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.

RECOMB-SEQ
03 May 2019
Washington D.C.

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, up to 1 Mbp (ONT ultra-long reads)
- Expected to solve various problem in the genome assembly field

Introduction

Context

- Long reads (LR) are very noisy (10-30% error rate)
- Display complex error profiles (errors are mostly indels)
- Efficiently handling these error rates is mandatory
- Can be done via correction: hybrid or self

Introduction

Hybrid correction

- First efficient approach for LR error correction
- Makes use of complementary short reads (SR) data
- Different approaches: Alignment of SRs to the LRs, use of a De Bruijn graph (DBG), ...
- Particularly useful on old sequencing experiments (very high error rates)

Introduction

Self-correction

- Corrects the LRs solely based on the information they contain
- Third generation sequencing technologies evolve fast
- Error rates of the LRs now reach 10-12% on average
- Error correction is still the first step of many analysis projects
- Self-correction is now a viable alternative with such error rates

Introduction

Self-correction

State-of-the-art:

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

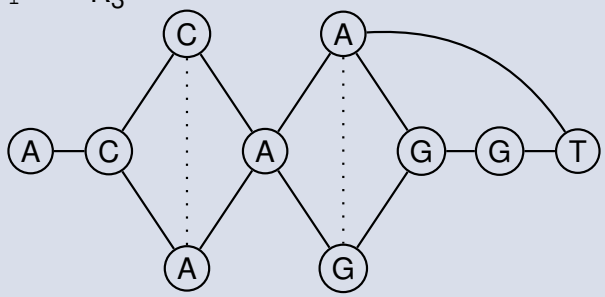
Introduction

Pseudo Multiple Sequence Alignment (MSA)

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

```
ACCAAGGT    R1
ACAAGGGT    R2

ACCAAGGT    R1
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
```

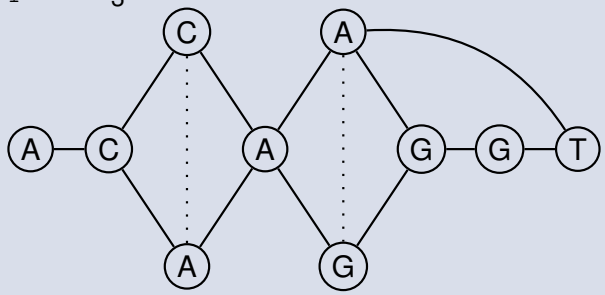
Introduction

Pseudo Multiple Sequence Alignment (MSA)

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

```
ACCAAGGT    R1
ACAAGGGT    R2

ACCAAGGT    R1
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
```

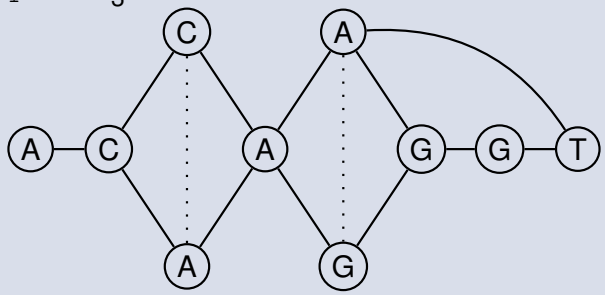
Introduction

Pseudo Multiple Sequence Alignment (MSA)

