Set-Min sketch

A probabilistic map for power-law distributions with applications to k-mer annotation

Yoshihiro Shibuya[1] and Gregory Kucherov[1,2]

[1] Université Gustave Eiffel
[2] CNRS

k-mer counting

Computing the frequency of k-mers is a rather common task in bioinformatics.

Two great families of counting software:

- Fast in-memory counters like Jellyfish [Marçais and Kingsford, 2011] or Counting Quotient Filters [Pandey et al., 2018].
- Disk-based counters such as KMC [Marçais and Kingsford, 2011] and DSK [Rizk et al., 2013]

.

Note: neither of the two type tries to aggressively reduce the size of the final representation.

Full counting tables

AAA...A	20
AAA...C	5
AAA...G	8
AAA...T	11
...	...
TTT...T	3

The aforementioned tools output sequence **AND** counting information.

In many applications this is not necessary.

Think, for example, to an alignment algorithm which wants to filter out repetitive portions of reads. In this case k-mers are retrieved from the reads with a sliding window.

Removing genomic information - MPHFs

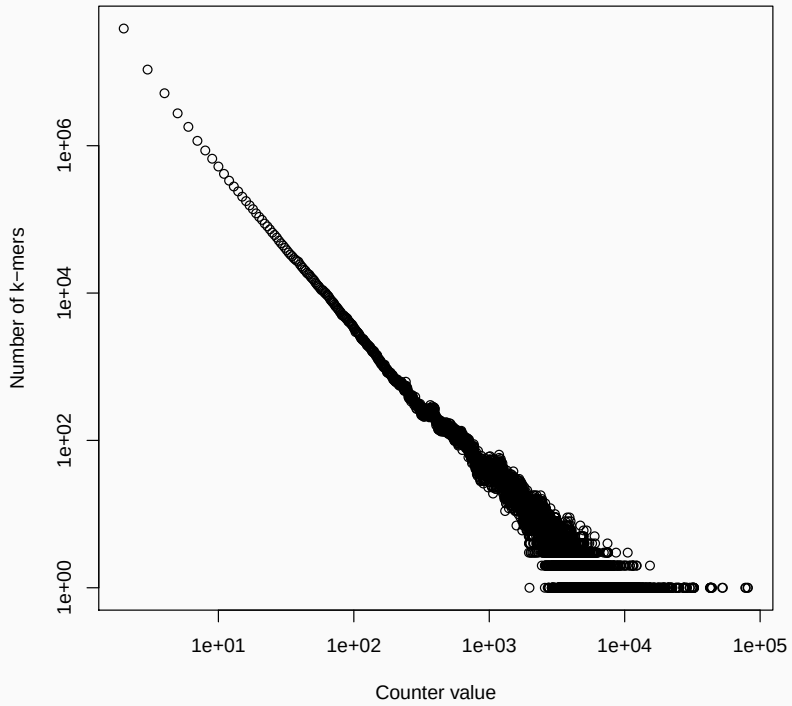
Minimal Perfect Hash Functions (MPHFs) can be used to map k-mers to their frequencies without the need to store them explicitly.

A MPHF is a biijective map between a set of keys S of size $|S|$ to the interval $[0, |S| - 1]$.

The actual frequencies can be stored in an array indexed by the MPHF.

Note that the external array can store any type of additional information associated to the k-mers [Limasset et al., 2017, Marchet et al., 2020].

k-mer spectrum



Usually, for large ks the k-mer frequency spectrum follows a Power-Law distribution.

Most k-mers have very low frequencies. Only a handful are very repetitive.

Observation: many algorithms only require a qualitative measure of frequency. We can save space by working with **approximate** counters thanks to Count-Min sketch [Cormode and Muthukrishnan, 2005].

Count-Min sketch

- $R \times B$ matrix of counters.
- Each row is associated to a different hash function h_i.
- All functions are pair-wise independent and used to retrieve the buckets for a key (one in each row).
- Updates are performed by summing the counter of a key to all its buckets.
- Queries return the minimum bucket among those associated with the key.

Count-Min sketch - Update

We start with an empty sketch:

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Count-Min sketch - Update

We add a k-mer of frequency $= 5$.

