Galanti *et al. BMC Bioinformatics* (2021) 22:359
https://doi.org/10.1186/s12859-021-04267-5

BMC Bioinformatics

SOFTWARE
Open Access

Phenigs 2.0: accurate, high-performance Bayesian decoding and confidence estimation for combinatorial barcode indexing

Lior Galanti[1,2], Dennis Shasha[3] and Kristin C. Gunsalus[1,2*]

*Correspondence: kcg1@nyu.edu
[2] NYU Abu Dhabi Center for Genomics and System Biology, New York University, Abu Dhabi, United Arab Emirates
Full list of author information is available at the end of the article

Abstract

Background: Systems biology increasingly relies on deep sequencing with combinatorial index tags to associate biological sequences with their sample, cell, or molecule of origin. Accurate data interpretation depends on the ability to classify sequences based on correct decoding of these combinatorial barcodes. The probability of correct decoding is influenced by both sequence quality and the number and arrangement of barcodes. The rising complexity of experimental designs calls for a probability model that accounts for both sequencing errors and random noise, generalizes to multiple combinatorial tags, and can handle any barcoding scheme. The needs for reproducibility and community benchmark standards demand a peer-reviewed tool that preserves decoding quality scores and provides tunable control over classification confidence that balances precision and recall. Moreover, continuous improvements in sequencing throughput require a fast, parallelized and scalable implementation.

Results and discussion: We developed a flexible, robustly engineered software that performs probabilistic decoding and supports arbitrarily complex barcoding designs. Pheniqs computes the full posterior decoding error probability of observed barcodes by consulting basecalling quality scores and prior distributions, and reports sequences and confidence scores in Sequence Alignment/Map (SAM) fields. The product of posteriors for multiple independent barcodes provides an overall confidence score for each read. Pheniqs achieves greater accuracy than minimum edit distance or simple maximum likelihood estimation, and it scales linearly with core count to enable the classification of > 11 billion reads in 1 h 15 m using < 50 megabytes of memory. Pheniqs has been in production use for seven years in our genomics core facility.

Conclusion: We introduce a computationally efficient software that implements both probabilistic and minimum distance decoders and show that decoding barcodes using posterior probabilities is more accurate than available methods. Pheniqs allows fine-tuning of decoding sensitivity using intuitive confidence thresholds and is extensible with alternative decoders and new error models. Any arbitrary arrangement of barcodes is easily configured, enabling computation of combinatorial confidence scores for any barcoding strategy. An optimized multithreaded implementation assures that Pheniqs is faster and scales better with complex barcode sets than existing tools.

Support for POSIX streams and multiple sequencing formats enables easy integration with automated analysis pipelines.

Keywords: Sequence demultiplexing, Combinatorial indexing, Barcode decoding confidence, Single-cell split-pooling, Barcode noise filtering, Barcode simulation

Background

High-throughput next-gen bulk sequencing with multiplexed sample barcodes is now standard practice to increase throughput and reduce sequencing costs, and single-cell applications are rapidly proliferating and evolving [1]. The advent of barcodes to tag individual cells and molecules from which sequence reads originate both enhances resolution and helps control for quantification biases. Yet indeterminate reads that are discarded, and uneven barcode distributions that differ from expectation, are common and decrease both accuracy and sensitivity.

The fast pace of innovation in sequencing technologies and experimental designs continuously presents new challenges to the classification of sequence reads. Increasingly complex barcoding schemes are being devised to accommodate novel experimental designs, and combinatorial cellular indexing protocols involving several successive rounds of barcoding expand the potential space exponentially [2, 3]. Just over the horizon, multimodal profiling—the simultaneous measurement of gene expression, protein abundance, chromatin state, spatial transcriptomics, and/or CRISPR-based readouts (e.g. lineage tracing, genetic screens)—is poised to become a new industry standard [1]. Third-generation platforms overcome some of the limitations of short-read sequencing by providing very long reads, but current long-read sequencers operate with very high error rates of 15–40%. To compensate, these are often complemented with high-fidelity short-read sequences or self-corrected using many-fold coverage [4]. Multiplexing with barcoding is now emerging as a way to increase throughput and lower costs for these platforms as well.

Several tools are currently available for decoding and classifying barcodes, an essential task for demultiplexing pooled bulk sequence libraries and for assigning reads to individual cells in single-cell workflows. However, existing tools for decoding index tags do not provide flexible support for the rising complexity of novel barcoding schemes. As a result, numerous custom solutions to handle different barcoding schemes are being implemented downstream of standard demultiplexing software, often geared to a specific application and implemented using convenient scripting languages such as R or Python. Various integrated workflows focusing on scRNA-seq analysis now bundle identification of cellular barcodes and molecular barcodes (UMIs) with transcript quantification and other downstream analyses to identify and study different cell types, including CellRanger [5], Seurat [6], Salmon-Alevin [7], and BUStools [8]. Such workflows could benefit from preprocessing by an efficient and flexible decoder that natively supports any arbitrary experimental design and can be easily integrated into automated pipelines. As sequence applications diversify, decoupling barcode classification from downstream analysis tasks becomes increasingly desirable, both to simplify workflows and to encourage the development of community standards for benchmarking and best practices.

Sequencing platforms generally tie in demultiplexing as a preprocessing step that is not directly accessible to users. The most widely used first-generation tool for sample demultiplexing is Illumina's bcl2fastq, which combines basecalling with sample demultiplexing. bcl2fastq accepts input only in the proprietary Illumina BCL format and is specifically geared toward sequence data from Illumina platforms. It relies on a naïve decoding method based on exact string matching with up to one mismatch, in other words a simple minimum Hamming distance decoder, and quality assessment procedures that are not obviously accessible to end-users. bcl2fastq can also generate FASTQ files without demultiplexing, but most other tools that classify sequences using barcodes, such as Picard [9], also use simple minimum distance decoding.

Standalone, peer-reviewed, classifiers include deML [10], Bayexer [11], and Axe [12]. deML is capable of handling a single barcode set with either one or two segments. While it consults basecalling qualities and reports classification scores in Sequence Alignment/Map (SAM) [13] auxiliary tags, deML does not accurately reflect the probability of correct classification for two reasons: it estimates maximum likelihood only for the top candidates that are within a short Hamming distance, and it operates under restrictive assumptions that do not hold in most complex cellular indexing designs—i.e. that barcodes are uniformly distributed and that contaminating sequences are extremely rare. Bayexer attempts to train a naïve Bayes classifier by studying the error pattern when the insert sequence is shorter than the read length and thus provides a second observation of the barcode. Although this approach can potentially increase accuracy in those specific cases, it is not applicable to general barcode decoding and the tool fails to produce output when such redundancy is not present. Axe uses a set of pre-computed prefix trees to find a match within a given Hamming distance and can partially handle barcodes that differ in length. Although this method can be very fast, it ignores quality scores and is highly sensitive to upstream errors in the prefix, and so sacrifices accuracy.

