# Comparative Genome Analysis using Sample-Specific String Detection in Accurate Long Reads

Parsoa Khorsand
**Luca Denti**
HGSVC
Paola Bonizzoni
Rayan Chikhi
Fereydoun Hormozdiari

# MOTIVATION AND CONTRIBUTION

Long reads improve variant detection:

- SVs in repeated regions of the genome
- Hard to detect variations

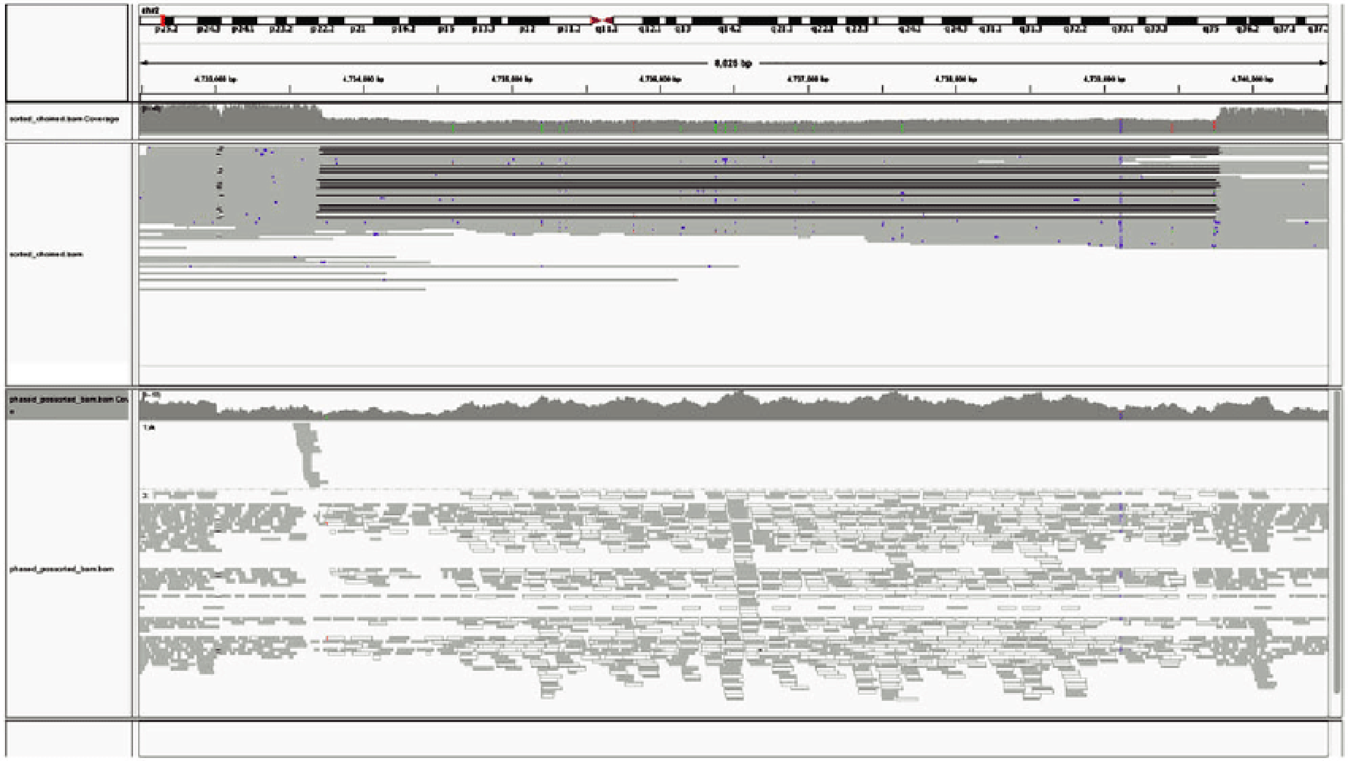

chr2
sorted_chained.bam Coverage
sorted_chained.bam
phased_possorted_bam.bam Coverage
phased_possorted_bam.bam

Fig.3 from Pollard et al. **Long reads: their purpose and place.** *Human molecular genetics* (2018)

# MOTIVATION AND CONTRIBUTION

Long reads improve variant detection:

- SVs in repeated regions of the genome
- Hard to detect variations

Current approaches:

- Alignment-based (*e.g.*, CuteSV) fails to detect complex SVs (inaccurate alignments)
- Alignment-free (*e.g.*, MALVA, Nebula) are limited by kmer size (short reads)

# MOTIVATION AND CONTRIBUTION

Long reads improve variant detection:

- SVs in repeated regions of the genome
- Hard to detect variations

Current approaches:

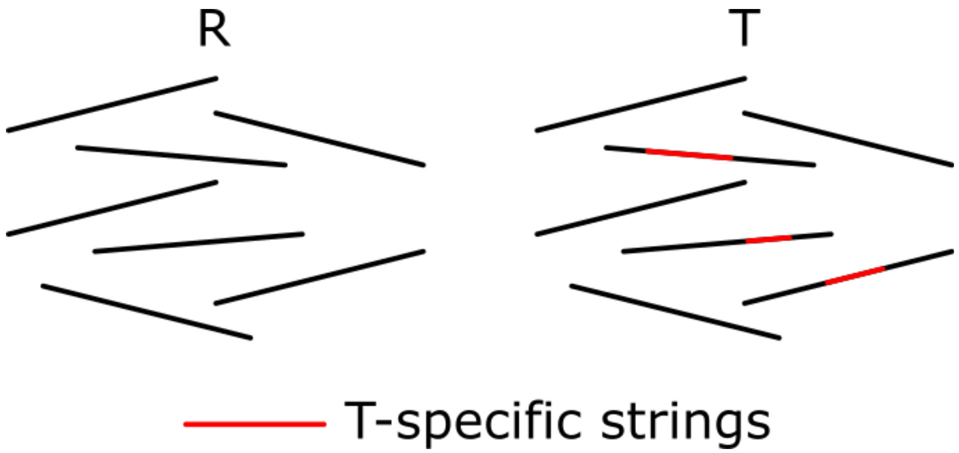
- Alignment-based (*e.g.*, CuteSV) fails to detect complex SVs (inaccurate alignments)
- Alignment-free (*e.g.*, MALVA, Nebula) are limited by kmer size (short reads)

Novel **alignment-free framework** for variant detection from HiFi long reads **not limited by** kmer size

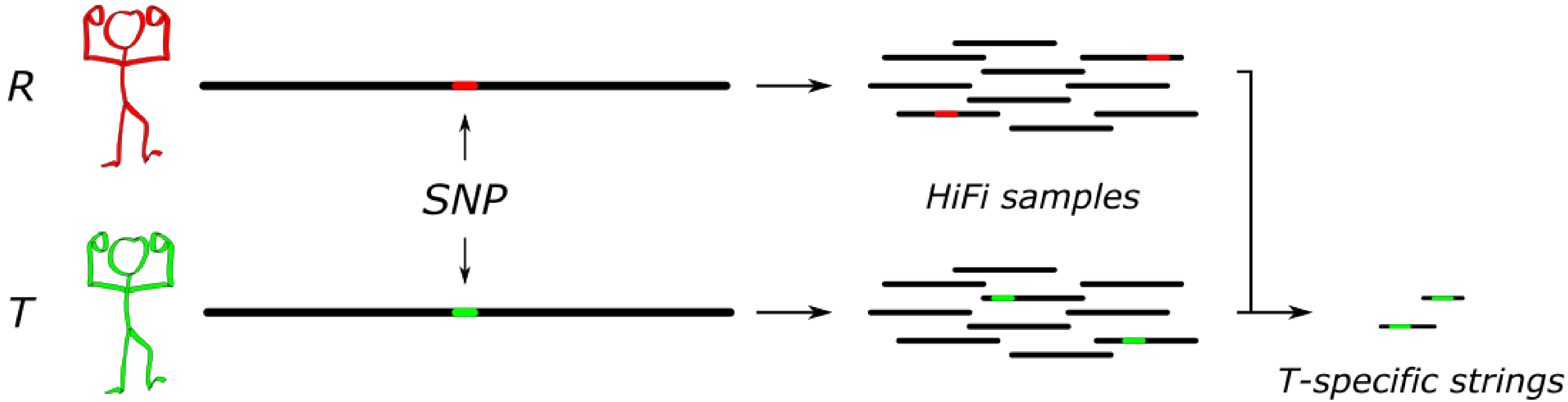
# SAMPLE-SPECIFIC STRINGS

Given two set of strings R and T, we define **T-specific** any string:

