Mapping-friendly sequence reductions:

Going beyond homopolymer compression

Luc Blassel, Paul Medvedev, Rayan Chikhi

INSTITUT PASTEUR

PR[AI]RIE
PaRis Artificial Intelligence Research InstitutE

SORBONNE UNIVERSITÉ

dy

A little bit of context

Context - Long read mapping

- Sequencing **errors complicate mapping** *(Gusfield, 1997)*
- **Long read** sequencing errors *(Dohm et al. 2020)*:
  - Short **indels**
  - Particularly **in homopolymers**

10.1145/270563.571472
10.1093/nargab/lqaa037

Context - what is HPC ?

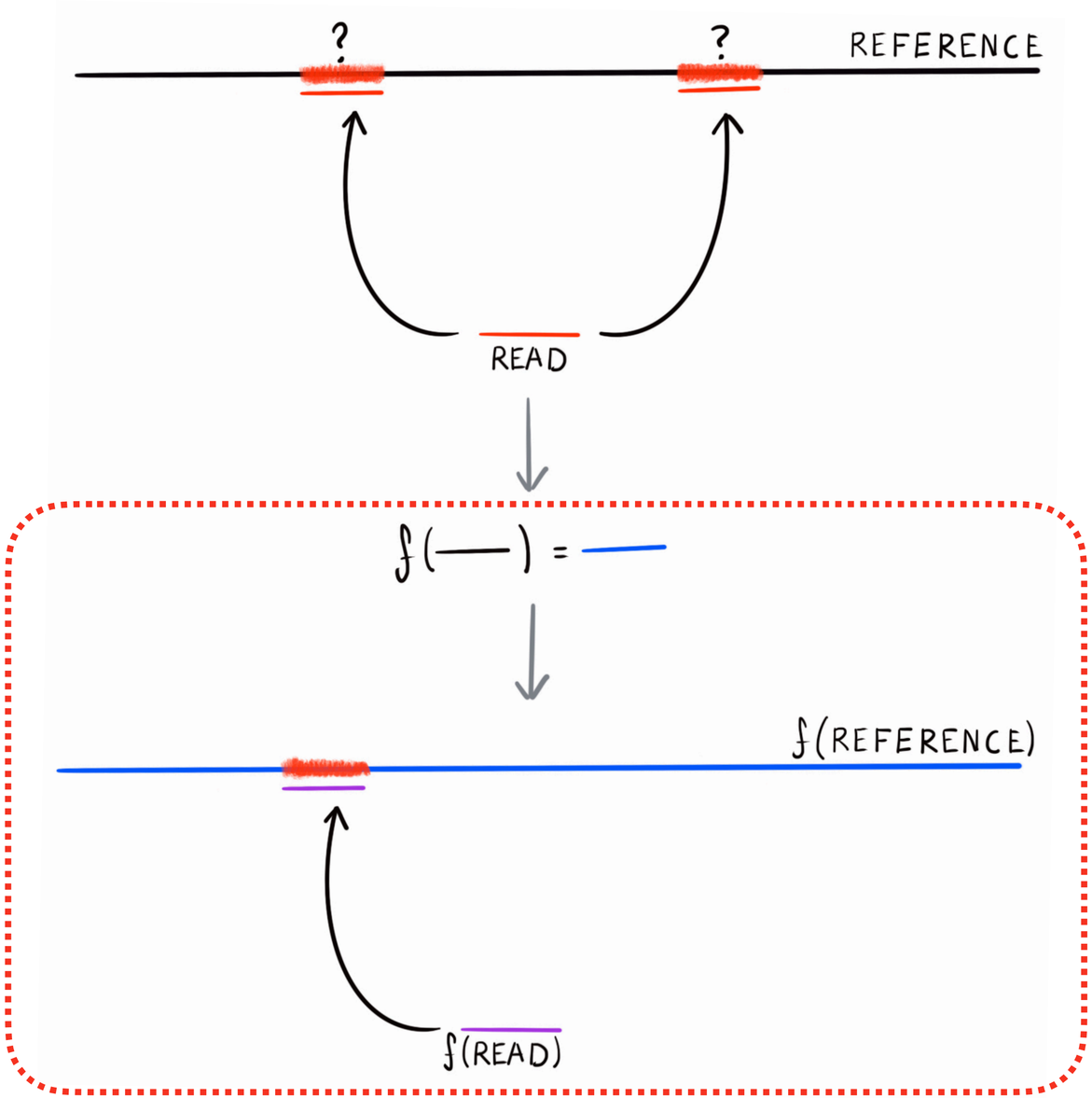
- **Homopolymer compression** (HPC) transforms sequences:

HPC(AAAATTTGGGCCCCCGGGTA) → ATGCGTA

- HPC is a **function** that **transforms strings** of characters into other strings

- Empirically it **improves mapping**, but **no guarantee it's the best**

Goals



Can we find functions f that **improve mapping more than HPC** ?

Mapping-friendly sequence reductions

A formal definition of HPC

- Let us define $\Sigma = \{A, C, G, T\}$ and ε the empty character
- Let $g^{HPC} : \Sigma^2 \rightarrow \Sigma \cup \{\varepsilon\}$ s.t. $\forall (x_1, x_2) \in \Sigma^2$

$$g^{HPC}(x_1 \cdot x_2) = \begin{cases} x_2 & if\ x_1 \neq x_2 \\ \varepsilon & if\ x_1 = x_2 \end{cases}$$

- HPC(x) = applying g^{HPC} on a sliding window of size 2 along x and concatenating outputs.
- Different g = MSR

