Sublinear Random Sampling and Applications

Conrado Martínez
Universitat Politècnica de Catalunya

Joint works with: M. R. Hera (Penn State), D. Koslicki (Penn State), J. Lumbroso (Princeton), H. Sanz-González (UPC), A. Viola (UdelaR) & J. Wang (UPC)

January 2026

Outline of the Talk

Outline of the Talk

Introduction

Enter GenBank:

1. $5.7 \cdot 10^9$ DNA sequences
2. $44 \cdot 10^{12}$ nucleotides (8.4 Terabytes)
3. Average length: 7700 nucleotides/sequence
4. $2.6 \cdot 10^8$ "traditional", curated sequences; the vast majority of sequences are WGS, TSL & TSA records: $\approx$ fragments from automatic genome sequencing with no annotation or automatic annotations, high redundancy
5. > 500000 species

Introduction

- We need for very fast, scalable, space efficient methods to cope with these vast amounts of information
- We often need to compare sequences to quantify how similar/dissimilar they are:
  - Fast similarity search: which sequences are similar to a given one?
  - Building phylogenies
  - Clustering sequences which are similar
  - . . .

Introduction

How do we compare two sequences?

- ANI (Average Nucleotide Identity) and other measures are quite reliable and used often in practice, but computationally expensive ($\mathcal{O}(m \cdot n)$)
- In practice, extract the set of k-mers of each sequence, and use the similarity of the sets of k-mers as a proxy for the true similarity
- A k-mer is a subsequence of k consecutive symbols (=nucleotides, in this context)
- Typical values for k in bioinformatics are $k = 21$ and $k = 31$ (too small k or too large are useless or computationally expensive)

Introduction

- Storing the sets of k-mers, and computing the similarity of the sets is too costly in terms of space and time
- If we keep just a "small" random sample (a.k.a. sketch), we can use these samples to infer the true similarity, using a lot less space and much faster
- This is the main idea behind `mash`, a standard *de facto* in genomic studies

Outline of the Talk

Similarity Estimation

- In abstract terms, we have a data stream

  $$\mathcal{Z} = z_1, z_2, \ldots, z_N,$$

  each $z_i \in \mathcal{U}$
- In our running example, $z_i = w_i \cdot w_{i+1} \cdots w_{i+k-1}$, the i-th k-mer in the DNA strand $w = w_1 \cdots w_L$, $1 \leqslant i \leqslant L - k + 1$
- The data stream $\mathcal{Z}$ (or the dataset $\{z_1, \ldots, z_N\}$) usually contains repetitions, each $z_i = x_j \in \mathcal{U}$for some j, and

  $$X = \{x_1, \ldots, x_n\}$$

  is the set of distinct elements in $\mathcal{Z}$. $n \leqslant N$ is the cardinality of $\mathcal{Z}$
- f_j denotes the frequency of x_j in $\mathcal{Z}$

Similarity Estimation

Consider two data streams or datasets $\mathcal{Z}_A$ and $\mathcal{Z}_B$. Let A and B denote their respective sets of distinct elements. Similarity between the two sets is often measured by their Jaccard index

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Another important measure is the containment index: how much of A is a subset of B

$$c(A, B) = \frac{|A \cap B|}{|A|}$$

Similarity Estimation

We can estimate similarity and containment from random samples S_A and S_B of the two streams. S_A and S_B must be “filtered” to obtain $\langle S'_A, S'_B \rangle = \phi(S_A, S_B)$ such that $S'_A \cup S'_B$ is a random sample of $A \cup B$ and $S'_A \cap S'_B$ is a random sample of $A \cap B$.

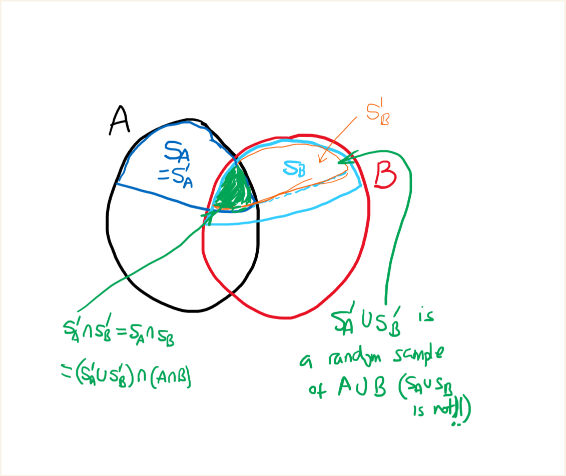
N.B. In general, $S_A \cup S_B$ is not a random sample of $A \cup B$ (likewise with $\cap$), but it is often easy to obtain the filtering step ϕ

Similarity Estimation

Theorem

1. $\mathbb{E}\left[J(S'_A, S'_B)\right] = J(A, B) = \frac{|A \cap B|}{|A \cup B|}$
2. $\mathbb{V}\left[J(S'_A, S'_B)\right] =$
   $J(A, B) \cdot (1 - J(A, B)) \cdot \left(\mathbb{E}\left[\frac{1}{|S'_A \cup S'_B|}\right] - \frac{1}{|A \cup B|}\right)$

Similarity Estimation



A
$S_A = S'_A$
S'_B
S_B
B
$S'_A \cap S'_B = S_A \cap S_B$
$= (S'_A \cup S'_B) \cap (A \cap B)$
$S'_A \cup S'_B$ is a random sample of $A \cup B$ ($S_A \cup S_B$ is not!!)

Estimating the size of the intersection

Using Affirmative Sampling with parameter b (stay tuned!) We can estimate the size of the intersection with:

$$Z_1 = \frac{|S_A \cap S_B|}{|S_A|} \cdot \left(b \left(1 + \frac{1}{b}\right)^{|S_A|-b+1} - 1 \right)$$

$$Z_2 = \frac{|S_A \cap S_B|}{|S_A|} \cdot \frac{|S_A|-1}{M_{S_A}}, \qquad M_{S_A} = \max\{\mathrm{hash}(z) \,|\, z \in S_A\}$$

$$\mathbb{E}[Z_1] = \mathbb{E}[Z_2] = |A \cap B|$$

N.B. No need to “filter” the samples

Other similarity measures

Jaccard's index	$\frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$
Otsuka-Ochiai (a.k.a. Cosine)	$\frac{\lvert A \cap B \rvert}{\sqrt{\lvert A \rvert \cdot \lvert B \rvert}}$
Sørensen-Dice	$2\frac{\lvert A \cap B \rvert}{\lvert A \rvert + \lvert B \rvert}$
Kulczynski 1	$\frac{\lvert A \cap B \rvert}{\lvert A \triangle B \rvert}$
Kulczynski 2	$\frac{1}{2}\left(\frac{\lvert A \cap B \rvert}{\lvert A \rvert} + \frac{\lvert A \cap B \rvert}{\lvert B \rvert}\right)$
Simpson	$\frac{\lvert A \cap B \rvert}{\min(\lvert A \rvert, \lvert B \rvert)}$
Braun-Blanquet	$\frac{\lvert A \cap B \rvert}{\max(\lvert A \rvert, \lvert B \rvert)}$
Correlation	$\cos^2(A, B) = \frac{\lvert A \cap B \rvert^2}{\lvert A \rvert \cdot \lvert B \rvert}$
...	...