- occurring in T *(substring)*
- not occurring in R *(substring)*



# COMPARING TWO INDIVIDUALS



**Rationale:** any variation specific to T should produce one *(or more)* T-specific string

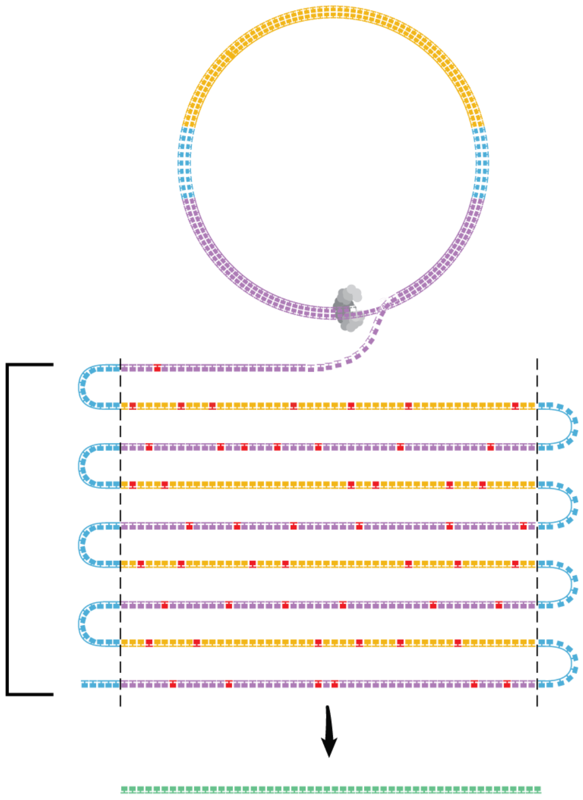
# PRELIMINARIES

# PACBIO SINGLE MOLECULE HIGH-FIDELITY (HiFi reads)

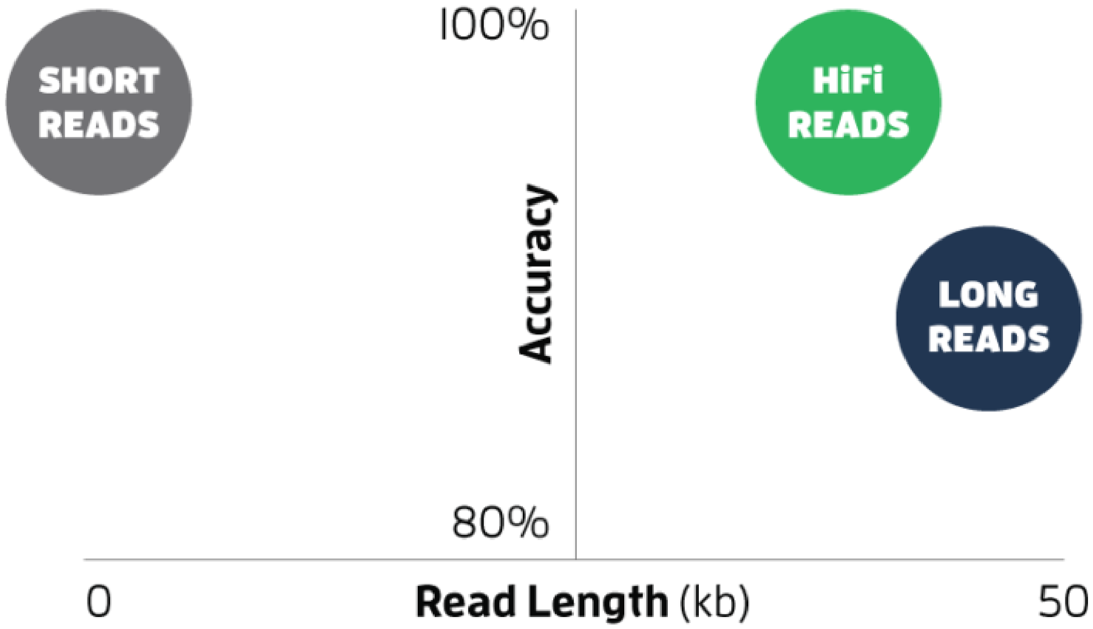
Circularized DNA is sequenced in repeated passes

The polymerase reads are trimmed of adapters to yield subreads

Consensus is called from subreads



HiFi READ
(>99% accuracy)



# FMD-INDEX[1]

FM-Index of a *bidirectional collection* of DNA sequences:

$$R_1\overline{R_1}R_2\overline{R_2}\ldots R_n\overline{R_n}$$

*( $\overline{R}$ is the reverse-and-complement of $R$ )*

[1] *Heng Li*. ***Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly.*** *Bioinformatics (2012)*

# FMD-INDEX[1]

FM-Index of a *bidirectional collection* of DNA sequences:

$$R_1\overline{R_1}R_2\overline{R_2}\ldots R_n\overline{R_n}$$

*( $\overline{R}$ is the reverse-and-complement of $R$ )*

Single index for both forward and reverse strands that allows efficient, *O(1)*, queries:

- backward extensions
- **forward extensions**

[1] *Heng Li. **Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly.** Bioinformatics (2012)*

# FMD-INDEX[1]

FM-Index of a *bidirectional collection* of DNA sequences:

$$R_1\overline{R_1}R_2\overline{R_2}\ldots R_n\overline{R_n}$$

*( $\overline{R}$ is the reverse-and-complement of $R$ )*

Single index for both forward and reverse strands that allows efficient, *O(1)*, queries:

- backward extensions
- **forward extensions**

Indeed, by backward extending pattern P with *a*, we also forward extend $\overline{P}$ with *t* .

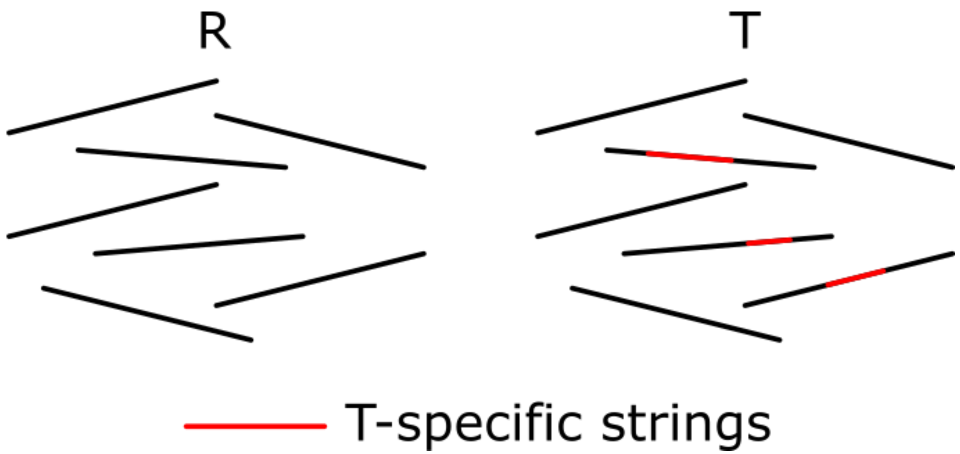
[1] *Heng Li.* ***Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly.*** *Bioinformatics (2012)*

# CONTRIBUTION

# T-SPECIFIC STRINGS

Given (R,T), we define **T-specific** any string:

- occurring in T (substring)
- not occurring in R (substring)

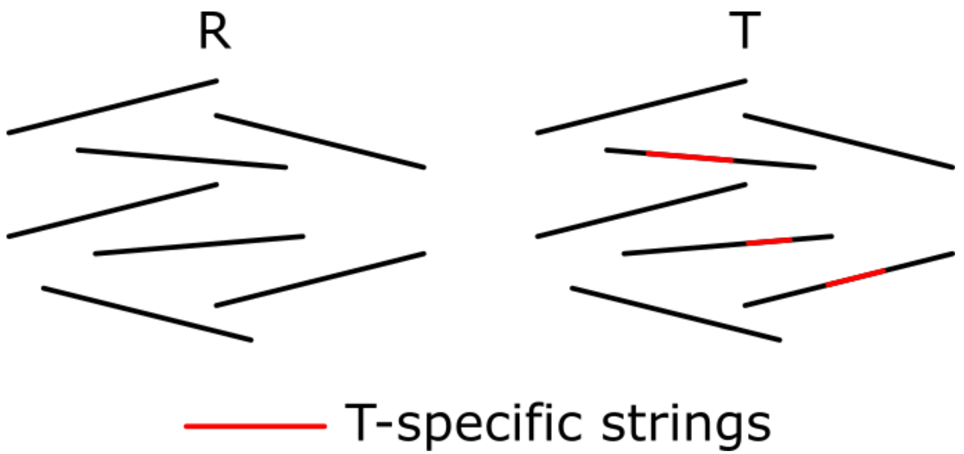


# T-SPECIFIC STRINGS

Given (R,T), we define **T-specific** any string:

- occurring in T (substring)
- not occurring in R (substring)

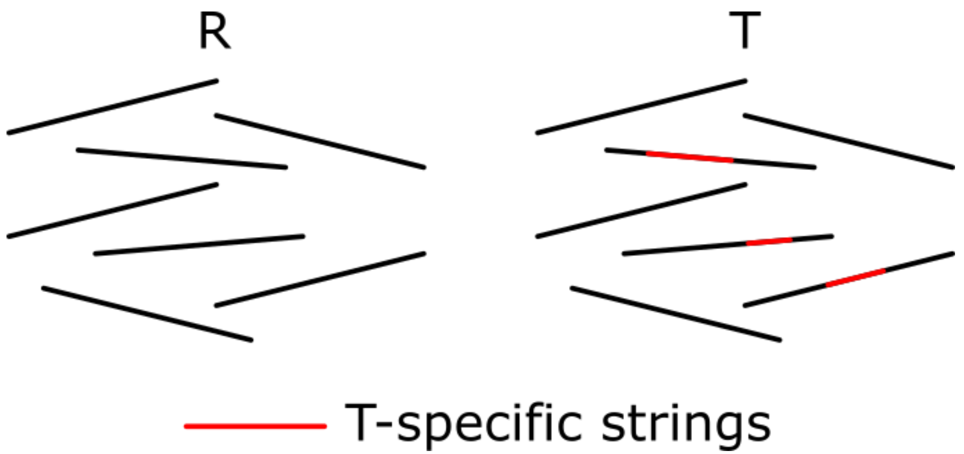
0 1 2 3 4 5 6
R = {ACATGAG}
T = {ACAAGAG}



# T-SPECIFIC STRINGS

Given (R,T), we define **T-specific** any string:

- occurring in T (substring)
- not occurring in R (substring)



```
     0 1 2 3 4 5 6
R = {A C A T G A G}
T = {A C A A G A G}
```

```
ACAAGAG
ACAAGA
 CAAGAG
  AAGAG
ACAAG
  AAG
   ...
```

$O(|T|^2)$ T-specific strings

# T-SPECIFIC STRINGS

Given (R,T), we define **T-specific** any string:

- occurring in T (substring)
- not occurring in R (substring)

- s.t. no other T-specific strings are contained in it

  e.g., it's the shortest *(substring-free property)*

0 1 2 3 4 5 6

R = {ACATGAG}

T = {ACAAGAG}

ACAAGAG

ACAAGA

CAAGAG

AAGAG

ACAAG

AAG

...

