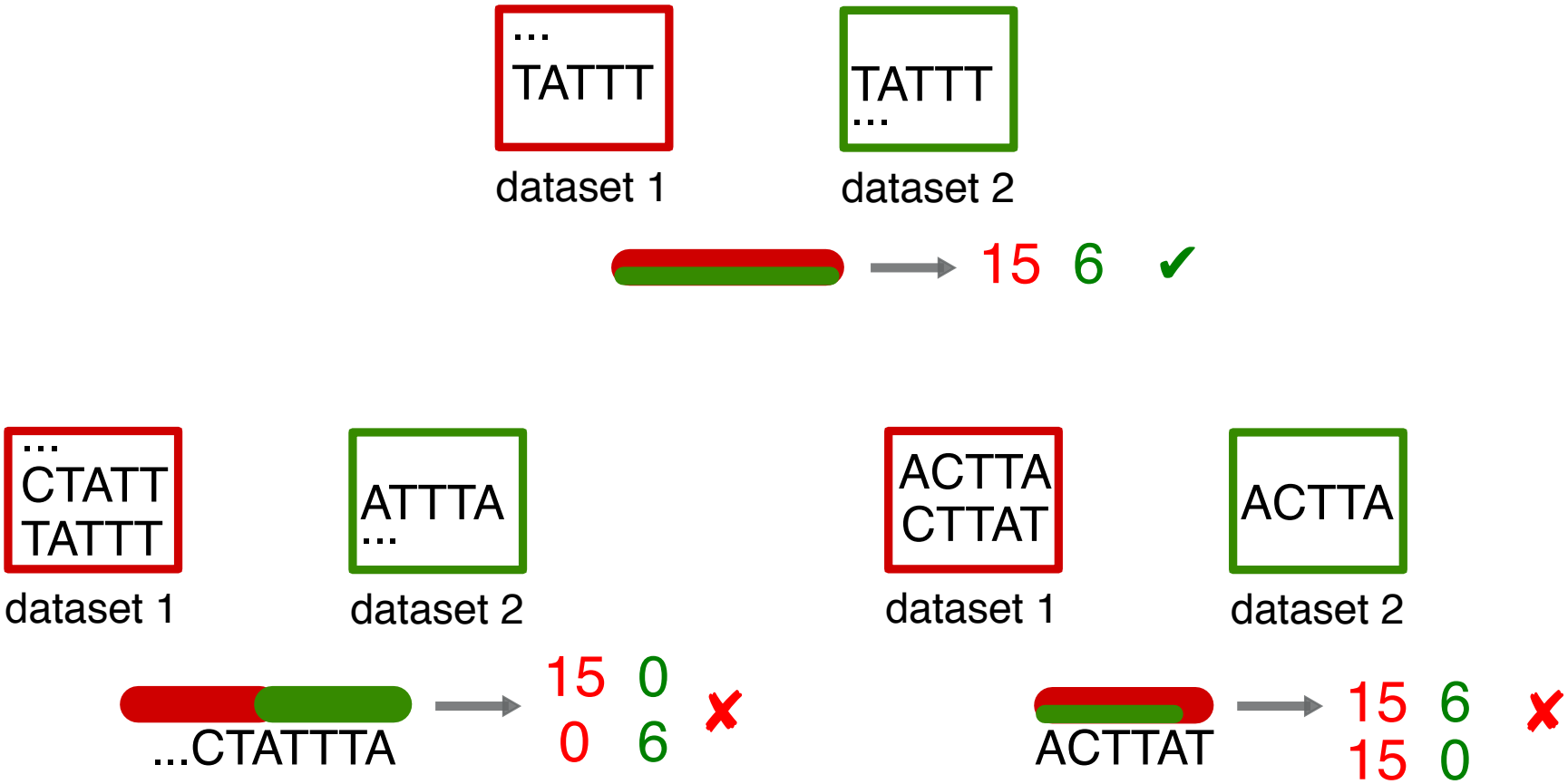
union graph: k-mer set



1 count vector per unitig



Associate counts to kmers



Represent a set of k-mers: Spectrum Preserving String Sets

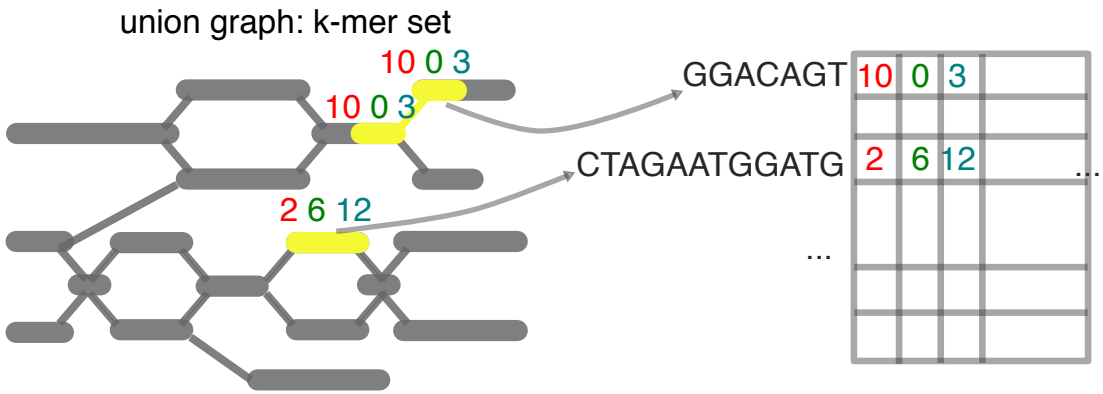
A SPSS of a k-mer set S is a **set of strings having same k-mer spectrum as S**

- k-mer set itself
- Unitigs
- Super k-mers from reads [Deorowicz et al.'15]
- Super k-mers from unitigs [Marchet et al.'19]
- Simplitigs[Brinda et al.'20]/UST[Rahman et al.'20]