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

```
ACCAAGGT    R1
ACAAGGGT    R2

ACCAAGGT    R1
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

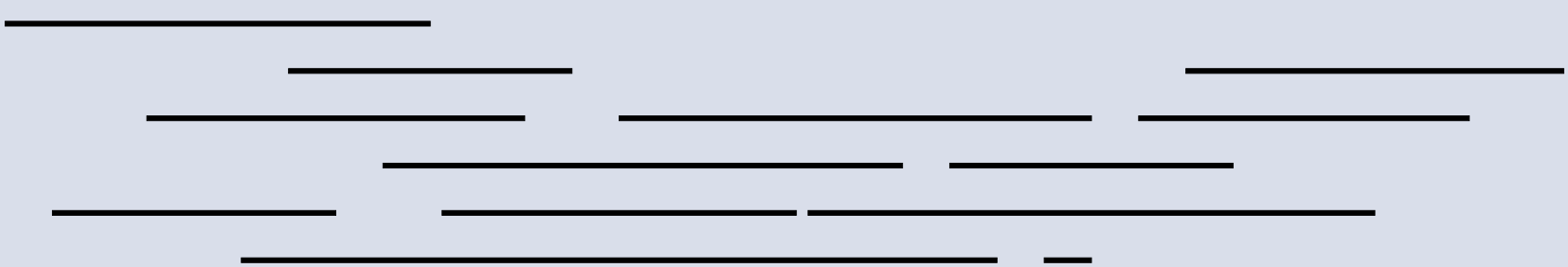
Contribution

- 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

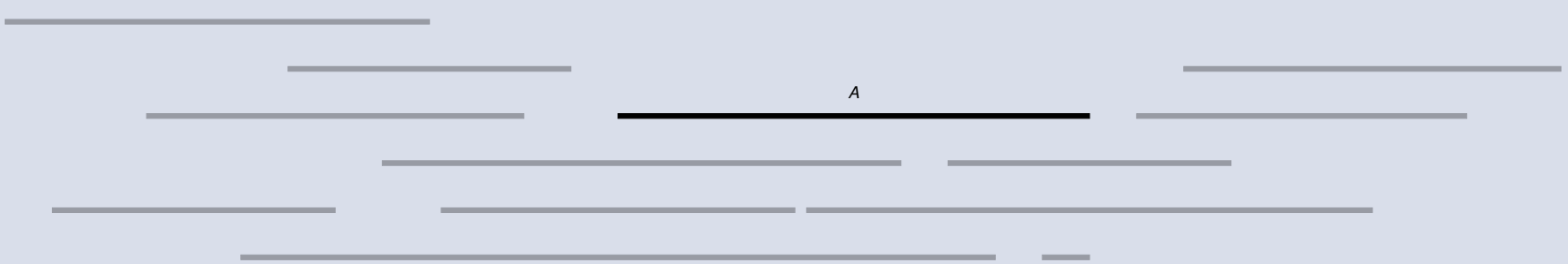
Overlap the long reads

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



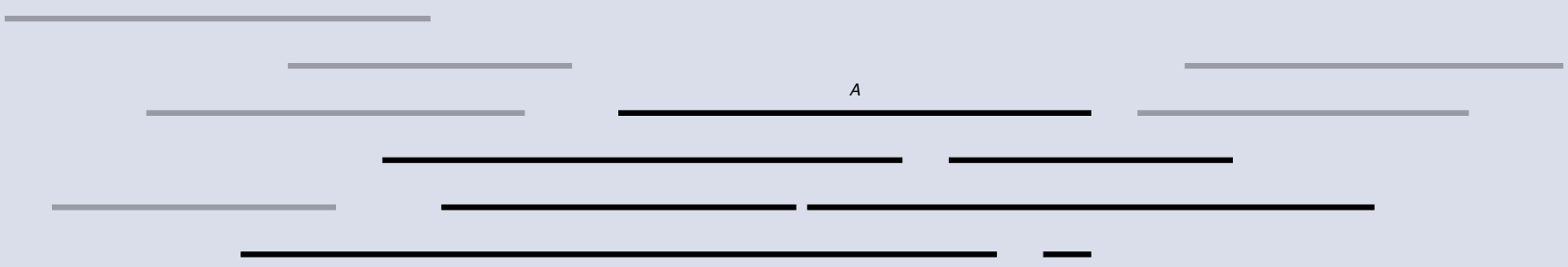
First step: Retrieve alignment piles

Select a long read to correct



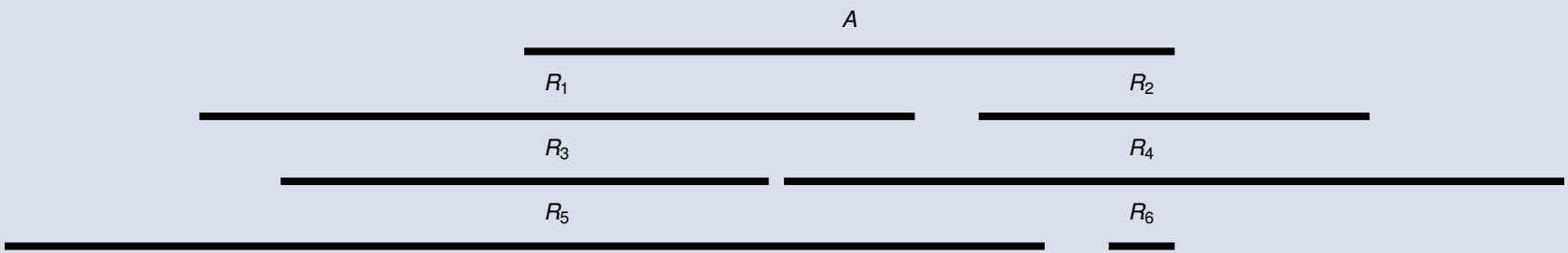
First step: Retrieve alignment piles

Retrieve overlapping long reads



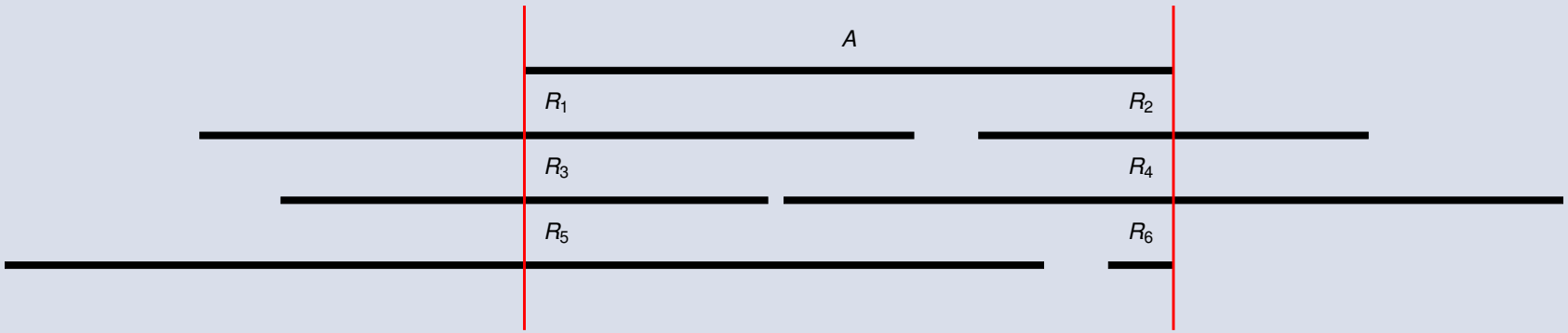
First step: Retrieve alignment piles

Get the alignment pile



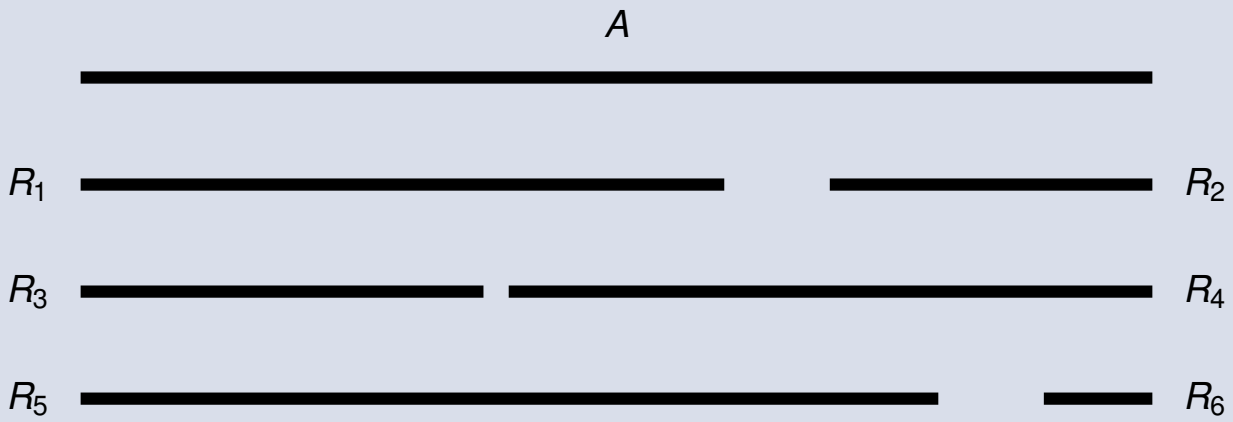
First step: Retrieve alignment piles

Trim the alignment pile



First step: Retrieve alignment piles

Trim the alignment pile



Second step: Divide piles into windows

Definition

A window $w = (beg, end)$ is a "factor" of an alignment pile

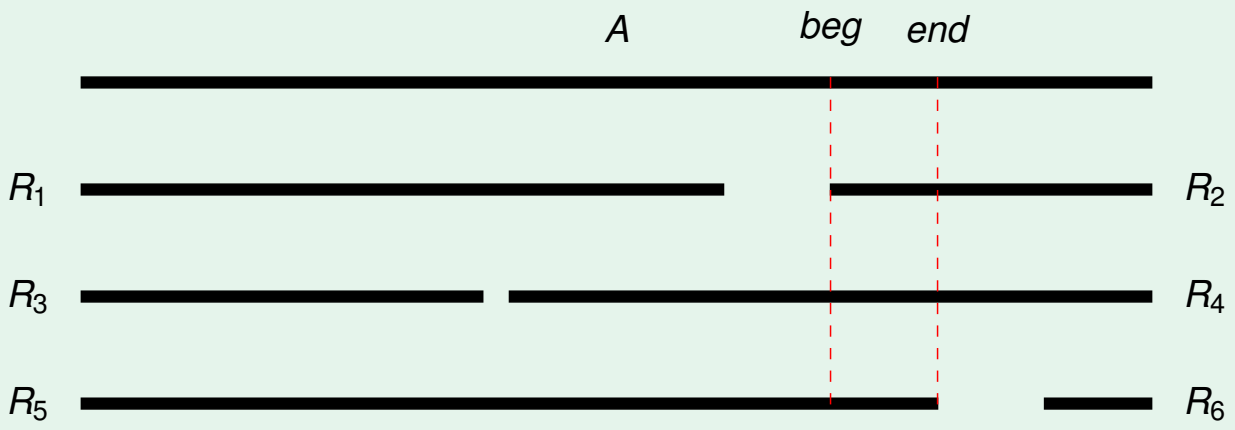
Example

Second step: Divide piles into windows

Definition