$O(|T|^2)$ T-specific strings

# T-SPECIFIC STRINGS

Given (R,T), we define **T-specific** any string:

- occurring in T (substring)
- not occurring in R (substring)

- s.t. no other T-specific strings are contained in it

  e.g., it’s the shortest *(substring-free property)*

$$\begin{array}{l} \phantom{R = \{}\texttt{0 1 2 3 4 5 6} \\ R = \{\texttt{ACATGAG}\} \\ T = \{\texttt{ACAAGAG}\} \end{array}$$

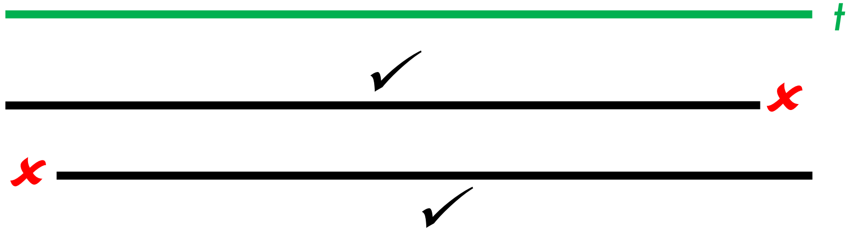
$$\left.\begin{array}{l} \texttt{ACAAGAG} \\ \texttt{ACAAGA} \\ \texttt{\phantom{A}CAAGAG} \\ \texttt{\phantom{AC}AAGAG} \\ \texttt{ACAAG} \\ \texttt{\phantom{AC}AAG} \\ \quad\ldots \end{array}\right\} O(|T|^2) \text{ T-specific strings}$$

$$\begin{array}{l} \texttt{\phantom{ACA}AGA} \\ \texttt{\phantom{AC}AA} \end{array} \Rightarrow \textit{Substring-free T-specific strings}$$

# PROPERTY OF T-SPECIFIC STRINGS

Let *t* be a *n*-long T-specific string. By definition:

| | |
|---|---|
| *t[0:n]* | **isn't** found in *R* |
| *t[0:n-1]* | **is** found in *R* |
| *t[1:n]* | **is** found in *R* |

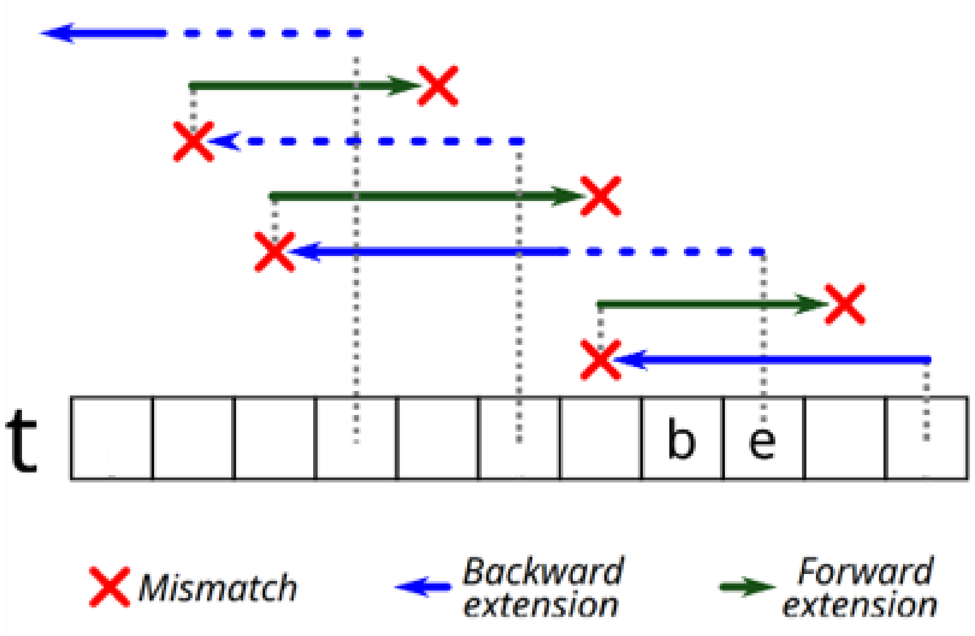


The first and last base of *t* are **mismatches.**

# ALGORITHM: PING-PONG

**Input:** two sets of strings (T,R)

**Output:** T-specific strings w.r.t. R

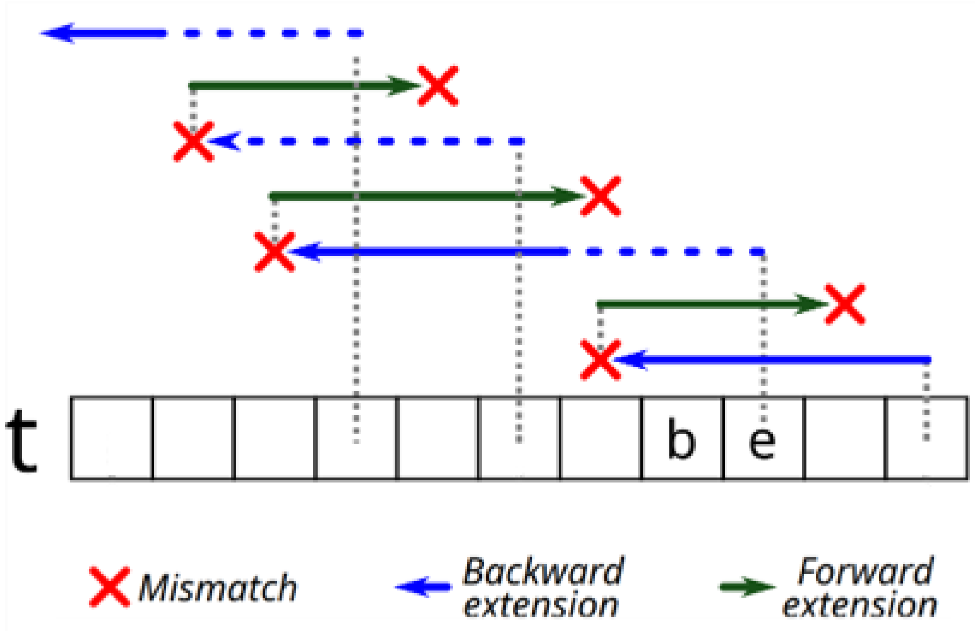


# ALGORITHM: PING-PONG

**Input:** two sets of strings (T,R)

**Output:** T-specific strings w.r.t. R

1. Build the FMD-Index of R

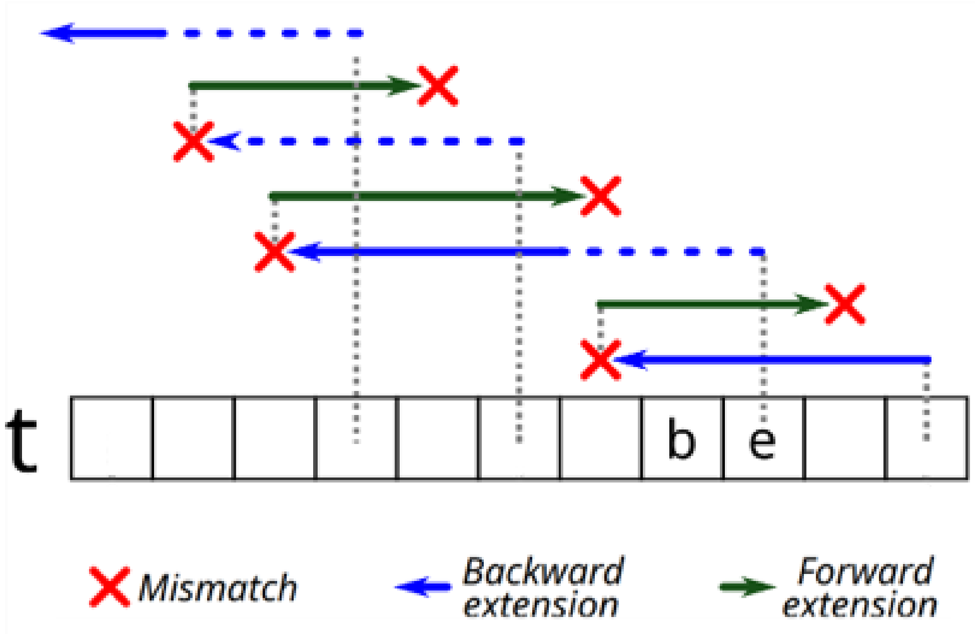


# ALGORITHM: PING-PONG

**Input:** two sets of strings (T,R)

**Output:** T-specific strings w.r.t. R

1. Build the FMD-Index of R
2. For each string t in T:
    1. Traverse backward the index until a mismatch
    2. Traverse forward the index until a mismatch
    3. Return the string between the mismatches (included)
    4. Reiterate

