ELECTOR:
Evaluator for long reads correction methods

Camille Marchet [1†], Pierre Morisse [2†], Lolita Lecompte [3], Antoine Limasset [1], Arnaud Lefebvre [2], Thierry Lecroq [2] and Pierre Peterlongo [3]
[1]Univ. Lille, CNRS, Inria, UMR 9189 - CRIStAL.
[2]Normandie Univ, UNIROUEN, LITIS, Rouen 76000, France.
[3]Univ Rennes, CNRS, Inria, IRISA - UMR 6074, F-35000 Rennes, France.
[†]these authors contributed equally to this work

Abstract

Motivation: In the last few years, third generation sequencing error rates have been capped above 5%, including many insertions and deletions. Thereby, an increasing number of long reads correction methods have been proposed to reduce the noise in these sequences. Whether hybrid or self-correction methods, there exist multiple approaches to correct long reads data. As the quality of the error correction has huge impacts on downstream processes, developing methods allowing to evaluate error correction tools with precise and reliable statistics is therefore a crucial need. Since error correction is often a resource bottleneck in long reads pipelines. A key feature of assessment methods is therefore to be efficient, in order to allow the fast comparison of different tools.
Results: We propose ELECTOR, a reliable and efficient tool to evaluate long reads correction, that enables the evaluation of hybrid and self-correction methods. Our tool provides a complete and relevant set of metrics to assess the read quality improvement after correction and scales to large datasets. ELECTOR is directly compatible with a wide range of state-of-the-art error correction tools, using whether simulated or real long reads. We show that ELECTOR displays a wider range of metrics than the state-of-the-art tool, LRCstats, and additionally importantly decreases the runtime needed for assessment on most datasets.
Availability: ELECTOR is available at `https://github.com/kamimrcht/ELECTOR`.
Contact: `camille.marchet@univ-lille.fr` or `pierre.morisse2@univ-rouen.fr`

1 Introduction

1.1 Motivation

Pacific Biosciences (PB) and Oxford Nanopore Technologies (ONT) long reads, despite their high error rates and complex error profiles, were rapidly adopted for various applications [1]. These reads display high error rates (from 9% to as much as 30%, according to technologies and libraries), that largely surpass those of Illumina reads. Moreover, contrary to Illumina, where the majority of errors are substitutions, long reads mainly contain insertions and deletions (indels) errors (ONT reads are more deletion-prone whereas PB reads contain more insertions). This combination of issues requires novel and specific algorithmic developments. To this extent, dozens of error correction methods directly targeting these long reads emerged in the last five years. A first range of error correction tools, called hybrid correctors, uses both short and long reads to perform error correction, relying on the important coverage and low error rate of the short reads in order to enhance long reads sequences. A second group of methods, called self-correctors, intends to correct long reads with the sole information contained in their sequences (see [2] for a review of correctors). Both paradigms

include quite diverse algorithmic solutions, which makes it difficult to globally compare the correction results (in terms of throughput, quality and performances) without a proper benchmark.

An increasing number of projects adopts long reads to benefit from the long range information they provide, such as assembly or structural variant calling [1]. Given the long reads' error rates, the first step of many applications is error correction. However, this stage can be a time bottleneck [1].

In addition, the quality of the error correction has considerable impacts on downstream processes. Hence, it is interesting to know beforehand which corrector is best suited for a particular experimental design (coverage, read type, or genome, for instance). Developing methods allowing to evaluate error correction tools with precise and reliable statistics is therefore a crucial need.

Such evaluation methods should allow to perform reproducible and comprehensive benchmarks, and thus to efficiently identify which error correction method is best suited for a given case. They must be usable on datasets of various complexity (from bacteria to eukaryotes) in order to reproduce a wide variety of the scenarios that can be encountered. They also should be fast and lightweight, and should not be purple orders of magnitude more resource and time consuming than the actual correction methods they assess. This aspect is particularly critical when correction evaluators also stand in the perspective of new correction methods developments. They can help providing accurate and rapid comparisons with state-of-the-art correctors. For developers as well as users, correction evaluators should describe with precision the correction method's behavior (i.e. quantity of corrected bases, introduced errors or read break ups, and throughput), in order to identify its potential pitfalls.

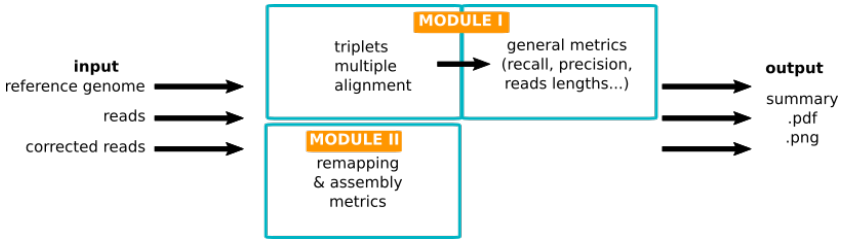

Figure 1: Overview of ELECTOR pipeline. Input are the sequences at the different stages: without errors (from the reference genome), with errors (simulated or real reads) and corrected (after running a correction method). For each sequence, a multiple alignment of the three versions is computed, and the results are analyzed to provide correction quality measures. In a second module, reads are assembled using Minimap2 and Miniasm [3], and both the reads and the contigs are mapped on the reference genome, to provide remapping and assembly statistics. A text summary, plots and a pdf summary are output.

1.2 Previous works

Works introducing novel correction methods usually evaluate the quality of their tools based on how well the corrected long reads realign to the reference. Despite being interesting, this information remains incomplete. In particular, it is likely not to mention poor quality reads, or regions to which it is difficult to align. Inspired by earlier works by Yang et al. [4] and Miclotte et al. [5], La et al. introduced a new way to obtain metrics describing the quality of the error correction itself [6], that does not solely present the similarity between the aligned corrected reads and the reference genome. Relying on simulated data, La et al. proposed the idea of a three way alignment between the reference genome, the uncorrected reads, and the corrected reads. They presented results on Pacific Biosciences data for hybrid error correction tools, by introducing LRCstats, an evaluation tool aiming at answering to the aforementioned problematics. LRCstats is based on an alignment scheme that allows to compare corrected, uncorrected and reference versions of a read. This way it provides reads' error rate before and after correction, as well as the detailed count of every type of error. However, only studying the reads' error rate after correction is not a satisfying indication of the corrector's behavior. For instance, there is no clue about the putative insertions of new errors by the corrector, because its precision is not assessed. To overcome this issue, additional metrics such as precision (relevant corrected bases among

all bases changed by the corrector), but also recall (correct bases that have been retrieved by the corrector among all bases to be corrected) should be given, in order to better understand the correction methods' pros and cons.

