# CONSENT: Scalable self-correction of long reads with multiple sequence alignment

**Pierre Morisse** [1], Camille Marchet [2], Antoine Limasset [2], Arnaud Lefebvre [1], Thierry Lecroq [1]

[1] Normandie Univ, UNIROUEN, LITIS, Rouen 76000, France.
[2] Lille Univ, CNRS, CRIStAL, Lille 59000, France.

JOBIM 2019
Nantes
July 5th

# Introduction

## Context

- 2011: Inception of third generation sequencing technologies
- Two main actors: Pacific Biosciences (PacBio) and Oxford Nanopore Technologies (ONT)
- Sequencing of much longer reads, tens of kbps on average
- Expected to solve various problem in the genome assembly field
- But also very noisy (10-30% error rates), most errors being indels

# Introduction

## Error correction

- Correction: efficient way to handle these errors
- Two approaches:
  - Hybrid correction (makes use of complementary short reads)
  - Self-correction (corrects the long reads solely based on the information they contain)

# Introduction

## Self-correction

- Third generation sequencing technologies evolve fast:
  - Error rates greatly decreased, and now reach 10-12% on average
  - Read length is evergrowing, especially with ONT ultra-long reads (up to 1Mbp)
- Error correction is still the first step of many analysis projects
- Self-correction is now much more developped

# Introduction

## Self-correction

State-of-the-art:

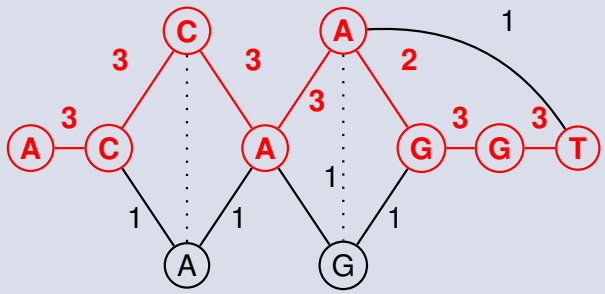
1. Compute overlaps between the LRs
2. Compute consensus from the overlaps

# Introduction

## Pseudo Multiple Sequence Alignment (MSA)

- Build a directed acyclic graph (DAG) to represent the pseudo MSA and compute consensus

```
ACCAAGGT  R1        ACCAAGGT  R1
ACAAGGGT  R2        ACCAA..T  R3
```




## De Bruijn graph

- Divide the alignments into small windows
- Correct the windows independently with DBGs

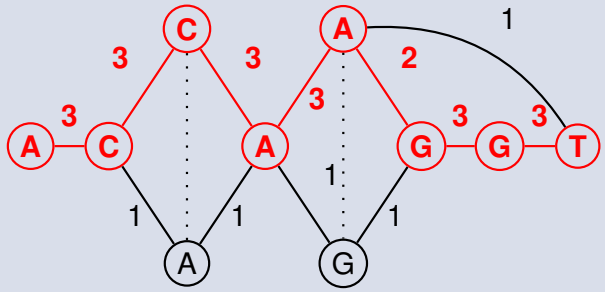
```
.GATCGGG..TAT.TGCCCGTGTTTATGCGTGTG   R1
TGTTCAGGCAAATATG...GAAACAAGGCCTG..   R2

GAT..CGGGTATTGCCCGTGTTTATGCGTG..TG   R1
TATTTCTG..AT.GCGC.TGACTTTTCTTGGCAG   R3
```

# Introduction

## Pseudo Multiple Sequence Alignment (MSA)

- Build a directed acyclic graph (DAG) to represent the pseudo MSA and compute consensus

```
ACCAAGGT  R1        ACCAAGGT  R1
ACAAGGGT  R2        ACCAA..T  R3
```




## De Bruijn graph

- Divide the alignments into small windows
- Correct the windows independently with DBGs

```
.GATCGGG..TAT.TGCCCGTGTTTATGCGTGTG     R1
TGTTCAGGCAAATATG...GAAACAAGGCCTG..     R2

GAT..CGGGTATTGCCCGTGTTTATGCGTG..TG     R1
TATTTCTG..AT.GCGC.TGACTTTTCTTGGCAG     R3
```