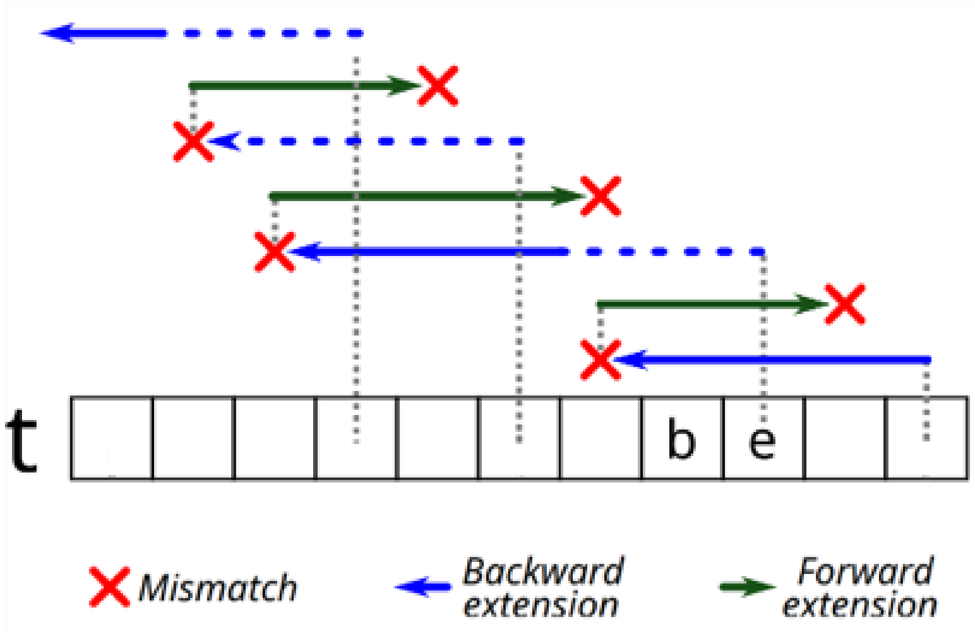


# ALGORITHM: PING-PONG

**Input:** two sets of strings (T,R)

**Output:** T-specific strings w.r.t. R

1. Build the FMD-Index of R
2. For each string t in T:
    1. Traverse backward the index until a mismatch
    2. Traverse forward the index until a mismatch
    3. Return the string between the mismatches (included)
    4. Reiterate



*Q1 to audience:* *is this notion related to the one of SMEMs?*

# COMPLEXITY

**Property:**

$$S_T = \bigcup_{t \in T} S_t$$

# COMPLEXITY

**Property:**

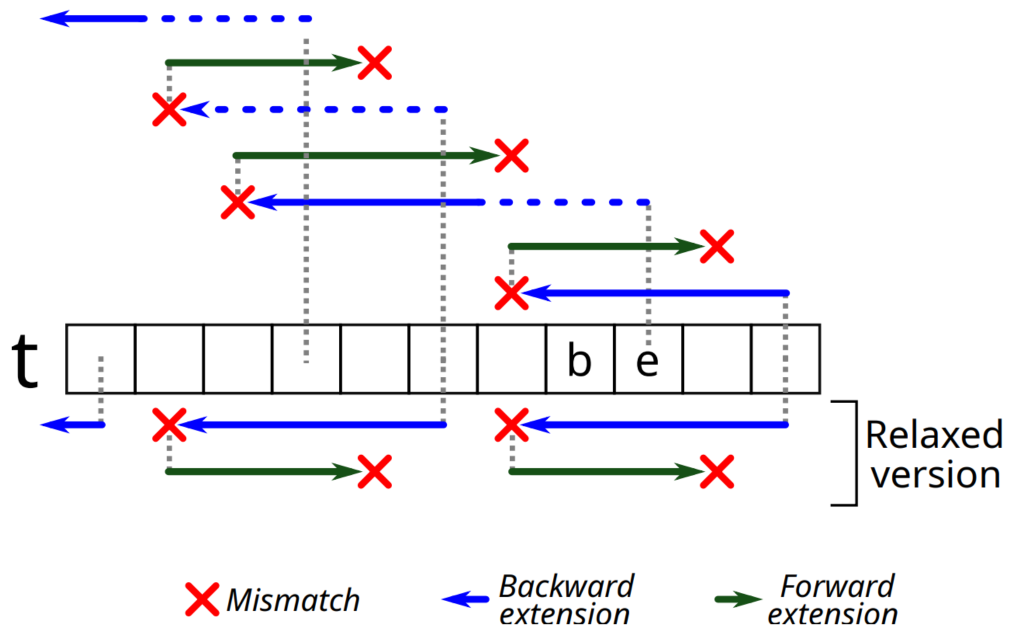
$$S_T = \bigcup_{t \in T} S_t$$

**Theorem:**

*Ping-Pong algorithm retrieves all T-specific strings in* $O(n^2)$*, where n is the total length of T.*

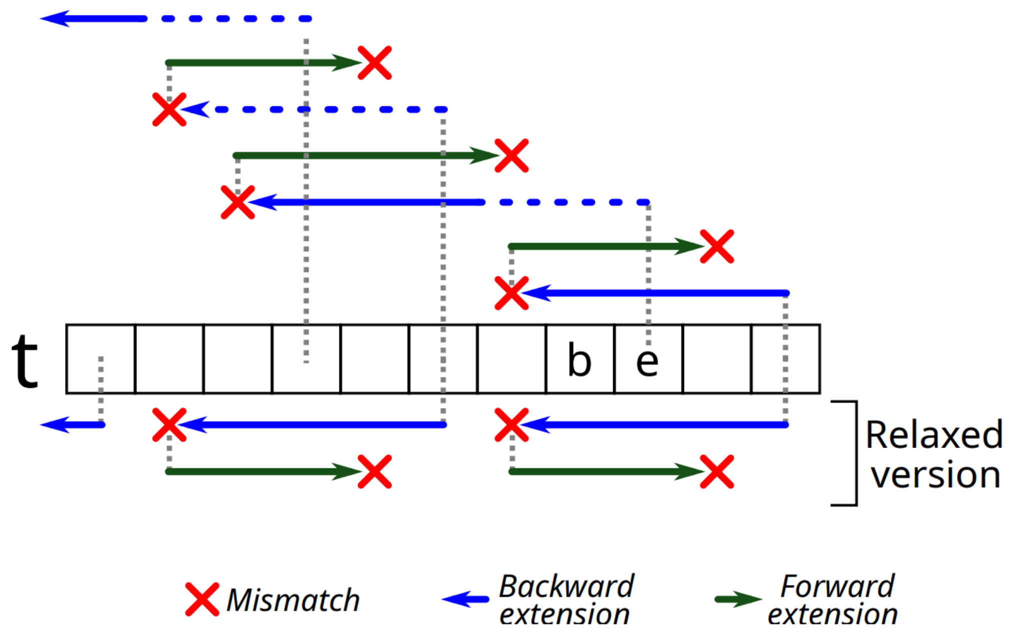
# PING-PONG (RELAXED)

Instead of retrieving all t-specific strings, we can retrieve all strings “non-overlapping” on t.



# PING-PONG (RELAXED)

Instead of retrieving all t-specific strings, we can retrieve all strings “non-overlapping” on t.

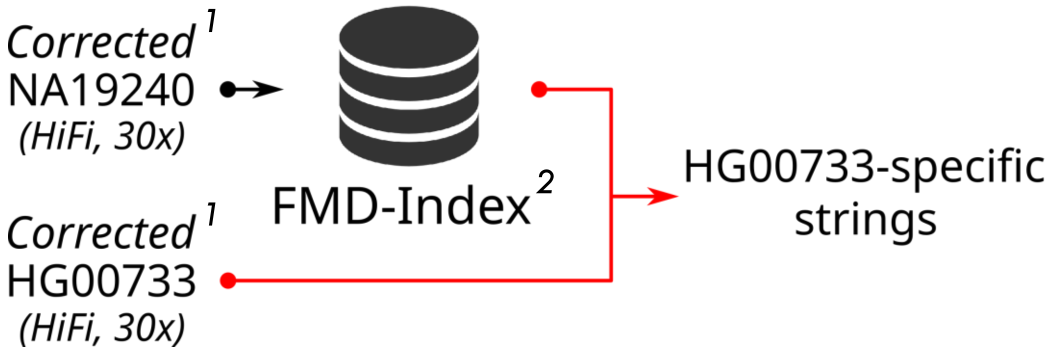


**Theorem:**

*Relaxed Ping-Pong algorithm retrieves a subset of T-specific strings in O(n).*

# EXPERIMENTAL EVALUATION

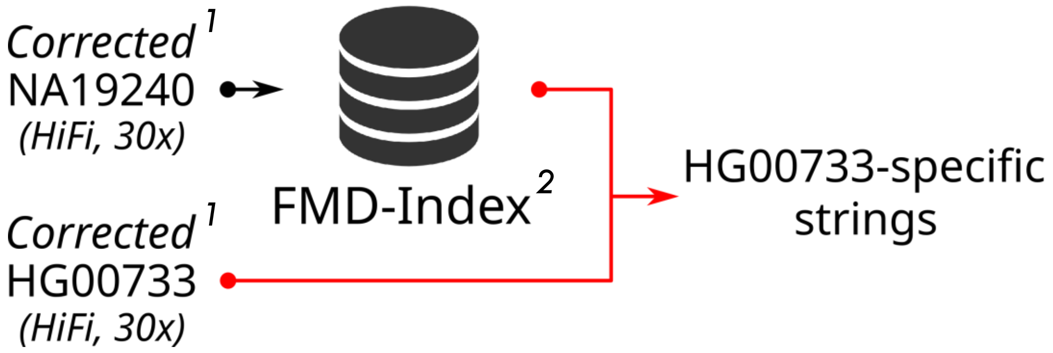
# REAL HIFI EXPERIMENTS



[1] Warren et al. ***ntEdit: scalable genome sequence polishing.*** *Bioinformatics (2019)*

