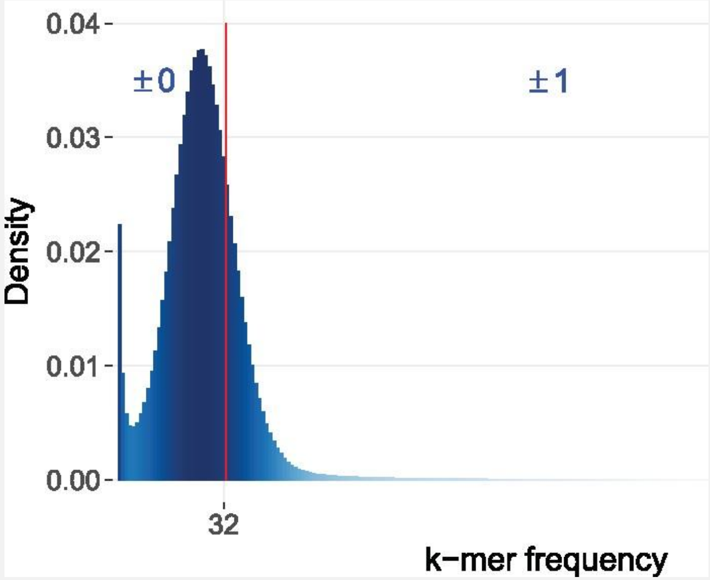
CS4330: Combinatorial Methods in Bioinformatics

K-mers count packing

Wong Limsoon

Too many K-mers to keep in memory for convenient access

Data	$\|k\text{-mer}_1\|$(m)	$\|k\text{-mer}_{2-1000}\|$(m)
D1	35.67	3.49
D2	54.13	5.91
D3	372.09	99.92
D4	4643.11	543.89
D5	4171.45	2748.5



Distribution of *31*-mers in dataset D3 (human chr 14) having value larger than 2.

Jiang et al., "kmcEx: memory frugal and retrieval efficient encoding of counted k-mers", *Bioinformatics* 35(23):4871-4878, 2019

Keep in one big Bloom filter?

n = size of Bloom filter
m = # of elements inserted
ε = false positive rate

Optimal size of Bloom filter is $n = -2.08\ m\ (\ln \varepsilon)$ bits

For dataset D5,

# of K-mers ≈ 7 billions

$n = -2.08\ (7 \times 10^9)\ (\ln \varepsilon)$

≈ 100×10^9 bits ≈ 12 GB at $\varepsilon = 0.01\%$

But this Bloom filter cannot tell you frequency of K-mers ☹

Separation trick

Use separate Bloom filters to store K-mers of different frequency; i.e., use H_j to store K-mers of frequency j

K-mer frequencies can go from 1, 2, …, to thousands

Use $H_1, \ldots, H_h$ to store K-mers of frequencies 1 to h

And look for clever idea to deal with K-mers having frequency > h

Exercise

Cf. D5, suppose

4×10^9 K-mers with freq = 1

90×10^6 K-mers with freq = 2

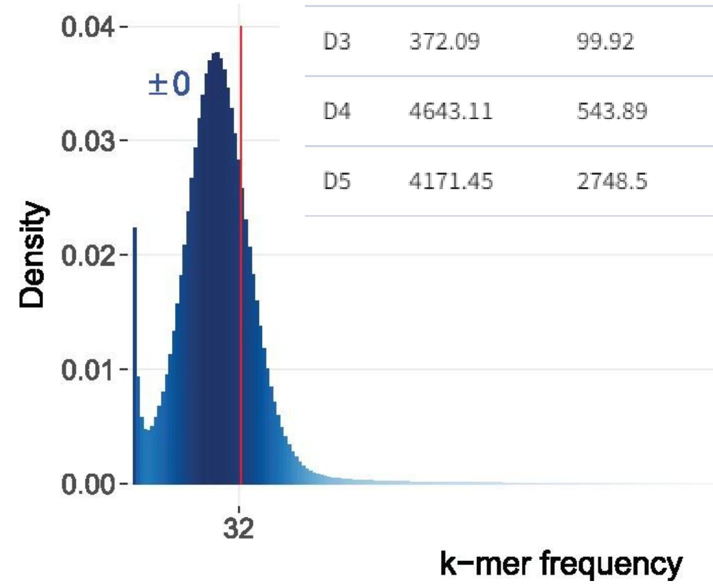
15×10^6 K-mers with freq = 3

18×10^6 K-mers with freq = 4

21×10^6 K-mers with freq = 5

3×10^9 K-mers with freq > 5

Data	$\lvert k\text{-mer}_1 \rvert$(m)	$\lvert k\text{-mer}_{2-1000} \rvert$(m)
D1	35.67	3.49
D2	54.13	5.91
D3	372.09	99.92
D4	4643.11	543.89
D5	4171.45	2748.5



Distribution of *31*-mers in dataset D3 (human chr 14) having value larger than 2.

