IFT6299 (Topics in bioinformatics) — Development and Evaluation of a K-mer Based Error Correction Tool for DNA Sequencing Data

Université de Montréal
Prof: Miklos Csuros

March 2024

Abstract

Accurate DNA sequencing is critical for reliable genomic analysis, yet it often faces challenges from base-call errors. This report presents a error correction tool that dynamically adapts to error patterns using k-mer frequency analysis and an adaptive weighted adjustment mechanism. Our tool integrates Jellyfish for efficient k-mer counting and employs a probabilistic approach inspired by reinforcement learning to optimize base correction decisions, thereby reducing computational overhead. The effectiveness of this approach is demonstrated through detailed performance evaluation on trimmed FASTQ files from the Agrobacterium vitis AT6 genome. These results indicate substantial improvements in data accuracy, showcasing the tool's potential to facilitate error correction in dna sequencing data.

1 Introduction

High-throughput sequencing technologies, while cost-effective, are prone to errors that can significantly impact downstream analyses. Traditional error correction methods often struggle with the computational demands of large datasets and the nuanced nature of error patterns. Our approach utilizes k-mer frequency distributions to identify potential errors and employs a novel correction strategy that adapts based on correction efficacy, thereby refining its performance iteratively.

2 Methodology

2.1 K-mer Frequency Analysis

The tool utilizes Jellyfish[1] for efficient k-mer counting, which conserves disk space by avoiding the need for large in-memory data structures. This feature is particularly advantageous when processing genomes with extensive repetitive sequences. Jellyfish performs this by using a disk-based hash table to manage and count k-mers directly from the FASTQ files. The resulting counts are stored in a binary format that Jellyfish can later convert (or dump) to a human-readable format, facilitating subsequent analysis. This method allows for rapid processing of large datasets and provides the raw data necessary for constructing the frequency spectrum of k-mers observed in the sequencing data.

2.2 Error Correction Algorithm

Objective: This section introduces the concept of weighted adjustment as implemented in our error correction tool. Our initial approach in the script involved exhaustive searches across each nucleotide to determine the optimal corrections using scoring mechanisms. This direct method proved computationally intensive for large genomic datasets due to significant time and space requirements. In contrast, the weighted adjustment approach, inspired by the principles of ***Q-learning*** [2] a model of reinforcement learning provides a dynamic and computationally efficient alternative for optimizing base correction decisions during sequencing read processing. This advanced approach not only reduces the computational burden but also dynamically adapts to the evolving characteristics of sequencing error patterns, enhancing correction accuracy over time.

For a detailed implementation of the correction algorithm, see Algorithm 7.2 in the Appendix.

2.2.1 Probabilistic Decision Making

Correction of each base within a k-mer is guided by a probabilistic model that dynamically adjusts based on the outcomes of previous corrections. This approach draws inspiration from the principles of **Q-learning**, a model of reinforcement learning, where actions *(in this case, base corrections)* are taken based on a policy that maximizes the expected value of the reward function. Here, rewards are conferred for successful corrections (*leading to an increase in corresponding weights*), and penalties are applied for unsuccessful attempts (*resulting in decreased weights*). Each base position within the k-mer acts as an agent making decisions that are influenced by an accumulated history of correction successes and failures, analogous to the way ***Q-learning*** agents update their policy based on past experiences.

2.2.2 Weight Adjustment Formula

We begin with a uniform distribution of weights, where $w_{i,j} = 1$ for all i in the range from 1 to k (the length of the k-mer), and for each nucleotide j in {A, C, G, T}. This initial condition represents an unbiased approach where no preference is given to any nucleotide at the start of the error correction process.

Mathematically, this can be represented as:

$$w_{i,j} = 1 \quad \forall i \in [1, k],\ j \in \{A, C, G, T\}$$

where $w_{i,j}$ is the weight for the j-th nucleotide at the i-th position within the k-mer, and k is the length of the k-mer.

Weight Update on Correction Outcome:

- Success Adjustment: When a correction is deemed successful (i.e., replacing a nucleotide increases the frequency of the k-mer to above the rare threshold i_0), the weight of the corrected base at that position is increased. Mathematically:

  $$w'_{i,j} = w_{i,j} \times \alpha$$

  where $\alpha > 1$ is the success adjustment factor (commonly 1.1).