None of the above tools account for the prior distribution of barcoded samples (which is often nonuniform) or explicitly account for noise (i.e. contaminating or indeterminate sequences), nor do they compute or report the posterior classification probability for individual reads. They also lack the ability to address arbitrary offsets within read segments, decode barcodes with more than two components, or handle multiple types or combinations of barcodes, and thus often require custom pre- or post-processing.

Pheniqs overcomes many of the limitations of other decoders by taking into account key aspects of current decoding requirements: scalability, assumptions about the number of tags and their location, integration of richer metadata, and the ability to explicitly account for basecalling quality scores, uneven barcode distributions, and presence of indeterminate sequences.

Implementation

Pheniqs (PHilology ENcoder wIth Quality Statistics, pronounced *phoenix*) combines a generic and extensible approach to barcode decoding with flexible configuration options that easily accommodate custom experimental designs. It implements both a standard minimum distance decoder (MDD) based on Hamming distance and a probabilistic decoder (PAMLD). PAMLD consults base-calling quality scores as well as priors to compute the full posterior decoding probability for classification, rather than a simple maximum likelihood estimate.

Decoding with the posterior probability

Classification based on barcodes involves extracting a subsequence r from an observed read, along with the basecall quality scores associated with the individual nucleotides in r, and decoding the original sequence s. Let $r \in \{A, C, G, T, N\}^n$ be an observed sequence of length n extracted from the read and $\mathcal{B} \subseteq \{A, C, G, T\}^n$ a given set of distinct barcodes where each $b \in \mathcal{B}$ identifies an individual class. A decoder is denoted as a decision function $\phi : \{A, C, G, T, N\}^n \mapsto \mathcal{B} \cup \varepsilon$, where ε is a decoding failure for an indeterminate sequence $s \notin \mathcal{B}$.

A maximum likelihood decoder will identify the barcode $\hat{b} \in \mathcal{B}$ which maximizes the posterior probability that $\hat{b}$ was sequenced given that r was observed.

$$\hat{b} = \underset{b \in \mathcal{B}}{\arg\max}\, P(b|r) \tag{1}$$

Applying Bayes' rule we can compute $P(b|r)$ using

$$P(b|r) = \frac{P(r|b)P(b)}{P(r|b \notin \mathcal{B})P_\varepsilon + \sum_{b' \in \mathcal{B}} P(r|b')P(b')} \tag{2}$$

where P_ε is the prior probability of encountering indeterminate sequences and $P(r|b \notin \mathcal{B})$ is the probability of observing a particular such sequence.

The *Phred-adjusted maximum likelihood decoder* (PAMLD) implemented by Pheniqs solves Eq. 2 by computing $P(r|b)$ for each $b \in \mathcal{B}$ from the basecalling quality scores [14]. $P(b)$, the expected fraction of reads identified by b, can be either provided *a priori* by the user or estimated directly from the data. In the absence of any prior information about potential sequence composition (base distribution or GC bias), we can only assume indeterminate sequences occur with maximum entropy so $P(r|b \notin \mathcal{B}) = 1/4^n$. Such reads may arise from spiked in controls used for instrument calibration, contamination during library preparation or other unknown factors such as defective sequencing kits. Realistically, however, not every sequence in $\{A, C, G, T\}^n$ is equally likely to appear in sequencing, indeterminate entropy is lower, and $P(r|b \notin \mathcal{B}) > 1/4^n$. Empirical studies can determine a more refined lower bound for $P(r|b \notin \mathcal{B})$. Pheniqs accommodates such refinements to the noise model by allowing advanced users to manually set this value.

When $P(r|\hat{b}) < P(r|b \notin \mathcal{B})$, the initial evidence supporting the classification provided by the conditional probability is inferior to that provided by a random sequence, indicating that the $\hat{b}$ recovered in Eq. 1 cannot be distinguished from noise. The *noise filter* considers those a decoding failure without further consideration.

Reads that pass the *noise filter* are evaluated by the *confidence filter*, which compares $P(\hat{b}|r)$ to a user-provided confidence threshold C for the minimum acceptable probability of a correct decoding and declares a failure if $P(\hat{b}|r) \leq C$. The probability of a decoding error is

$$P_{\text{decoding\_error}}(\hat{b}, r) = 1 - P(\hat{b}|r) \tag{3}$$

Directly estimating $P(\hat{b}|r)$ allows Pheniqs to report intuitive classification confidence scores for every read. Deriving a confidence score for a combinatorial barcode, made up of several independent components, requires to simply multiply the confidence scores of the individual components. The governing threshold C allows researchers to choose between assignment confidence and yield of classified reads and defaults to 0.95. The PAMLD decoding workflow is summarized in Fig. 1B.

By contrast, deML [10] assumes that P_ε is infinitesimally small and that samples are uniformly pooled (and thus equally likely), thereby suggesting that for every $b \in \mathcal{B}$

$$P(b) = \frac{1 - P_\varepsilon}{|\mathcal{B}|} \approx \frac{1}{|\mathcal{B}|} \tag{4}$$

Under such conditions $P(b|r) \propto P(r|b)$ and Eq. 1 can be simplified to

$$\hat{b} = \arg\max_{b \in \mathcal{B}} P(r|b) \tag{5}$$

While such assumptions simplify maximum likelihood estimation of $\hat{b}$, they are often grossly imprecise. For example, the relatively low yield in (e.g. single-cell) experiments that rely on several layers of combinatorial indexing often results in an extremely uneven barcode distribution, with P_ε representing a significant portion of the sequenced DNA. Furthermore, implementations that refrain from computing $P(\hat{b}|r)$ cannot report the posterior classification probability or a confidence score for combinatorial barcodes.

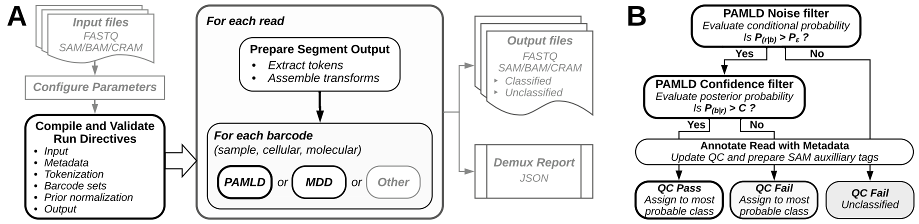


Fig. 1 Pheniqs decoding pipeline architecture. **A** Pheniqs requires as input sequence data files in any standard format and (if not using default parameters) a JSON configuration file. Python API tools (not shown) assist with IO management and can automatically generate an initial configuration using metadata from an Illumina run directory. Pheniqs evaluates each configuration component to determine how the data should be processed and to ensure that all required directives are present and properly specified. Any validation failures trigger clear, explicit error messages. Prior distributions of expected barcodes either derive from initial sample proportions as given (e.g. per a sample sheet), or are estimated directly from the data during a preliminary PAMLD decoding run. Barcode *tokens* are extracted from read segments using transform directives and then passed to a decoder (PAMLD or MDD). New decoding algorithms may be implemented as derived classes. Decoded barcodes and quality scores are written to specific SAM auxiliary field for each barcode type. Pheniqs can emit FASTQ files split by sample barcode, but SAM format is preferred since it preserves all associated metadata, and binary (BAM) and compressed (CRAM) versions produce considerably smaller files. POSIX integration allows direct piping to automated workflows, and support for real-time translation of file formats enables teeing to multiple outputs (thus avoiding the need to write temporary files). A JSON-encoded run report is also generated that provides summary statistics for the analysis. **B** PAMLD noise and confidence filters. Reads with a lower conditional probability than random sequences fail the *noise filter* and are classified as noise without further consideration. Reads with a posterior probability that does not meet the confidence threshold fail the *confidence filter*; these reads are classified, but they are marked as *qc fail* so the confidence threshold can be reconsidered at a later stage