What space is needed to store them in H_1, …, H_5 and a hash table (for the counts of K-mers with freq > 5) ?

The coupled bit arrays of kmcEx

kmcEx stores K-mers and their counts using a pair of Bloom filter-like bit arrays $B = (B^+, B^-)$

Encoding

Let $\mathbb{K}$ be a set of m K-mers

and $\mathbb{F} = \{ f_\kappa \mid \kappa \in \mathbb{K} \}$ be their counts

Let $H_0, H_1, \ldots, H_{h-1}$ be hash functions

B^+, B^- new bit arrays with n bits,

$n = -2.08\ m\ (\ln \varepsilon)$

For each $\kappa \in K$, $i \in \{ 0, 1, \ldots, h-1 \}$,

$B^+[H_i(\kappa)] = 1$

$B^-[H_i(\kappa)] = \text{Binary}(f_\kappa)^h[i]$

where $\text{Binary}(f_\kappa)^h$ is the binary representation of f_κ by h bits, and $\text{Binary}(f_\kappa)^h[i]$ returns the value of i-th bit. For instance, $\text{Binary}(50)^7 = 0110010$, and $\text{Binary}(50)^7[2] = 1$.

Decoding

To "decode" κ, i..e obtain its count

If $\Pi_{i \in \{0, 1, \ldots, h-1\}} B+[H_i(\kappa)] \neq 1$,

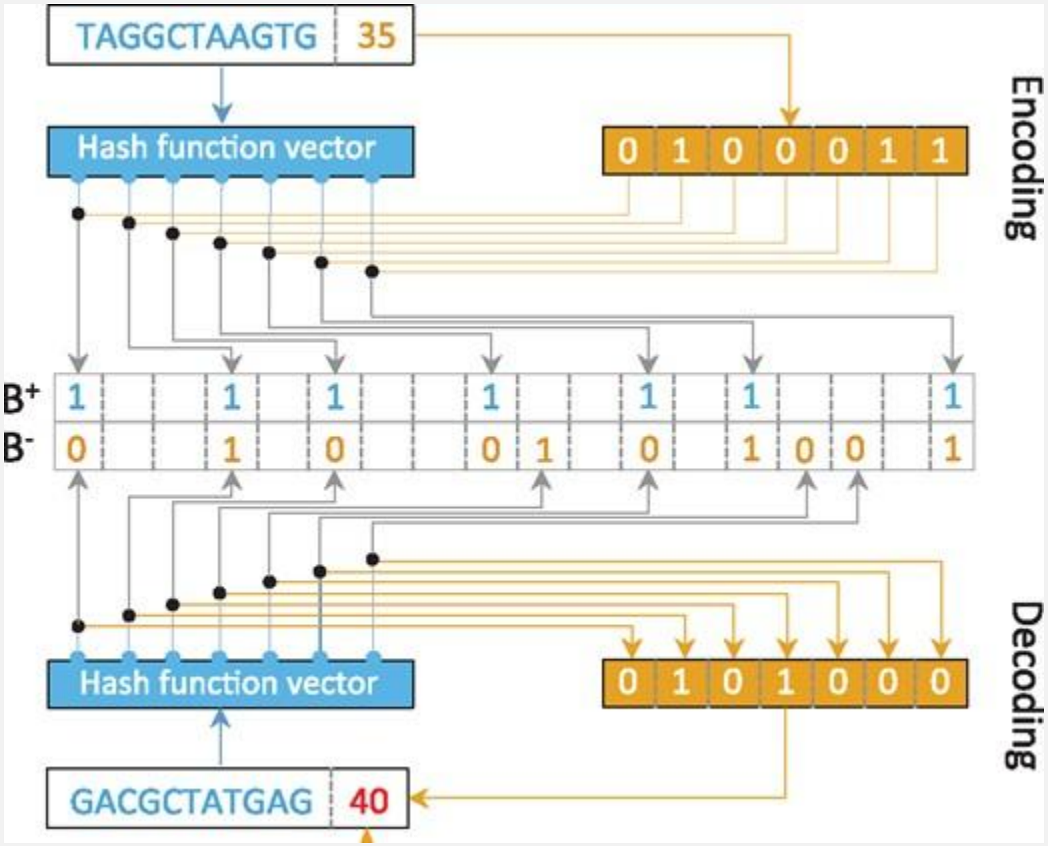
Then κ does not exist

Else $f_\kappa = \text{Denary}(B^-[H_0(\kappa)] \ldots B^-[H_{h-1}(\kappa)])$

where $\text{Denary}(\cdot)$ transforms the binary represented number into the decimal mode. For instance, $\text{Denary}(0100011) = 35$.

Jiang et al., "kmcEx: Memory-frugal and retrieval-efficient encoding of counted k-mers", *Bioinformatics* 35(23):4871-4878, 2019

Example



Jiang et al., "kmcEx: Memory-frugal and retrieval-efficient encoding of counted k-mers", *Bioinformatics* 35(23):4871-4878, 2019

Collisions