[2] *Heng Li.* ***Fast construction of FM-index for long sequence reads.*** *Bioinformatics (2014)*

# REAL HIFI EXPERIMENTS

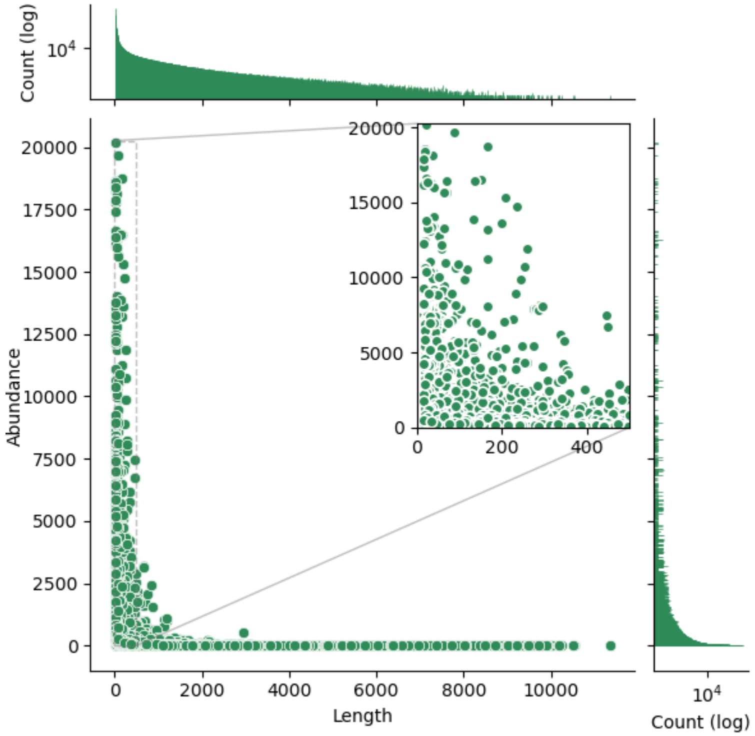


Corrected[1] NA19240 (HiFi, 30x)

FMD-Index[2]

Corrected[1] HG00733 (HiFi, 30x)

HG00733-specific strings



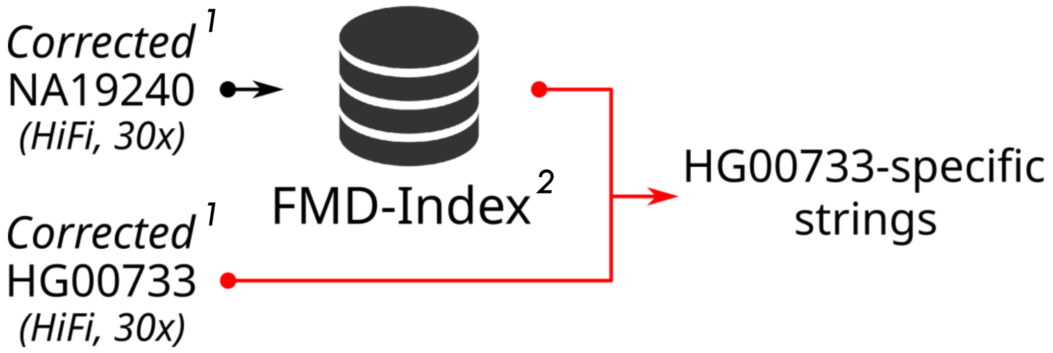
34 219 149 HG00733-specific strings (≥5 abundance)

*( 7 125 436 relaxed)*

[1] *Warren et al.* ***ntEdit: scalable genome sequence polishing.*** *Bioinformatics (2019)*

[2] *Heng Li.* ***Fast construction of FM-index for long sequence reads.*** *Bioinformatics (2014)*

# REAL HIFI EXPERIMENTS

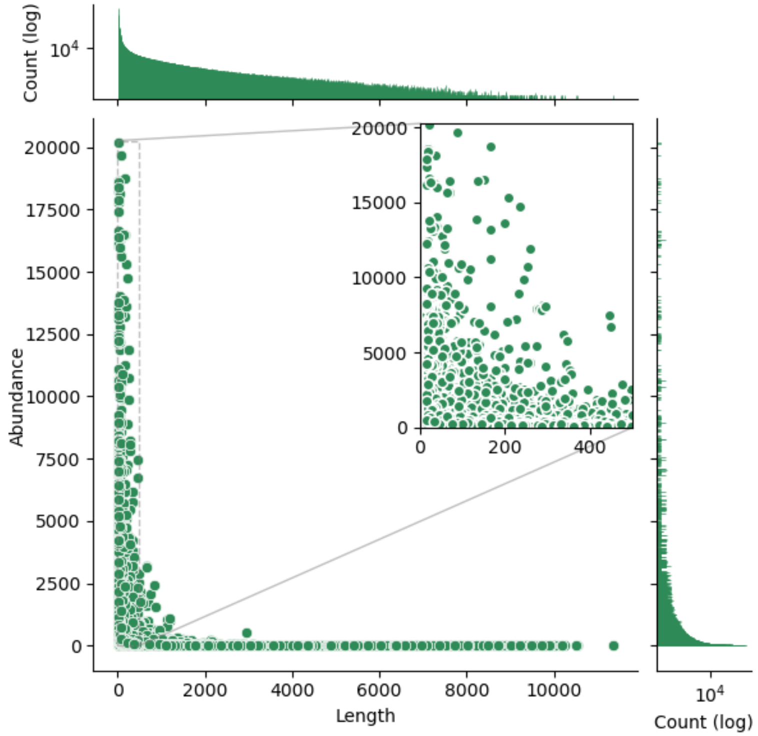


Corrected[1] NA19240 (HiFi, 30x)

Corrected[1] HG00733 (HiFi, 30x)

FMD-Index[2]

HG00733-specific strings



34 219 149 HG00733-specific strings (≥5 abundance)

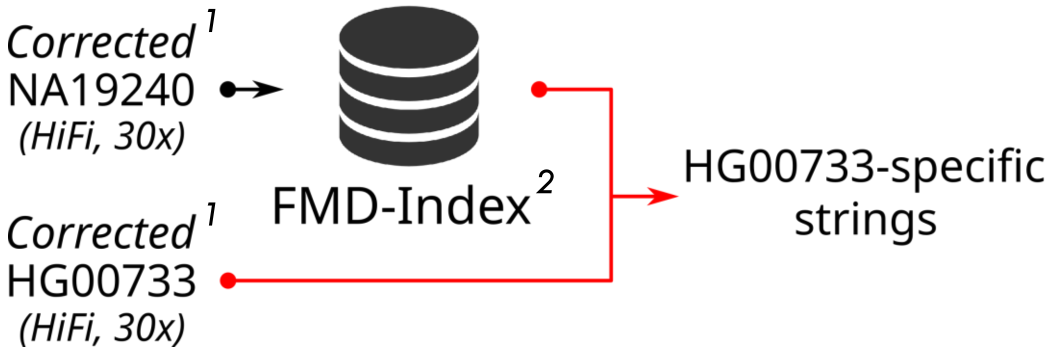
*( 7 125 436 relaxed)*

| | ntEdit | Indexing | Retrieval | Total/Peak |
|---|---|---|---|---|
| Time (hh:mm) | 05:03 | 20:30 😕 | 11:59 | 37:32 |
| RAM (GB) | 36 | 25 | 242 😅 | 242 |

[1] Warren et al. ***ntEdit: scalable genome sequence polishing.*** *Bioinformatics (2019)*

[2] Heng Li. ***Fast construction of FM-index for long sequence reads.*** *Bioinformatics (2014)*