Estimating the prior distribution

Statistics from a preliminary PAMLD decoding run can be used to estimate the relative proportions of the individual barcodes $P(b)$ for each $b \in \mathcal{B}$ and the noise P_ε from the data. The *high confidence estimator* bundled with Pheniqs estimates the relative proportions from the *high confidence* reads alone, assuming that *low confidence* reads (those that passed the *noise filter* but failed the *confidence filter*) and *high confidence* reads (those that passed both filters) come from the same distribution.

Let S_ε be the number of reads rejected by the *noise filter*, S_b the number of reads classified to b with confidence higher than C, and $S_\mathcal{B} = \sum_{b \in \mathcal{B}} S_b$. A *high confidence estimator* for the noise prior is

$$\hat{P}_\varepsilon = \frac{S_\varepsilon}{S_\varepsilon + S_\mathcal{B}} \tag{6}$$

and for an individual barcode is

$$\hat{P}(b) = \frac{S_b}{S_\varepsilon + S_\mathcal{B}} \tag{7}$$

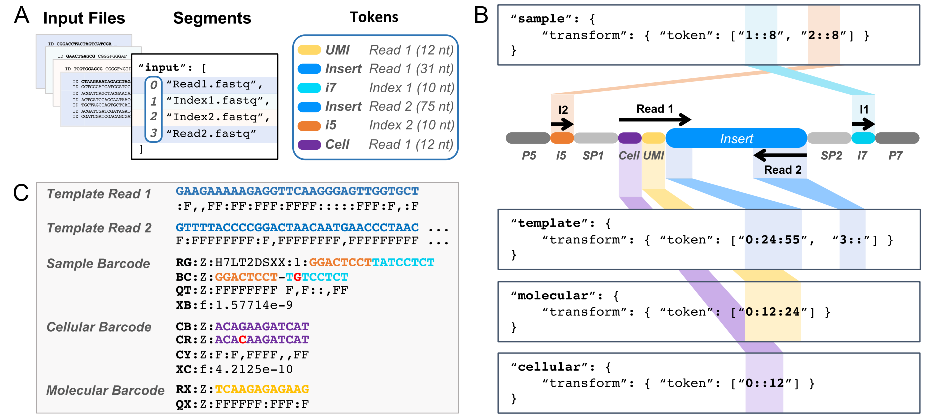
The *high confidence estimator* is a general purpose estimator, but Pheniqs can be used with any prior estimates devised by the user.

Software architecture

Pheniqs is distributed as a compiled binary with a command-line interface that accepts a JSON encoded configuration file (Fig. 1A). Stable releases are also packaged in BioConda. To accommodate a rapidly changing landscape, Pheniqs is designed to be easily extensible with alternative error models and additional decoder implementations. An efficient codebase with robust input validation and comprehensive documentation supports individual installations and large-scale production facilities alike.

Using a familiar syntax that mimics Python array slicing, Pheniqs can decode multiple barcodes located anywhere in any sequence read. It extracts tokens from multiple read segments by addressing either the 5' end, 3' end, or both (and optionally reverse complemented) to construct the output template segments and the sample, cellular and molecular barcodes (Fig. 2). For convenience, Pheniqs provides reusable definitions for several standard sample barcode sets that can be imported into any configuration file. Additional types of barcodes can easily be defined by declaring them in the configuration file (e.g. barcodes for split-pooling, antibody tags, spatial sequencing, etc.). This generic approach allows for arbitrary manipulation of sequence tokens that accommodates any potential barcoding scheme and obviates the need for pre- and post-processing for most experimental designs.

By directly interfacing with the low level HTSlib [15] C API Pheniqs can read, write, and manipulate either uncompressed or gzip compressed FASTQ files as well as the SAM file format or its binary compressed variants BAM and CRAM. Unlike FASTQ, the SAM format can encode sequencing data in a single, smaller file that supports richer metadata annotations. To allow multithreaded performance to scale linearly with core count, Pheniqs carefully synchronizes the many threads that read, decode and write with a consumer/producer model [16]. This allows threads that compute the posterior



```
"input": [
  0 "Read1.fastq",
  1 "Index1.fastq",
  2 "Index2.fastq",
  3 "Read2.fastq"
]
```

```
"sample": {
    "transform": { "token": ["1::8", "2::8"] }
}
```

```
"template": {
    "transform": { "token": ["0:24:55",  "3::"] }
}
```

```
"molecular": {
    "transform": { "token": ["0:12:24"] }
}
```

```
"cellular": {
    "transform": { "token": ["0::12"] }
}
```

```
Template Read 1     GAAGAAAAAGAGGTTCAAGGGAGTTGGTGCT
                    :F,,FF:FF:FFF:FFFF::::FFF:F,:F
Template Read 2     GTTTTACCCCGGACTAACAATGAACCCTAAC ...
                    F:FFFFFFFF:F,FFFFFFFF,FFFFFFFFF ...
Sample Barcode      RG:Z:H7LT2DSXX:1:GGACTCCTTATCCTCT
                    BC:Z:GGACTCCT-TGTCCTCT
                    QT:Z:FFFFFFFF F,F::,FF
                    XB:f:1.57714e-9
Cellular Barcode    CB:Z:ACAGAAGATCAT
                    CR:Z:ACACAAGATCAT
                    CY:Z:F:F,FFFF,,FF
                    XC:f:4.2125e-10
Molecular Barcode   RX:Z:TCAAGAGAGAAG
                    QX:Z:FFFFFF:FFF:F
```