# Introduction

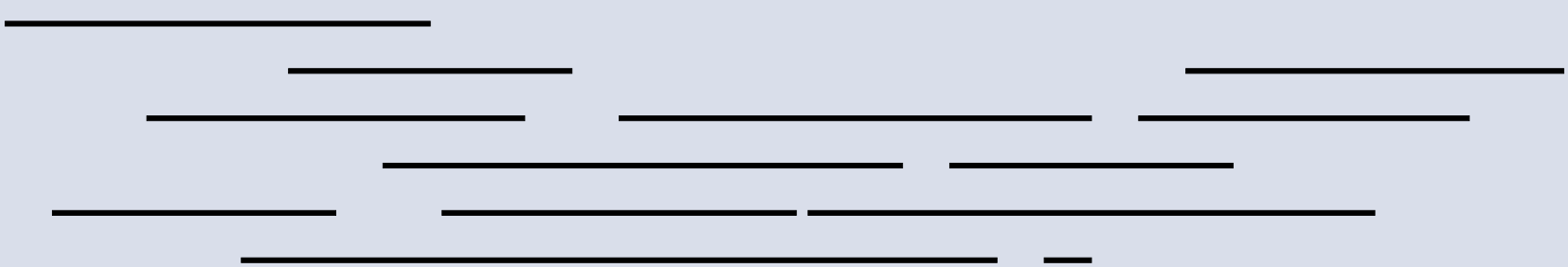
## Contribution

- Major issue: no self-correction tool scales to ONT ultra-long reads
- We introduce CONSENT, a new self-correction method that:
  - Combines the two previous approaches (MSA + DBG)
  - Computes *actual* MSA
  - Compares well to the state-of-the-art, and scales better
  - Is also able to polish contigs