# REAL HIFI EXPERIMENTS

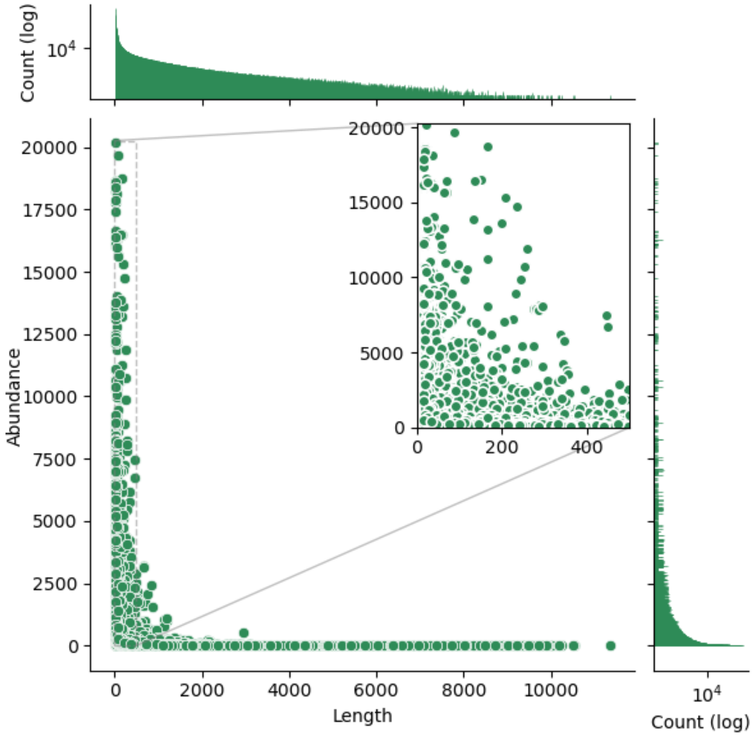


Corrected[1] NA19240 (HiFi, 30x)

FMD-Index[2]

Corrected[1] HG00733 (HiFi, 30x)

HG00733-specific strings



34 219 149 HG00733-specific strings (≥5 abundance)

*( 7 125 436 relaxed)*

| | ntEdit | Indexing | Retrieval | Total/Peak |
|---|---|---|---|---|
| Time (hh:mm) | 05:03 | 20:30 😕 | 11:59 | 37:32 |
| RAM (GB) | 36 | 25 | 242 😅 | 242 |

[1] Warren et al. ***ntEdit: scalable genome sequence polishing.*** *Bioinformatics (2019)*

[2] Heng Li. ***Fast construction of FM-index for long sequence reads.*** *Bioinformatics (2014)*

***Q2 to audience:*** *can we make indexing faster?*

# VALIDATION

1. **Contigs-based:** *check if our algorithm is correct*

2. **Haplotype-based:** *check if HG00733-specific strings cover variations specific to HG00733 (w.r.t. NA19240)*

**Comparison:** specific 31-mers and 101-mers *(KMC)*

# VALIDATION USING CONTIGS

**Goal:** *check if HG00733-specific strings are effectively specific* (algorithm correctness)

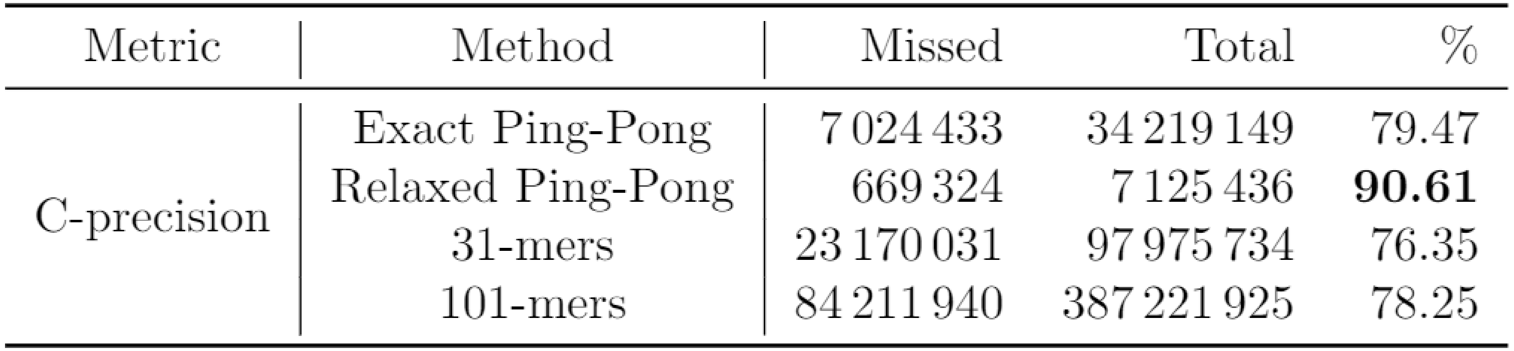
**How:**

1. get the HG00733 and NA19240 contigs *[Porubsky et al. Nat. Comm. 2020]*
2. align specific strings/kmers to the contigs *(bbmap2/bwa/minimap2)*
3. evaluate alignment quality

**Metric:** C-Precision *(fraction of strings aligned perfectly only to HG00733 contigs)*

# VALIDATION USING CONTIGS

**Goal:** *check if HG00733-specific strings are effectively specific (algorithm correctness)*



| Metric | Method | Missed | Total | % |
|---|---|---|---|---|
| C-precision | Exact Ping-Pong | 7 024 433 | 34 219 149 | 79.47 |
| | Relaxed Ping-Pong | 669 324 | 7 125 436 | **90.61** |
| | 31-mers | 23 170 031 | 97 975 734 | 76.35 |
| | 101-mers | 84 211 940 | 387 221 925 | 78.25 |

**Metric:** C-Precision *(fraction of strings aligned perfectly only to HG00733 contigs)*

# VALIDATION USING HAPLOTYPES

**Goal:** *check if HG00733-specific strings cover variations specific to HG00733*

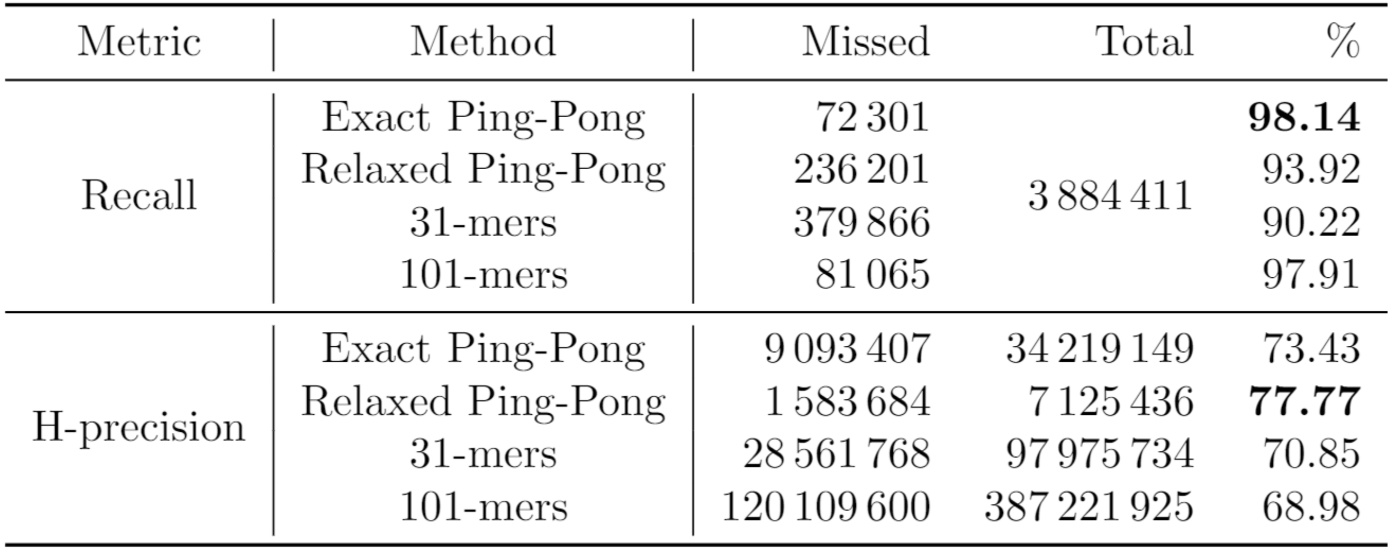
**How:**

| | |
|---|---|
| 1. get the known variations (VCF) | *[Porubsky et al. Nat. Comm. 2020]* |
| 2. build HG00733 haplotypes | *(BCFtools)* |
| 3. align specific strings/kmers to the haplotypes | *(bbmap2/bwa/minimap2)* |
| 4. Intersect alignments and specific variations | *(BEDtools)* |