Fig. 2 Tokenizing a standard Illumina read. Example of tokenization syntax for a 150-nt paired-end dual-indexed sequencing run. **A** Input files containing read *segments* emitted by the sequencer are indexed as an array, where 0 = Read1, 1 = Index1, 2 = Index2, 3 = Read2. Barcode *tokens* are defined for each type of barcode included in the experimental design and may appear at any position and orientation in any read segment. **B** This example contains a *sample* barcode composed of two 8nt elements (i5 and i7), a 12nt inline *cellular* barcode (Cell), and a 12nt inline *molecular* barcode (UMI). The biological sequences of interest (*template*) are located in Read 1 (31nt just downstream of the Cell and UMI) and all of Read 2 (here, 75nt). Each *token* comprises three colon separated components, "segment:start:end". Per Python array slicing syntax, the *start* coordinate (offset) is inclusive and the *end* coordinate is exclusive. Start and end coordinates default to 0 and the end of the segment, respectively. **C** Template read segments, observed and most likely inferred barcode sequences, quality scores, and error probabilities are emitted to designated SAM fields. The reported error for each barcode is one minus its estimated confidence (the posterior decoding probability); for a combinatorial barcode, the reported error is one minus the product of the confidence scores for each component

probabilities to work without waiting for threads that read and write. Each input feed uses two independent memory buffers: one for accepting incoming reads from the input file and one for supplying reads to the barcode decoding threads. When the first is full and the second is empty, a special thread momentarily locks both buffers and switches between them so that all operations may proceed with no interruption. The same principle is mirrored for output files. Since buffers may only be modified by one operation at a time to prevent data corruption, this design allows Pheniqs to concurrently receive input from multiple files, decode barcodes with multiple threads, and write output to multiple files, all with optimal efficiency. When integrated into a pipeline, Pheniqs can take advantage of POSIX standard streams to avoid the speed and storage bottlenecks associated with reading and writing temporary files.

Pheniqs reports the decoded sample, cellular, and molecular barcodes as well as their corresponding quality scores and the posterior decoding error probability in SAM auxiliary fields. It can associate standard SAM read groups with sample barcodes and can optionally perform an exhaustive quality assessment during processing that it includes in the final report.

Results and discussion

We used a variety of performance metrics to evaluate the decoding accuracy and computational efficiency of Phred-aware maximum likelihood decoding with Pheniqs (PAMLD), simple maximum likelihood estimation with deML, and minimum distance decoding (MDD, as implemented by Pheniqs). We used a semi-synthetic short-read dataset with a known true barcode set to measure accuracy across a range of error rates, and data from one lane of an Illumina NovaSeq run (containing >11 billion reads) to compare run time and memory usage.

Accuracy

Decoding accuracy was analyzed using semi-synthetic barcoded sequence reads generated from the run published with deML [10]. To establish a ground truth for testing purposes, we simulated barcoded sequence reads by replacing the barcode nucleotides in each read with a perfect barcode sequence sampled from a known prior distribution. To simulate noise, we replaced a barcode with a sequence from a random offset in the genome of *PhiX174*, a 5386nt DNA bacteriophage with 45% GC content [17]). We used *PhiX174* sequences because Illumina sequencers require balanced and random nucleotide composition for instrument calibration and quality control, and *PhiX174* DNA is often spiked in as a control for this purpose at concentrations of 1–5% or up to 40% for low-complexity libraries. *PhiX174* reads do not carry barcodes and should not be classified but instead labeled as Undetermined during demultiplexing. However, *PhiX174* sequences can sometimes contribute to noise contamination because they resemble an expected barcode by chance and must be removed in a downstream step by read sequence alignment. Notably, a recent study has found that 1000 genomes in the Integrated Microbial Genomes Database are contaminated with *PhiX174* sequences [18], suggesting that these are a common source of noise in Illumina sequence data.

We simulated sequencing errors by introducing a substitution at a nucleotide according to the basecall quality score and substitution frequencies made available with LRSim [19]. For this analysis we simulated substitution errors only since current short read platforms generate indel errors at a much lower rate than substitutions [20]. Finally, to simulate reads with different overall error rates, we recalibrated the quality scores produced in the first step and then simulated substitution errors on the recalibrated data. Additional file 1: Figure S1 shows the calibrated quality score distributions of each simulated dataset. The modular architecture of Pheniqs allows for the addition of alternative error models that may be better suited for sequencing platforms with different error characteristics and could be tested similarly.

We used the above datasets to evaluate classification accuracy across a range of error rates for Pheniqs MDD with default settings (MDD); deML with default settings (deML); and Pheniqs PAMLD with default settings (PAMLD uniform), true priors (PAMLD true), and *high confidence* estimated priors (PAMLD estimated). Pheniqs can compute estimated priors in a preliminary run by updating an initial set of priors using observed barcode frequencies (Fig. 3A shows the true sample barcode distribution in the dataset). *High confidence* estimated priors were computed here using statistics from a preliminary PAMLD run with the default 0.95 confidence threshold, 0.05 noise, and uniform barcode priors as input. We evaluated each barcode and the noise class as a binary classifier, so

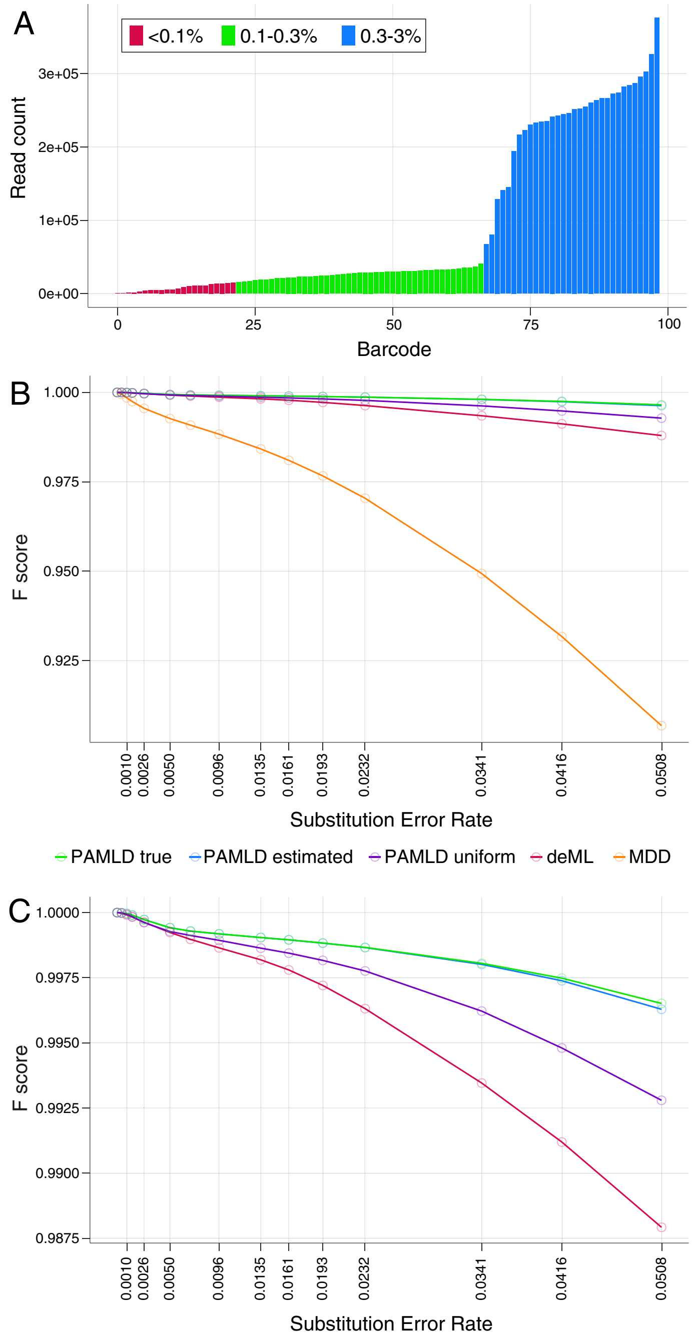


Fig. 3 Decoding F-score. **A** True frequency distribution of individual barcodes in the synthetic data. The *control* barcode (the only one > 3%, comprising 32% of the data used for evaluation) and *noise* (9%) were omitted for visual clarity. **B** All probabilistic decoders outperform minimum distance decoding by a wide margin, with differences becoming more pronounced as the error rate increases. **C** Among the probabilistic decoders, all configurations of PAMLD are more accurate than deML. The *high confidence* estimated prior performs nearly as well as the true prior