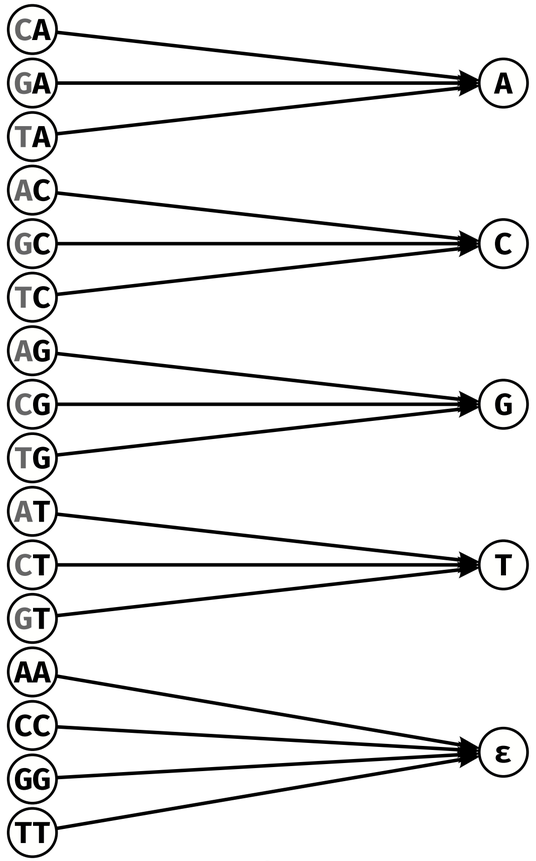
The DAG visualisation

- Each g function can be visualised as a directed graph defined by a mapping between $|\Sigma^{\ell}|$ inputs and $|\Sigma| + 1$ outputs

- HPC as a directed graph

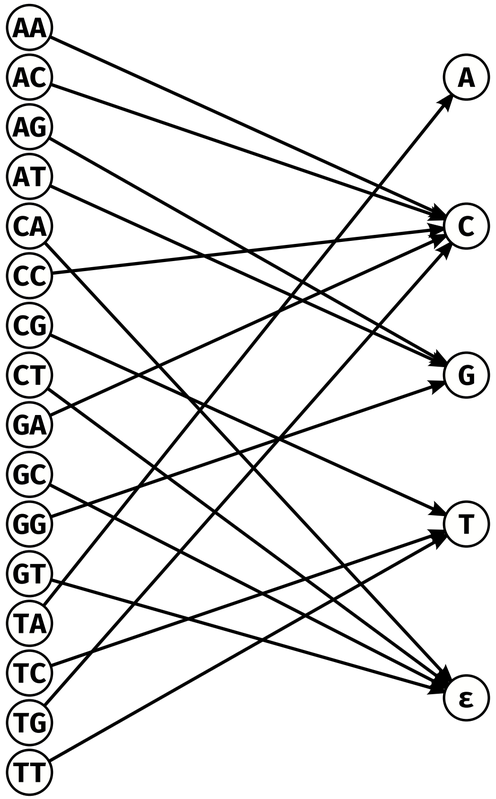
  (n=16 inputs k=5 outputs)

- There are 5^{16} functions $g : \Sigma^2 \rightarrow \Sigma \cup \{\varepsilon\}$



Reducing the search space - RC

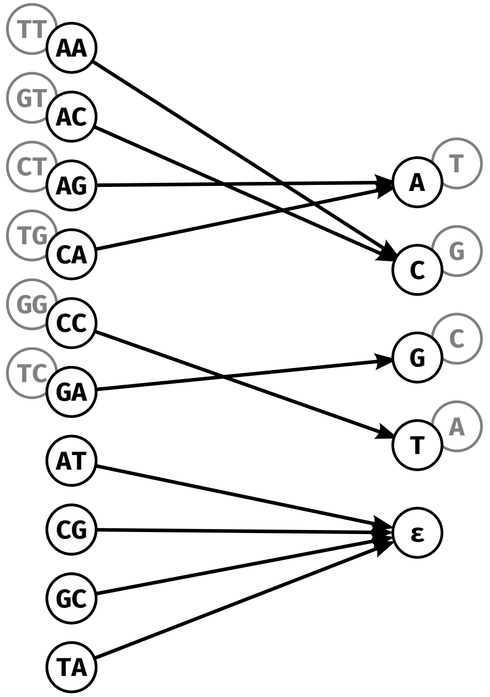
Random MSR f_r ❌



$$x = TAAGTTGA$$
$$f_r(x) = TACGTCC$$
$$RC(x) = TCAACTTA$$

$$f_r(RC(x)) = TTCCTA$$
$$RC(f_r(x)) = GGACGTA$$

RC-core-insensitive MSR f ✅



$$x = TAAGTTGA$$
$$f(x) = TCAGGTG$$
$$RC(x) = TCAACTTA$$

$$f(RC(x)) = TCACCTG$$
$$RC(f(x)) = CACCTGA$$

Reducing the search space - RC

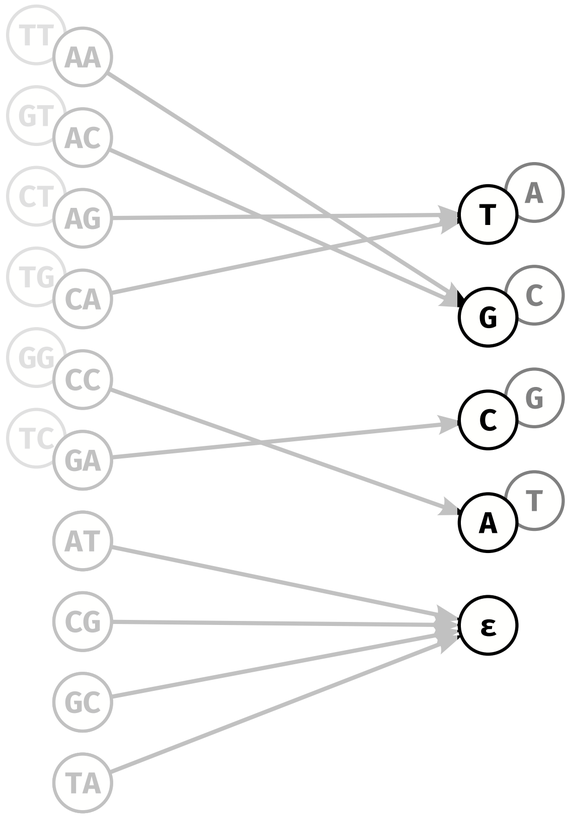
- There are $5^6 \approx 1.5 \cdot 10^4$ RC-core-insensitive MSRs
- Mapping is **computationally expensive**

