Noise2read: Accurately Rectify Millions of Erroneous Short Reads Through Graph Learning on Edit Distances

Pengyao Ping[1], Shuquan Su[1,2], Xinhui Cai[1], Tian Lan[1], Xuan Zhang[1], Hui Peng[2], Yi Pan[2], Wei Liu[1], Jinyan Li[2,*]

[1] *School of Computer Science, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney 2007, Australia*

[2] *Faculty of Computer Science and Control Engineering, Shenzhen University of Advanced Technology, Shenzhen 518000, China*

[*] Corresponding author.

E-mail: lijinyan@suat-sz.edu.cn (Li J).

Running title: *Ping P et al / Noise2read*: *Turn Noise to Signal for Short Read*

Total number of words (from “Introduction” to “Discussion”): 9366.

Total number of letters for the article title: 92.

Total number of letters for the running title: 50.

Total number of words for Abstract: 239.

Total number of Keywords: 5.

Total number of figures: 9.

Total number of tables: 6.

Total number of supplementary figures: 25.

Total number of supplementary tables: 20.

Total number of supplementary files: 4.

Total number of references: 62.

Abstract

Although the per-base error rate of short-read sequencing data is very low at 0.1%–0.5%, the percentage/probability of erroneous reads in a dataset can be as high as 10%–15% or in the number millions. As current methods correct only some errors while introducing many new errors, we solve this problem by turning erroneous reads into their original states, without bringing up any non-existing reads to keep the data integrity. The novelty is originated in a computable rule translated from polymerase chain reaction (PCR) erring mechanism that: a rare read is erroneous if it has a neighbouring read of high abundance. With this principle, we construct a graph to link each pair of reads of tiny edit distances to detect a solid part of erroneous reads; then we consider these pairs of reads of tiny edit distances as training data to learn the erring mechanisms to identify possibly remaining hard-case errors between pairs of high-abundance reads. The proposed approach, noise2read, is competent to handle the rectification of erroneous reads from short-read sequencing data whenever PCR is involved. Compared with state-of-the-art methods on tens of evaluation datasets of unique molecular identifier (UMI) based ground truth, noise2read performs significantly better on 19 metrics. Case studies found that noise2read can greatly improve short-reads quality and make substantial impact on genome abundance quantification, isoform identification, single nucleotide polymorphisms (SNP) profiling, and genome editing efficiency estimation. Noise2read is publicly available at https://github.com/JappyPing/noise2read and https://ngdc.cncb.ac.cn/biocode/tool/7951.

KEYWORDS: Short reads error correction; Polymerase chain reaction erring; Graph of reads; Edit distance of two reads; Machine learning

Introduction

Next generation sequencing (NGS) techniques and platforms have dramatically changed the world of genomics and computational biology [1–3]. High throughput DNA sequencing has enabled large-scale whole-genome sequencing and gene-targeted sequencing; NGS-based RNA-seq has provided ever higher coverage and sharper resolution of dynamic transcriptomes for a wide range of applications such as isoform discovery, differential gene expression analysis, alternative gene splicing, and allele-specific expression profiling [2]. However, NGS inevitably self-made sequencing errors including base deletions, insertions and substitutions at various steps like sample handling, library preparation, polymerase chain reaction (PCR), and/or at the base calling step [4,5]. Although the erring rate is estimated very low at 0.1%–0.5% per base in Illumina short-read sequencing, huge numbers of erroneous bases have been generated and stored at every sequencing dataset (*e.g.*, about 197,402 base errors in a miRNA-sequencing dataset ERR187525, and about 997,020 base errors in a pair-end whole-genome sequencing dataset SRR22085311 which have been found through this study). As these mistaken bases are randomly distributed across possibly all the reads in a dataset, the percentage/probability of *erroneous reads* in a dataset can be very high (*e.g.*, as high as 10%–15%).

Suppose the per-base erring probability is estimated as p at a sequencing platform, and assume these erring events are independent at all the base positions in a read, then the probability $p_{error}(r)$ of a read r containing one or multiple base errors is given by

$$p_{error}(r) = \sum_{i=1}^{L} \binom{L}{i} p^i (1-p)^{(L-i)} = 1 - (1-p)^L \quad (1)$$

where $L = \|r\|$, the length of read r. If $p = 0.1\%$ and L = 100, then $p_{error}(r) =$ 9.52%. In other words, the percentage of erroneous reads in a dataset is about 9.52% when the per-base erring rate is estimated as 0.1% and the length of reads $L = 100$ bp. If the per-base erring rate p is estimated as 0.15%, then there are about 13.94% of erroneous reads in the dataset.

This is a fundamental issue previously unrecognized concerning the high percentages of erroneous reads in NGS datasets. These erroneous reads are usually treated as data noise implicitly or explicitly excluded for downstream data analysis such as *de novo* genome/transcriptome assembly and differential gene expression profiling

[6,7]. Or these erroneous reads are un-purposely considered as genuine true reads in the data analysis which may have led to inaccurate or wrong conclusions. To restore the huge missing value of these high percentages of erroneous reads in each sequencing dataset, it is highly demanded to do accurate rectification of all these errors, as opposed to treating them as noise removal, to boost the data quality and integrity so as to improve the downstream applications.

One of the main sources of the sequencing errors is from PCR, a technique that makes fast duplications of small segments of DNA which has been used by NGS to amplify the fragmented DNA/RNA molecules for effective sequencing. Most of the time, PCR makes perfect copies of the fragmented segments of DNA/RNA, but occasionally it introduces base-pair substitutions, deletions, insertions, or even yields new hybrid sequences during template switching [8]. Thus, after the PCR amplification, one or two copies in the duplications of a DNA segment may show inconsistent bases. **Figure 1**A illustrates how base errors arise when amplifying one DNA template during PCR amplification. PCR errors not only occur in the library preparation but also during sequencing processes such as clonal molecules [5]; Figure 1B is an example that depicts how errors are introduced in the process of bridge amplification during Illumina sequencing. Such PCR erring incidents are then inherited by NGS's base calling step that converts a nucleotide sequence into a digital string (named a read). The conversion is not 100% accurate as well, similar to PCR introducing minor mistakes (Figure 1C) [4,9]. Therefore, sequencing errors can occur in various ways. However, it is almost certain that an erroneous read will appear at low frequency if the error occurred at the late cycles of PCR. This is because the probability of the same error occurring at the same position is extremely low, especially in 200–300 bp reads.

Efficient detection of these erroneous reads from a dataset of hundreds of millions of reads is challenging. First, some low-frequency rare reads are genuine reads not containing any sequencing errors. This is attributed to the uneven PCR amplification rates at different segments of the DNA — poorly amplified molecules will be sequenced to a lesser extent than the highly amplified molecules [10,11]. Second, an amplified segment after PCR erring may become identical to a high-frequency molecule. As a result, for two highly similar high-frequency reads (A and B), it is impossible to determine, without machine learning of PCR erring mechanisms, whether B represents an erroneous amplification of A or vice versa.

We construct a graph $rG(R)$ using the unique reads $r_1, r_2, \ldots, r_n$ along with their frequencies from a read dataset R (a multiset of reads) to detect erroneous reads under the sophisticated help of graph-based machine learning. Let $freq(r)$ represent the abundance level or the frequency of a read r, or the number of copies of r in the sequencing data. For each of the unique reads in R, we represent it as a node in the graph and label the node with the read's frequency. There is an edge $e_{(i,j)}$ between node r_i and node r_j if the *edit distance* between read r_i and read r_j is 1 or 2. Specifically, when searching for edges with an edit distance of 2, only substitutions are taken into account. A read u is a neighbouring read of read v if there is an edge between them. As understood from the PCR erring mechanism in NGS, the pairing of two neighbouring reads u and v implies that a copy of u is a wrongly amplified/sequenced copy of the v molecule, or a copy of v is a wrongly amplified/sequenced copy of the u molecule, or both. When $freq(v)$ is low while $freq(u)$ is high, we rectify the erroneous read v by removing this node from the graph, while increase $freq(u)$ by $freq(v)$. That is, we turn the "noise" read (*i.e.*, a read that contains erroneous bases) v (low-frequency rare read) into its normal state u. We denote such a set of erroneous reads in the graph as edit-erring-READS and the isolated nodes with high frequencies as error-free-READS. Notably, "noise" refers to erroneous bases contained in reads in this study. Our correction procedure turns individual base errors into their correct state (signal) without changing other bases in the reads, and the rectified reads can be used for any downstream applications.

We use a small edit distance of 1 or 2 to define the edges of the graph because those erroneous reads containing one mistaken base or two constitute the majority of the total erroneous reads in the dataset. The majority percentage is given by

$$error\%(p, 1, Emax) = \frac{\sum_{i=1}^{Emax} \binom{L}{i} p^i (1-p)^{(L-i)}}{1-(1-p)^L} \quad (2)$$

where p is the base erring probability, $Emax$ is a maximum edit distance allowable to define an edge. If $L = 100$, $p = 0.1\%$, $Emax = 2$, then $error\%(p, 1, Emax) = 99.84\%$. This indicates that 99.84% of all the possible erroneous reads in the dataset are those reads containing one base error or two (*Emax*).

The second challenge in the correction of erroneous reads in the graph $rG(R)$ is to deal with the situation when a low-frequency read is linked to multiple high-frequency reads, and/or two (or more) high-frequency reads are linked each other in the graph

(denoted as ambiguous errors). Hence, we model the situations as a classification problem and use machine learning techniques to predict whether the duplications of a high-frequency read contain or not contain wrongly sequenced copies of its neighbouring high-frequency reads.

This is a novel classification problem not formulated in any literature. In this work, we use edit-erring-READS and error-free-READS as training data and extract multiple features of different dimensions from the data and then utilize an optimized gradient boosting classifier of the extreme gradient boosting (XGBoost) [12] to make the prediction under a supervised learning framework. As the training data is $rG(R)$-specific, the prediction model can learn the inherent erring patterns of each specific sequencing platform that conducts the specific biomolecular samples' sequencing. Therefore, our machine learning approach is competent to handle the rectification of erroneous reads that have a length less than 300 bp produced by any PCR-involved single/pair-end DNA/RNA sequencing, whole-genome sequencing, miRNA-sequencing, or synthetic sequencing regardless of the difference in the platforms or in the biomolecular samples.

Method

Overview of noise2read algorithm

We present an error correction method to improve the short-read sequencing data quality by turning millions of erroneous short reads into their normal state through graph learning on edit distances between reads. We name our method "noise2read". As introduced above, its novelty sits in the computable rule translated from PCR erring mechanism: a rare read is erroneous if it has a neighbouring read of high abundance. With this principle, we construct a graph to link each pair of reads of a small edit distance to detect a substantial part of erroneous reads in the graph. Then we take them as training data to learn the platform-specific erring mechanism to identify possibly remaining hard-case errors between pairs of frequent reads in the graph, namely specific training data is used at different platforms.

Noise2read is a progressive three-stage error correction method, and an overview of the workflow of noise2read is illustrated in **Figure 2**. An auto machine learning (AutoML) module is centred in the process of noise2read, which is used multiple times in the different stages for the prediction of ambiguous or amplicon errors. AutoML has

a component for the preparation of training and objective data and has a component for the parameter optimization of the gradient boosting-based classifiers. The first stage (shaded in blue) rectifies low-frequency leaf nodes (genuine errors) and ambiguous errors by a traversal on the 1-nt-edit-distance read graph $1-nt-rG(R_0)$ constructed from the original reads of dataset R_0. Here, every edge in the 1-nt-edit-distance read graph means the edit distance between the two nodes is one nucleotide (*i.e.*, 1 nt). The second stage (shaded in pink) conducts correction of genuine and ambiguous errors at the 2-nt-edit-distance read graph $2-nt-rG(R_1)$ constructed from the first stage corrected dataset R_1. Here, every edge in the 2-nt-edit-distance read graph means the edit distance between the two nodes is two nucleotides (*i.e.*, 2 nt). Particularly, we consider only substitution relationships for constructing 2-nt-edit-distance edges since the majority of NGS data conforms to a consistent read length. The third stage (shaded in yellow) is designed to eliminate specific errors at an updated 1-nt-edit-distance graph $1-nt-rG(R_2)$ only for the amplicon sequencing data but using the same AutoML module for prediction.

Graph $rG(R)$ is often a disconnected graph. For example, nine subgraphs of $rG(D1)$ constructed in the first stage are shown in Figure S1, where $D1$ is a simplified version of SRR1543964. There are many clustered low-frequency leaf reads linked to one high-frequency read, while there also exist edges that link pairs of high-frequency reads. **Figure 3** is a zoomed version with more details about subgraph A in Figure S1, where the high-frequency nodes are highlighted in orange and the low-frequency nodes are highlighted in pink. Every edge in this graph implies that the linked reads have only one base difference. With these sub-graphs, noise2read (1) directly turns those leaf nodes of low-frequency into their high-frequency parent nodes (their normal states r_1, r_2, ..., or r_7); (2) uses the AutoML module to identify the parent node of two low-frequency nodes r_8 and r_9, as these two low-frequency nodes are each linked to more than one high-frequency read (r_8 is linked to r_2, r_3 and r_4; r_9 is linked to both of r_1 and r_3); and (3) uses the AutoML module to judge whether there are erroneous reads between the linked high-frequency nodes (*e.g.*, between r_2 and r_3, between r_5 and r_7).

Although noise2read is a three-stage progressive error correction method, we usually take the first two stages because they are sufficient to eliminate the majority of the errors in many typical NGS datasets. Only in the cases where the data has extensive

coverage, such as amplicon sequencing, the option to use the third step is chosen for additional error correction.

Special considerations in the construction of edit-distance graph of short reads

By setting a high-frequency threshold τ, noise2read finds the 1-nt- or 2-nt-edit-distance edges between unique high-frequency reads (with frequency $> \tau$) and all the other unique reads in a read dataset, and then it takes all these unique reads as nodes, their counts as attributes and the detected associations to build a graph. The rationale for not detecting the 1-nt- or 2-nt-edit-distance read pairs in the low-frequency reads is that it is computationally challenging and meaningless to distinguish whether one read of low abundance is mutated from the other low-frequency read (*e.g.*, it is hard to determine if there are mutations or sequencing errors between two reads each with a frequency of one and with one- or two-base difference). The rationale for 2-nt-edit-distance error correction is that some NGS data contain two base errors in some long read (*e.g.*, 150 bp), and we set a threshold l (*e.g.*, 30 bp) of the sequence's minimum length to determine whether to perform 2-nt-edit-distance error correction.

Noise2read does not perform a pairwise alignment for searching the 1-nt- or 2-nt-edit-distance edges between the high-frequency reads and all the other reads in the read set. Instead, it enumerates all the possible 1-nt- or 2-nt-edit-distance (substitutions only for the 2-nt) reads for all the high-frequency reads and stores them in the Python Set. Then, it invokes the Python built-in function intersection to obtain the edges. It may not be the best way to find all the edges using hash tables in this manner. However, such a strategy can find all required edges instead of finding an approximate number of edges. We constructed the 2-nt-edit-distance graph by searching only substitution relations as edges. This idea is based on the observation that substitutions are the most prevalent type of sequencing error [13], and on that ambiguous nucleotides are often denoted by the symbol "N" [14,15] during sequencing. Moreover, NGS read lengths are usually consistent and fixed in a single sequencing run, owing to the fixed number of sequencing cycles in technologies like Illumina sequencing. This uniform read length is achieved since the read size is directly tied to the number of sequencing cycles performed, and each cycle corresponds to the sequencing of a single base. On the other hand, if a deletion or insertion exists in the read, the sequence length will change, and such a sequence will not appear in a uniform-length sequencing dataset. Noteworthy,

noise2read can handle indel errors when insertion or deletions are represented by the symbol "N".

The time complexity of constructing the 1-nt- or 2-nt-edit-distance read graph in noise2read is $O(h \cdot L^2 + n)$, where h represents the number of high-frequency reads, L denotes the uniform read length, and n is the total number of unique reads in a dataset. This complexity arises from two processes. First, noise2read enumerates all possible 1-nt-edit-distance variants ($O(h \cdot L)$) and 2-nt-edit-distance variants ($O(h \cdot L^2)$) for the high-frequency reads, storing them in a Python set in $O(h \cdot L^2)$ time. It then intersects this set of all n reads to identify edges in $O(h \cdot L^2)$ time. Second, the resulting edges, numbering $E = O(h \cdot L^2, n)$, are used to construct an undirected graph with NetworkX [16] in $O(n + E)$ time. Combining these steps, the overall complexity simplifies to $O(h \cdot L^2 + n)$ for graph construction by noise2read.

Construction of edit-erring-READS and error-free-READs as training data

By defining a maximum frequency threshold τ_{err} ($\tau_{err} \leq \tau$), we considered two kinds of erroneous reads: genuine errors and ambiguous errors. Genuine errors are referred to those leaf nodes whose frequency τ' is less than or equal to τ_{err} ($\tau' \leq \tau_{err}$) and which have a neighbouring node with a higher frequency than τ. This set of erroneous reads is denoted as edit-erring-READS. These genuine errors can be directly rectified to their correct states. While we define two kinds of ambiguous errors: (1) those nodes (reads) r with a low-frequency τ' that are each connected to multiple ≥ 2 high-frequency nodes; (2) wrongly sequenced reads existing between a pair of similar high-frequency reads as the second kind of ambiguous error instances. In other words, in the constructed 1-nt-edit-distance-based read graph, if there are edges between two similar high-frequency sequences, there may be sequencing errors between them. Moreover, amplicon sequencing utilises ultra-deep PCR amplifications for a specific gene target and supports hundreds to thousands of amplicons multiplexed sequencing in one assay to achieve high coverage, but ultra-deep PCR simultaneously amplifies PCR errors. To this end, we further construct a 1-nt-edit-distance-based read graph for amplicon sequencing data and consider those reads of frequencies less than τ_{amp}^{min} (*e.g.*, 50) as potential amplicon errors mutated from its neighbouring reads of extremely high-frequency larger than τ_{amp}^{max} (*e.g.*, 1500).

We consider isolated nodes of high frequencies bigger than τ as error-free reads. We take those isolated nodes of high frequencies in the 1-nt- or 2-nt-edit-distance graphs to build the training set error-free-READS.

Auto machine learning prediction

Unlike the direct rectification of genuine errors into their original state, we model whether a high-frequency read contains true mutations or sequencing errors from its high- or low-frequency neighbours as a classification problem. We created the AutoML module for its end-to-end prediction. The flowchart illustrated in **Figure 4** outlines the steps involved in the AutoML module.

Formulation of the classification problem

We consider edit-erring-READS as positive training instances, while error-free-READS as negatives. For a low-frequency node with a degree greater than two, we calculate its probability of mutation from all its high-frequency neighbouring nodes and take the node with the highest probability as its correct sequence. For the second type of ambiguous error prediction, we integrate the predicted results of the first kind into the training data. In the current version, we only use the predicted ambiguous samples as negative samples for high-ambiguous error prediction to reduce training time and complexity. The mutations observed in high-frequency reads exhibit a bidirectional nature. Therefore, we only consider the prediction result with a higher probability when the bidirectional predictions match. In other words, if the absolute difference between the probabilities of the two-way predictions is less than a specific value, we discard the prediction; otherwise, we choose the prediction having a higher probability.