a correct assignment was counted as a true positive (*TP*), while an incorrect assignment was counted as a false negative (*FN*) for the correct class and as a false positive (*FP*) for the incorrectly assigned class. Reads marked as failing quality control (Fig. 1B) were classified to the noise class for this analysis. We then summed up the values from all classes and computed the *false discovery rate* (FDR), *miss rate* (MR) and *F-score* (harmonic mean of the precision and recall).

The resulting analysis showed that the probabilistic decoders consistently outperform MDD, which has very little resilience to errors and noise (Fig. 3B). Differences in accuracy become more pronounced as the substitution rate increases: at a rate of 0.05, the F-score for MDD is nearly 10% lower than other decoders. PAMLD with uniform priors is more accurate than deML, and *high confidence* prior estimation provides further gains that closely approximate performance with the true prior (Fig. 3C). Thus, computing the full posterior probability outperforms simple maximum likelihood estimation, even under the same assumption of a uniform prior, and estimating the true prior distribution approaches near-optimal decoding.

The effect of the *noise filter* is illustrated in Fig. 4, which shows the error in *high confidence* prior estimation for true barcode classes binned by their relative abundance across a range of substitution error rates. The difference between true and estimated priors is negligible at low overall error rates, but as basecalling quality decreases it becomes difficult to distinguish true barcodes from noise since $P(r|b)$ for a true barcode is more likely to fail the *noise filter*. As a result, individual barcode classes are increasingly underestimated and $\hat{P}_\varepsilon$ is correspondingly overestimated.

To examine the sources of performance gains at a more fine-grained level, we plotted FDR, MR and *F-score* statistics separately for *classified*, *classifiable*, and *unclassified* reads across a range of error rates. *Classified* reads represent true barcodes or noise reads that were classified to a real barcode. These include correct assignments (TP), noise or barcode reads assigned to the wrong barcode class (FP), and true barcodes not assigned to their proper class (FN). While all decoders perform reasonably well on *classified* reads overall (Additional file 1: Figure S2), Pheniqs with true or

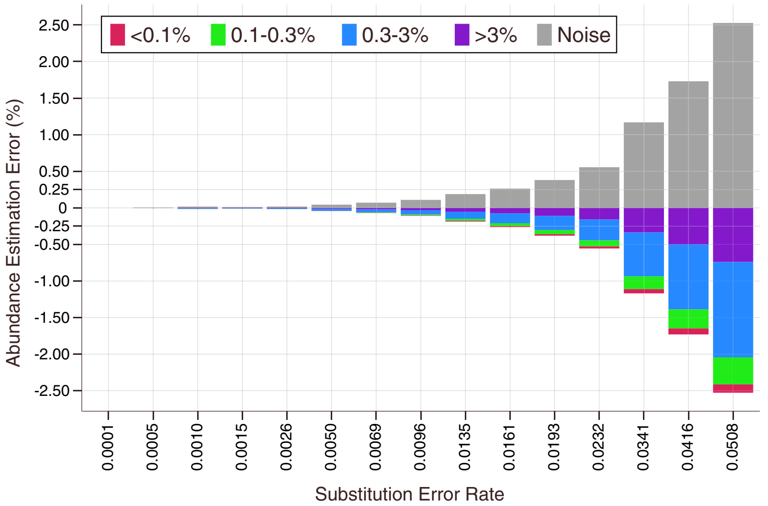


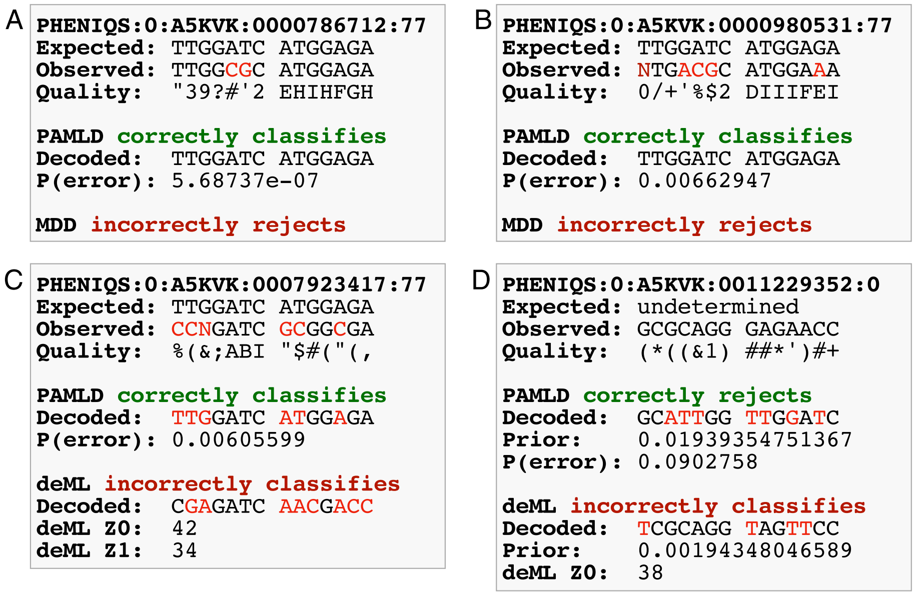
Fig. 4 Error in high confidence prior estimation for true barcodes at different overall error rates. Results binned by relative barcode abundance reveals the tradeoff between noise and correct classification. As basecalling quality deteriorates, distinguishing true barcodes from noise becomes increasingly difficult, resulting in underestimation of the true barcode priors and overestimation of the noise prior $\hat{P}_\varepsilon$

estimated priors consistently results in a lower rate of misclassified reads (lower FDR) and recovers more reads (lower MR) than other decoders. Filtering noise helps Pheniqs outperform deML in any configuration, primarily by reducing FDR (Additional file 1: Figure S2A). In the lower error range relevant to most Illumina runs (∼ 1/1000), PAMLD shows a lower FDR than even MDD (Additional file 1: Figure S2B). The FDR is flat for MDD because it ignores any reads with more than one mismatch, and for the same reason the miss rate climbs dramatically relative to the other decoders as the error rate increases.