# Pre-treatment

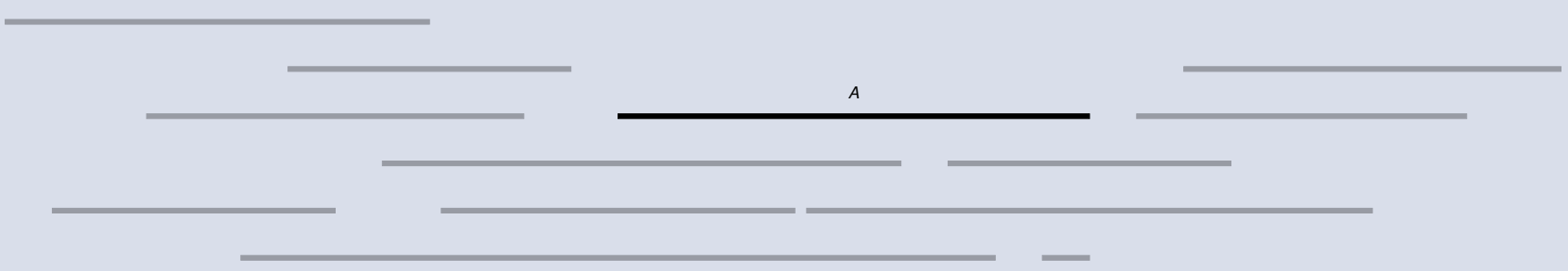
## Overlap the long reads

- Currently with Minimap2 [Li, 2018]
- But not dependent on the aligner



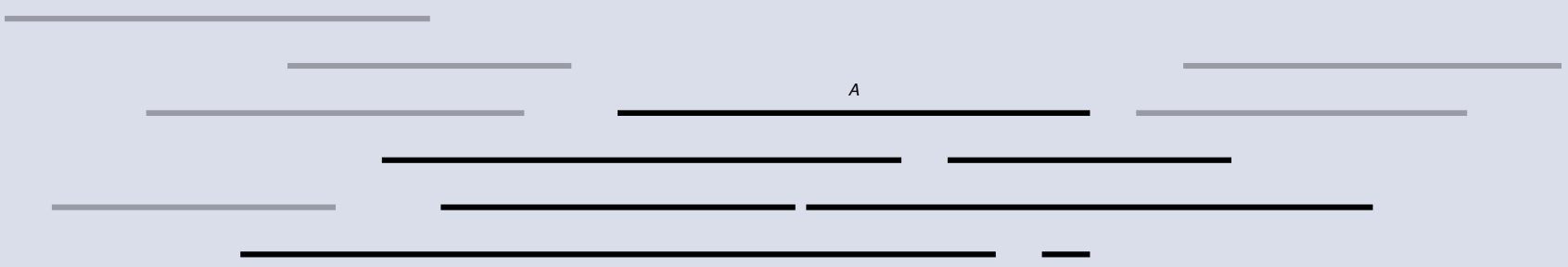
# First step: retrieve alignment piles

## Select a long read to correct



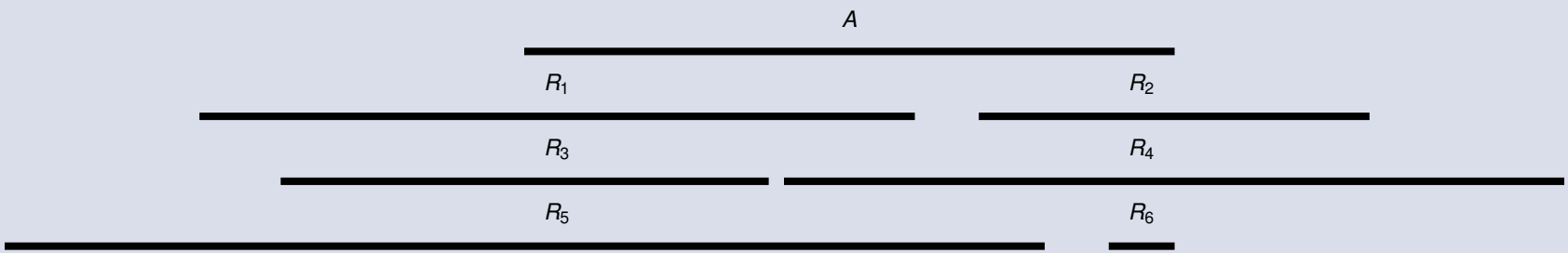
# First step: retrieve alignment piles

## Retrieve overlapping long reads



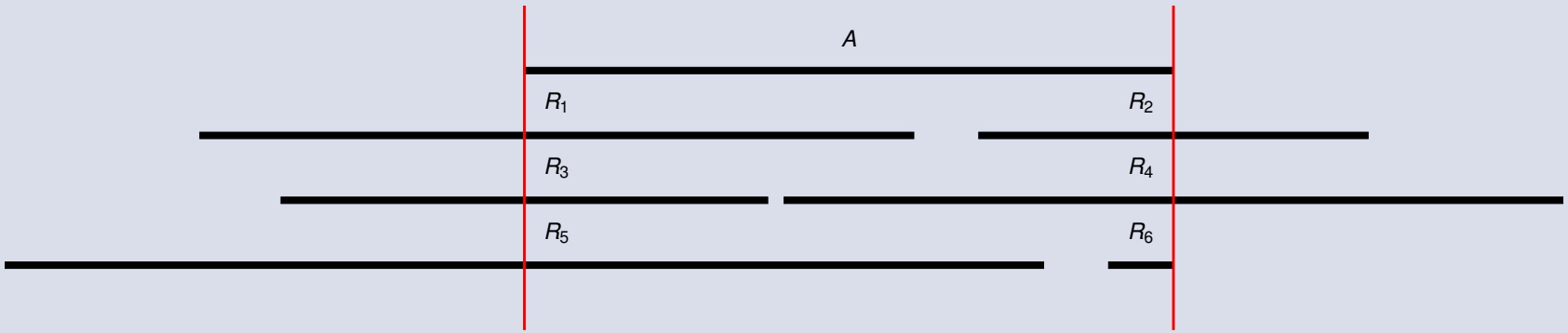
# First step: retrieve alignment piles

## Get the alignment pile



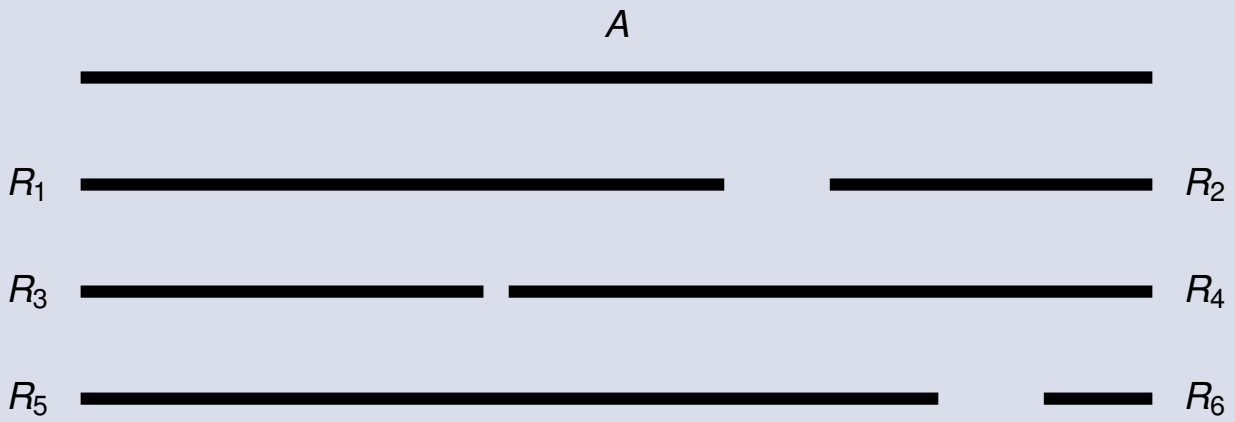
# First step: retrieve alignment piles

## Trim the alignment pile



# First step: retrieve alignment piles

## Trim the alignment pile



# Second step: divide piles into windows

For correction, we will only consider windows that:

- Have a fixed length
- Are supported by at least $c$ reads

**Example**

On the previous example, with $c = 4$:

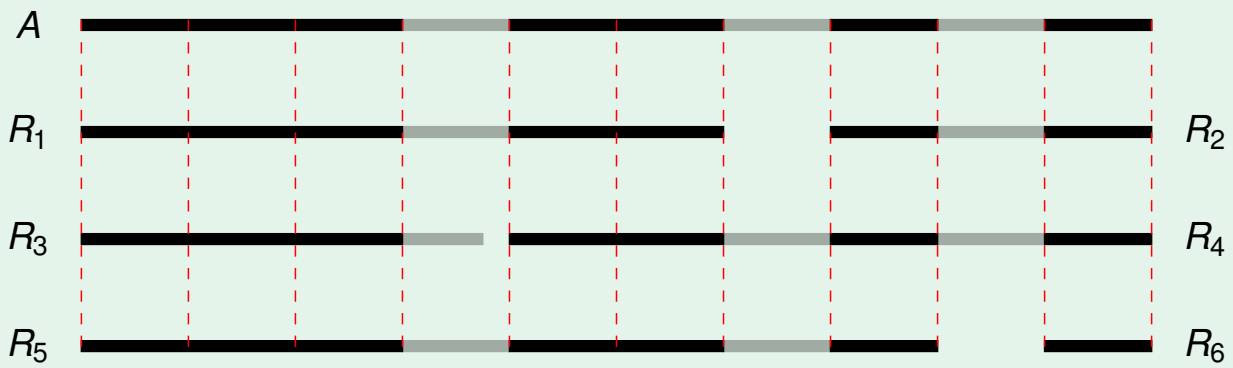
# Second step: divide piles into windows

For correction, we will only consider windows that:

- Have a fixed length
- Are supported by at least $c$ reads

## Example

On the previous example, with $c = 4$:



# Third step: compute consensus of a window

## 2. Compute consensus

- Compute MSA of the sequences
- Compute consensus from the MSA
- Unlike other methods, *actual* MSA is computed
- $\Rightarrow$ POA [Lee et al., 2002]

# Third step: compute consensus of a window

## POA (Partial Order Alignment)

- Multiple sequence alignment strategy based on partial order graphs
- Two interests:
  1. Computes *actual* multiple sequence alignment
  2. Directly builds the DAG representing the multiple sequence alignment

# Third step: compute consensus of a window

## POA (Partial Order Alignment)

- Multiple sequence alignment strategy based on partial order graphs
- Two interests:
  1. Computes *actual* multiple sequence alignment
  2. Directly builds the DAG representing the multiple sequence alignment

# Third step: compute consensus of a window

## POA (Partial Order Alignment)

- Multiple sequence alignment strategy based on partial order graphs
- Two interests:
  1. Computes *actual* multiple sequence alignment
  2. Directly builds the DAG representing the multiple sequence alignment

# Third step: compute consensus of a window

## Segmentation strategy

- In practice, we use windows of a few hundred bases
- POA is time consuming, even on such windows
- We developed a segmentation strategy
- Compute MSA and consensus for smaller sequences $\Rightarrow$ faster

# Third step: compute consensus of a window

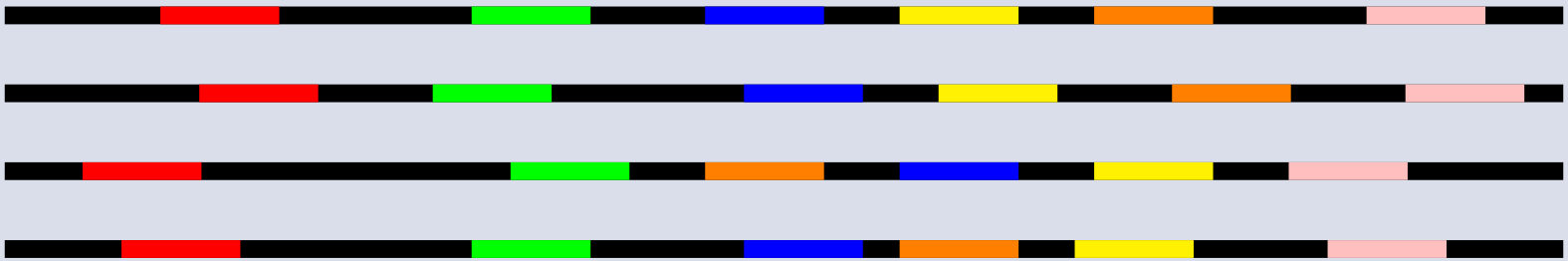
## Segmentation strategy

1. Compute shared anchors between the window's sequences

# Third step: compute consensus of a window

## Segmentation strategy

1. Compute shared anchors between the window's sequences



# Third step: compute consensus of a window

## Segmentation strategy

2. Search for the longest anchors chain such as $\forall A_i, A_{i+1}$:

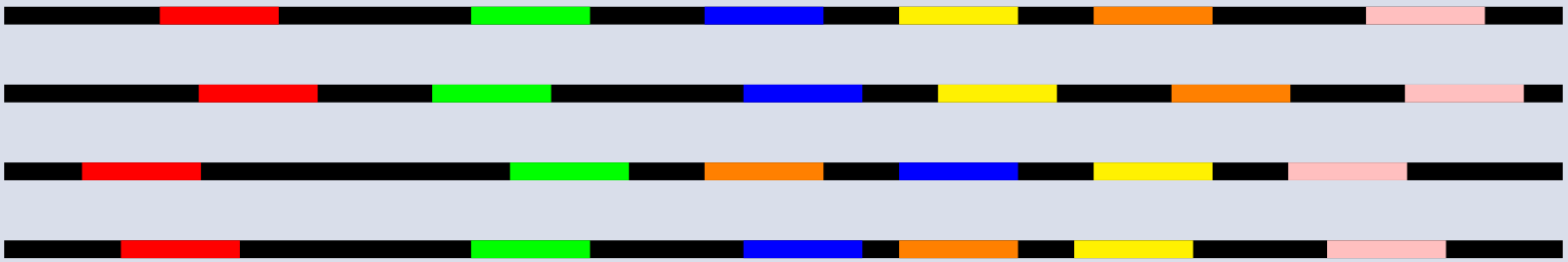
1. $A_i$ is followed by $A_{i+1}$ in at least $N$ sequences
2. $A_{i+1}$ is never followed by $A_i$