A window $w = (beg, end)$ is a "factor" of an alignment pile

Example



Second step: Divide piles into windows

For correction, we will only consider windows $w = (beg, end)$ such as:

- $end - beg + 1 = l$
- $\forall i, beg \leq i \leq end$, i is covered by at least c reads

Example

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

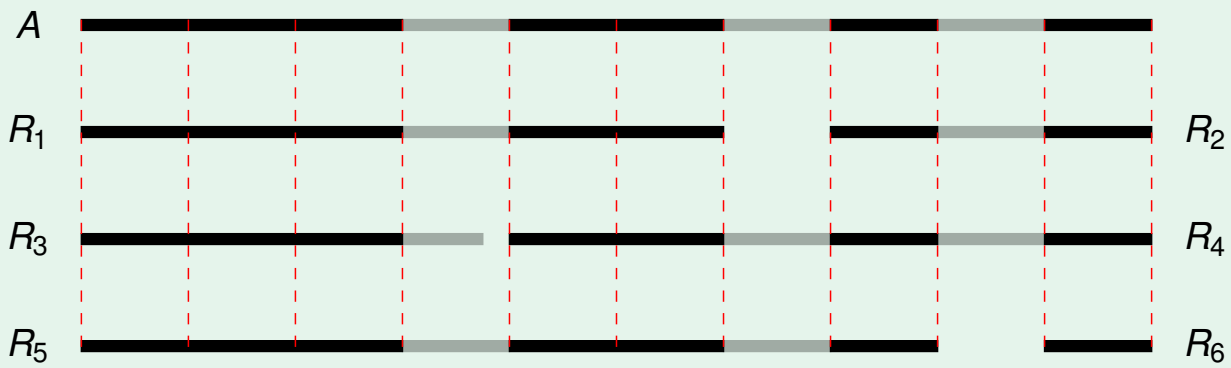
Second step: Divide piles into windows

For correction, we will only consider windows $w = (beg, end)$ such as:

- $end - beg + 1 = l$
- $\forall i, beg \leq i \leq end$, i is covered 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 these 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

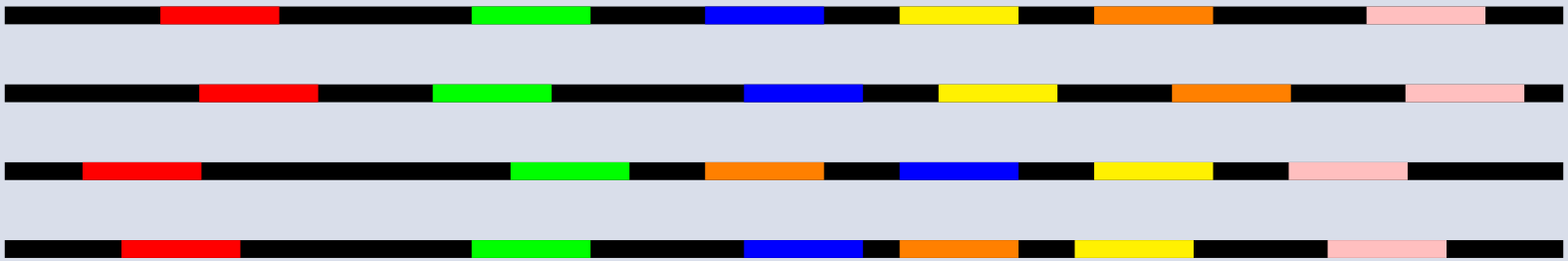
Segmentation strategy

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 $\forall A_i, A_{i+1}$:

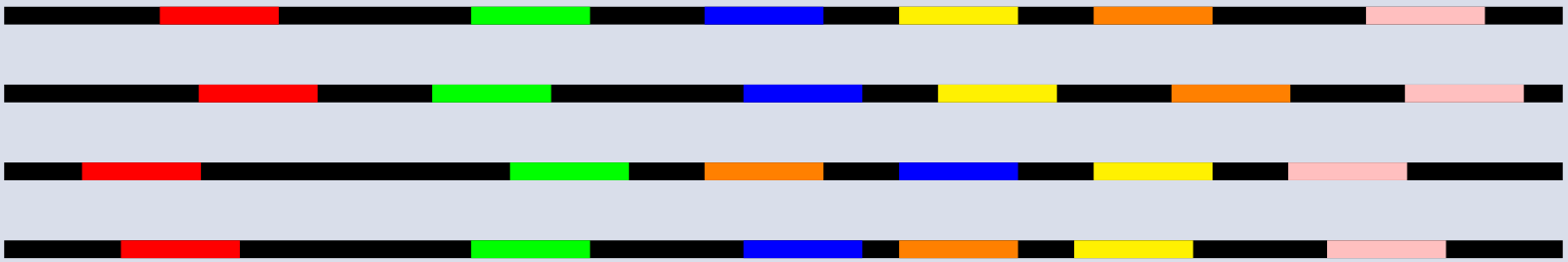
1. A_i is followed by A_{i+1} in at least N sequences
2. A_{i+1} is never followed by A_i

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

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 $\forall A_i, A_{i+1}$:

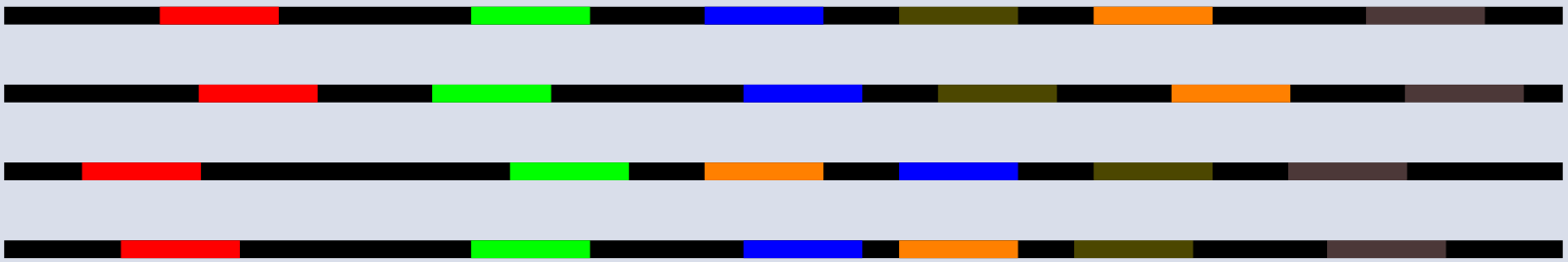
1. A_i is followed by A_{i+1} in at least N sequences
2. A_{i+1} is never followed by A_i

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

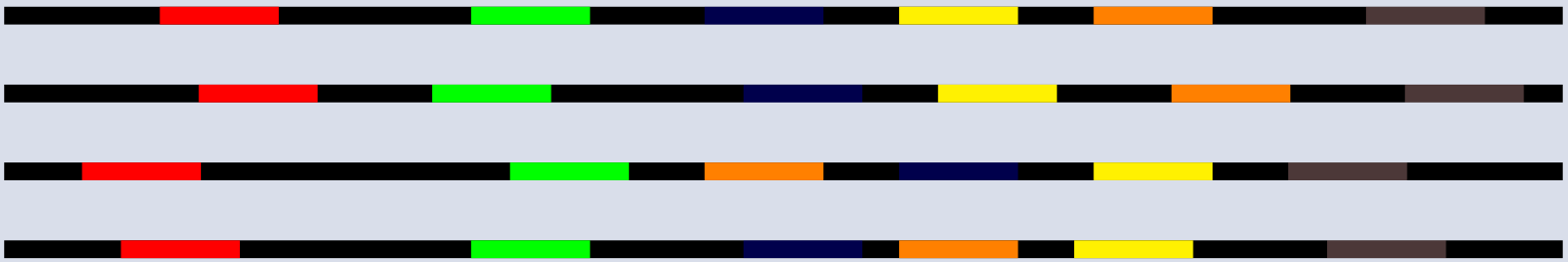


Third step: Compute consensus of a window

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

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

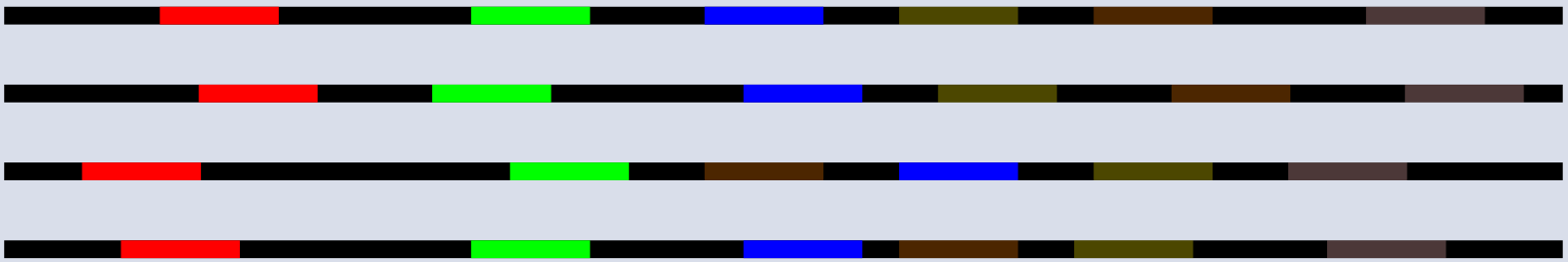
1. A_i is followed by A_{i+1} in at least N sequences
2. A_{i+1} is never followed by A_i

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

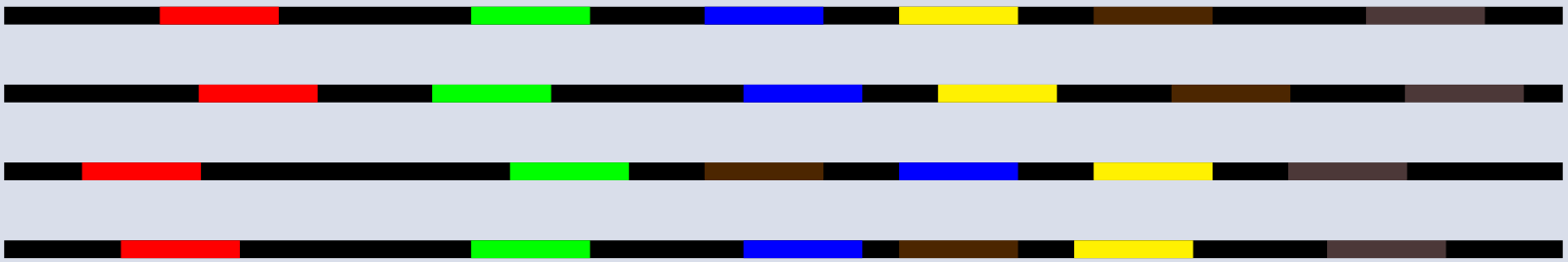


Third step: Compute consensus of a window

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

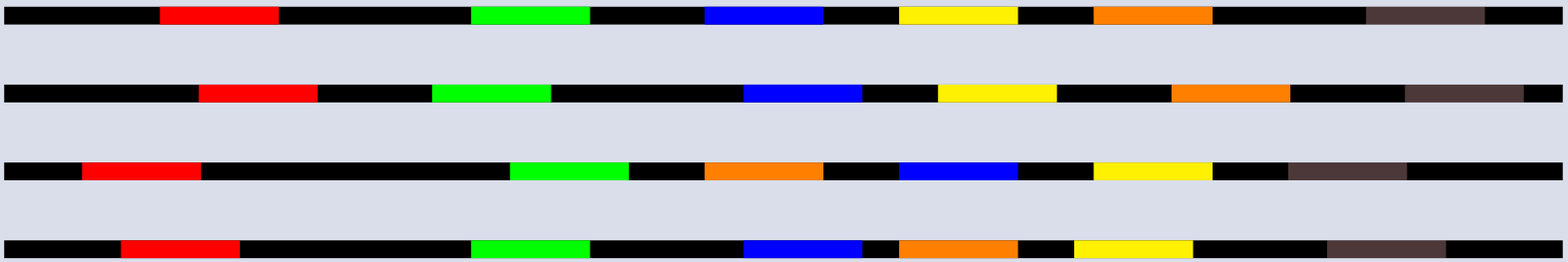


Third step: Compute consensus of a window

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

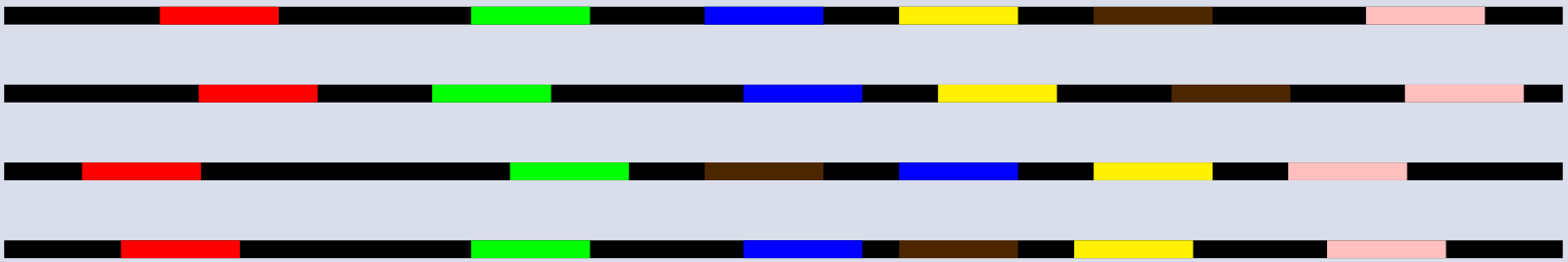


Third step: Compute consensus of a window

Segmentation strategy

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

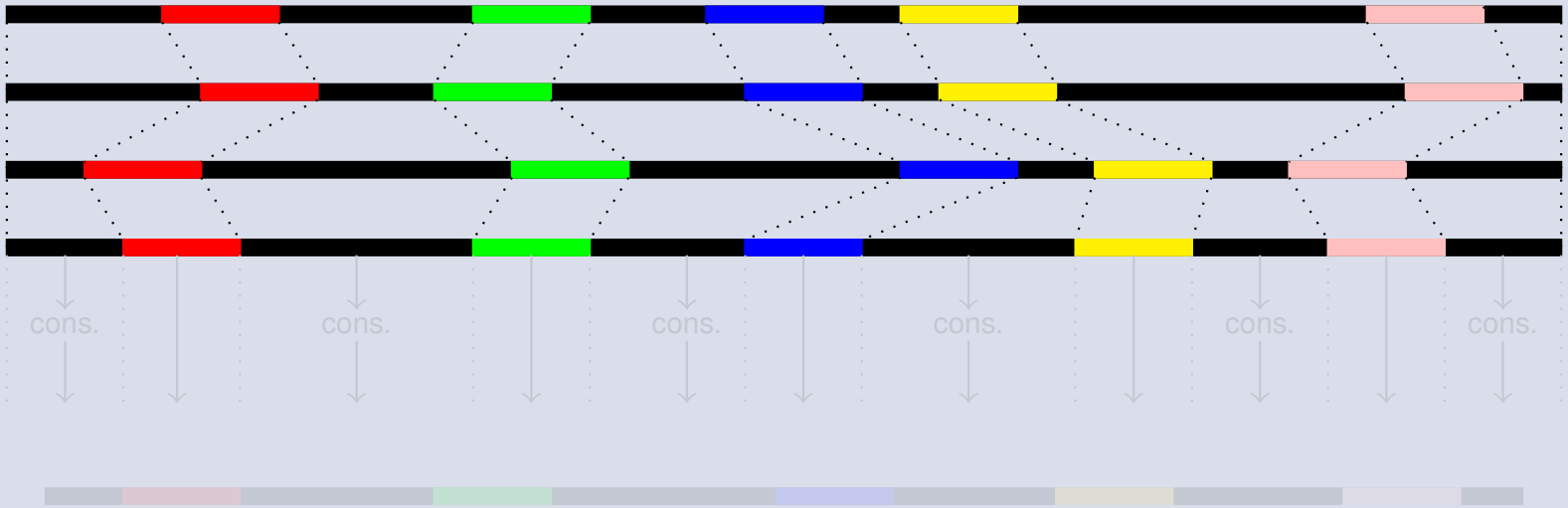
1. A_i is followed by A_{i+1} in at least N sequences
