

# **Efficient sequence similarity searches with strobemers and applications in read mapping**

Kristoffer Sahlin

Department of Mathematics, Science for Life Laboratory  
Stockholm University

# Finding similarity between sequences

- Used in:
  - Sequence {mapping, clustering, assembly}
  - SNP and SV detection
  - Storage and retrieval
- Larger datasets and more redundant databases
- I will discuss read mapping to reference(s)

# Finding similarity between sequences

- Used in:
  - Sequence {mapping, clustering, assembly}
  - SNP and SV detection
  - Storage and retrieval
- Larger datasets and more redundant databases
- I will discuss read mapping to reference(s)

## Read mapping

- Lossless approaches (guaranteed best match):
    - Pairwise alignment (slow)
    - Efficient search schemes for up to X ( $\leq 4$ ) errors.
      - Pigeonhole principle with ILP
  - Lossy approaches (seed–filter–extend)
    - FM-index: BWA, Bowtie2, ...
    - Hash table: BLAT, minimap2, Winnowmap2, ...
- ← This talk

# Hash-based seed-filter-extend

## Indexing

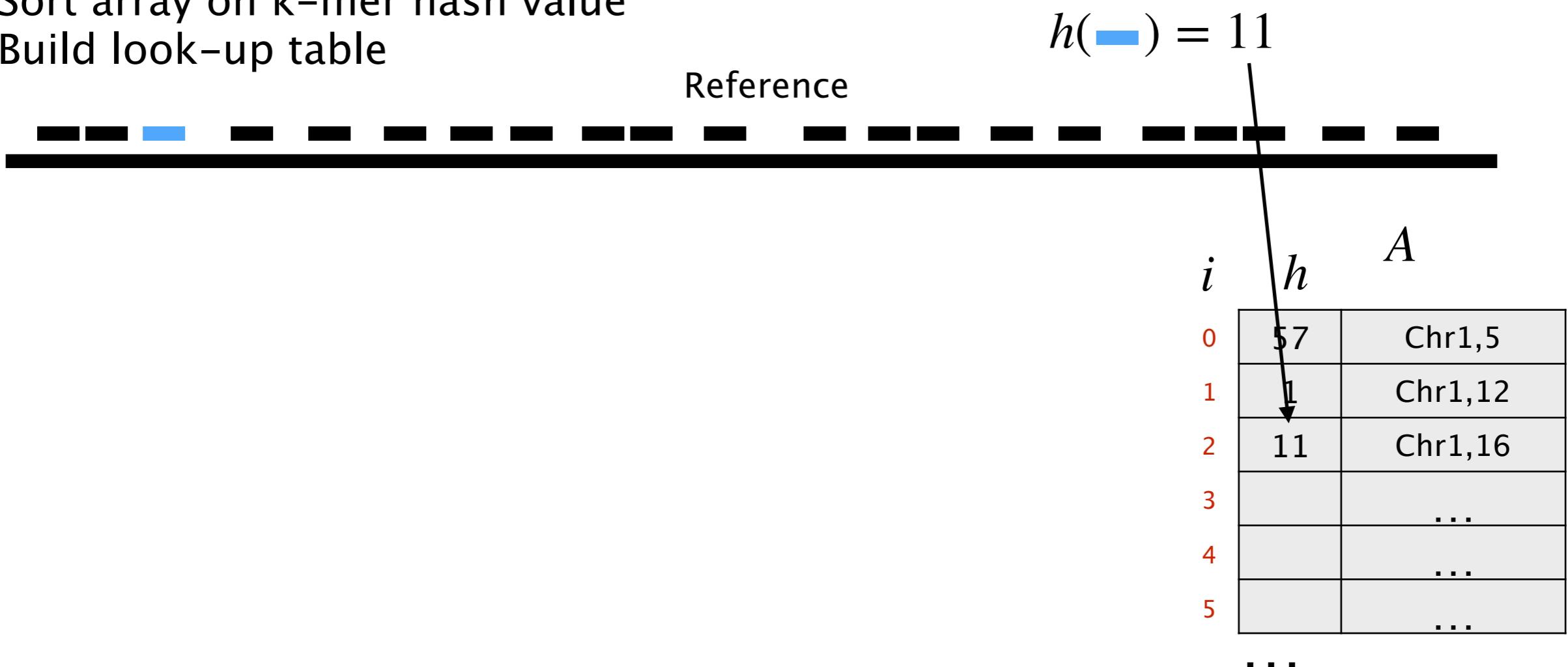
1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



# Hash-based seed-filter-extend

## Indexing

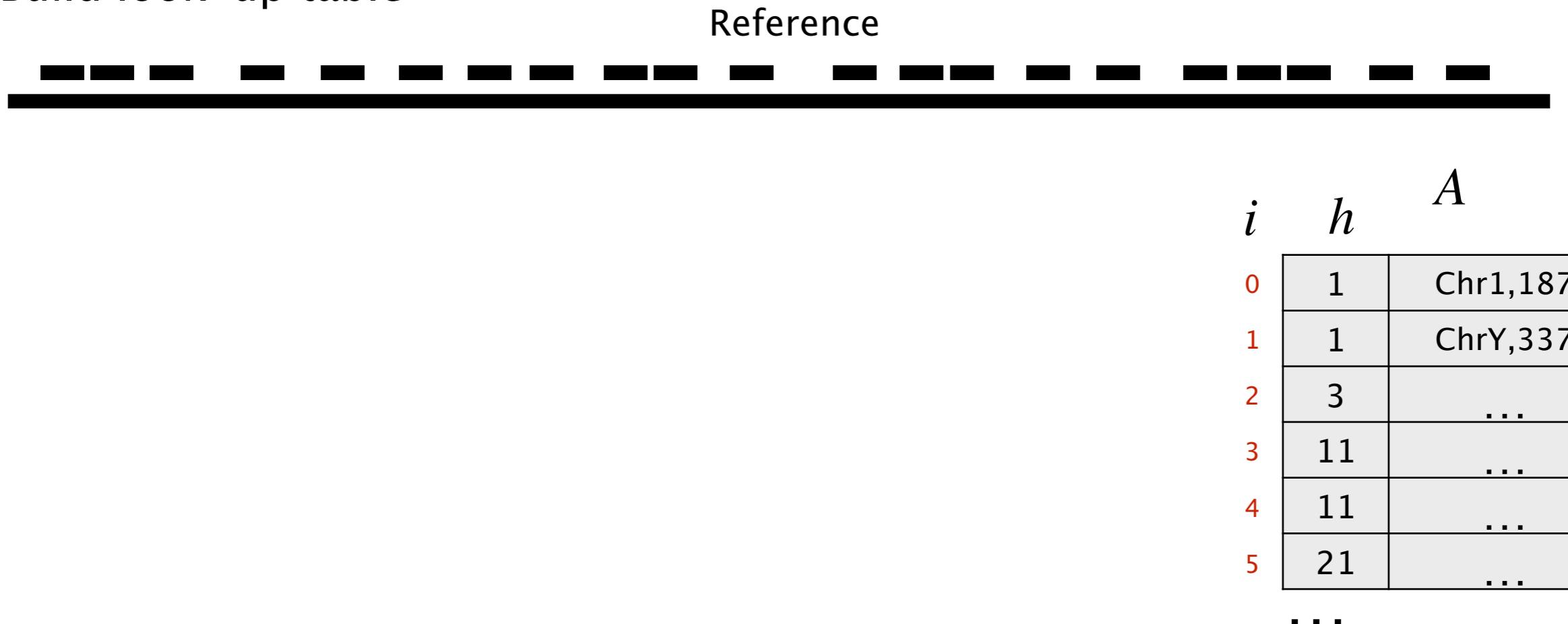
1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



# Hash-based seed-filter-extend

## Indexing

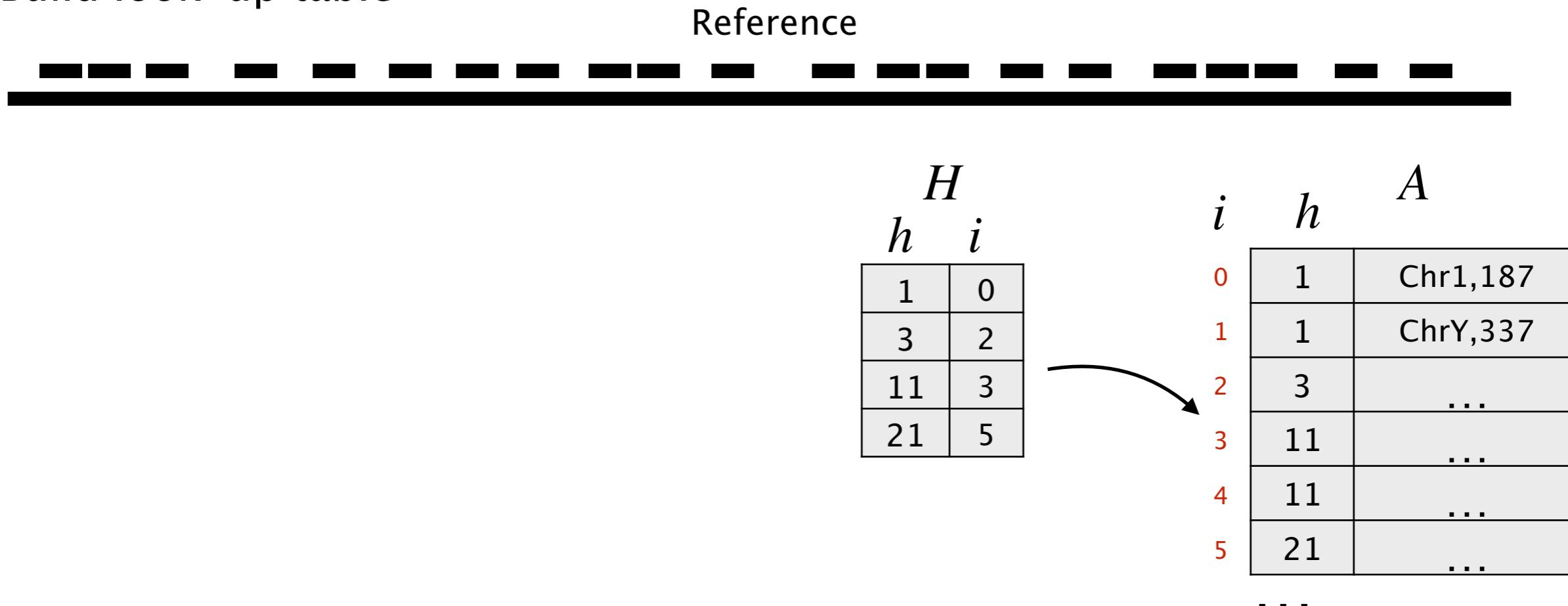
1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



# Hash-based seed-filter-extend

## Indexing

1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



# Hash-based seed-filter-extend

## Indexing

1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



## Mapping

1. Seed: Find shared k-mers
2. Filter: Decide regions of interest
3. Extend: Do pairwise alignment

Query



$$h(\text{---}) = 11$$

Reference

$H$	
$h$	$i$
1	0
3	2
11	3
21	5

$i$	$h$	$A$
0	1	Chr1,187
1	1	ChrY,337
2	3	...
3	11	...
4	11	...
5	21	...
...		

# Hash-based seed-filter-extend

## Indexing

1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



## Mapping

1. Seed: Find shared k-mers
2. Filter: Decide regions of interest
3. Extend: Do pairwise alignment

Query



$$h(\text{---}) = 11$$

Reference

$H$	
$h$	$i$
1	0
3	2
11	3
21	5

$i$	$h$	$A$
0	1	Chr1,187
1	1	ChrY,337
2	3	...
3	11	...
4	11	...
5	21	...
...		

# Hash-based seed-filter-extend

## Indexing

1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



## Mapping

1. Seed: Find shared k-mers
2. Filter: Decide regions of interest
3. Extend: Do pairwise alignment

Query



	$H$		$A$
	$h$	$i$	
0	1	0	Chr1,187
1	1	2	ChrY,337
2	3		...
3	11		...
4	11		...
5	21		...
			...

# Hash-based seed-filter-extend

## Indexing

1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



## Mapping

1. Seed: Find shared k-mers
2. Filter: Decide regions of interest
3. Extend: Do pairwise alignment

Query



Reference

$H$	$i$
$h$	
1	0
3	2
11	3
21	5

$i$	$h$	$A$
0	1	Chr1,187
1	1	ChrY,337
2	3	...
3	11	...
4	11	...
5	21	...
...		

# Hash-based seed-filter-extend

## Indexing

1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



## Mapping

### This talk

1. Seed: Find shared k-mers subsequences
2. Filter: Decide regions of interest
3. Extend: Do pairwise alignment

### Query



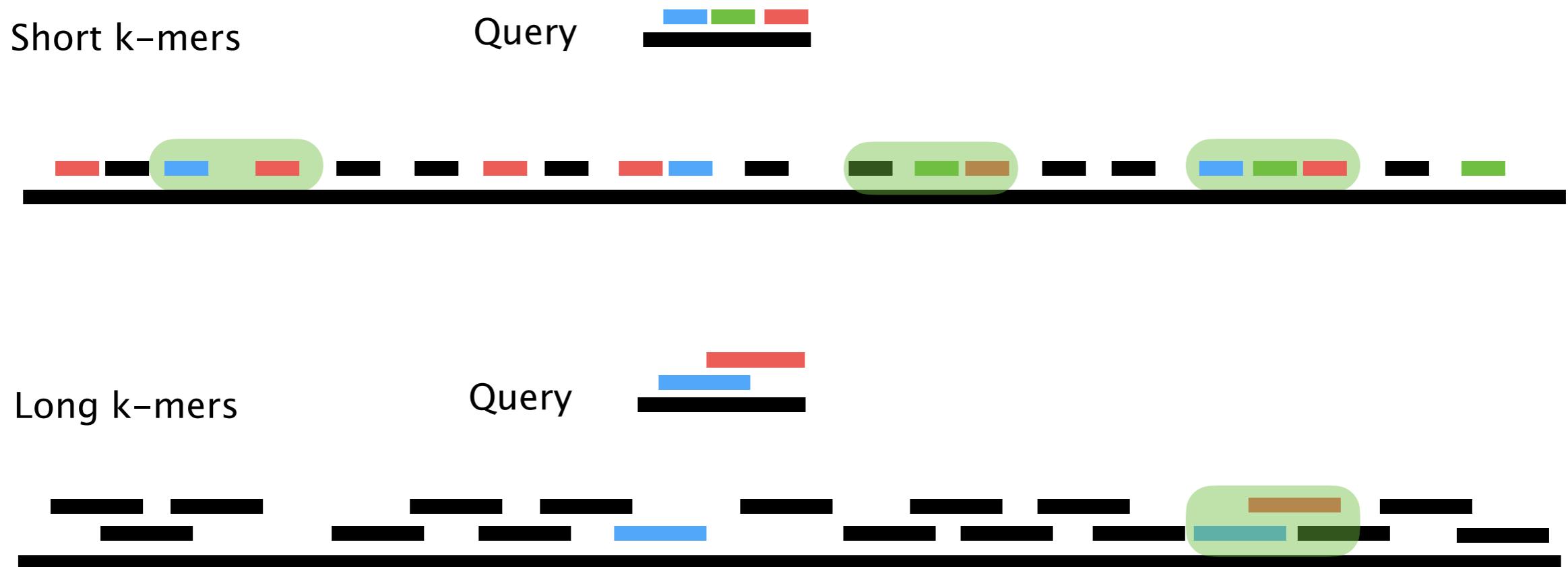
Reference

$H$	
$h$	$i$
1	0
3	2
11	3
21	5

$i$	$h$	$A$
0	1	Chr1,187
1	1	ChrY,337
2	3	...
3	11	...
4	11	...
5	21	...
...		

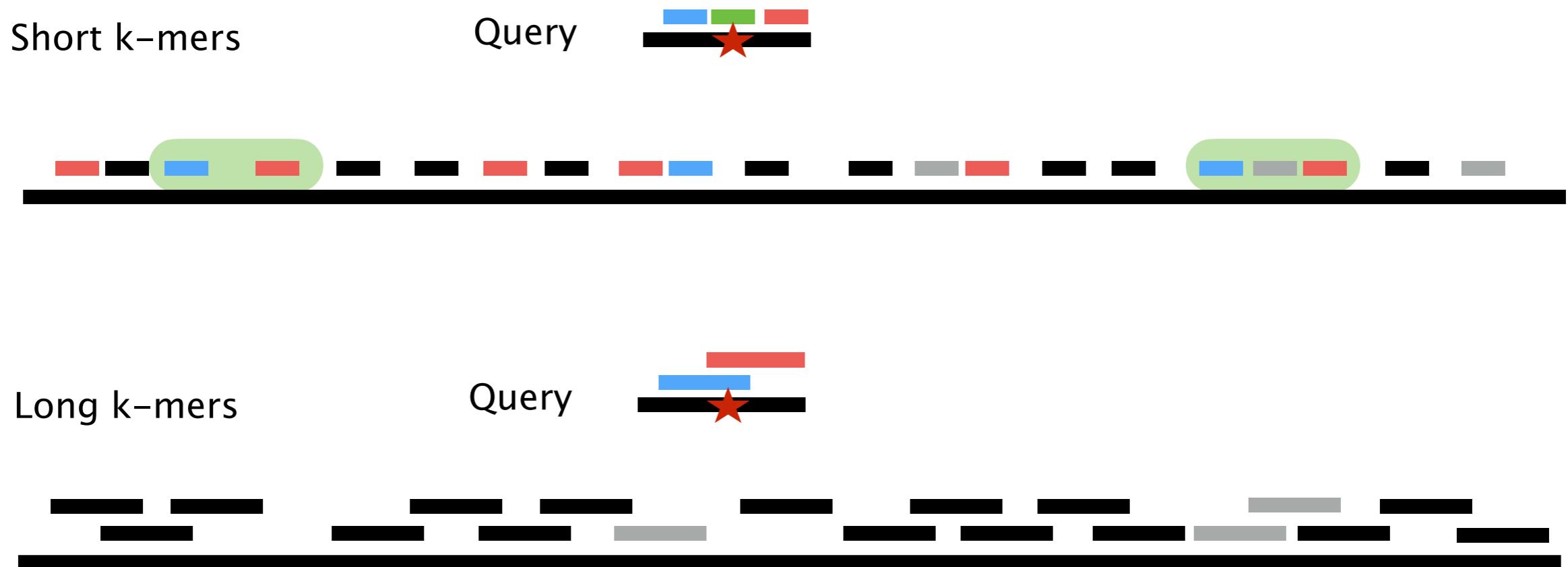
# Sensitivity-specificity trade-off

- Short k-mers: high sensitivity, low specificity
- Long k-mers: low sensitivity, high specificity



# Sensitivity-specificity trade-off

- Short k-mers: high sensitivity, low specificity
- Long k-mers: low sensitivity, high specificity



# Approximate-matching (fuzzy) seeds

- Spaced k-mers<sup>1</sup>
- Vector seeds<sup>2</sup>
- Permutation techniques<sup>3,4</sup>
- Covering template families<sup>5</sup>
- SimHash<sup>3</sup> (BLEND<sup>6</sup>)
- TensorSketch<sup>7</sup>
- SubSeqHash<sup>8</sup>
- Neighbouring minimizers<sup>9</sup>
- k-min-mers<sup>10</sup>
- Strobemers<sup>11</sup>



ACTC GGTA CTAG  
GACT TGGT CCTA  
CGAC CTGG ACCT  
ACGA TCTG TACC AGAC  
CACG CTCT GTAC TAGA  
CACCGACTCTGGTACCTAGAC  
||| ||| ||| ||| ||| |||  
CACGG**C**TC**A**GGT-**C**CTAGAC