We need to **reduce** the search space **even more**

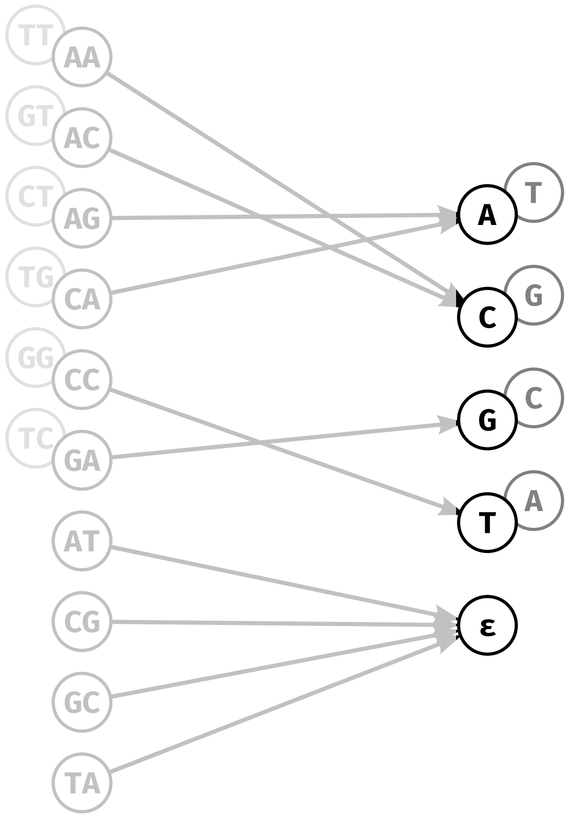
Reducing the search space - equivalence

- Symmetries:
  - $A \Leftrightarrow T$ and $G \Leftrightarrow C$
  - $(G/C)_{pair} \Leftrightarrow (A/T)_{pair}$
- We define **equivalence classes** from them

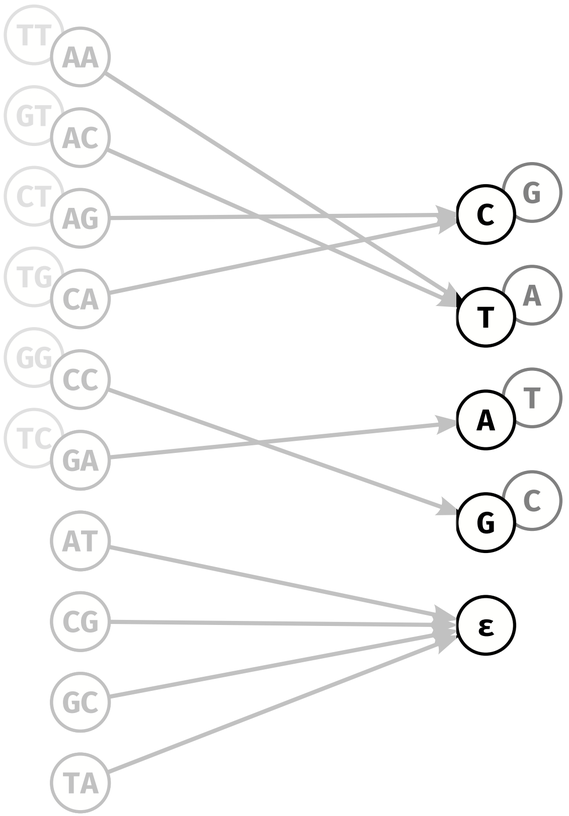
Reducing the search space - equivalence



$$A \leftrightarrow T$$
$$C \leftrightarrow G$$



$$A/T \leftrightarrow C/G$$



Reducing the search space - final words

5^{16}

↓

5^{6}

↓

2135 MSRs to look at

How do we evaluate MSRs ?

Evaluating MSRs - datasets

- Simulate ONT reads, with **`nanosim`**, on 4 references:
  - **Whole human genome**, CHM13hTERT human cell line by the T2T
  - **Whole *Drosophila* genome**, *Adams et al. (2022)*
  - **Whole *Escherichia coli* genome**, *Blattner et al. (1977)*
  - **Synthetic human centromeric sequence**, *Mikheenko et al. (2020)* `tandemtools` mapper test data

Can MSRs **improve mapping** of simulated reads?

Evaluating MSRs - evaluation pipeline

- For **each (MSR, reference) pair** (and no MSR i.e. *raw*):
  1. **Transform** the reference and reads with the MSR
  2. **Map** transformed reads to transformed reference with `minimap2`
  3. **Evaluate** mapping with `paftools mapeval`

Evaluating MSRs - evaluation pipeline

- Mapping quality *(**mapq**)* is a measure of how confident the aligner is in its read placement. 0 *(worse)* ≤ mapq ≤ 60 *(best)*