Other similarity measures

The proof that works for Jaccard's similarity also works for containment, and a similar proofs can be derived many other similarity measures:

1. $\mathbb{E}\left[c(S_A, S_B)\right] = c(A, B) = |A \cap B|/|A|$
2. If σ is any of Jaccard, Kulczynski 2, correlation or Sørensen-Dice (M., Viola, Wang, 2022):
$$\mathbb{E}\left[\sigma(S'_A, S'_B)\right] = \sigma(A, B)$$
3. We also have asymptotically unbiased estimation for cosine similarity, Kulczynski 1 and many others (Sanz-González, M., 2025):
$$\begin{aligned}\mathbb{E}\left[\sigma(S'_A, S'_B)\right] - \sigma(A, B) &= \mathrm{bias}(\sigma(S'_A, S'_B)) \\ &= \mathcal{O}\left(\mathbb{E}\left[\frac{1}{|S'_A \cup S'_B|}\right]\right)\end{aligned}$$

Other similarity measures

1. For all the similarity measures σ mentioned, we also have (M., Viola, Wang, 2023, Sanz-González, M., 2025)

$$\mathbb{V}\left[\sigma(S'_A, S'_B)\right] = \mathcal{O}\left(\mathbb{E}\left[\frac{1}{|S'_A \cup S'_B|}\right]\right)$$

2. It follows that the mean square error (MSE) of the estimator $\hat{\sigma} = \sigma(S'_A, S'_B)$ also tends to 0 as the size of S_A and S_B grow

$$\text{MSE}(\hat{\sigma}) = \mathbb{V}\left[\hat{\sigma}\right] + (\text{bias}(\hat{\sigma}))^2 \to 0$$

Outline of the Talk

Drawing Random Samples

- In a random sample from the data stream (e.g., using the reservoir method) each distinct element x_j appears with relative frequency in the sample equal to its relative frequency f_j/N in the data stream $\Rightarrow$ needle-on-a-haystack
- Elements of low frequency will seldom be sampled, and we cannot keep exact counts as we don't know if the sampled elements have been "monitored" from the beginning

Drawing Random Samples

- In a random sample from the data stream (e.g., using the reservoir method) each distinct element x_j appears with relative frequency in the sample equal to its relative frequency f_j/N in the data stream $\Rightarrow$ needle-on-a-haystack
- Elements of low frequency will seldom be sampled, and we cannot keep exact counts as we don't know if the sampled elements have been "monitored" from the beginning

Random Sampling

- The distinct sampling problem is to draw a random sample of distinct elements, and it has many applications in data stream analysis
- For example, to estimate the number of k-elephants ($f_i \geqslant k$) or k-mice ($f_i < k$) in the stream we can draw a random sample of S distinct elements, together with their frequency counts
- Let S_P be the number of mice (or elephants) in the sample, and n_P the number of mice (or elephants) in the data stream. Then

$$\mathbb{E}\left[\frac{S_P}{S}\right] = \frac{n_P}{n}$$

Random Sampling

- The distinct sampling problem is to draw a random sample of distinct elements, and it has many applications in data stream analysis
- For example, to estimate the number of k-elephants ($f_i \geqslant k$) or k-mice ($f_i < k$) in the stream we can draw a random sample of S distinct elements, together with their frequency counts
- Let S_P be the number of mice (or elephants) in the sample, and n_P the number of mice (or elephants) in the data stream. Then

$$\mathbb{E}\left[\frac{S_P}{S}\right] = \frac{n_P}{n}$$

Random Sampling

- The distinct sampling problem is to draw a random sample of distinct elements, and it has many applications in data stream analysis
- For example, to estimate the number of k-elephants ($f_i \geqslant k$) or k-mice ($f_i < k$) in the stream we can draw a random sample of S distinct elements, together with their frequency counts
- Let S_P be the number of mice (or elephants) in the sample, and n_P the number of mice (or elephants) in the data stream. Then

$$\mathbb{E}\left[\frac{S_P}{S}\right] = \frac{n_P}{n}$$

Random Sampling

Let P some property.

- $n = \#$ of distinct elements in $\mathcal{Z}$
- $n_P = \#$ of distinct elements in $\mathcal{Z}$ that satisfy P
- $S =$ size of the sample $\Leftarrow$ in general, a r.v., assume $2 \leqslant S \leqslant n$
- $S_P = \#$ of elements in the sample that satisfy P

Theorem

1. $\mathbb{E}\left[\frac{S_P}{S}\right] = \frac{n_P}{n}$
2. $\mathbb{V}\left[\frac{S_P}{S}\right] = \frac{n_p}{n} \cdot \left(1 - \frac{n_p}{n}\right) \cdot \left(\mathbb{E}\left[\frac{1}{S}\right] - \frac{1}{n}\right)$

Random Sampling

Let P some property.

- $n = \#$ of distinct elements in $\mathcal{Z}$
- $n_P = \#$ of distinct elements in $\mathcal{Z}$ that satisfy P
- $S =$ size of the sample $\Leftarrow$ in general, a r.v., assume $2 \leqslant S \leqslant n$
- $S_P = \#$ of elements in the sample that satisfy P

Theorem

1. $\mathbb{E}\left[\frac{S_P}{S}\right] = \frac{n_P}{n}$
2. $\mathbb{V}\left[\frac{S_P}{S}\right] = \frac{n_p}{n} \cdot \left(1 - \frac{n_p}{n}\right) \cdot \left(\mathbb{E}\left[\frac{1}{S}\right] - \frac{1}{n}\right)$

Mergeable & Dependable Sampling

Definition

A sampling algorithm is dependable if every element returned has been part of the sample since its first occurrence and never kicked off; we can maintain exact frequency counts for each sampled element.