None can guarantees that all k-mers in a given string have the same count-vector

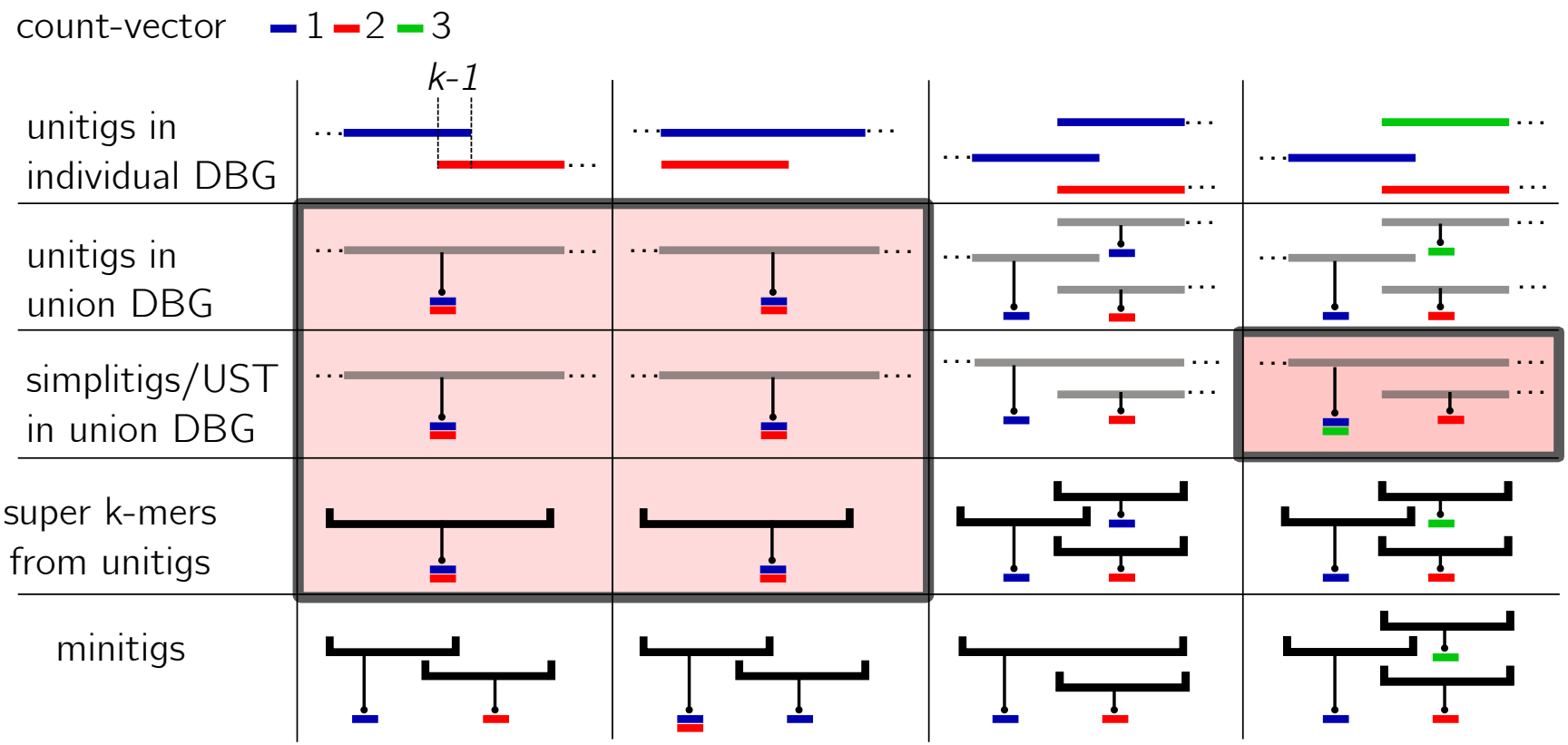
A new SPSS: Minitigs

Minitigs are paths of the union DBG:

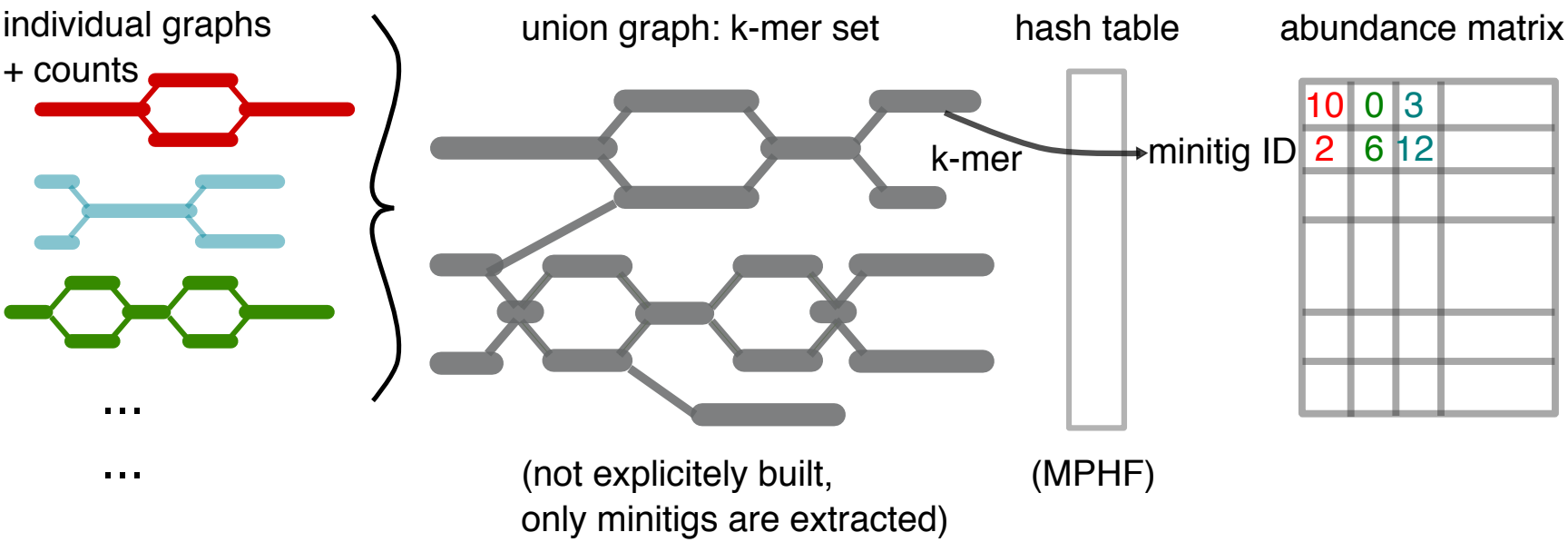


- All k-mers in a minitig have the same count vector
- Each k-mers is in one and only one minitig
- Minitigs can span several unitigs
- **In practice**
  - All k-mers in a minitig have the same minimizer
  - Greedy algorithm for construction