Moreover, LRCstats suffers from high resource consumption when processing large numbers of reads, i.e. when coverage or genome size are large. However, deep coverage is expected to help the correction of very long sequences [1]. Thus, the correction of such datasets must be assessed in a reasonable amount of time. Additionally, LRCstats's alignment scheme becomes limited when sequences to process grow longer. However, extremely long reads start to appear in recent works for larger genomes [7], and require correction as well.

1.3 Contribution

In order to cope with the identified limits of LRCstats, we propose ELECTOR, a new evaluation tool for long read error correction methods. ELECTOR provides a wider range of metrics than LRCstats, that assess the actual quality of the correction, such as recall, precision, and correct bases rate for each read. Such metrics have already been proposed in earlier works dedicated to short reads, such as ECTools [4]. However, ECTools' contribution is out of the scope of this work since algorithms to process short reads are different from those at stake in our case. ELECTOR also informs about typical difficulties long read correctors can encounter, such as homopolymers, and reads that have been trimmed, split or extended during the correction. Our multiple alignment strategy allows to compare three different versions of each read: the uncorrected version, as provided by the sequencing experiment or by the read simulator, the corrected version, as provided by the error correction method, and the reference version, that represents a perfect version of the original read, on which no error would have been introduced. In addition, ELECTOR performs and evaluates reads remapping and assembly, which were not included in the LRCstats pipeline.

In order to provide additional metrics, the three-way alignment paradigm used in LRCstats is replaced by a scalable multiple sequence alignment in ELECTOR. In order to allow the multiple sequence alignment strategy to scale to ultra-long reads, we also propose a novel heuristic that combines anchoring and partial order alignment. This way, we also propose a faster and more scalable evaluation pipeline than LRCstats.

ELECTOR can be used on simulated as well as real long read datasets, provided a reference genome is available for the sequenced species. For simulated reads, it works with it is compatible with state-of-the-art long reads simulation tools, such as Nanosim [8] or SimLord [9], on which introduced errors are precisely known. ELECTOR is meant to be a user friendly tool, that delivers its results through different output formats, such as graphics than can be directly integrated to the users' projects. This tool was designed to be directly compatible with a wide range of state-of-the-art error correction tools, without requiring any preprocessing by the user. In particular, ELECTOR is compatible with the latest self-correction methods, and we thus present novel results on such tools, that were not tackled by LRCstats.

2 Material and methods

2.1 Input sequences

ELECTOR is implemented as a pipeline that is divided in two modules. An overview is shown in Figure 1. Input sequences are passed to the two modules independently. The full evaluation pipeline was initially designed for simulated long reads. This choice was motivated by the need to know the *reference* sequences (which, we recall, represent perfect versions of the original reads, on which no error would have been introduced) in order to precisely control the results brought by the assessed correction method. However, the reference sequence requisite only depends on the availability of a reference genome, and ELECTOR can thus be used on real data as well. According to the nature of the reads (simulated or real), these *reference* sequences are retrieved in different ways.

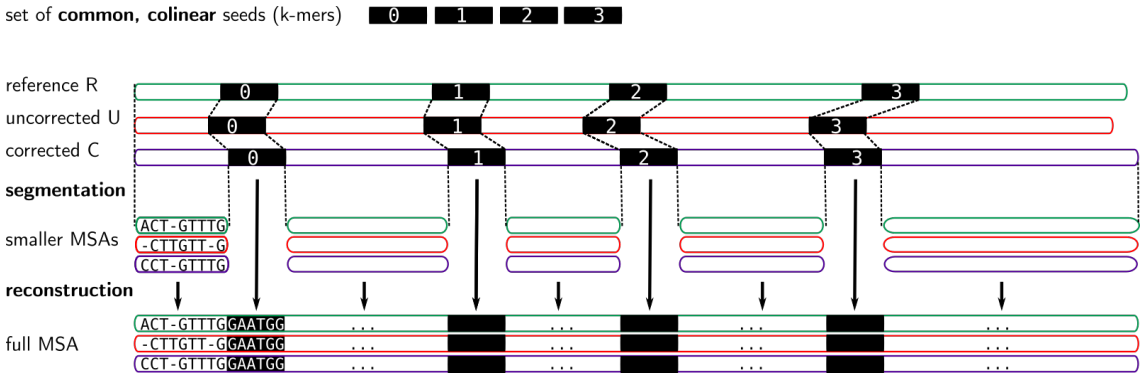

Figure 2: **Segmentation strategy to compute a multiple sequence alignment for a triplet of *reference*, *uncorrected* and *corrected* versions of a read.** Instead of computing a multiple alignment on the whole lengths of the sequences, we rather divide this problem in smaller multiple alignment. As each version is different, in order to decide were to start and end alignments we find seed k-mers (in black that are local exact matches between the three sequences.

2.1.1 Input reads

Our pipeline is compatible with long reads simulators SimLord and NanoSim. This means that when using long reads simulated with one of these two tools, the *reference* sequences are directly retrieved by ELECTOR, by parsing the files generated during the simulation. By using these state-of-the-art long reads simulation tools, we ensure to take as input sequences that closely simulate the actual characteristics of the long reads. However, other long reads simulation tools can also be used. In this case, the user must provide the *reference* sequences to ELECTOR itself. Further configuration of the simulation tools such as the error rate, or the long reads coverage, is the user's call and has no impact on the ELECTOR pipeline. The genome used for the simulation, the simulated erroneous reads, and their corrected versions, output by the desired correction method, are then provided as an input to our pipeline. For hybrid correction methods, whether the short reads are real or simulated has no impact on ELECTOR.

In the case of real data, the reads are also passed along with their corrected versions and with the genome to our pipeline. The *reference* sequences are then retrieved by aligning the uncorrected reads to the reference genome, using Minimap2 [10]. Only the best hit for each read is kept, and used to determine the corresponding reference sequence. In the case a read cannot align to the reference genome, and thus cannot produce a reference, the read is excluded from the analysis. Apart from that, the rest of the pipeline remains the same, although the computation of some metrics slightly vary, due to the alignment.

2.1.2 Prepare triplets of sequences for multiple alignment

We propose to compare three different versions of each read in a triplet multiple alignment. The *reference* version, the *uncorrected* version, and the *corrected* version are used. These three versions of each read undergo a multiple sequence alignment, in order to collect their differences/similarities at each position of the alignment. Metrics can then be derived from these observations. A sequencing experiment or a long reads simulation provides the *uncorrected* version of the reads. The correction method applied to these uncorrected reads provides the *corrected* version. Finally, the *reference* reads are retrieved either from the files generated by the simulation tools, or from the alignment of the long reads to the reference genome, as mentioned in the previous Section.

The three different versions of a given long read corresponding to one another are then retrieved by using their headers. The files containing the different versions of the long reads are sorted in the same order, so that corresponding *reference*, *uncorrected*, and *corrected* appear in the same order in each file. In the particular case of a corrected read that would be split into several (FASTA) fragments, because its correction led

to fragmented subsequences of the initial read, we duplicate the corresponding *uncorrected* and *reference* versions, so that we always have triplets. Each triplet is then aligned using a multiple sequence alignment scheme.