Definition

A sampling algorithm $\mathcal{A}$ is mergeable if for any partition of $\mathcal{Z}$ into disjoint substreams $\mathcal{Z}_1, \ldots, \mathcal{Z}_p$, the algorithm can be separately applied to obtain random samples $\mathcal{S}_i = \mathcal{A}(\mathcal{Z}_i)$, and the random samples $\mathcal{S}_i$ combined to obtain the same random sample that we would obtain by applying $\mathcal{A}$ to $\mathcal{Z}$.
We often have

$$\mathcal{A}(\mathcal{Z}_1 \cup \mathcal{Z}_2) = \mathcal{A}(\mathcal{A}(\mathcal{Z}_1) \cup \mathcal{A}(\mathcal{Z}_2))$$

Outline of the Talk

Bottom-b Sampling & MinHash

- Bottom-b sampling collects the elements with the b smallest hash values in the sequence
- Such b elements constitute a random sample of b distinct elements, because hash values behave as random numbers
- A similar algorithm that yields random samples (of hash values) of fixed size b is the famous MinHash
- For every element in the data stream/dataset, MinHash applies b hash functions and keeps the minimum value of each hash function seen so far

$$\text{minhash-sketch} = (v_1, \ldots, v_b)$$

with $v_i = \min\{\text{hash}_i(z_j) \mid 1 \leqslant j \leqslant N\}$

Bottom-$\mathfrak{b}$ Sampling & MinHash

- Bottom-$\mathfrak{b}$ sampling collects the elements with the $\mathfrak{b}$ smallest hash values in the sequence
- Such $\mathfrak{b}$ elements constitute a random sample of $\mathfrak{b}$ distinct elements, because hash values behave as random numbers
- A similar algorithm that yields random samples (of hash values) of fixed size $\mathfrak{b}$ is the famous MinHash
- For every element in the data stream/dataset, MinHash applies $\mathfrak{b}$ hash functions and keeps the minimum value of each hash function seen so far

$$\text{minhash-sketch} = (\nu_1, \ldots, \nu_{\mathfrak{b}})$$

with $\nu_i = \min\{\text{hash}_i(z_j) \mid 1 \leqslant j \leqslant N\}$

Bottom-$\mathfrak{b}$ Sampling & MinHash

- Bottom-$\mathfrak{b}$ sampling collects the elements with the $\mathfrak{b}$ smallest hash values in the sequence
- Such $\mathfrak{b}$ elements constitute a random sample of $\mathfrak{b}$ distinct elements, because hash values behave as random numbers
- A similar algorithm that yields random samples (of hash values) of fixed size $\mathfrak{b}$ is the famous MinHash
- For every element in the data stream/dataset, MinHash applies $\mathfrak{b}$ hash functions and keeps the minimum value of each hash function seen so far

$$\text{minhash-sketch} = (v_1, \ldots, v_{\mathfrak{b}})$$

with $v_i = \min\{\text{hash}_i(z_j) \mid 1 \leqslant j \leqslant N\}$

Bottom-$\mathfrak{b}$ Sampling & MinHash

- Bottom-$\mathfrak{b}$ sampling collects the elements with the $\mathfrak{b}$ smallest hash values in the sequence
- Such $\mathfrak{b}$ elements constitute a random sample of $\mathfrak{b}$ distinct elements, because hash values behave as random numbers
- A similar algorithm that yields random samples (of hash values) of fixed size $\mathfrak{b}$ is the famous MinHash
- For every element in the data stream/dataset, MinHash applies $\mathfrak{b}$ hash functions and keeps the minimum value of each hash function seen so far

$$\text{minhash-sketch} = (v_1, \dots, v_{\mathfrak{b}})$$

with $v_i = \min\{\text{hash}_i(z_j) \mid 1 \leqslant j \leqslant N\}$

Bottom-$\mathfrak{b}$ Sampling & MinHash

- It is very difficult to fix a value $\mathfrak{b}$ which will work well with data streams/datasets of many different sizes
- Bottom-$\mathfrak{b}$ and MinHash are simple, they can be very efficiently implemented, they need little space (samples of size $\mathfrak{b}$) and computation with the samples/sketches is lightning fast
- MinHash and bottom-$\mathfrak{b}$ samples are mergeable; bottom-$\mathfrak{b}$ is also dependable (MinHash is technically not, as it only keeps hashes)
- They are notoriously biased and inaccurate when estimating certain indices, namely, the containment index

Bottom-$\mathfrak{b}$ Sampling & MinHash

- It is very difficult to fix a value $\mathfrak{b}$ which will work well with data streams/datasets of many different sizes
- Bottom-$\mathfrak{b}$ and MinHash are simple, they can be very efficiently implemented, they need little space (samples of size $\mathfrak{b}$) and computation with the samples/sketches is lightning fast
- MinHash and bottom-$\mathfrak{b}$ samples are mergeable; bottom-$\mathfrak{b}$ is also dependable (MinHash is technically not, as it only keeps hashes)
- They are notoriously biased and inaccurate when estimating certain indices, namely, the containment index

Bottom-$\mathfrak{b}$ Sampling & MinHash

- It is very difficult to fix a value $\mathfrak{b}$ which will work well with data streams/datasets of many different sizes
- Bottom-$\mathfrak{b}$ and MinHash are simple, they can be very efficiently implemented, they need little space (samples of size $\mathfrak{b}$) and computation with the samples/sketches is lightning fast
- MinHash and bottom-$\mathfrak{b}$ samples are mergeable; bottom-$\mathfrak{b}$ is also dependable (MinHash is technically not, as it only keeps hashes)
- They are notoriously biased and inaccurate when estimating certain indices, namely, the containment index

Bottom-$\mathfrak{b}$ Sampling & MinHash

- It is very difficult to fix a value $\mathfrak{b}$ which will work well with data streams/datasets of many different sizes
- Bottom-$\mathfrak{b}$ and MinHash are simple, they can be very efficiently implemented, they need little space (samples of size $\mathfrak{b}$) and computation with the samples/sketches is lightning fast
- MinHash and bottom-$\mathfrak{b}$ samples are mergeable; bottom-$\mathfrak{b}$ is also dependable (MinHash is technically not, as it only keeps hashes)
- They are notoriously biased and inaccurate when estimating certain indices, namely, the containment index

