SeqCorrect – A Modular Toolkit for DNA Sequencing Error Correction and Evaluation

Sarah Lutteropp[1] and Alexandros Stamatakis[1,2]

[1] The Exelixis Lab, Scientific Computing Group, Heidelberg Institute for Theoretical Studies, Heidelberg D-68159, Germany
[2] Department of Informatics, Institute of Theoretical Informatics, Karlsruhe Institute of Technology, 76128 Karlsruhe, Germany

Abstract.
Motivation: While there exists a plethora of sequencing error correction tools, the field is still lacking a generalized modular framework for this task. Especially wetlab- and run-dependent error profile characteristics are often ignored by current sequencing error correction methods. Many sequence correction tools ignore sequence-specific errors and do not explicitly model the G/C coverage bias of sequencers. Encapsulating this functionality in separate, user-friendly modules will facilitate the development of future sequencing error correction tools.
Results: We compute expected k-mer counts under an idealized sequencing model and infer run-dependent median G/C coverage biases by counting k-mers in the read dataset and comparing the observed counts with their expected values. We classify k-mers into *untrusted*, *unique*, and *repetitive*. We correct substitution, insertion, and deletion errors and handle repetitive regions by locally and adaptively increasing the k-mer size in a read. Our error correction approach introduces less new errors (false positives) than other tools, but also corrects less errors in total. The main purpose of our toolkit is to simplify the design and evaluation of future error correction approaches.
Availability: SeqCorrect is implemented in C++ and is available for download at `https://github.com/lutteropp/SeqCorrect` under the GNU GPL-3.0 license.
Contact: `sarah.lutteropp@h-its.org`, `alexandros.stamatakis@h-its.org`

1 Introduction

While there already exist numerous error correction tools, there is still a need for an unified correction approach that is easy to extend and to adapt to the user's needs [4]. In this work, we present a re-engineered version of the sequencing error correction toolkit introduced in [6]. While most components remain unchanged, we modified and improved the error correction and evaluation components of our software (see Figure 3).

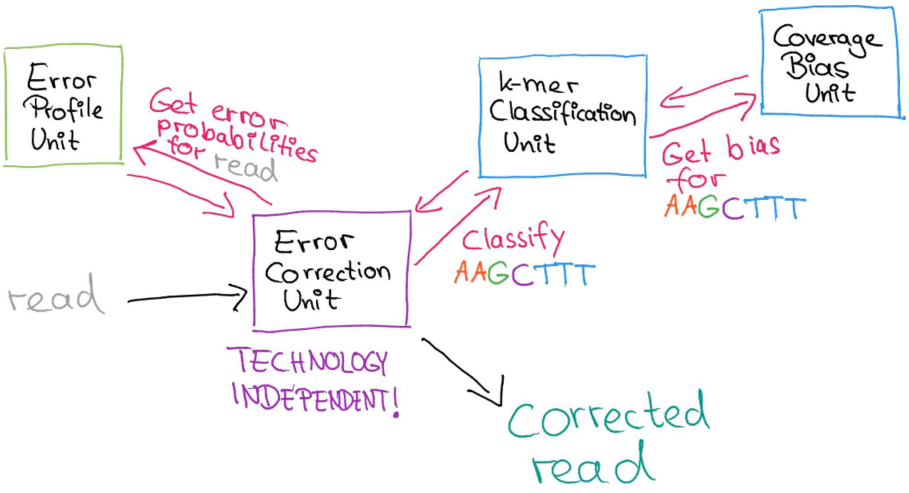

Fig. 1: The modular structure of our toolkit. By encapsulating technology-specific characteristics in the Error Profile Unit and the Coverage Bias Unit, the error correction algorithm itself can remain technology-agnostic.

2 Coverage Bias and *K*-mer Classification

Based on its expected count in the (unknown) genome, we classify a k-mer into either being *UNTRUSTED* (0 occurrences), *TRUSTED* (1 occurrence), or *REPETITIVE* (> 1 occurrences). We compute the expected count of a given k-mer under an idealized sequencing setting (which we call the *Perfect Uniform Sequencing (PUS) Model*), and distinguish between linear and circular genomes (see Figure 2). Estimated k-mer counts in idealized sequencing settings have already been computed before by Schulz *et al.* [11], but they do not distinguish between linear and circular genomes. In real sequencing datasets, the reads can contain artificial DNA sequences which do not originate from the sequenced genome. For example, due to technical restrictions, reads obtained by the Illumina technology contain adapter sequences.