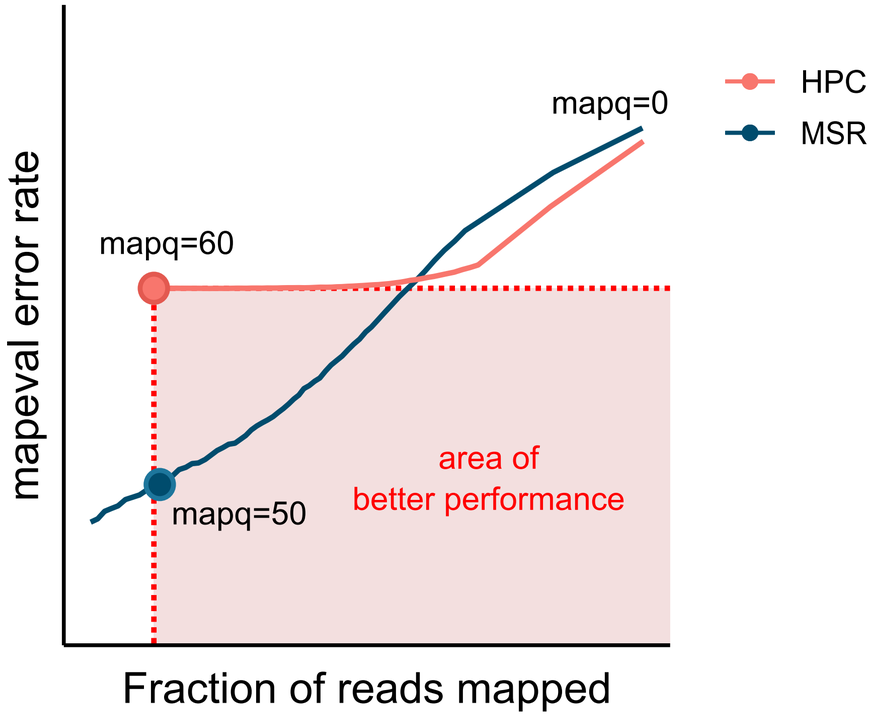
- **`mapeval`** gives results for **mapq thresholds**
  i.e. sets of mapped reads with mapq ≥ than a given value

- **`mapeval`** reports for each threshold:

  - **Number of reads** mapped

  - Mapping **error rate**

Evaluating MSRs - MSR selection

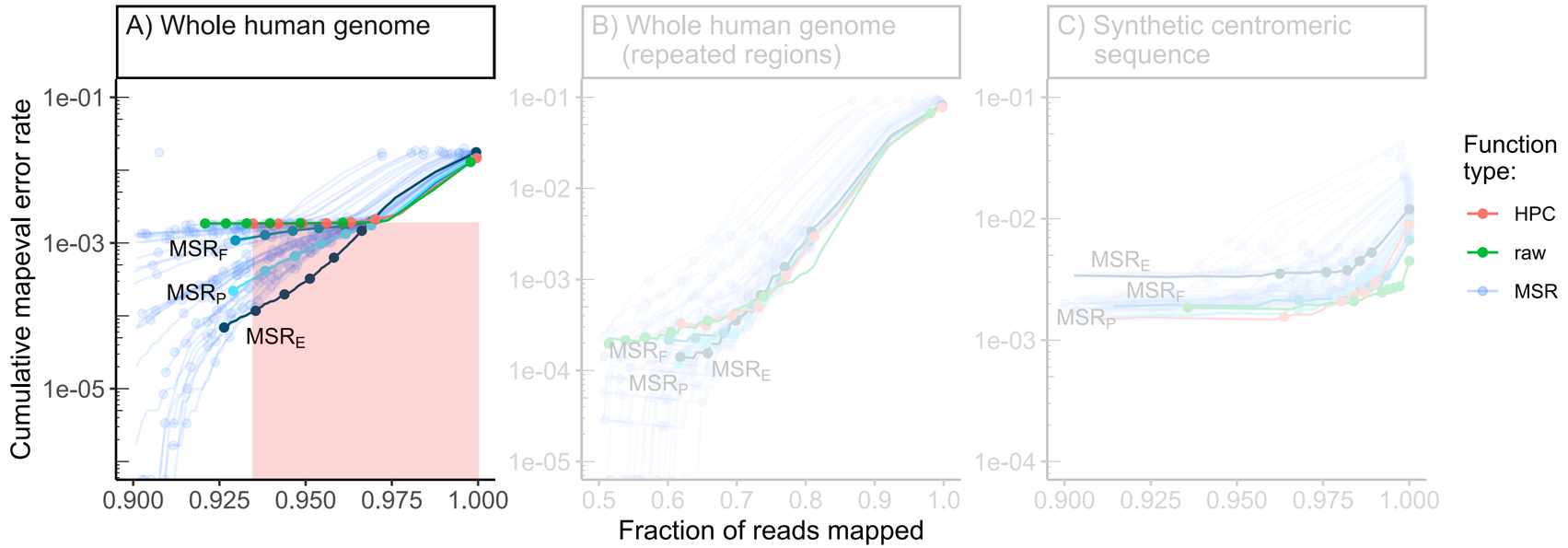


We compare **MSRs** to **HPC at mapq 60**

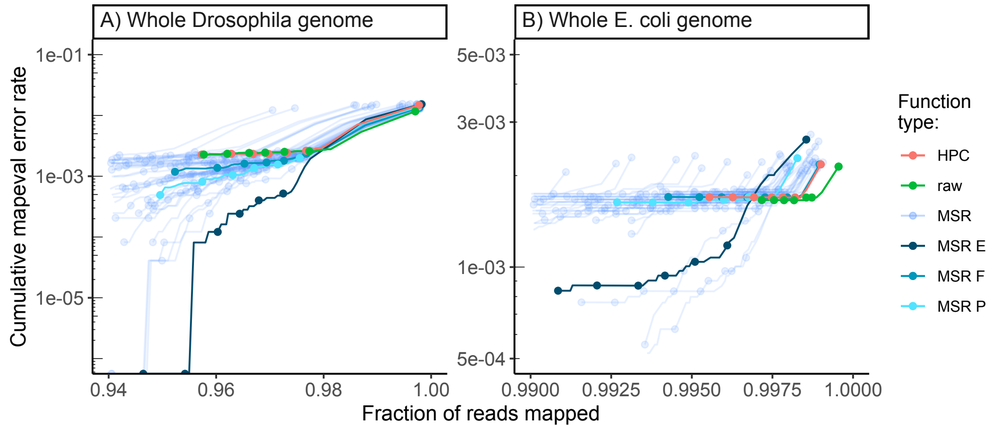
We **select top MSRs** *(error, %mapped, %in shaded area)*