FracMinHash

- In FracMinHash we keep all elements with hash value $h \leqslant \tau$, for some given threshold $\tau \in (0, 1]$
- FracMinHash produces samples of size Binomial(n, τ)
- The average size of the random samples is $n \cdot \tau$ (with high concentration around the mean)
- Similarity estimates using FracMinHash are extremely accurate because of the large size of the samples

FracMinHash

- The algorithm is very easy to implement and each item in the data stream can be very efficiently processed
- It is both dependable and mergeable; actually

$$F(\mathcal{Z}_1 \cup \mathcal{Z}_2) = F(\mathcal{Z}_1) \cup F(\mathcal{Z}_2)$$

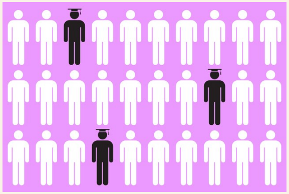
- Because of the large samples, the space used by the samples/sketches and the time to compute the similarity of samples, FracMinHash is computationally expensive —for instance, if compared to MinHash or bottom-b sampling

Sublinear Random Sampling

- Goal: obtain random sampling algorithms such that they return random samples of sublinear size (at least on expectation)
- Single pass, without knowledge of n
- Computationally efficient in terms of time and space
- Dependable and mergeable, if possible
- Two solutions
  1. Affirmative Sampling (Lumbroso, M., 2022)
  2. MaxGeomHash (Hera, Koslicki, M., 2025)

Outline of the Talk

Affirmative Sampling



- Early ideas date back to the original paper on Recordinality (2012); developed and analyzed in detail in (Lumbroso, M., 2022)
- The larger the cardinality (n) the larger the samples $\Rightarrow$ samples better represent diversity
- But we want samples to be sublinear in n (unlike FracMinHash)

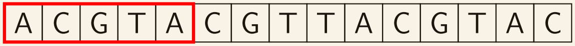
Affirmative Sampling

```
procedure AffirmativeSampling($\mathfrak{b}, \mathcal{Z}$)
    fill $\mathcal{S}$ with the first $\mathfrak{b}$ distinct elements
    (and hash values) of the stream $\mathcal{Z}$
    for $z \in \mathcal{Z}$ do
        if $z \in \mathcal{S}$ then
            Update $z$ stats; continue
        end if
        if Hash($z$) $>$ $\mathfrak{b}$-th largest hash value in $\mathcal{S}$ then
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{z\}$
        else if Hash($z$) $>$ min hash value in $\mathcal{S}$ then
            ▷ replace elem of min. hash in $\mathcal{S}$ with $z$
            $\mathcal{S} \leftarrow \mathcal{S} \setminus \{$elem. with min. hash in $\mathcal{S}\} \cup \{z\}$
        end if
    end for
    return $\mathcal{S}$
end procedure
```

Affirmative Sampling

- The size S of the sample $\mathcal{S}$ is a random variable = the number of b-records in a random permutation of size n $\Rightarrow$ $\mathbb{E}[S] = b\ln(n/b) + \mathcal{O}(1)$
- The sample does not contain the b-records, but the S elements with the largest hash values seen so far $\Rightarrow$ $\mathcal{S}$ is a random sample
- If $x \in \mathcal{S}$ then x has been added to S in its very first occurrence, and it has remained in $\mathcal{S}$ ever since $\Rightarrow$ can collect exact stats (e.g. frequency counts) for x

Affirmative Sampling

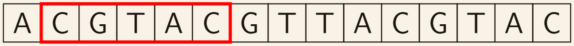


k-mer: **ACGTA** hash=0.1234

sample size = 1

k-mer	hash
ACGTA	0.1234
⋮	⋮

Affirmative Sampling

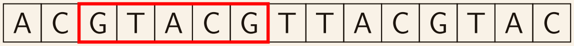


k-mer: **CGTAC** hash=0.5321

sample size = 2

k-mer	hash
ACGTA	0.1234
CGTAC	0.5321
⋮	⋮

Affirmative Sampling

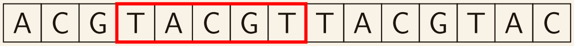


k-mer: **GTACG** hash=0.3456

sample size = 3

k-mer	hash
ACGTA	0.1234
GTACG	0.3456
CGTAC	0.5321
⋮	⋮

Affirmative Sampling

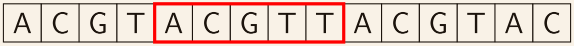


k-mer: **TACGT** hash=0.8765

sample size = 3

k-mer	hash
ACGTA	0.1234
GTACG	0.3456
CGTAC	0.5321
⋮	⋮

Affirmative Sampling

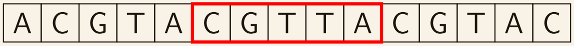


k-mer: **ACGTT** hash=0.2222

sample size = 4

k-mer	hash
ACGTA	0.1234
ACGTT	0.2222
GTACG	0.3456
CGTAC	0.5321
⋮	⋮

Affirmative Sampling

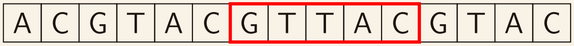


k-mer: **CGTTA** hash=0.6543

sample size = 4

k-mer	hash
ACGTA	0.1234
ACGTT	0.2222
GTACG	0.3456
CGTAC	0.5321
⋮	⋮

Affirmative Sampling

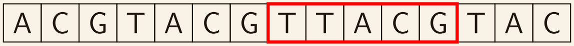


k-mer: **GTTAC** hash=0.1111

sample size = 5

k-mer	hash
GTTAC	0.1111
ACGTA	0.1234
ACGTT	0.2222
GTACG	0.3456
CGTAC	0.5321
	⋮

Affirmative Sampling

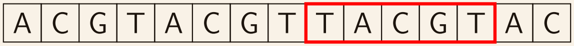


k-mer: **TTACG** hash=0.4444

sample size = 5

k-mer	hash
GTTAC	0.1111
ACGTA	0.1234
ACGTT	0.2222
GTACG	0.3456
TTACG	0.4444
	⋮

Affirmative Sampling

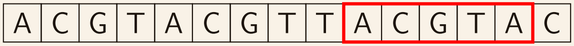


k-mer: **TACGT** hash=0.8765

sample size = 5

k-mer	hash
GTTAC	0.1111
ACGTA	0.1234
ACGTT	0.2222
GTACG	0.3456
TTACG	0.4444
	⋮

Affirmative Sampling

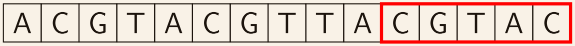


k-mer: **ACGTA**   hash=0.1234

sample size = 5

k-mer	hash
GTTAC	0.1111
ACGTA	0.1234
ACGTT	0.2222
GTACG	0.3456
TTACG	0.4444
	⋮

Affirmative Sampling



k-mer: **CGTAC** hash=0.5321

sample size = 5

k-mer	hash
GTTAC	0.1111
ACGTA	0.1234
ACGTT	0.2222
GTACG	0.3456
TTACG	0.4444
	⋮

Affirmative Sampling

- We also understand fairly well F = number of times an element substitutes another in the sample (not a b-record, but larger than some b-record):