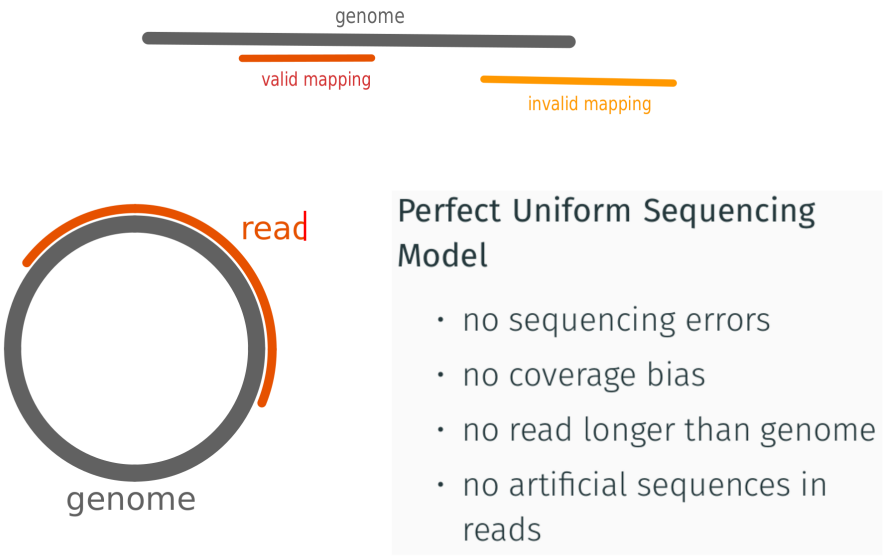

Fig. 2: The Perfect Uniform Sequencing Model describes an idealized sequencing setting.

We assume that reads are independent from each other and each read has the same probability to cover a given area of k consecutive bases in the genome. We can compute the expected coverage of a unique k-mer under our PUS model, using a binomial distribution. Let $(l_i, n_i)_{i=1,\dots,m}$ be the distribution of read lengths. An entry (l_i, n_i) means that there are n_i reads of length l_i in the dataset. Let $\mathbb{P}(l, k)$ be the probability that a read of length l covers a given unique k-mer. This gives us the expected count

$$\text{cov}_{\text{expected}}(k) = \sum_{i=1}^{m} n_i * \mathbb{P}(l_i, k) * (1 - \mathbb{P}(l_i, k))$$

Let N be the estimated genome size. In case of a circular genome, we obtain

$$\mathbb{P}(l, k) = \frac{l - k + 1}{N}.$$

For a linear genome, the expected number of reads depends on the position of the k-mer in the genome. As this is not known, we assume that it is equally likely for the k-mer to occur at any

position in the genome. We obtain

$$\mathbb{P}(l,k) = \frac{(l-k)*(N-l)+1}{(N-k+1)(N-l+1)}.$$

Since the assumption of uniform coverage does not hold in reality, we explicitly model the G/C-coverage bias. The G/C-content of a k-mer is defined as

$$\text{gc}(k\text{-mer}) := \frac{\text{Number of G or C bases in the } k\text{-mer}}{\text{Total number of bases in the } k\text{-mer}}.$$

Thus, for size k, there are $k+1$ possible G/C-contents in total. For each G/C-content and size k, we compute the median coverage bias of all k-mers with the given G/C-content, as in the EDAR paper [14]. For each k-mer large enough such that it is likely to appear at most once in the genome, we compute it's bias as

$$\text{bias}(\text{k-mer}) = \frac{\text{cov}_{\text{observed}}(\text{k-mer})}{\text{cov}_{\text{expected}}(k)}.$$

Then, we define the median coverage bias $bias(k,c)$ for a given size k and a given G/C-content c as the median bias of all k-mers with G/C-content c in the dataset.

For classifying a k-mer (`UNTRUSTED`, `TRUSTED`, `REPETITIVE`), we first transform its observed count in the read dataset into a bias-corrected count. Then, we compare the bias-corrected count with the expected count for the k-mer. We define the bias-corrected count of a k-mer as

$$\text{cov}_{\text{bias-corrected}}(k\text{-mer}) := \frac{1}{\text{bias}(k,\ \text{gc}(k\text{-mer}))} * \text{cov}_{\text{observed}}(k\text{-mer}).$$

Then, we use the following classification rule (with $\mu := \text{cov}_{\text{expected}}(k)$ and $x := \text{cov}_{\text{bias-corrected}}(k\text{-mer})$):

$$type(k\text{-mer}) = \begin{Bmatrix} \texttt{UNTRUSTED} & , \text{ if } x < 0.5*\mu \\ \texttt{TRUSTED} & , \text{ if } 0.5*\mu \leq x \leq 1.5*\mu \\ \texttt{REPETITIVE} & , \text{ if } x > 1.5*\mu \end{Bmatrix}$$

3 Error Correction Approach

For correcting sequencing errors, we distinguish between *base-errors* and *gap-errors* (see Figure 3). An erroneous base can either occur due to an insertion or a substitution of A, C, G, or T. An erroneous gap is caused by a deletion of one or more bases.