Some results

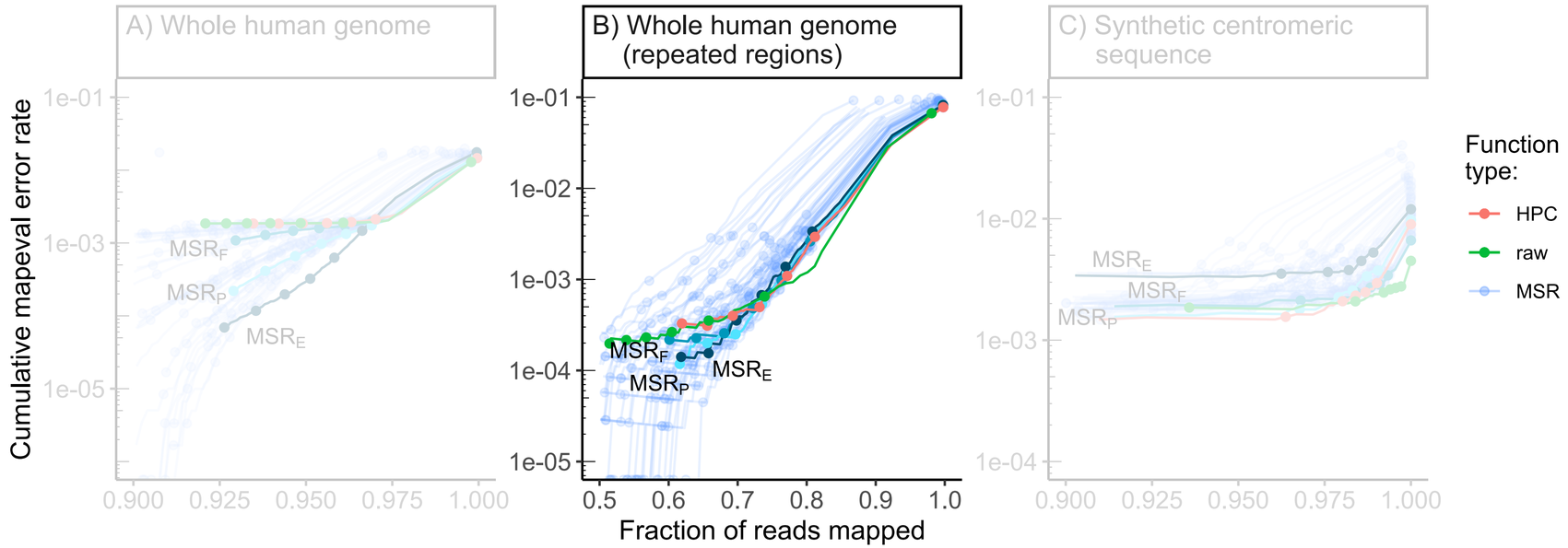
Results - across whole genomes



Many MSRs are **better** than HPC60

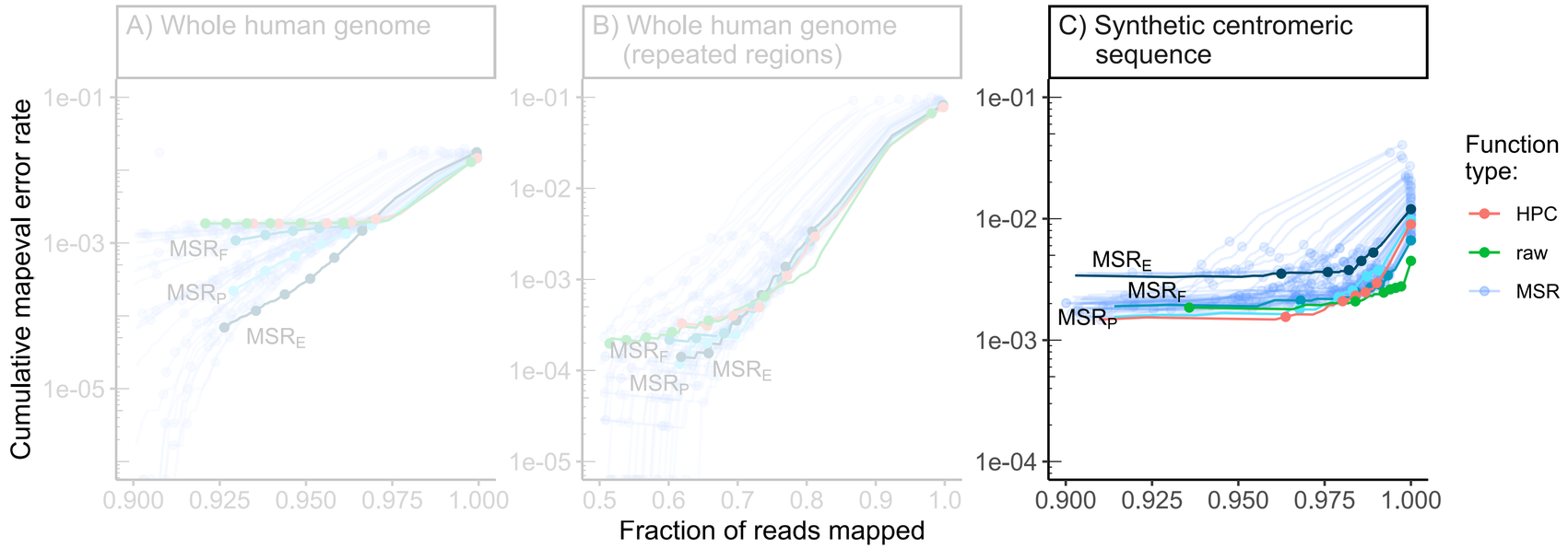


Results - repeated regions



MSRs are still **better** than HPC60
but performance **gap is smaller**

Results - centromeric sequence



Mapping to **centromeres** is **hard**,
best **not to apply any function**

Take home message

- Some **MSRs** are **better than HPC**
- In some cases, the **mapping error rate** goes from 10^{-3} to $\mathbf{10^{-6}}$
- **MSRs** are **easy to implement** in existing aligners,
  i.e. **cheap performance gains**

Where do we go now ?