Feature representation for the training and objective data

A short DNA or RNA sequence can be represented as $r = b_1 b_2 \dots b_i \dots b_l$, where $b_i \in \{A, C, G, T, N\}$ or $b_i \in \{A, C, G, U, N\}$. Here, A, G, C, T and U represent the nitrogenous bases Adenine, Guanine, Cytosine, Thymine and Uracil, respectively. The letter N denotes an uncertain nucleotide, and $l \in \mathbb{N}$ represents the total number of bases in r. We extract features from r by considering its substrings of length k (where $1 \leq k \leq l$), also known as k-mers.

Each training instance consists of two reads in the edit-erring-READS or error-free-READS, examples of the training instances can be found in File S1. The features in reads are extracted using descriptors: (1) Fourier Transformation [17–20], (2) Shannon's and Tsallis's entropy [17,21], and (3) Fickett's score [22,23]. Specifically, the features for a pair of reads with one or two base differences in a training instance may be identical. Therefore, features are only extracted from the absolute correct read (*i.e.*, the first read in training instances) to avoid redundancy. These feature extraction methods are depicted in File S1. Additionally, we used the (4) read counts and (5) characterised the error types and respective motifs as features. For instance, consider two reads ACATG and ACGTG, the error is a substitution of C with G. Here, C-G is the error type, and CAT and CGT are the corresponding motifs. Similarly, for two reads CGTG and ACGTG, the error is an insertion of A, the error type is represented as X-A, and the motifs are XA and AC. We define and normalise the feature vector V of error types or motifs as follows

$$V = (v_1, v_2, \dots, v_i, \dots, v_n) \quad (3)$$

$$v_i = \frac{f_i + \delta}{\sum |i|} \quad (4)$$

Here, i represents an error type, where $count_i$ refers to the total number of occurrences of error type i and $\sum |i|$ refers to the total number of all error types present in the data; δ is a small pre-defined value (*e.g.*, 0.01) assigned to each item to avoid dividing by zero in cases where a certain error type or motif is not present in the data.

Before training, each type of feature is standardised separately by removing the mean and scaling to the unit variance using the pre-processing method "StandardScaler" of Scikit-learn [24]. To address class imbalance issue in the training data, we used the synthetic minority over-sampling Technique (SMOTE) [25] with sampling performed by Imbalanced-learn [26].

Model optimisation and prediction

XGBoost [12] is a well-established and efficient machine learning algorithm for classification. Optuna [27] is a framework that employs sampling and pruning heuristics to automatically discover optimal hyperparameter settings by conducting multiple trials. We chose XGBoost as our classifier and utilised Optuna to optimise the hyperparameters to achieve fast and accurate predictions.

We have pre-set some parameters for the classifier, including the tree method, regularisation term, number of estimators, and learning rate. A logistic regression was used to produce the probability for binary classification, and we aimed to maximize the test accuracy as the objective for selecting the best model via multiple trials (*e.g.*, 20). For each task, noise2read utilised AutoML to create a new Optuna study object for training and selecting the best prediction model. For example, we trained and selected the four best models for predicting 1-nt- and 2-nt-based ambiguous, 1-nt-based high ambiguous, and amplicon errors.

The time complexity of training XGBoost optimised by Optuna in noise2read, using "hist" or "gpu_hist" tree method, is $O(n_{tasks} \cdot n_{trials} \cdot n_{trees} \cdot n_{samples} \cdot n_{features})$. Here, n_{tasks} (*e.g.*, 1–4) is the number of tasks, n_{trials} is the number of Optuna trials (*e.g.*, 10–20) per task, n_{trees} is the pre-set number of trees, $n_{samples}$ is the number of constructed training samples, and $n_{features}$ is the number of constructed features.

Error correction for isolated nodes

After prediction, we restore all the edit-erring-READS to their normal state. We then adopted a third-party method Bcool [28] to deal with the errors contained in the isolated nodes, including many singletons, in the 1-nt-edit-distance read graph. However, we keep only those corrected sequences by Bcool, which present in the original or in the first round of the corrected dataset without any genuine and ambiguous errors, to prevent generating non-existing new sequences.

Evaluation criteria

Generation of the gold-standard wet-lab datasets with UMI-based ground truth

In this study, we developed a novel approach for generating ground truth datasets, motivated by Mitchel's method presented in literature [29]. One of the differences is we use the error-corrected Unique Molecular Identifier (UMI) to construct ground truth datasets. More details of our approach can be found in supplementary methods in File S2. To ensure validity, credibility and fairness in performance comparison, the UMI-based ground-truth datasets and evaluation procedures established in the benchmarking study [29] were also adopted for performance evaluation.

Generation of UMI-based simulated datasets

Taking motivations from the simulation approach for generating simulated miRNA sequencing data in [30], we introduced an innovative method to generate UMI-based simulation datasets that can be applied to a broader range of NGS datasets, extending beyond just miRNA sequencing data. The details of simulation process can be found in supplementary methods in File S1.

We also employed the simulation process in [30] to generate simulated single-end miRNA sequencing datasets to evaluate the proposed method's performance. Differently, we additionally incorporate mimic UMIs (numbers are used here instead of base sequences) for unique sequences in the generated error-free read set based on the assumption that each unique read corresponds to a UMI to adapt the evaluation framework developed in this study.

Evaluation metrics

To accurately evaluate the performance of error correction methods, we propose using confusion matrices at the read-level and base-level to measure changes in reads within the same UMI cluster rather than relying on the sequencing IDs generated by the instruments. The rationale is that for the constructed UMI-based datasets in this study, there is only one unique error-free sequence (of multiple occurrences) in each UMI cluster. Therefore, in a UMI cluster, we only need to compare the edit distance between every other unique read and this error-free read before and after error correction. Then, we can compute the confusion matrix using the relevant read count information before and after correction. More than half of the calculation time was saved this way. Otherwise, if we use the sequencing ID as the index, we must compare the edit distances twice for each group (same sequencing ID) of the raw, error-free, and corrected reads, even if they are the same. The absolute values of the true positives in each dataset are associated with the number of reads rather than the number of UMIs. The total number of positives of the actual condition equals the sum of the True positive and False negative. At the read level, a read is deemed erroneous if even a single base is incorrect. Conversely, a read is considered error-free only when all its bases are correct. TP defines the number of edit-erring reads perfectly modified after correction, and TN is the amounts of error-free reads without any changes after modification. While FP denotes the counts of error-free reads that are incorrectly adjusted by introducing new errors, FN represents the number of unchanged or wrongly fixed edit-erring reads.

Similarly, at the base level, TP, TN, FP and FN concerned about the mistaken or accurate bases changing before and after correction.

Additionally, we employ the edit distance changes instead of the multiple sequence alignment (MLA) strategies used in [29] among raw, authentic and modified reads to get the confusion conditions as the MLA is highly time-consuming. Another reason is that MLA has more alternative alignment results since it compares three reads. In contrast, the edit-distance-based strategy only compares the ground truth read to its raw or corrected one, respectively. When counting the FN on the base level, we measure the absolute edit distance difference with the accurate read before and after correction.

Then, we derive the True Positive Ratio (TPR, a.k.a. recall or sensitivity), False Negative Ratio (FNR), True Negative Ratio (TNR), False Positive Ratio (FPR, a.k.a. fall-out), precision, gain and accuracy from the confusion matrix. TPR and FNR are the ratio of the number of edit-erring reads or bases correctly rectified and wrongly kept as negatives to the total number of actual edit-erring reads or bases, respectively. TNR and FPR are defined as the ratio of the number of error-free reads or bases correctly kept as negatives and wrongly rectified to the total number of actual error-free reads or bases, respectively. From the information theory perspective, TPR is the ratio of noise turning to signal; in contrast, FNR is the unconverted ratio of noise to signal. While FPR is the percentage of new noise introduced, TNR is the ratio of the original signal preserved.

$$TPR = \frac{TP}{TP + TN} \tag{5}$$

$$FNR = \frac{FN}{TP + FN} \tag{6}$$

$$TNR = \frac{TN}{FP + TN} \tag{7}$$

$$FPR = \frac{FP}{FP + TN} \tag{8}$$

An ideal performance should achieve high TPR and TNR while keeping FNR and FPR low. Therefore, we can construct a cross-coordinate system where derived scores from the confusion matrix are assigned to each of the four directions. The four index values of TPR in the upper axis, FPR in the lower axis, TNR in the right axis and FNR in the left axis form a rectangle. The larger the overlapping area between the rectangle and the upper right quadrant, the better the performance. Therefore, we define a quantitative metric of the overlapping Area Difference (AD) to assess the performance as follows,

$$AD = TPR \cdot TNR - TPR \cdot FNR - FNR \cdot FPR - FPR \cdot TNR \quad (9)$$

Moreover, we also calculate the Precision, Positive Gain and Accuracy denoted as follows to evaluate the correction performance at read-level or base-level. Precision evaluates the ratio of precise modifications among all the completed corrections and all the errors, while Positive Gain indicates the positive effect among all the real errors. The accuracy is the proportion of accurate modifications, including true positives and negatives, to the total number of reads or bases concerned.

$$Precision = \frac{TP}{TP + FP} \quad (10)$$

$$Positive \;\; Gain = \frac{TP - FP}{TP + FN} \quad (11)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (12)$$

Furthermore, based on the read-level definition for the dataset of known ground truth, we classify reads into two categories: edit-erring and error-free. Then we measure the purity of the dataset using entropy defined as

$$E = -p \cdot \log_2 p - (1 - p) \cdot \log_2 (1 - p) \quad (13)$$

where p is the probability of randomly selecting one error-prone or error-free read from all sequences. The lower the dataset entropy, the fewer edit-erring reads exist in the dataset.

Results

To assess the performance of noise2read, we generated UMI-based ground-truth and simulated datasets based on the methods developed in this study and literature [29,30]. After evaluating the performance of noise2read, we conducted case studies on abundance change of viral reference genomes, isoform identification, single nucleotide polymorphism (SNP) profiling, and genome base editing efficiency estimation to assess the impact of noise2read's error correction on downstream applications. The flowchart illustrated in **Figure 5** outlines the analytical framework and key concepts in this study. As a result, we used 39 datasets generated in this study or from third-party studies to evaluate noise2read. Datasets $D1 - D8$ are UMI-based ground-truth datasets derived from eight real sequencing runs SRR1543964–SRR1543971 using a UMI-contained high-fidelity sequencing technique (a.k.a. safe-SeqS) [29,31]. $D9 - D13$ are simulated datasets with mimic UMIs based on actual sequencing data (with read lengths of 75 bp,

101 bp and 150 bp). $D14 - D17$ are four single-end miRNA datasets generated using the simulation procedure proposed in miREC [30] and an additional step of mimicking UMIs to these datasets. The eight UMI-based ground-truth datasets [29] were labelled as $D18 - D25$ in this study. $D26 - D37$ are one paired-end and ten single-end sequencing datasets used for case studies. Detailed information about these datasets $D1 - D39$ can be found in Table S1.

Comparing with state-of-the-art methods including k-mer-methods [32–38], multiple sequence alignment based methods [39–43], and other methods [28,30,44], our noise2read consistently outperforms under 19 metrics on eight UMI-based wet-lab datasets and five simulated single-end and paired-end datasets constructed in this study. It also has superior performance on eight UMI-based wet-lab datasets and four simulated miRNA datasets established previously in published literature. Moreover, case studies on abundance change of viral reference genomes, isoform identification and SNP profiling, and genome base editing efficiency estimation revealed that noise2read can make substantial impacts on downstream applications. The versions of noise2read and other methods, along with the corresponding commands and parameters used in the experiments, are provided in File S2.

High prevalence of erroneous reads containing one or two base errors from UMI-based cluster and distribution analysis

We utilised the sequence information of UMI tags to investigate the distributions of erroneous reads that contain different numbers of base errors. Specifically, we divided the reads in a UMI group into high-frequency reads and low-frequency reads. Then, we calculated the edit distance between each unique low-frequency read and each unique high-frequency read. Then, each of the unique low-frequency read has the smallest edit distance with the set of unique high-frequency reads. Given each of these smallest distances, we record the number of low-frequency sequences that have this edit distance with the set of high-frequency reads.

We applied the above process to the datasets of SRR1543964–SRR1543971 by defining a high-frequency read as a read with a copy count no less than five (note: clusters with ambiguous base "N" in the UMI sequence are not included in this analysis). We observed that there exist two different high-frequency sequences that have been tagged with the same UMI, as similarly reported in the literature [45]. For instance, as

seen in Figure S2, each of these UMI clusters has two high-frequency reads, between which the edit distance is larger than 100 (111 or 129 respectively), demonstrating that such two high-frequency reads within the same UMI cluster should be originated from two different molecules, although they were tagged with the same UMI.

Moreover, there exist big editing distances (*e.g.*, 116) between high and low-frequency reads within the same UMI cluster, it would be unreasonable to assume only base-editing-error-relationship between all the low and high frequency reads. In fact, a low-frequency sequence with a small edit distance to a high-frequency read is more likely caused by PCR or sequencing errors. Here, we assume that those low frequency reads with an edit distance ≤ 4 may be erroneous reads caused by PCR or sequencing errors. In this context, among these eight data sets, at least 60% of the erroneous reads in 95.21%–96.70% of the UMI clusters are caused by 1 or 2 base errors, as depicted in the stacked bar chart in Figure S3. Five more UMI clusters are shown in Figure S4: 81.25%, 95.24%, 100%, 94.73% and 100% of low-frequency reads have the 1 and 2 base differences with the set of high-frequency reads in the same UMI cluster. These findings indicate that those erroneous reads containing one mistaken base or two constitute a more significant proportion of the total erroneous reads in the dataset. Based on our theoretical analysis and UMI cluster analysis, the proposed algorithm, noise2read, is set to correct erroneous reads containing base errors < 3.

Entropy reduction and information gain after error correction

The error correction effect or the noise/uncertainty reduction by an error correction method in a dataset can be measured by Shannon's entropy and information gain. For a read dataset R, its Shannon entropy H is given by

$$H = -\sum_{r \in R} p_r \cdot \log_2 p_r \tag{14}$$

where p_r is the percentage frequency of r in R.

An ideal correction should eliminate all the errors/noises in the dataset while not introducing any new errors, or new sequences. Therefore, the entropy $H'(R')$ of a corrected read dataset R' should consist of two parts: one is about the original reads, the other is about the wrongly introduced reads. We define $H'(R')$ as

$$H'(R') = H'(R' \cap R) + H'(R' - R) \tag{15}$$

$$H'(R' \cap R) = -\sum_{r \in \{R' \cap R\}} p_r \cdot \log_2 p_r \tag{16}$$

$$H'(R'-R) = -\sum_{r' \in \{R'-R\}} p_{r'} \cdot \log_2 p_{r'} \quad (17)$$

Information gain reflects the amount of information gained from the original state of the reads after the error correction, defined as

$$\triangle I(R';R) = H(R) - H'(R') = H(R) - H'(R' \cap R) - H'(R'-R) \quad (18)$$

The fewer unique reads are in a dataset, the less uncertainty, and the entropy will decrease after mistaken bases are corrected. To see the error correction effect on the data quality improvement, we propose to visualise the information change via a heatmap of taking items of $\triangle I$ as minor rectangular points and marking the wrongly introduced sequences as noises red dots. A visualization of $\triangle I$ for $D1$ is shown in **Figure 6** and those for $D2 - D8$ are presented at Figures S5–S11. The primary colour of the heatmaps close to zero strongly suggests that the correction conserves the original high-frequency information by all the methods. The negative and positive scores on the colour bar describe the information gain and loss, respectively. As shown in Figure 6, noise2read is better than the other methods to reduce noise level as there is nearly no score > 0. Those red points depict information loss brought by wrongly introduced new sequences, leading to new errors to increase false positives and negatives. Noise2read does not yield any non-existing reads, and the colour in Figure 6B darker than that in Figure 6A implies that noise2read has more information gained from the additional amplicon sequencing correction.

Moreover, to intuitively quantify the information gain or loss, we considered the changes only in low-frequency sequences before and after error correction. We denote the frequent reads as a subset FR_τ of R, and we calculate the entropy by removing FR_τ from R or R'. Then, we focus on the entropy change given by

$$\triangle H = H(R - FR_\tau) - H'(R' - FR_\tau) \quad (19)$$

where τ is a threshold for defining high frequency reads, $H(R - FR_\tau)$ and $H'(R' - FR_\tau)$ represent the non-frequent reads' entropy before and after correction, respectively.

We calculated the entropy change $\triangle H$ for the sequencing datasets $D1 - D8$ and five simulated datasets $D9 - D13$ after error correction (details shown in **Table 1**). noise2read achieves the most considerable information gain on all these datasets. Specifically, the increased information by noise2read on the simulation datasets

outperforms the other methods. The extensive information gain is because noise2read can rectify almost all the errors in the simulated datasets.

Performance comparison on UMI-contained sequencing datasets established in this study

We evaluated the performance of noise2read in comparison with seven other computational error correction methods at both the base-level and read-level under various metrics, including TPR, TNR, FPR, FNR, AD, Precision, Positive Gain, Accuracy, and Purity Entropy on UMI-based ground truth data $D1 - D8$.

The comparative performance with the seven methods Coral [39], RACER [33], Fiona [40], Lighter [34], Pollux [36], Bcool [28], and Care [42,43] are summarized in **Figure 7**A for $D1$ and those in Figures S12 and S13 for $D2 - D8$. Tables S2–S9 are provided to further supplement the results. Our method noise2read has achieved the best performance on all the datasets under all the metrics.

The high-TPR and low-FNR performance indicate that noise2read can turn most noise while leaving the lowest number of actual noises as signals; The high TNR illustrates that noise2read can introduce fewer new errors by preserving most signals unchanged, while the low FPR suggests that noise2read introduces none or few new noises without bringing up any non-existing sequences after the correction process. In detail, noise2read surpassed all the other methods on $D1$, achieving a score 0.924 higher than the second-best method RACER which has a score of 0.859. Notably, noise2read exhibited exceptional performance in Recall, Precision, Positive Gain, Accuracy, and Purity Entropy, as evidenced by the values in **Table 2**. The positive gain percentage of noise2read is 7.26% and 48.15% higher than RACER and Care. noise2read and its amplicon mode achieved the finest purity entropy of 0.05 and 0.077, sounder than the second-best method RACER which has a score of 0.110.

The progressive process gradually converts noise into signals; for example, in Table 2, the 1st, 2nd and 3rd stages convert 72.9% (81,630), 93.2% (104,373), and 96.2% (107,717) of the errors into signals on $D1$, respectively. Noise2read is mainly designed for any short-read sequencing data whenever PCR is involved. Without the 3rd step, it also achieves sound performance (refer to Tables S2–S9) by restoring most erroneous reads into their normal states and not introducing false positive reads. The 3rd step for further correction on amplicon sequencing data maintains fewer original error-free

reads than the second stage and correspondingly introduces more noise but not new sequences. RACER can rectify 93.4% of the noise but newly introduces almost 22 times the number of new errors compared to our method. Care newly introduces 52 false positives but can only correct 64.9% of the erroneous reads. The other methods can only correct less than 65% of the errors but simultaneously give rise to thousands of new mistakes.