- Failure Adjustment: Conversely, if a correction does not result in a k-mer frequency increase above i_0, the weight is decreased:

  $$w'_{i,j} = w_{i,j} \times \beta$$

  where $\beta < 1$ (typically 0.95).

- Normalization: After adjustment, weights are normalized to ensure they sum to 1 across all nucleotides at each position:

  $$w''_{i,j} = \frac{w'_{i,j}}{\sum_{j=1}^{4} w'_{i,j}}$$

2.2.3 Application of Weights in Correction Decisions

Epsilon-Greedy Strategy[3]: Incorporate an epsilon-greedy algorithm to balance exploitation (using the best-known correction based on weights) and exploration (trying less likely corrections). Define the probability of random correction (exploration) as ϵ (usually small, e.g., 0.05):

Choose random base with probability ϵ, otherwise choose base j maximizing $w_{i,j}$.

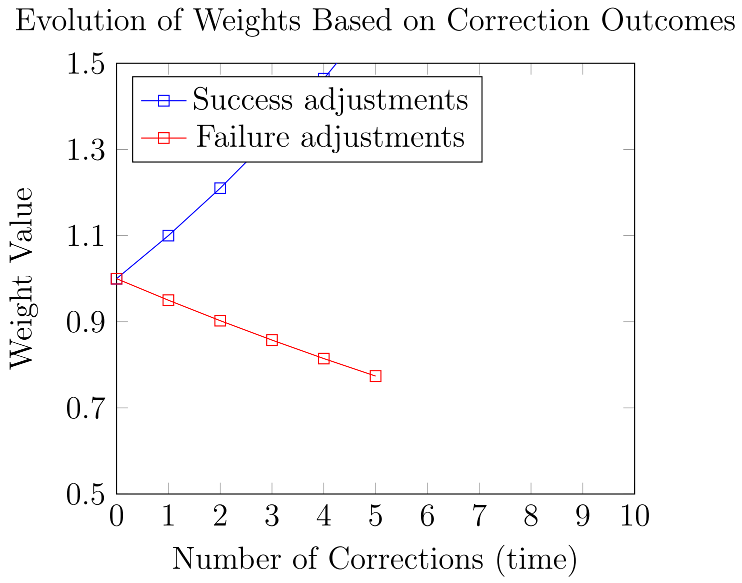

Figure 1: Graphical representation of weight adjustments following correction outcomes. Blue represents weight increases with successful corrections, and red represents weight decreases with unsuccessful ones.

2.3 Algorithm Complexity

Computational Analysis: The computational complexity of the error correction algorithm plays a critical role in its efficiency and scalability. For each read, the complexity of the process is determined by multiple factors, primarily influenced by the number of reads n and the length of each read m. This complexity can be expressed as $O(n \times m)$, reflecting the three primary operations:

1. **Sequential k-mer Processing:** For each read, k-mers are generated and evaluated in a sequential manner. Since each read of length m contains approximately $m-k+1$ k-mers (where k is the length of the k-mer), processing all k-mers across all reads constitutes a significant portion of the computational effort. The need to handle each k-mer individually for both counting and correction assessment multiplies the computational load linearly with the length of the read.

2. **Dynamic Weight Adjustments:** Each correction decision is influenced by a dynamically maintained weight matrix, which needs to be updated based on the outcomes of correction attempts. These updates occur whenever a k-mer that falls below the frequency threshold i_0 is encountered and a correction is attempted. The complexity of updating the weight matrix is dependent on the number of bases in the k-mer (k), and this update needs to be performed potentially for each k-mer in every read, adding a further layer of computational effort.

3. **Complexity Amplifiers:** Additional complexity arises from the need to calculate the reverse complement of each k-mer and to integrate these counts into the overall frequency analysis. This effectively doubles the number of k-mers to be processed and weighed for potential correction.

Implications: The $O(n \times m)$ complexity indicates that the algorithm's performance is linearly dependent on the total number of nucleotides processed. This linear relationship is crucial for ensuring that the algorithm scales effectively with increasing dataset sizes.

3 Results and Analysis

3.1 Correction Outcomes

Detailed metrics from the error correction evaluations on different datasets are presented below. This table summarizes the total number of corrections made, the percentage of reads corrected, and the changes in the counts of rare k-mers before and after correction.