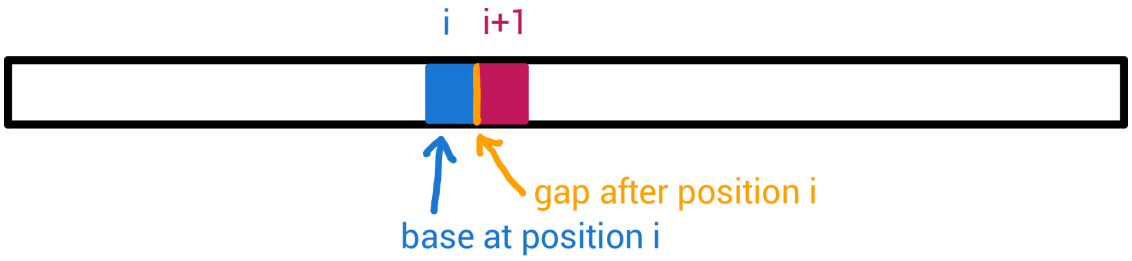

Fig. 3: A base-error at position i refers to the base at position i, whereas a gap-error at position i refers to the gap *after* position i in the read.

In order to identify the erroneous positions in the read, we count how many *UNTRUSTED* extended k-mers cover each base in the read (see Figure 4). For each position, we extend a k-mer by increasing k, base by base, until we find the smallest k'-mer starting at this position which is not being classified as *REPETITIVE*. We also stop the extension if the end of the read is reached. If the k'-mer is classified as *UNTRUSTED*, we increase the count for the positions covered by this k'-mer, this is, positions i to $i+k'-1$ in the read.

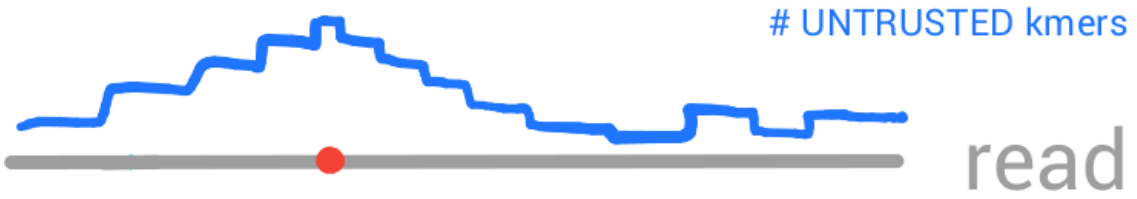

Fig. 4: Positions in the read which are covered by a high number of untrusted k-mers are more likely to be erroneous.

We obtain correction candidates by trying to apply a single base- or gap-error to the read sequence, starting from positions which are covered by the highest number of `UNTRUSTED` k-mers in the read. We accept a correction only if the correction candidate is unambiguous and all k-mers covering the presumably erroneous position become *TRUSTED* after correction (see Figure 5).

Fig. 5: Given a value k and a position i, we only accept a correction candidate for i if it is unambiguous and all extended k-mers starting from $i - k$ to i get classified as *TRUSTED*.

4 Sequence-Specific Error Profile

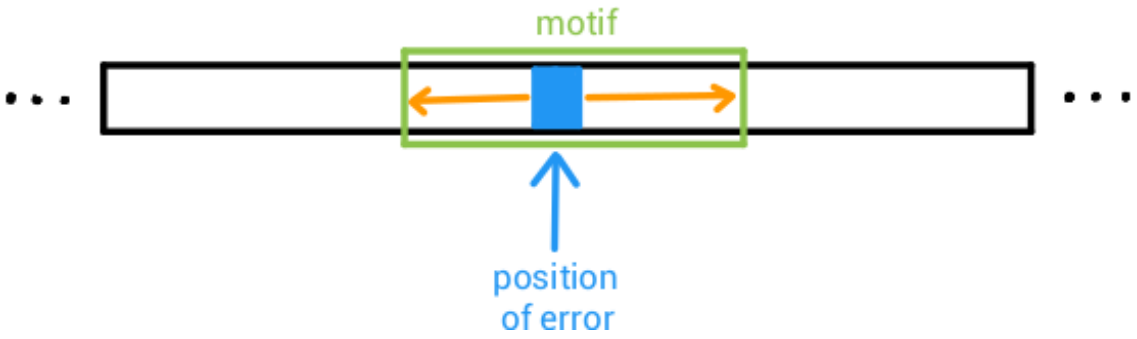

Fig. 6: A motif is a short DNA-sequence surrounding an erroneous position in a read.