$$\mathbb{E}[F] = b \ln^2(n/b) + \text{l.o.t.}$$

- Expected cost $C_{N,n}$ of Affirmative Sampling

$$\begin{aligned}\mathbb{E}[C_{N,n}] &= \Theta(N + (\mathbb{E}[S] + \mathbb{E}[F]) \log \mathbb{E}[S]) \\ &= \Theta(N + \log\log n \cdot \log^2 n)\end{aligned}$$

using appropriate data structures for the sample $\mathcal{S}$

Affirmative Sampling

- The variant α-Affirmative Sampling (α-AS), with $\alpha \in (0,1)$ chosen in advance, will add a new item z to the sample when $z \notin \mathcal{S}$ and $\mathrm{hash}(z)$ is among the $(100 \cdot \alpha)\%$ smaller hash values seen so far—instead of when $\mathrm{hash}(z)$ is among the $\mathfrak{b}$ smallest hash values seen so far
- For instance, if $\alpha = 1/2$ then an element not already in the sample will be added to the sample if its hash value is below the median of the hash values in the sample
- That modification yields that $\mathbb{E}\left[|\mathcal{S}|\right] = \Theta(n^{\alpha})$ and $\mathbb{V}\left[|\mathcal{S}|\right] = \Theta(n^{2\alpha})$, and explicit forms for the multiplicative constant factors exist (Langowski & Ward, 2019), e.g.,
$\mathbb{E}\left[|\mathcal{S}|\right] = \frac{2}{3}\sqrt{\pi n} + \text{l.o.t.} \approx 1.18\sqrt{n}$,
$\mathbb{V}\left[|\mathcal{S}|\right] = \left(2 - \frac{4\pi}{9}\right) n + \text{l.o.t.} \approx 0.6n$

Affirmative Sampling

- Both AS and α-AS render random samples; in α-AS there is no need to do replacements as long as we only return elements below the "threshold" as sample
- They are dependable; if an element is sampled it was put in the sample on its first occurrence, and if an element is ever discarded or evicted from the sample it will never be added to the sample again
- But, unfortunately, they are not mergeable, as the returned samples depend on the particular permutation of elements in the data stream
- One last disadvantage of α-AS is that the variance of $|\mathcal{S}|$ is $\Theta(\mathbb{E}[|\mathcal{S}|]^2) = \Theta(n^{2\alpha})$, hence we don't have concentration around the mean

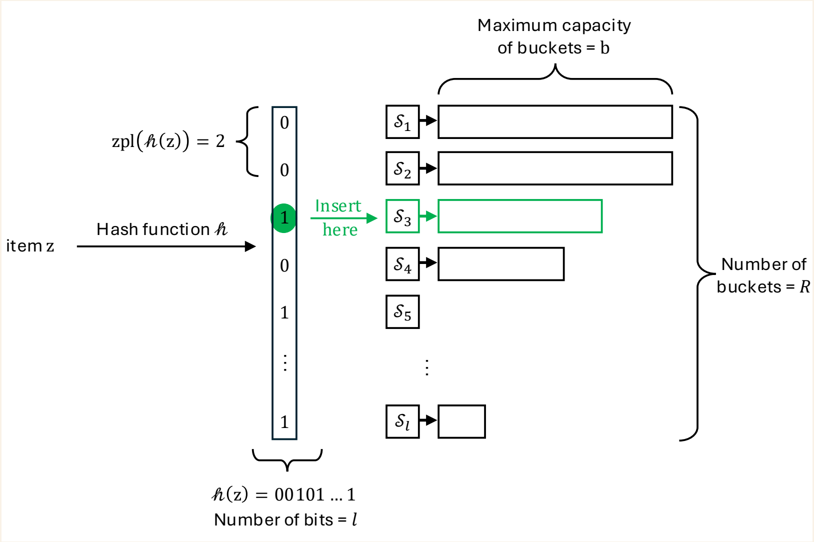
Outline of the Talk

MaxGeomHash

MaxGeomHash (MGH for short; Hera, Koslicki, M., 2025) combines ideas like those behind stochastic averaging, and LogLog and bottom-$\mathtt{b}$ sketches. MGH keeps a variable number of buckets $\mathcal{S}_1$, $\mathcal{S}_2$, … ad it also has a fixed parameter $\mathtt{b}$ given in advance.

1. For every $z \in \mathcal{S}$, compute its hash value $\mathtt{h}(z)$ (a bit string of ℓ bits) and find the position $\mathtt{i}$ of the leftmost 1 in $\mathtt{h}(z)$
2. Let $\mathtt{h}'$ be the trailing $\ell - \mathtt{i}$ bits in $\mathtt{h}(z)$ after the first 1
3. Apply bottom-$\mathtt{b}$ to decide if z is kept in bucket $\mathcal{S}_\mathtt{i}$ or not using $\mathtt{h}'$

MaxGeomHash



MaxGeomHash

- We assume $n^2 \ll 2^\ell$, so $\ell \gg 2 \lg n$
- For every distinct x_j in the dataset

$$\rho_j = \text{position of the leftmost 1 in } h(x_j) \sim \text{Geom}(1/2)$$

- Total number of buckets (N.B. some might be empty)

$$R = \max\{\rho_1, \ldots, \rho_n\}$$

- The maximum of n i.i.d. Geometric random variables has been very well studies (Szpankowski, Rego, 1990)

$$\mathbb{E}[R] = \lg n + \mathcal{O}(1), \qquad \mathbb{V}[R] = \Theta(1)$$

MaxGeomHash

- We assume $n^2 \ll 2^\ell$, so $\ell \gg 2 \lg n$
- For every distinct x_j in the dataset

$$\rho_j = \text{position of the leftmost 1 in } h(x_j) \sim \text{Geom}(1/2)$$

- Total number of buckets (N.B. some might be empty)

$$R = \max\{\rho_1, \ldots, \rho_n\}$$

- The maximum of n i.i.d. Geometric random variables has been very well studies (Szpankowski, Rego, 1990)