Traditional Bloom filters no need to care for collisions

But kmcEx must take care of collisions in B^- because the bits can change from 0 to 1 and 1 to 0

Collison happens in B^-

If there $i \in \{0, 1, \ldots, h-1\}$ *such that*

$B^+[H_i(\kappa)] = 1$ *and* $B^-[H_i(\kappa)] \neq B'^-[H_i(\kappa)]$

where κ is the K-mer to be inserted and B' is the updated coupled bit-arrays if κ is inserted

Exercise

Suggest a simple and effective way for kmcEx to deal with collisions

Other ideas in kmcEx

False positive reduction

Check if any of κ's neighbours is found and has similar count as κ

Frequency binning

Discretize counts into bins of progressively larger width

e.g., use 60 to represent frequencies 59, 60, & 61

K-mer separation

Use separate vanilla Bloom filter for K-mers of freq = 1

Memory usage, count fidelity, & FPR

Dataset	Genome size	Read length	Coverage	No. paired-end reads	Input size (fastq)
D1	2.8M	101	46.3×	1 294 104	280M
D2	4.6M	101	33.6×	766 646	446M
D3	88.3M	101	38.3×	16 757 120	9.4G
D4	249.2M	124	150.8×	303 118 594	92G
D5	3121.8M	101	27.6×	854 084 773	442G

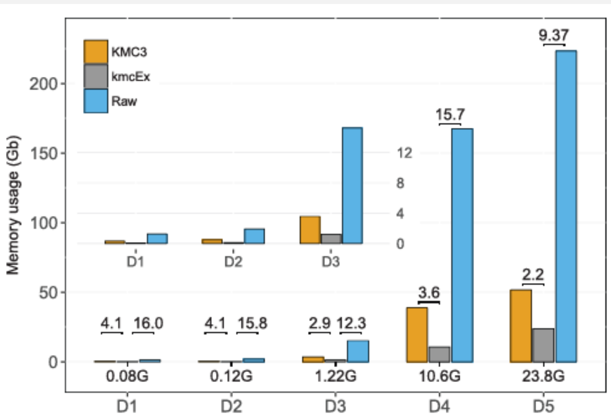


Fig. 2. Memory usage comparison between kmcEx, KMC3 and the raw input. The number over each pair of bars shows the ratio of memory usage between the two approaches, while the number under a gray bar is the real memory usage of kmcEx. For the sake of clarity, the results of D1, D2 and D3 are enlarged in the inset figure. Results shown here are obtained at $k = 31$, $h = 7$ and frequency≥1