2. A_{i+1} is never followed by A_i



Third step: Compute consensus of a window

Segmentation strategy

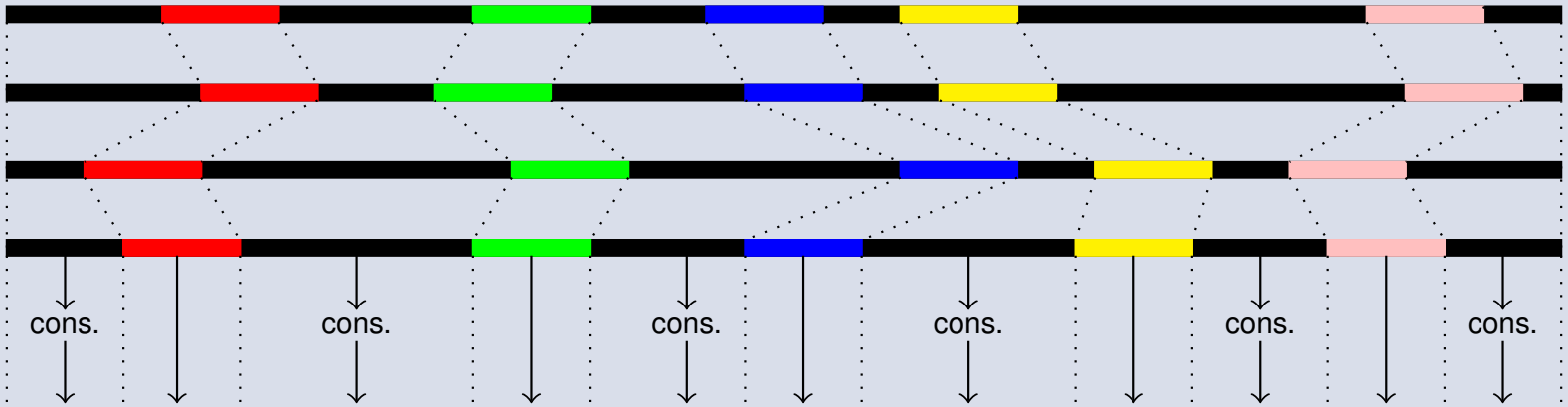
3. Compute MSA / consensus for sequences bordered by anchors



Third step: Compute consensus of a window

Segmentation strategy

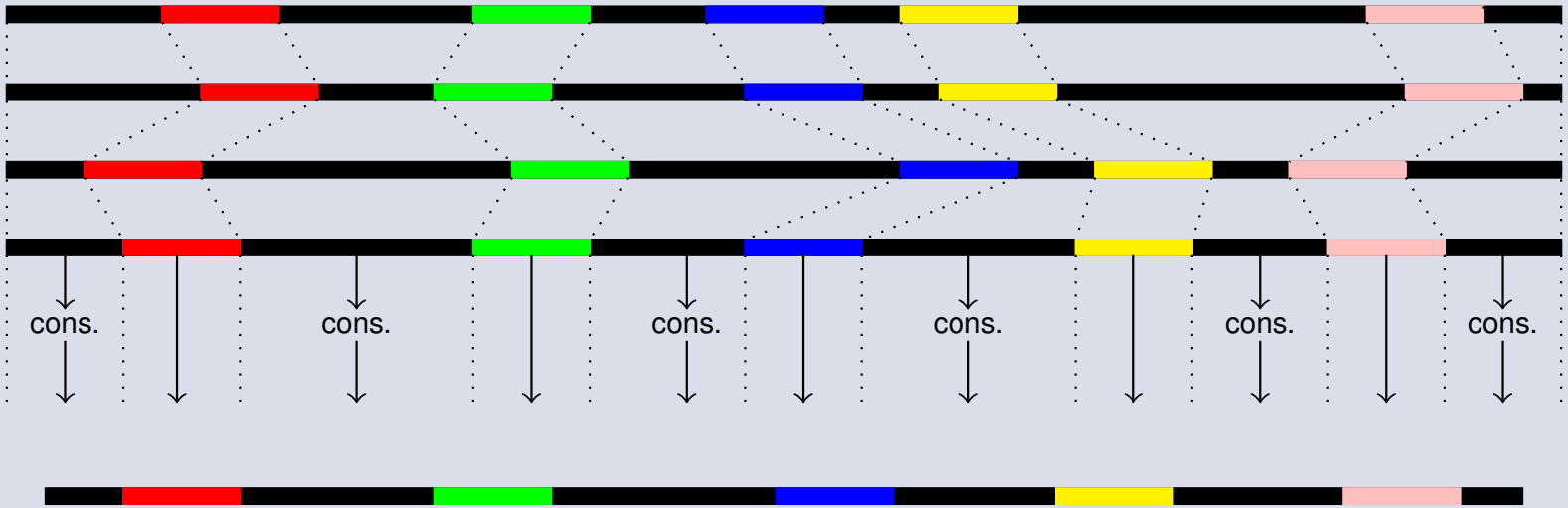
3. Compute MSA / consensus for sequences bordered by anchors



Third step: Compute consensus of a window

Segmentation strategy

3. Compute MSA / consensus for sequences bordered by anchors



Fourth step: Polish the consensus

Approach

- Build a DBG from the window's sequences
- Consensus ⇒ solid *k*-mers in uppercase, weak *k*-mers in lowercase

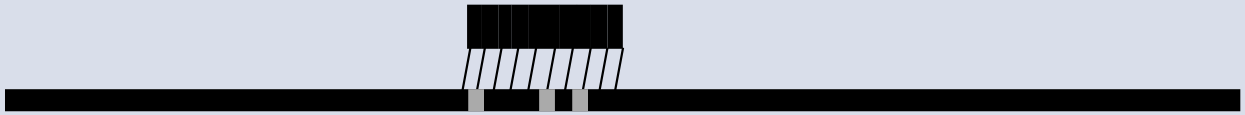
GATCGGGTcatTGCCCGTGTTTATGCGTgtg

- Correct lowercase regions
- Bordered regions ⇒ Traverse the graph to find a path between solid, anchor *k*-mers
- Extremities ⇒ Traverse the graph as much as possible

Fifth step: Anchor the consensus to the read

By alignment

- Get the polished consensus
- Locally align it to the LR, 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]
- Statistics obtained with LRCstats [La et al., 2017]