0	5	0	0	0	0
0	0	0	5	0	0
5	0	0	0	0	0
0	0	0	0	5	0

Count-Min sketch - Update

And a k-mer of frequency $= 7$.

0	5	0	7	0	0
0	0	0	12	0	0
5	0	0	0	7	0
0	0	0	0	5	7

Collisions (red cells) might happen.

Count-Min sketch - Query

We query the sketch for the first k-mer we inserted (frequency = 5).

0	5	0	7	0	0
0	0	0	12	0	0
5	0	0	0	7	0
0	0	0	0	5	7

$\hat{f} = min_j(S[j, h_j(i)])$

In the specific case: $min_j(5, 12, 5, 5) = 5$

Count-Min

Given $0 < \varepsilon \leq 1$ and $0 < \delta \leq 1$ we can choose $R = \lceil \ln(\frac{1}{\delta}) \rceil$ and $B = \lceil \frac{e}{\varepsilon} \rceil$ to achieve a point-query error of $\varepsilon \|\boldsymbol{a}\|_1$ with probability at least $1 - \delta$, where $\|\boldsymbol{a}\|_1$ is the L_1-norm of $\boldsymbol{a}$.

The error in this case can only be an over-estimation of the true frequency.

For very big datasets the $\varepsilon \|\boldsymbol{a}\|_1$ could be too large to give meaningful bounds.

Set-Min sketch

Same structure as a Count-Min sketch but...

- Each cell is a set of labels, not necessarily numeric
- Insertions are performed by union
- Queries are set intersections

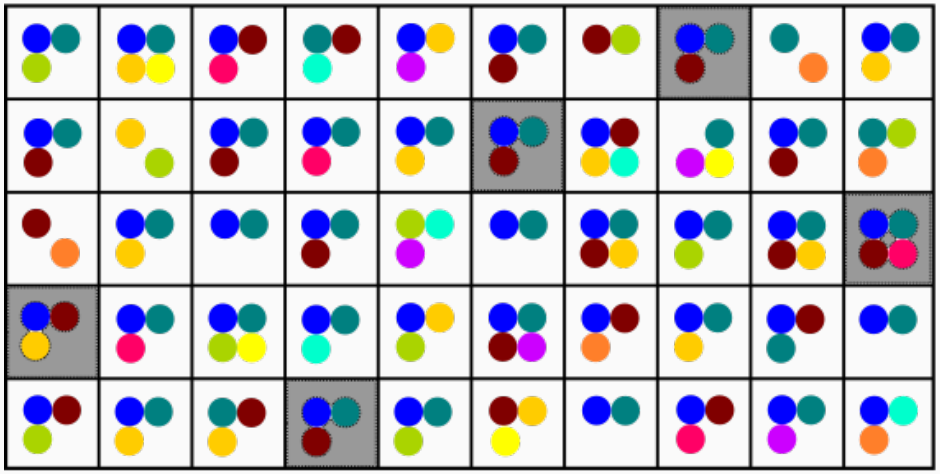
We lose the possibility to work in the streaming setting.

Set-Min sketch

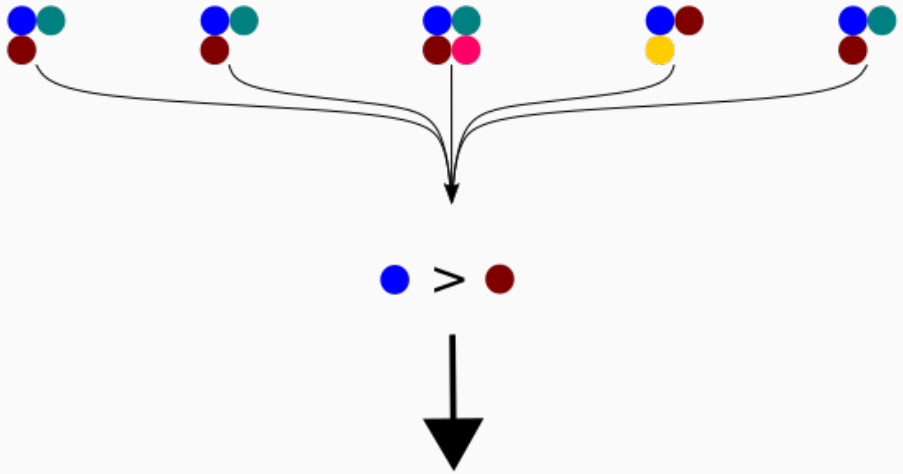
Each query can have one of the following outcomes:

- Empty intersection: the k-mer was not in the dataset
- Singleton frequency: right frequency of the k-mer
- Multiple frequencies: collision

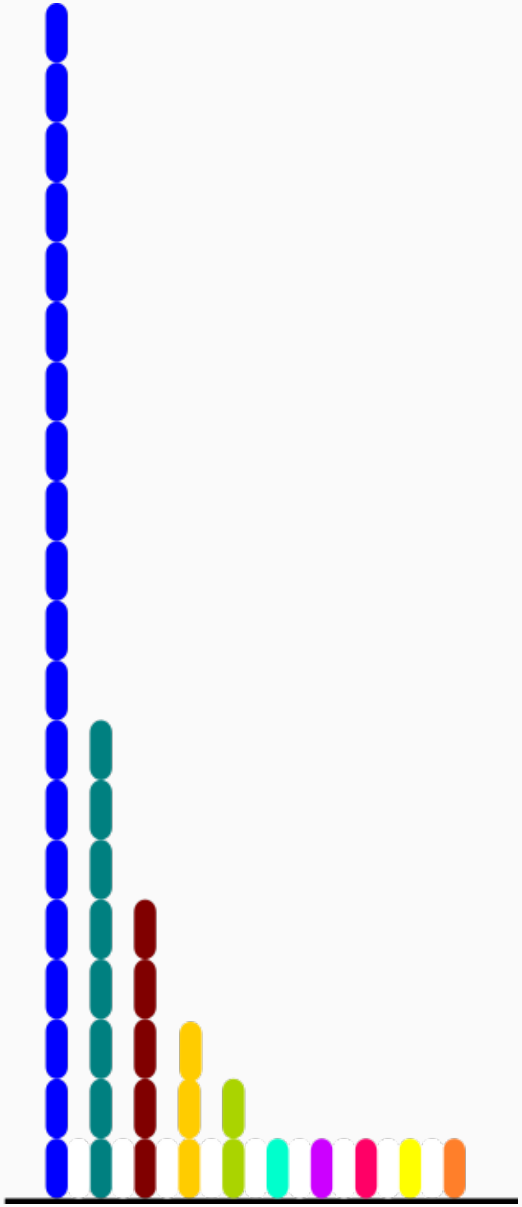
In case of a collision, the frequency with the minimal number of k-mers is returned. The rationale for this is that it is the least likely counter to appear by chance, which means it is probably the most likely to be correct.



Intersect:



Final answer:



Dimensioning

The expected **sum of errors** given our scheme is:

$$\sum_{\ell \in L} c_\ell \sum_{\substack{c_m < c_\ell \\ m \in L}} |m - \ell| \left(1 - e^{-\frac{c_m}{B}}\right)^R$$

Dimensioning

The expected **sum of errors** given our scheme is:

$$\sum_{\ell \in L} c_\ell \sum_{\substack{c_m < c_\ell \\ m \in L}} |m - \ell| \left(1 - e^{-\frac{c_m}{B}}\right)^R$$

Where:

- is the probability of returning frequency m instead of the right frequency ℓ. c_m is the number of k-mers having frequency m.