2.2 Scalable triplet multiple alignment

2.2.1 Principle

For each of the three versions of a read, the triplet multiple alignment module computes a multiple sequence alignment (MSA) using a partial ordered alignment algorithm, starting with the *reference* sequence, then adding the *corrected* one, and finally the *uncorrected* version. This step yields a multiple alignment matrix that is output in pseudo FASTA (PIR) format for each triplet. The triplet multiple alignment is computed using an implementation of partial ordered alignment graphs [11]. Partial ordered alignment graphs are used as structures containing the information of the multiple aligned sequences. In this method a directed acyclic graph (DAG) contains the previous multiple alignment result. Successive nucleotides from the sequences are stored in vertices, and each new sequence is aligned to this DAG in a generalization of the Needleman-Wunsch algorithm. Paths in the graph represent the successive alignments.

However, such a procedure can be time-consuming when applied to long reads. Thus, we propose a novel multiple alignment heuristic, that we implemented for ELECTOR's purpose, and describe below.

2.2.2 Segmentation strategy for the MSA

The time complexity of the partial ordered alignment algorithm for MSA increases linearly with the average number of branches in the DAG. Its global complexity is in $\mathcal{O}(vMN)$ with v the average number of predecessors per vertex in the graph, M the length of the sequence to be aligned and N the number of vertices in the graph. Thus, very long reads induce longer running times according to their length, but they also imply more errors and branches in the graph that further increase the computation duration.
In order to reduce the time footprint of our approach, we propose a segmentation strategy. It consists in dividing the triplet multiple alignment into several smaller multiple sequence alignments. Drawing inspiration from Mummer's [12] and Minimap's [3] longest increasing subsequence approaches, we divide the multiple alignment problem into instances of short windows shared by all three versions of a given read. We therefore compute several short multiple alignment and concatenate them instead of computing a large one. See Figure 2 for an example.

If we were able to bound the size of the windows we could guarantee an asymptotic time linear to the read length. In practice our implementation can produce large windows but we observe a running time almost linear in the size of the reads.

The windows are computed as follow. For each triplet, we compute seed k-mers that have the following properties: 1-they appear in each of the three versions of the sequence, 2- they are not repeated across any of the versions of the sequence, 3-they are not overlapping in any sequence.

Using dynamic programming the longest seed k-mers subsequence common to the three sequence is computed. Thus, those anchors delineate positions where each version of the read have an exact match of k nucleotides. For each pair of consecutive seeds, we then extract the left seed followed by the subsequences from each version of the read. We align these subsequence triplet independently, as described in the previous paragraph, using subsequently smaller alignment matrices. This way we divide the global multiple alignment problem into smaller problems, separated by regions of exact matches. Then, the multiple small MSAs are concatenated to obtain a single MSA of the whole length of the triplet.

The size of these seed k-mers is adapted according to the current observed error rates [3, 13], *i.e.* 9 to 15 nucleotides. As it is difficult to *a priori* set a k-mer size, we designed a quick iterative strategy that tries several values of k, in order to choose the most suitable for a given triplet.

To avoid to compute metrics on poorly corrected reads we filter out corrected reads which length is below l% of the reference length (l being a parameter set to 10) or reads for which no of seed k-mers could be found. These two types of filtered reads are tagged and reported apart in ELECTOR's summary to inform

the user about their numbers.

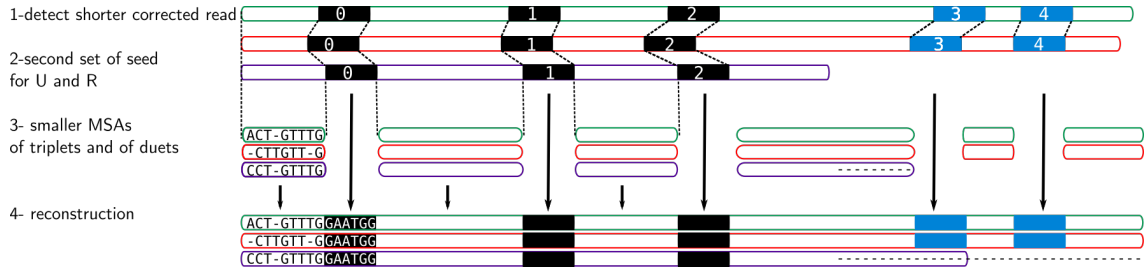

Figure 3: **Segmentation strategy when the corrected read is smaller.** The corrected read is shortened on its right end. In order to avoid passing subsequences from seed 2 to the MSA module, which would be long to compute, we perform a second seed strategy (gray seeds) in order to divide the remaining sequences in reference and uncorrected into smaller pieces to compute the MSA on. The full MSA is reconstructed by concatenation and we add dots to complete the corrected MSA line.

2.2.3 Handle reads of different sizes in the segmentation strategy

In the case of a trimmed/split read, the *corrected* version is shortened in comparison to the two other versions and a part of the *reference* is missing from the *corrected*. A prefix and/or a suffix of the *reference* can be missing depending on the case, different scenarios are outlined in Figure 3. In the case of a missing prefix, the first window selected by the segmentation strategy will contain a very large prefix in the *reference* and *uncorrected* version and a very small sequence in the *corrected* version. This is due to the fact that we only use anchors shared among the three sequences. It would be irrelevant to compute a MSA between those three sequences. Furthermore computing a MSA on two very large sequences is extremely expensive. To cope with this problem, we detect such cases by checking the sequences length and compute a segmentation using only the *reference* and *uncorrected* sequences unused prefix. We therefore compute a MSA using the *corrected* prefix sequence along with its corresponding regions from *reference* and *uncorrected* extracted from the segmentation. This way we are able to efficiently compute a MSA when the corrected reads do not cover all the original region avoiding to run a MSA on large/unrelated sequences. The procedure is symmetrical for a missing suffix in the corrected sequences. This procedure is extremely important for correctors that output numerous split reads, which would induce extremely long runtime due to large sequence MSA computations described before.