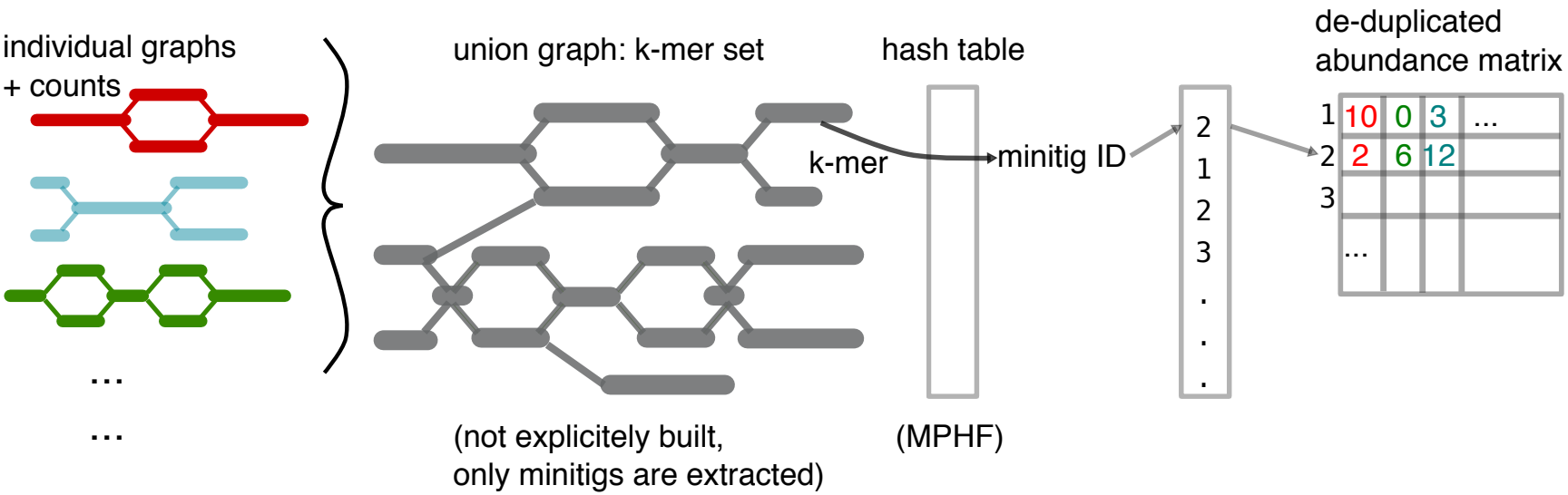
Minitig example



REINDEER

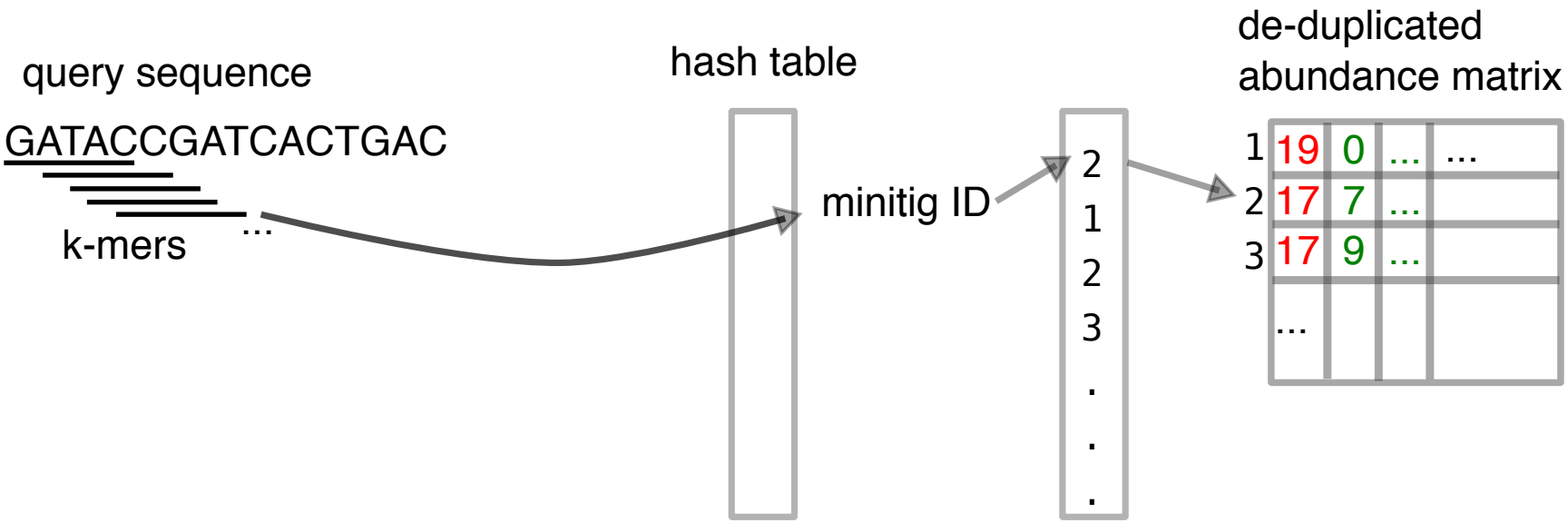

individual graphs
+ counts
...
...
union graph: k-mer set
(not explicitely built,
only minitigs are extracted)
hash table
k-mer
minitig ID
(MPHF)
abundance matrix
10 0 3
2 6 12