In Illumina MiSeq reads, a GGC motif (see Figure 6) increases the likelihood of a substitution error to occur [10]. Shin and Park detect motifs which occur more or less frequently than expected around an error by computing Z-Scores [12]. A positive Z-Score means that an error occurs more frequently than expected within a given motif, while a negative Z-Score means that it occurs less often than expected. Shirn and Park limit their observations to cases where an error occurs exactly in the middle of a motif. We extended their approach by also computing Z-Scores for motifs where the error does not occur exactly in the middle of the motif. To this end, we need to carry out a case distinction that also considers the extreme cases where the error occurs at the first or last base of the motif. For details, see our previous work [6].

5 Evaluation by Mapping to Reference

We evaluate the corrected reads from both, simulated datasets and genome-resequencing datasets by mapping them to the reference genome which *is* known in these cases. We detect errors by mapping the uncorrected original reads to the reference genome, using the widely-used `bwa-mem` alignment tool [5]. For detecting the modifications applied by the error correction algorithm, we align each corrected read with its corresponding original read using the Needleman-Wunsch

algorithm. We apply this procedure instead of directly tracking corrections because this facilitates comparison with other error correction tools. Using this approach no additional information ,other than the original read dataset and the set of corrected reads is required.

After identifying the errors in the original read set as well as the corrected read set, we compute the following evaluation metrics:

- Confusion matrices for base-errors and gap-errors
- F-Scores for all error types
- Unweighted average F-Scores for base-errors and gap-errors
- NMI (Normalized mutual information [13])-Scores for base-errors and gap-errors
- The number of discovered, confused, and newly introduced errors

We use analogous metrics for evaluating the k-mer classification.

6 Implementation

The toolkit is implemented in C++11. For counting k-mers, we use the succinct FM-index implementation by Gog *et al.* [2]. Reads are parsed with the SeqAn library [1].

7 Experimental Results

We evaluated our toolkit with a simulated Ebola Illumina dataset and a set of empirical Escherichia coli str. K-12 substr. MG1655 Illumina sequencing reads (accession number SRR396536). We preprocessed the empirical Illumina dataset by removing adapter sequences using the `cutadapt` [8] tool.

7.1 *K*-mer Classification

For both datasets, we chose a value for k such that a randomly selected k-mer has 1 percent probability to appear in a random genome sequence of size N. We used the formula from the Quake [3] assembler for choosing k:

$$k \approx \frac{\log(200 * N)}{\log(4)}.$$

This gave us $k = 11$ for the simulated Ebola Illumina dataset and $k = 15$ for the empirical E.coli Illumina dataset.

In the simulated Ebola Illumina dataset, we correctly identify all untrusted and all repetitive 11-mers (see Table 1). While our k-mer classification works perfectly in the simulated dataset, the empirical dataset implies that simply using the median G/C coverage bias values is not sufficient for canceling out the effects of coverage bias on k-mer classification (see Table 2). The median coverage bias plots we obtain (see Figures 7 and 8) show decreasing coverage of G/C-rich k-mers in both the simulated and the empirical dataset.

Simulated Dataset

	UNTRUSTED	TRUSTED	REPETITVE
UNTRUSTED	100 %	0 %	0 %
TRUSTED	0.54 %	99.29 %	0.17 %
REPETITIVE	0 %	0 %	100 %

Table 1: Confusion matrix for 11-mer classification in the simulated Ebola Illumina dataset.