Performance on simulated short-read datasets and those with artificially modified bases

Error correction performances on the simulated ground truth are shown in Figure 7B–L, Figure S14 and Table S10 for dataset $D9$ and Figures S15–S22 and Tables S11–S14 for datasets $D10 - D13$. Noise2read super outperforms the other methods under all the metrics on all the simulated datasets. Specifically, for the AD performance, noise2read (0.989) has 39.3%, 61.9% and 123.3% higher performance than that of Lighter (0.71), Care (0.611) and Fiona (0.443) (Figure 7B). Noise2read reaches the best precision, gain and accuracy and achieves a substantial positive gain (Figure 7C). As shown in Figure 7D, noise2read is the only method significantly decreasing the purity Entropy after correction. Information gain visualisations in Figure 7E–L indicate the information is still dominated by most of the original signal after correction. All the other methods wrongly introduced new sequences (in a number of 164 to 9698) after correction. The other methods' performance fluctuates widely. For instance, at the read level, the performance ranking of the top three methods in terms of AD on dataset $D9$ (Figure 7B) is Lighter, Care and Fiona. However, the performance ranking is Fiona, Care, and Coral on $D10$ (Figure S15), and Lighter, Fiona and Care (Figure S17) on $D12$.

Performance evaluation using previously established simulated miRNA sequencing datasets

Comparison results between noise2read (with or without high-frequency ambiguous error prediction) and miREC are shown in **Table 3** and Table S15. Noise2read can rectify more errors than miREC, achieving more TP and less FN after correction. The miREC method and noise2read can achieve similar, reasonably good results in accuracy, precision and fall-out (Table S15). However, from the recall, Positive Gain, purity

Entropy (E) and information gain ΔH performance on all four datasets, noise2read is better than miREC.

Performance evaluation using independently established UMI-based benchmarks

Table 4 shows the comparative performance of noise2read on $D25$ in comparison with Bless [37], Coral [39], Lighter [34], Reckoner [38], Sga [44], BFC [35], Pollux [36], Fiona [40], RACER [33], and Care [42,43]. The comparison results on $D18 - D25$ (see Tables S16 and S17) highlight that noise2read consistently achieved the highest number of true positives on all these datasets, except for Fiona's TP on $D25$, which is slightly bigger than noise2read. Importantly, noise2read demonstrates the lowest count of false positives among all these datasets. noise2read performs exceptionally good in Precision, Accuracy, AD and Positive Gain on all the eight benchmark datasets.

Runtime and memory consumption

We compared CPU runtime and peak memory used by noise2read with those by Care [42,43], RACER [33], Bcool [28], Pollux [36], Lighter [34], Fiona [40], and Coral [39] on data sets $D1 - D8$. We executed all the programs on an Intel(R) Xeon(R) Gold 6238R CPU clocked at 2.20GHz, leveraging 56 CPU cores for parallel computing. For the model training of noise2read, a single Tesla V100S-PCIE-32GB GPU was employed. To gauge memory usage across all the programs, we used the library psutil (https://github.com/giampaolo/psutil). These runtime and memory consumption comparisons are presented **Table 5**.

Lighter and Care exhibited fast speeds, completing corrections within a minute by taking a small amount of memory consumption for each of the 8 data sets. On the other hand, Pollux had the slowest speed due to its inability to run in parallel. Noise2read spent the second-highest memory consumption and made the second-slowest speed. (We do not suggest using many multiprocessing processes for noise2read to run, as we have observed that those situations could suddenly consume a significant amount of memory, and the program ran out of memory and terminated.)

The separate time consumption by noise2read at its different stages as recorded in the built-in log files across all datasets $D1 - D39$ are presented in Table S18. It can be observed that a significant amount of time was spent on tasks such as constructing "2-nt-edit-distance read graphs", "performing feature extraction", and "model training".

To shorten the running time on large data sets, it is suggested to choose a smaller number of negative samples and set a smaller number of trials (*e.g.*, 20) for the construction of suboptimal models. Additionally, opting not to predict errors within high-frequency reads will also save noise2read a substantial amount of time and memory usage but fortunately without much performance sacrifice on error correction. We note that although noise2read is slow, it never introduces any non-existing reads into the datasets. This is a unique merit in all current sequencing error correction methods.

While the speed of error correction is undeniably a crucial factor in evaluating the performance of a correction method, an even more critical consideration is whether the method introduces new errors. A fast error-introducing method damages the quality of the whole dataset and may become unexpectedly harmful to downstream data analysis, although its fast speed is advantageous in the preprocessing error correction stage. Our method does not have this speed advantage so far, but it never introduces new errors, guaranteeing the integrity of the datasets. In future work, we consider efficiency tricks to improve the speed of feature extraction and machine learning.

Error correction increases Monkeypox virus genome abundance by 52.12%

The Monkeypox virus has severely affected the health of human beings and its reference genome sequence has been extensively utilised to understand the origin and phylogeny, and as a fundamental framework for the design of mRNA vaccines. We investigate how much abundance is changed for the reference genome after our algorithm noise2read rectifies the base errors contained in the short-read sequencing data. The study will help understand the within-host viral mutants of the reference genome and the abundance compositions.

We used the paired-end whole-genome sequencing dataset SRR22085311 (its paired R1 and R2 denoted as $D26$ and $D27$ here) and the reference genome GCA_025947495.1 [46]. Our noise2read rectified a huge number of erroneous reads in $D26$ (400,622 out of 3,599,812 reads, *i.e.*, 11.13%), and another huge number of erroneous reads in $D27$ (456,242 out of 3,599,812 reads, *i.e.*, 12.67%). **Figure 8**A presents a coverage comparison chart before and after the error correction, alongside a coverage difference chart (Figure 8B), where the average base coverage of the reference genome is increased by 52.12% from depth 1216.75 to 1850.95 after a huge number of

651,410 reads were retrieved to perfectly align with the genome. The frequency distribution of the base coverage differences as another angle viewing the abundance change for the Monkeypox virus is presented in (Figure 8C), where the abundant and perfectly matched reads aligned to the genome are highlighted again. Especially, those positive shifts towards a higher coverage (Figure 8B and C) confirm much more about the ground truth of the known reference genome and the detection of possible new variants of the genome.

The substantial changes in genome abundance for Monkeypox after error correction prompt a revaluation of genome sequences and how we detect new variants. Although this alignment-based analysis focused on viral data, our method has broader applications, as it effectively corrects errors in PCR amplified short-read sequencing without introducing non-existent reads, while preserving data integrity. Since reference genome alignment is a widely used strategy in bioinformatics, our findings suggest that noise2read can enhance the accuracy and conclusions of alignment-based studies across a wide range of organisms and datasets.

Accurate error correction improves detection of isomiRs and refines SNPs profiling

MicroRNAs (miRNAs), non-coding RNA molecules approximately 22 nt, can modulate gene expression post-transcriptionally through the silencing and decay of target mRNAs [47]. Dysregulation of miRNAs plays crucial roles in many biological mechanisms, and it is also a main reason in cancer and autoimmune disorders [48,49]. By miRNA sequencing, various types of isoforms (*i.e.*, isomiRs) have been detected [50]. However, whether the base differences found in the isomiRs are actual biological variations or synthetic artefacts due to the PCR or sequencing errors or both is difficult to judge. Here, we study how our error correction changes the identification and quantification of isomiRs from short RNA-seq datasets and how it refines the profiling of known SNPs in isomiRs.

We downloaded ten single-end small RNA-sequencing datasets of lymphoblastoid cell lines from five population groups in the 1000 Genomes Project [51]. These datasets (denoted as $D28 - D37$ here) were cleaned by removing the adapter sequences via cutadapt [52]. We used IsoMiRmap [53] under the setting of pre-defined miRNA reference sets from the database miRbase (v22) [54] as a "miR-space" to quantify known isomiRs and SNPs for $D28 - D37$ before and after our sequencing error

correction. IsoMiRmap tags an identified isomiR as an exclusive isomiR if it only exists in the miR-space with one or more occurrences but not elsewhere in the human reference genome, otherwise recognized as an ambiguous isomiR.

These quantification results are summarised in **Table 6**. The number of unique ambiguous isomiRs is decreased by 24.12%–31.75% or in numbers from 151 to 245, but their total counts are increased by a number between 160 and 640 among the ten datasets after the error correction; the number of exclusive isomiRs is decreased by 34.46%–37.48% but their total counts are increased by a number between 5095 and 14,441. These results suggest that some previously identified isomiRs are artifacts containing sequencing errors rather than natural isoforms. On the other hand, for the profiling of the known SNPs, the number of unique SNPs decreased by 34.13%–59.09%, and their counts also decreased by 4.40%–35.56% except for two increased by 1.41% and 1.61% respectively. This observation unveils that some of the previously annotated SNPs are sequencing errors. Similar quantitative and qualitative changes observed in the profiling of these known SNPs in the isomiRs distinguishing true SNPs from sequencing errors enable more accurate annotation of SNPs. The significant change of the isomiRs quantification after correction is because an average of 235,146 (2.62%) sequences were corrected by noise2read in the ten datasets (Table 6).

To understand more about the frequency change of isomiRs and SNPs, we categorised the isomiRs according to their original miRNAs, then we utilised scatter graphs with Kepler plots to understand the associations between the number of identical isomiRs and total isomiRs' count ($\log_{10}$ transformation) before and after the error correction of the sequencing reads. The leftward shift on the x-axis (**Figure 9**A and B for exclusive isomiRs of $D28$ and $D29$, respectively, Figure 9C and D for ambiguous isomiRs and known SNPs of $D28$, and Figures S23–S25 for the other miRNA datasets) indicates a reduction of the count of unique isomiRs, while the upward change on the y-axis indicates an increase in authentic isomiRs. These significant changes in isomiRs and SNPs highlight the importance of correction for accurately characterizing isomiR and SNP profiles, making contributions to the annotation of isomiRnome.

Accurate error correction significantly improves ABE/CBE editing outcomes

Base editing is a new genome editing technique that uses CRISPR systems and enzymes to introduce point mutations into cellular DNA or RNA for modelling and

understanding genetic diseases [55,56]. However, deciding whether a nucleotide position is exactly editable in a genomic context is inefficient by wet-lab experiments, and the base editors may yield many unexpected genotypic output sequences when the editable window covers multiple target nucleotides. Deep-learning-based prediction tools have been developed to predict the base-editing efficiency and outcome-sequence copy numbers from Adenine and cytosine base editors (ABEs and CBEs) [57]. The training data used by these prediction tools are extracted from short-read DNA/RNA sequencing data. Here, we investigate how much the number of unique reads (unique outcome sequences) changes after our sequencing error correction.

We removed those records in which the target sequence has only one outcome sequence from the training data of HT_ABE_Train and HT_CBE_Train used in the literature [57]. Then, we cleaned them to form two datasets (denoted by $D38$ for ABEs and D39 for CBEs here), and applied noise2read to $D38$ and $D39$ separately. As a result, the number of unique outcome sequences in $D38$ is reduced by 2309 from 28,892 to 26,583 (7.99%), and the number of unique outcome sequences in $D39$ is reduced by 5042 from 27,312 to 22,270 (18.46%). The number reduction of unique outcome sequences is because some low-frequency reads are not a result of base editing but due to sequencing errors. In total, noise2read recognised 5109 erroneous reads in the ABE dataset and 10,271 erroneous reads in the CBE dataset and turned all of them into normal states. This error correction has significantly improved the quality of the training data that would be very helpful for enhancing the prediction of base editing efficiencies.

Discussion

A long-standing problem in sequencing data analysis is how to reduce sequencing base errors and erroneous reads as much as possible before any downstream applications. Existing short reads correction methods utilize biochemical-based experimental designs such as unique molecular identifiers (UMIs) to count and track molecules [10], or take computational methods including k-mer-methods [32–38], multiple sequence alignment based methods [39–43], and other methods [28,30,44]. One limit of the UMI-based strategies is that errors/mutations can also happen at UMIs. Serious concern about the computational methods is that they have significantly overcorrected reads by introducing pseudo new sequences or shifting one type of error into another, often

leaving numerous reads uncorrected. Some of these methods only focus on restoring substitution mistakes but do not support indels' correction. Besides, instance-based methods such as miREC [30] were designed to handle specific sequencing data type miRNA sequencing reads. And it assumes that frequent sequences contain no mistakes, thus it cannot be used to correct potential errors between high-frequency reads or cannot deal with those singletons with no relationships to the high-frequency reads.

Following the principle of the PCR erring incidents and sequencing process, we constructed special graphs of short reads to capture the relationships between edit-erring and error-free reads. Through novel modelling of the errors between high-frequency reads and their high- or low-frequency neighbours as a classification problem, we have successfully predicted almost all the errors using machine learning techniques. Validation experiments on the UMI-based wet lab and simulated datasets of known ground truth have demonstrated that the proposed noise2read algorithm can eliminate most of the PCR and sequencing errors without introducing any non-existing sequences into the read set.

Moreover, we investigated the impact of error-corrected data on downstream data applications. We have found that: (1) The abundance level change of the reference genome of Monkeypox virus after the sequencing error correction is remarkable, which may allow us to rethink how to get a precise genome sequence for the virus; (2) For the isomiRs and SNPs profiling, the counts of some isomiRs and SNPs are decreased while some others are increased, which is of great significance to identifying actual isomiRs and SNPs and re-annotating the isomiRnome. (3) Both ABE and CBE should have higher base editing efficiency than currently estimated. The accurate and higher base editing efficiency with correct preprocessing may improve the original deep-learning prediction accuracy. Altogether, these observations and advantages lay down strong evidence to question the accuracies of current downstream research outcomes and open new avenues to conduct downstream analysis whenever short-read data are adopted. In addition to the significant impact demonstrated across the three case studies, our algorithm holds broader potential for applications in cutting-edge research areas that rely on short-read sequencing data. These include advanced research fields such as genomics, epigenomics, infectious disease diagnostics [58,59], low-frequency mutation or rare mutation detection [60], and virus detection [61]. Additionally, a recent study [62] has already highlighted the potential advantages of using error-corrected NGS in assessing off-target effects of gene therapies, enhancing carcinogenicity assessment and

advancing genetic toxicology and underscored the potential application of error-corrected NGS for human cancer risk assessment and genetic toxicology testing. We recommend that researchers employ our method to conduct sensitivity analyses based on raw and error corrected short read sequencing data in their cutting-edge studies.

A small edit distance such as 1 or 2 is currently used to define the edges of $rG(R)$. When the edit distance threshold Emax is enlarged, more edges will be created for $rG(R)$ and possibly more erroneous reads will be identified. The trade-off is that the computational complexity of constructing these new edges is exponential while newly identified erroneous reads become less and less when Emax increase. In fact, these erroneous reads constitute an extremely small percentage (< 0.16%) of the total erroneous reads in theory. In future work, we will test the computational complexity when $Emax$ is set as 3 and explore how to change the correction steps. Additionally, the optimal value of the parameter τ may vary across different sequencing platforms, applications, and experimental conditions. Conducting wet-lab experiments using synthetic sequencing is a more effective strategy for assessing the adaptability of τ in various settings. In our future work, we will design and conduct experiments to further investigate the optimal τ under different experimental conditions.

The speed and memory usage of noise2read still needs improvement, especially the parts for building the 1-nt- and 2-nt-edit-distance read graphs and AutoML training for prediction. The easy-usable and automatic tuning of the classifiers' parameters facilitates wide-range explorations, but we note that noise2read may yield a slightly different result at different trials, even setting the same seeds. We also note that noise2read will derive more false positives when dealing with errors between high frequency reads of extremely short length (*e.g.*, < 30 bp). This limit may be overcome by extracting more or fewer features from the reads. Furthermore, we already attempted using deep learning architecture (*e.g.*, CNN and LSTM) to detect the errors, but a better performance was not achieved than by current noise2read. To elevate noise2read from a good tool to an exceptional one, we plan to explore novel feature representations for short reads and incorporate attention-based deep learning models in future work. Additionally, noise2read operates independently of sequencing quality scores, allowing it to address errors across various sequencing platforms and conditions. However, we acknowledge that incorporating quality scores may further improve the accuracy of our

correction procedure. As part of our future work, we also plan to explore integrating quality scores as an additional feature to enhance the correction process.

Code availability

The algorithm, noise2read, developed in this study is packaged and released on the Python Package Index (PyPI) at https://pypi.org/project/noise2read/ and Bioconda at https://anaconda.org/bioconda/noise2read with source code publicly available at https://github.com/JappyPing/noise2read and documentation publicly available at https://noise2read.readthedocs.io/en/latest/. The code has also been submitted to BioCode at the National Genomics Data Center (NGDC), China National Center for Bioinformation (CNCB) (BioCode: BT007951), which is publicly accessible at https://ngdc.cncb.ac.cn/biocode/tools/BT007951.

Data availability

No new raw sequencing data were generated in this study.

CRediT author statement

Pengyao Ping: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – original draft preparation, Writing – review & editing. **Shuquan Su:** Formal analysis, Methodology, Visualization. **Xinhui Cai:** Data curation, Formal analysis, Visualization. **Tian Lan:** Formal analysis, Methodology. **Xuan Zhang:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology. **Hui Peng:** Data curation, Formal analysis. **Yi Pan:** Resources, Writing – review & editing. **Wei Liu:** Supervision, Resources, Writing – review & editing. **Jinyan Li:** Conceptualization, Formal analysis, Funding acquisition, Methodology, Project administration, Resources, Supervision, Writing – original draft preparation, Writing – review & editing. All authors have read and approved the final manuscript.

Competing interests

The authors declare that they have no competing financial interests.

Acknowledgments

We would like to thank the computational resources provided by the University of Technology Sydney eResearch High-Performance Computer Facilities.

Supplementary material

Supplementary material is available at *Genomics, Proteomics & Bioinformatics* online (https://doi.org/10.1093/gpbjnl/XXXX).

ORCID

0000-0002-1829-3273 (Pengyao Ping)
0000-0003-3957-6325 (Shuquan Su)
0009-0002-7590-172x (Xinhui Cai)
0009-0008-8714-2744 (Tian Lan)
0000-0002-3089-9809 (Xuan Zhang)
0000-0002-5137-1827 (Hui Peng)
0000-0002-2766-3096 (Yi Pan)
0000-0002-3003-1313 (Wei Liu)
0000-0003-1833-7413 (Jinyan Li)

References

[1] Shendure J, Balasubramanian S, Church GM, Gilbert W, Rogers J, Schloss JA, et al. DNA sequencing at 40: past, present and future. Nature 2017;550:345–53.
[2] Kukurba KR, Montgomery SB. RNA sequencing and analysis. Cold Spring Harb Protoc 2015;2015:951–69.
[3] McCombie WR, McPherson JD, Mardis ER. Next-generation sequencing technologies. Cold Spring Harb Perspect Med 2019;9:a036798.
[4] Ross MG, Russ C, Costello M, Hollinger A, Lennon NJ, Hegarty R, et al. Characterizing and measuring bias in sequence data. Genome Biol 2013;14:R51.
[5] Ma X, Shao Y, Tian L, Flasch DA, Mulder HL, Edmonson MN, et al. Analysis of error profiles in deep next-generation sequencing data. Genome Biol 2019;20:50.
[6] Freedman AH, Clamp M, Sackton TB. Error, noise and bias in *de novo* transcriptome assemblies. Mol Ecol Resour 2021;21:18–29.
[7] Ma KY, Schonnesen AA, He C, Xia AY, Sun E, Chen E, et al. High-throughput and high-dimensional single-cell analysis of antigen-specific CD8+ T cells. Nat Immunol 2021;22:1590–8.
[8] Kebschull JM, Zador AM. Sources of PCR-induced distortions in high-throughput sequencing data sets. Nucleic Acids Res 2015;43:e143.