# Third step: compute consensus of a window

## Segmentation strategy

2. Search for the longest anchors chain such as $\forall A_i, A_{i+1}$:

1. $A_i$ is followed by $A_{i+1}$ in at least $N$ sequences
2. $A_{i+1}$ is never followed by $A_i$



# Third step: compute consensus of a window

## Segmentation strategy

2. Search for the longest anchors chain such as $\forall A_i, A_{i+1}$:

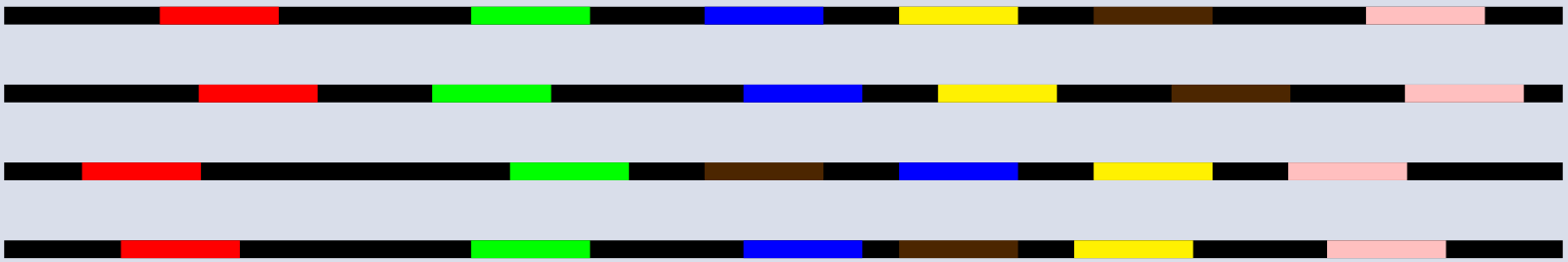
1. $A_i$ is followed by $A_{i+1}$ in at least $N$ sequences
2. $A_{i+1}$ is never followed by $A_i$

# Third step: compute consensus of a window

## Segmentation strategy

2. Search for the longest anchors chain such as $\forall A_i, A_{i+1}$:

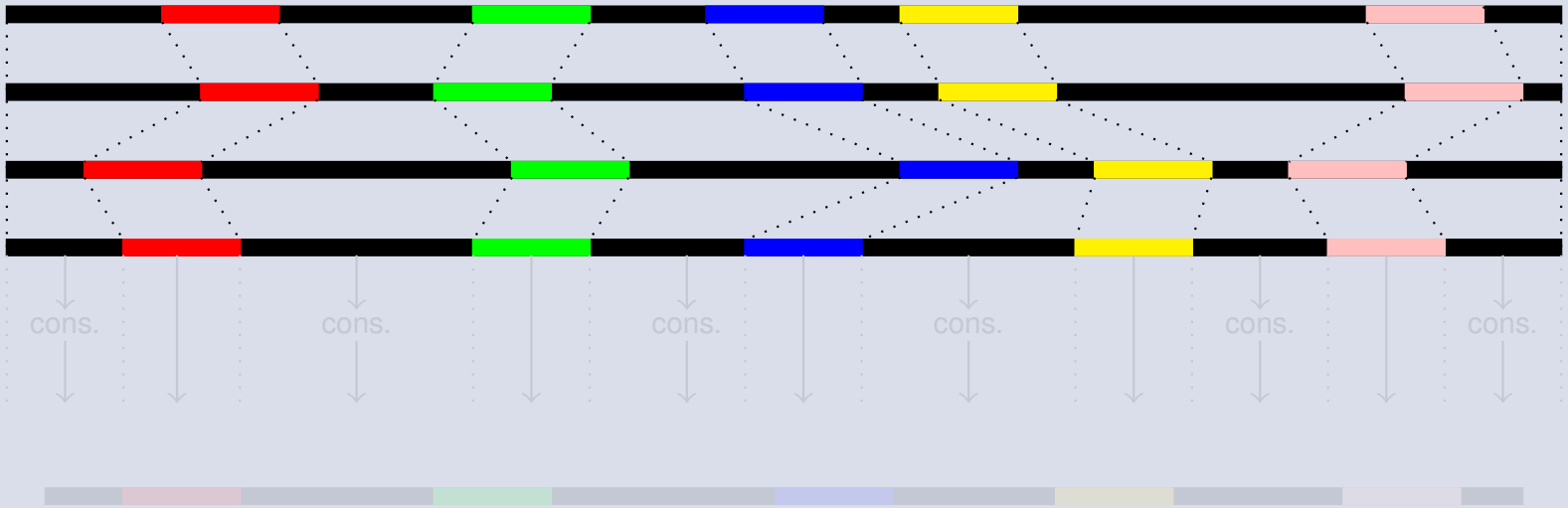
1. $A_i$ is followed by $A_{i+1}$ in at least $N$ sequences
2. $A_{i+1}$ is never followed by $A_i$