**Metric:** Recall and H-Precision

# VALIDATION USING HAPLOTYPES

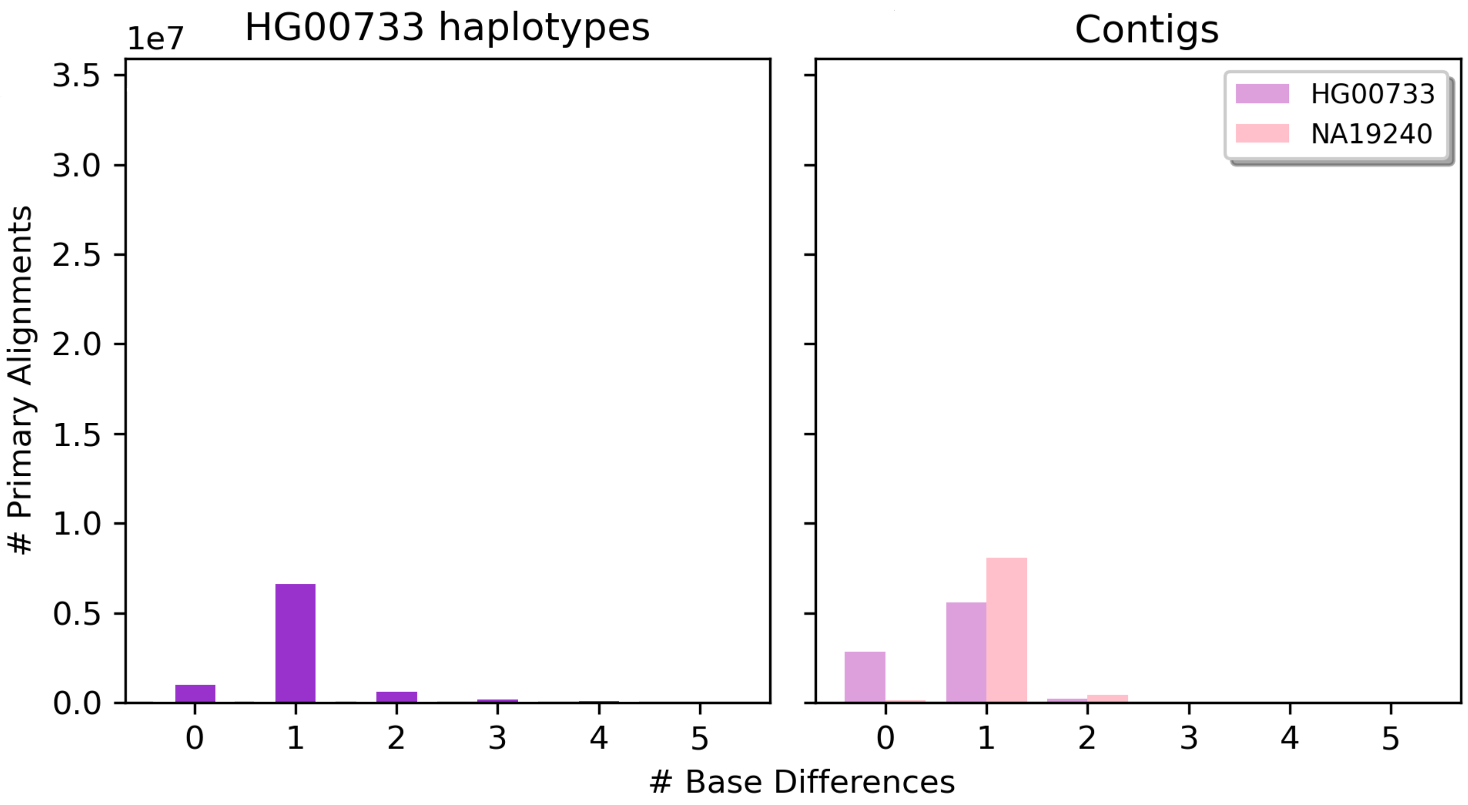
**Goal:** *check if HG00733-specific strings cover variations specific to HG00733*



| Metric | Method | Missed | Total | % |
|---|---|---|---|---|
| Recall | Exact Ping-Pong | 72 301 | 3 884 411 | **98.14** |
| | Relaxed Ping-Pong | 236 201 | | 93.92 |
| | 31-mers | 379 866 | | 90.22 |
| | 101-mers | 81 065 | | 97.91 |
| H-precision | Exact Ping-Pong | 9 093 407 | 34 219 149 | 73.43 |
| | Relaxed Ping-Pong | 1 583 684 | 7 125 436 | **77.77** |
| | 31-mers | 28 561 768 | 97 975 734 | 70.85 |
| | 101-mers | 120 109 600 | 387 221 925 | 68.98 |

**Metric:** Recall and H-Precision

# VALIDATION USING HAPLOTYPES



**Conjecture:** *the considered VCF is partially incomplete*

# CONCLUSIONS

https://github.com/Parsoa/PingPong

Sample-specific strings and Ping-Pong search *(FMD-Index based)*

**Take-home messages:**

- Specific strings effectively cover variations *(even better than specific kmers)*
- They may replace kmers in some scenarios - thanks to their variable-length nature

# CONCLUSIONS

https://github.com/Parsoa/PingPong

Sample-specific strings and Ping-Pong search *(FMD-Index based)*

**Take-home messages:**

- Specific strings effectively cover variations *(even better than specific kmers)*
- They may replace kmers in some scenarios - thanks to their variable-length nature

**Open questions:**

- Is our notion of sample-specific strings related to the one of SMEMs?
- Can we make the indexing faster?

# CONCLUSIONS

https://github.com/Parsoa/PingPong

Sample-specific strings and Ping-Pong search *(FMD-Index based)*

**Take-home messages:**

- Specific strings effectively cover variations *(even better than specific kmers)*
- They may replace kmers in some scenarios - thanks to their variable-length nature

**Open questions:**

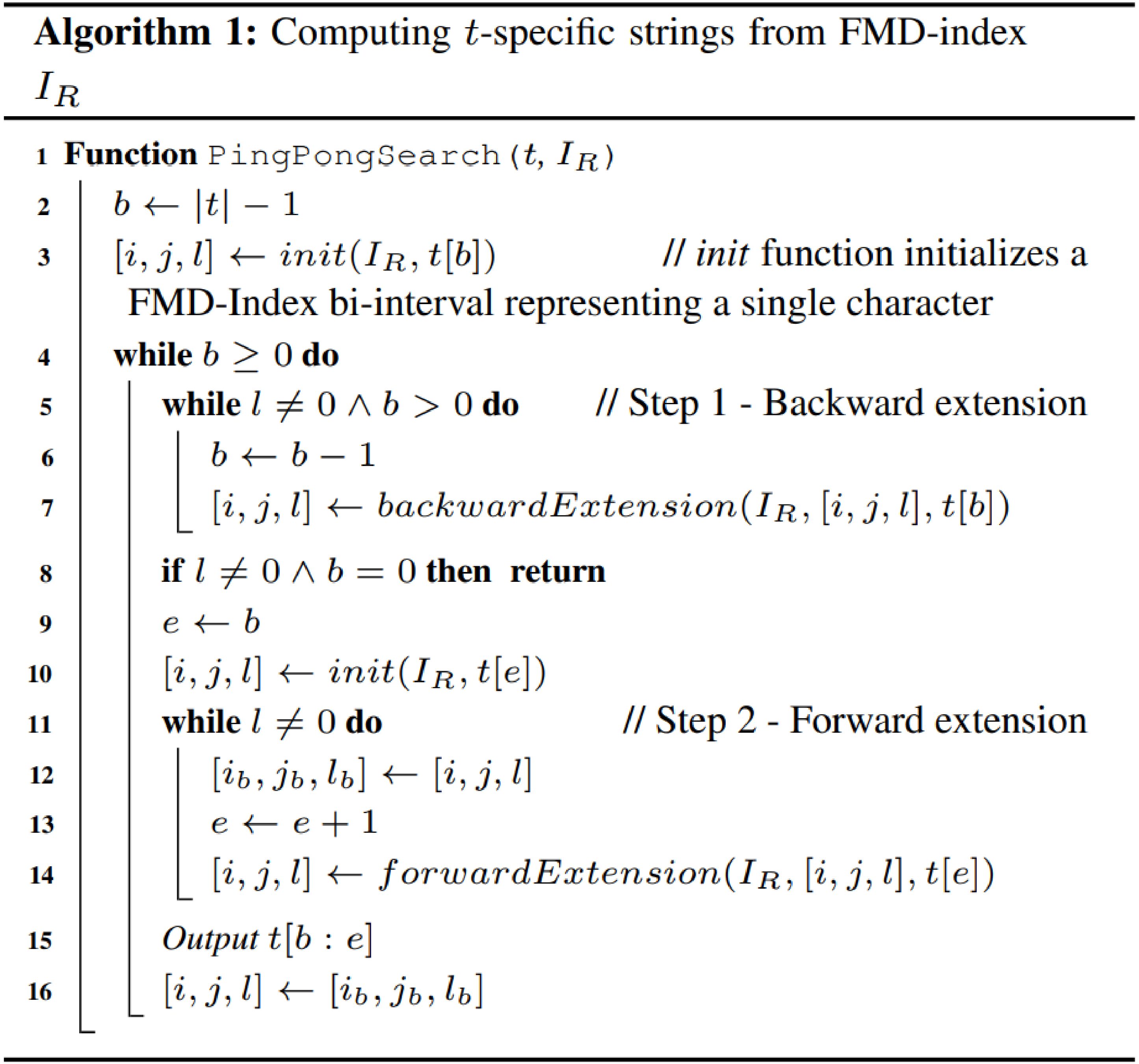
- Is our notion of sample-specific strings related to the one of SMEMs?
- Can we make the indexing faster?

**Future steps:** variation discovery via sample-specific strings detection

# THANK YOU!

Questions?