	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

Datasets

Dataset	Number of reads	Average length	Error rate	Coverage
Simulated Pacific Biosciences data				
E. coli	33,918	8,211	12.28	60x
S. cerevisiae	90,397	8,204	12.29	60x
C. elegans	732,832	8,220	12.28	60x
Real Oxford Nanopore data				
D. melanogaster	1,327,569	6,828	14.57	63x
H. sapiens, chr1	1,075,867	6,744	17.60	29x

Comparison to state-of-the-art

Compared tools

- Canu [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

Simulated data

Dataset	Corrector	Throughput (Mbp)	Error rate (%)	Deletions (%)	Insertions (%)	Substitutions (%)	Total Runtime	Total Memory (MB)
E. coli	Original	279	12.2788	2.6437	8.7919	0.8432	N/A	N/A
	Canu	219	0.5211	0.1390	0.4045	0.0243	24 min	3,674
	Daccord	**261**	**0.0175**	**0.0026**	**0.0062**	0.0103	54 min	18,450
	FLAS	260	0.1039	0.0907	0.0220	0.0010	38 min	2,428
	MECAT	233	0.1011	0.0896	0.0203	**0.0008**	**5 min**	**2,387**
	CONSENT	259	0.0590	0,0368	0.0241	0.0037	36 min	4,849
S. cerevisiae	Original	742	12.2886	2.6484	8.7963	0.8439	N/A	N/A
	Canu	600	0.5615	0.1518	0.4309	0.0292	1 h 11 min	**3,710**