<sup>1</sup> Ma et al., 2002

<sup>2</sup> Brejová et al., 2003

<sup>3</sup> Charikar, 2002

<sup>4</sup> Lederman, 2013

<sup>5</sup> Giladi et al., 2010

<sup>6</sup> Firtina et al., 2021

<sup>7</sup> Joudaki et al., 2020

<sup>8</sup> Li et al., 2023

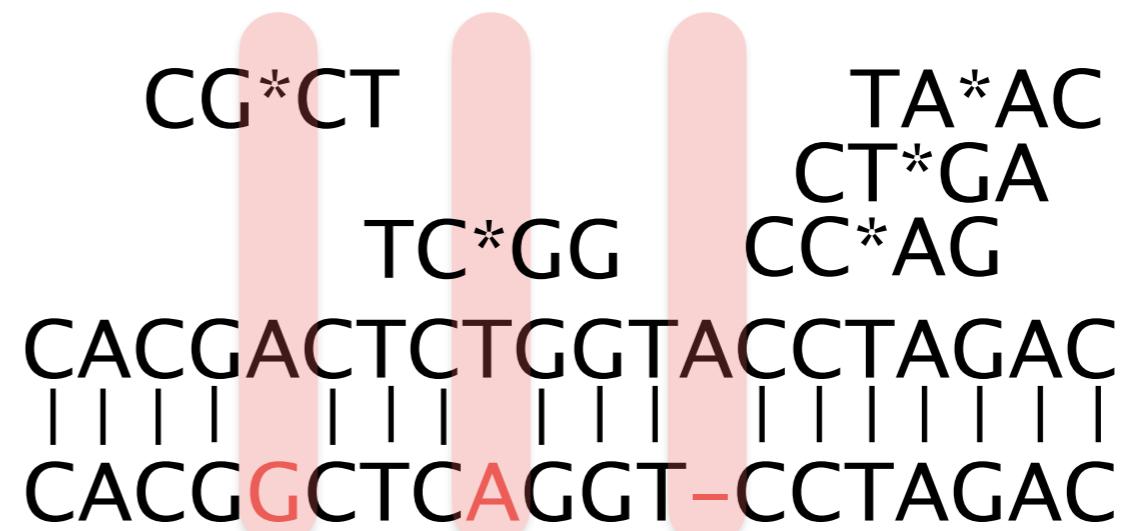
<sup>9</sup> Chin and Khalak, 2019

<sup>10</sup> Ekim, Berger, Chikhi, 2021

<sup>11</sup> Sahlin, 2021

# Approximate-matching (fuzzy) seeds

- Spaced k-mers<sup>1</sup>
  - Vector seeds<sup>2</sup>
  - Permutation techniques<sup>3,4</sup>
  - Covering template families<sup>5</sup>
  - SimHash<sup>3</sup> (BLEND<sup>6</sup>)
  - TensorSketch<sup>7</sup>
  - SubSeqHash<sup>8</sup>
  - Neighbouring minimizers<sup>9</sup>
  - k-min-mers<sup>10</sup>
  - Strobemers<sup>11</sup>
- + Match over substitutions
  - Do not handle indels



<sup>1</sup> Ma et al., 2002

<sup>2</sup> Brejová et al., 2003

<sup>3</sup> Charikar, 2002

<sup>4</sup> Lederman, 2013

<sup>5</sup> Giladi et al., 2010

<sup>6</sup> Firtina et al., 2021

<sup>7</sup> Joudaki et al., 2020

<sup>8</sup> Li et al., 2023

<sup>9</sup> Chin and Khalak, 2019

<sup>10</sup> Ekim, Berger, Chikhi, 2021

<sup>11</sup> Sahlin, 2021

# Approximate-matching (fuzzy) seeds

- Spaced k-mers<sup>1</sup>
  - Vector seeds<sup>2</sup>
  - Permutation techniques<sup>3,4</sup>
  - Covering template families<sup>5</sup>
  - SimHash<sup>3</sup> (BLEND<sup>6</sup>)
  - TensorSketch<sup>7</sup>
  - SubSeqHash<sup>8</sup>
  - **Neighbouring minimizers<sup>9</sup>**
  - k-min-mers<sup>10</sup>
  - Strobemers<sup>11</sup>

ACTC GGT A CTAG  
GACT TGGT CCTA  
CGAC CTGG ACCT  
ACGA TCTG TACC AGAC  
CACG CTCT GTAC TAGA  
CACG ACTCTGGTACCTAGAC  
||| ||| ||| |||  
CACG ACTCTGG -- CCTAGAC

<sup>1</sup> Ma et al., 2002

<sup>2</sup> Brejová et al., 2003

<sup>3</sup> Charikar, 2002

<sup>4</sup> Lederman, 2013

<sup>5</sup> Giladi et al., 2010

<sup>6</sup> Firtina et al., 2021

<sup>7</sup> Joudaki et al., 2020

<sup>8</sup> Li et al., 2023

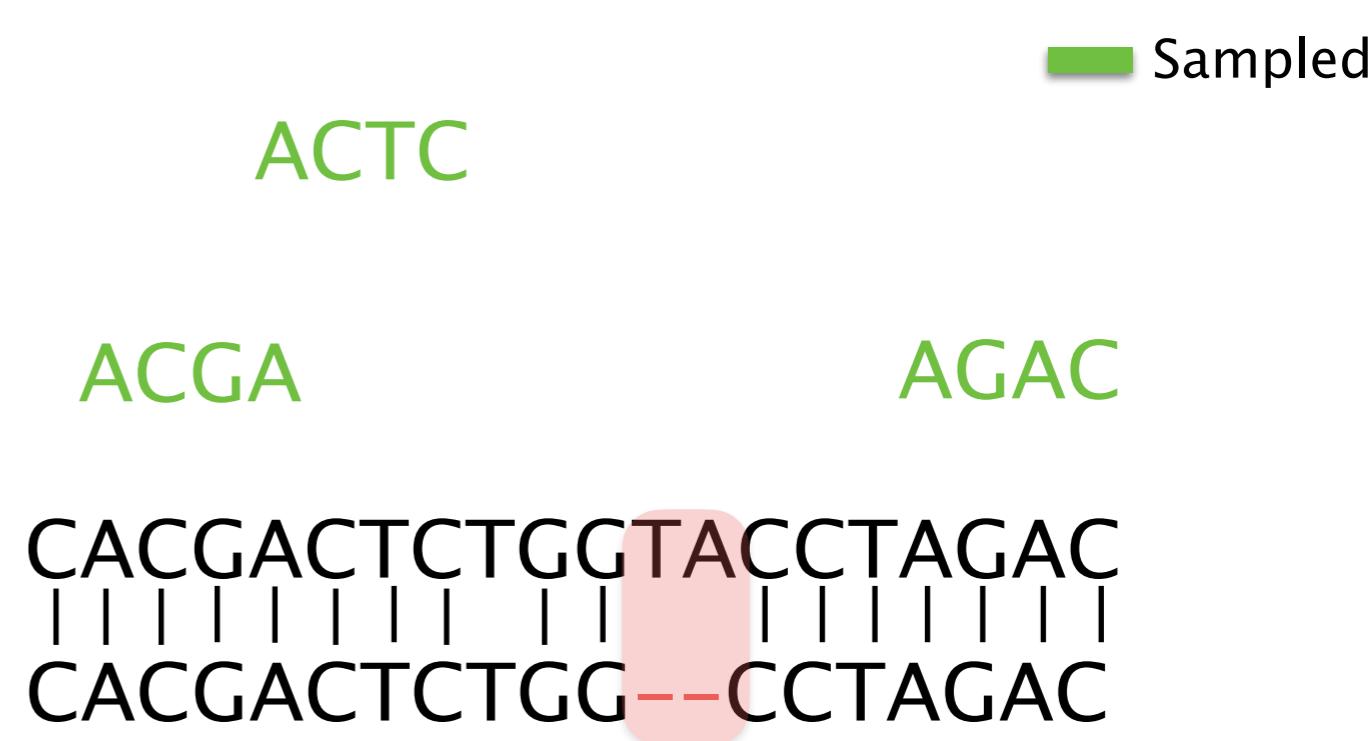
<sup>9</sup> Chin and Khalak, 2019

<sup>10</sup> Ekim, Berger, Chikhi, 2021

<sup>11</sup> Sahlin, 2021

# Approximate-matching (fuzzy) seeds

- Spaced k-mers<sup>1</sup>
- Vector seeds<sup>2</sup>
- Permutation techniques<sup>3,4</sup>
- Covering template families<sup>5</sup>
- SimHash<sup>3</sup> (BLEND<sup>6</sup>)
- TensorSketch<sup>7</sup>
- SubSeqHash<sup>8</sup>
- **Neighbouring minimizers<sup>9</sup>**
- k-min-mers<sup>10</sup>
- Strobemers<sup>11</sup>



<sup>1</sup> Ma et al., 2002

<sup>2</sup> Brejová et al., 2003

<sup>3</sup> Charikar, 2002

<sup>4</sup> Lederman, 2013

<sup>5</sup> Giladi et al., 2010

<sup>6</sup> Firtina et al., 2021

<sup>7</sup> Joudaki et al., 2020

<sup>8</sup> Li et al., 2023

<sup>9</sup> Chin and Khalak, 2019

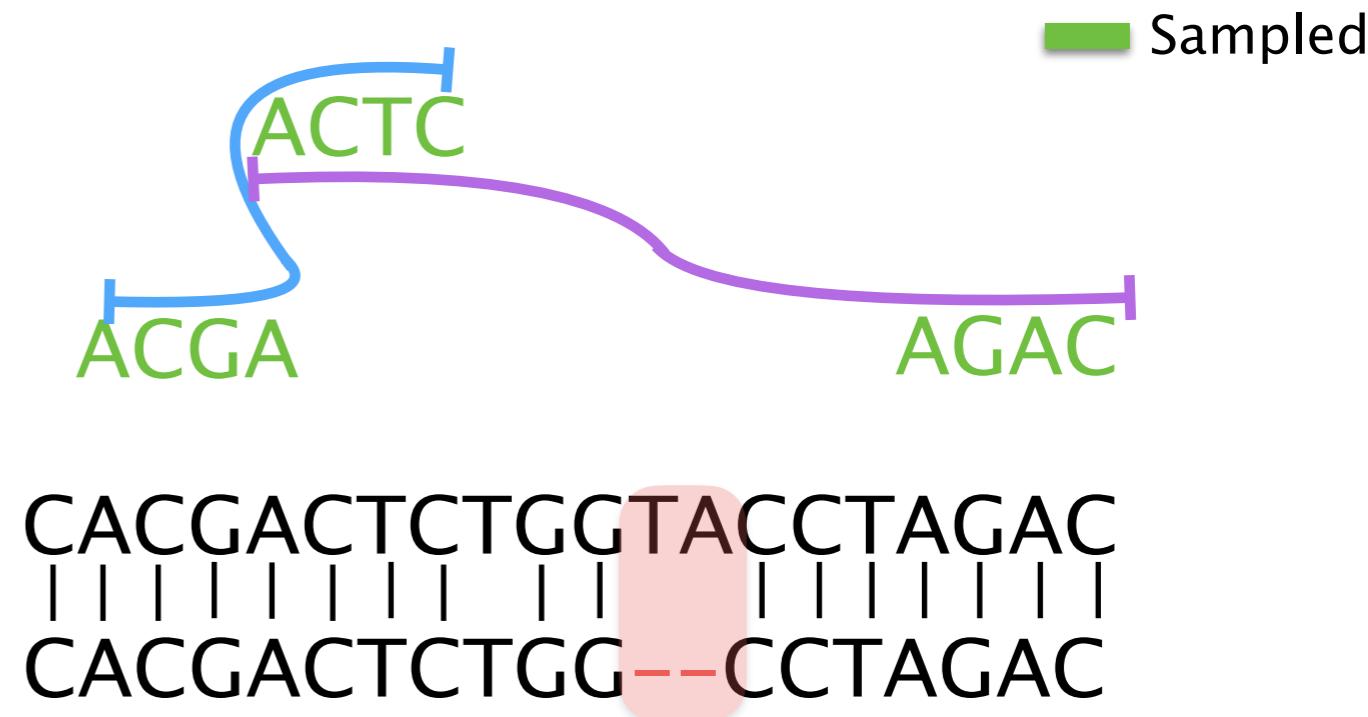
<sup>10</sup> Ekim, Berger, Chikhi, 2021

<sup>11</sup> Sahlin, 2021

# Approximate-matching (fuzzy) seeds

- Spaced k-mers<sup>1</sup>
- Vector seeds<sup>2</sup>
- Permutation techniques<sup>3,4</sup>
- Covering template families<sup>5</sup>
- SimHash<sup>3</sup> (BLEND<sup>6</sup>)
- TensorSketch<sup>7</sup>
- SubSeqHash<sup>8</sup>
- **Neighbouring minimizers<sup>9</sup>**
- k-min-mers<sup>10</sup>
- Strobemers<sup>11</sup>

- + Match over subs. and indels
- “k-mer redundancy”



<sup>1</sup> Ma et al., 2002

<sup>2</sup> Brejová et al., 2003

<sup>3</sup> Charikar, 2002

<sup>4</sup> Lederman, 2013

<sup>5</sup> Giladi et al., 2010

<sup>6</sup> Firtina et al., 2021

<sup>7</sup> Joudaki et al., 2020

<sup>8</sup> Li et al., 2023

<sup>9</sup> Chin and Khalak, 2019

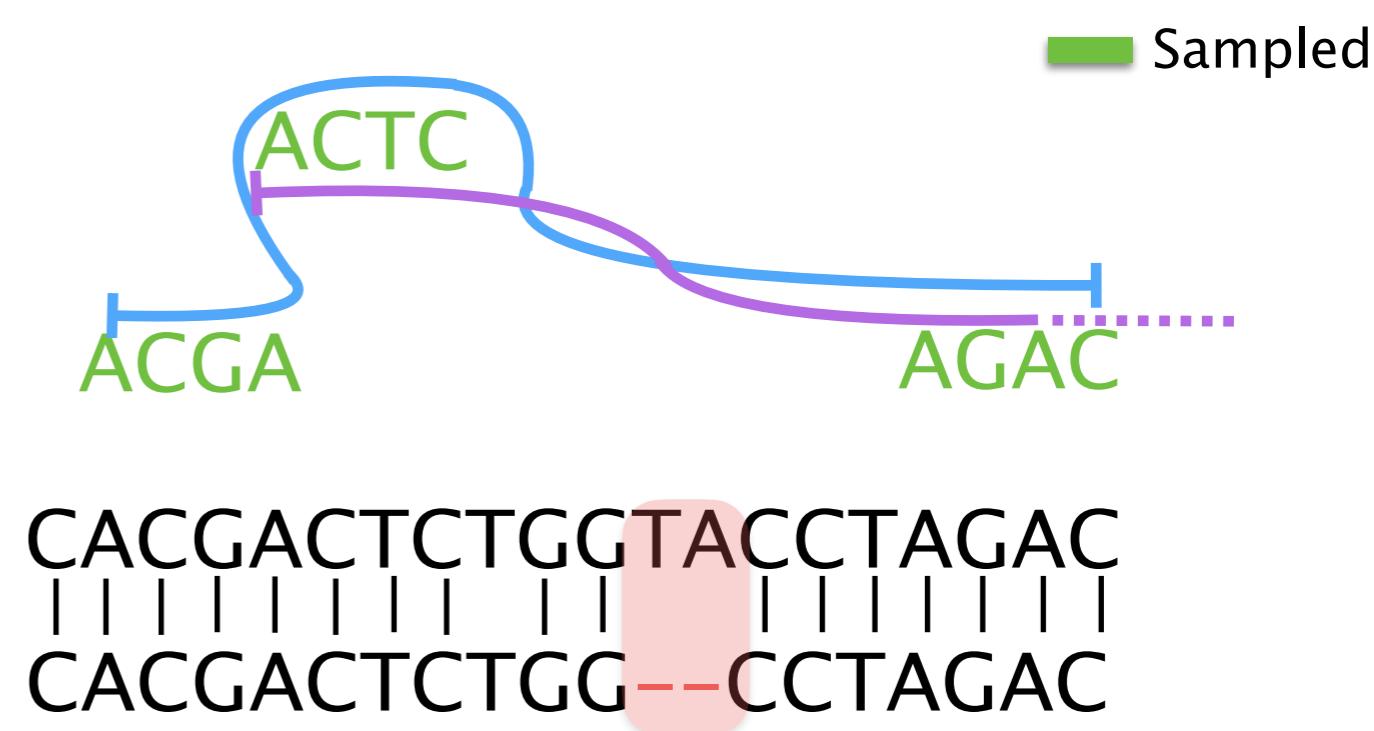
<sup>10</sup> Ekim, Berger, Chikhi, 2021

<sup>11</sup> Sahlin, 2021

# Approximate-matching (fuzzy) seeds

- Spaced k-mers<sup>1</sup>
- Vector seeds<sup>2</sup>
- Permutation techniques<sup>3,4</sup>
- Covering template families<sup>5</sup>
- SimHash<sup>3</sup> (BLEND<sup>6</sup>)
- TensorSketch<sup>7</sup>
- SubSeqHash<sup>8</sup>
- Neighbouring minimizers<sup>9</sup>
- **k-min-mers**<sup>10</sup>
- Strobemers<sup>11</sup>

- + Match over subs. and indels
- “k-mer redundancy”



<sup>1</sup> Ma et al., 2002

<sup>2</sup> Brejová et al., 2003

<sup>3</sup> Charikar, 2002

<sup>4</sup> Lederman, 2013

<sup>5</sup> Giladi et al., 2010

<sup>6</sup> Firtina et al., 2021

<sup>7</sup> Joudaki et al., 2020

<sup>8</sup> Li et al., 2023

<sup>9</sup> Chin and Khalak, 2019

<sup>10</sup> Ekim, Berger, Chikhi, 2021

<sup>11</sup> Sahlin, 2021

# Strobemers

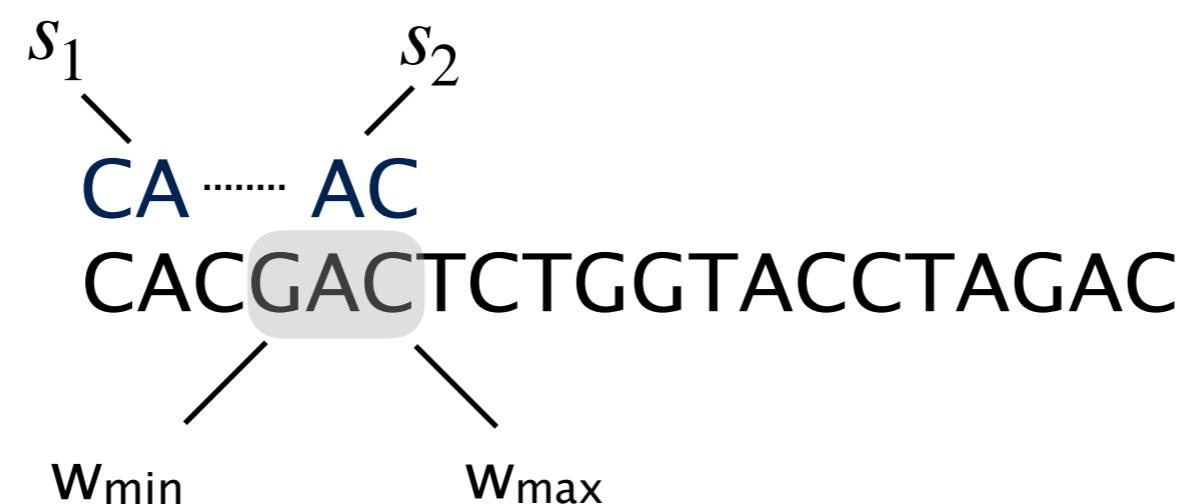
- Strobemers:  $n$  k-mers  $s_1, s_2, \dots, s_n$  (strobos) of length  $\ell$ 
  - $s_1$  is fixed
  - $s_2, \dots, s_n$  chosen from downstream window

# Strobemers

- Strobemers:  $n$  k-mers  $s_1, s_2, \dots, s_n$  (strobos) of length  $\ell$ 
  - $s_1$  is fixed
  - $s_2, \dots, s_n$  chosen from downstream window  $[w_{\min}, w_{\max}]$