Perspectives

- MSRs work on simulated data → How do we evaluate on **real datasets** ? *(fraction of mapped reads, mismatch rate, …)*
- Explore **higher-order MSRs** ($N(3) \approx 3 \cdot 10^{21}$ and $N(4) \approx 10^{85}$):
  - **Reduce** the search space
  - **Explore** search space **better**:
    - Define objective function and **optimise**
    - **“Learn”** MSRs (connections, sequence to sequence models, …)

Thank you !

Camila Duitama Gomez

Fransesco Andreace

Riccardo Vicedomini

Mélanie Bernardini Ridel

Luc Blassel

Téo Lemane

Luca Denti

Yoann Dufresne

Rayan Chikhi

Paul Medvedev

SeqBio Team

INSTITUT PASTEUR

PennState Eberly College of Science

PR[AI]RIE PaRis Artificial Intelligence Research InstitutE

SORBONNE UNIVERSITÉ

What is an MSR ?

- An order-ℓ MSR is defined by a function $g : \Sigma^{\ell} \rightarrow \Sigma \cup \{\varepsilon\}$ applied to a sliding window of length ℓ along the sequence to transform
- The output sequence of the MSR is the concatenation of the first $\ell - 1$ characters of the original sequence plus all the outputs of g
- Let x be a string and f an MSR defined by g:
  $f(x) = x[1,\ell - 1] \cdot g(x[1,\ell]) \cdot g(x[2,\ell + 1]) \cdots g(x[|x| - \ell + 1, |x|])$
- By exploring the space of g functions we can explore different sequence transformation functions and see how the impact mapping
- We explore order-2 MSRs in this project

Reducing the search space - RC

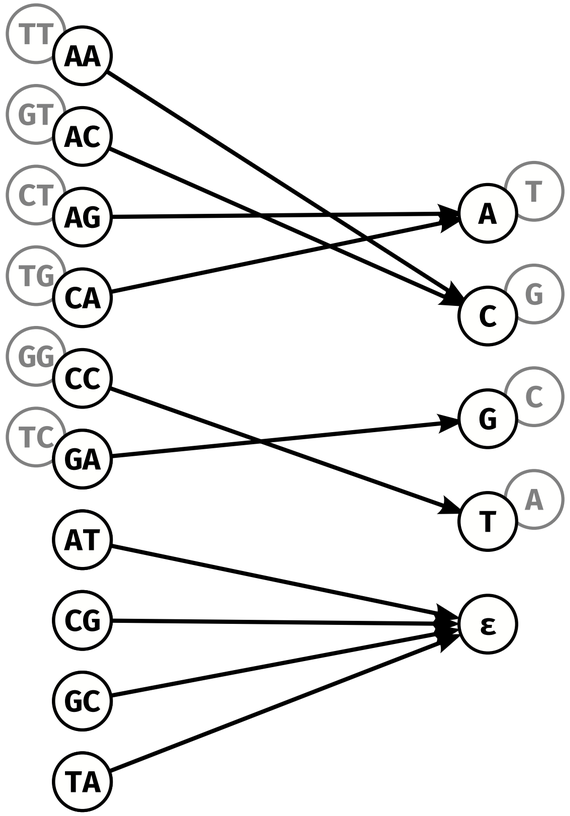
- Since we want to apply MSRs to biological sequences we want it to be **commutative** with reverse complementation *(RC)*.
  i.e. for an MSR f we want $RC(f(x)) = f(RC(x))$

- We define *RC-core-insensitive* MSRs by restricting g. Let x be an ℓ-mer and y its RC; then either $g(x)$ must be the RC of $g(y)$ or $g(x) = g(y) = \varepsilon$

- For order-2 MSRs, there are 16 ℓ-mers, 4 are their own RC (AT,TA,GC,CG) so they must be mapped to ε.
  The remaining 12 ℓ-mers can be grouped into 6 pairs of RCs.

- defining g amounts to choosing an output for each pair. So we have 5^6 RC-core-insensitive MSRs.

Reducing the search space - equivalence



$\rightarrow$

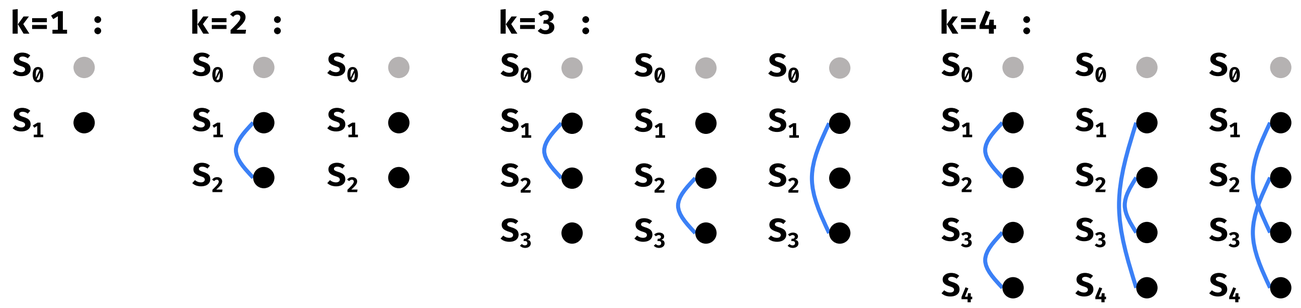
$$S_0 = \{AT, TA, CG, GC\}$$
$$S_1 = \{AG, CA\}$$
$$S_2 = \{CC\}$$
$$S_3 = \{AA, AC\}$$
$$S_4 = \{GA\}$$

and

$$t(1) = A, \;\; t(2) = T, \;\; t(3) = C, \;\; t(4) = G$$

Reducing the search space - equivalence

- For 2 MSRs defined by $S_0, \ldots, S_k, t$ and $S'_0, \ldots, S'_k, t'$ respectively.
- They are equivalent iff:
  - $S_0 = S'_0$
  - there is a permutation π of $[[1; k]]$ such that $\forall i \in [[1; k]]\ S_i = S'_{\pi(i)}$
  - $\forall (i, j) \in [[1; k]]^2$ we have $IsComp(t(i), t(j)) = IsComp(t'(\pi(i)), t'(\pi(j)))$