Table 1: Summary of correction outcomes for each dataset.

Dataset	**Total Corrections**	**% Corrected Reads**	**Rare K-mers Before**	**Rare K-mers After**	**% Change**
AT-6 R1 Paired	463,220	16.42	45,956,711	42,433,555	-7.67
AT-6 R2 Paired	890,300	31.56	161,575,244	156,811,066	-2.95
AT-6 R1 Unpaired	222	0.34	6,460,952	6,460,686	-0.0041
AT-6 R2 Unpaired	557	0.83	10,022,708	10,021,230	-0.015

3.1.1 Understanding the Final Weights

The final weights for nucleotides across different datasets reveal crucial insights into the correction mechanics. The weights are particularly skewed towards thymine (T), with datasets showing a predominance of this nucleotide in the corrected k-mers. Here, we present a visualization of these final weights to better illustrate their distribution:

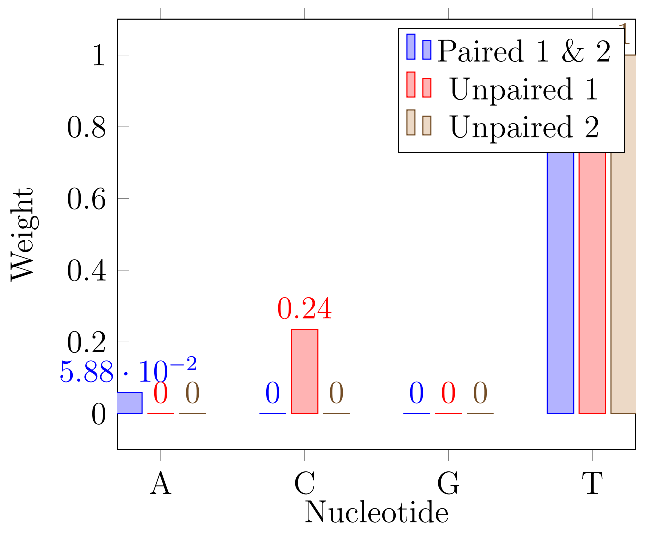

Figure 2: Distribution of final weights for nucleotides across different datasets.

The final weights observed for each dataset reveal some critical dynamics about the correction process:

- **Nucleotide Preference:** The weights heavily favor thymine (T) in most datasets, indicating that the script often replaces other nucleotides with thymine during the correction process. This could suggest that thymine is less frequently involved in erroneous k-mers, or it could be an artifact of the genomic composition of the particular datasets being analyzed.
- **Lack of Diversity in Corrections:** The absence or near-zero weights for certain nucleotides (e.g., cytosine (C) and guanine (G)) in some datasets suggests a potential bias or inefficiency in the correction process. This may be due to various factors, such as the fact that our program does not explore the correction space sufficiently, or that it is too conservative in its correction strategy, or simply

that these ignored bases are not effective enough to be used as correct corrective selection..

3.2 Post-Correction Sequence Analysis

Following the k-mer based corrections, the quality of the corrected FASTQ files was evaluated using BWA[4] and Samtools[5] to align the reads to the reference genome (contigs.fasta). This process included generating a reference genome index, aligning reads to the reference, and generating .sam files for compatibility with subsequent analysis.

3.2.1 Alignment Process and Quality Control

- **Reference Genome Indexing:** The reference genome was indexed to facilitate rapid read alignment.
- **Read Alignment and SAM File Generation:** Reads were aligned to the reference genome, and the alignments were saved in SAM format, which is compatible with further processing tools.
- **Conversion to BAM and Indexing:** The SAM files were converted to BAM format for efficient storage and quick access. BAM files were sorted and indexed to enhance data retrieval performance during downstream analyses.

3.2.2 Quality Control Statistics

The following table presents a FLAGSTAT report summarizing the number of reads passing quality control checks before and after correction:

Table 2: FLAGSTAT quality control report before and after k-mer based correction.

Dataset	**QC-passed Before**	**QC-failed Before**	**QC-passed After**	**QC-failed After**
AT6 paired 1	2,848,827	0	2,849,077	0
AT6 paired 2	2,859,835	0	2,860,859	0
AT6 unpaired 1	65,038	0	65,038	0
AT6 unpaired 2	67,195	0	67,195	0

3.3 Comparative Analysis

The comparative analysis reveals significant variations in correction efficacy across different types of sequencing data, specifically between paired and unpaired reads.