## Example

( $n=2, \ell=2, w_{\min}=3, w_{\max}=5$ )

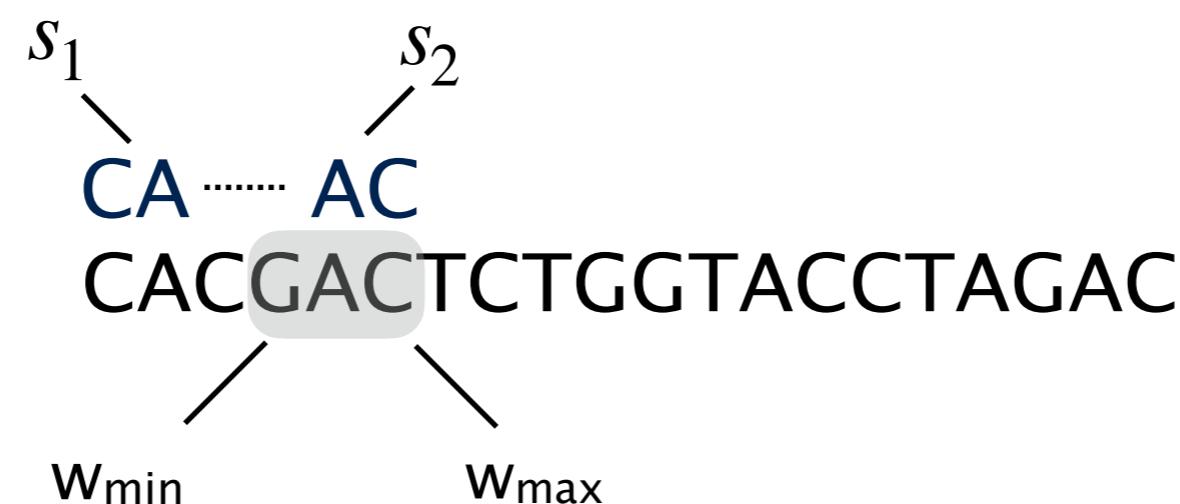


# Strobemers

- Strobemers:  $n$  k-mers  $s_1, s_2, \dots, s_n$  (strokes) of length  $\ell$ 
  - $s_1$  is fixed
  - $s_2, \dots, s_n$  chosen from downstream window  $[w_{\min}, w_{\max}]$
  - Methods: minstrokes, randstrokes, hybridstrokes

## Example

( $n=2, \ell=2, w_{\min}=3, w_{\max}=5$ )



# Randstrobes

- Pick strobe  $s_2$  given  $s_1$  with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

← Integer  
Hash function

## Example

( $n=2$ ,  $\ell=2$ ,  $w_{\min}=3$ ,  $w_{\max}=5$ )

$s_1$   
CA  
CACGACTCTGGTACCTAGAC

# Randstrokes

- Pick strobe  $s_2$  given  $s_1$  with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

## Example

( $n=2$ ,  $\ell=2$ ,  $w_{\min}=3$ ,  $w_{\max}=5$ )

$$\begin{aligned} h(AA) &= 0 \\ h(AC) &= 1 \end{aligned}$$

$$\dots$$
  
$$h(TT) = 15$$

$$q=5$$

CA  
CACGACTGTGGTACCTAGAC

$$h(CA) = 4$$

# Randstrokes

- Pick strobe  $s_2$  given  $s_1$  with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

## Example

( $n=2$ ,  $\ell=2$ ,  $w_{\min}=3$ ,  $w_{\max}=5$ )

$$\begin{aligned} h(AA) &= 0 \\ h(AC) &= 1 \end{aligned}$$

...

$$h(TT) = 15$$

$$q=5$$

CA  
CACGACTGTGGTACCTAGAC

$$h(CA) = 4$$

$$\begin{aligned} h(GA) &= 8 \\ h(AC) &= 1 \\ h(CT) &= 7 \end{aligned}$$

# Randstrokes

- Pick strobe  $s_2$  given  $s_1$  with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

## Example

( $n=2$ ,  $\ell=2$ ,  $w_{\min}=3$ ,  $w_{\max}=5$ )

$$\begin{aligned}h(AA) &= 0 \\h(AC) &= 1\end{aligned}$$

$$\dots \\h(TT) = 15$$

$$q=5$$

CA  
CACGACTGTGGTACCTAGAC

$$h(CA) = 4$$

$$\begin{aligned}h(GA) &= 8 & (4 + 8) \% 5 &= 2 \\h(AC) &= 1 & (4 + 1) \% 5 &= 0 \\h(CT) &= 7 & (4 + 7) \% 5 &= 1\end{aligned}$$

# Randstrokes

- Pick strobe  $s_2$  given  $s_1$  with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

## Example

( $n=2$ ,  $\ell=2$ ,  $w_{\min}=3$ ,  $w_{\max}=5$ )

$$\begin{aligned} h(AA) &= 0 \\ h(AC) &= 1 \end{aligned}$$

$$\dots$$

$$h(TT) = 15$$

$$q=5$$

CA.....AC  
 CACGACTGTGGTACCTAGAC

$$h(CA) = 4$$

$$h(GA) = 8 \quad (4 + 8) \% 5 = 2$$

$$h(AC) = 1 \quad (4 + 1) \% 5 = 0$$

$$h(CT) = 7 \quad (4 + 7) \% 5 = 1$$

# Randstrokes

- Pick strobe  $s_2$  given  $s_1$  with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

## Example

( $n=2$ ,  $\ell=2$ ,  $w_{\min}=3$ ,  $w_{\max}=5$ )

$$\begin{aligned} h(AA) &= 0 \\ h(AC) &= 1 \end{aligned}$$

$$\dots$$
  
$$h(TT) = 15$$

$$q=5$$

AC  
CA ..... AC  
CACGACTGTGGTACCTAGAC

$$h(AC) = 1$$

$$\begin{aligned} h(AC) &= 1 & (1 + 1) \% 5 &= 2 \\ h(CT) &= 7 & (1 + 7) \% 5 &= 3 \\ h(TG) &= 14 & (1 + 14) \% 5 &= 0 \end{aligned}$$

# Randstrokes

- Pick strobe  $s_2$  given  $s_1$  with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

## Example

( $n=2$ ,  $\ell=2$ ,  $w_{\min}=3$ ,  $w_{\max}=5$ )

$$\begin{aligned} h(AA) &= 0 \\ h(AC) &= 1 \end{aligned}$$

$$\dots$$
  
$$h(TT) = 15$$

$$q=5$$

AC ..... TG  
CA ..... AC  
CACGACTGTGGTACCTAGAC

$$h(AC) = 1$$

$$\begin{aligned} h(AC) &= 1 & (1 + 1) \% 5 &= 2 \\ h(CT) &= 7 & (1 + 7) \% 5 &= 3 \\ h(TG) &= 14 & (1 + 14) \% 5 &= 0 \end{aligned}$$

# Randstrokes

- Pick strobe  $s_2$  given  $s_1$  with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

## Example

( $n=2$ ,  $\ell=2$ ,  $w_{\min}=3$ ,  $w_{\max}=5$ )

CG .. CT  
AC ..... TG  
CA ..... AC  
CACGACTGTGGTACCTAGAC

# Randstrokes

- Pick strobe  $s_2$  given  $s_1$  with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

## Example

( $n=2$ ,  $\ell=2$ ,  $w_{\min}=3$ ,  $w_{\max}=5$ )

CG .. CT  
AC ..... TG ...  
CA ..... AC      GG ..... CT  
CAC GACTGTGGT ACCTAGAC  
||| ||| ||| ||| ||| |||  
CAC -GCTGAGGT -CCTAGAC

# Randstrokes

- Pick strobe  $s_3$  given  $s_1$  and  $s_2$  with equation:

$$s_3 = \arg \min_{s' \in W} h(s_1, s_2) + h(s') \mod q$$

## Example

( $n=2$ ,  $\ell=2$ ,  $w_{\min}=3$ ,  $w_{\max}=5$ )

CA ..... AC ..... GG  
CACGACTGTGGTACCTAGAC

# Randstrokes

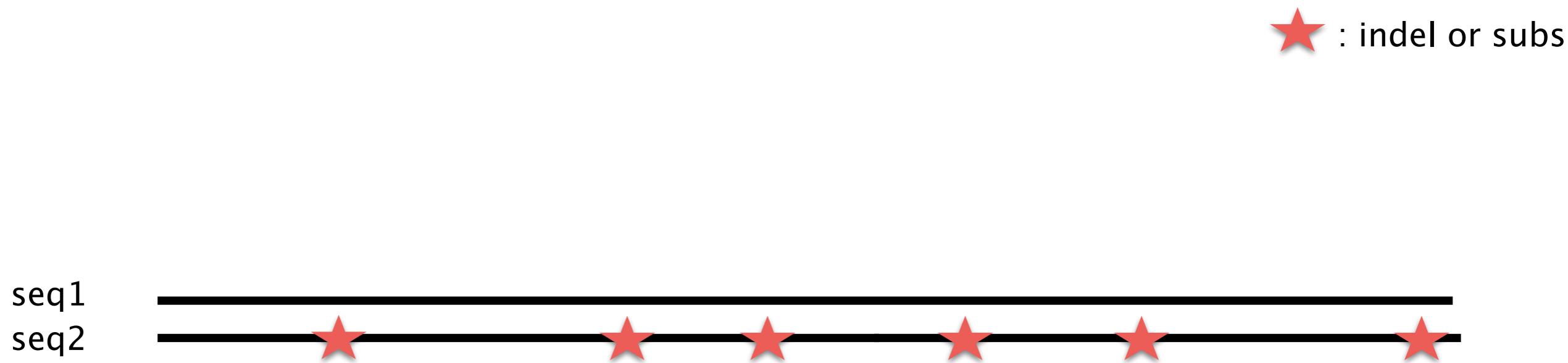
- ✓ Match over substitutions and indels
- ✓ Low ‘redundancy’ between seeds

## Example

( $n=2$ ,  $\ell=2$ ,  $w_{min}=3$ ,  $w_{max}=5$ )

CA ..... AC ..... GG  
CACGACTGTGGTACCTAGAC

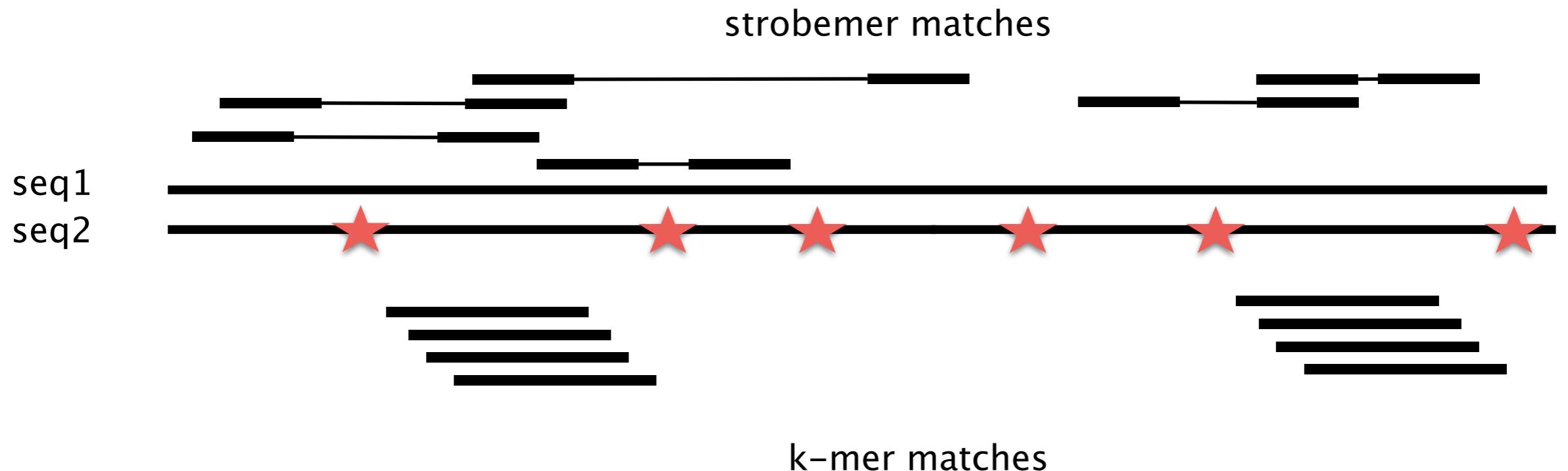
# Randstrokes are sensitive



# Randstrokes are sensitive

- Number of matches ( $m$ )
  - Sequence coverage (sc)
  - Match coverage (mc)
  - Match gap E-size (E)

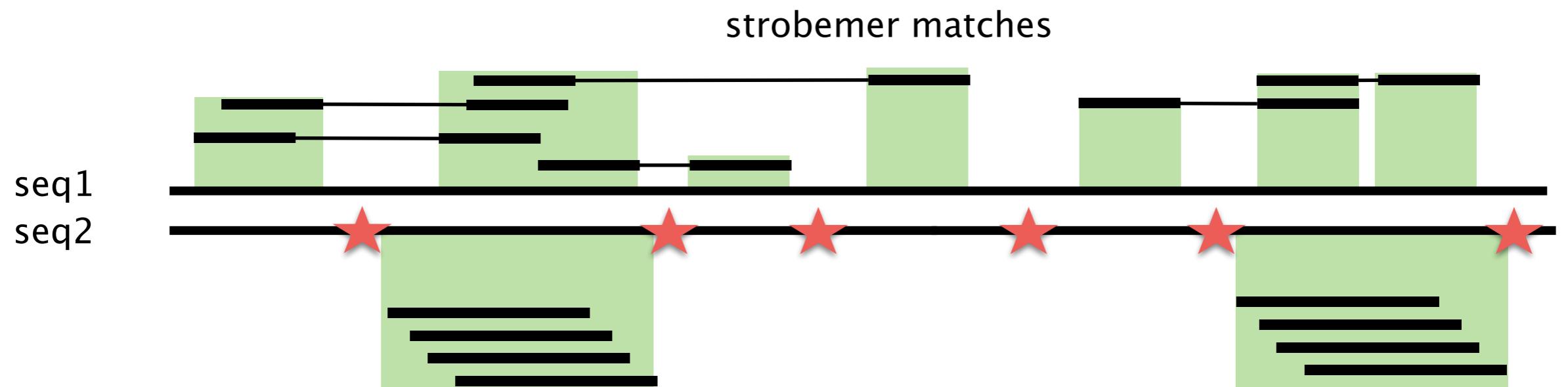
★ : indel or subs



# Randstrokes are sensitive

- Number of matches (m)
- Sequence coverage (sc)
- Match coverage (mc)
- Match gap E-size (E)

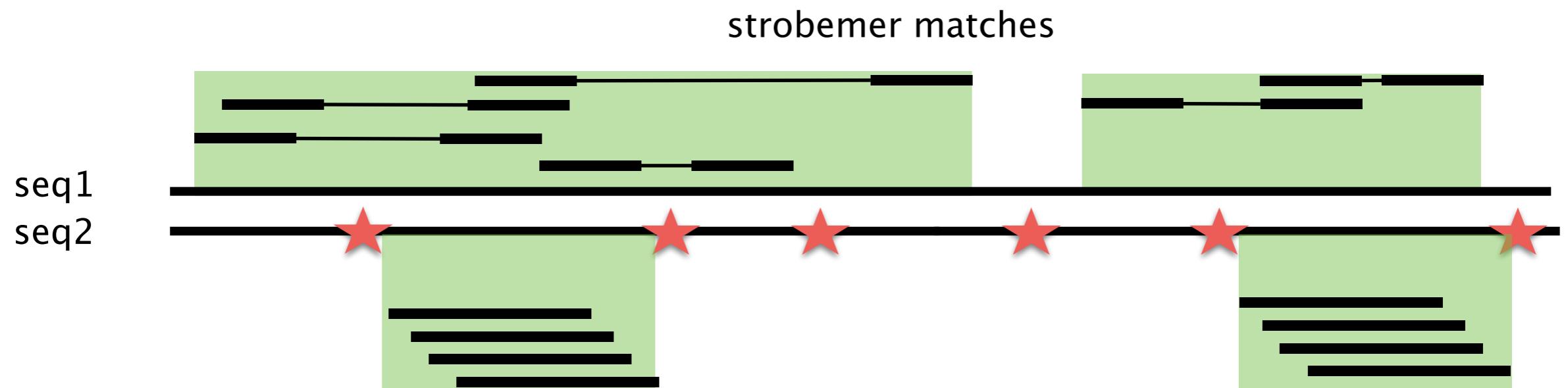
★ : indel or subs



# Randstrokes are sensitive

- Number of matches (m)
- Sequence coverage (sc)
- Match coverage (mc)
- Match gap E-size (E)

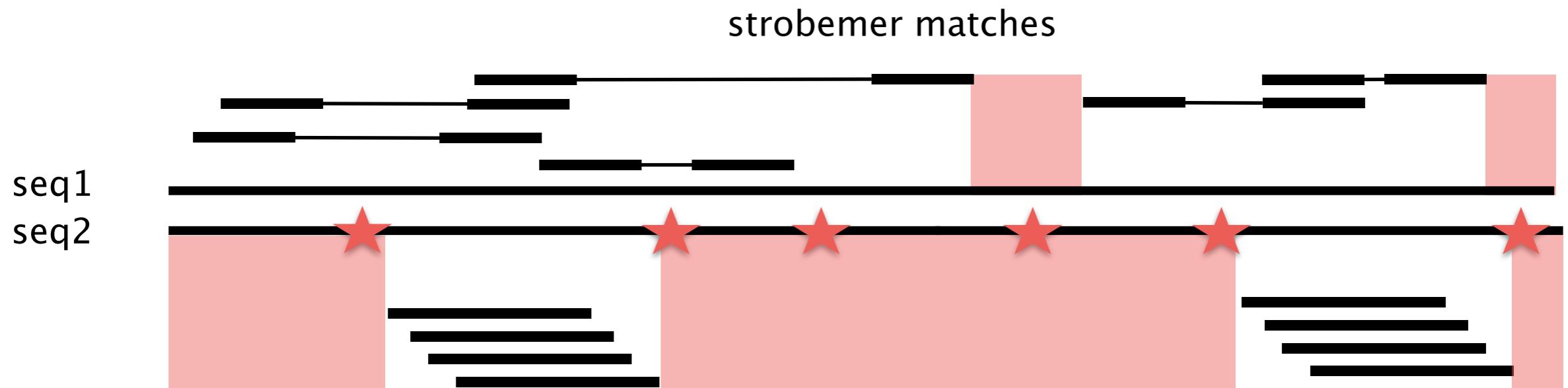
★ : indel or subs



# Randstrokes are sensitive

- Number of matches (m)
- Sequence coverage (sc)
- Match coverage (mc)
- Match gap E-size (E)

★ : indel or subs



# Randstrokes are specific

## Metrics

- Number of distinct hash values
- Fraction of hash values occurring once

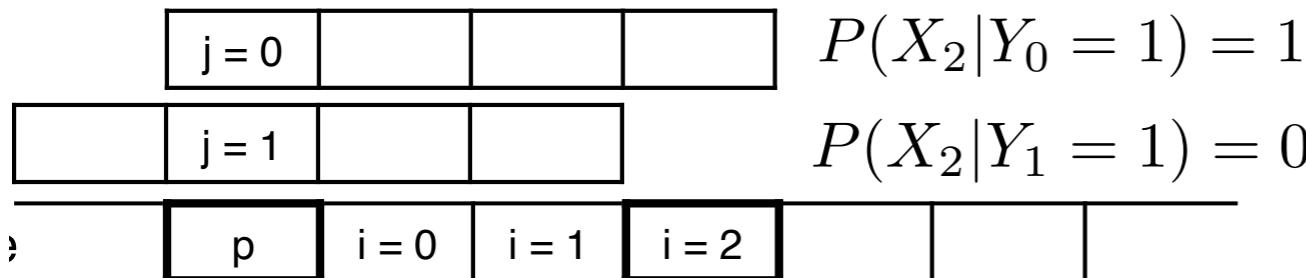
## Result

- No large difference between k-mers and strobemers
- (Assuming corresponding parameters, e.g.,  $k = 2\ell$ )

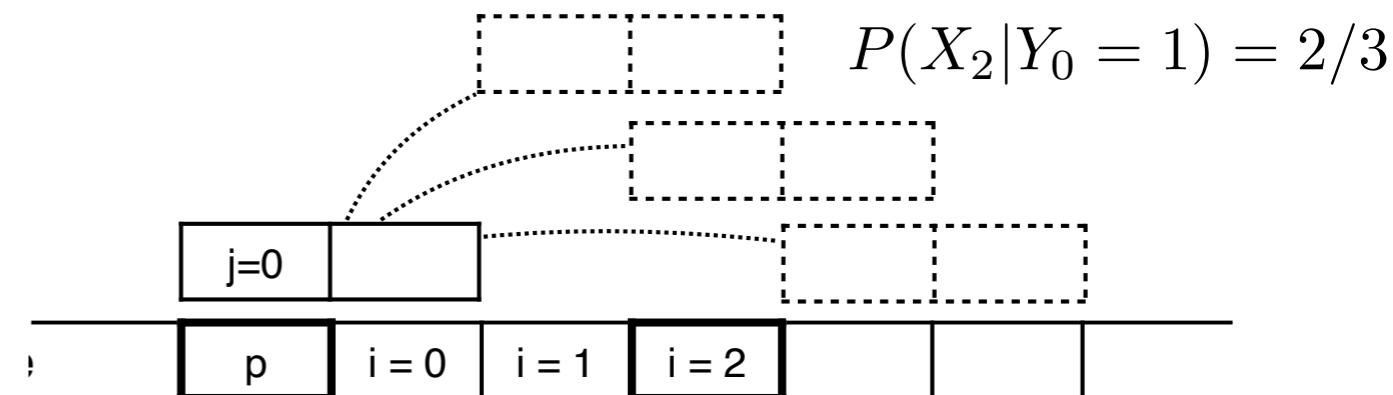
# Why are strobemers sensitive?