	Daccord	**696**	**0.0305**	**0.0055**	**0.0180**	0.0100	2 h 26 min	32,190
	FLAS	690	0.1430	0.1215	0.0319	0.0031	1 h 30 min	4,984
	MECAT	617	0.1365	0.1189	0.0286	**0.0020**	**16 min**	4,954
	CONSENT	690	0.1418	0.0735	0.0650	0,0166	1 h 46 min	11,325
C. elegans	Original	6,024	12.2825	2.6457	8.7937	0.8432	N/A	N/A
	Canu	5,112	0.7934	0.2881	0.4107	0.0371	9 h 30 min	7,050
	Daccord	–	–	–	–	–	–	–
	FLAS	5,584	0.3997	0.4604	0.1008	0.0224	10 h 45 min	13,682
	MECAT	4,938	**0.2675**	**0.2535**	**0.0402**	**0.0022**	**2 h 43 min**	10,563
	CONSENT	**5,604**	0.5730	0.3282	0.2273	0.0504	27 h 04 min	32,284

Comparison to state-of-the-art

Real data

Dataset	Corrector	Number of reads	Throughput (Mbp)	N50 (bp)	Aligned reads (%)	Alignment identity (%)	Genome coverage (%)	Total Runtime	Total Memory (MB)
D. melanogaster	Original	1,327,569	9,064	11,853	85.52	85.43	98.47	N/A	N/A
	Canu	829,965	6,993	**12,694**	98.05	95.20	97.89	14 h 04 min	10,295
	Daccord	–	–	–	–	–	–	–	–
	FLAS	855,275	7,866	11,742	95.65	94.99	98.09	10 h 18 min	18,820
	MECAT	849,704	7,288	11,676	**99.87**	96.52	97.34	**1 h 54 min**	**13,443**
	CONSENT	**1,065,621**	**8,178**	12,297	99.26	**96.72**	**98.20**	38 h	51,361
H. sapiens	Original	1,075,867	7,256	10,568	88.24	82.40	92.46	N/A	N/A
	Canu[1]	–	–	–	–	–	–	–	–
	Daccord[1]	–	–	–	–	–	–	–	–
	FLAS[1]	670,708	5,695	10,198	99.06	91.00	92.37	4 h 57 min	14,957
	MECAT[1]	667,532	5,479	10,343	**99.95**	91.69	91.44	**1 h 53 min**	**11,075**
	CONSENT	**869,462**	**6,349**	**10,839**	99.59	**93.00**	**92.40**	8 h 30 min	45,869

[1] ultra-long reads were filtered out

Comparison to state-of-the-art

Contigs polishing

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 k, etc)
- Reduce runtime: Deeply covered windows
- Segmentation strategy seems promising $\Rightarrow$ Apply it to a greater scale

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.

RECOMB-SEQ
03 May 2019
Washington D.C.

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.

La, S., Haghshenas, E., and Chauve, C. (2017).
LRCstats, a tool for evaluating long reads correction methods.
Bioinformatics, 33:3652–3654.

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.

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.