Number of equivalence classes for fixed partitions with output cardinality k

Reducing the search space - equivalence

- In preliminary tests, when swapping $A \leftrightarrow T$, $G \leftrightarrow C$ or whole G/C pairs with A/T pairs in the MSR outputs, it did not affect performance of the MSRs.
- We can reduce the search space even more by defining MSR equivalence classes.
- Let's consider an MSR defined by g, it's output cardinality k is the number of distinct nucleotides *(i.e. $\neq \varepsilon$)* that can be output by g
- Since g maps all ℓ-mers to $k+1$ values, we can view it as a partition of the set of all ℓ-mers into $k+1$ sets $S_0, \ldots, S_k$ with an injection $t : \{1,\ldots,k\} \to \Sigma$ That assigns an output letter to each set (S_0 is assigned ε)

Reducing the search space - final words

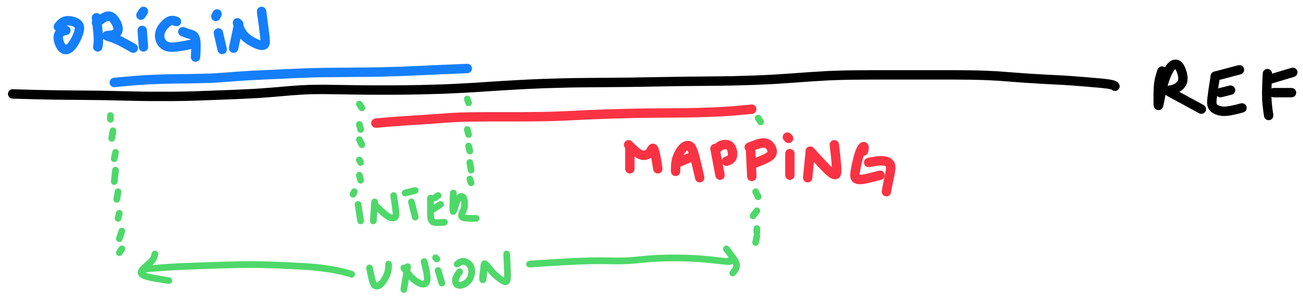
- Let $i(\ell)$ be the number of inputs necessary to define an order-ℓ RC-core-insensitive MSR, and $o(k)$ the number of equivalence classes for a partition with output cardinality k.
- Let $C(\ell, k)$ be the number of ways we can partition $i(\ell)$ ℓ-mers into $k+1$ sets ($S_1, \ldots, S_k \neq \emptyset$).
- Then the total number of MSRs of "interest" is:

$$N(\ell) = \sum_{k=1}^{4} C(\ell, k) \cdot o(k)$$

- For order-2 MSRs we have $N(2) = 2135 \ll 5^6 \ll 5^{16}$

Evaluating MSRs - evaluation pipeline

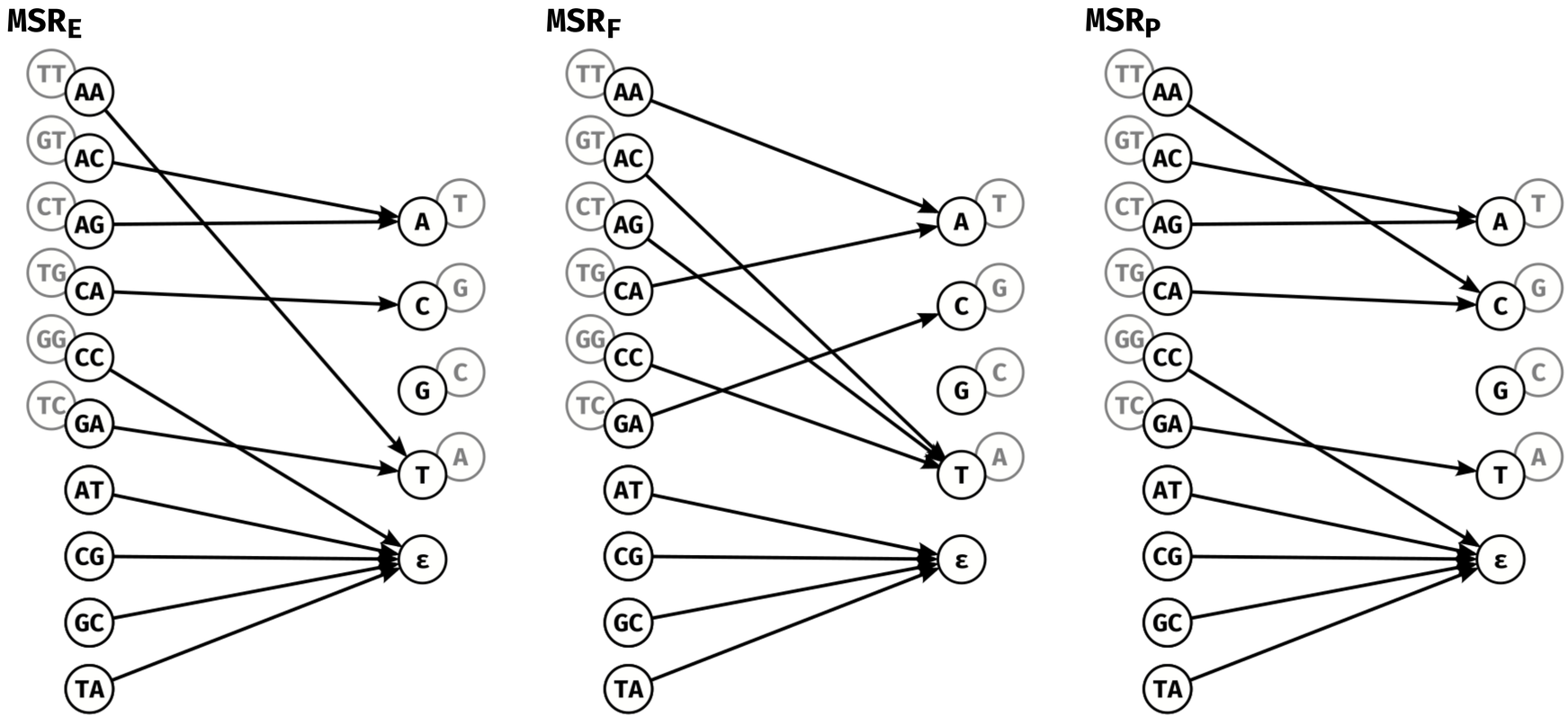
- For each evaluated MSR (and no MSR i.e. *raw*) and each reference we:
  1. Transform the reference and reads with the MSR
  2. Map transformed reads to transformed reference with `minimap2`
  3. Evaluate mapping with `paftools mapeval`
- `mapeval` considers a read correctly mapped if the intersection of the mapped interval and the origin interval on the reference is $\geq 10\,\%$ of the union