- Understanding matching sensitivity from a theoretical perspective
- Conditional entropy of seed sampling a position  $i$  given that it covers  $p$

**k-mers (4)**



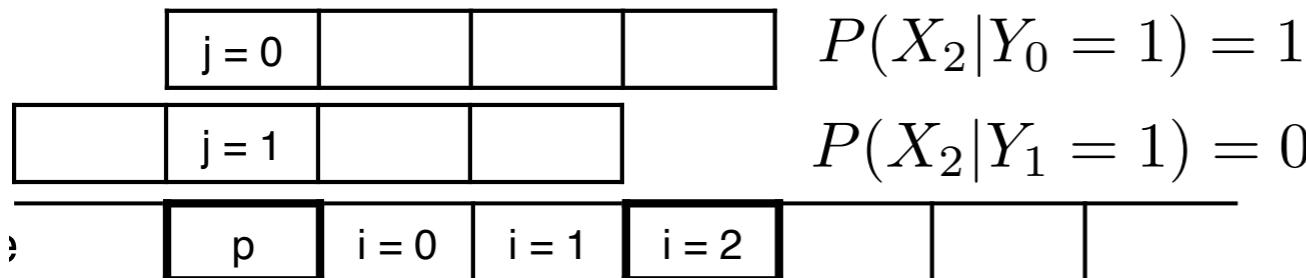
**randstrokes (2,2,2,4)**



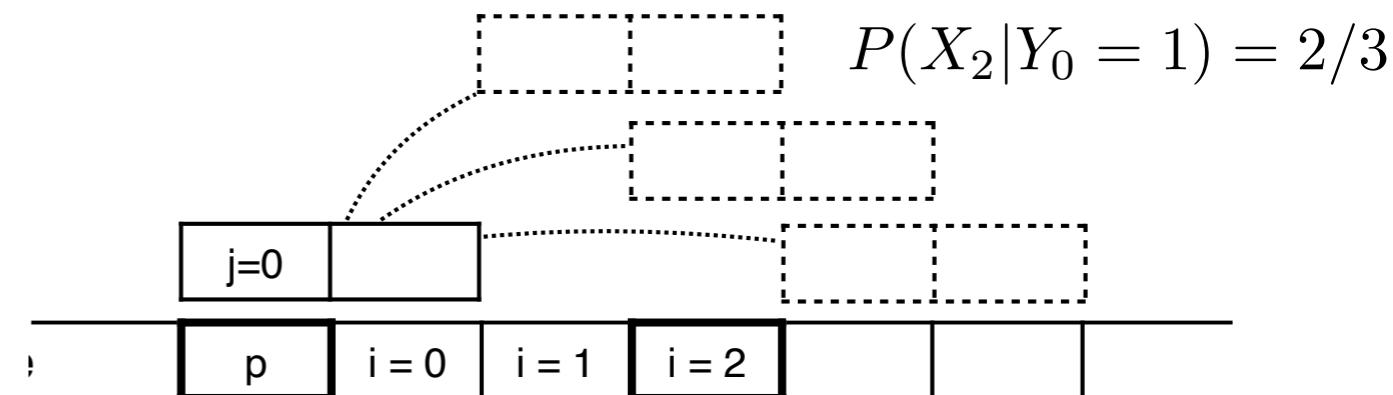
# Why are strobemers sensitive?

- Understanding matching sensitivity from a theoretical perspective
- Conditional entropy of seed sampling a position  $i$  given that it covers  $p$

**k-mers (4)**



**randstrokes (2,2,2,4)**



$$H(X|Y) = - \sum_{j=0}^{k-1} \sum_{i=0}^{w-2} P(Y_j) P(X_i|Y_j) \log_2 P(X_i|Y_j)$$

# Can we design seeds with higher entropy?

- Altstrobes: Altering the size of first and second strobe ( $s_1 + s_2 = k$ )
- Mixedstrobes: Mixing k-mers and randstrobes
- Multistrobes: Selecting strobe size from multiple lengths (still with  $s_1 + s_2 = k$ )

**Randstrobes**



**Altstrobes**



or



**Mixedstrobes**



or



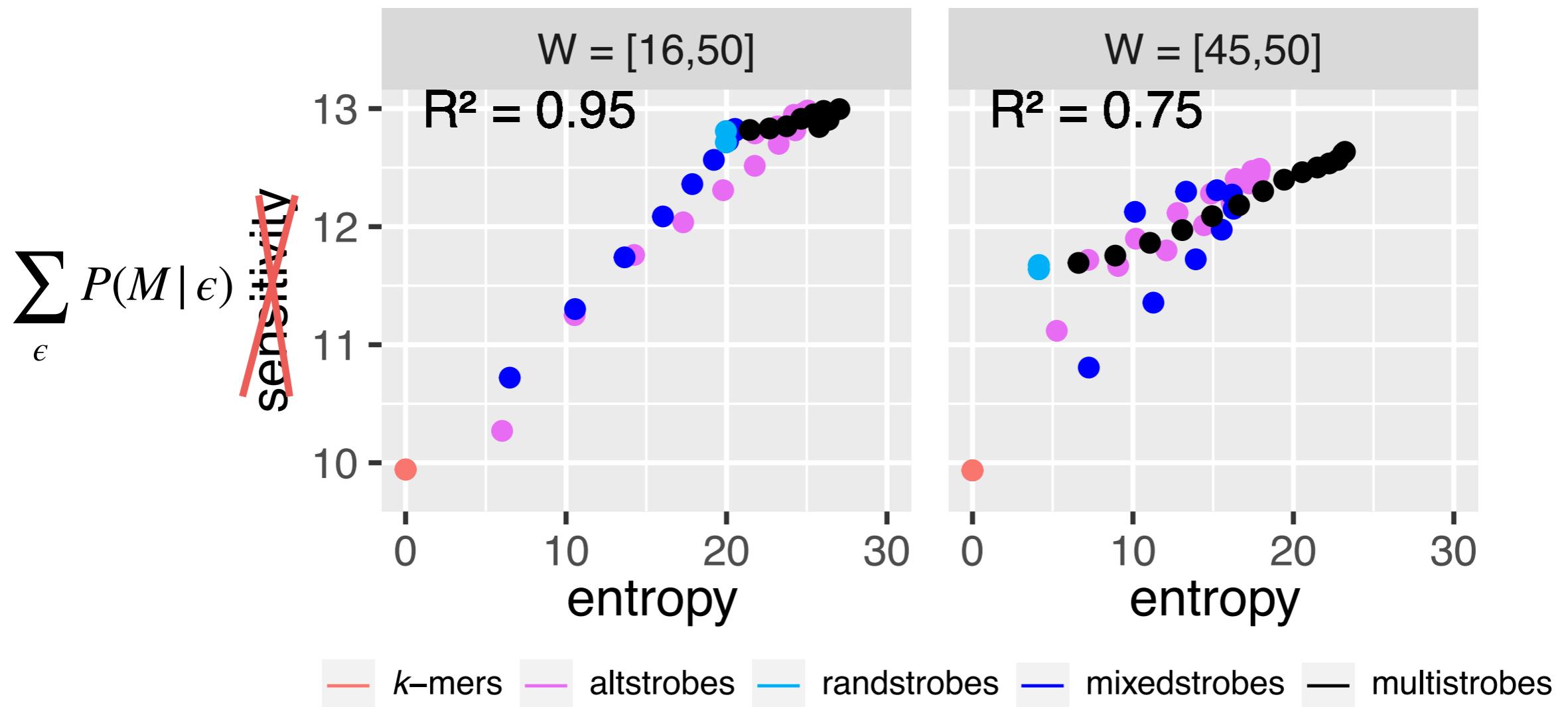
**Multistrobes**



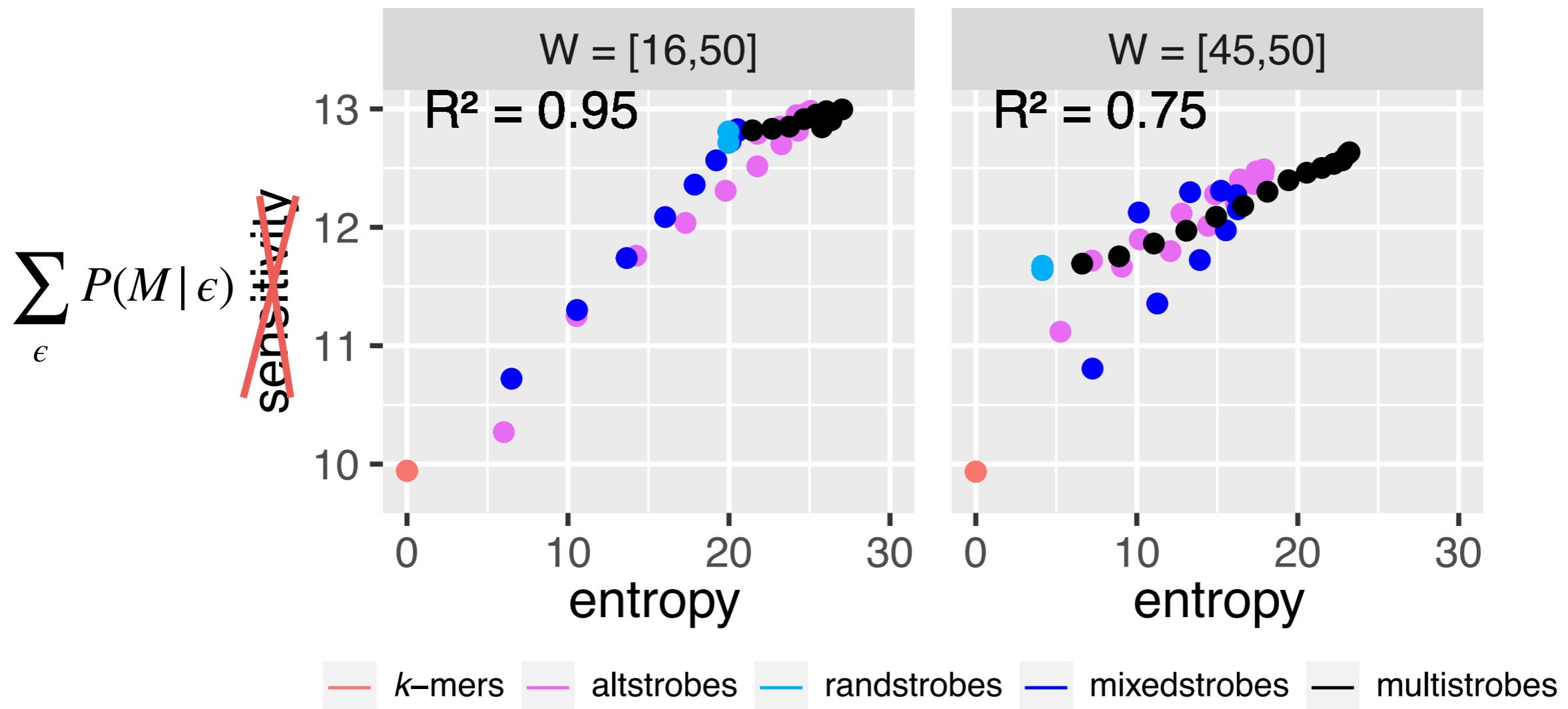
...



# Higher entropy leads to higher sensitivity



# Higher entropy leads to higher sensitivity



- Result1: Our entropy model predicts seed sensitivity
- Result2: Some seeds produce higher sensitivity than randstrobes

# Runtime: Probably not a bottleneck

		Human Chr1
k-mers	30bp	12,4
randstrokes	(2,15,16,40)	19,1
randstrokes	(2,15,16,100)	19,4
randstrokes	(3,10,11,40)	25,1
randstrokes	(3,10,11,100)	27,7

- Result:
  - Number of strobes affects runtime, window size not so much
- Benchmark (in C++):
  - Bitpacking k-mers/strobes
  - **Timings based on Bitwise AND ( $h(s_1) + h(s_2)$ ) & q**

# Can we optimise the implementation?

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

# Can we optimise the implementation?

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \quad \text{and } q$$

Bitwise AND with bitmask (Wei Shen)

# Can we optimise the implementation?

$$s_2 = \arg \min_{s' \in W} h(s_1) \cancel{+} h(s') \text{ m}\cancel{+} d q$$

Bitwise XOR (Lidong Guo ,Giulio Pibiri)

# Can we optimise the implementation?

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \bmod q$$

## Variables

- $h$  (e.g., no hash, xxHash, wyhash, Thomas Wang hash – mm2).
- “Link function” – the method to select  $s_2$
- Sampling comparator (e.g., argmin or argmax)
- Final seed hash value  $f(s_1, s_2)$

# Can we optimise the implementation?

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \bmod q$$

## Variables

- $h$  (e.g., no hash, xxHash, wyhash, Thomas Wang hash – mm2).
- “Link function” – the method to select  $s_2$
- Sampling comparator (e.g., argmin or argmax)
- Final seed hash value  $f(s_1, s_2)$

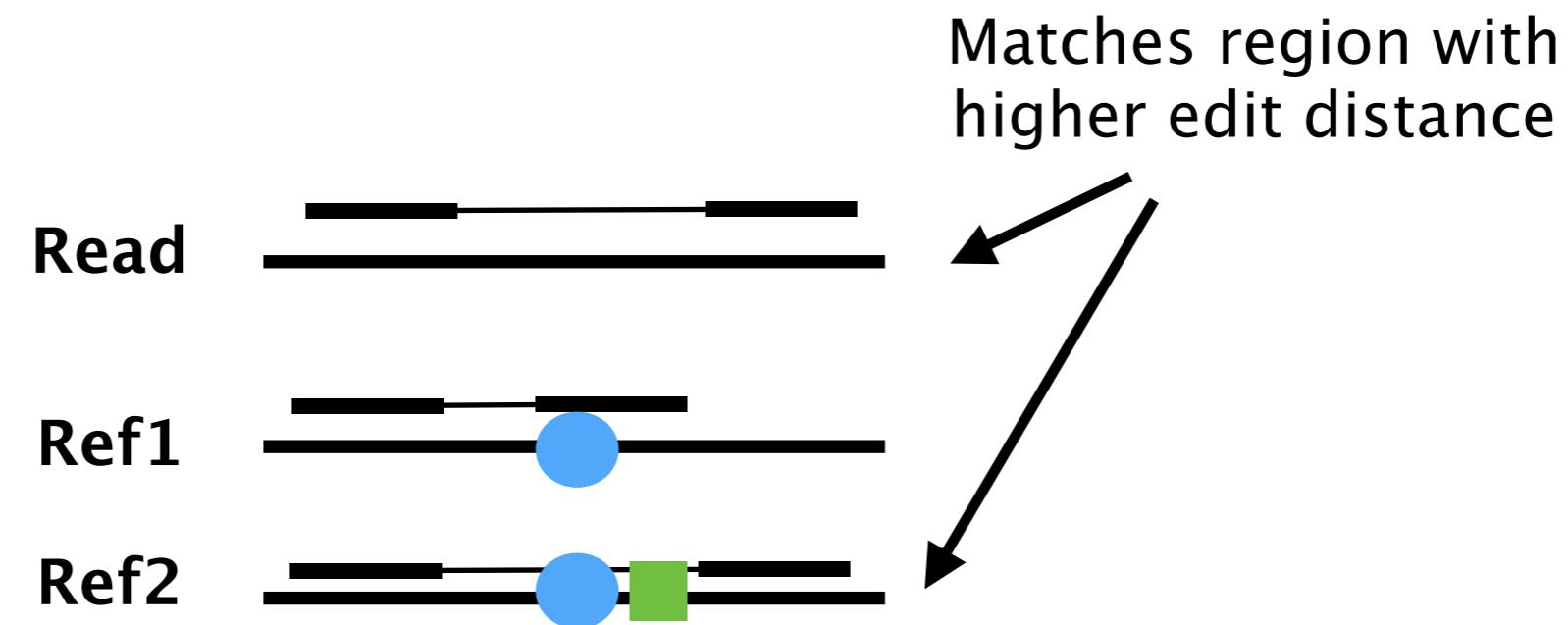
## Objectives

- Fast
- Maximize entropy of sampled strobe in window

## Results

- Link method influences both time and randomness

# Limitation: Approximate seeds do not guarantee matching the most similar region 😞



## Implications?

- May not be reliable for applications that only count matches
- Using positions (for chaining/clustering) can alleviate the issue

# Strobemers are useful in practice

## Other groups

- RNAblloom<sup>1</sup>: Reference-free long-read transcriptome assembly
- BLEND<sup>2</sup>: Long-read overlap detection
- Symbiont-screener<sup>3</sup>: Reference-free long read phasing based on “markers”
- SAKE<sup>4</sup>: Identify all k-mers in a genome from long reads

## Our work

- strobealign<sup>5</sup>: Short-read alignment.

<sup>1</sup>Nip et al, Nat Commun, 2023

<sup>2</sup>Firtina et al., NAR genom. bioinform., 2023

<sup>3</sup>Xu et al., Front. mar. sci., 2023.

<sup>4</sup>Leinonen and Salmela, Plos One, 2023.

<sup>5</sup>Sahlin, Genome Biology, 2022.