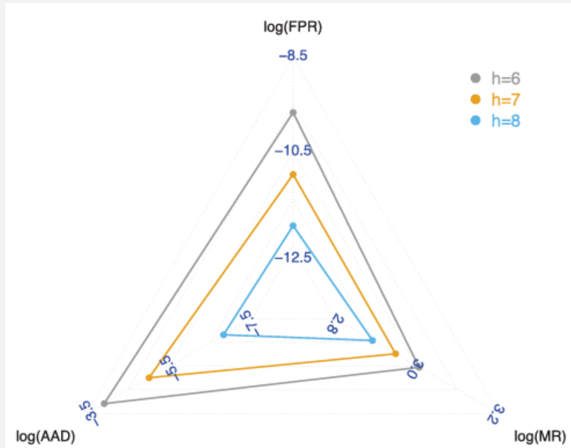


Fig. 3. The effect of number of hash functions h to false-positive rate (FPR), averaged absolute distance (AAD) and memory-saving ratio (MR). The data shown here is the mean value obtained from the five real datasets for each metrics having $k = 31$ and frequency≥1. For ease of reading, the axes are shown in log scale having base of e

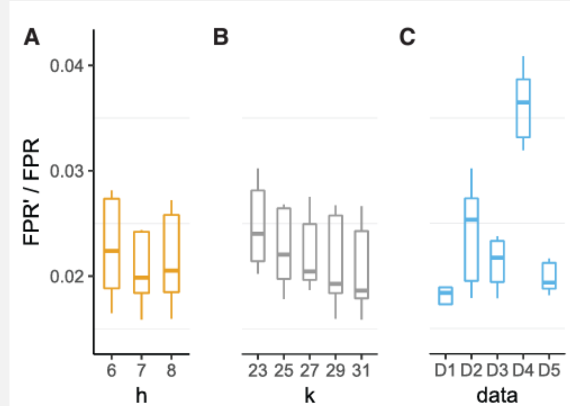


Fig. 5. A comprehensive comparison of FPR reduction via joint examination on k-mers having frequency≥2. Panel (**A**) shows the FPR reduction w.r.t. the number of hash functions (h). Panel (**B**) is the relation between the reduction and the k-mer size (k) and Panel (**C**) reveals the reduction on different datasets. The 'FPR' is the original false-positive rate, while the 'FPR'' is the reduced FPR obtained when the neighbors of a k-mer and the (k-2)-mer are jointly considered

Jiang et al., "kmcEx: Memory-frugal and retrieval-efficient encoding of counted k-mers", *Bioinformatics* 35(23):4871-4878, 2019

Running time

Encoding = ~1.3 mps (mil K-mers per sec)

Decoding = ~0.5 mps (present K-mers),

~0.7 mps (absent K-mers)

Dataset	Genome size	Read length	Coverage	No. paired-end reads	Input size (fastq)
D1	2.8M	101	46.3×	1 294 104	280M
D2	4.6M	101	33.6×	766 646	446M
D3	88.3M	101	38.3×	16 757 120	9.4G
D4	249.2M	124	150.8×	303 118 594	92G
D5	3121.8M	101	27.6×	854 084 773	442G

Table 4. Running time of encoding and decoding on the five datasets having $k=31$, $h=7$ and frequency≥ 1

Data	Encoding		Decoding								
	|*k*-mers| (m)	Time (s)	|*k*-mers| (k)	Present				Absent			
				$\text{Time}_{\text{open}}$ (s)	$\text{Time}_{\text{query}}$ (s)	FPR	FNR	$\text{Time}_{\text{open}}$ (s)	$\text{Time}_{\text{query}}$ (s)	FPR	FNR
D1	39.2	14.6	500	0.269	1.005	0	0	0.241	0.671	4.8e-4	0
D2	60.0	24.2	500	0.304	0.846	0	0	0.224	0.589	5.2e-4	0
D3	472.0	238.2	500	3.049	0.974	0	0	3.107	0.738	8.6e-4	0
D4	5187.0	2152.9	500	19.28	1.197	0	0	20.78	0.701	1.1e-3	0
D5	6919.1	3969.3	500	86.35	1.222	0	0	88.15	1.062	1.5e-3	0

Note: FPR, false-positive rate; FNR, false-negative rate; encoding and decoding are run by four threads. Note that the opening time of query is the whole time of loading all the encoded *k*-mers of a dataset.

Expt performed on a computer w/ 256GB RAM, 2 x E5-2683V4 CPU, CentOS 7.0

Jiang et al., "kmcEx: Memory-frugal and retrieval-efficient encoding of counted k-mers", *Bioinformatics* 35(23):4871-4878, 2019

Good to read

P. Jiang et al., “kmcEx: Memory-frugal and retrieval-efficient encoding of counted k-mers”, Bioinformatics 35(23):4871-4878, 2019. https://pubmed.ncbi.nlm.nih.gov/31038666/