Since datasets now contain increasingly large numbers of multiplexed samples or individual cells, the ability to accurately identify rare barcode classes is of high interest. We used read counts for each barcode class (Fig. 3A) to examine performance for *classified* reads binned by their relative abundance: *very low* (< 0.1% of reads), *low* (0.1–0.3%), *similar to uniform* (0.3–3%) and *overrepresented* (> 3%; here a single control barcode accounted for 32% of all reads). Across the entire spectrum, PAMLD with estimated or true priors shows better overall performance (F-score) than either deML or PAMLD with uniform priors (Additional file 1: Figure S3). For more abundant barcode classes, gains are mainly due to higher sensitivity (lower MR). For lower abundance barcodes (< 0.3% of total reads), PAMLD makes ∼ 10-fold fewer incorrect assignments than deML (lower FDR) with only a modest loss of sensitivity. Thus PAMLD is especially beneficial for the detection of rare barcodes.

Indeterminate barcodes are not uncommon in short-read datasets and can greatly reduce the yield of usable data. We found that PAMLD improves accuracy for both *classifiable* reads (true barcode reads, correctly classified or not) and *unclassified* reads (reads that are noise or fail quality control). For *classifiable* reads, both FDR and MR are consistently lower, leading to improved F-scores (Additional file 1: Figure S4; examples in Fig. 5A–C). For *unclassified* reads, probabilistic decoders have a lower FDR (Additional file 1: Figure S5) and also classify many fewer true barcodes as noise than MDD (Additional file 1: Figure S5; examples shown in Fig. 5A, B, D). The PAMLD *noise filter* rejects reads when the evidence for the barcode with the highest posterior is no better that for a random sequence (Fig. 1B). This is arguably a desired property since such reads have very weak evidence for classification. When the overall error rate is low, PAMLD errs on the side of caution and classifies slightly more true barcodes as noise than deML (FDR, Additional file 1: Figure S5A). On the other hand, PAMLD misses many fewer true noise reads (lower MR) than deML (Additional file 1: Figure S5A; example shown in Fig. 5D), and in the range of error rates for Illumina sequencers filters true noise better than even MDD (lower MR, Additional file 1: Figure S5B).

In summary, we find that using prior knowledge of barcode frequency and sequence quality statistics, combined with noise filtering, increases the recovery of true barcodes and reduces the number of misclassified reads. These benefits are most pronounced for barcodes present at low frequency and for data with high error rates. Thus probabilistic, quality-aware decoding offers distinct advantages for classifying short-read datasets that are highly complex and/or use combinatorial barcoding strategies. Pheniqs may also prove advantageous for third-generation single-molecule sequencing platforms that provide long sequence reads at much higher error rates.



A

```
PHENIQS:0:A5KVK:0000786712:77
Expected: TTGGATC ATGGAGA
Observed: TTGGCGC ATGGAGA
Quality:  "39?#'2 EHIHFGH

PAMLD correctly classifies
Decoded:  TTGGATC ATGGAGA
P(error): 5.68737e-07

MDD incorrectly rejects
```

B

```
PHENIQS:0:A5KVK:0000980531:77
Expected: TTGGATC ATGGAGA
Observed: NTGACGC ATGGAAA
Quality:  0/+'%$2 DIIIFEI

PAMLD correctly classifies
Decoded:  TTGGATC ATGGAGA
P(error): 0.00662947

MDD incorrectly rejects
```

C

```
PHENIQS:0:A5KVK:0007923417:77
Expected: TTGGATC ATGGAGA
Observed: CCNGATC GCGGCGA
Quality:  %(&;ABI "$#("(,

PAMLD correctly classifies
Decoded:  TTGGATC ATGGAGA
P(error): 0.00605599

deML incorrectly classifies
Decoded:  CGAGATC AACGACC
deML Z0:  42
deML Z1:  34
```

D

```
PHENIQS:0:A5KVK:0011229352:0
Expected: undetermined
Observed: GCGCAGG GAGAACC
Quality:  (*((&1) ##*')#+

PAMLD correctly rejects
Decoded:  GCATTGG TTGGATC
Prior:    0.01939354751367
P(error): 0.0902758

deML incorrectly classifies
Decoded:  TCGCAGG TAGTTCC
Prior:    0.00194348046589
deML Z0:  38
```

Fig. 5 Decoding examples. **A** The barcode on this read has only two mismatches (highlighted in red in *Observed*). MDD rejects it and assigns it to the undetermined bin. PAMLD easily classifies it with a confidence of more than 99.9999%. This is a good example of why MDD is overly conservative and has such a high rate of false negatives. **B** This example has 4 mismatches and one uncalled nucleotide (highlighted in red in *Observed*). MDD rejects it while PAMLD still classifies it with relatively good confidence of 99.337%. **C** In this example there are 5 mismatches and one uncalled nucleotide (highlighted in red in *Observed*). PAMLD classifies it correctly with a good confidence of 99.3944%, while deML classifies the read to the wrong bin. Mismatched nucleotides in the *Observed* sequence are marked in red. **D** In this example the read was noise and should have been rejected and assigned to the undetermined bin. PAMLD initially classifies it to a barcode with 6 mismatches, but it rejects the classification because the error probability is above 9%. deML incorrectly classifies the read. PAMLD's decision is influenced by the prior for the barcode it chooses, which is about 10 times higher than the barcode chosen by deML

Runtime speed and memory

To evaluate speed and memory usage we used a single lane from an Illumina NovaSeq, the highest throughput instrument available today, containing 94 multiplexed libraries with standard dual i7 and i5 barcodes. 151 cycles were sequenced on each of the two template segments and 8 cycles on each of the i7 and i5 index segments.

Benchmarking for speed (Fig. 6A) and memory (Fig. 6B) was executed on an Intel Xeon CPU E5-2690 v4 @ 2.60 GHz with 14 cores and 28 threads. Basecalling the lane with bcl2fastq (without demultiplexing) took 47 min and yielded 11,578,868,372 dual-indexed paired-end reads that passed quality filtering. bcl2fastq produces reads with segments split over four gzip compressed FASTQ files that were used as input to Pheniqs. Pheniqs runtime benchmarks show that output format encoding greatly impacts speed. With null output Pheniqs decodes the barcodes and collects statistics (for instance to be used for prior estimation) but does not write any output. The null benchmark completed in less than half the time of all others, demonstrating that writing output files is the rate-limiting factor for performance. Pheniqs performs fastest when producing interleaved BAM output and demultiplexed the NovaSeq run in 3 h and 10 min. Even computing the full posterior probability, Pheniqs scales almost linearly with the number of available cores and is at least 20 times faster than deML, which lacks support for multithreading.