# Short-read mapping with strobealign

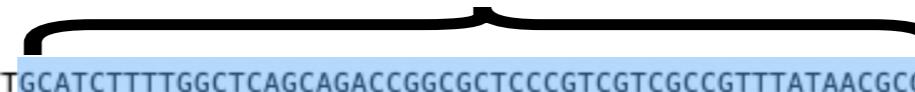
- strobealign<sup>1</sup> combines syncmers<sup>2</sup> and randstrokes<sup>3</sup>
- Main idea: Can afford to use longer seeds

```
GGTCGATTGGACTCTGCCGGAGTTGCATCTTGGCTCAGCAGACCGGCCTCCGTCGCCGTTATAACCGACCCAGGGGGGGGAGTTCCGGATGCCATCCACGATGCTCGTCTGGGGTGGAGAGC...
 TTGGACTCTGCCGGAGTTGC
  GCCGGAGTTGCATCTTGG
   GAGTTGCATCTTGGCTCA
    GCATCTTGGCTCAGCAGA
     TTTGGCTCAGCAGACCGGC
      TCAGCAGACCGGCCTCCG
       AGACCGGCCTCCGTC
        CTCCCGTCGTGCCGTTAT
         GTCGTGCCGTTATAACGC
          CGCCGTTATAACCGACCC
           ATAACCGACCCAGGGGGGG
            ACCCAGGGGGGGGAGTTCC
             GGGGGGGAGTTCCGGATGC
              CGAGTTCCGGATGCCATCCA
               CGGGATGCCATCCACGATGC
```

# Short-read mapping with strobealign

- strobealign<sup>1</sup> combines syncmers<sup>2</sup> and randstrokes<sup>3</sup>
- Main idea: Can afford to use longer seeds

Strobe from window of syncmers instead of k-mers



GGTCGATTGGACTCTGCCGGAGTTGCATCTTTGGCTCAGCAGACCGGGCGTCCCGTCGCCGTTATAACCGCGACCAGGGGGGGGAGTTCCGGATGCCATCCACGATGCTCGTCTGGGGTGGAGAGC...  
TTGGACTCTGCCGGAGTTGC---TTTGGCTCAGCAGACCGGGCG  
GCCGGAGTTGCATCTTTGG-----CTCCCGTCGTGCCGTTAT  
GAGTTGCATCTTTGGCTCA-----CTCCCGTCGTGCCGTTAT  
GCATCTTTGGCTCAGCAGA-----ATAACCGCGACCCAGGGGGGG  
TTTGGCTCAGCAGACCGGGCG-----GTCGTGCCGTTATAACGC  
TCAGCAGACCGGGCGCTCCCG-----CCGGATGCCATCCACGATGC  
AGACCGGGCGCTCCCGTC-----GGGGGGCGAGTTCCGGATGC  
CTCCCGTCGTGCCGTTAT-----CGAGTTCCGGATGCCATCCA  
GTCGTGCCGTTATAACGC-----GATGCCATCCACGATGCTCG  
CGCCGTTATAACCGCGACCC-----CGAGTTCCGGATGCCATCCA  
ATAACCGCGACCCAGGGGGGG-----GATGCCATCCACGATGCTCG  
ACCCAGGGGGGGCGAGTTCC-----AGC  
GGGGGGCGAGTTCCGGATGC---CCACGATGCTCGTCTGGGGGG  
CGAGTTCCGGATGCCATCCACGATGCTCGTCTGGGGTGG  
CGGGATGCCATCCACGATGC-----GGGTGGAGAGC

<sup>1</sup>Sahlin, Genome Biology, 2022.

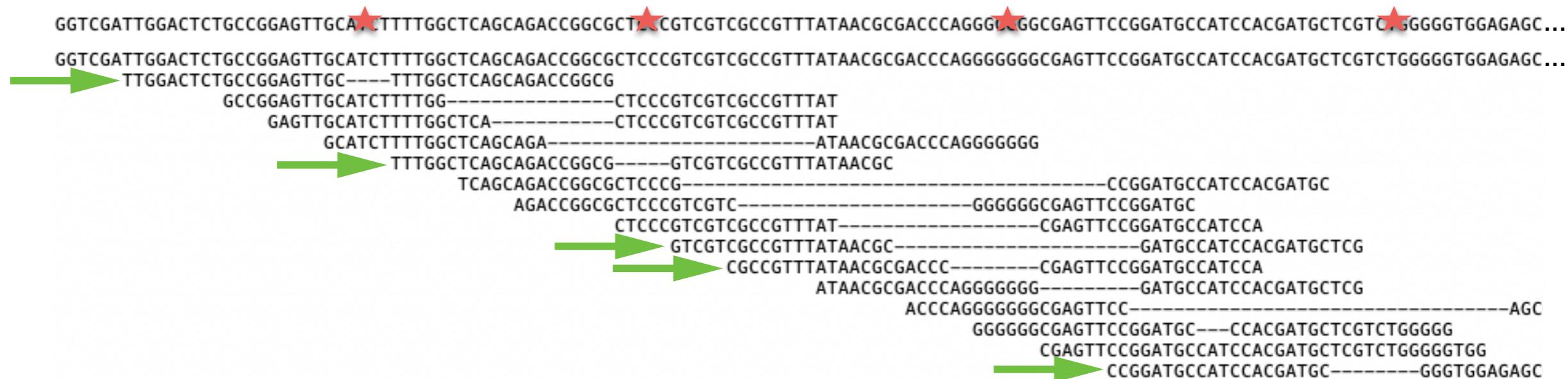
<sup>2</sup> Edgar, PeerJ, 2020

<sup>3</sup> Sahlin, Genome Research. 2021

# Short-read mapping with strobealign

- strobealign<sup>1</sup> combines syncmers<sup>2</sup> and randstrokes<sup>3</sup>
- Main idea: Can afford to use longer seeds

## Reference



GGTCGATTGGACTCTGCCGGAGTTGCA★TTTTGGCTCAGCAGACCGGGCGT★CGTCGTGCCGTTATAACCGCACCCAGGG★GGCGAGTTCCGGATGCCATCCACGATGCTCGT★GGGGTGGAGAGC...

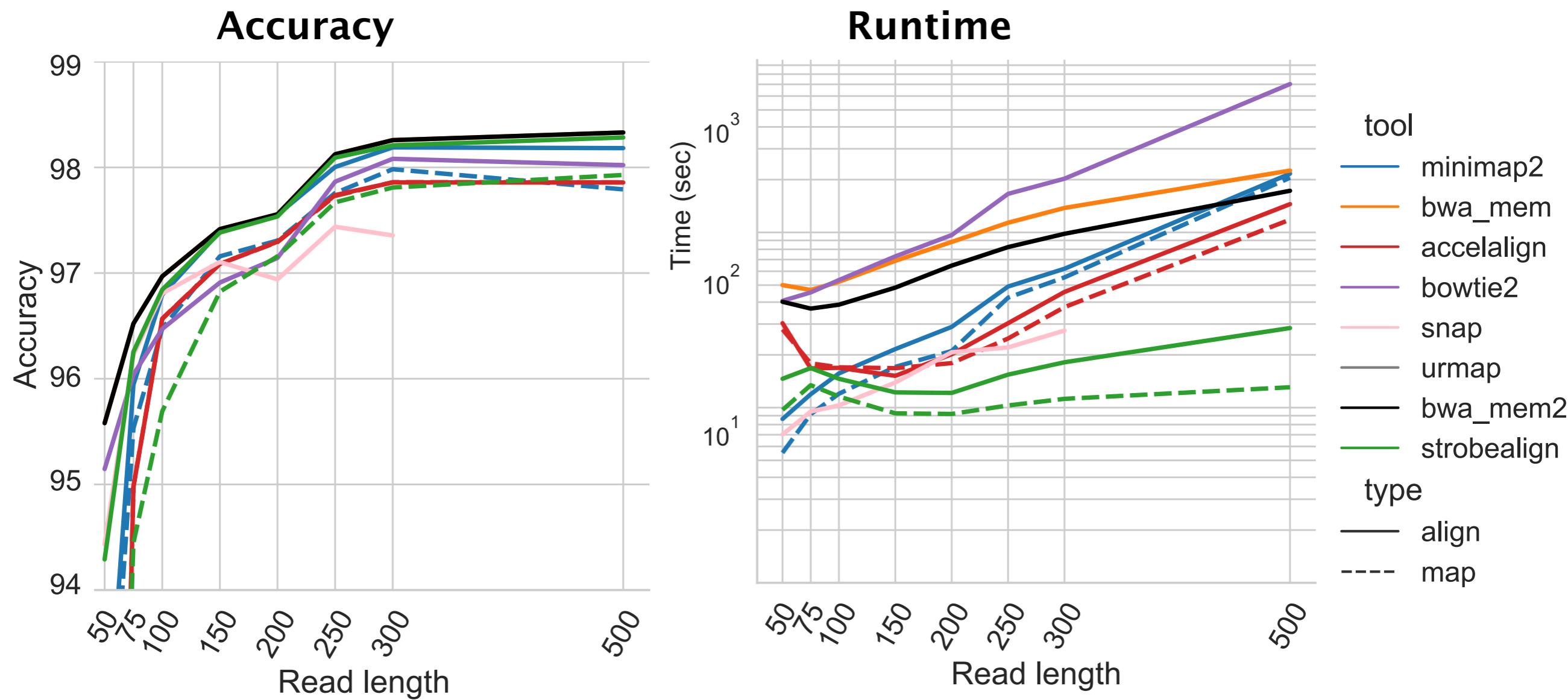
GGTCGATTGGACTCTGCCGGAGTTGCATCTTGCTCAGCAGACCGGGCGTCCCGTCGTGCCGTTATAACCGCACCCAGGGGGCGAGTTCCGGATGCCATCCACGATGCTCGTCTGGGGTGGAGAGC...

→ TTGGACTCTGCCGGAGTTGC---TTTGGCTCAGCAGACCGGGCG  
GCCGGAGTTGCATCTTG-----CTCCCCTCGTCGCCGTTTAT  
GAGTTGCATCTTGCTCA-----CTCCCCTCGTCGCCGTTTAT  
GCATCTTGCTCAGCAGA-----ATAACCGCACCCAGGGGGGG  
→ TTTGGCTCAGCAGACCGGGCG---GTCGTGCCGTTATAACGC  
TCAGCAGACCGGGCGCTCCCG-----CCGGATGCCATCCACGATGC  
AGACCGGGCGCTCCCGTC-----GGGGGGCGAGTTCCGGATGC  
CTCCCCTCGTCGCCGTTTAT-----CGAGTTCCGGATGCCATCCA  
→ GTCGTGCCGTTATAACGC-----GATGCCATCCACGATGCTCG  
CGCCGTTATAACCGCACCC-----CGAGTTCCGGATGCCATCCA  
ATAACCGCACCCAGGGGGGG-----GATGCCATCCACGATGCTCG  
ACCCAGGGGGGGCGAGTTCC-----AGC  
GGGGGGCGAGTTCCGGATGC---CCACGATGCTCGTCTGGGGGG  
CGAGTTCCGGATGCCATCCACGATGCTCGTCTGGGGTGG  
→ CGGGATGCCATCCACGATGC-----GGGTGGAGAGC

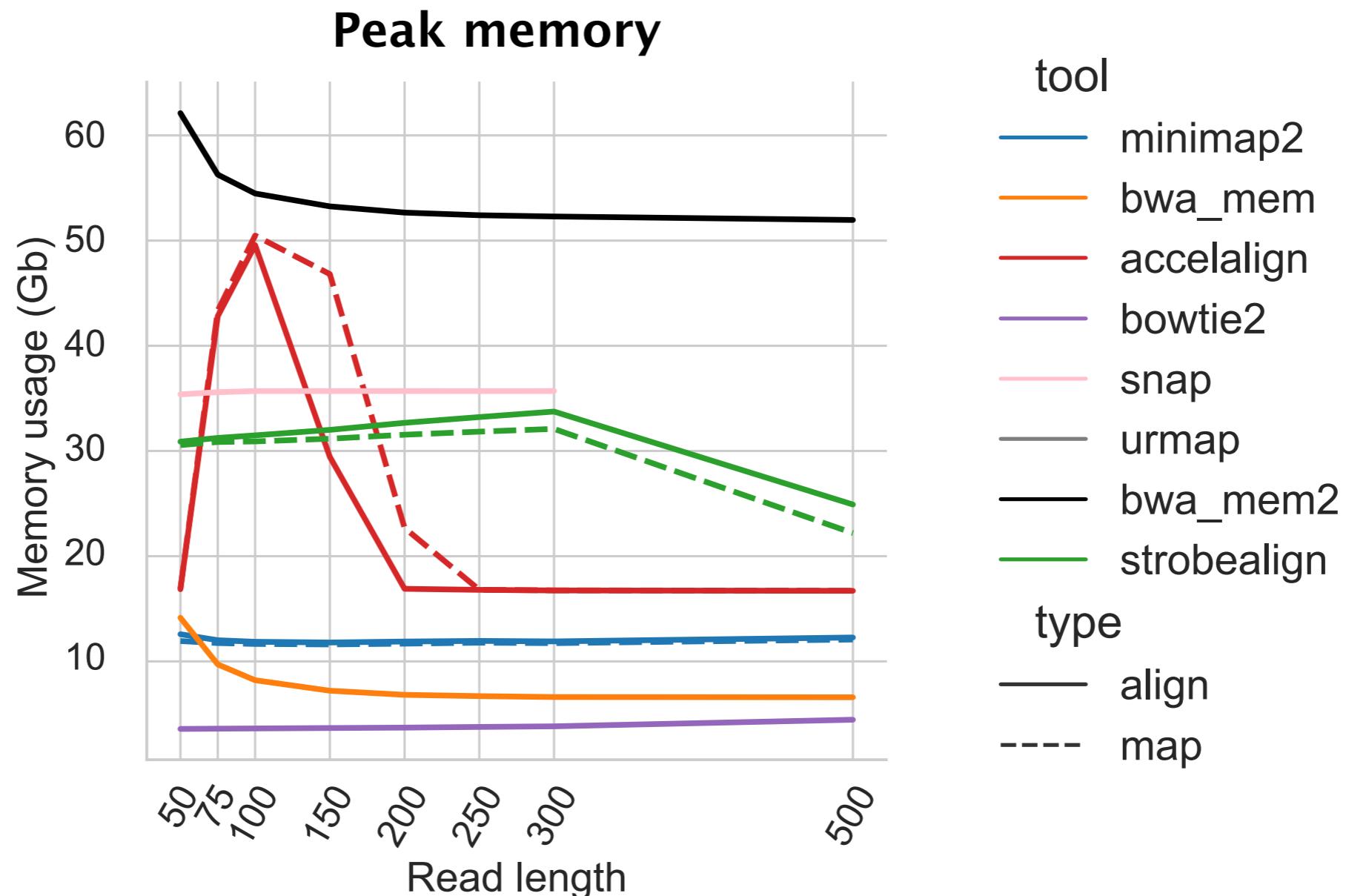
Chance to match

# Strobealign has high accuracy ( $\geq 150bp$ ) and is fast

- Paired-end reads at different variation rates to hg38
- Tools run with 16 threads



# ..but has high memory consumption!

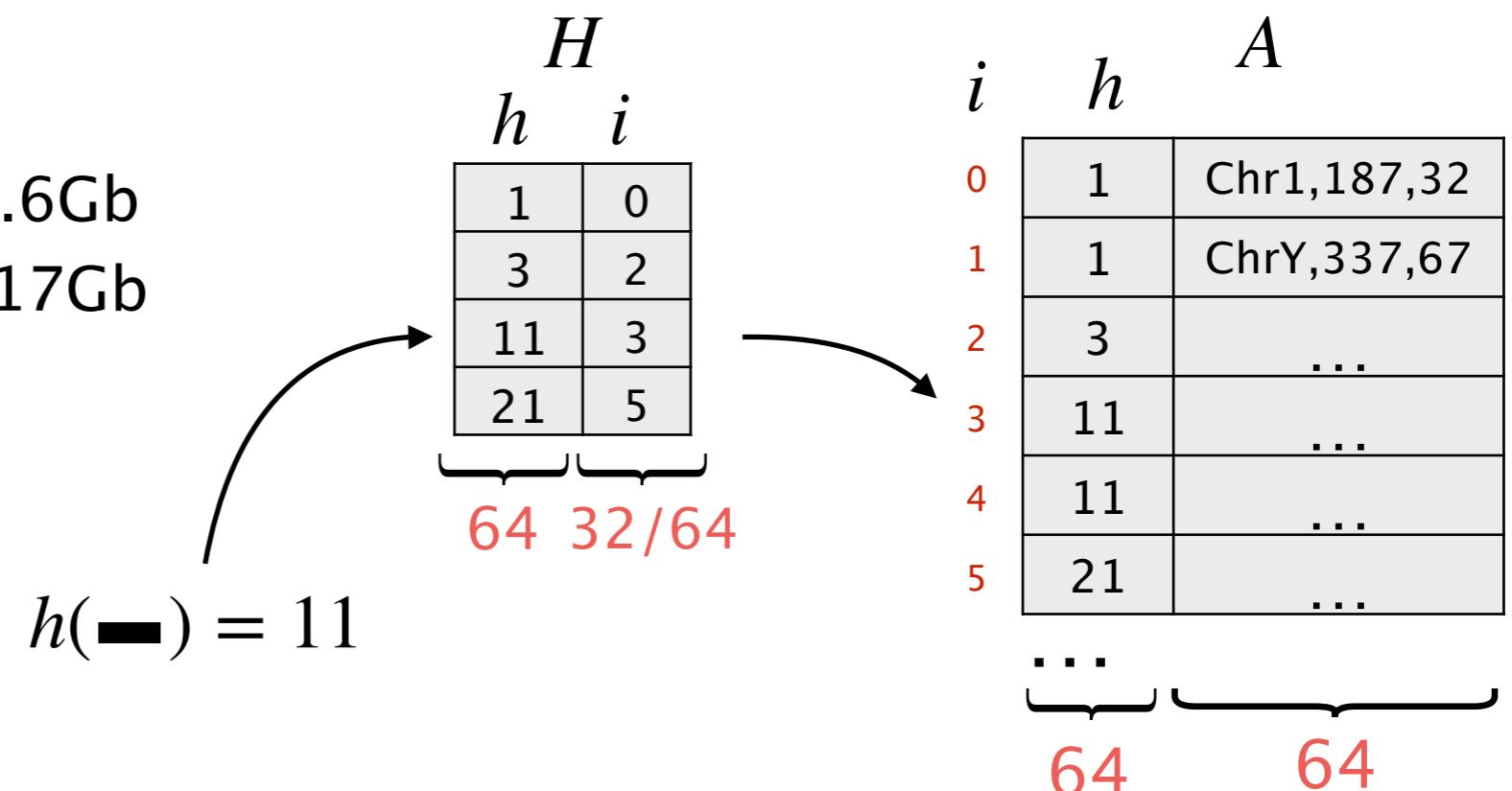


# What is consuming memory?

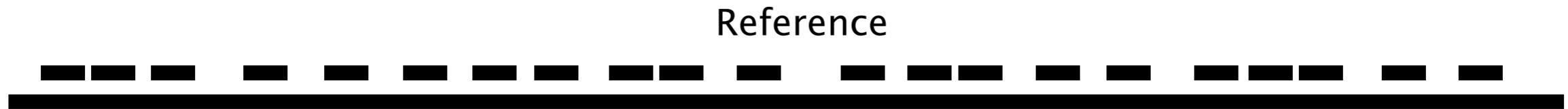


## Human

- Genome: 3Gb
- $A: 3B*(1/5)*128 \text{ bits} = 9.6\text{Gb}$
- $H = \text{load factor etc...?} \approx 17\text{Gb}$