Evaluating MSRs - simulation pipeline

- On each reference sequence we simulate ONT long reads using **`nanosim`**
- We aimed for a theoretical coverage of 1.5x (more for centromeric sequence):
    - Whole human genome: $5.5 \cdot 10^5$ reads
    - Whole *Drosophila* genome: $2.6 \cdot 10^4$ reads
    - Synthetic centromere: $1.3 \cdot 10^4$ reads *(i.e. 100x theoretical coverage)*

Results - The top MSRs

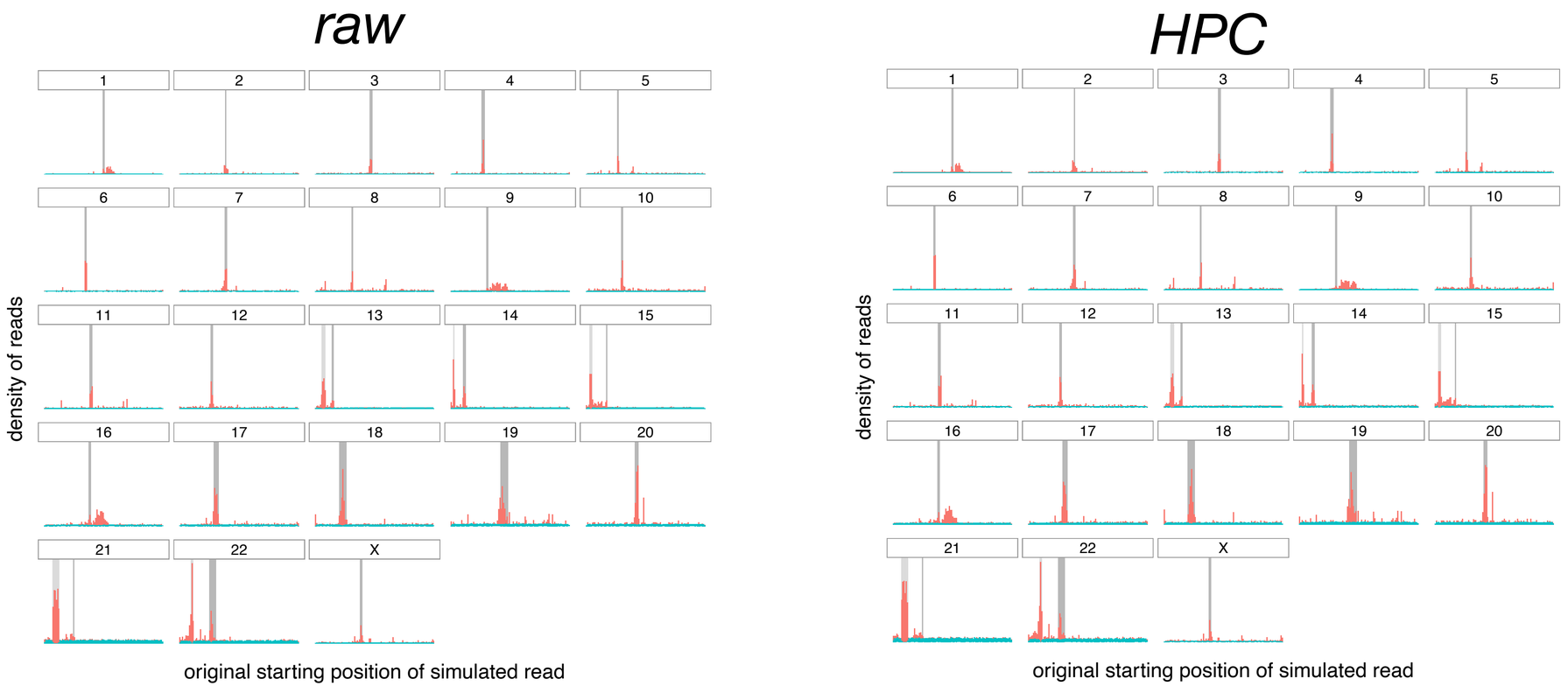


Results - where do MSRs go wrong ?

Origin of incorrectly mapped reads on chromosome

- Incorrectly mapped reads with mapq ≥ 50 for MSRs and 60 for HPC & *raw*
- These reads originate from across the whole genome
- Overall, there is a lower number of these mappings for MSRs (E:549, F:970, P:261) than HPC (1130) or *raw* (1118)

Results - where do MSRs go wrong ?



- We looked at all reads, correctly (blue) and incorrectly (red) mapped
- Most of the mistakes come from centromeres and highly repetitive regions

Why not use PacBio reads ?

- We tried simulating PacBio reads with PBSim2:
- We were getting identical read error profiles with: R94 (ONT) and PC6 (PB) pre-trained models
- We decided not to investigate further
- It might be worth it to test other simulators