[9] Bentley DR, Balasubramanian S, Swerdlow HP, Smith GP, Milton J, Brown CG, et al. Accurate whole human genome sequencing using reversible terminator chemistry. Nature 2008;456:53–9.
[10] Marx V. How to deduplicate PCR. Nat Methods 2017;14:473–6.
[11] Kivioja T, Vähärautio A, Karlsson K, Bonke M, Enge M, Linnarsson S, et al. Counting absolute numbers of molecules using unique molecular identifiers. Nat Methods 2012;9:72–4.
[12] Chen T, Guestrin C. XGBoost: a scalable tree boosting system. Proc 22nd ACM SIGKDD Int Conf Knowl Discov Data Min. New York: Association for Computing Machinery; 2016, p. 785–94.
[13] Schirmer M, Ijaz UZ, D'Amore R, Hall N, Sloan WT, Quince C. Insight into biases and sequencing errors for amplicon sequencing with the Illumina MiSeq platform. Nucleic Acids Res 2015;43:e37.
[14] Metzker ML. Sequencing technologies — the next generation. Nat Rev Genet 2010;11:31–46.
[15] F. Löchel H, Heider D. Comparative analyses of error handling strategies for next-generation sequencing in precision medicine. Sci Rep 2020;10:5750.
[16] Hagberg AA, Schult DA, Swart PJ. Exploring network structure, dynamics, and function using NetworkX. In: Varoquaux G, Vaught T, Millman J, editors. Proc 7th Python Sci Conf. Pasadena; 2008, p. 11–5.
[17] Bonidia RP, Domingues DS, Sanches DS, de Carvalho ACPLF. MathFeature: feature extraction package for DNA, RNA and protein sequences based on mathematical descriptors. Brief Bioinform 2022;23:bbab434.
[18] Holden T, Subramaniam R, Sullivan R, Cheung E, Schneider C, Tremberger Jr. G, et al. ATCG nucleotide fluctuation of Deinococcus radiodurans radiation genes. Instruments, Methods, and Missions for Astrobiology X. SPIE; 2007, p. 402–11.
[19] Anastassiou D. Genomic signal processing. IEEE Signal Process Mag 2001;18:8–20.
[20] Marsella L, Sirocco F, Trovato A, Seno F, Tosatto SCE. REPETITA: detection and discrimination of the periodicity of protein solenoid repeats by discrete Fourier transform. Bioinformatics 2009;25:i289–95.
[21] Bonidia RP, Sampaio LDH, Domingues DS, Paschoal AR, Lopes FM, de Carvalho ACPLF, et al. Feature extraction approaches for biological sequences: a comparative study of mathematical features. Brief Bioinform 2021;22:bbab011.
[22] Fickett JW. Recognition of protein coding regions in DNA sequences. Nucleic Acids Res 1982;10:5303–18.
[23] Wang L, Park HJ, Dasari S, Wang S, Kocher JP, Li W. CPAT: coding-potential assessment tool using an alignment-free logistic regression model. Nucleic Acids Res 2013;41:e74.
[24] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in python. J Mach Learn Res 2011;12:2825–30.
[25] Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 2002;16:321–57.

[26] Lemaître G, Nogueira F, Aridas CK. Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. J Mach Learn Res 2017;18:559–63.
[27] Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: a next-generation hyperparameter optimization framework. arXiv 2019;1907.10902
[28] Limasset A, Flot JF, Peterlongo P. Toward perfect reads: self-correction of short reads via mapping on de Bruijn graphs. Bioinforma Oxf Engl 2020;36:1374–81.
[29] Mitchell K, Brito JJ, Mandric I, Wu Q, Knyazev S, Chang S, et al. Benchmarking of computational error-correction methods for next-generation sequencing data. Genome Biol 2020;21:71.
[30] Zhang X, Ping P, Hutvagner G, Blumenstein M, Li J. Aberration-corrected ultrafine analysis of miRNA reads at single-base resolution: a k-mer lattice approach. Nucleic Acids Res 2021;49:e106.
[31] Kinde I, Wu J, Papadopoulos N, Kinzler KW, Vogelstein B. Detection and quantification of rare mutations with massively parallel sequencing. Proc Natl Acad Sci U S A 2011;108:9530–5.
[32] Liu Y, Schröder J, Schmidt B. Musket: a multistage k-mer spectrum-based error corrector for Illumina sequence data. Bioinformatics 2013;29:308–15.
[33] Ilie L, Molnar M. RACER: rapid and accurate correction of errors in reads. Bioinforma Oxf Engl 2013;29:2490–3.
[34] Song L, Florea L, Langmead B. Lighter: fast and memory-efficient sequencing error correction without counting. Genome Biol 2014;15:509.
[35] Li H. BFC: correcting Illumina sequencing errors. Bioinforma Oxf Engl 2015;31:2885–7.
[36] Marinier E, Brown DG, McConkey BJ. Pollux: platform independent error correction of single and mixed genomes. BMC Bioinformatics 2015;16:10.
[37] Heo Y, Wu XL, Chen D, Ma J, Hwu WM. BLESS: bloom filter-based error correction solution for high-throughput sequencing reads. Bioinforma Oxf Engl 2014;30:1354–62.
[38] Dlugosz M, Deorowicz S. RECKONER: read error corrector based on KMC. Bioinforma Oxf Engl 2017;33:1086–9.
[39] Salmela L, Schröder J. Correcting errors in short reads by multiple alignments. Bioinformatics 2011;27:1455–61.
[40] Schulz MH, Weese D, Holtgrewe M, Dimitrova V, Niu S, Reinert K, et al. Fiona: a parallel and automatic strategy for read error correction. Bioinformatics 2014;30:i356–63.
[41] Allam A, Kalnis P, Solovyev V. Karect: accurate correction of substitution, insertion and deletion errors for next-generation sequencing data. Bioinforma Oxf Engl 2015;31:3421–8.
[42] Kallenborn F, Hildebrandt A, Schmidt B. CARE: context-aware sequencing read error correction. Bioinformatics 2021;37:889–95.
[43] Kallenborn F, Cascitti J, Schmidt B. CARE 2.0: reducing false-positive sequencing error corrections using machine learning. BMC Bioinformatics 2022;23:227.

[44] Simpson JT, Durbin R. Efficient *de novo* assembly of large genomes using compressed data structures. Genome Res 2012;22:549–56.
[45] Petukhov V, Guo J, Baryawno N, Severe N, Scadden DT, Samsonova MG, et al. dropEst: pipeline for accurate estimation of molecular counts in droplet-based single-cell RNA-seq experiments. Genome Biol 2018;19:78.
[46] Sereewit J, Lieberman NAP, Xie H, Bakhash SAKM, Nunley BE, Chung B, et al. ORF-interrupting mutations in monkeypox virus genomes from Washington and Ohio, 2022. Viruses 2022;14:2393.
[47] Iwakawa H, Tomari Y. The functions of microRNAs: mRNA decay and translational repression. Trends Cell Biol 2015;25:651–65.
[48] Telonis AG, Magee R, Loher P, Chervoneva I, Londin E, Rigoutsos I. Knowledge about the presence or absence of miRNA isoforms (isomiRs) can successfully discriminate amongst 32 TCGA cancer types. Nucleic Acids Res 2017;45:2973–85.
[49] Meng L, Liu C, Lü J, Zhao Q, Deng S, Wang G, et al. Small RNA zippers lock miRNA molecules and block miRNA function in mammalian cells. Nat Commun 2017;8:13964.
[50] Martí E, Pantano L, Bañez-Coronel M, Llorens F, Miñones-Moyano E, Porta S, et al. A myriad of miRNA variants in control and Huntington's disease brain regions detected by massively parallel sequencing. Nucleic Acids Res 2010;38:7219–35.
[51] Lappalainen T, Sammeth M, Friedländer MR, 't Hoen PAC, Monlong J, Rivas MA, et al. Transcriptome and genome sequencing uncovers functional variation in humans. Nature 2013;501:506–11.
[52] Martin M. Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet J 2011;17:10–2.
[53] Loher P, Karathanasis N, Londin E, Bray PF, Pliatsika V, Telonis AG, et al. IsoMiRmap: fast, deterministic and exhaustive mining of isomiRs from short RNA-seq datasets. Bioinformatics 2021;37:1828–38.
[54] Kozomara A, Birgaoanu M, Griffiths-Jones S. miRBase: from microRNA sequences to function. Nucleic Acids Res 2019;47:D155–62.
[55] Gaudelli NM, Komor AC, Rees HA, Packer MS, Badran AH, Bryson DI, et al. Programmable base editing of A•T to G•C in genomic DNA without DNA cleavage. Nature 2017;551:464–71.
[56] Rees HA, Liu DR. Base editing: precision chemistry on the genome and transcriptome of living cells. Nat Rev Genet 2018;19:770–88.
[57] Song M, Kim HK, Lee S, Kim Y, Seo SY, Park J, et al. Sequence-specific prediction of the efficiencies of adenine and cytosine base editors. Nat Biotechnol 2020;38:1037–43.
[58] Satam H, Joshi K, Mangrolia U, Waghoo S, Zaidi G, Rawool S, et al. Next-generation sequencing technology: current trends and advancements. Biology 2023;12:997.
[59] Polonis K, Blommel JH, Hughes AEO, Spencer D, Thompson JA, Schroeder MC. Innovations in short-read sequencing technologies and their applications to clinical genomics. Clin Chem 2025;71:97–108.

[60] Salk JJ, Schmitt MW, Loeb LA. Enhancing the accuracy of next-generation sequencing for detecting rare and subclonal mutations. Nat Rev Genet 2018;19:269–85.
[61] Nayfach S, Páez-Espino D, Call L, Low SJ, Sberro H, Ivanova NN, et al. Metagenomic compendium of 189,680 DNA viruses from the human gut microbiome. Nat Microbiol 2021;6:960–70.
[62] Marchetti F, Cardoso R, Chen CL, Douglas GR, Elloway J, Escobar PA, et al. Error-corrected next generation sequencing – promises and challenges for genotoxicity and cancer risk assessment. Mutat Res Rev Mutat Res 2023;792:108466.

Figure legends

Figure 1 Schematic diagrams illustrating how base errors are generated during library preparation and sequencing process

A. Schematic illustration of base error generation when amplifying one DNA template during conventional polymerase chain reaction (PCR) amplification. Base "T" mutated to "G" between the third cycle and the fourth cycle and this error is inherited by the subsequent cycles. **B.** Schematic graph depicting PCR errors generated in the process of bridge amplification during Illumina sequencing. An example of "A"-to-"G" is inherited. **C.** An overview of base calling during Illumina sequencing.

Figure 2 Overview of the workflow of noise2read

The first stage (1a–1f) and the second stage (2a–2f) rectify 1-nt and 2-nt based-errors to their normal states, respectively. The third stage (3a–3f) is optional only for further correction specified to the amplicon sequencing data. The integrative auto machine learning (AutoML) module is used multiple times for training and predicting based on different edit-erring-reads and error-free-reads in each stage.

Figure 3 Zoomed-in view of subgraph A in Figure S1

This subgraph contains six high-frequency (16 to 234) reads labelled as r_1 to r_7 and 72 low frequency (1 to 3) reads.

Figure 4 An overview workflow of the AutoML module for end-to-end prediction on ambiguous errors

The edit-erring-reads and error-free-reads extracted from the nt-edit-distance graph

are categorised into three types of data. Training data is constructed through the workflow steps of ②③–⑤⑥–⑧–⑨–⑩; the scaled training data is then fed (⑪) into XGBoost classifier, with Optuna used to optimize parameters, resulting in the best prediction model (⑫). Following similar preprocessing steps, the transformed objective data is created through steps of ①–④–⑦–⑭. Finally, the prediction is completed by feeding the objective data into the optimized model via steps ⑮–⑯.

Figure 5 Flowchart illustrating the analytical framework and key concepts in this study

Figure 6 Visualisation of information gain for different methods on dataset $D1$

A. and **B.** Information gain by noise2read with and without amplicon correction, respectively. **C.**–**I.** Incorrectly introduced reads as red points. The number of red points shown on each heatmap corresponds to 502, 2310, 7808, 2935, 8523, 13,899 and 722, respectively.

Figure 7 Performance comparison between noise2read and seven other methods on datasets $D1$ and $D9$

A. Comparison of true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), false negative rate (FNR), and area difference (AD), at the read-level for noise2read and seven other methods on the UMI-based wet-lab dataset $D1$. noise2read* denotes the result without amplicon correction. **B.**–**D.** Performance comparisons at the read-level on simulated dataset $D9$. **E.**–**L.** Information gain visualisations for $D9$. Heatmaps in **F**–**L** display 1223, 164, 9698, 377, 2378, 3651 and 1255 red dots, respectively. Each red dot represents a new sequence introduced after error correction.

Figure 8 Comparison of base coverage before and after correction for Monkeypox virus genome using perfectly matched reads

A. Base coverage for Monkeypox virus using the original and corrected sequencing data. **B.** Coverage differences before and after error correction for the Monkeypox virus data. **C.** Frequency distribution of coverage differences for the Monkeypox virus data, also plotted with scaled density curves.

Figure 9 Comparison of isomiR and known SNP counts before and after error correction using scatter plots and Kepler plots

A. and **B.** Scatter plots comparing the number of exclusive isomiRs identified in the original and error-corrected datasets $D28$ and $D29$, respectively. **C.** and **D.** Comparisons for ambiguous isomiRs and known SNPs identified in the original and error-corrected dataset $D28$, respectively.

Tables

Table 1 Non-frequent reads' information gain $\triangle H$ on the datasets $D1 - D8$ and $D9 - D13$

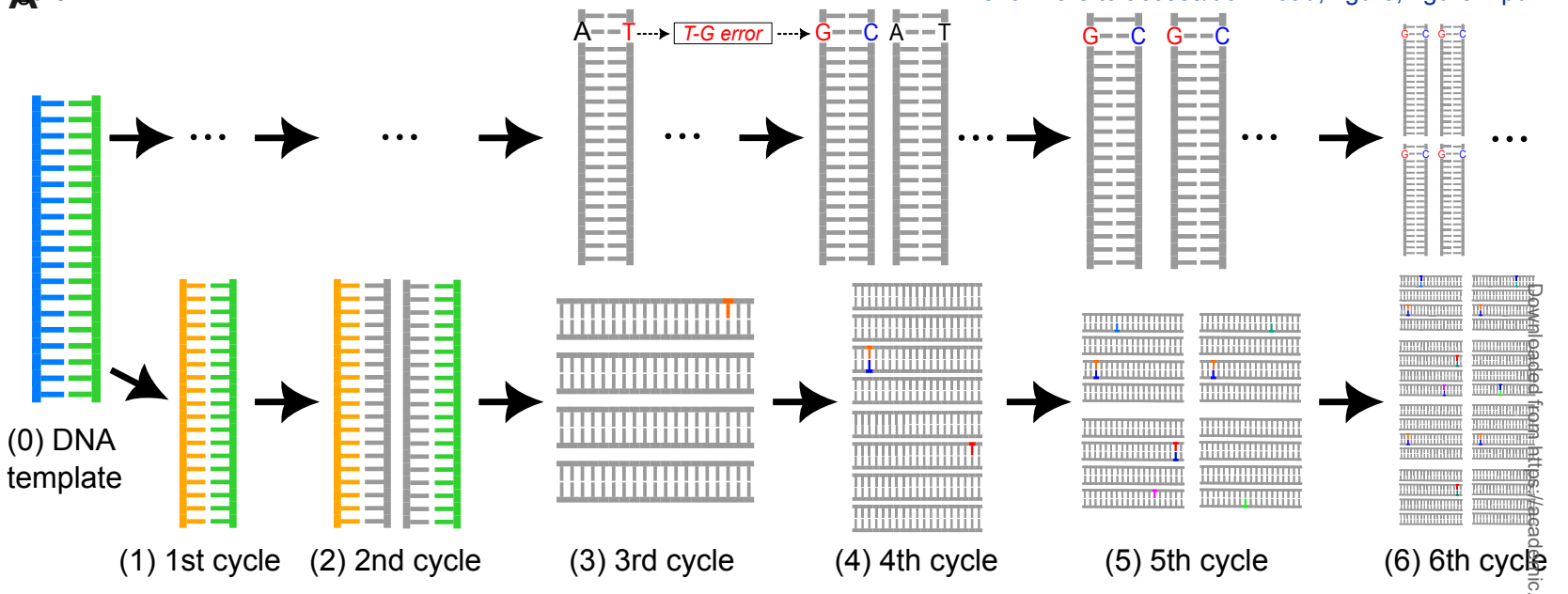
Table 2 Performance comparison between noise2read and seven methods on the dataset $D1$

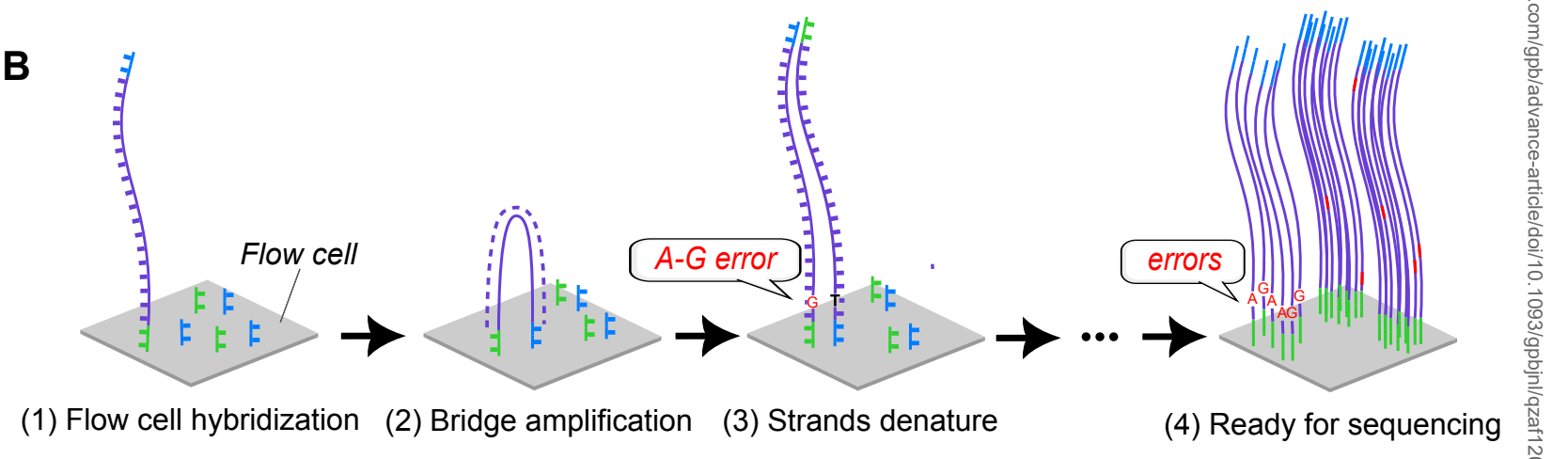
Table 3 Performance comparison between noise2read and miREC at the read level

Table 4 Performance comparison between noise2read and ten methods on the dataset $D25$

Table 3 Time and memory usage by different methods on the datasets $D1 - D8$

Table 4 Known isomiRs and SNPs profiling change from miRNA sequencing data before and after correction