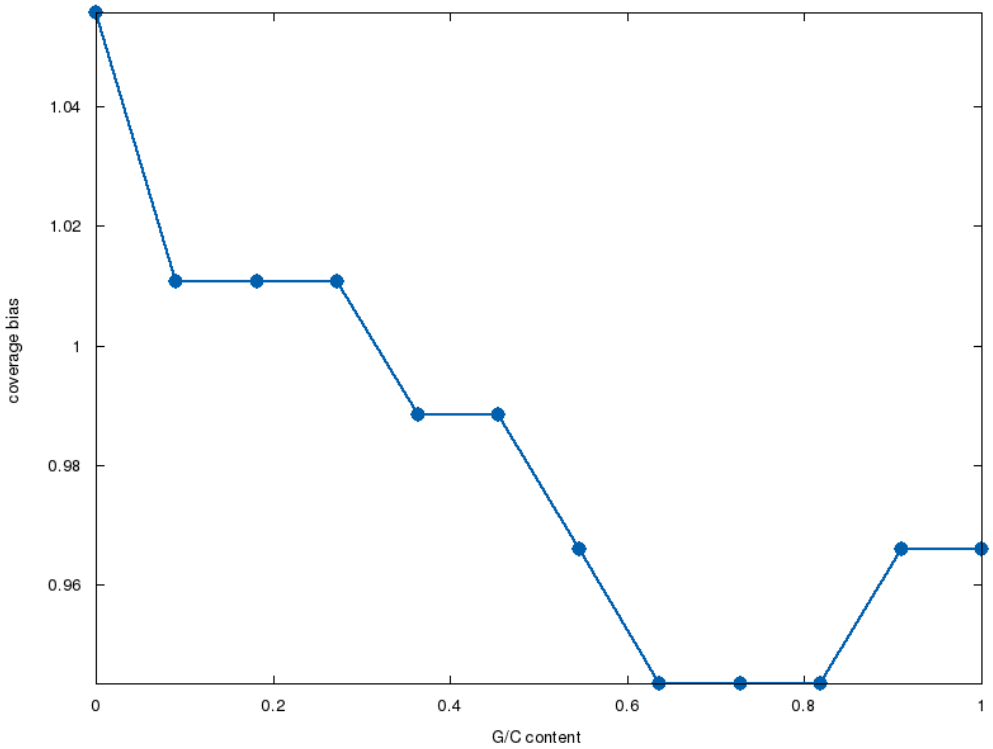

Fig. 7: Median coverage biases for 11-mers in the simulated Illumina dataset.

Empirical Dataset

	UNTRUSTED	TRUSTED	REPETITVE
UNTRUSTED	89.8355 %	1.20574 %	8.9588 %
TRUSTED	2.3638 %	76.139 %	21.4972 %
REPETITIVE	0.0269554 %	7.77019 %	92.2029 %

Table 2: Confusion matrix for 15-mer classification in the real Ebola Illumina dataset.

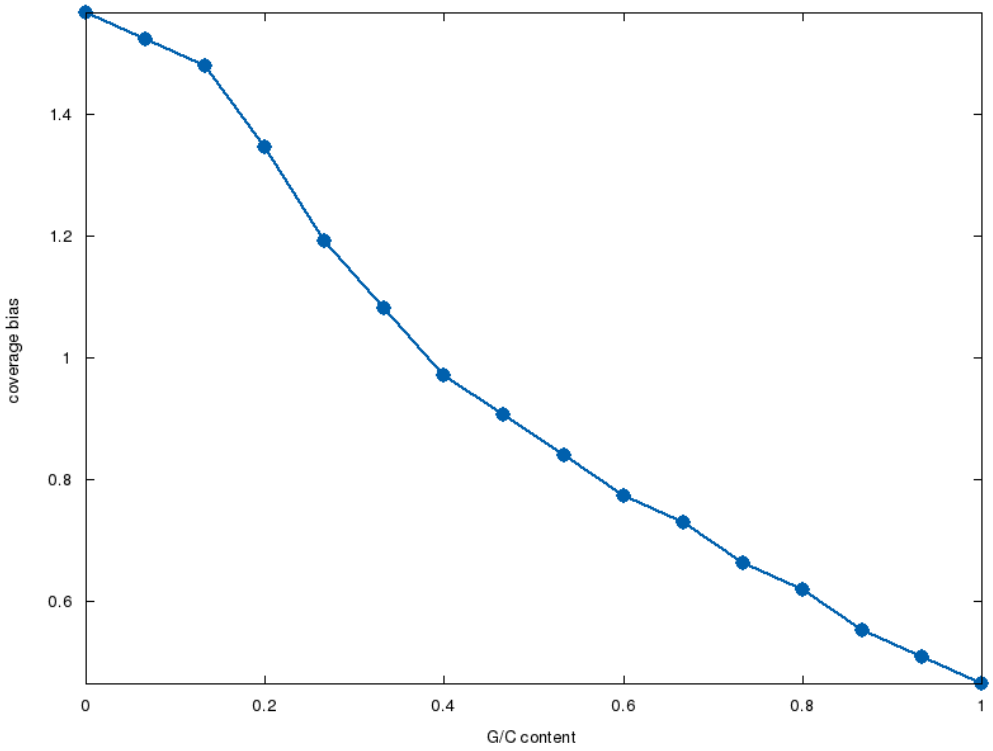

Fig. 8: Median coverage biases for 15-mers in the real Illumina dataset.