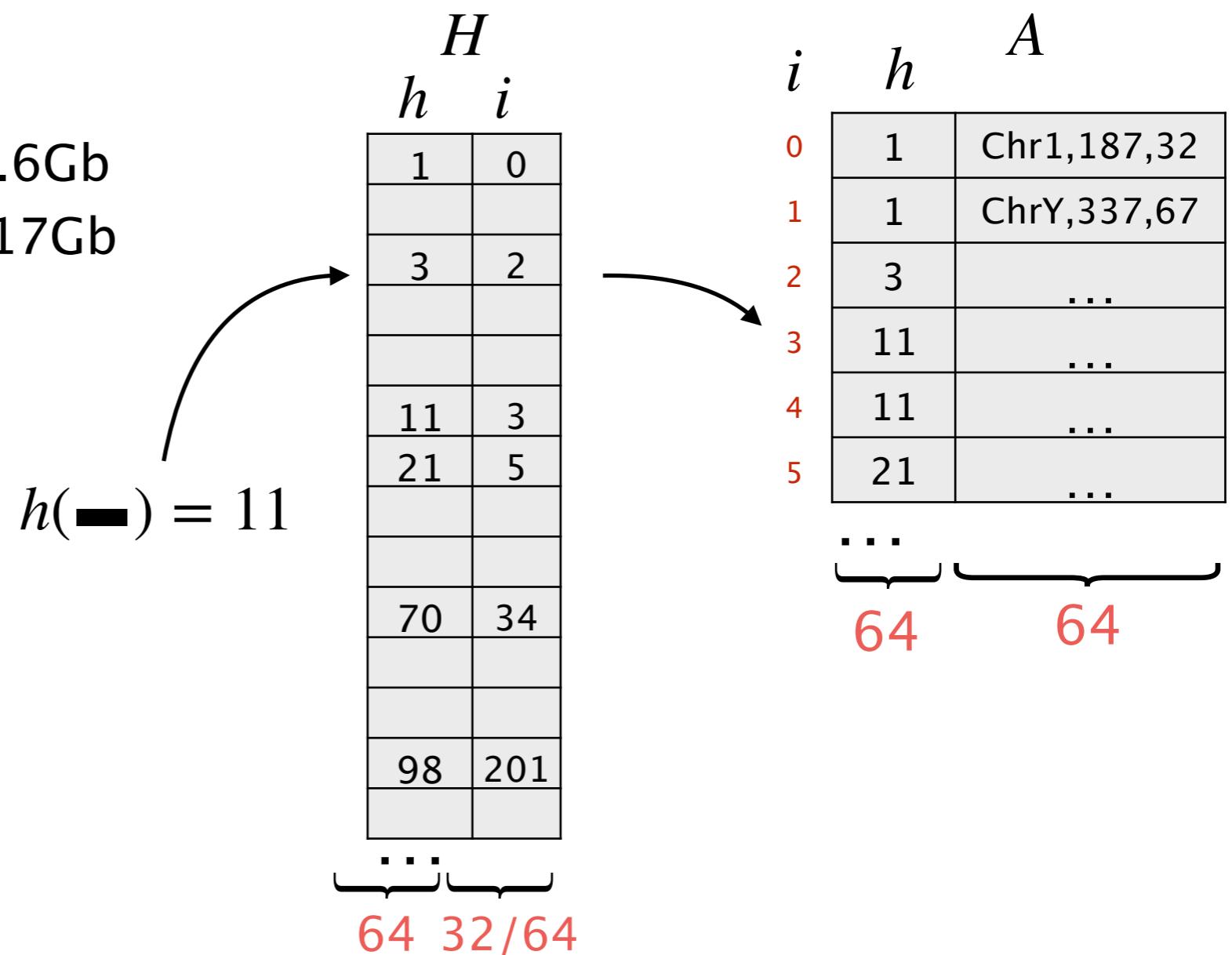


# What is consuming memory?



## Human

- Genome: 3Gb
- $A: 3B * (1/5) * 128 \text{ bits} = 9.6\text{Gb}$
- $H = \text{load factor etc...?} \approx 17\text{Gb}$

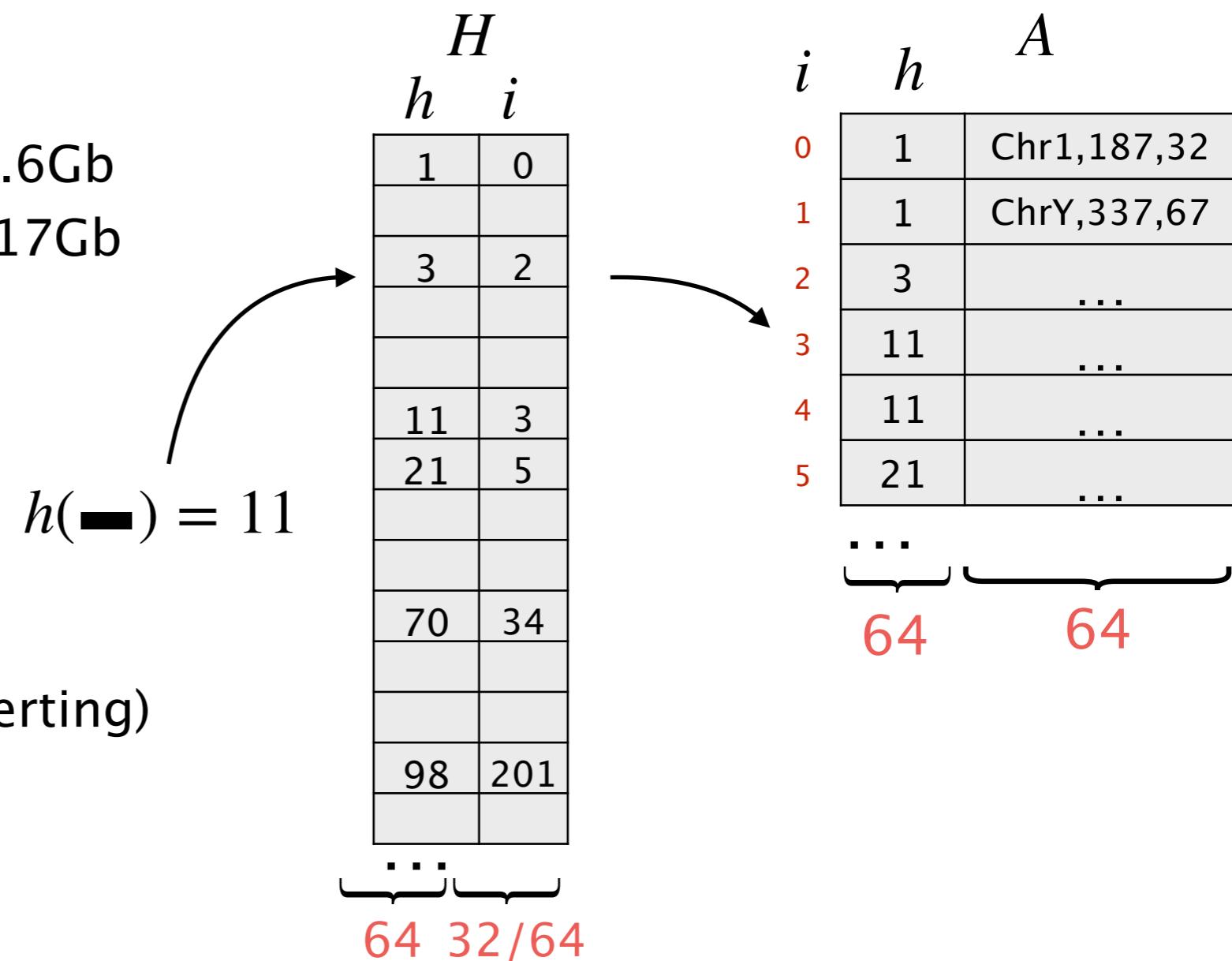


# What is consuming memory?



## Human

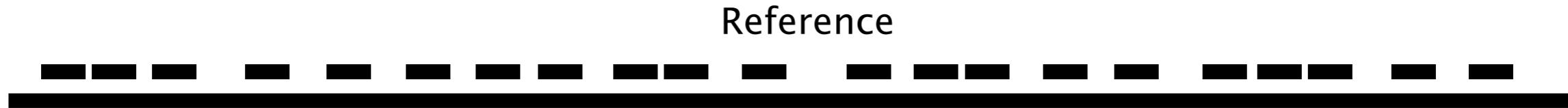
- Genome: 3Gb
- $A: 3B * (1/5) * 128 \text{ bits} = 9.6\text{Gb}$
- $H$ = load factor etc...?  $\approx 17\text{Gb}$



## Idea (Cred. Luis Coelho)

- $A$  is static (no deleting/inserting)
- Hash table not needed
- Use prefix lookup vector!
- $N = 28 \rightarrow 268\text{Mb}$

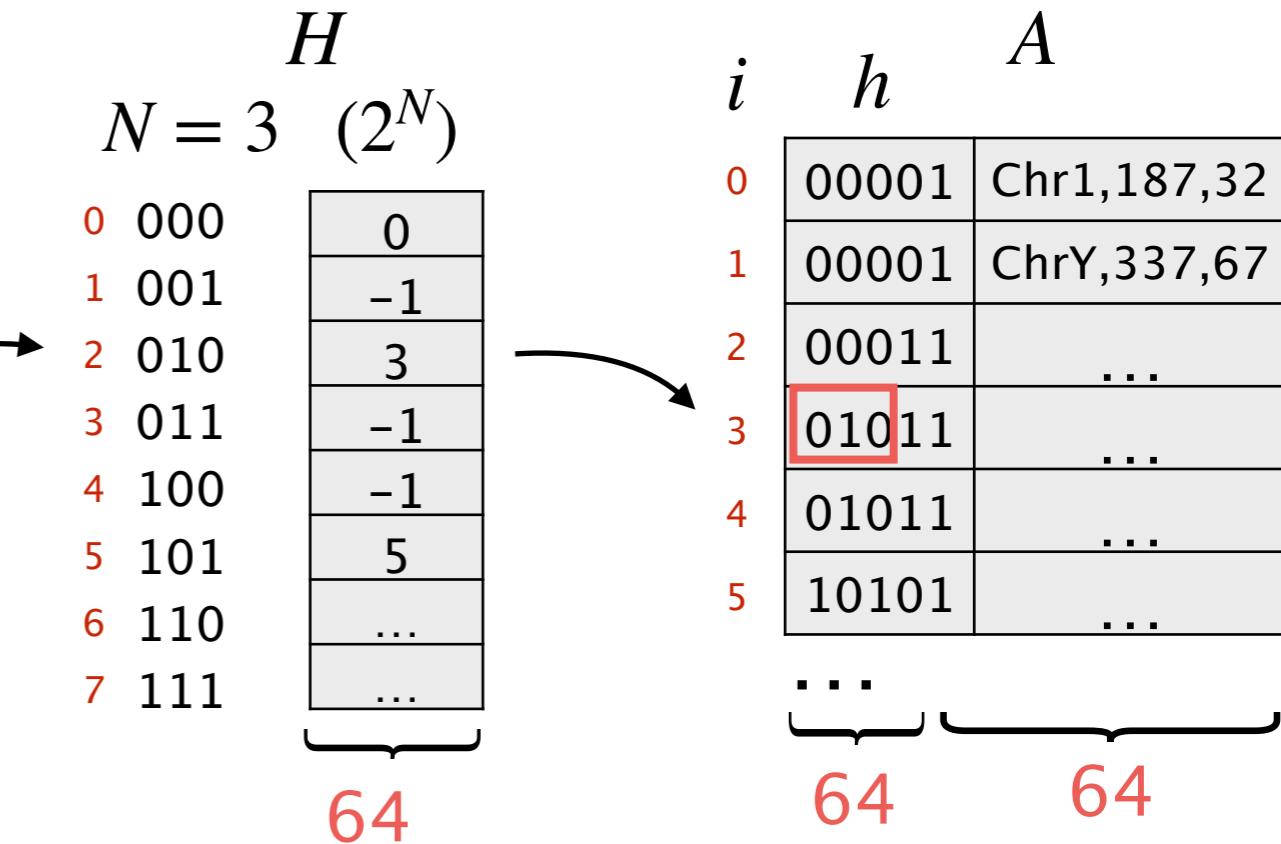
# What is consuming memory?



## Human

- Genome: 3Gb
- $A: 3B * (1/5) * 128 \text{ bits} = 9.6\text{Gb}$
- $H = \text{load factor etc...?} \approx 17\text{Gb}$

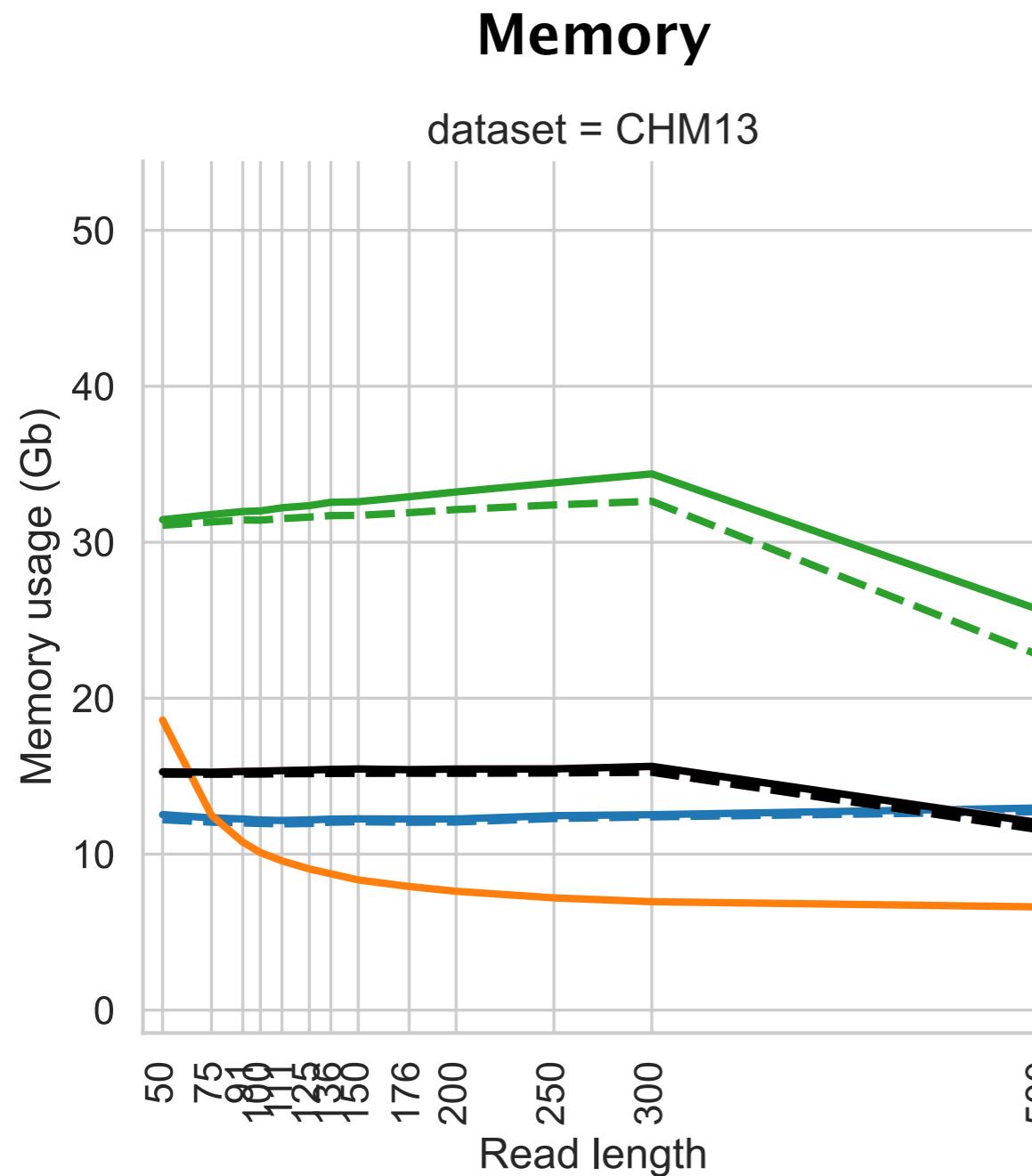
$$h(\text{---}) = 11 = \boxed{01011}$$



## Idea (Cred. Luis Coelho)

- $A$  is static (no deleting/inserting)
- Hash table not needed
- Use prefix lookup vector!
- $N = 28 \rightarrow 268\text{M slots} \rightarrow \sim 2\text{Gb}$

# A prefix-lookup table reduces index size (and lowers construction time)



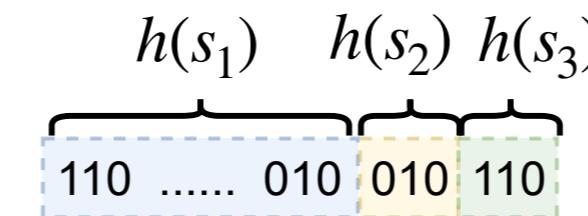
### Time

Tool (1 thread)	time
minimap2	191s
strobealign-0.7.1	259s
strobealign-0.13.0*	150–160s

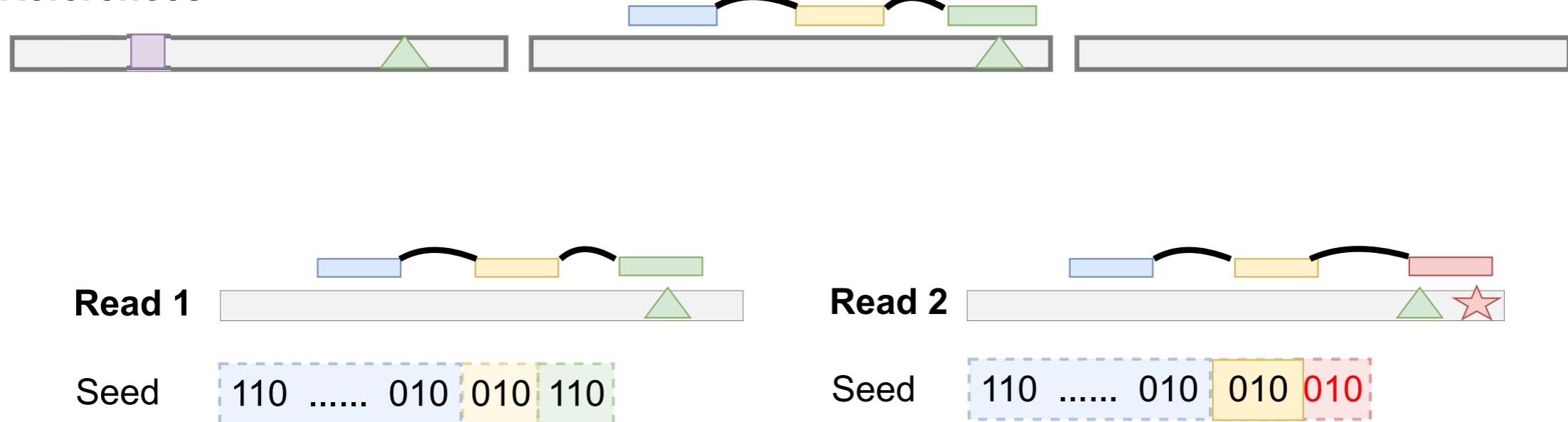
# Multi-context seeds (MCS) improves accuracy

Strobe information lost!