2.3 Inference of quality assessment metrics from MSA

2.3.1 Classification of corrected reads

We report different categories of corrected reads in ELECTOR. First, the reads which quality is so bad they were removed before the MSA step. As mentioned before, these are the reads for which an insufficient number of seed k-mers were found. We only report their number as no metric can be computed, since they are not aligned. Second the split reads. The corrected fragments come from a single original read that could only be corrected on one or several distinct parts that are reported apart. We collect all fragments that come from a single initial read and report how many reads were split. For each trimmed or split corrected read, we report the total uncorrected length of its associated reference read (*i.e.* the length that is covered by no fragment). Third, extended reads are reads that have a subsequence at their left and/or right end that was not present in the reference sequence. These reads can be chimeras from the correction step. However they can also be reads that were over-corrected by a graph-based correction method, that kept on traversing the graph after reaching the uncorrected reads' extremities. We do not compute quality assessment metrics on the extended regions, but we report the number of extended reads, as well as their mean extension

size, with respect to the reference reads. Fourth, "normal" corrected reads that are neither trimmed/split nor extended. Figure 4 shows how we deduce reads categories from the MSA result. Finally, soft clipped reads are reads from a real dataset for which the extremities were soft clipped during the alignment to the reference genome. This category can only arise when processing real data, as we only retrieve reference reads by aligning the uncorrected reads to the reference genome in the case of real data. For such reads, we do not compute quality assessment metrics on the soft clipped regions, as they could not be properly aligned to the reference genome, and were therefore not used to determine the reference read. In addition to this soft clipped category, such reads can also be trimmed, split, extended or "normal".

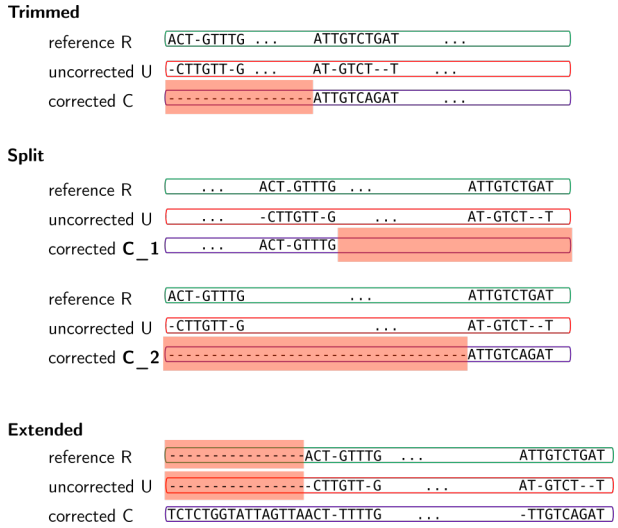

Figure 4: **Three scenarios of corrected read categories in MSA result.** Trimmed/split reads have a corrected version with missing left or right part. Extended corrected reads have a corrected version with a longer left or right part, which is not present in the two other versions.

2.3.2 Recall, precision, error rate

Once the MSA is computed, we have a base-wise information of the differences and similarities in nucleotide content for each of the three versions of a sequence. Insertions or deletions are represented by a "." in the deleted parts, and by the corresponding nucleotide (`A`,`C`,`T` or `G`) in the sequence containing an insertion relatively to the two others. Let us denote $nt(R,p_i), nt(C,p_i), nt(U,p_i)$ the characters of *reference, corrected* and *uncorrected* versions in $\{A,C,G,T,.\}$, at position p_i $(0 \leq i < N)$, in a MSA of size N. Figure 5 shows how recall and precision are computed. The set of positions to correct $\mathcal{P}$ contains positions p_i such as $nt(R,p_i) \neq nt(U,p_i)$. The set of existing position in the corrected version $\mathcal{E}$ is defined by including any position p_x from the *corrected* version that is not counted in a trimmed/split/extended region. The processed positions set $\mathcal{C}$ is defined as $\mathcal{P} \cup \{p_j / nt(C,p_j) \neq nt(R,p_j)\} \cap \mathcal{E}$. The correct positions set $\mathcal{C}o$ is defined as $\mathcal{C} \cap \{p_j / nt(C,p_j) = nt(R,p_j)\}$. The recall, precision and error rate are computed as such:

$$Recall = \frac{card(\mathcal{C} \cap \mathcal{P})}{card(\mathcal{P})} \tag{1}$$

$$Precision = \frac{card(\mathcal{C}o \cap \mathcal{C})}{card(\mathcal{C})} \tag{2}$$

$$Error\_rate = 1 - \frac{card(\mathcal{C}o)}{\sum_{i=0}^{c-1} i} \tag{3}$$

with c the length of the corrected read.

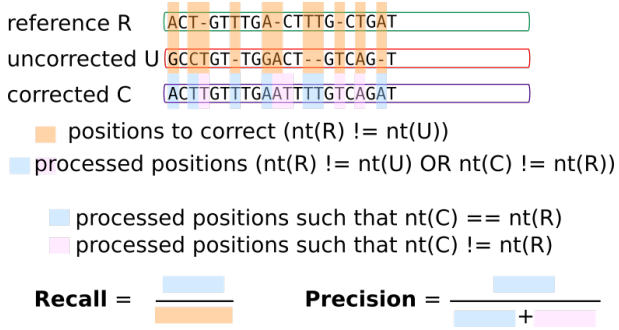

Figure 5: Compute recall and precision using triple base-wise comparison at each MSA's position. $nt(R)$ (respectively $nt(U), nt(C)$) represents the character in reference (respectively uncorrected, corrected) line of the MSA at a given position.

2.3.3 Additional metrics

ELECTOR's first module also provides information on the number of split or trimmed corrected reads, on the mean missing size of the trimmed / split reads, and on the mean extension size of the extended reads. The size distribution of sequences before and after correction is presented graphically. In the case of split reads, we report the length of each split fragment in the distribution. The %GC is also output, as well as the insertion/deletion/substitution counts, before and after correction. We show the ratio of homopolymer sizes in the corrected version over the reference version. The closer it is to one, the better the corrector overcame possible non-systematic errors in ONT reads. More details on the computation of these metrics are shown in Supplementary.

2.3.4 Remapping of corrected reads

We perform remapping of the corrected reads to the reference genome using BWA-MEM [14], due to the high quality of the long reads after error correction. We report the average identity of the alignments, as well as the genome coverage, *i.e.* the percentage of bases of the reference genome to which at least a nucleotide aligned.

2.3.5 Post-correction assembly metrics

We perform the assembly of the corrected reads using Miniasm [3], as we mainly seek to develop a pipeline providing fast results. We acknowledge that assemblers such as Smartdenovo [15] or Canu [16] are more sensitive, but as they display much larger runtimes, Miniasm provides a satisfying compromise.

As for the metrics of the assembly, we output the overall number of contigs, the number of contigs that could be aligned, the number of breakpoints of the aligned contigs, and the NGA50 and NGA75 sizes of the aligned contigs. The alignment of the contigs is also performed with BWA-MEM [14], and the computation of the different metrics is performed by parsing the generated SAM file.

Experiment	Recall	Precision	Correct bases	Time
"1k" MSA	93.964%	93.479%	97.639%	11h
"1k" segmentation + MSA	93.809%	93.507%	97.631%	38min
"10k" MSA	84.505 %	88.347%	95.290%	107h
"10k" segmentation + MSA	84.587 %	88.278%	95.250%	42min

Figure 6: Comparison of the two multiple alignment strategies on a simulated datasets from *E. coli* genome.The reads from the "1k" experiment were simulated with a 1k mean length, a 10% error rate and a coverage of 100X. The reads from the "10k" experiment were simulated with a 10k mean length, a 15% error rate and a coverage of 100X. The reads were corrected with MECAT with default parameters.