7.2 Error Correction

Using our evaluation tool, we compared our current error correction approach to our previous work [6], the Fiona tool [11], the Pollux tool [7], and the Hybrid SHREC tool [9] on the simulated dataset. Our results show that our error correction approach corrects less errors than current state-of-the-art error correction methods (see Figure 9). Compared to our previous work, our updated error correction approach introduces less new errors (see Figure 10).

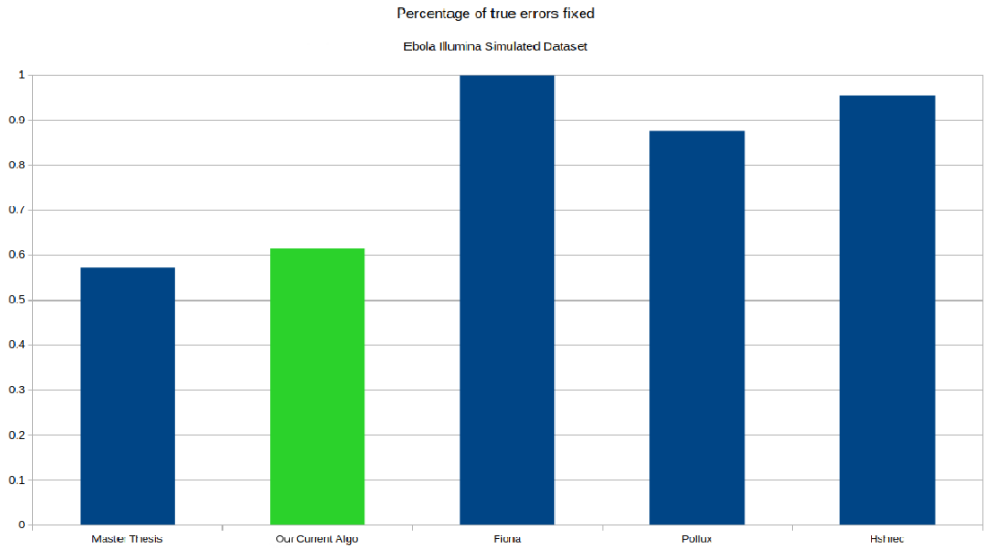

Fig. 9: Percentage of true errors fixed by our previous work, our current work, and the state-of-the-art error correction tools Fiona, Pollux, and HShrec.