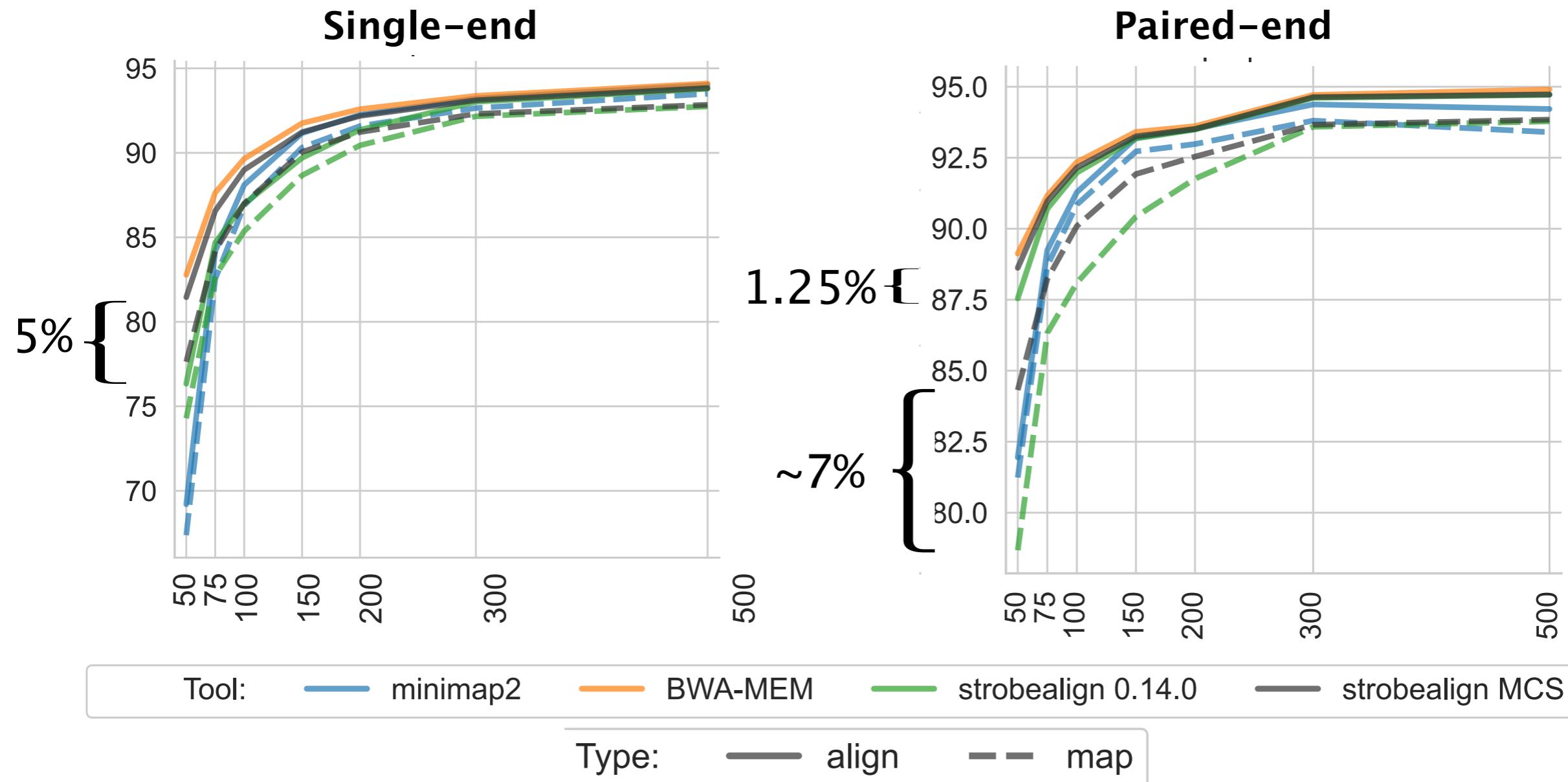
- Strobemers seed hash:  $f(s_1, s_2)$ , something like  $2h(s_1) - h(s_2)$  or  $h(s_1) + h(s_2)$
- Idea: store hashes of strobes individually



## References



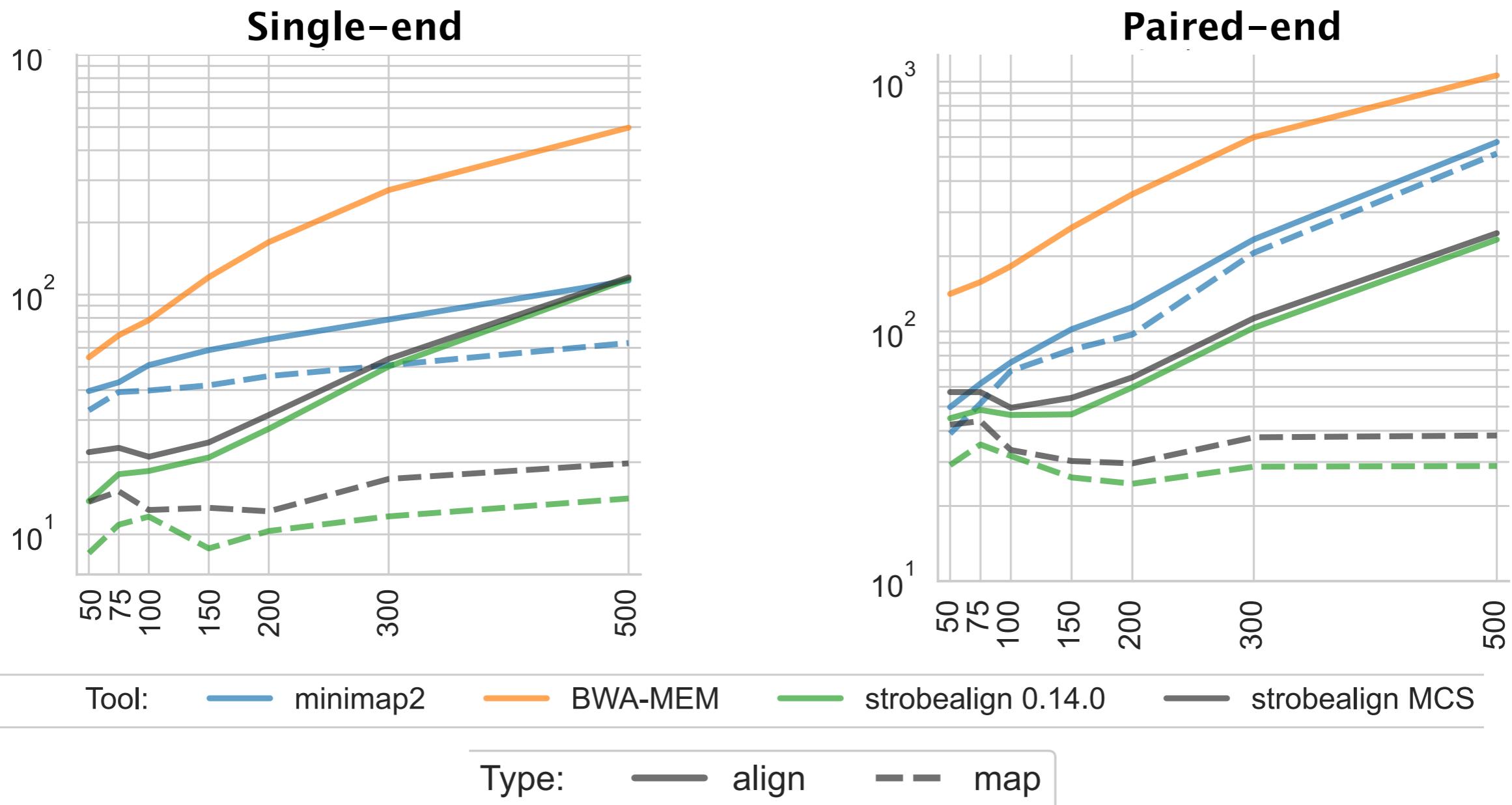
# Mapping accuracy (simulated Illumina reads to CHM13)



\*Tolstoganov et al., 2024, bioRxiv

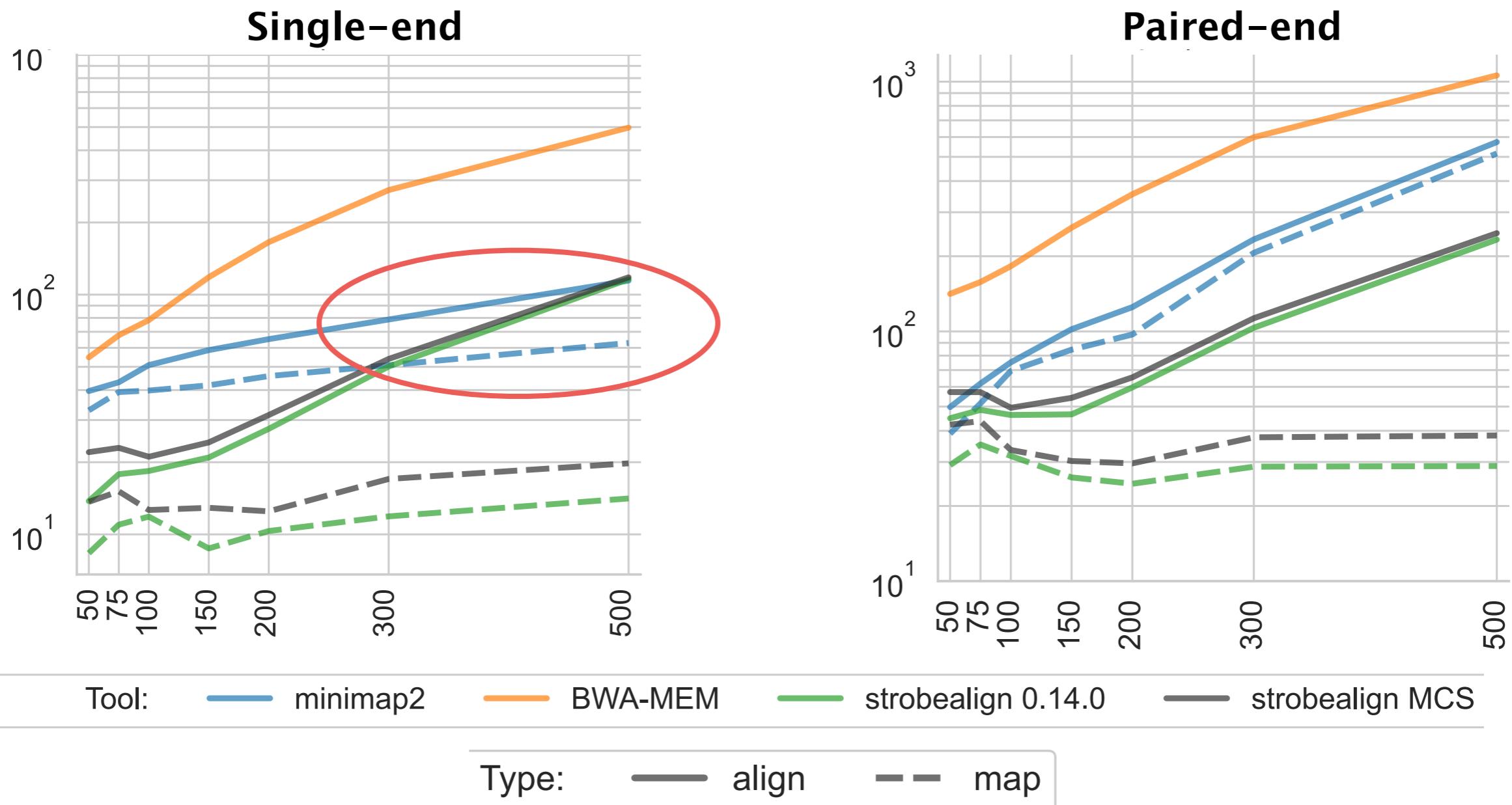
# Mapping time (simulated Illumina reads to CHM13)

- strobealign-MCS is a bit slower than v0.14.0 (more seeds)
- We believe we can optimise aemb a bit



# Mapping time (simulated Illumina reads to CHM13)

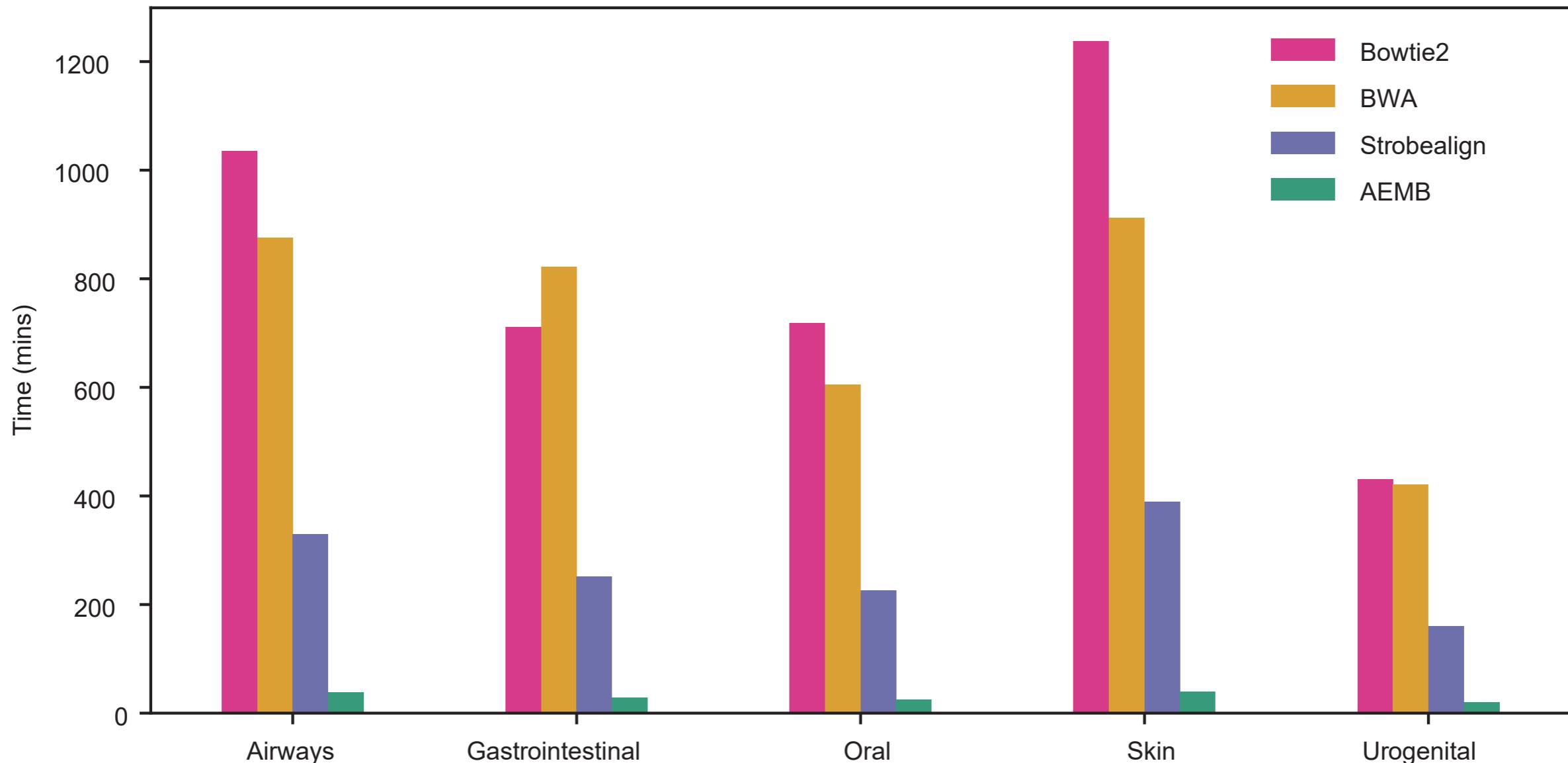
- strobealign-MCS is a bit slower than v0.14.0 (more seeds)
- We believe we can optimise aemb a bit
- Another conclusion: We need to do piece-wise extension



# strobealign new feature: --aemb

- **aemb**: “Abundance estimation metagenomic (contig) binning”
- Mapping-based coverage computation
- Multi-against-multi binning (Mattock and Watson, 2023)

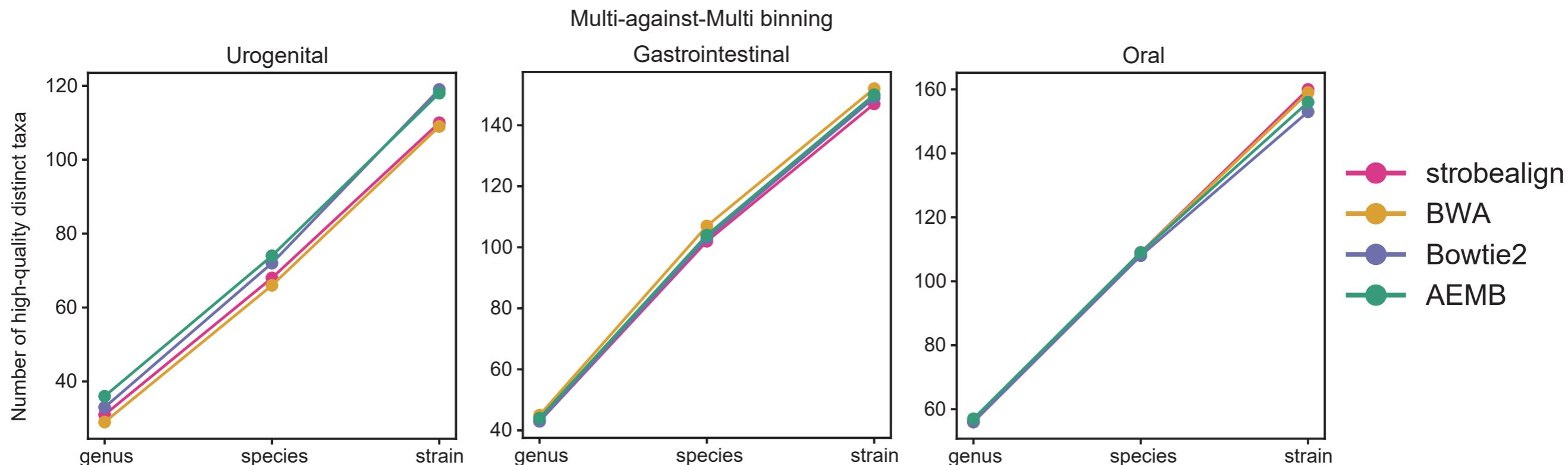
4 threads



\*Pan et al., forthcoming study

# strobealign new feature: --aemb

- **aemb**: “Abundance estimation metagenomic (contig) binning”
- Mapping-based coverage computation
- Multi-against-multi binning (Mattock and Watson, 2023)



# Conclusions and future work

## Strobemers

- Sensitive and specific (Sahlin, 2021)
- Entropy ('randomness') predicts sensitivity (Maier and Sahlin, 2023)
- Implementation affects the practical result (Karami et al., 2024)
- Multi-context seeds (MCS): query full and partial seed (Tolstoganov et al., 2024)

# Conclusions and future work

## Strobemers

- Sensitive and specific (Sahlin, 2021)
- Entropy ('randomness') predicts sensitivity (Maier and Sahlin, 2023)
- Implementation affects the practical result (Karami et al., 2024)
- Multi-context seeds (MCS): query full and partial seed (Tolstoganov et al., 2024)

## Strobealign

- Fast and accurate – for some lengths.. (Sahlin, 2022)
- Lower memory with prefix-table (Pan et al., forthcoming)
- Accurate for all lengths with MCS (Tolstoganov et al., 2024)
- aemb: rapid metagenomic coverage computation (Pan et al., forthcoming)

# Conclusions and future work

## Strobemers

- Sensitive and specific (Sahlin, 2021)
- Entropy ('randomness') predicts sensitivity (Maier and Sahlin, 2023)
- Implementation affects the practical result (Karami et al., 2024)
- Multi-context seeds (MCS): query full and partial seed (Tolstoganov et al., 2024)

## Strobealign

- Fast and accurate – for some lengths.. (Sahlin, 2022)
- Lower memory with prefix-table (Pan et al., forthcoming)
- Accurate for all lengths with MCS (Tolstoganov et al., 2024)
- aemb: rapid metagenomic coverage computation (Pan et al., forthcoming)

## Future work

- Split-read mapping: enabled with MCS
- Long-read mapping
- External-memory version
- (Piecewise extension, altstrobes)

# Conclusions and future work

## Strobemers

- Sensitive and specific (Sahlin, 2021)
- Entropy ('randomness') predicts sensitivity (Maier and Sahlin, 2023)
- Implementation affects the practical result (Karami et al., 2024)
- Multi-context seeds (MCS): query full and partial seed (Tolstoganov et al., 2024)

## Strobealign

- Fast and accurate – for some lengths.. (Sahlin, 2022)
- Lower memory with prefix-table (Pan et al., forthcoming)
- Accurate for all lengths with MCS (Tolstoganov et al., 2024)
- aemb: rapid metagenomic coverage computation (Pan et al., forthcoming)

## Future work

- Split-read mapping: enabled with MCS
- Long-read mapping
- External-memory version
- (Piecewise extension, altstrobes)

## Is there still time left?

- E-hits: a measure for seed repetitiveness given a reference (Sahlin, 2022)

# Acknowledgements

## Strobemers

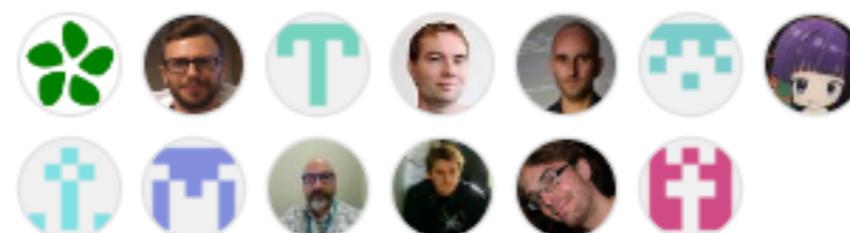
Benjamin Maier  
Moein Karami  
Aryan Soltani  
Wei Shen  
Rob Patro  
Giulio Pibri  
Lidong Guo  
Barış Ekim  
Mengyang Xu  
Daniel Liu  
Heng Li

## Strobealign

Marcel Martin  
Ivan Tolstoganov  
Luis Coelho  
Shaojun Pan  
Johan Gustavsson  
Sebastian Jaenicke  
Tom Conway  
Paul Theodor Pyl  
Martin Larralde  
Richèl Bilderbeek

...