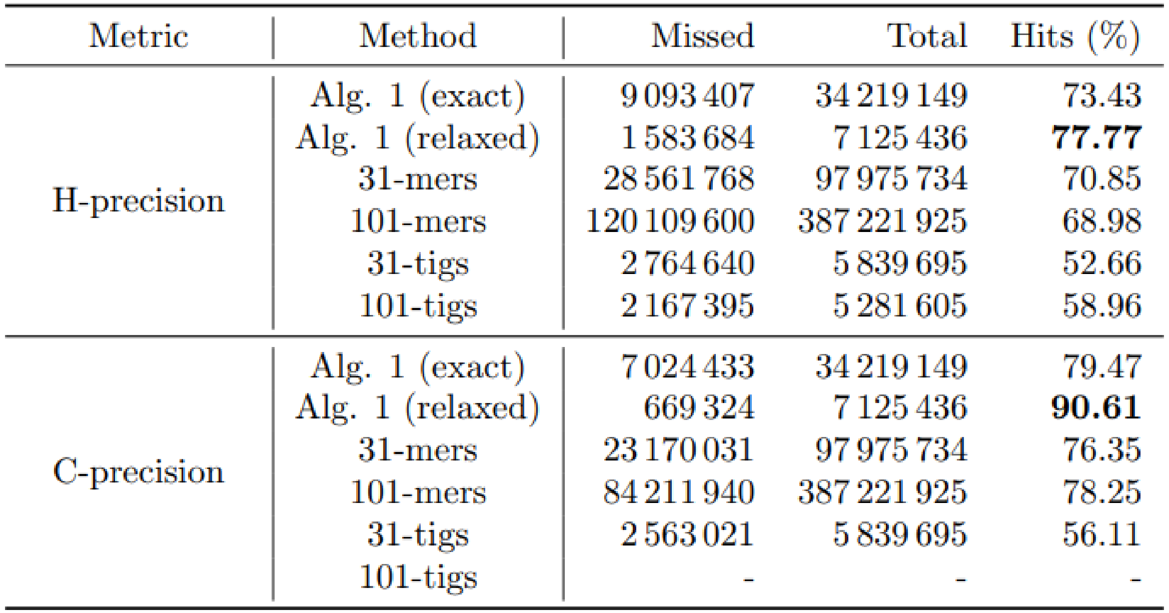
# PSEUDOCODE



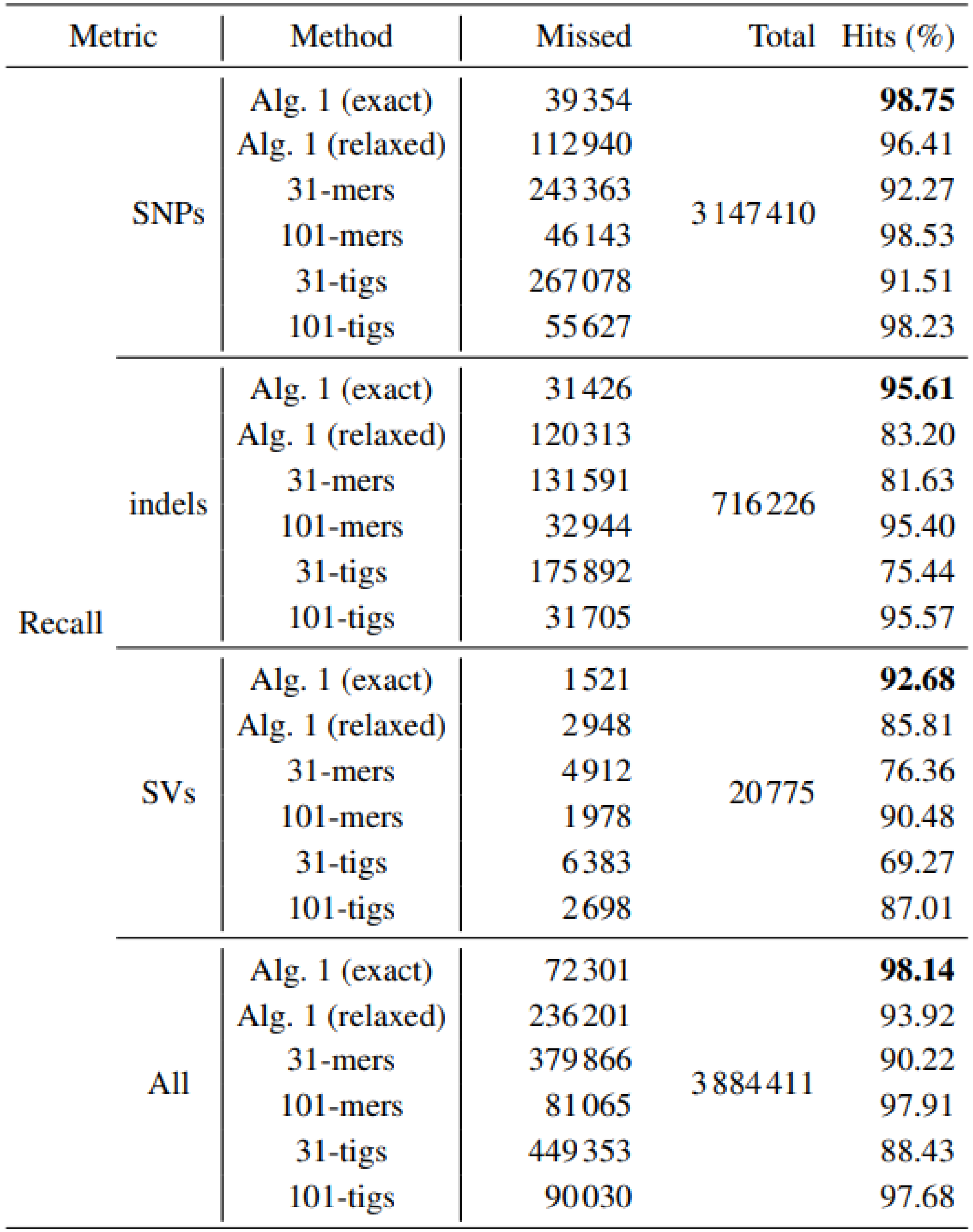
**Algorithm 1:** Computing $t$-specific strings from FMD-index $I_R$

```
Function PingPongSearch($t$, $I_R$)
    $b \leftarrow |t| - 1$
    $[i, j, l] \leftarrow init(I_R, t[b])$    // init function initializes a FMD-Index bi-interval representing a single character
    while $b \geq 0$ do
        while $l \neq 0 \wedge b > 0$ do    // Step 1 - Backward extension
            $b \leftarrow b - 1$
            $[i, j, l] \leftarrow backwardExtension(I_R, [i, j, l], t[b])$
        if $l \neq 0 \wedge b = 0$ then return
        $e \leftarrow b$
        $[i, j, l] \leftarrow init(I_R, t[e])$
        while $l \neq 0$ do    // Step 2 - Forward extension
            $[i_b, j_b, l_b] \leftarrow [i, j, l]$
            $e \leftarrow e + 1$
            $[i, j, l] \leftarrow forwardExtension(I_R, [i, j, l], t[e])$
        Output $t[b : e]$
        $[i, j, l] \leftarrow [i_b, j_b, l_b]$
```

# FULL RESULTS



| Metric | Method | Missed | Total | Hits (%) |
|---|---|---|---|---|
| H-precision | Alg. 1 (exact) | 9 093 407 | 34 219 149 | 73.43 |
| | Alg. 1 (relaxed) | 1 583 684 | 7 125 436 | **77.77** |
| | 31-mers | 28 561 768 | 97 975 734 | 70.85 |
| | 101-mers | 120 109 600 | 387 221 925 | 68.98 |
| | 31-tigs | 2 764 640 | 5 839 695 | 52.66 |
| | 101-tigs | 2 167 395 | 5 281 605 | 58.96 |
| C-precision | Alg. 1 (exact) | 7 024 433 | 34 219 149 | 79.47 |
| | Alg. 1 (relaxed) | 669 324 | 7 125 436 | **90.61** |
| | 31-mers | 23 170 031 | 97 975 734 | 76.35 |
| | 101-mers | 84 211 940 | 387 221 925 | 78.25 |
| | 31-tigs | 2 563 021 | 5 839 695 | 56.11 |
| | 101-tigs | - | - | - |



| Metric | | Method | Missed | Total | Hits (%) |
|---|---|---|---|---|---|
| Recall | SNPs | Alg. 1 (exact) | 39 354 | 3 147 410 | **98.75** |
| | | Alg. 1 (relaxed) | 112 940 | | 96.41 |
| | | 31-mers | 243 363 | | 92.27 |
| | | 101-mers | 46 143 | | 98.53 |
| | | 31-tigs | 267 078 | | 91.51 |
| | | 101-tigs | 55 627 | | 98.23 |
| | indels | Alg. 1 (exact) | 31 426 | 716 226 | **95.61** |
| | | Alg. 1 (relaxed) | 120 313 | | 83.20 |
| | | 31-mers | 131 591 | | 81.63 |
| | | 101-mers | 32 944 | | 95.40 |
| | | 31-tigs | 175 892 | | 75.44 |
| | | 101-tigs | 31 705 | | 95.57 |
| | SVs | Alg. 1 (exact) | 1 521 | 20 775 | **92.68** |
| | | Alg. 1 (relaxed) | 2 948 | | 85.81 |
| | | 31-mers | 4 912 | | 76.36 |
| | | 101-mers | 1 978 | | 90.48 |
| | | 31-tigs | 6 383 | | 69.27 |
| | | 101-tigs | 2 698 | | 87.01 |
| | All | Alg. 1 (exact) | 72 301 | 3 884 411 | **98.14** |
| | | Alg. 1 (relaxed) | 236 201 | | 93.92 |
| | | 31-mers | 379 866 | | 90.22 |
| | | 101-mers | 81 065 | | 97.91 |
| | | 31-tigs | 449 353 | | 88.43 |
| | | 101-tigs | 90 030 | | 97.68 |