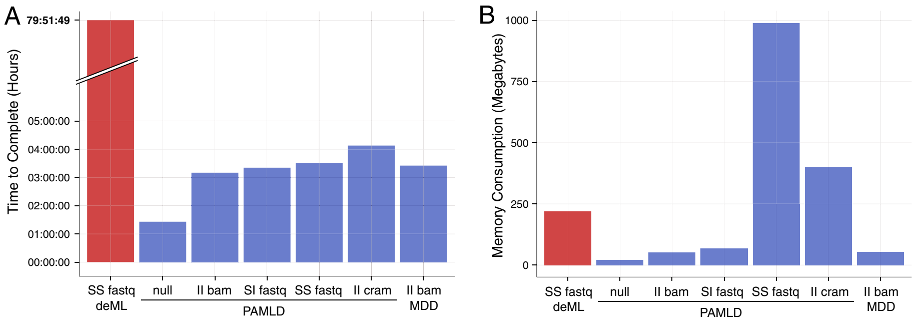


Fig. 6 Speed and memory benchmarks. One lane of an Illumina NovaSeq run was analyzed. The two letter codes describe the output layout: S for *split* and I for *interleaved*, first for library and second for segment. **A** Pheniqs with PAMLD generated interleaved BAM output in 3:10 h. CRAM is much more complex than BAM and employs several compression algorithms that are more aggressive than gzip to produce smaller files, so writing CRAM output took slightly longer as expected. Pheniqs scales almost linearly with the number of available cores and is at least 20 times faster than deML, which took almost 80 h to complete. **B** Pheniqs memory consumption increases linearly with the number of output files, since each is allocated separate IO buffers, but even with 200 output files and generous buffers Pheniqs still used less than 1 GB of memory

deML could only produce FASTQ output with FASTQ input, so we could not test BAM output.

Future work

We can envision a variety of ways to enhance the performance of Pheniqs. Our initial focus will be to facilitate integration into standard analysis workflows. To assist with configuration, we are building a library of barcode configuration templates for common sequencing kits and experimental designs. The Pheniqs website currently includes vignettes for configuring standard Illumina sequencing runs, single-cell RNA sequencing using the Fluidigm platform, and two plate-based split-pooling methods for single-cell RNA-seq: sciRNA-seq [2] and SPLiT-seq [3]. We also plan to add vignettes for other types of single-cell profiling, as well as multimodal profiling, spatial transcriptomics, and other novel sequencing applications as they arise.

One limitation of Bayesian decoding is that computing the posterior probability requires knowing the list of expected barcodes in advance. To overcome this problem, the same strategy used by Pheniqs to estimate priors for a known set of barcodes could be extended to an unknown set of barcodes. First, a preliminary run configured with a whitelist (e.g. a list of all known cellular barcodes in a single-cell sequencing kit) can be used to estimate the relative abundances of observed barcodes. A shorter list of barcodes can then be extracted from the preliminary run report by placing a threshold on the abundance of observed barcodes, which may then be decoded in a second run using the imputed priors. Providing decoding quality scores for whitelisted cellular barcodes based on the full posterior probability will allow improved estimation of their relative abundance, thus increasing accuracy and sensitivity for single-cell applications. An overall classification quality score combining multiple types of barcodes can then be computed, which will be easy to both report and understand. Since the complexity of computing the posterior is linearly correlated with the size of the barcode set, however, computing priors for a very large whitelist may become impractically slow. To address this, we are considering alternative strategies to reduce the complexity of the problem in such cases.

Another issue that Pheniqs currently does not address is error correction for UMIs (unique molecular identifiers)-barcodes that tag individual molecules in a library, which are used to identify PCR duplicates and to quantify the abundance of distinct molecular species. Since reads classified to a single combination of sample, cellular and molecular barcode are assumed to be a clone of the same biological sequence, the barcodes are not independent, making computing the full posterior probability unfeasible. To incorporate error correction for UMIs, we are looking into heuristic algorithms that rely on established peer-reviewed methods.

We are also interested in applying Pheniqs to data from other sequencing platforms. Performance evaluations of PAMLD on synthetic data show that gains in decoding accuracy become more pronounced as the rate of substitution errors increases. This suggests that Pheniqs could greatly enhance the analysis of sequence data with high error rates, such as third generation single-molecule sequencing platforms, which currently operate with error rates in the range of several percent [21–23]. In order to apply PAMLD to PacBio and Oxford Nanopore sequencing, we will need to incorporate support for insertions/deletion errors, which are more common on these platforms (particularly at homopolymers). We thus plan to evaluate performance and to investigate alternative noise models for these sequencing technologies.

Finally, while barcode decoding quality can have profound impacts on data quality, this issue has not so far received widespread attention. To address this, we advocate for working toward common benchmarks for barcode decoding quality, which would be tremendously useful for the development of community standards for evaluation and reporting conventions.

Conclusion

We show that barcode classification using full posterior probabilities with noise filtering is more accurate than other available methods. We provide a multithreaded software package with comprehensive input validation, Pheniqs, that is faster and more scalable than existing tools. The probability model implemented by Pheniqs accounts for both erroneous codewords and non-codeword random noise, can handle arbitrarily complex barcoding designs, and generalizes to multiple combinatorial tags. It relies on intuitive confidence thresholds for fine-tuning decoding accuracy and reports decoding confidence scores and barcode sequences for individual reads by populating standardized SAM format auxiliary fields. Pheniqs is designed for integration into automated analysis workflows and can be extended with new error models and alternative decoders. An efficient implementation, coupled with comprehensive documentation and a suite of helper tools, supports both individual users and core facilities alike. These combined features offer a new level of efficiency and flexibility for barcode decoding that is widely applicable to current experimental designs and is easily adaptable to the rapidly evolving landscape of sequencing applications.

Availability and requirements

- Project name: Pheniqs
- Project home page: http://biosails.github.io/pheniqs
- Bioconda package: https://anaconda.org/bioconda/pheniqs
- Operating systems: Linux, macOS.
- Programming language: C++11, Python 3.
- Other requirements: clang, gcc 4.8 or newer.
- License: NYU non-commercial research license, free for academic use.
- Commercial use: License required.

Abbreviations

SAM: Sequence alignment/map format; MDD: Minimum distance decoding; PAMLD: Phred-adjusted maximum likelihood decoding; UMI: Unique molecular identifier. Also referred to a molecular barcode; TP: True positive; FP: False positive; FN: False negative; FDR: False discovery rate $\left(\frac{FP}{TP+FP}\right)$; MR: Miss rate $\left(\frac{FN}{TP+FN}\right)$; F-score: Harmonic mean of precision and recall; nt: Nucleotide; IO: Input/output.

Supplementary Information

The online version supplementary material available at https://doi.org/10.1186/s12859-021-04267-5.

Additional file 1. Supplementary figures with legends.

Acknowledgements

We are grateful to Kapil Thadani and Or Biran for discussions on the statistical evaluation and mathematical notation; to Alan Twaddle for his feedback on the user manual and his help with benchmark analysis and figure preparation; to Jillian Rowe for her assistance with Conda packaging, and continuous integration; to Nizar Drou for constructive input and assistance with benchmarking; to Mohammed Khalfan for feedback on Pheniqs version 1 and for helpful discussions on the documentation; and to Giuseppe Saldi for his insightful comments.