- **High Correction Volumes:** Datasets like AT-6 R1 Paired, which had a significant reduction in rare k-mers, show that the script can effectively identify and correct errors, especially in datasets where extensive corrections are applied.

- **Minor Improvements in Some Cases:** The minimal changes in the unpaired datasets suggest that the script may struggle with datasets where the errors are not as clearly defined by k-mer frequency, or where the initial quality of sequencing is lower.

The **FLAGSTAT** quality control results provide further corroboration of this observation, demonstrating that there is no significant difference in the QC-passed reads before and after correction for unpaired datasets. The quality control statistics before and after correction, particularly the lack of change in the number of QC-failed reads, indicate that the script maintains the overall read integrity while performing corrections. This is of significant importance, as it suggests that the correction process does not introduce additional errors or artifacts that could compromise downstream analyses. Concurrently, the number of reads that passed the quality control (QC) threshold increased following the correction.

4 Challenges

The script, while effective in certain scenarios, faces several challenges that may hinder its broader applicability and efficiency:

- **Bias in Correction Strategy:** The observed bias towards thymine in the weight adjustment indicates a potential overfitting to specific error types or sequencing conditions, which may not generalize well across different datasets or sequencing technologies.

- **Handling of Complex Error Patterns:** The script primarily utilizes k-mer frequency to identify and correct errors. This approach may not capture more complex error patterns that involve structural variations or low-frequency but significant errors.

- **Dependency on External Tools:** While Jellyfish effectively reduces memory usage during k-mer counting by utilizing a disk-based hash table, challenges remain in managing memory efficiently in the subsequent steps of the error correction process. The current implementation may still encounter memory overhead issues when storing and manipulating large sets of k-mer data and weights in Python.

5 Future Directions

5.1 Mathematical Enhancements

To address the current limitations and enhance the script's capability, the following mathematical enhancements are proposed:

- **Integration of Statistical Models:** Incorporating statistical models such as Hidden Markov Models (HMMs) or Bayesian Networks could help in understanding the probabilistic nature of sequencing errors better, allowing for more nuanced corrections that consider not only the frequency but also the likelihood of errors given the genomic context.

- **Advanced Weight Optimization Techniques:** Implementing optimization algorithms such as gradient descent or evolutionary algorithms to find the optimal set of weights for nucleotide correction, which could dynamically adapt to different error distributions and sequencing conditions.

5.2 Algorithm Improvements

Principal improvements to the algorithm that could significantly boost its performance and applicability include:

- **Adaptive Learning Mechanisms:** Introducing machine learning techniques such as reinforcement learning to adapt the correction strategies based on feedback from the correction outcomes. This approach would allow the system to learn the most effective strategies over time and adjust its parameters for optimal performance.
- **Parallel Processing Implementation:** Redesigning the algorithm to take advantage of multi-threading and parallel processing capabilities of modern computing environments. This would address scalability issues and reduce the time required to process large datasets.
- **Enhanced Error Profiling:** Developing a more sophisticated error profiling mechanism that can identify different types of errors (e.g., substitutions, insertions, deletions) and apply targeted corrections based on the specific error characteristics and their impact on downstream analyses.

6 Conclusion

The script has demonstrated significant effectiveness in contexts where error patterns are predominantly characterized by rare k-mers, leading to substantial reductions in their counts across several datasets. This capability highlights the tool's utility in pinpointing and rectifying clear-cut errors within sequencing data, which is especially crucial for datasets where such errors are prevalent. However, the observed heavy bias towards thymine and the inadequate correction observed in certain datasets suggest a need for more diverse and context-aware correction strategies.

Moreover, the script maintains robust data integrity post-correction, as evidenced by stable quality control metrics. This attribute is critical, suggesting that the tool can be seamlessly integrated into genomic workflows without compromising the quality of the data, thereby ensuring reliable outcomes for downstream analyses. Despite these strengths, there is considerable potential for further improvements and adaptations. The tool could significantly benefit from integrating machine learning approaches that dynamically learn and adapt based on a broader range of error characteristics. Such adaptive learning could refine correction strategies in real-time, meeting the evolving demands of genomic research and potentially broadening the script's applicability to a more diverse array of genomic datasets.