3 Results

3.1 Validation of segmentation strategy for MSA

3.1.1 Comparison of regular and segmentation strategies for strategies for multiple alignment

In order to validate our segmentation strategy for MSA, we show to which extent its results differ from the classic MSA approach. We expect that recall, precision and correct base rate hardly differ, thus showing that both behaviors produce very similar results. Conversely, we expect an important gain in time with our segmentation strategy compared to the original algorithm. We thus compared multiple alignment results obtained with our strategy to results obtained with the regular implementation of the partial ordered alignment on two datasets of different read lengths, which affects the runtime of the alignments. Results are presented in Table 6.

They show that our segmentation strategy and the regular approach only differ by a few digits in the presented metrics for both experiments. However, using segmentation, a substantial gain in time is achieved. Moreover, while the classic MSA strategy runtime raises with respect to the read length, our approach no longer suffers from this drawback.

3.1.2 Validation of metrics values

In the following sections, we show comparisons of ELECTOR and LRCstats results. However, LRCstats is itself based on alignment schemes, and thus does not provide a "ground truth", unbiased by alignment or mapping, to compare ELECTOR to. Moreover some of ELECTOR's metrics do not exist in LRCstats outputs. Thus, in order to obtain a "ground truth" to assess ELECTOR's results, we designed a simulation experiment where both corrected and uncorrected reads / versions are produced. The simulation of uncorrected reads follows typical long reads error rates and profiles. The uncorrected reads simulated using a genomic reference from *E. coli* genome by extracting read sequences at random positions. The simulation of correction for a read is performed by choosing for each position on the read whether it is modified by the corrector. If the position is modified, the corrected base is a false positive (wrongly transformed during the correction), a false negative (not corrected although it needed a correction) or a true positive (corrected). This way we control recall, precision, error rate and indels/substitution levels. We simulated a correction process so that precision and recall would reach around 99%, as observed in best current correctors.

We used a 10X, 20% error rate dataset, designed to represent a difficult case. Then we ran ELECTOR on these two datasets and compared the results to ground truth metrics. Results are shown in Table 1. In the second column we show ELECTOR's values. In the third column we assess how different ELECTOR is from ground truth by computing $\frac{abs(GT-EL)}{GT} \times 100$, as the percent of the ground truth the difference between ELECTOR's value and ground truth's value represents, with GT the ground truth for a given metric and EL the ELECTOR's value.

The highest percentage of difference is reached for the sum of indels and mismatches in the uncorrected

Metric	ELECTOR	difference (% ground truth)
recall(%)	98.99	4.0 E-2
precision(%)	99.92	1.0 E-1
error rate	9.920E-2	2.3
indels/mismatches in uncorrected	8380984	4.1
indels/mismatches in corrected	491728	3.4

Table 1: ELECTOR results on 10X *E. coli* simulated reads and comparison with ground truth. The third column presents how far ELECTOR's output values are from the ground truth values we computed during our simulation.

sequence (4.1%). The precision and recall displayed by ELECTOR show little difference with the real values (1.0 E-1% or less).

3.2 Assessment of metrics on genomic datasets

3.2.1 Datasets

We present results of ELECTOR on several simulated datasets of several species (*A. baylyi*, *E. coli*, *S. cerevisiae*, *C. elegans*). Further details on each data set are given in Supplementary Material.

3.2.2 ELECTOR results

We display the ELECTOR pipeline results using reads corrected by the following list of correctors: HALC [17], HG-CoLoR [18], LoRDEC [19], Canu, Daccord [20], MECAT [21] and LoRMA [22]. The complete results provided by LCRstats and ELECTOR on each corrected dataset are presented in Supplementary Material. As previously detailed in Section 2.3.2, the first module of ELECTOR computes general metrics: mean recall, precision, correct base rate and other additional metrics. For the three first results, a graphic representation of their distribution is also made available.

Examples of results as output by this first module of ELECTOR are also presented for the six correction methods and three datasets in Table 3. This table shows that ELECTOR reported recalls from 95 to almost 100% and precisions from 94 to more than 99% for all tools. These results are consistent with results presented in the different tools' publications on datasets from the same species.

The second module of ELECTOR performs remapping of the reads on the reference and assembles them, thus providing additional information that are not available through LRCstats. The metrics output by this second module are detailed in Section 2.3.4 and Section 2.3.5.

3.2.3 Comparison to LRCstats

The comparison of the metrics displayed by ELECTOR and LRCstats are shown in Table 2. This Table shows the results obtained for each tool, and the supplementary metrics offered by ELECTOR that are not displayed by LRCstats. We chose to present two different examples using a self and a hybrid correction method, respectively HALC and Canu, on the *S. cerevisiae* data set. ELECTOR's novel metrics point out important differences between the two correction methods, such as the high quantity of trimmed and/or split reads when using HALC in comparison to Canu. Results computed by LRCstats and ELECTOR on other data sets and other correction methods can be found in Supplementary Materials.

Both LRCstats and ELECTOR compute metrics for uncorrected and corrected versions of the reads. The first result to notice is that the error rate announced in uncorrected sequences can differ from one correction method to another, both for ELECTOR and LRCstats. This is explained by the fact that HALC and Canu do not correct the same set of reads. As a result, the corresponding uncorrected reads used to compute the error rates are not the same either.

Metric	Uncorrected		Corrected by HALC	
	ELECTOR	LRCstats	ELECTOR	LRCstats
Throughput	298,370,918	237,655,341	212,266,193	214,152,119
Error Rate	0.1403	0.1751	0.0042	0.0023
Insertions	28,772,841	32,589,970	100,874	215,507
Deletions	5,235,890	8,991,984	1,035,978	120,743
Substitutions	4,058,953	1,633,123	198,853	221,646
Recall	-	-	0.9997	✖
Precision	-	-	0.9959	✖
Trimmed/split	-	-	12,043	✖
Mean missing size	-	-	577.5	✖
Extension	-	-	71	✖
Mean extension size	-	-	53.2	✖
Low quality	-	-	160	✖
Small reads	-	-	3436	✖
%GC	-	-	38.2	✖
	Uncorrected		Corrected by CANU	
Throughput	244,560,743	244,633,066	229,555,492	229,825,812
Error Rate	0.1425	0.1781	0.0506	0.0694
Insertions	30,090,583	34,105,075	12,252,413	12,942,568
Deletions	5,483,119	9,489,618	2,574,320	3,134,365
Substitutions	4,375,017	1,748,302	2,197,172	1,591,650
Recall	-	-	0.9515	✖
Precision	-	-	0.9495	✖
Trimmed/split	-	-	2,216	✖
Mean missing size	-	-	35.1	✖
Extension	-	-	178	✖
Mean extension size	-	-	30.7	✖
Low quality	-	-	43.0	✖
Small reads	-	-	0.0	✖
%GC	-	-	38.7	✖

Table 2: Comparative results of ELECTOR's and LRCStats' ouptuts on the *S. cerevisiae* dataset using a hybrid corrector (HALC) and a self corrector (Canu). Dashes indicate that the metric is not accessible. Crosses indicate that the assessment tool does not provide this metric.