Dimensioning

The expected **sum of errors** given our scheme is:

$$\sum_{\ell \in L} c_\ell \sum_{\substack{c_m < c_\ell \\ m \in L}} |m - \ell| \left(1 - e^{-\frac{c_m}{B}}\right)^R$$

Where:

- is the probability of returning frequency m instead of the right frequency ℓ. c_m is the number of k-mers having frequency m.
- is the error when considering frequency m instead of ℓ

Dimensioning

The expected **sum of errors** given our scheme is:

$$\sum_{\ell \in L} c_\ell \sum_{\substack{c_m < c_\ell \\ m \in L}} |m - \ell| \left(1 - e^{-\frac{c_m}{B}}\right)^R$$

Where:

- is the probability of returning frequency m instead of the right frequency ℓ. c_m is the number of k-mers having frequency m.
- is the error when considering frequency m instead of ℓ
- is the number of k-mers having frequency ℓ

Dimensioning

The expected **sum of errors** given our scheme is:

$$\sum_{\ell \in L} c_\ell \sum_{\substack{c_m < c_\ell \\ m \in L}} |m - \ell| \left(1 - e^{-\frac{c_m}{B}}\right)^R < \varepsilon \sum_{\ell \in L} \ell c_\ell$$

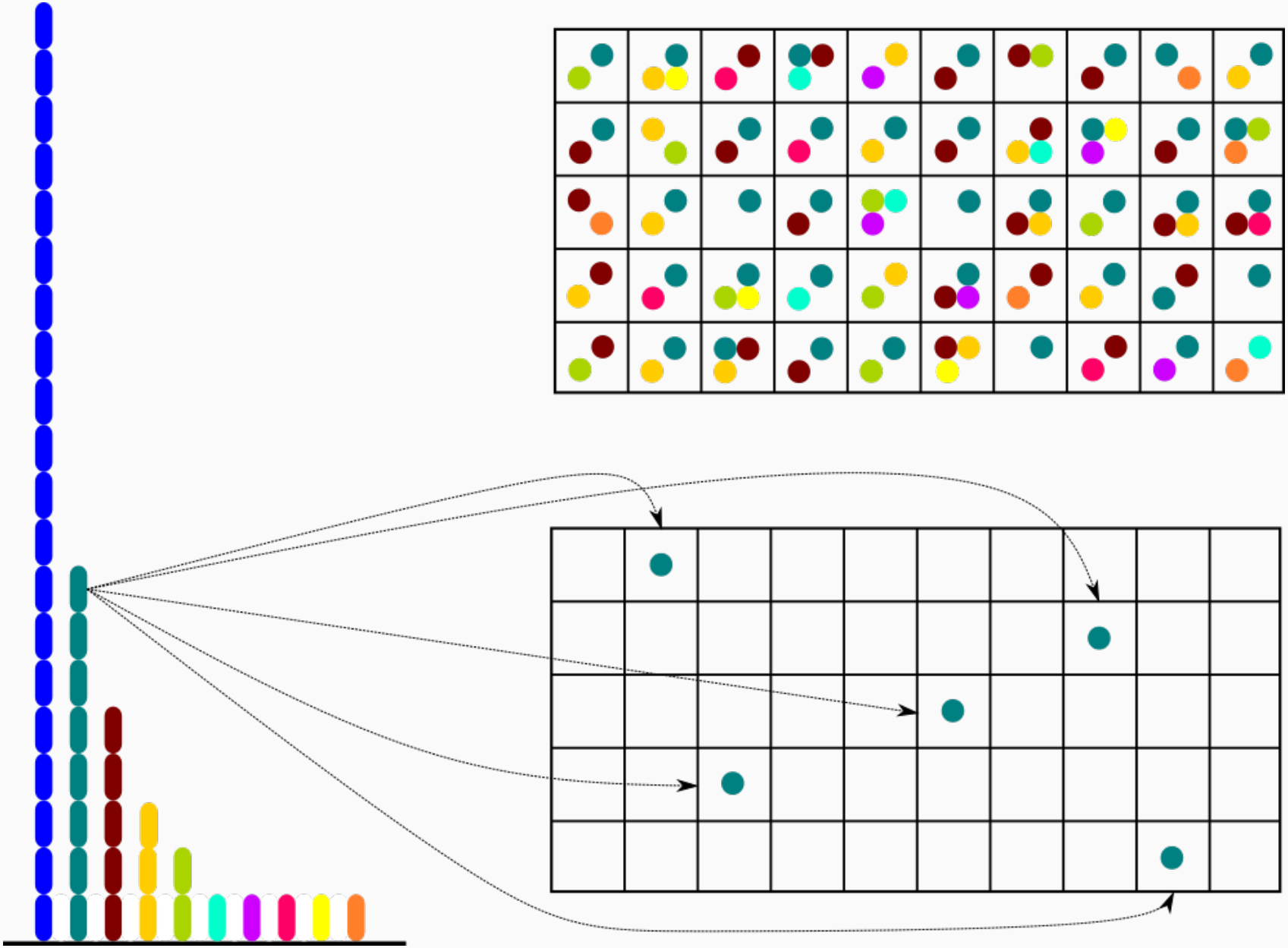
Where:

- is the probability of returning frequency m instead of the right frequency ℓ. c_m is the number of k-mers having frequency m.
- is the error when considering frequency m instead of ℓ
- is the number of k-mers having frequency ℓ
- is the $\varepsilon \|\boldsymbol{a}\|_1$ threshold.

Reducing space

If we are not interested in answering presence/absence queries we can omit the frequency with the largest number of k-mers during insertion.

With this modification, in case of an empty intersection, the returned element will be the ignored frequency.



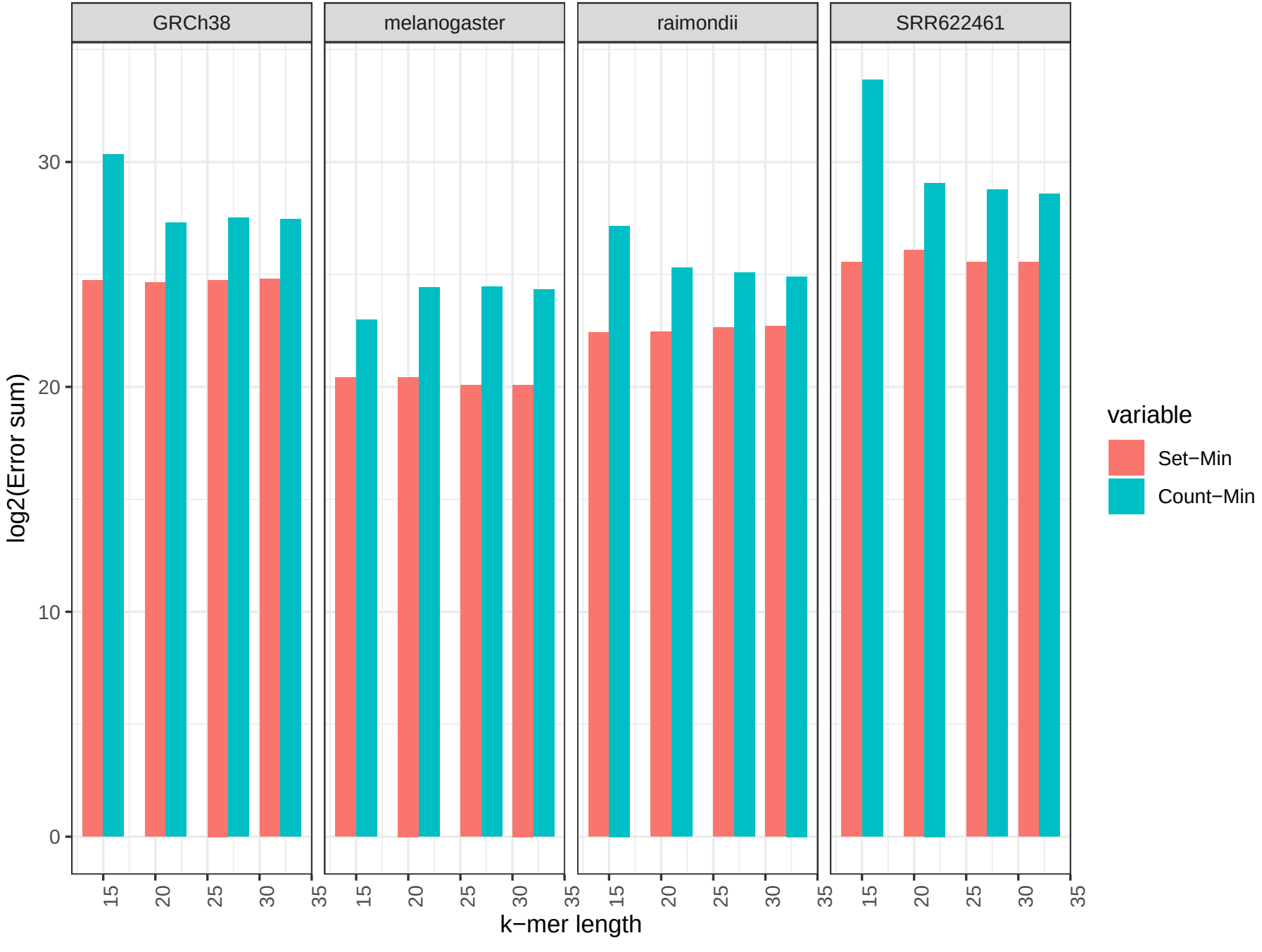
Set-Min vs. Count-Min

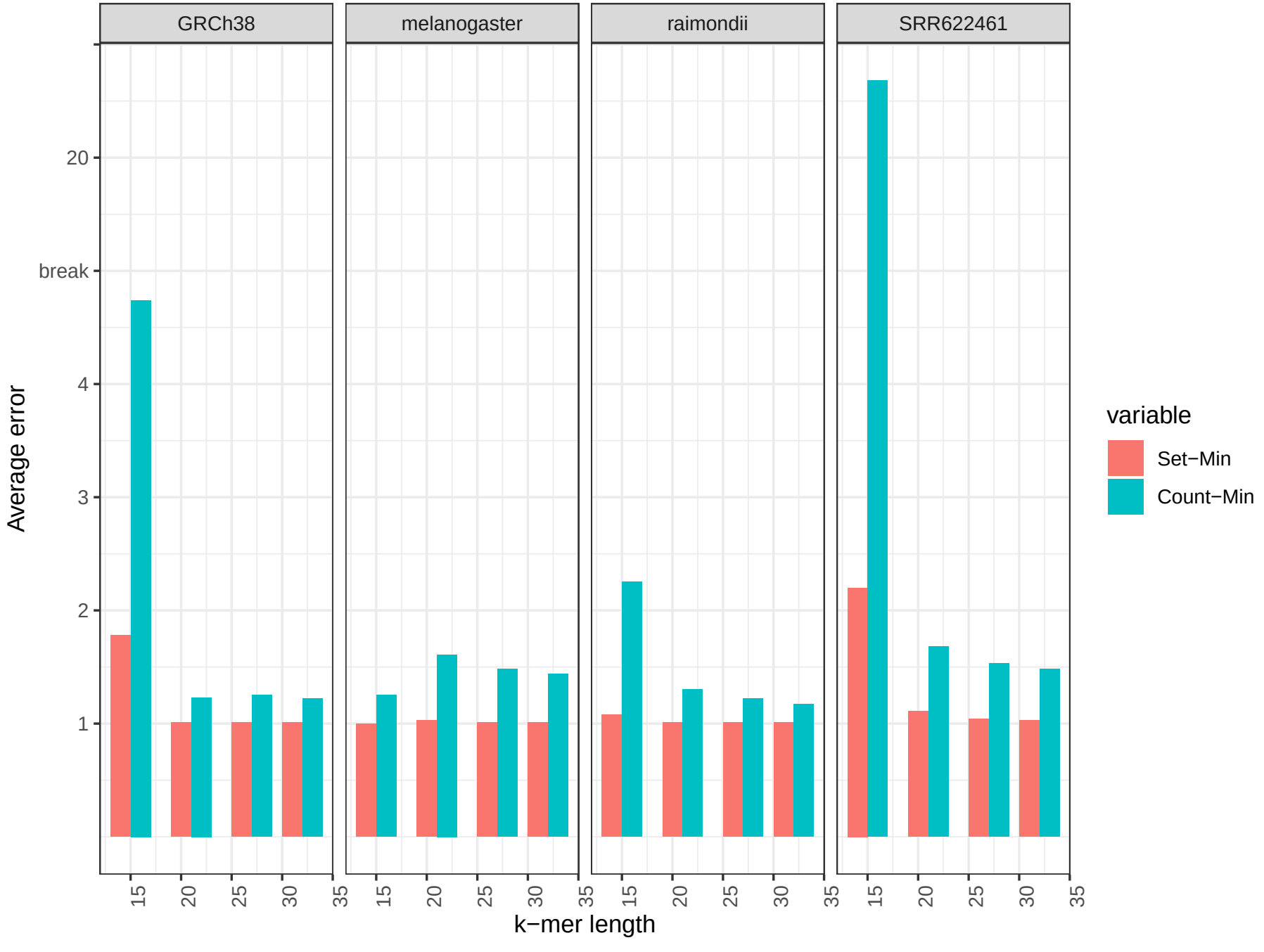
4 datasets:

- 3 fully assembled genomes of different sizes (*drosophila melanogaster*, *gossypium raimondii*, human).
- 1 human unassembled dataset of Illumina short reads (NA12878).

Set-Min with $\varepsilon = 0.01$, Count-Min use the same dimensions.

Both sketches ignore the heaviest element of the spectrum.


GRCh38
melanogaster
raimondii
SRR622461
log2(Error sum)
30
20
10
0
15
20
25
30
35
k−mer length
variable
Set−Min
Count−Min


GRCh38
melanogaster
raimondii
SRR622461
Average error
20
break
4
3
2
1
15
20
25
30
35
k−mer length
variable
Set−Min
Count−Min

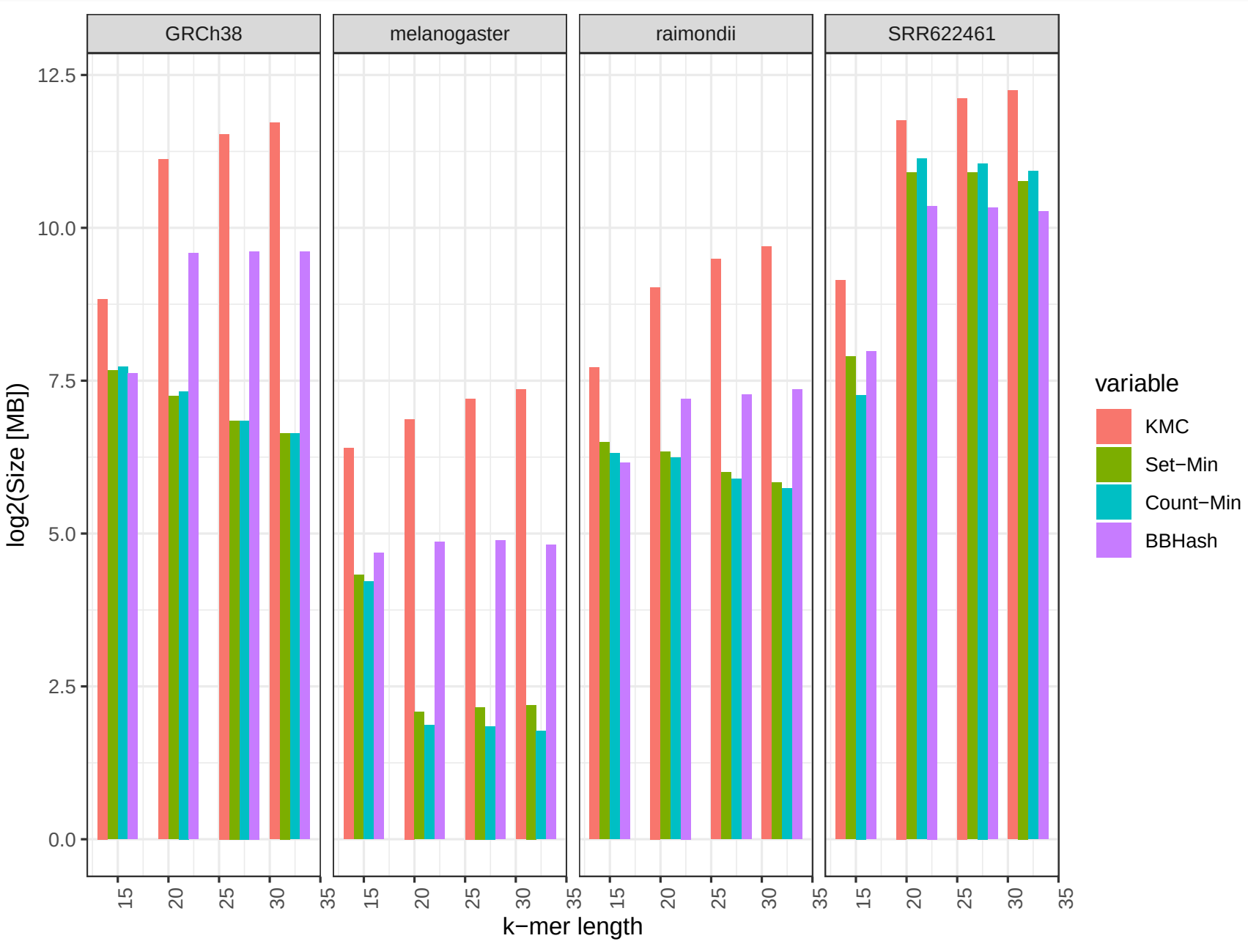
Set-Min vs. Count-Min

The total sum of errors in case of Set-Min was always below the theoretical threshold.

Set-Min sketch gives more accurate estimations than a Count-Min sketch under the same conditions.

In case of Set-Min, most errors occur between the most abundant labels, which, in our particular case, are the smallest frequencies.

Memory comparison



Memory comparison

For assembled genomes and large ks, Set-Min sketch can achieve smaller space than MPHFs.

Exact and fully static methods, such as MPHFs, might be preferable for small ks.

For unassembled genomes, MPHFs seem to have an edge over Set-Min sketches, but bigger ε might not introduce deleterious errors for down-stream analysis. We can tolerate mixing low-frequencies much better than higher ones.

Conclusions

Set-Min sketch is a Count-Min sketch working with sets instead of counters. Unlike its parent method it does not work in a streaming setting.

It bounds the (expected) **sum** of errors.

Most of the errors only occur between the heaviest elements of the spectrum (low frequencies).