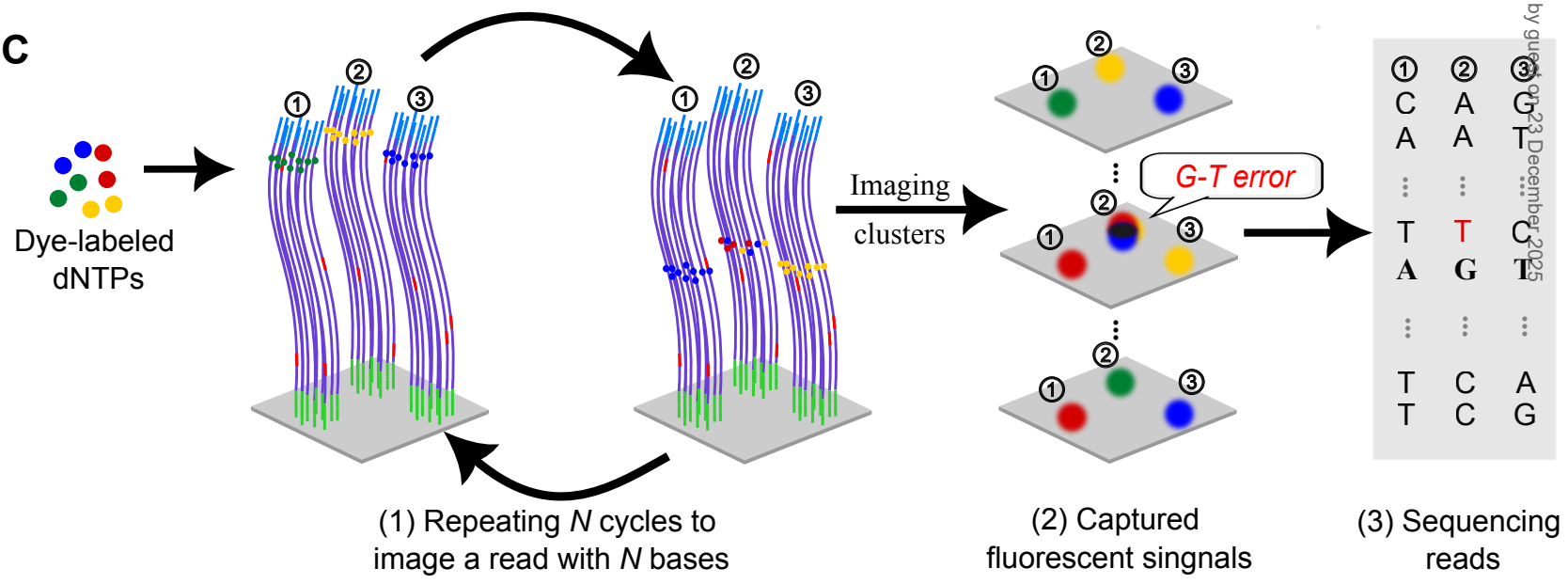


Figure 2

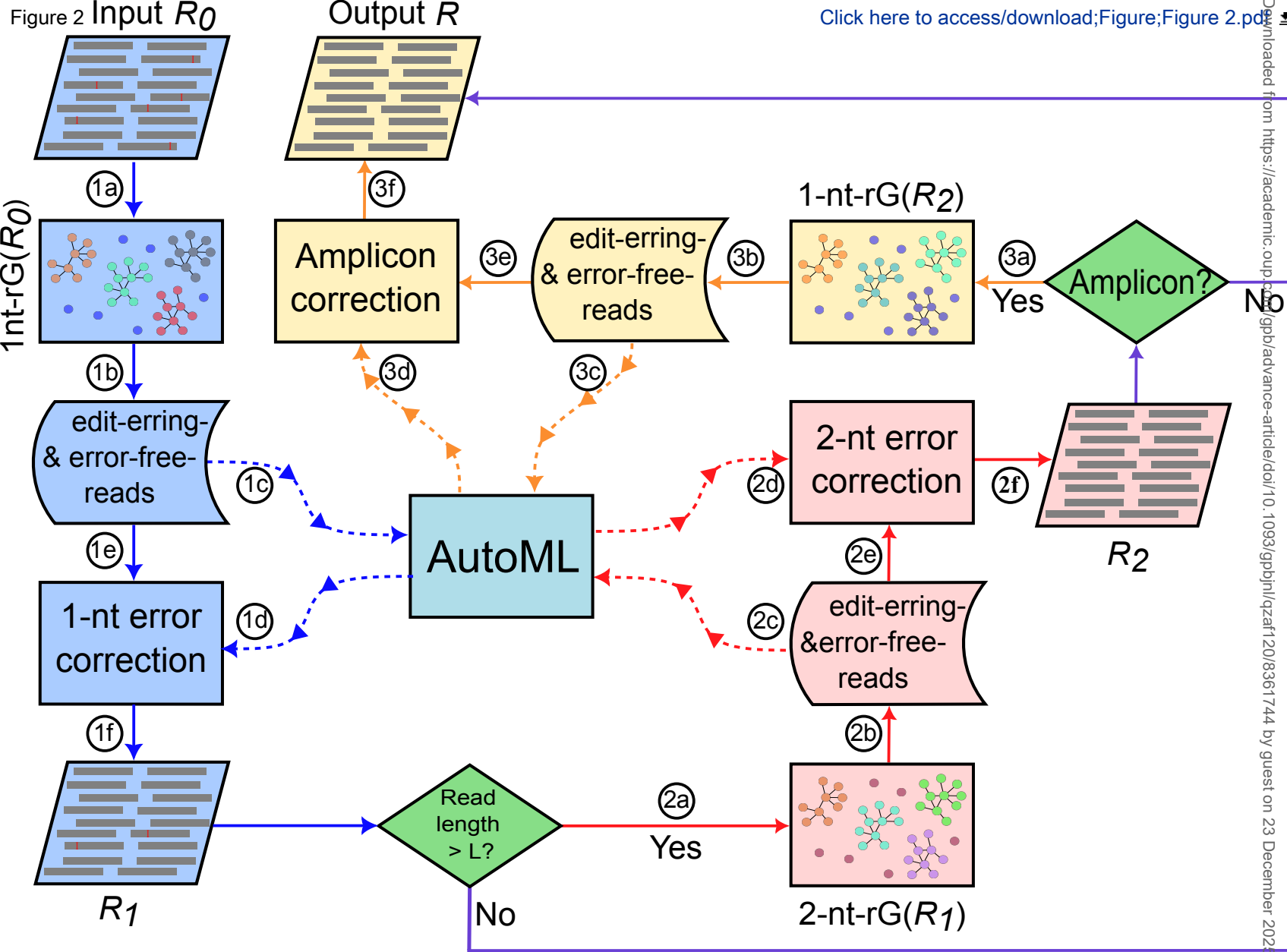


Figure 3

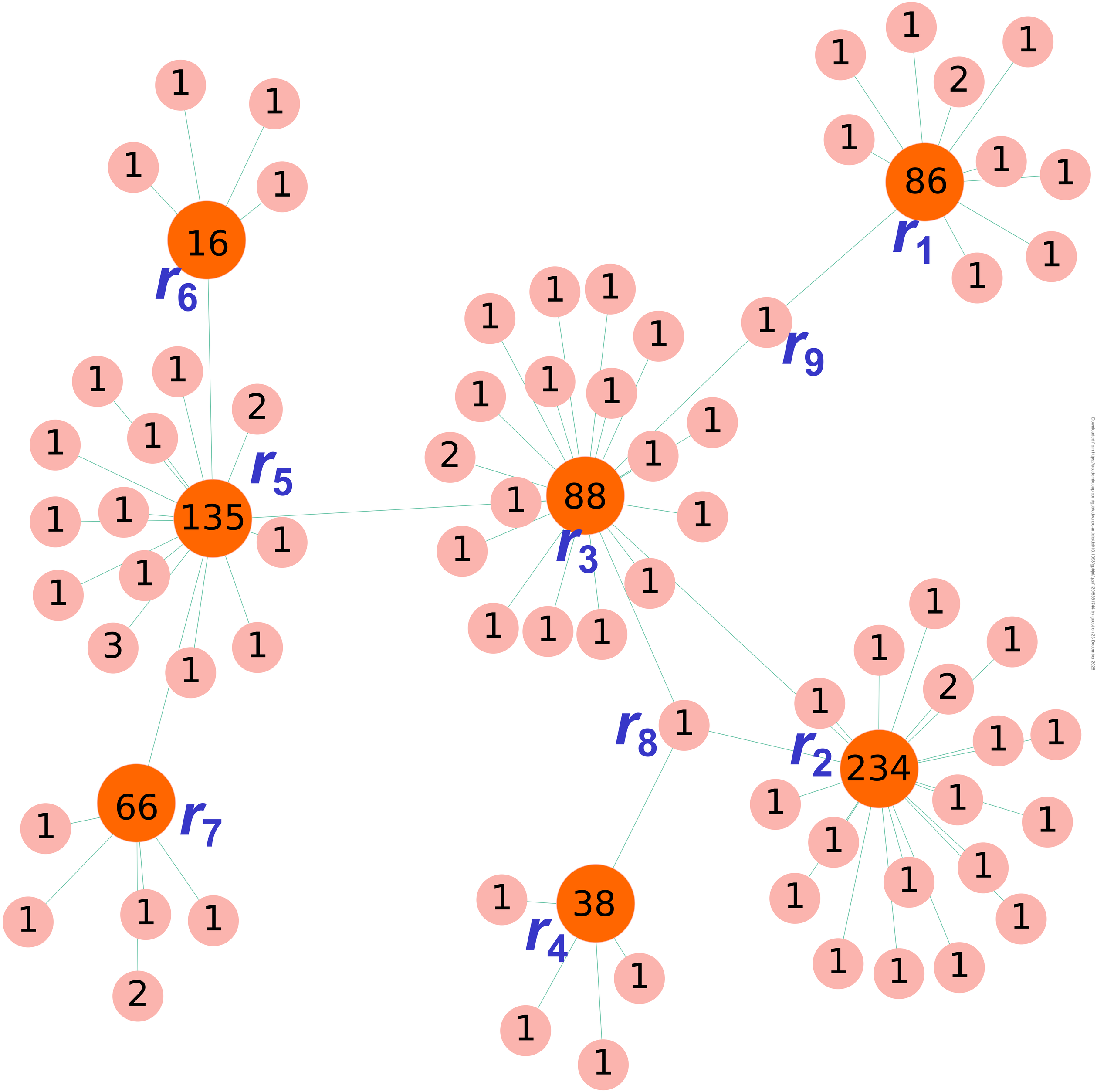


Figure 4

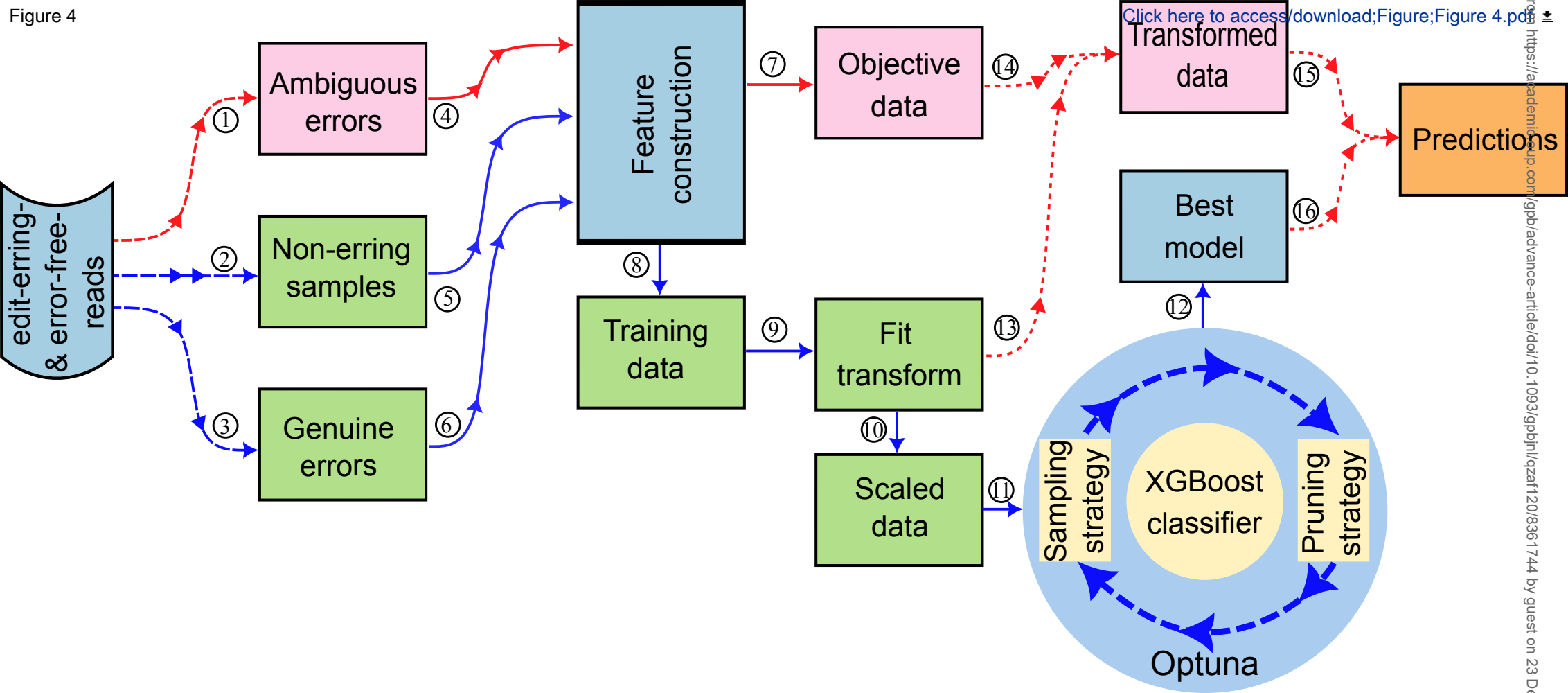


Figure 5

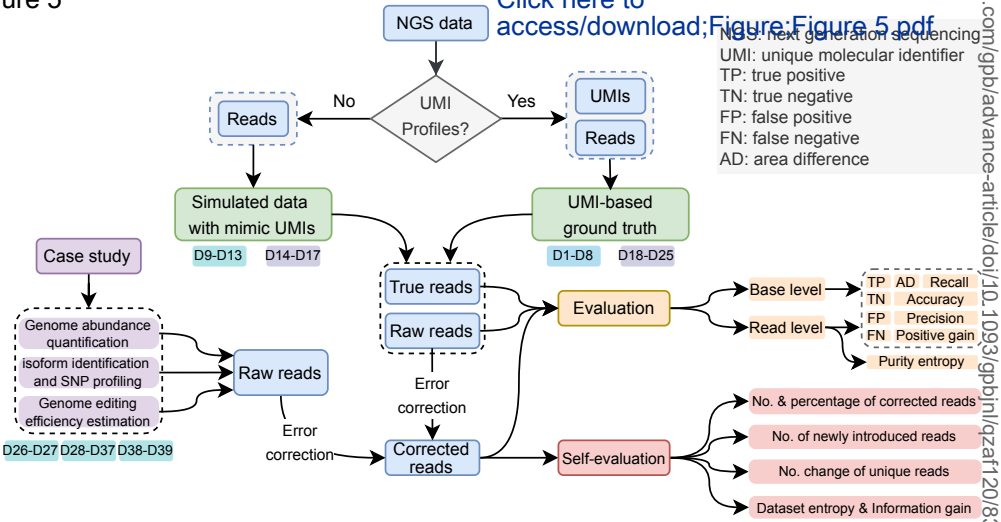
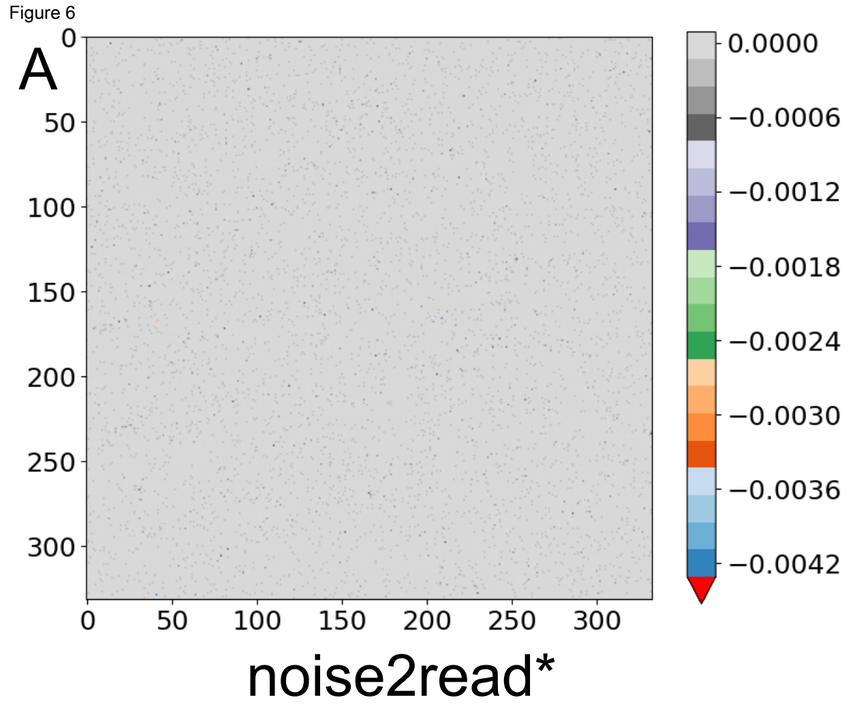
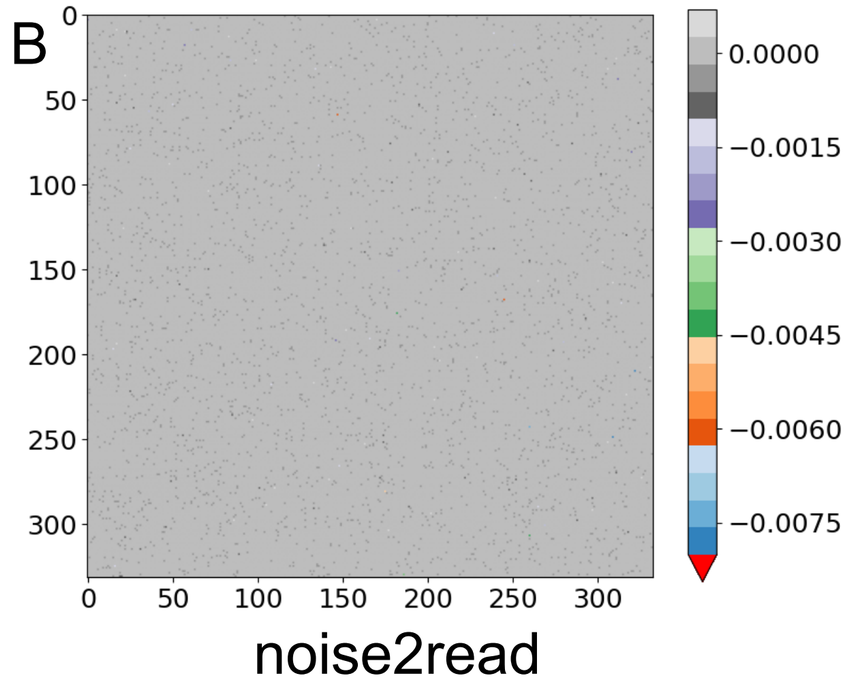
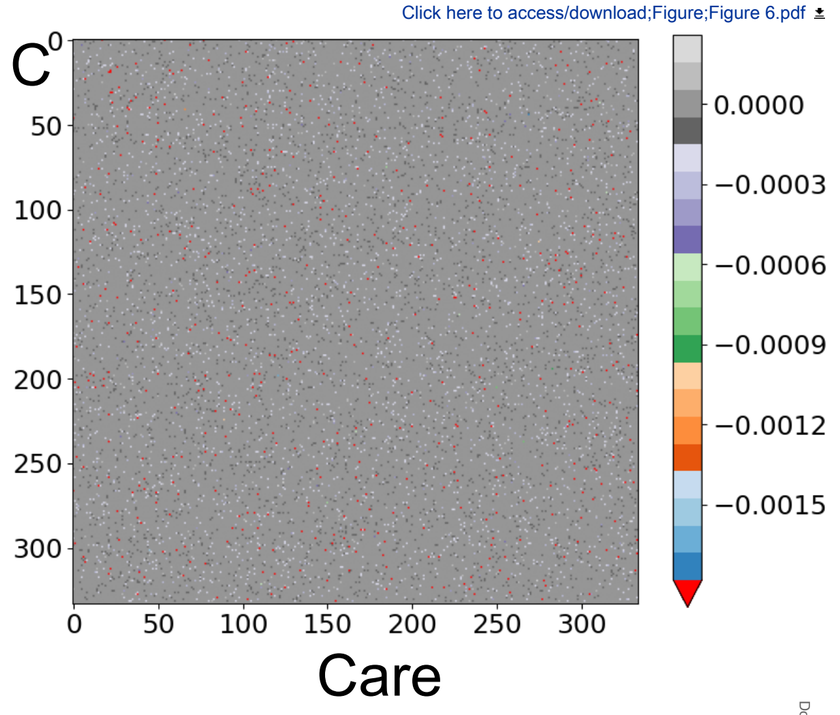
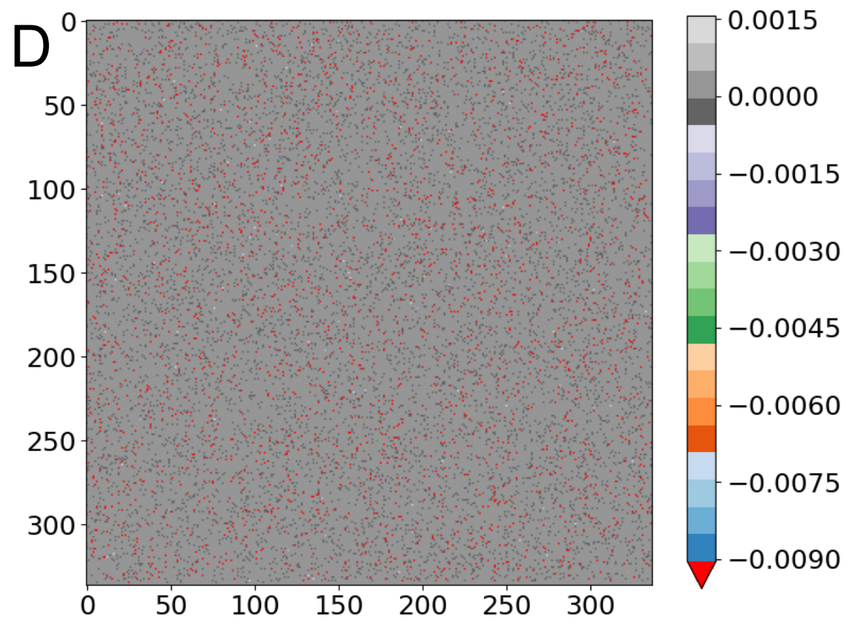