Second, as it can be seen from the throughput metrics, ELECTOR and LRCstats do not process the same quantity of reads. This is due to the fact that they rely on different rules to exclude reads that are too difficult to process in the alignment schemes, but also because of their respective behaviors toward split reads. These differences thus have an impact on the metrics displayed for corrected reads. ELECTOR's throughput is a little smaller than LRCstats', however reads uncounted in the throughput are directly reported in precise categories in ELECTOR (very small reads and low quality reads), while they are lost in LRCstats' output.

Different alignment strategies in both tools also have impacts on the results, which explains the differences seen in indels and substitution counts. However, ELECTOR and LRCstats globally report the same trends of two successful corrections that decreased the error rates.

3.2.4 Performances comparison

We compared LRCstats and ELECTOR's runtimes on several datasets in Tables 5,4. The datasets are chosen to represent different factors of pitfalls. We thus vary genome sizes and read throughput, as well as reads size distribution. ELECTOR's runtime and memory peaks are computed for the two independent modules of the ELECTOR pipeline: the read triplets multiple alignment step, allowing to access general metrics, and the remapping and assembly step. The first module is comparable to the LRCstats pipeline, as both perform

Metric	HALC		HG-CoLoR		LoRDEC		CANU		daccord		MECAT	
	uncorrected	corrected	uncorrected	corrected	uncorrected	corrected	uncorrected	corrected	uncorrected	corrected	uncorrected	corrected
E. coli												
Throughput	101829227	81199351.0	93003632	84089814.0	132043801	77969503.0	91933413	86443218.0	93273280	83773362.0	81143184	58979203.0
Error Rate	0.1415	0.0015	0.1428	0.0007	0.1384	0.0015	0.1432	0.0524	0.1433	0.004	0.1332	0.0052
Recall	-	0.9999	-	1.0	-	0.9999	-	0.9495	-	0.9988	-	0.9983
Precision	-	0.9985	-	0.9993	-	0.9986	-	0.9476	-	0.9961	-	0.9949
S. cerevisiae												
Throughput	298370918	212266193.0	259555450	219744436.0	529254926	188228237.0	244560743	229555492.0	251569142	222050951.0	219909311	162057920.0
Error Rate	0.1403	0.0042	0.1414	0.003	0.1325	0.0054	0.1425	0.0506	0.1426	0.0054	0.1339	0.0066
Recall	-	0.9997	-	0.9999	-	0.9995	-	0.9515	-	0.9986	-	0.998
Precision	-	0.9959	-	0.9971	-	0.9947	-	0.9495	-	0.9946	-	0.9936
C.elegans												
Throughput	3547144593	1588220052.0	2158838211	1726223265.0	3954331369	1154508245.0	2063369590	1934674074.0			1289267335	870965775.0
Error Rate	0.1377	0.0153	0.1397	0.0065	0.1242	0.0126	0.1427	0.0496			0.1199	0.0065
Recall	-	0.9989	-	0.9997	-	0.9989	-	0.9527	-		-	0.9982
Precision	-	0.985	-	0.9936	-	0.9875	-	0.9505	-		-	0.9936

Table 3: Examples of the main statistics reported by ELECTOR on simulated datasets. A dash in a row/column indicates that the metric in not computable from the data. On the *C. elegans* dataset, daccord could not be run.

similar operations. However, LRCstats does not include modules to perform read remapping and assembly, thus we present this second module of ELECTOR apart.

We first assess in Table 5 the performances of both tools on simulated data with different read length. We observe that the runtime and memory consumption of LRCstats grow with the read length and we were unable to run it from the 100,000 base pairs dataset on our 250GB cluster. This behavior may be a problem to work on very long reads. Due to its segmentation strategy ELECTOR is able to handle larger reads, up to one megabase length. Furthermore its time and memory consumption do not raise much with the read length. This shows that ELECTOR is able to scale to extremely long reads. Considering the availability and usefulness of such very long reads library, we believe that this ability to efficiently handle long sequences is one of the main advantages of ELECTOR.

In another experiment, represented Table 4, we evaluate the respective runtime of ELECTOR and LRCstats on several corrected datasets. We added, for the perspective, the runtime of the correction method itself. Interestingly we observe that LRCstats is often more time consuming than the correction method, which is not desirable. ELECTOR presents reduced runtime, showing that it could be used to mitigate that heavy analysis time.

3.3 Assessment of real data

3.3.1 Validation of ELECTOR on real data

In order to validate the real data mode of ELECTOR, we ran the following experiment, reported in Table 6. For this, we used a simulated dataset, and assessed its correction using the two different ELECTOR mode, simulated and real data. First, we ran it classically, by providing the simulation files as input, so ELECTOR could retrieve the actual reference reads by parsing the files. Second, we ran it by only providing the fasta file of simulated reads as input, so ELECTOR had to retrieve the reference reads my mapping the uncorrected long reads to the reference genome, as if they were actual real long reads. We ran this experiment on the *S. cerevisiae* dataset, and to further validate ELECTOR's behaviour on real data, we assessed correction of both a hybrid corrector, HALC, and a self-corrector, Canu. Results of these experiments are shown in Table 6. We observe that ELECTOR's results in simulated and real mode are consistent. In particular, the recalls and precisions are very similar. The same trend appears as for the comparison to LRCstats: the two modes show some differences in the input reads (as shown by the throughputs), that have an impact on the differences observed between their results. This is due to the bias induced by the additional alignment step that is required in the real mode. The main differences that appear in the metrics occur on metrics that are highly dependent on the alignment results, such as the number of trimmed and extended reads, and the sizes of these events.