# Third step: compute consensus of a window

## Segmentation strategy

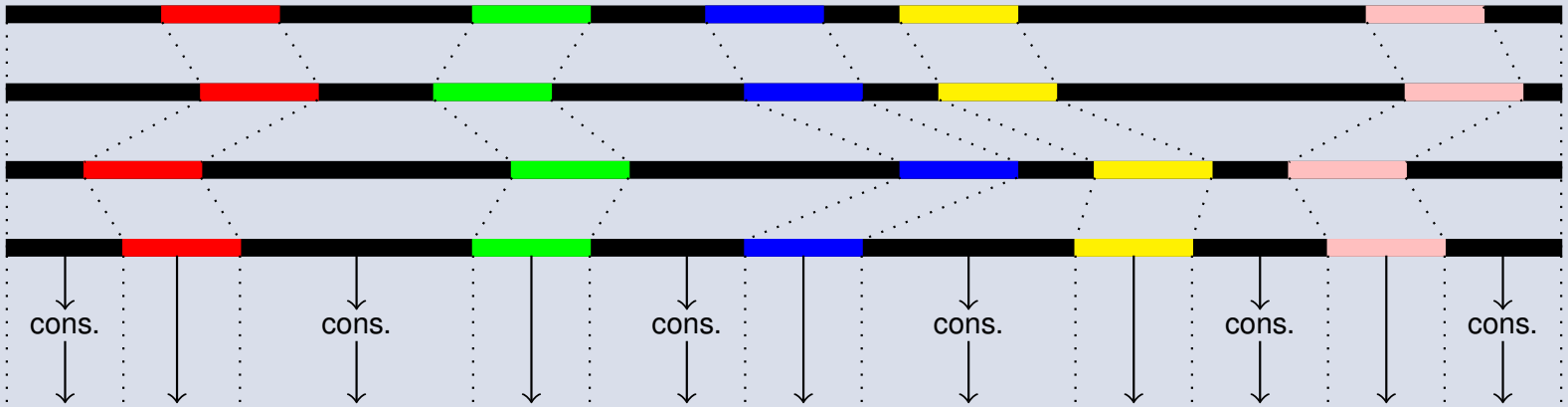
3. Compute MSA / consensus for sequences bordered by anchors



# Third step: compute consensus of a window

## Segmentation strategy

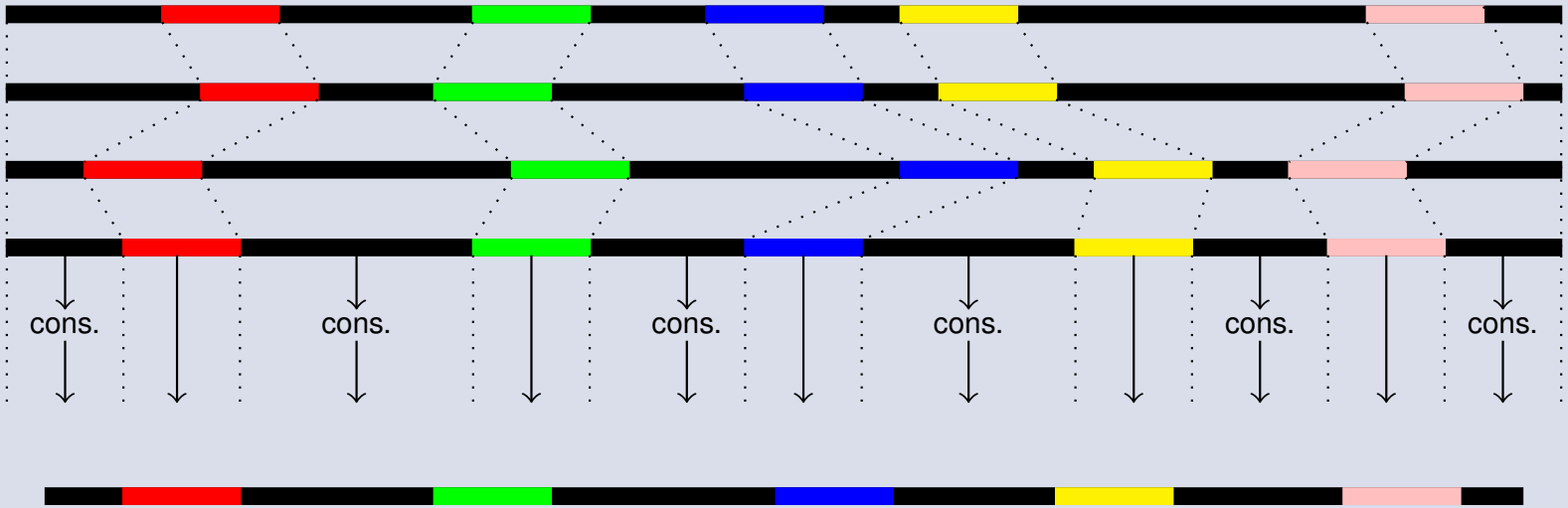
3. Compute MSA / consensus for sequences bordered by anchors