## Contributors 13



 @ksahlin

 @ksahlin.bsky.social

 @krsahlin

## Funding

SciLifeLab

 Vetenskapsrådet  
Swedish Research Council

  
NATIONAL BIOINFORMATICS  
INFRASTRUCTURE SWEDEN





Funded by  
the European Union



# **Appendix**

# Measuring specificity

1. #Distinct seeds
2. Fraction of them occurring once
3. E-hits (Sahlin, 2022)
  - Pick a seed uniformly at random from index
  - What is the expected number of hits?

# Measuring specificity

1. #Distinct seeds
2. Fraction of them occurring once
3. E-hits (Sahlin, 2022)
  - Pick a seed uniformly at random from index
  - What is the expected number of hits?

$x_i$  : count seed i

$$p_i = x_i/N$$

$N$  : total seeds,  $M$  : distinct seeds

$$E[X] = \sum_{i=1}^M x_i p_i = \sum_{i=1}^M x_i \frac{x_i}{N} = \frac{1}{N} \sum_{i=1}^M x_i^2$$

# Measuring specificity

1. #Distinct seeds
2. Fraction of them occurring once
3. E-hits (Sahlin, 2022)
  - Pick a seed uniformly at random from index
  - What is the expected number of hits?

$$E[X] = \dots = \frac{1}{N} \sum_{i=1}^M x_i^2$$

Expected seed count      Total seeds      Distinct seeds      Count seed  $i$

Applicable to any seeding method  
A measure of average case query time?

# Measuring specificity

1. #Distinct seeds
2. Fraction of them occurring once
3. E-hits (Sahlin, 2022)
  - Pick a seed uniformly at random from index
  - What is the expected number of hits?

		CHM13		
		% Unique	E-hits	#Distinct (millions)
kmers	48	0.97	152.92	2706.93
randstrobes	(2,24,36,72)	0.98	181.70	2754.53

Conclusion: Comparable specificity for reasonable parameter choices

# E-hits

$x_i$  : count seed i

$p_i = x_i/N$

$N$  : total seeds,  $M$  : distinct seeds

$$E[X] = \sum_{i=1}^M x_i p_i = \sum_{i=1}^M x_i \frac{x_i}{N} = \frac{1}{N} \sum_{i=1}^M x_i^2$$

# Conclusions Moein et al?

- Link method influences both time and randomness
- Link method  $\text{XOR } (h(s_1) \oplus h(s_1))$  varies with argmin vs argmax

XOR with argmin: selects  $s_2 = s_1$  if present

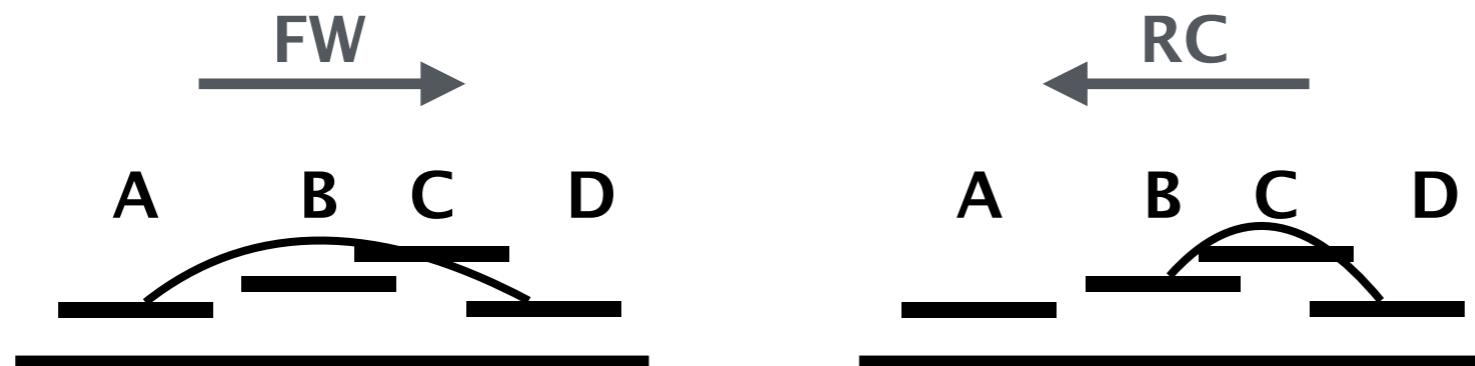


XOR with argmax – variations are selected for



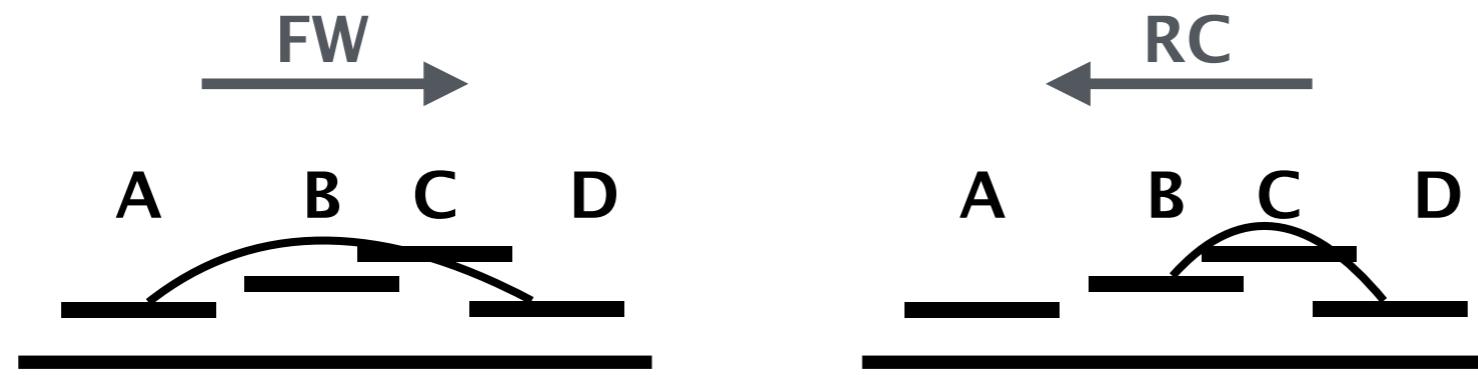
# Strobemers: Limitations

1. No guarantee to match most similar region 😞
2. True canonicity requires twice the work 😞



# Strobemers: Limitations

1. No guarantee to match most similar region 😞
2. True canonicity requires twice the work 😕



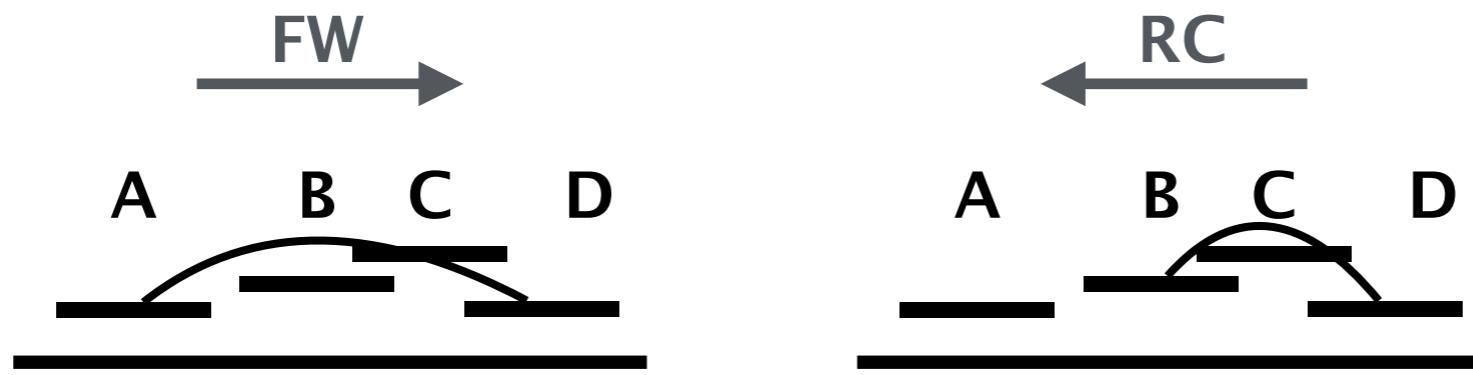
Remember:  $f(X, Y)$  computes the hash value of the seed

$$\min\{f(A, D), f(D, B)\}$$

Full canonicity

# Strobemers: Limitations

1. No guarantee to match most similar region 😞
2. True canonicity requires twice the work 😕



Remember:  $f(X, Y)$  computes the hash value of the seed

$$\min\{f(A, D), f(D, B)\}$$

Full canonicity

Symm.  $f(A, D) = f(D, A)$

Ex: A + B

'Some canonicity' (lol)

Asymm.  $f(A, D) \neq f(D, A)$

Ex: 2A - B

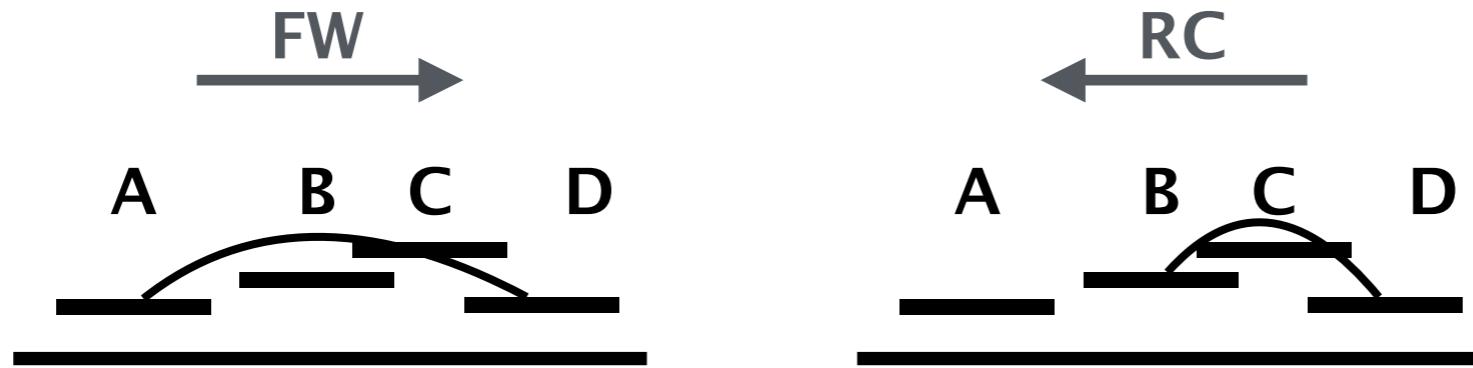
No canonicity

# Strobemers: Limitations

1. No guarantee to match most similar region



2. True canonicity requires twice the work



Remember:  $f(X, Y)$  computes the hash value of the seed

$$\min\{f(A, D), f(D, B)\}$$

Full canonicity

Symm.  $f(A, D) = f(D, A)$

Ex:  $A + B$

'Some canonicity' 😊

Asymm.  $f(A, D) \neq f(D, A)$

Ex:  $2A - B$

No canonicity

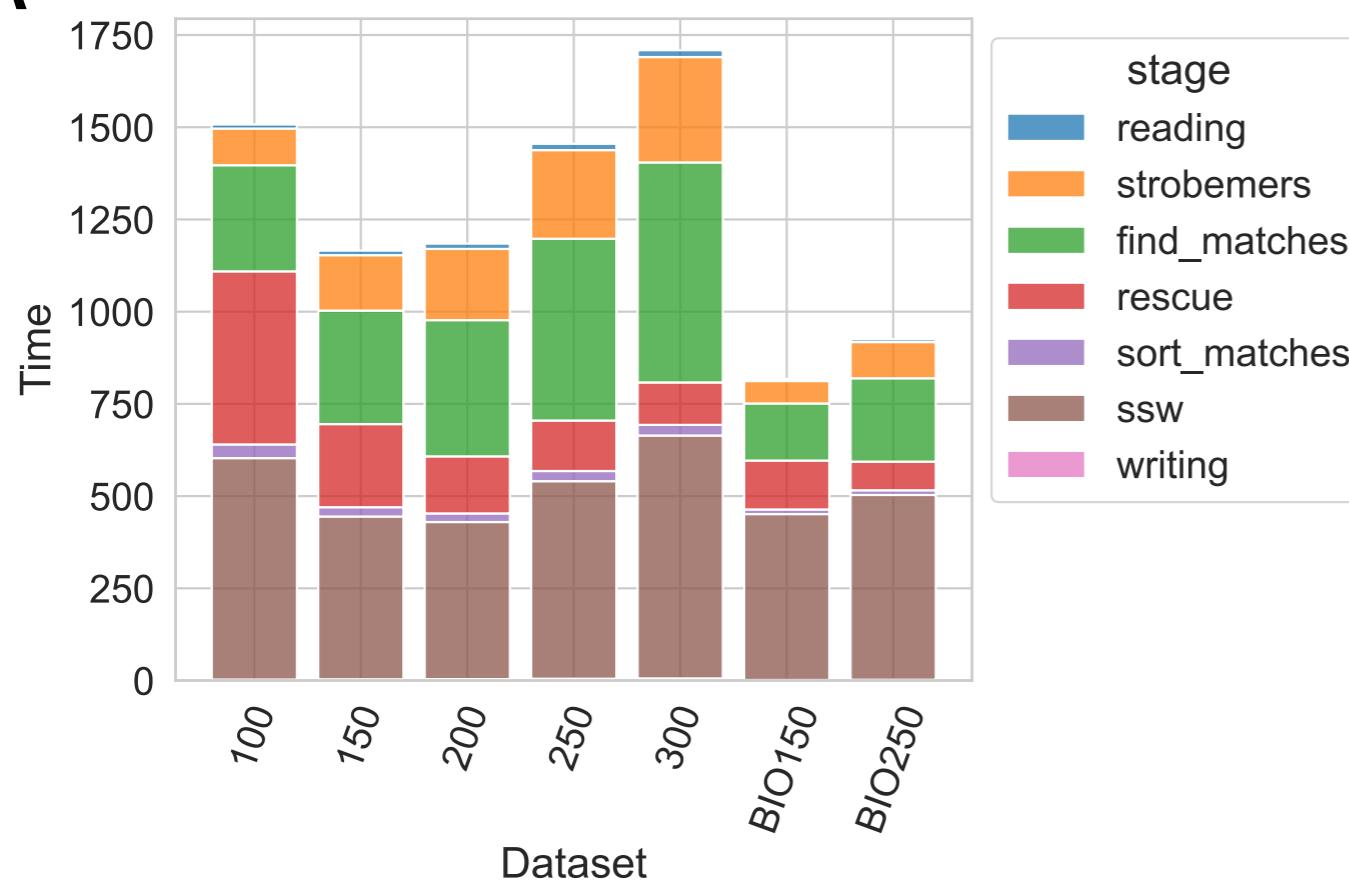
Indexing: one pass (FW)

Querying: both FW and RC

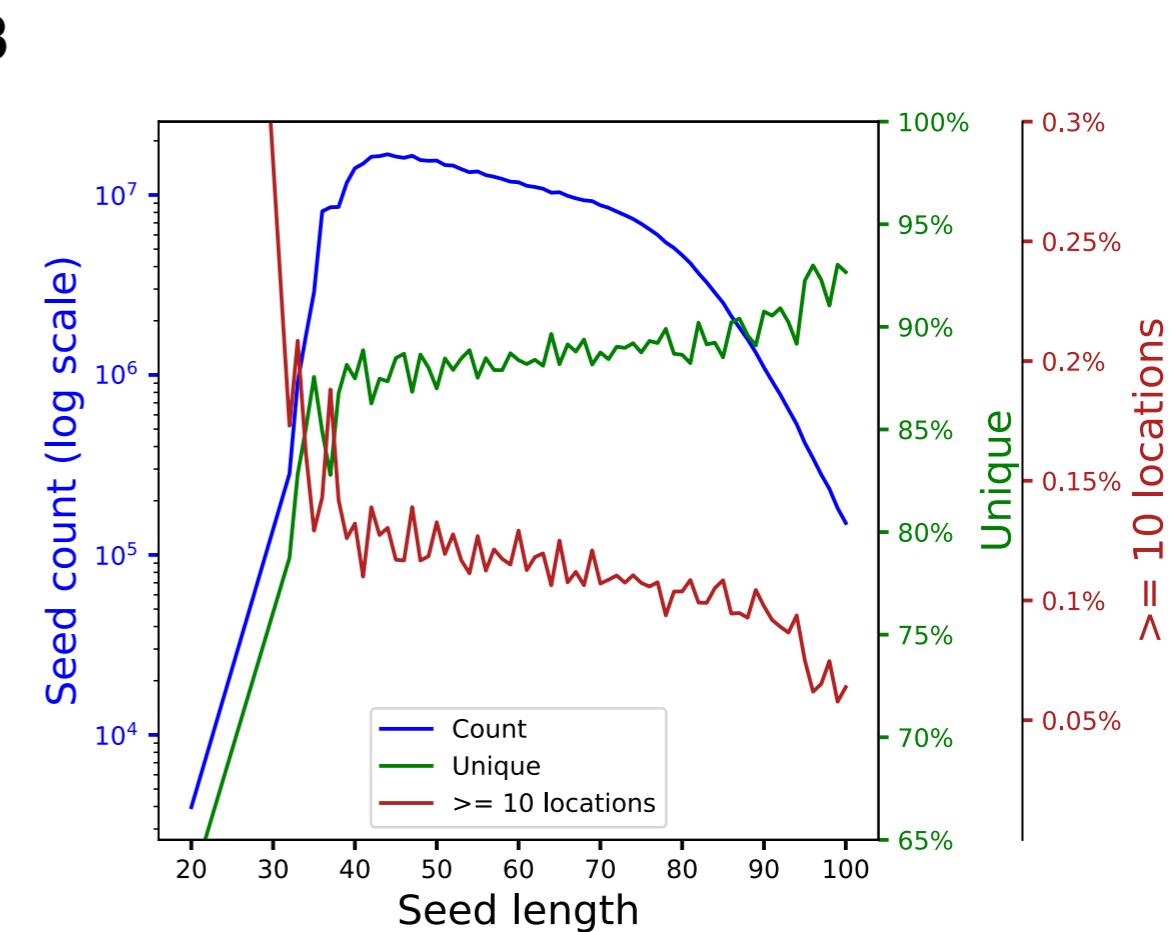
Symmetric  $f$  speeds up indexing, but at some cost:  
inversions together with masking repeat masking seeds

# Time allocation

A



B



# E-hits: Expected seed hits

- draw a seed uniformly at random from index
  - What is the expected number of hits?

$x_i$  : count seed

$$p_i = x_i/N$$

$N$ : total seeds,  $M$ : distinct seeds

$$E[X] = \sum_{i=1}^M x_i p_i = \sum_{i=1}^M x_i \frac{x_i}{N} = \frac{1}{N} \sum_{i=1}^M x_i^2$$