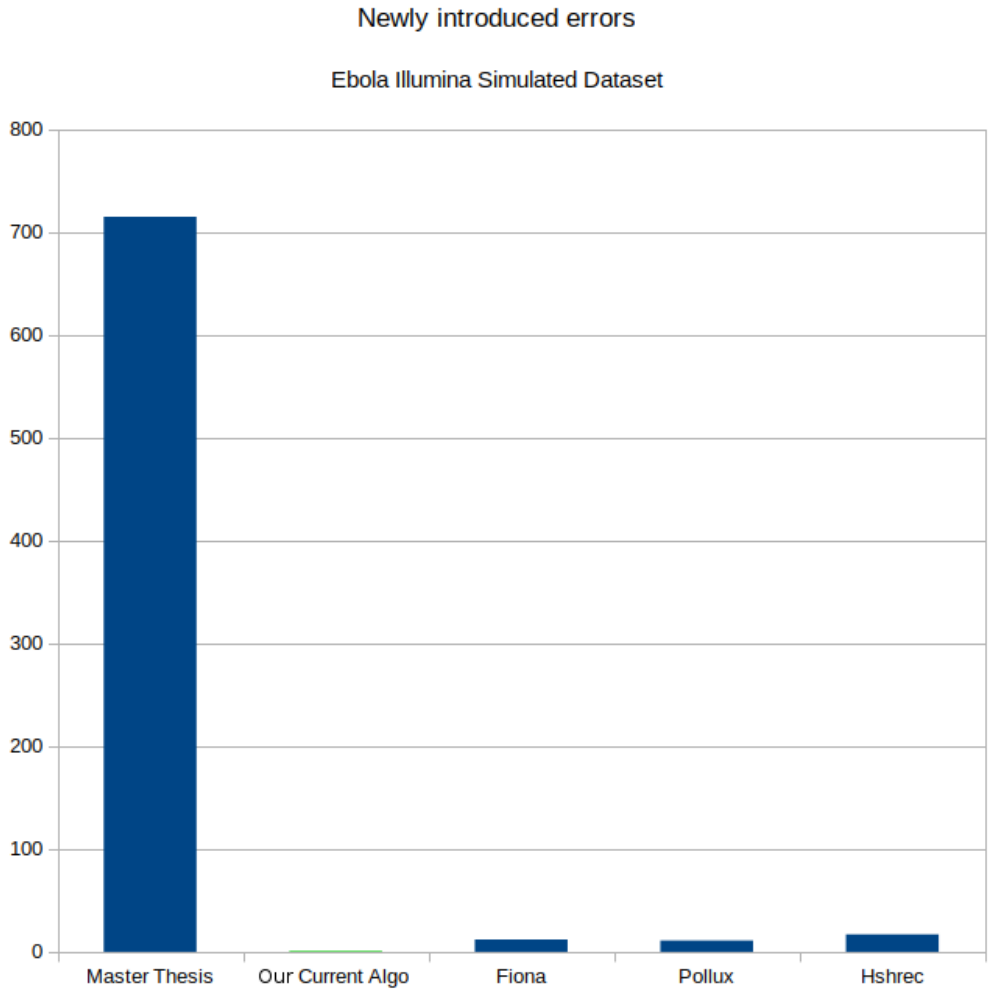

Fig. 10: Total number of newly introduced errors by our previous work, our current work, and the state-of-the-art error correction tools Fiona, Pollux, and HShrec.

7.3 Error Profile

Figure 11 shows the sequence-specific errors we encountered in the simulated and in the empirical dataset. For example, the motif `GGG` in the empirical dataset shows a high correlation with a substitution error (`A` $\rightarrow$ `G`) in the third base of the motif (the corrected motif would be `GGA`). Our results further show that the simulated dataset does not express a high amount of sequence-specific errors as the Z-scores are close to zero.

Z-Score	Motif	Error Type
4.88954	**T**GACG	SUB_FROM_A
4.76725	ACTTC**C**	SUB_FROM_A
4.60939	**A**GATGT	SUB_FROM_A
4.34847	**A**TAACG	SUB_FROM_A
4.0	T**C**ATTA	SUB_FROM_T
3.98949	CGTT**C**	SUB_FROM_G
3.97199	T**C**TTG	SUB_FROM_A
3.95368	CG**A**CG	SUB_FROM_G
3.8243	CGTAC**G**	SUB_FROM_G
3.78517	**A**AGACG	SUB_FROM_C
3.74166	CTT**G**AG	SUB_FROM_A
3.71059	T**C**GC	SUB_FROM_T
3.68951	CT**A**C	SUB_FROM_A
3.67945	**G**GATA	SUB_FROM_T
3.61401	CGATA**G**	SUB_FROM_G
3.61401	CGATA**A**	SUB_FROM_A
3.60555	CGT**G**T	SUB_FROM_T
3.60555	C**C**TTTC	SUB_FROM_A
3.60555	AAT**C**AC	SUB_FROM_A
3.56126	**A**CATCG	SUB_FROM_G

Table 8.7.: The 20 highest Z-scores for ebola illumina simulated

Z-Score	Motif	Error Type
268.119	GG**G**	SUB_FROM_A
266.19	GG**G**	SUB_FROM_T
165.627	**C**TG	SUB_FROM_G
141.773	C**T**G	SUB_FROM_A
131.915	GG**C**	SUB_FROM_A
129.91	GG**C**	SUB_FROM_T
117.812	**C**AG	SUB_FROM_G
115.038	C**A**G	SUB_FROM_T
102.898	CT**G**	SUB_FROM_C
91.4345	GG**A**	SUB_FROM_A
79.6875	CA**G**	SUB_FROM_C
78.2799	TT**T**	SUB_FROM_G
77.8018	TT**T**	SUB_FROM_C
73.8515	GGG**G**	SUB_FROM_A
72.015	GGG**G**	SUB_FROM_T
68.691	GGCGG**G**	SUB_FROM_A
66.5363	GGCGG**G**	SUB_FROM_T
65.6276	GG**G**	SUB_FROM_G
65.5733	GT**T**	SUB_FROM_A
65.3116	A**C**C	SUB_FROM_G

Table 8.4.: The 20 highest Z-scores for SRR396536

Fig. 11: Motifs that are highly associated with the occurrence of sequencing errors. The erroneous base is highlighted in bold.