# Third step: compute consensus of a window

## Segmentation strategy

3. Compute MSA / consensus for sequences bordered by anchors



# Fourth step: polish the window's consensus

## Approach

- Consensus ⇒ solid *k*-mers in uppercase, weak *k*-mers in lowercase

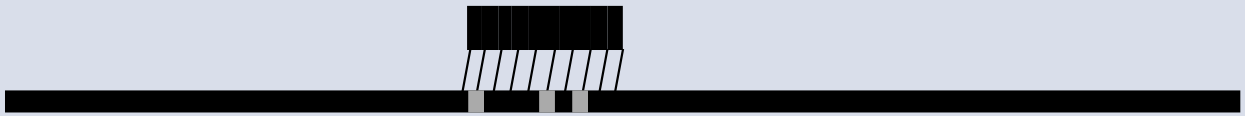
GATCGGGTcatTGCCCGTGTTTATGCGTgtg

- Build a DBG from the window's sequences
- Correct lowercase regions

# Fifth step: anchor the consensus to the read

## By alignment

- Local alignment, around the positions of the window



- Repeat with other windows

# Segmentation strategy validation

## Results

- Simulated PacBio dataset from *E. coli*, 50x, 12% error rate
- Simulated with SimLoRD [Stöcker et al., 2016]

| | Without segmentation | With segmentation |
|---|---|---|
| Throughput | 214,667,382 | 215,693,736 |
| Error rate (%) | 0.0757 | 0.0722 |
| Runtime | 5 h 31min | 7 min |
| Memory (MB) | 750 | 675 |

# Comparison to state-of-the-art

## Compared tools

- Canu correction module [Koren et al., 2017]
- Daccord [Tischler and Myers, 2017]
- FLAS [Bao et al., 2018]
- MECAT [Xiao et al., 2017]

# Comparison to state-of-the-art

## Datasets

Two real Oxford Nanopore datasets :

| Dataset | Number of reads | Average length | Error rate | Coverage |
|---|---|---|---|---|
| *D. melanogaster* | 1,327,569 | 6,828 | 14.57 | 63x |
| *H. sapiens, chr1*[1] | 1,075,867 | 6,744 | 17.60 | 29x |

[1] containts ultra-long reads

# Comparison to state-of-the-art

## Alignment assessment

| Dataset | Corrector | Number of reads | Throughput (Mbp) | N50 (bp) | Alignment identity (%) | Genome coverage (%) | Runtime | Memory (MB) |
|---|---|---|---|---|---|---|---|---|
| *D. melanogaster* | Original | 1,327,569 | 9,064 | 11,853 | 85.43 | 98.47 | N/A | N/A |
| | Canu | 829,965 | 6,993 | **12,694** | 95.20 | 97.89 | 14 h 04 min | **10,295** |
| | Daccord | – | – | – | – | – | – | – |
| | FLAS | 855,275 | 7,866 | 11,742 | 94.99 | 98.09 | 10 h 18 min | 18,820 |
| | MECAT | 849,704 | 7,288 | 11,676 | 96.52 | 97.34 | **1 h 54 min** | 13,443 |
| | CONSENT | **1,065,621** | **8,178** | 12,297 | **96.72** | **98.20** | 38 h | 51,361 |
| *H. sapiens* | Original | 1,075,867 | 7,256 | 10,568 | 82.40 | 92.46 | N/A | N/A |
| | Canu[1] | – | – | – | – | – | – | – |
| | Daccord[1] | – | – | – | – | – | – | – |
| | FLAS[1] | 670,708 | 5,695 | 10,198 | 91.00 | 92.37 | 4 h 57 min | 14,957 |
| | MECAT[1] | 667,532 | 5,479 | 10,343 | 91.69 | 91.44 | **1 h 53 min** | **11,075** |
| | CONSENT | **869,462** | **6,349** | **10,839** | **93.00** | **92.40** | 8 h 30 min | 45,869 |