Authors' contributions

LG conceived the project, developed the algorithms, wrote the code, and co-authored the manuscript. DS inspired the probabilistic likelihood model, reviewed the mathematical formulation, and provided feedback on the manuscript. KCG supervised the project, motivated the methodology for combinatorial barcode decoding, and co-authored the manuscript with LG. All authors read and approved the final manuscript.

Funding

This work was supported by a grant from the New York University Abu Dhabi (NYUAD) Research Institute to the NYUAD Center for Genomics and Systems Biology (ADHPG-CGSB) and by other research funding from NYUAD to KCG.

Availability of data and materials

Synthetic data was generated using the method described in the text with data made publicly available with deML. deML: robust demultiplexing of Illumina sequences using a likelihood-based approach. 2015. http://dx.doi.org/10.1093/bioinformatics/btu719 The BAM file is available at https://bioinf.eva.mpg.de/deml.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Department of Biology, Center for Genomics and System Biology, New York University, New York, USA. [2]NYU Abu Dhabi Center for Genomics and System Biology, New York University, Abu Dhabi, United Arab Emirates. [3]Department of Computer Science, Courant Institute, New York University, New York, USA.

Received: 4 March 2021 Accepted: 14 June 2021
Published online: 02 July 2021

References

1. Stuart T, Satija R. Integrative single-cell analysis. Nat Rev Genet. 2019;20(5):257–72. https://doi.org/10.1093/bioinformatics/btp324.
2. Cao J, Packer JS, Ramani V, Cusanovich DA, Huynh C, Daza R, Qiu X, Lee C, Furlan SN, Steemers FJ, Adey A, Waterston RH, Trapnell C, Shendure J. Comprehensive single-cell transcriptional profiling of a multicellular organism. Science. 2017;357(6352):661–7. https://doi.org/10.1126/science.aam8940.
3. Rosenberg AB, Roco CM, Muscat RA, Kuchina A, Sample P, Yao Z, Graybuck LT, Peeler DJ, Mukherjee S, Chen W, Pun SH, Sellers DL, Tasic B, Seelig G. Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. Science. 2018;360(6385):176–82. https://doi.org/10.1126/science.aam8999.
4. Fu S, Wang A, Au KF. A comparative evaluation of hybrid error correction methods for error-prone long reads. Genome Biol. 2019;20(1):26. https://doi.org/10.1186/s13059-018-1605-z.
5. CellRanger. 10X Genomics (2019)
6. Stuart T, Butler A, Hoffman P, Hafemeister C, Papalexi E, Mauck WM, Hao Y, Stoeckius M, Smibert P, Satija R. Comprehensive integration of single-cell data. Cell. 2019;177(7):1888–190221. https://doi.org/10.1016/j.cell.2019.05.031.
7. Srivastava A, Malik L, Smith T, Sudbery I, Patro R. Alevin efficiently estimates accurate gene abundances from dscrna-seq data. Genome Biol. 2019;20(1):65.
8. Melsted P, Ntranos V, Pachter L. The barcode, UMI, set format and BUStools. Bioinformatics. 2019;35(21):4472–3. https://doi.org/10.1093/bioinformatics/btz279.
9. Picard toolkit. Broad Institute (2019)
10. Renaud G, Stenzel U, Maricic T, Wiebe V, Kelso J. deml: robust demultiplexing of illumina sequences using a likelihood-based approach. Bioinformatics. 2015;31(5):770–2. https://doi.org/10.1093/bioinformatics/btu719.
11. Yi H, Li Z, Li T, Zhao J. Bayexer: an accurate and fast bayesian demultiplexer for illumina sequences. Bioinformatics. 2015;31(24):4000–2. https://doi.org/10.1093/bioinformatics/btv501.
12. Murray KD, Borevitz JO. Axe: rapid, competitive sequence read demultiplexing using a trie. Bioinformatics. 2018. https://doi.org/10.1093/bioinformatics/bty432.
13. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, Subgroup GPDP. The sequence alignment/map format and samtools. Bioinformatics. 2009;25(16):2078–9. https://doi.org/10.1093/bioinformatics/btp352.
14. Edgar RC, Flyvbjerg H. Error filtering, pair assembly and error correction for next-generation sequencing reads. Bioinformatics. 2015;31(21):3476–82. https://doi.org/10.1093/bioinformatics/btv401.
15. Bonfield JK, Marshall J, Danecek P, Li H, Ohan V, Whitwham A, Keane T, Davies RM. HTSlib: C library for reading/writing high-throughput sequencing data. GigaScience. 2021. https://doi.org/10.1093/gigascience/giab007.
16. Dijkstra EW. Information streams sharing a finite buffer. Inf Process Lett. 1972;1(5):179–80. https://doi.org/10.1016/0020-0190(72)90034-8.
17. Sanger F, Air GM, Barrell BG, Brown NL, Coulson AR, Fiddes JC, Hutchison CA, Slocombe PM, Smith M. Nucleotide sequence of bacteriophage phix174 dna. Nature. 1977;265:687–95. https://doi.org/10.1038/265687a0.
18. Mukherjee S, Huntemann M, Ivanova N, Kyrpides NC, Pati A. Large-scale contamination of microbial isolate genomes by Illumina phix control. Stand Genomic Sci. 2015;10:18. https://doi.org/10.1186/1944-3277-10-18.
19. Luo R, Sedlazeck FJ, Darby CA, Kelly SM, Schatz MC. Lrsim: a linked-reads simulator generating insights for better genome partitioning. Comput Struct Biotechnol J. 2017;15:478–84. https://doi.org/10.1016/j.csbj.2017.10.002.
20. Franziska Pfeiffer CG. Systematic evaluation of error rates and causes in short samples in next-generation sequencing. Sci Rep. 2018;8(1):10950.
21. Workman RE, Tang AD, Tang PS, Jain M, Tyson JR, Razaghi R, Zuzarte PC, Gilpatrick T, Payne A, Quick J, Sadowski N, Holmes N, de Jesus JG, Jones KL, Soulette CM, Snutch TP, Loman N, Paten B, Loose M, Simpson JT, Olsen HE, Brooks AN, Akeson M, Timp W. Nanopore native rna sequencing of a human poly(a) transcriptome. Nat Methods. 2019;16(12):1297–305. https://doi.org/10.1038/s41592-019-0617-2.
22. Dohm JC, Peters P, Stralis-Pavese N, Himmelbauer H. Benchmarking of long-read correction methods. NAR Genomics Bioinform. 2020. https://doi.org/10.1093/nargab/lqaa037.
23. Sahlin K, Medvedev P. Error correction enables use of oxford nanopore technology for reference-free transcriptome analysis. Nat Commun. 2021;12(1):2. https://doi.org/10.1038/s41467-020-20340-8.

Publisher's Note