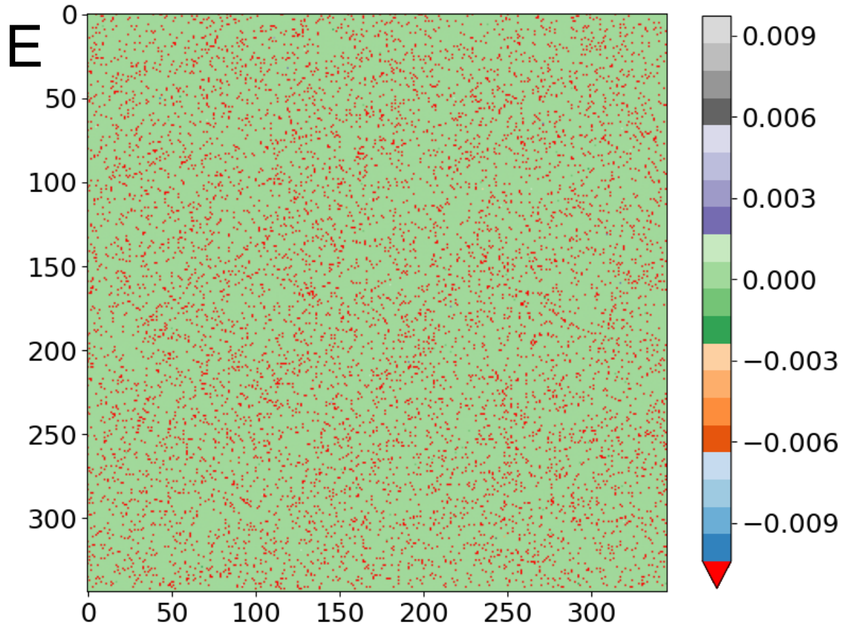
Figure 6

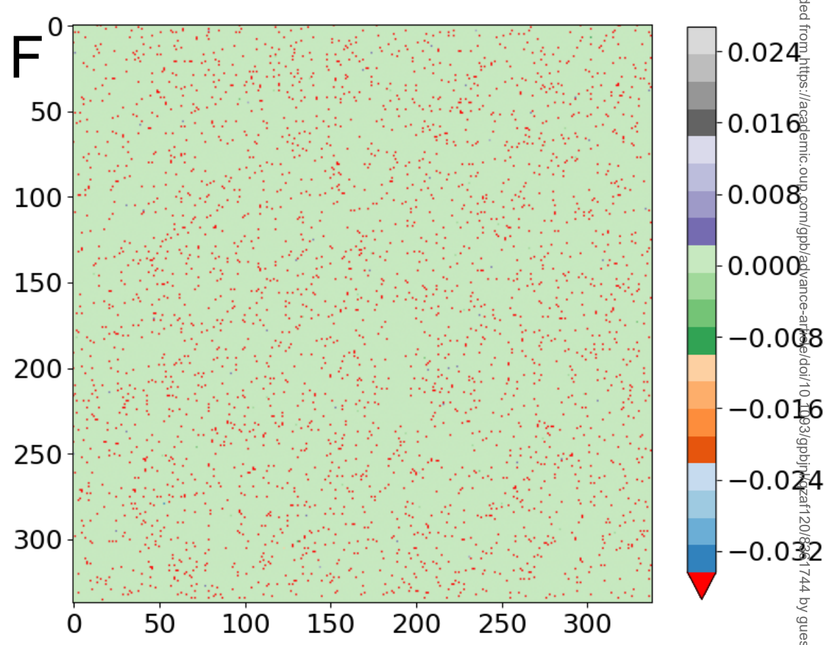


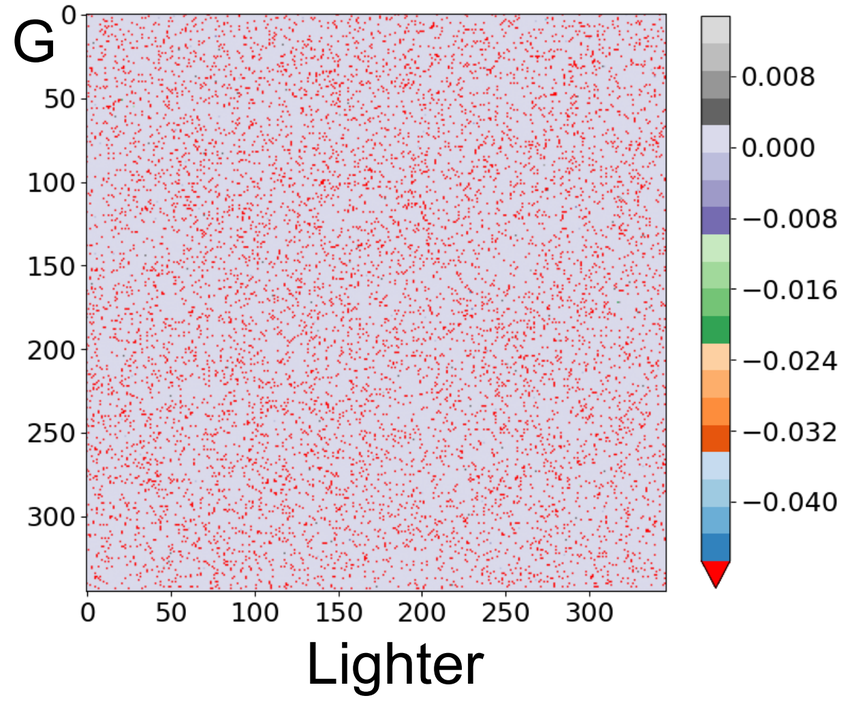


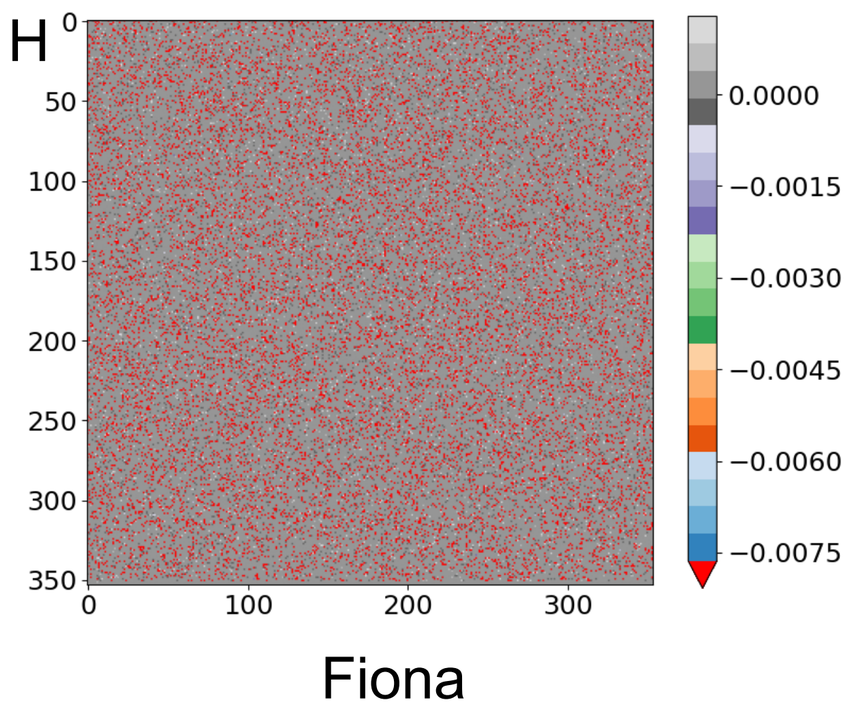


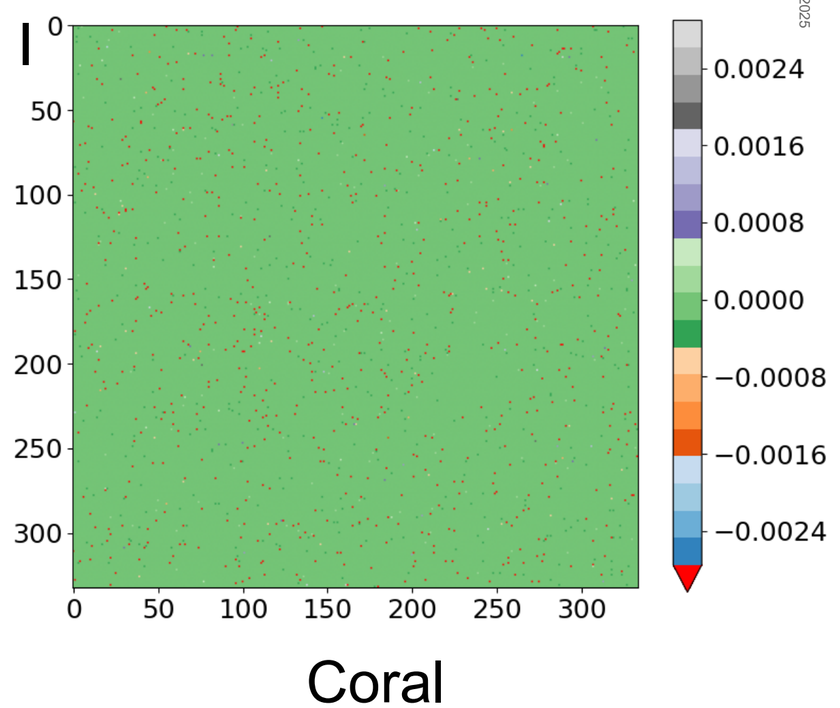


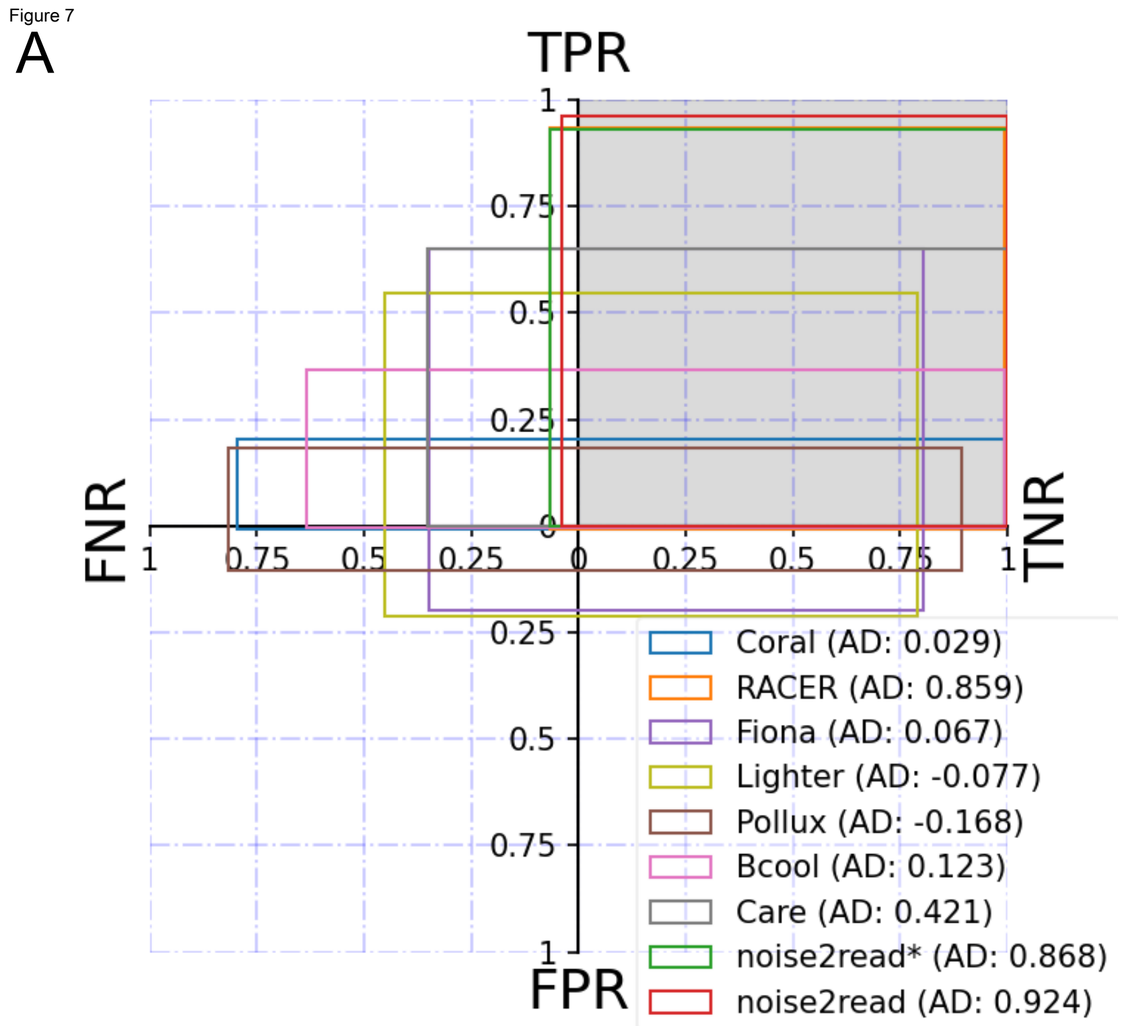


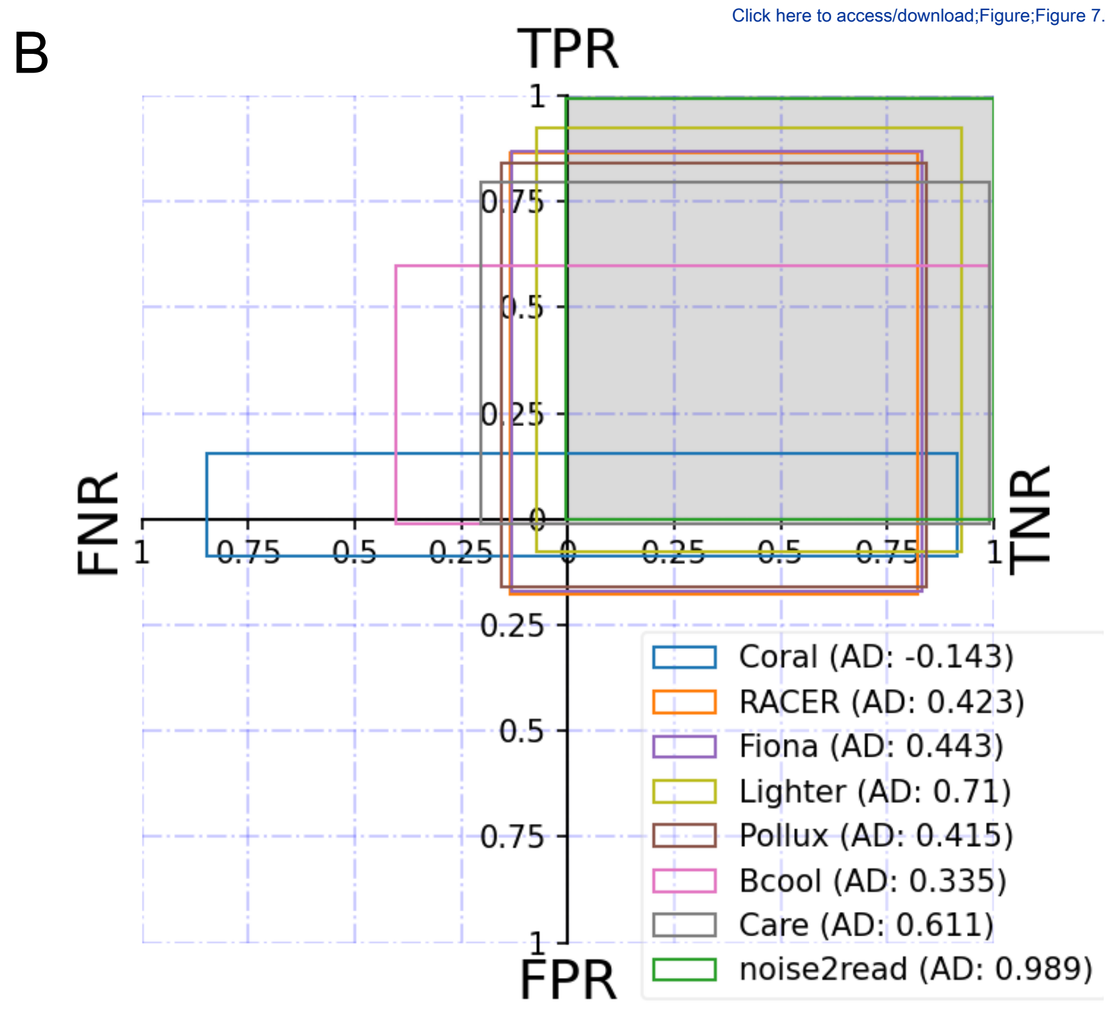


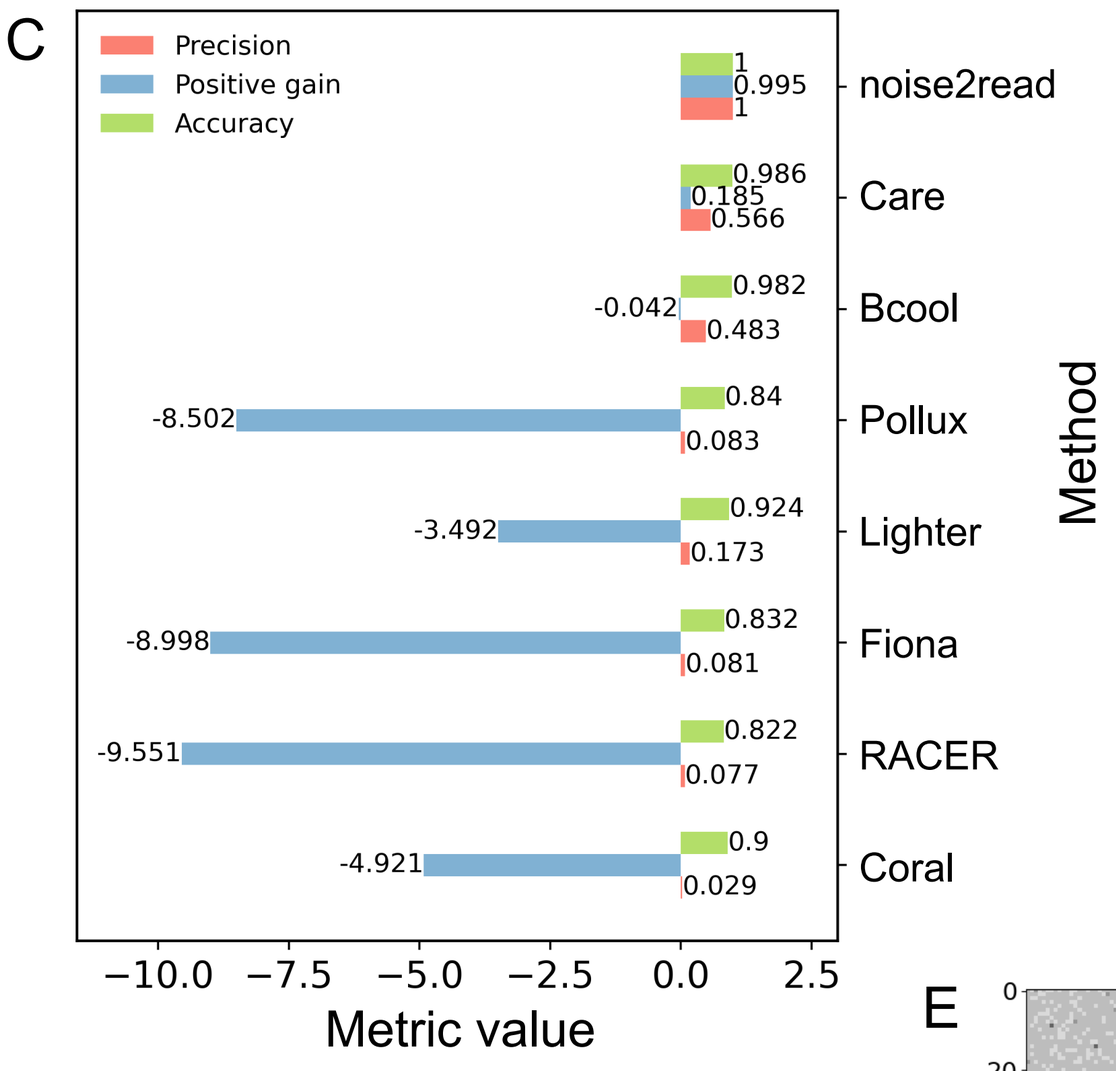


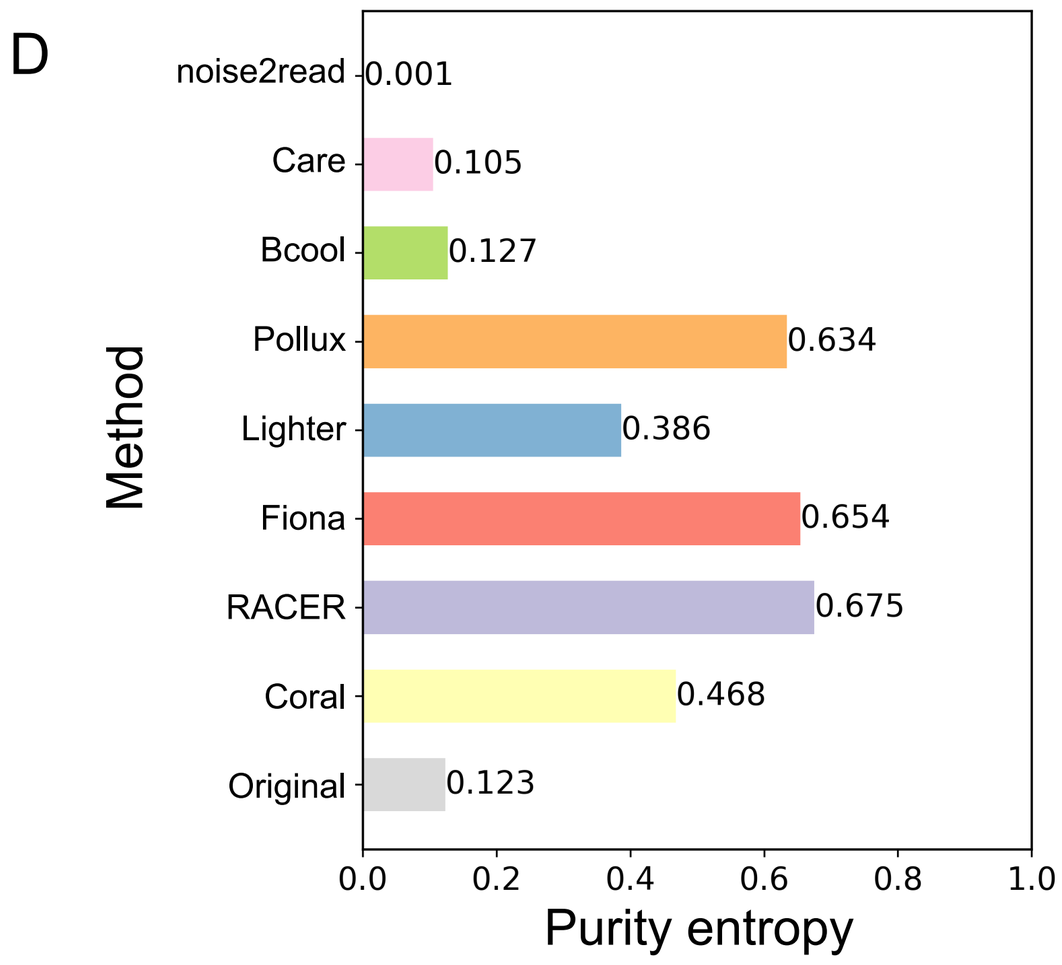


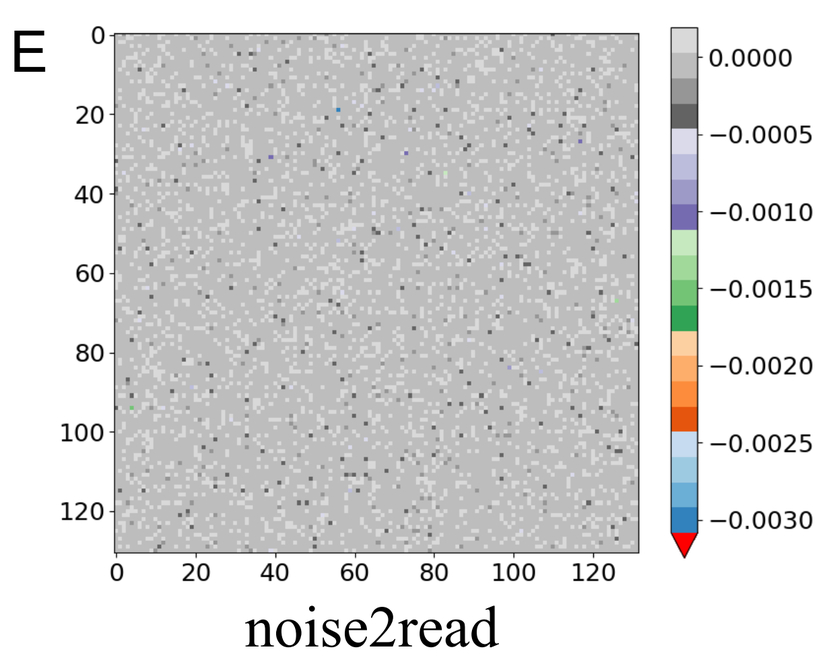


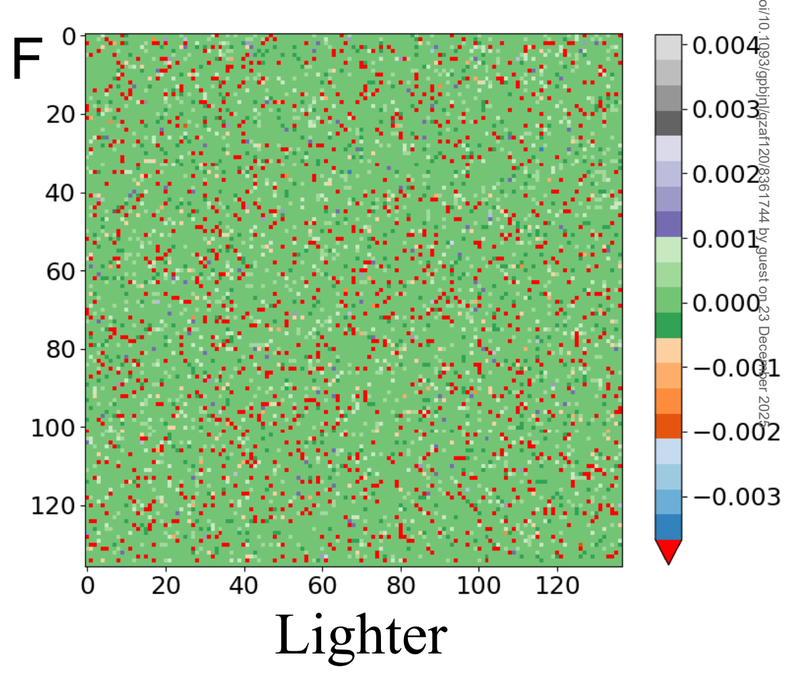


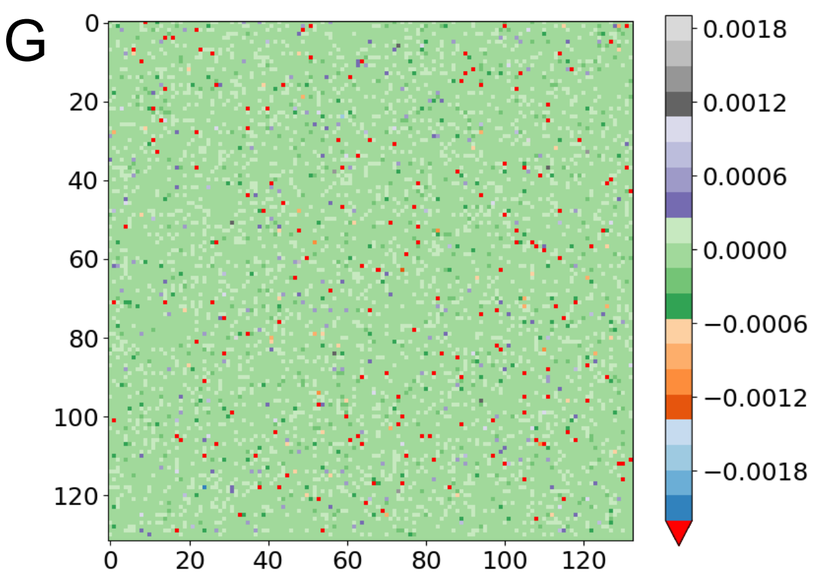


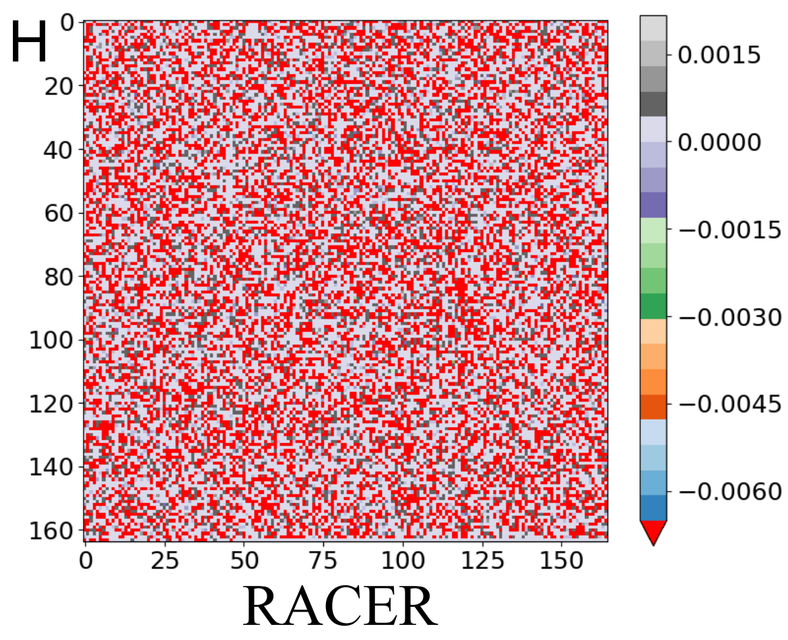


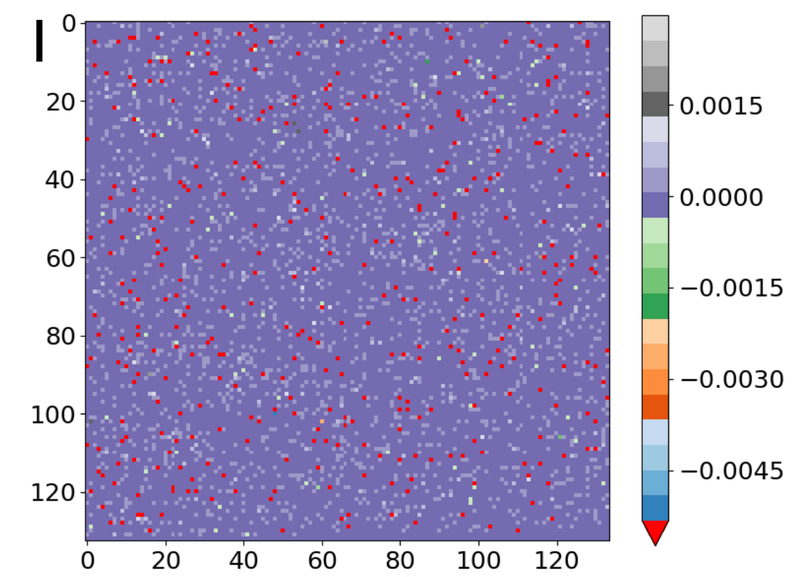


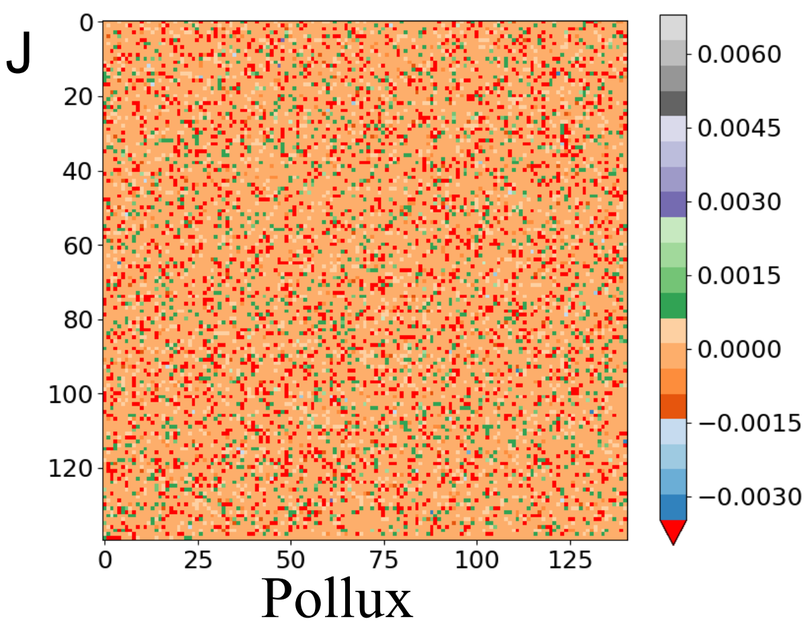


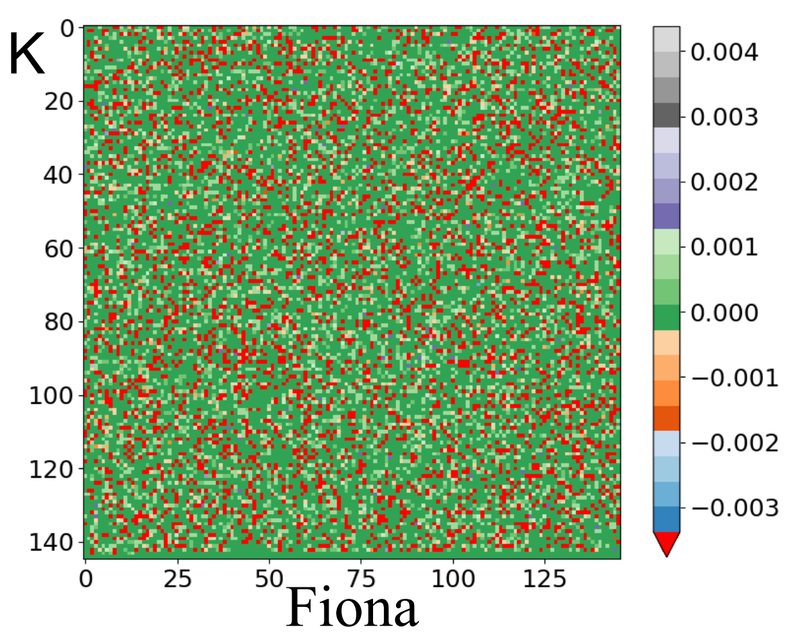












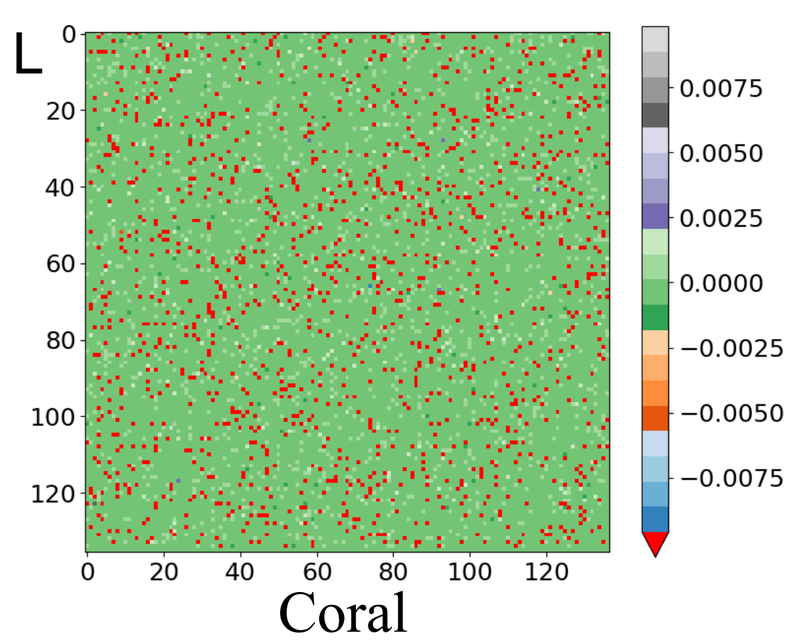


Figure 8

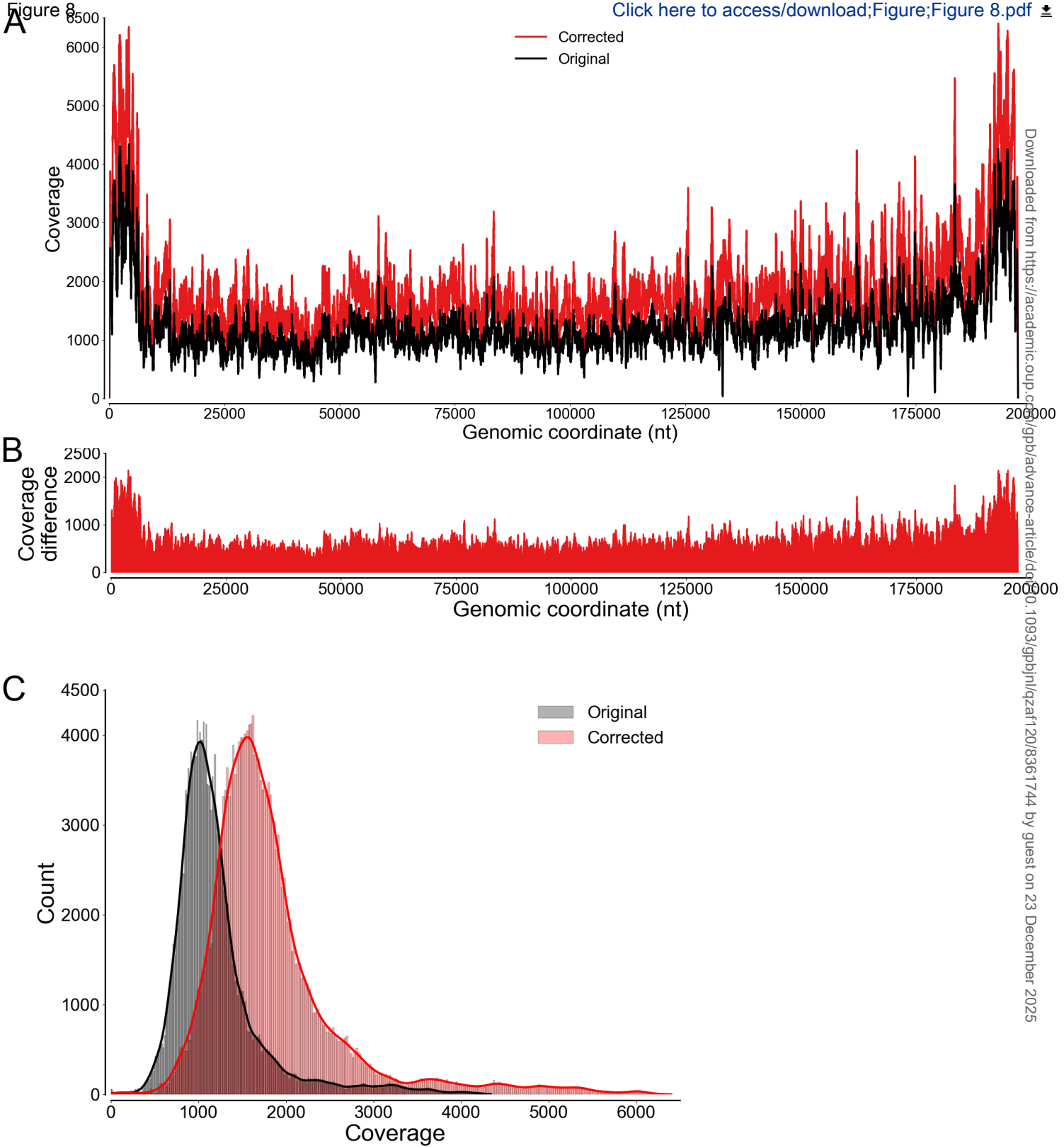


Figure 9

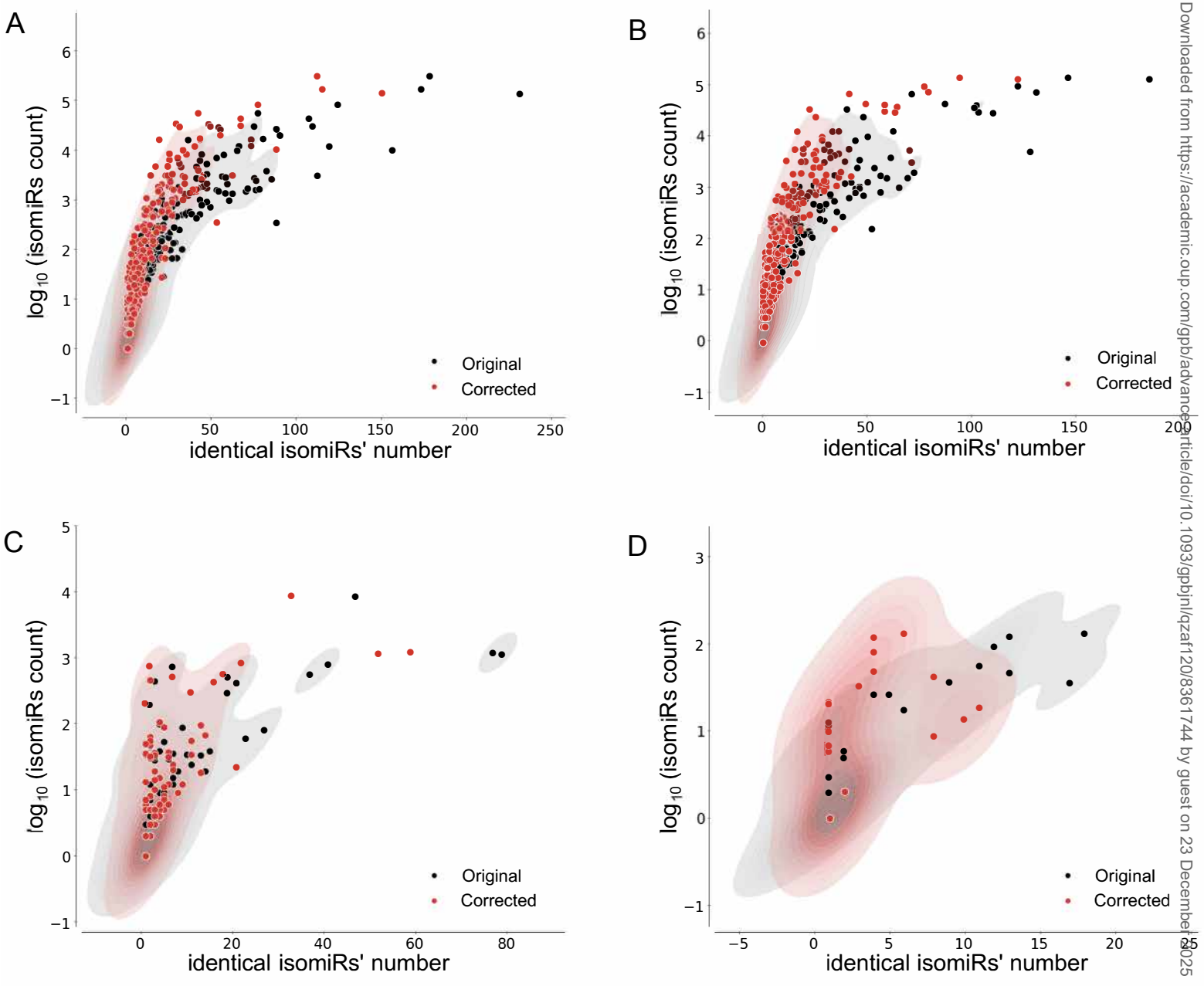


Table 1 Non–frequent reads' information gain $\triangle H$ on the datasets $D1-D8$ and $D9-D13$

Datasets	Method							
	Coral	**RACER**	**Fiona**	**Lighter**	**Pollux**	**Bcool**	**Care**	**noise2read**
D1	0.588	6.023	4.077	4.686	2.508	0.916	1.776	**6.409**
D2	0.770	6.116	4.393	4.842	2.534	0.023	1.557	**6.339**
D3	0.921	**6.369**	4.891	5.505	3.044	2.333	0.469	6.239
D4	0.590	**6.215**	4.638	5.083	3.011	1.702	0.907	6.202
D5	0.752	**6.360**	4.589	5.127	2.821	0.046	1.302	6.297
D6	0.970	6.316	4.800	5.026	2.930	1.986	1.357	**6.567**
D7	0.898	**6.306**	4.883	5.422	3.096	0.100	0.446	6.194
D8	1.156	5.974	4.344	4.604	3.014	1.824	1.598	**6.187**
Average	0.831	6.210	4.577	5.037	2.870	1.116	1.176	**6.304**
D9	0.621	–1.349	–0.209	1.362	0.462	1.874	2.941	**10.484**
D10	2.072	–0.673	3.318	3.040	3.152	0.960	1.721	**12.733**
D11	2.849	0.273	2.129	3.910	3.459	0.723	1.483	**12.380**
D12	1.250	0.039	2.476	4.535	2.784	1.103	0.881	**12.249**
D13	1.159	0.811	2.991	5.193	3.159	1.725	0.909	**13.161**
Average	1.590	–0.180	2.141	3.608	2.603	1.277	1.587	**12.201**

Note: $\tau = 4$ was used for calculating $\triangle H$. $D1-D8$ are UMI–based wet–lab datasets and $D9-D13$ are UMI–based simulated datasets. Best scores are highlighted in bold.

Table 2 Performance comparison between noise2read and seven methods on the dataset $D1$

Method		Metric								
		TP	**FP**	**FN**	**TN**	**Recall**	**Precision**	**Positive Gain**	**Accuracy**	**E**
noise2read[a]	1st stage	81630	0	30391	693059	0.729	1.000	0.729	0.962	0.232
	2nd stage	104373	**0**	7648	693059	0.932	**1.000**	0.932	0.991	0.077
	3rd stage	**107717**	201	**4304**	692858	**0.962**	0.998	**0.960**	**0.994**	**0.050**
Coral		22677	4249	89344	688810	0.202	0.842	0.165	0.884	0.518
RACER		104589	4347	7432	688712	0.934	0.960	0.895	0.985	0.110
Fiona		72792	136472	39229	556587	0.650	0.348	–0.568	0.782	0.757
Lighter		61330	145999	50691	547060	0.547	0.296	–0.756	0.756	0.802