Method	HALC	HG-CoLoR	LoRDEC	Canu	daccord	MECAT
A. baylyi						
Corrector	22min	47min	6min	10min	20min	43sec
LRCstats	3h50	3h52	3h38	3h10	3h59	2h02
ELECTOR	**14min**	**10min**	**45min**	**9min**	**12min**	**9min**
E. coli						
Corrector	24min	45min	8min	12min	27min	52sec
LRCstats	4h58	5h02	4h37	4h05	4h20	2h30
ELECTOR	**28min**	**13min**	**1h17**	**11min**	**12min**	**11min**
S. cerevisiae						
Corrector	1h19	4h32	28min	31min	1h15	2min
LRCstats	10h56	12h26	12h14	10h46	12h04	6h59
ELECTOR	**1h55**	**1h07**	**4h59**	**32min**	**44min**	**32min**
C. elegans						
Corrector	5h59	88h56	6h01	4h33	-	22min
LRCstats	83h29	81h05	70h00	85h08	-	-
ELECTOR	**32h35**	**10h30**	**29h48**	**4h19**	-	**3h12**

Table 4: Runtimes of ELECTOR and LRCstats on different datasets and different correctors. Both were launched with 9 threads. The runtimes of the correctors are also included as a matter of comparison. The speediest assessment method is shown in bold for each case. When the assessment method is also speeder than the correction method itself, it is underlined. daccord could be run on the *C. elegans* dataset, and LRCstats crashed on the *C. elegans* dataset corrected by Canu.

Genome	Read length	Memory (MB)	Elapsed time	CPU time
E. coli	1k	699	2:07	9:37
E. coli	10k	621	2:13	9:13
E. coli	100k	604	2:14	9:39
E. coli	1M	817	2:44	11:05

Table 5: Performance results obtained from a simulated coverage of 10x of the respective reference genome on a 20 core cluster with 250GB.

3.3.2 Results on a real human dataset

In order to demonstrate ELECTOR's results in a realistic scenario for large genomes, we selected a correction of human sequencing data, and show results in Table 7. The reads were corrected with MECAT before running ELECTOR. Using 20 threads, we were able to obtain the results for the 123,410 corrected reads of the dataset in less than 7 hours. We report ELECTOR's metrics and runtime on this correction in Table 7. ELECTOR shown that MECAT is able to correct human reads with a 20% error rate with more than 90% of recall and precision. Such results are consistent with results shown in MECAT's publication.

4 Discussion and perspectives

We described and demonstrated ELECTOR's heuristics for multiple sequence alignment and its important speedup in comparison to LRCstats. While showing same trends in the results they display for correctors, LRCstats and ELECTOR have differences in their common metrics. This is explained by a set of different choices and heuristics between the two tools. First, split reads are not aligned and reported the same way. LRCstats concatenates the different parts of a split read before aligning the concatenation, even if a missing zone can exist between two fragments. This behavior can complicate the alignment task and introduce a bias in the output metrics. On the contrary, ELECTOR processes the different fragments separately be-

Metric	Uncorrected		Corrected by Halc	
	Simulated	Real	Simualted	Real
Throughput	298,370,918	297,798,663	212,266,193	212,141,319
Error Rate	0.1403	0.1449	0.0042	0.0104
Recall	-	-	0.9997	0.9938
Precision	-	-	0.9959	0.9897
Insertions	28,772,841	26,796,500	100,874	90,737
Deletions	5,235,890	5,042,365	1,035,978	1,490,680
Substitutions	4,058,953	3,682,863	198,853	182,590
Trimmed/split	-	-	12,043	13,320
Mean missing size	-	-	577.5	896.0
Extension	-	-	71.0	39.0
Mean extension size	-	-	53.2	72.0
Low quality	-	-	160.0	152.0
Small reads	-	-	3436.0	3438.0
%GC	-	-	38.2	38.2
	Uncorrected		Corrected by Canu	
Throughput	244,560,743	244,402,568	229,555,492	229,403,697
Error Rate	0.1425	0.1442	0.0506	0.052
Recall	-	-	0.9515	0.9499
Precision	-	-	0.9495	0.9481
Insertions	30,090,583	28,452,967	12,252,413	10,965,458
Deletions	5,483,119	5,800,286	2,574,320	2,916,564
Substitutions	4,375,017	4,081,445	2,197,172	1,940,888
Trimmed/split	-	-	2216.0	4943.0
Mean missing size	-	-	35.1	74.7
Extension	-	-	178.0	169.0
Mean extension size	-	-	30.7	31.9
Low quality	-	-	43.0	42.0
Small reads	-	-	0.0	0.0
%GC	-	-	38.7	38.7

Table 6: Comparative results output ELECTOR's results using simulated and real modes on the same set of reads from yeast reads using a hybrid corrector (HALC) and a self corrector (Canu).

	uncorrected	corrected with MECAT
Throughput	1108877795	1017050250
Recall(%)	-	91.48
Precision(%)	-	90.11
Error rate	0.2111	0.1002
Average correct bases rate	0.7889	0.8998
Number of trimmed/split reads	-	109228
Mean missing size in trimmed/split reads	-	423.4
Number of over-corrected reads by extention	-	48
Mean extension size in over-corrected reads	-	101.8
%GC	47.4	47.0
Small reads	-	131
Low quality corrected reads	-	313
Insertions	49191851	2524376
Deletions	152684864	104724600
Substitutions	30976164	1619551
Homopolymer ratio	-	0.8179
Runtime	-	6h27min

Table 7: ELECTOR results for MECAT correction of real human dataset. ELECTOR assessed 123,410 reads. Small reads are corrected reads which length is lower than 10.0% of the original read. Homopolymer ratio is the ratio of homopolymer sizes in corrected vs reference. We reported the wallclock time of the run, using 20 threads.

fore reconstituting the whole alignment, and takes into account missing parts. Another key point is that LRCstats starts from the uncorrected versus reference read alignment provided by the SAM file of the simulation. This SAM file is generated sequentially during the simulation, and can thus contain events such as "1I1D" (one insertion followed by one deletion), that are considered as such by LRCstats. On the contrary, this example double event is detected as a single mismatch by ELECTOR. Such differences have an impact on the insertion/deletion/substitution counts. Finally, LRCstats' general alignment scheme is different from ELECTOR's. We could report events where LRCstat's alignment is suboptimal in comparison to Needleman and Wunsch results, while ELECTOR's result is close to Needleman and Wunsch.

ELECTOR is designed to work with simulated reads, and is expected to give the best results when working with it. The real data mode uses a prior alignment of the reads on a reference genome to retrieve the reference versions of the reads. We demonstrated that ELECTOR's main metrics in its real data mode remain similar to what would be obtained in the simulated mode. This shows two limitations of ELECTOR: first, even if the data can come from an actual sequencing experiment, a reference genome needs to exist for the sequenced species in order to retrieve the reference reads. Second, we encourage users to be very cautious about ELECTOR's results with real data when looking at trimmed/split reads numbers and their sizes, since these metrics highly depend on the results of the reference alignment.

With ELECTOR we provide a tool to rapidly assess the quality of a correction through a wide range of metrics. When looking at recent correctors publications, self-correction seems to be giving better results than hybrid correction, as soon as the long read coverage is sufficient (over 30X) [1]. This can be explained by difficulties to align short reads on long reads, for instance because of repeated sequences covered by long reads. It also seems that self correction leads to less fragmented corrected sequences. However, such statements lacks a global study that goes over the sum of individual publications. ELECTOR could be the basis for such a benchmark study.