7 Supplementary Material

Additional outputs, data files, and related resources generated by this error correction tool are available for review and further analysis. These resources are hosted on a secure institutional repository to ensure data integrity and accessibility for authorized users.

7.1 Access to Supplementary Files

The supplementary materials can be accessed through the following University of Montreal's SharePoint link. Please note that access might be restricted to specific institutional members or project collaborators.

SharePoint Link to Supplementary Files:

`https://udemontreal-my.sharepoint.com/:f:`
`/g/personal/raphael_avocegamou_umontreal_ca/`
`EkSOn0RzIE9KsPaXYW7WziYBwEp3k_WY2laSROSO-uaHng?e=hvKKQg`

Please ensure that you have the necessary permissions to access these files.

7.2 Contents of Supplementary Files

The supplementary files include:

- Corrected FASTQ files post-error correction.
- Logs and diagnostic outputs detailing the correction process.
- Comparative analysis reports and performance benchmarks.

These resources provide a comprehensive view of the tool's performance and are intended to facilitate further research and validation of the error correction methodologies employed.

Appendices

Pseudocode

This appendix contains the pseudocode for essential algorithms used in the error correction tool as discussed in the Methodology section of this report.

Algorithm 1 Correct Read Algorithm

```
1: procedure CorrectRead($read, kmer\_counts, k, weights, quality\_scores, i0$)
2:     $corrected\_read \leftarrow \text{list}(read)$
3:     $num\_corrections \leftarrow 0$
4:     for $i \in [0, \text{len}(read) - k]$ do
5:         $kmer \leftarrow read[i : i + k]$
6:         $rc\_kmer \leftarrow \text{ReverseComplement}(kmer)$
7:         $combined\_count \leftarrow kmer\_counts[kmer] + kmer\_counts[rc\_kmer]$
8:         if $combined\_count < i0$ then
9:             $min\_q\_index \leftarrow$ index of minimum quality score in $quality\_scores[i : i + k]$
10:            $min\_q\_base \leftarrow kmer[min\_q\_index]$
11:            if $min\_q\_base \in \{A, C, G, T\}$ then
12:                $best\_base \leftarrow \text{SelectBase}(min\_q\_index, weights, min\_q\_base)$
13:                if $best\_base \neq min\_q\_base$ then
14:                    $new\_kmer \leftarrow kmer$ with $min\_q\_base$ replaced by $best\_base$
15:                    $new\_rc\_kmer \leftarrow \text{ReverseComplement}(new\_kmer)$
16:                    $new\_combined\_count \leftarrow kmer\_counts[new\_kmer] + kmer\_counts[new\_rc\_kmer]$
17:                    if $new\_combined\_count > i0$ then
18:                        $corrected\_read[i + min\_q\_index] \leftarrow best\_base$
19:                        AdjustWeights($weights, min\_q\_index$, base index of $best\_base$, True)
20:                        $num\_corrections \leftarrow num\_corrections + 1$
21:                        break
22:                    else
23:                        AdjustWeights($weights, min\_q\_index$, base index of $min\_q\_base$, False)
24:                    end if
25:                end if
26:            end if
27:        end if
28:    end for
29:    return $corrected\_read, num\_corrections$
30: end procedure
```

References

[1] G. Marçais and C. Kingsford, "A fast, lock-free approach for efficient parallel counting of occurrences of k-mers," *Bioinformatics*, vol. 27, pp. 764–770, Mar. 2011.

[2] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Mach. Learn.*, vol. 8, pp. 279–292, May 1992.

[3] B. C. Stadie, S. Levine, and P. Abbeel, “Incentivizing exploration in reinforcement learning with deep predictive models,” 2015.

[4] H. Li and R. Durbin, “Fast and accurate short read alignment with Burrows-Wheeler transform,” *Bioinformatics*, vol. 25, pp. 1754–1760, July 2009.

[5] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and 1000 Genome Project Data Processing Subgroup, “The sequence Alignment/Map format and SAMtools,” *Bioinformatics*, vol. 25, pp. 2078–2079, Aug. 2009.