[1] ultra-long reads were filtered out

# Comparison to state-of-the-art

## Assembly assessment

| Dataset | Corrector | Number of contigs | Aligned contigs (%) | NGA50 | NGA75 | Genome coverage (%) |
|---|---|---|---|---|---|---|
| *D. melanogaster* | Original | 423 | 96.45 | 864,011 | 159,590 | 83.1900 |
| | Canu | 410 | 92.93 | **2,757,690** | **822,577** | 92.1034 |
| | Daccord | – | – | – | – | – |
| | FLAS | 374 | 96.52 | 1,123,351 | 364,884 | 92.1105 |
| | MECAT | **308** | **99.68** | 1,425,566 | 478,877 | 89.5839 |
| | CONSENT | 455 | 98.46 | 1,666,202 | 470,720 | **92.5688** |
| *H. sapiens* | Original | 201 | 93.53 | 1,025,355 | 247,806 | 77.5700 |
| | Canu[1] | – | – | – | – | – |
| | Daccord[1] | – | – | – | – | – |
| | FLAS[1] | 237 | **100** | 1,698,601 | 289,968 | 88.4068 |
| | MECAT[1] | 249 | 99.20 | 1,672,967 | 424,788 | 88.7002 |
| | CONSENT | **182** | 97.25 | **2,663,412** | **439,178** | **88.9587** |

[1] ultra-long reads were filtered out

# Additional feature

## Contigs polishing

- Allows to correct assemblies generated from raw reads
- Straightforward: compute overlaps between contigs and reads
- Rest of the pipeline remains the same
- First self-correction tool to propose such a feature

# Contigs polishing

## Experiments

- Simulated PacBio datasets from *E. coli*, *S. cerevisiae* and *C. elegans*
- Simulated with SimLoRD, 60x coverage, 12% error rate
- We compare CONSENT to RACON [Nagarajan et al., 2017]

| Dataset | Method | Contigs | Aligned contigs | NGA50 | Genome coverage | Errors / 100 kbp | Runtime (CPU sec) | Memory (MB) |
|---|---|---|---|---|---|---|---|---|
| *E. coli* | Original | 1 | 1 | - | 0.89 | 10,721 | N/A | N/A |
| | RACON | 1 | 1 | 4,663,914 | 99.90 | 499 | 5,597 | 628 |
| | CONSENT | 1 | 1 | 4,637,588 | 99.90 | **78** | **334** | 4,192 |
| *S. cerevisiae* | Original | 29 | 29 | - | 0.87 | 10,694 | N/A | N/A |
| | RACON | 29 | 29 | 539,433 | 96.09 | 637 | 14,931 | 1,673 |
| | CONSENT | 29 | 29 | 535,665 | 96.12 | **208** | **1,616** | 9,232 |
| *C. elegans* | Original | 47 | 46 | - | 0.95 | 10,611 | N/A | N/A |
| | RACON | 47 | 47 | 5,073,456 | 99.71 | 819 | 136,325 | 14,264 |
| | CONSENT | 47 | 47 | 3,737,577 | 99.57 | **330** | **30,907** | 32,144 |

# Take-home messages

- **CONSENT:**
  - Self-correction of long reads
  - Compares well to the state-of-the-art
  - Only method able to scale to ONT ultra-long reads
  - Also performs contigs polishing
- **Specificities:**
  - Combines two state-of-the-art approaches: MSA + DBG
  - Computes actual MSA
  - Uses a segmentation strategy to quickly compute MSA
- **Availability:**
  - Software: `https://github.com/morispi/CONSENT`
  - Preprint on bioRxiv: `https://doi.org/10.1101/546630`

# Future works

- Optimize the parameters (size of the windows, of the *k*-mers, etc)
- Reduce runtime: deeply covered windows
- Segmentation strategy seems promising $\Rightarrow$ apply it to a greater scale

Bao, E., Xie, F., Song, C., and Song, D. (2018).
HALS : Fast and High Throughput Algorithm for.
*RECOMB-SEQ 2018*, pages 1–7.

Koren, S., Walenz, B. P., Berlin, K., Miller, J. R., Bergman, N. H., and Phillippy, A. M. (2017).
Canu: scalable and accurate long-read assembly via adaptive k -mer weighting and repeat separation.
*Genome Research*, 27:722–736.

Lee, C., Grasso, C., and Sharlow, M. F. (2002).
Multiple sequence alignment using partial order graphs.
*Bioinformatics*, 18(3):452–464.

Li, H. (2018).
Minimap2: pairwise alignment for nucleotide sequences.
*Bioinformatics*, 34(18):3094–3100.

Nagarajan, N., Mile, Š., Vaser, R., and Sovic, I. (2017).

Fast and accurate de novo genome assembly from long uncorrected reads.
*Genome Research*, pages 1–10.

Stöcker, B. K., Köster, J., and Rahmann, S. (2016).
SimLoRD: Simulation of Long Read Data.
In *Bioinformatics*, volume 32, pages 2704–2706.

Tischler, G. and Myers, E. W. (2017).
Non Hybrid Long Read Consensus Using Local De Bruijn Graph Assembly.
*bioRxiv*.

Xiao, C. L., Chen, Y., Xie, S. Q., Chen, K. N., Wang, Y., Han, Y., Luo, F., and Xie, Z. (2017).
MECAT: Fast mapping, error correction, and de novo assembly for single-molecule sequencing reads.
*Nature Methods*, 14(11):1072–1074.