$$\mathbb{E}[R] = \lg n + \mathcal{O}(1), \qquad \mathbb{V}[R] = \Theta(1)$$

MaxGeomHash

- We assume $n^2 \ll 2^\ell$, so $\ell \gg 2\lg n$
- For every distinct x_j in the dataset

$$\rho_j = \text{position of the leftmost 1 in } h(x_j) \sim \text{Geom}(1/2)$$

- Total number of buckets (N.B. some might be empty)

$$R = \max\{\rho_1, \ldots, \rho_n\}$$

- The maximum of n i.i.d. Geometric random variables has been very well studies (Szpankowski, Rego, 1990)

$$\mathbb{E}[R] = \lg n + \mathcal{O}(1), \qquad \mathbb{V}[R] = \Theta(1)$$

MaxGeomHash

- We assume $n^2 \ll 2^\ell$, so $\ell \gg 2 \lg n$
- For every distinct x_j in the dataset

$$\rho_j = \text{position of the leftmost 1 in } h(x_j) \sim \text{Geom}(1/2)$$

- Total number of buckets (N.B. some might be empty)

$$R = \max\{\rho_1, \ldots, \rho_n\}$$

- The maximum of n i.i.d. Geometric random variables has been very well studies (Szpankowski, Rego, 1990)

$$\mathbb{E}[R] = \lg n + \mathcal{O}(1), \qquad \mathbb{V}[R] = \Theta(1)$$

MaxGeomHash

- Bucket $\mathcal{S}_i$ collects $Y_i = |\mathcal{S}_i|$ elements

$$Y_i \sim \min(\text{Binomial}(n, 2^{-i}), b)$$

  (an item z is a candidate to be sampled in the i-th bucket with probability $1/2^i$)

- $\mathbf{Y} = (Y_1, Y_2, \ldots)$ follows a multinomial with parameters n and $\mathbf{p} = (1/2, 1/4, 1/8, \ldots)$
- The size of the returned sample $\mathcal{S} = \cup_{i \geqslant i} \mathcal{S}_i$ is

$$S = \sum_{i \geqslant 1} Y_i$$

MaxGeomHash

- Bucket $\mathcal{S}_i$ collects $Y_i = |\mathcal{S}_i|$ elements

$$Y_i \sim \min(\text{Binomial}(n, 2^{-i}), b)$$

  (an item z is a candidate to be sampled in the i-th bucket with probability $1/2^i$)

- $\mathbf{Y} = (Y_1, Y_2, \ldots)$ follows a multinomial with parameters n and $\mathbf{p} = (1/2, 1/4, 1/8, \ldots)$
- The size of the returned sample $\mathcal{S} = \cup_{i \geqslant i} \mathcal{S}_i$ is

$$S = \sum_{i \geqslant 1} Y_i$$

MaxGeomHash

- Bucket $\mathcal{S}_i$ collects $Y_i = |\mathcal{S}_i|$ elements

$$Y_i \sim \min(\text{Binomial}(n, 2^{-i}), b)$$

  (an item z is a candidate to be sampled in the i-th bucket with probability $1/2^i$)
- $\mathbf{Y} = (Y_1, Y_2, \ldots)$ follows a multinomial with parameters n and $\mathbf{p} = (1/2, 1/4, 1/8, \ldots)$
- The size of the returned sample $\mathcal{S} = \cup_{i \geqslant i} \mathcal{S}_i$ is

$$S = \sum_{i \geqslant 1} Y_i$$

MaxGeomHash

- To do the analysis, consider three regimes: $i \ll i^*$, $i \approx i^*$ and $i \gg i^*$, with $i^* = \lg(n/b)$
- When $i \ll i^*$, we have $b \ll n/2^i$ and $Y_i = b$
- When $i \gg i^*$, we have $b \gg n/2^i$ and $\mathbb{E}[Y_i] = \mathbb{E}\left[\text{Binomial}(n, 2^{-i})\right] = n/2^i$
- We can use Poisson approximation to the binomial in the small region centered around i^*
- Collecting all contributions gives $\mathbb{E}[S] = b \lg(n/b) + O(1)$

MaxGeomHash

- To do the analysis, consider three regimes: $i \ll i^*$, $i \approx i^*$ and $i \gg i^*$, with $i^* = \lg(n/b)$
- When $i \ll i^*$, we have $b \ll n/2^i$ and $Y_i = b$
- When $i \gg i^*$, we have $b \gg n/2^i$ and $\mathbb{E}\left[Y_i\right] = \mathbb{E}\left[\text{Binomial}(n, 2^{-i})\right] = n/2^i$
- We can use Poisson approximation to the binomial in the small region centered around i^*
- Collecting all contributions gives $\mathbb{E}\left[S\right] = b \lg(n/b) + \mathcal{O}(1)$

MaxGeomHash

- To do the analysis, consider three regimes: $i \ll i^*$, $i \approx i^*$ and $i \gg i^*$, with $i^* = \lg(n/b)$
- When $i \ll i^*$, we have $b \ll n/2^i$ and $Y_i = b$
- When $i \gg i^*$, we have $b \gg n/2^i$ and $\mathbb{E}[Y_i] = \mathbb{E}\left[\text{Binomial}(n, 2^{-i})\right] = n/2^i$
- We can use Poisson approximation to the binomial in the small region centered around i^*
- Collecting all contributions gives $\mathbb{E}[S] = b \lg(n/b) + \mathcal{O}(1)$

MaxGeomHash

- To do the analysis, consider three regimes: $i \ll i^*$, $i \approx i^*$ and $i \gg i^*$, with $i^* = \lg(n/b)$
- When $i \ll i^*$, we have $b \ll n/2^i$ and $Y_i = b$
- When $i \gg i^*$, we have $b \gg n/2^i$ and $\mathbb{E}[Y_i] = \mathbb{E}\left[\text{Binomial}(n, 2^{-i})\right] = n/2^i$
- We can use Poisson approximation to the binomial in the small region centered around i^*
- Collecting all contributions gives $\mathbb{E}[S] = b \lg(n/b) + \mathcal{O}(1)$

MaxGeomHash

- To do the analysis, consider three regimes: $i \ll i^*$, $i \approx i^*$ and $i \gg i^*$, with $i^* = \lg(n/b)$
- When $i \ll i^*$, we have $b \ll n/2^i$ and $Y_i = b$
- When $i \gg i^*$, we have $b \gg n/2^i$ and $\mathbb{E}[Y_i] = \mathbb{E}\left[\text{Binomial}(n, 2^{-i})\right] = n/2^i$
- We can use Poisson approximation to the binomial in the small region centered around i^*
- Collecting all contributions gives $\mathbb{E}[S] = b \lg(n/b) + \mathcal{O}(1)$