REINDEER



- Each count-vector is compressed with RLE and dumped on the disk
- The MPHF can be dumped as well

Query




```
>829 LN:i:47 KC:i:331 km:f:19.5   L:-:134:- L:-:13      19      *
>830 LN:i:42 KC:i:213 km:f:17.8   L:+:5:+ L:+:482:      17      7:9
```

- Value reported only if X% of the query k-mers were found present in a dataset

Results: index construction

$\sim$ 2500 human RNA-seq datasets

$\sim$ 4 billions distinct k-mers

Tool	Ext. Memory (GB)	Time (h)	Peak RAM (GB)	Index Size (GB)	Counts (Y/N)
SBT	300	55	25	200	N
HowDeSBT	30	10	N/A	15	N
Mantis	3,500	20	N/A	30	N
SeqOthello	190	2	15	20	N
BIGSI	N/A	N/A	N/A	145	N
Reindeer - raw counts	6,800	55	36	60	Y
Reindeer - discretized	6,500	58	35	42	Y
Reindeer - log 2	5,500	68	28	40	Y
Reindeer - presence/absence	6,600	55	27	36	N

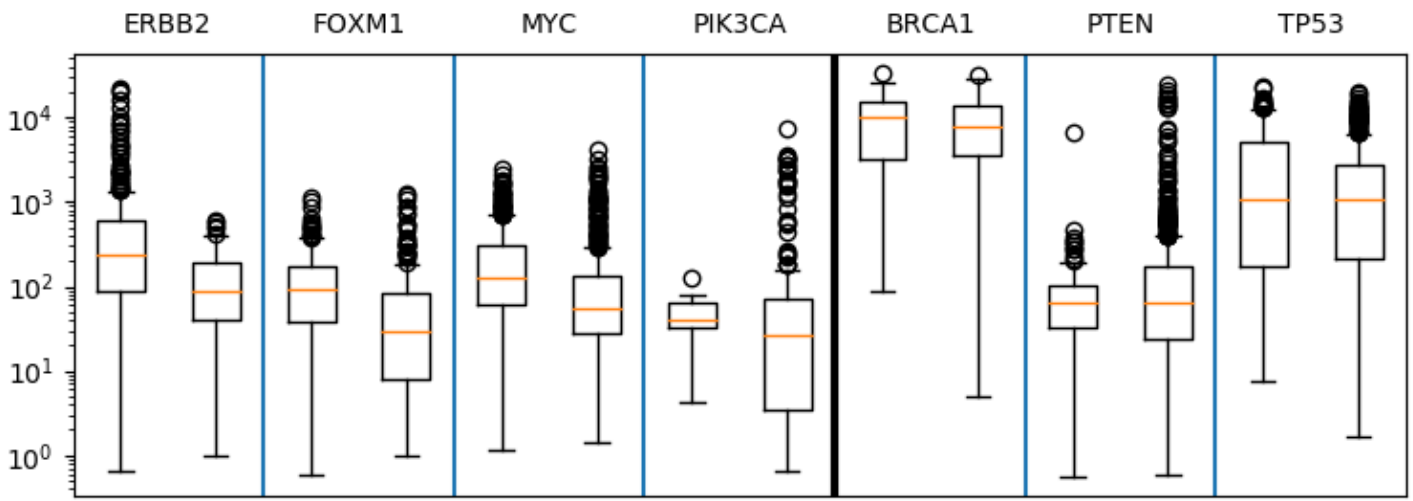
Results: query

Batches of sequences using Refseq human transcripts (mean size 3,300 bases)

Batch size	**Index loading time (s, wallclock)** **mean**/min/max	**Query time (s, wallclock)** **mean**/min/max	**Peak RAM (GB)**
10 sequences	**475.7**/459.8/506.5	**41.68**/40.55/42.97	75
100 sequences		**41.95**/40.35/45.98	
1000 sequences		**42.60**/41.62/46.20	
1000 sequences		**42.70**/40.47/46.28	

Application to transcriptomics

Find abundances of oncogenes/tumor repressor genes in a few minutes across 2585 datasets



Left boxplot:Cancer / Right boxplot: Non-cancer

- Need normalization to go further with biological conclusions

Take home messages

What REINDEER does:
query abundances of sequences in a collection of datasets of raw reads

- Represent the set of k-mers using minitigs
- Exact associative index for k-mer $\rightarrow$ count information
- Counts per dataset in compressed, non redundant abundance matrix
- Reindeer can do presence/absence but other data-structures perform better for this (HowDeSBT, BIGSI,...)