8 Conclusion and Future Work

During the development process, we observed that adaptively extending k-mers improves our ability to handle repetitive regions and thus, the number of miscorrections is reduced. Moreover, instead of correcting each untrusted k-mer in a read that is isolated from the others, it turned out to be advantageous to consider all (extended) k-mers affected by a correction.

Nevertheless, a substantial research effort is still required for developing an universal sequencing error correction toolkit.

Improvements to k-mer classification:

- Instead of arbitrarily selecting cutoff values, apply the traditional, global cutoff-value approach for k-mer classification of bias-corrected counts (instead of observed counts as it is usually done).
- Use more complex models to infer the coverage bias of a given k-mer in a read. For example, infer a read-based coverage bias by taking the average coverage bias of all k-mers in a read into account.
- Speed up median coverage bias inference.
- Extend the approach to also work with non-haploid genomes and metagenome data.

Improvements to error correction:

- When correcting, allow for more than one error in a k-mer to occur. This is the reason why at present several errors are not corrected, as they are located close to each other within a read.

– Add an option for quality-based read-trimming.
– Take into account other reads containing a given k-mer.
– Provide non-kmer-based error correction approaches.
– Improve handling of multi-base deletion errors.

Acknowledgements

The authors thank Prof. Nick Goldman for the great opportunity of a one-month visit at EBI Hinxton and the Klausch Tschira Foundation who made this visit possible.

The authors further thank Pierre Barbera, Johannes Bechberger, Vladimir Benes, Jane Charlesworth, William Coleman-Smith, Lucas Czech, Diego Darriba, Mat Davis, Tomas Fitzgerald, Tomas Flouri, Paschalia Kapli, Alexey Kozlov, Moritz von Looz, Khouloud Madhbouh, John Marioni, Tim Massingham, Iain Moal, Benoit Morel, Tobias Rausch, Eric Rivals, Jossy Sayir, Dora Serdari, Asif Tamuri, and Kevin Zerr for helpful discussions.

Funding

This work has been supported by the Klaus Tschira Foundation.

References

1. Andreas Döring, David Weese, Tobias Rausch, and Knut Reinert. Seqan an efficient, generic c++ library for sequence analysis. *BMC bioinformatics*, 9(1):11, 2008.
2. Simon Gog, Timo Beller, Alistair Moffat, and Matthias Petri. From theory to practice: Plug and play with succinct data structures. In *13th International Symposium on Experimental Algorithms, (SEA 2014)*, pp. 326–337, 2014.
3. David R Kelley, Michael C Schatz, and Steven L Salzberg. Quake: quality-aware detection and correction of sequencing errors. *Genome biology*, 11(11):R116, 2010.
4. 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.
5. Heng Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*, 2013.
6. Sarah Lutteropp. Error-profile-aware correction of next generation sequencing reads. Master's thesis, Karlsruhe Institute of Technology, Informatics Institute, 2017.
7. Eric Marinier, Daniel G Brown, and Brendan J McConkey. Pollux: platform independent error correction of single and mixed genomes. *BMC Bioinformatics*, 16(1):10, 2015.
8. Marcel Martin. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal*, 17(1), 2011.
9. Leena Salmela. Correction of sequencing errors in a mixed set of reads. *Bioinformatics*, 26(10):1284–1290, 2010.
10. Melanie Schirmer, Umer Z Ijaz, Rosalinda D'Amore, Neil Hall, William T Sloan, and Christopher Quince. Insight into biases and sequencing errors for amplicon sequencing with the illumina miseq platform. *Nucleic Acids Research*, p. e37, 2015.
11. Marcel H Schulz, David Weese, Manuel Holtgrewe, Viktoria Dimitrova, Sijia Niu, Knut Reinert, and Hugues Richard. Fiona: a parallel and automatic strategy for read error correction. *Bioinformatics*, 30(17):i356–i363, 2014.
12. Sunguk Shin and Joonhong Park. Characterization of sequence-specific errors in various next-generation sequencing systems. *Molecular BioSystems*, 12(3):914–922, 2016.
13. Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854, 2010.
14. Xiaohong Zhao, Lance E Palmer, Randall Bolanos, Cristian Mircean, Dan Fasulo, and Gayle M Wittenberg. Edar: an efficient error detection and removal algorithm for next generation sequencing data. *Journal of Computational Biology*, 17(11):1549–1560, 2010.