Pollux		20537	73710	91484	619349	0.183	0.218	–0.475	0.795	0.732
Bcool		40970	3889	71051	689170	0.366	0.913	0.331	0.907	0.447
Care		72673	52	39348	693007	0.649	0.999	0.648	0.951	0.3

Note: [a]The results obtained by noise2read was decomposed at different stages. $D1$ is a UMI-based wet-lab dataset. Best scores are highlighted in bold.

Table 3 Performance comparison between noise2read and miREC at the read level

Datasets	Methods	Metrics							
		TP	FP	FN	TN	Recall	Gain	E	ΔH
D14	miREC	4224	0	267	218980	0.941	0.941	0.129	4.755
	noise2read	4410	15	81	218965	0.982	0.979	0.137	9.053
	noise2read[a]	4408	3	83	218977	0.982	0.981	0.137	9.053
D15	miREC	4135	5	271	219060	0.938	0.937	0.126	4.821
	noise2read	4312	10	94	219055	0.979	0.976	0.134	8.485
	noise2read[a]	4310	4	96	219061	0.978	0.977	0.134	8.485
D16	miREC	6418	16	309	216728	0.954	0.952	0.179	5.301
	noise2read	6590	20	137	216724	0.980	0.977	0.187	8.184
	noise2read[a]	6588	16	139	216728	0.979	0.977	0.187	8.184
D17	miREC	6398	0	306	216767	0.954	0.954	0.179	5.337
	noise2read	6578	2	126	216765	0.981	0.981	0.187	8.769
	noise2read[a]	6576	0	128	216767	0.981	0.981	0.187	8.769

Note: High-frequency threshold $\tau = 4$ used for noise2read. $D14 - D17$ are simulated miRNA sequencing datasets. $D14 - D15$ contain substitution and indel errors, while $D16 - D17$ contain only substitution errors. [a] Performance by noise2read without prediction of errors between high frequency reads.

Table 4 Performance comparison between noise2read and ten methods on the dataset $D25$

Method	k-mer size	Metric											
		TP	**TN**	**FN**	**FP**	**TPR**	**FNR**	**TNR**	**FPR**	**Precision**	**Accuracy**	**AD**	**Positive Gain**
Bless	30	39345	509513	23498	1751	0.63	0.37	1	0	0.957	0.96	0.39	0.6
Coral	30	23172	497906	48255	4774	0.32	0.68	0.99	0.01	0.829	0.91	0.09	0.26
Lighter	30	51934	497165	19336	5672	0.73	0.27	0.99	0.01	0.902	0.96	0.51	0.65
Reckoner	30	24143	501767	47233	964	0.34	0.66	1	0	0.962	0.92	0.11	0.32
Sga	26	13129	501767	58582	629	0.18	0.82	1	0	0.954	0.9	0.03	0.17
BFC	30	18415	500964	53345	1383	0.26	0.74	1	0	0.93	0.9	0.06	0.24
Pollux	30	26308	430041	33643	83210	0.44	0.56	0.84	0.16	0.24	0.8	–0.11	–0.95
Fiona	NA	54983	483470	13675	21979	0.8	0.2	0.96	0.04	0.714	0.94	**0.56**	0.48
RACER	NA	50106	444352	9857	69792	0.84	0.16	0.86	0.14	0.418	0.86	0.45	–0.33
Care	NA	43213	501631	28896	367	0.6	0.4	1	0	0.992	0.95	0.36	0.59
noise2read	NA	54316	501759	18011	21	0.75	0.25	1	0	**0.9996**	**0.97**	**0.56**	**0.75**

Note: $D25$ is a UMI-based benchmark dataset previously established in the literature [29].

Table 5 Time and memory usage by different methods on the datasets $D1 - D8$

Method	CPU cores	D1		D2		D3		D4		D5		D6		D7		D8	
		Time	Memory	Time	Memory	Time	Memory	Time	Memory	Time	Memory	Time	Memory	Time	Memory	Time	Memory