MaxGeomHash

Theorem (Hera, Koslicki, M., 2025)

MGH with parameter $b \geqslant 1$ *will produce a random sample* $\mathcal{S}$ *of size* $S = |\mathcal{S}| = \sum_{1 \leqslant i \leqslant R} |\mathcal{S}_i|$ *such that*

$$\mathbb{E}[S] = \mathbb{E}\left[\sum_{i=1}^{R} |\mathcal{S}_i|\right] = b \lg(n/b) + b + \epsilon_{n,b}$$

where $R = \max\{\rho_j \mid 1 \leqslant j \leqslant n\}$, $\{\rho_j\}_{1 \leqslant j \leqslant n}$ *are i.i.d. Geom*$(1/2)$ *random variables, and* $\epsilon_{n,b} \leqslant E_b + o(1)$ *with* $E_b = \sum_{m=0}^{b-1} (b-m) e^{-b} \frac{b^m}{m!}$. *Moreover,* $\mathbb{V}[S] = \Theta(1)$.

α-MaxGeomHash

By setting different maximum capacities f_i of the buckets, we achieve different expected sizes

$$Y_i = |\mathcal{S}_i| \sim \min\left(\text{Binomial}(n, 2^{-i}), f_i\right)$$

- $f_i = b$, for all $i \Rightarrow$ standard MGH.
- $f_i = \lceil 2^{\beta i} \rceil$, $\beta > 0$ a real number yields α-MGH with

$$\alpha = \frac{\beta}{1+\beta}, \qquad 0 < \alpha < 1$$

  (*) $f_i = 2^i \Rightarrow \alpha = 1/2$
- α-MGH samples are of expected size $\Theta(n^\alpha)$
- The trick for the analysis is again to consider three regions around the *critical* index $i^* = \frac{1}{1+\beta} \lg n$, where $n/2^i = f_i$

α-MaxGeomHash

By setting different maximum capacities f_i of the buckets, we achieve different expected sizes

$$Y_i = |\mathcal{S}_i| \sim \min\left(\text{Binomial}(n, 2^{-i}), f_i\right)$$

- $f_i = b$, for all $i \Rightarrow$ standard MGH.
- $f_i = \lceil 2^{\beta i} \rceil$, $\beta > 0$ a real number yields α-MGH with

$$\alpha = \frac{\beta}{1+\beta}, \qquad 0 < \alpha < 1$$

  (*) $f_i = 2^i \Rightarrow \alpha = 1/2$
- α-MGH samples are of expected size $\Theta(n^\alpha)$
- The trick for the analysis is again to consider three regions around the *critical* index $i^* = \frac{1}{1+\beta} \lg n$, where $n/2^i = f_i$

α-MaxGeomHash

By setting different maximum capacities f_i of the buckets, we achieve different expected sizes

$$Y_i = |\mathcal{S}_i| \sim \min\left(\mathrm{Binomial}(n, 2^{-i}), f_i\right)$$

- $f_i = b$, for all $i \Rightarrow$ standard MGH.
- $f_i = \lceil 2^{\beta i} \rceil$, $\beta > 0$ a real number yields α-MGH with

$$\alpha = \frac{\beta}{1+\beta}, \qquad 0 < \alpha < 1$$

(*) $f_i = 2^i \Rightarrow \alpha = 1/2$

- α-MGH samples are of expected size $\Theta(n^{\alpha})$
- The trick for the analysis is again to consider three regions around the *critical* index $i^* = \frac{1}{1+\beta} \lg n$, where $n/2^i = f_i$

α-MaxGeomHash

By setting different maximum capacities f_i of the buckets, we achieve different expected sizes

$$Y_i = |\mathcal{S}_i| \sim \min\left(\text{Binomial}(n, 2^{-i}), f_i\right)$$

- $f_i = b$, for all $i \Rightarrow$ standard MGH.
- $f_i = \lceil 2^{\beta i} \rceil$, $\beta > 0$ a real number yields α-MGH with

$$\alpha = \frac{\beta}{1+\beta}, \qquad 0 < \alpha < 1$$

  (*) $f_i = 2^i \Rightarrow \alpha = 1/2$
- α-MGH samples are of expected size $\Theta(n^\alpha)$
- The trick for the analysis is again to consider three regions around the *critical* index $i^* = \frac{1}{1+\beta} \lg n$, where $n/2^i = f_i$

α-MaxGeomHash

Theorem (Hera, Koslicki, M., 2025)

α-MGH with parameter $\alpha \in (0,1)$ will produce a random sample $\mathcal{S}$ of size S such that

$$\mathbb{E}[S] = \left(\frac{2^{1/(1-\alpha)} - 1}{2^{\alpha/(1-\alpha)} - 1}\right) \cdot n^{\alpha} + \mathcal{O}(\log n)$$

$$\approx \left(\frac{1-\alpha}{\alpha \ln 2} + \frac{3}{2} + \frac{\alpha \ln 2}{12(1-\alpha)}\right) \cdot n^{\alpha}.$$

Moreover $\mathbb{V}[S] = \Theta(n^{\alpha})$

MaxGeomHash

- Both variants of MGH are dependable
- Both are mergeable: just merge the buckets of the two subsamples to obtain a new random sample; whatever f_i is, we always have

$$\mathsf{MGH}(A \cup B) = \mathsf{MGH}(\mathsf{MGH}(A) \cup \mathsf{MGH}(B))$$

- Likewise with intersections. Unions and intersections can be carried out on a bucket-by-bucket basis: merge $\mathcal{S}_i(A)$ with $\mathcal{S}_i(B)$ for every $1 \leqslant i \leqslant \max\{R_A, R_B\}$