In ELECTOR we propose an efficient segmentation heuristic for multiple alignment. We adapted this task for the original and specific long read application. There is a current interest for segmented multiple alignment schemes [23]. However, these methods are not specifically designed for noisy long reads. In such

a perspective, a generalization of our segmentation strategy for long reads multiple alignments of any size would be very interesting.

A future application is the assessment of correction methods directly targeted at RNA long reads sequencing. As shown in a recent study [24], RNA long reads have specific requirements that are not met by current methods, which calls for new correctors in the future. ELECTOR could be coupled with a reference transcriptome or a RNA long read simulator, although, currently, such simulation software does not exist to our knowledge.

5 Conclusion

We presented ELECTOR, a tool that enables the evaluation of self and hybrid correction methods, and that can be used in the conception of a benchmark. ELECTOR provides a wide variety of metrics that include base-wise computation of recall, precision, error rate of corrected reads as well as indels, substitutions and homopolymers correction. In particular, recall and precision allow to spot correction methods specific pitfalls, that remain unclear when only looking at the error rates of the corrected reads. These results are presented in a text summary and in pdf and png versions, which allows users to easily integrate them in their reports.

We used ELECTOR on a large list of state-of-the-art hybrid and self correctors, ran on reads from small bacterial genomes to large mammal genomes. With the applications on large genomes and ever increasing lengths of long reads in mind, we designed ELECTOR to be time-saving and scalable. We shown that for most datasets and correctors, ELECTOR runs in a similar amount of time as the corrector itself. We also demonstrated that it can scale to novel ultra long reads.
Thus, it represents a major improvement in comparison to LRCstats. Since ELECTOR is based on multiple sequence alignment, we adapted this strategy to our scaling objectives. We proposed an innovative and promising algorithm of segmentation for multiple sequence alignment of noisy long reads. This procedure drastically reduces the time footprint of the multiple alignment step in ELECTOR. We believe it could be generalized for broad applications implying multiple sequence alignment.

6 Acknowledgements

References

[1] Fritz J Sedlazeck, Hayan Lee, Charlotte A Darby, and Michael C Schatz. Piercing the dark matter: bioinformatics of long-range sequencing and mapping. *Nature Reviews Genetics*, page 1, 2018.

[2] David Laehnemann, Arndt Borkhardt, and Alice Carolyn McHardy. Denoising DNA deep sequencing data-high-throughput sequencing errors and their correction. *Briefings in bioinformatics*, 17(1):154–179, 2015.

[3] Heng Li. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, 32(14):2103–2110, 2016.

[4] Xiao Yang, Sriram P Chockalingam, and Srinivas Aluru. A survey of error-correction methods for next-generation sequencing. *Briefings in bioinformatics*, 14(1):56–66, 2012.

[5] Giles Miclotte, Mahdi Heydari, Piet Demeester, Stephane Rombauts, Yves Van de Peer, Pieter Audenaert, and Jan Fostier. Jabba: hybrid error correction for long sequencing reads. *Algorithms for Molecular Biology*, 11(1):10, 2016.

[6] Sean La, Ehsan Haghshenas, and Cedric Chauve. LRCstats, a tool for evaluating long reads correction methods. *Bioinformatics*, 33(22):3652–3654, 2017.

[7] Miten Jain, Sergey Koren, Karen H Miga, Josh Quick, Arthur C Rand, Thomas A Sasani, John R Tyson, Andrew D Beggs, Alexander T Dilthey, Ian T Fiddes, et al. Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature biotechnology*, 36(4):338, 2018.

[8] Chen Yang, Justin Chu, René L Warren, and Inanç Birol. NanoSim: nanopore sequence read simulator based on statistical characterization. *GigaScience*, 6(4):1–6, 2017.

[9] Bianca K Stöcker, Johannes Köster, and Sven Rahmann. Simlord: Simulation of long read data. *Bioinformatics*, 32(17):2704–2706, 2016.

[10] Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 1:7, 2018.

[11] Christopher Lee, Catherine Grasso, and Mark F Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464, 2002.

[12] Arthur L Delcher, Steven L Salzberg, and Adam M Phillippy. Using MUMmer to identify similar regions in large sequence sets. *Current protocols in bioinformatics*, (1):10–3, 2003.

[13] Mark J Chaisson and Glenn Tesler. Mapping single molecule sequencing reads using basic local alignment with successive refinement (blasr): application and theory. *BMC bioinformatics*, 13(1):238, 2012.

[14] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint arXiv:1303.3997*, 2013.

[15] J Ruan. Smartdenovo: Ultra-fast de novo assembler using long noisy reads, 2017.

[16] Sergey Koren, Brian P Walenz, Konstantin Berlin, Jason R Miller, Nicholas H Bergman, and Adam M Phillippy. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome research*, pages gr–215087, 2017.

[17] Ergude Bao and Lingxiao Lan. Halc: High throughput algorithm for long read error correction. *BMC bioinformatics*, 18(1):204, 2017.

[18] Pierre Morisse, Thierry Lecroq, and Arnaud Lefebvre. Hybrid correction of highly noisy long reads using a variable-order de Bruijn graph. *Bioinformatics*, 34(24):4213–4222, 2018.

[19] Leena Salmela and Eric Rivals. LoRDEC: accurate and efficient long read error correction. *Bioinformatics*, 30(24):3506–3514, 2014.

[20] German Tischler and Eugene W Myers. Non hybrid long read consensus using local de Bruijn graph assembly. *bioRxiv*, page 106252, 2017.

[21] Chuan-Le Xiao, Ying Chen, Shang-Qian Xie, Kai-Ning Chen, Yan Wang, Yue Han, Feng Luo, and Zhi Xie. MECAT: fast mapping, error correction, and de novo assembly for single-molecule sequencing reads. *nature methods*, 14(11):1072, 2017.

[22] Leena Salmela, Riku Walve, Eric Rivals, and Esko Ukkonen. Accurate self-correction of errors in long reads using de Bruijn graphs. *Bioinformatics*, 33(6):799–806, 2016.

[23] Edgar Garriga Nogales, Paolo Di Tommaso, Cedrik Magis, Ionas Erb, Hafid Laayouni, Fyodor Kondrashov, Evan Floden, and Cedric Notredame. Fast and accurate large multiple sequence alignments using root-to-leave regressive computation. *bioRxiv*, page 490235, 2018.

[24] Leandro Ishi Soares de Lima, Camille Marchet, Segolene Caboche, Corinne Da Silva, Benjamin Istace, Jean-Marc Aury, Helene Touzet, and Rayan Chikhi. Comparative assessment of long-read error-correction software applied to rna-sequencing data. *bioRxiv*, page 476622, 2018.