Coral	56	3.3	65528	3.4	91045	3.0	54182	3.8	99376	3.9	99673	3.3	68805	3.7	61512	2.4	49311
Fiona		36.4	1736	20.2	1702	22.5	1811	25.2	2111	29.1	1794	33.4	1755	19.9	1846	15.2	1353
Lighter		1	568	1	564	1	568	1	568	1	568	1	564	1	566	1	566
RACER	64	3.1	111	2.7	100	3.7	130	4.1	146	3.4	123	3.4	125	3.7	135	2.2	110
Pollux	1	1277.4	219	1072.2	197	1445.3	204	1722.2	217	1425.1	209.07	1455.3	219	1480.1	217	844.9	198
Bcool	56	15.9	12	4.6	12	8.1	12	11.7	12	3.9	12	11.6	12	5.5	12	5.3	12
Care		1	788	1	793	1.0	813	1	737	1	812	1	823	1	822	1	589
noise2read		137.3	4405	121.5	4012	126.8	6629	143.1	5723	125.8	4755	136.3	6557	123.0	5393	110.2	3472
noise2read[a]		171.0	4373	161.0	5350	146.0	4824	199.0	7449	173.0	4699	178.0	5189	160.0	5014	198.0	3767

Note: The CPU model of Intel(R) Xeon(R) Gold 6238R CPU @ 2.20GHz was used by all the methods. 1 GPU of Tesla V100S-PCIE-32GB was used for the model training of noise2read. The runtime is given in minutes; Memory consumption is given in MB. [a] The performance of noise2read is enhanced through additional amplicon correction.

Table 6 Known isomiRs and SNPs profiling change from miRNA sequencing data before and after correction

Dataset	Corr. read No.	Corr. PCT.	Ambiguous isomiRs						Exclusive isomiRs						SNPs					
			Unique reads No.			Total reads No.			Unique reads No.			Total reads No.			Unique reads No.			Total reads No.		
			Orig.	Corr.	Dec.	Orig.	Corr.	Inc.	Orig.	Corr.	Dec.	Orig.	Corr.	Inc.	Orig.	Corr.	Dec.	Orig.	Corr.	Dec.
D28	197071	2.70%	827	619	25.2%	16425	16859	2.6%	7158	4590	35.9%	1276397	1287399	0.9%	168	79	53.0%	704	606	13.9%
D29	236699	4.02%	485	331	31.8%	10867	11365	4.6%	5511	3489	36.7%	928151	940139	1.3%	116	54	53.4%	6220	6320	–1.6%
D30	324175	1.91%	851	632	25.7%	17653	17978	1.8%	5229	3427	34.5%	732858	739190	0.9%	144	73	49.3%	450	392	12.9%
D31	147763	2.17%	871	634	27.2%	24767	25191	1.7%	8134	5200	36.1%	1763443	1776084	0.7%	154	63	59.1%	396	266	32.8%
D32	268616	1.43%	688	508	26.2%	19761	20147	2.0%	5553	3512	36.8%	779496	786162	0.9%	101	56	44.6%	253	192	24.1%
D33	264327	2.24%	915	670	26.8%	29849	30344	1.7%	6788	4377	35.5%	1215062	1223420	0.7%	126	83	34.1%	362	318	12.2%
D34	288649	4.23%	449	309	31.2%	8735	9124	4.5%	4733	2959	37.5%	741500	751526	1.4%	119	51	57.1%	3909	3964	–1.4%
D35	144252	2.45%	821	594	27.6%	32558	33198	2.0%	7464	4855	35.0%	2598394	2612835	0.6%	185	84	54.6%	2639	2523	4.4%
D36	272107	3.40%	626	475	24.1%	14092	14517	3.0%	5367	3403	36.6%	938218	947607	1.0%	122	50	59.0%	284	183	35.6%
D37	207804	1.66%	547	407	25.6%	10752	10912	1.5%	5184	3347	35.4%	685935	691030	0.7%	87	51	41.4%	184	142	22.8%
AVE.	235146	2.62%	708	518	27.1%	18546	18964	2.5%	6112	3916	36.0%	1165945	1175539	0.9%	132	64	50.6%	1540	1491	15.6%

Note: Abbreviation: Corr., Correction; No., Number; PCT., Percentage; Ori., Original; Inc., Increase; Dec., Decrease.

Graphical Abstract