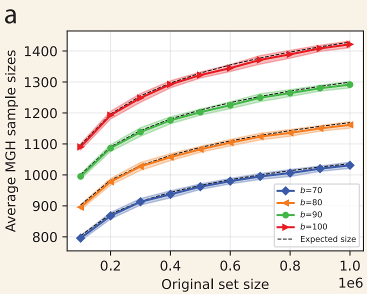
Random Sampling at a Glance

Algorithm	Dep./Merg.?	Param.	Size
MinHash	yes/yes	$b \in \mathbb{N}$	b $(*)$
ModHash	yes/yes	$m \in \mathbb{N}$	n/m
Reservoir Sampling $(**)$	no/no	$b \in \mathbb{N}$	$\leqslant b$ $(*)$
FracMinHash	yes/yes	$\tau \in (0,1)$	$n \cdot \tau$
Affirmative Sampling	yes/no	$b \in \mathbb{N}$	$b \ln(n/b)$
α-AS	yes/no	$\alpha \in (0,1)$	$\Theta(n^{\alpha})$
MaxGeomHash	yes/yes	$b \in \mathbb{N}$	$b \log_2(n/b)$
α-MGH	yes/yes	$\alpha \in (0,1)$	$\frac{2^{1/(1-\alpha)}-1}{2^{\alpha/(1-\alpha)}-1} \cdot n^{\alpha}$

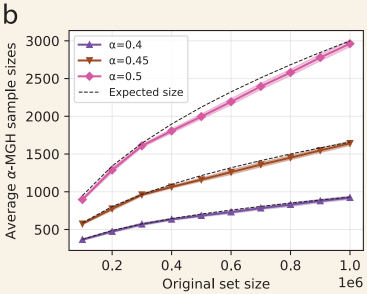
- $(*)$ Deterministic upper bound or not random
- $(**)$ Repetitions removed for distinct sampling

Outline of the Talk

Average size of MGH



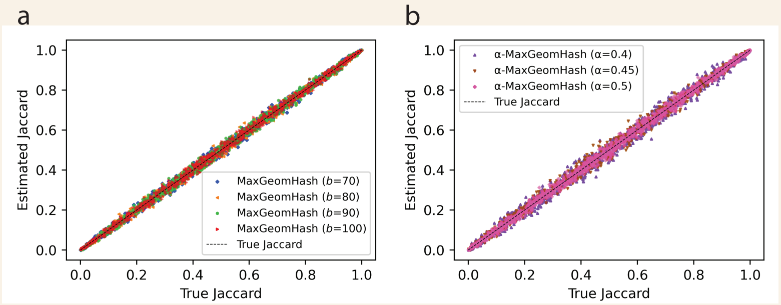
(a) MGH



(b) α-MGH

Solid = average size (empirical, 50 trials)
Shadow = standard deviation
Dashed = theoretical value

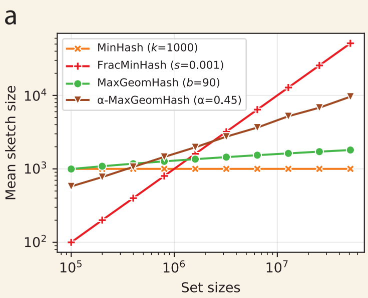
Estimating Jaccard similarity



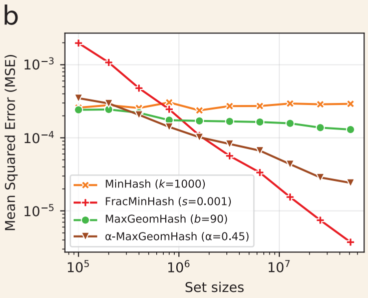
(a) MGH (b) α-MGH

Estimated vs true Jaccard similarities

Estimating Jaccard similarity



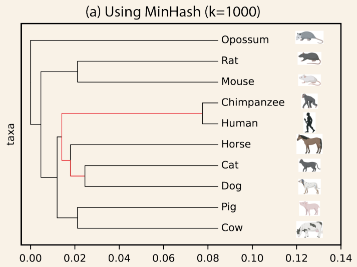
(a) Sample size

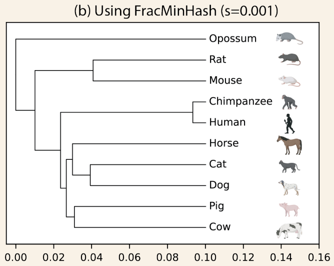


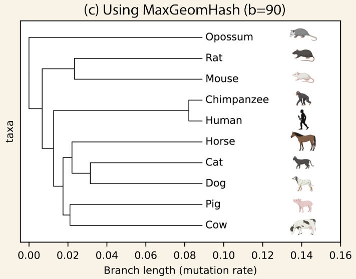
(b) Mean square error

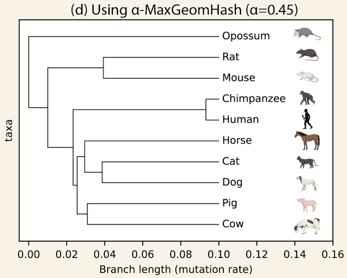
Accuracy vs sizes (Jaccard similarity = 0.5)

Estimating phylogenies





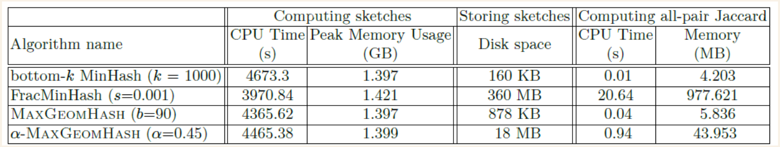




Estimating phylogenies

- Similarity trees for ten mammal species using k-mer ($k = 31$) sketches from the respective genome assemblies, using MinHash, FracMinHash, MaxGeomHash and α-MGH.
- Trees were constructed from estimated pairwise Jaccard similarities ($\Rightarrow$ average mutation rates = 1-ANI) using UPGMA (Unweighted Pair Group Method with Arithmetic mean).
- MinHash (a) misclassifies *Carnivora* (cats, dogs) and horses as closer to *Primates* than to pigs and cows.
  FracMinHash (b), MGH (c) and α-MGH do better and infer a closest solution to the correct philogeny (Rodents and Primates are wrongly put apart in all four trees). Estimates of the branch lengths are more accurate in (b) and (d).

Estimating phylogenies



	Computing sketches		Storing sketches	Computing all-pair Jaccard	
Algorithm name	CPU Time (s)	Peak Memory Usage (GB)	Disk space	CPU Time (s)	Memory (MB)
bottom-k MinHash (k = 1000)	4673.3	1.397	160 KB	0.01	4.203
FracMinHash (s=0.001)	3970.84	1.421	360 MB	20.64	977.621
MaxGeomHash (b=90)	4365.62	1.397	878 KB	0.04	5.836
α-MaxGeomHash (α=0.45)	4465.38	1.399	18 MB	0.94	43.953

Computational resources in the phylogeny experiment

Thanks a lot
for your attention!