For very skewed distributions Set-Min sketch can be smaller than MPHFs.

References

Graham Cormode and S. Muthukrishnan. Summarizing and mining skewed data streams. pages 44–55, 2005. 5th SIAM International Conference on Data Mining, SDM 2005 ; Conference date: 21-04-2005 Through 23-04-2005.

Antoine Limasset, Guillaume Rizk, Rayan Chikhi, and Pierre Peterlongo. Fast and scalable minimal perfect hashing for massive key sets. *arXiv:1702.03154 [cs]*, February 2017. URL `http://arxiv.org/abs/1702.03154`. arXiv: 1702.03154.

References ii

Camille Marchet, Lolita Lecompte, Antoine Limasset, Lucie Bittner, and Pierre Peterlongo. A resource-frugal probabilistic dictionary and applications in bioinformatics. *Discrete Applied Mathematics*, 274: 92–102, March 2020. ISSN 0166-218X. doi: 10.1016/j.dam.2018.03.035. URL http://www.sciencedirect.com/science/article/pii/S0166218X18301288.

Guillaume Marçais and Carl Kingsford. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27 (6):764, March 2011. doi: 10.1093/bioinformatics/btr011. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3051319/.

References iii

Prashant Pandey, Fatemeh Almodaresi, Michael A. Bender, Michael Ferdman, Rob Johnson, and Rob Patro. Mantis: A Fast, Small, and Exact Large-Scale Sequence-Search Index. *Cell Systems*, 7(2): 201–207.e4, August 2018. ISSN 2405-4712. doi: 10.1016/j.cels.2018.05.021. URL https://www.cell.com/cell-systems/abstract/S2405-4712(18)30239-4.

Guillaume Rizk, Dominique Lavenier, and Rayan Chikhi. DSK: k-mer counting with very low memory usage, March 2013. URL https://pubmed.ncbi.nlm.nih.gov/23325618/.