Bioinformatics, 2026, **42**(4), btag182
https://doi.org/10.1093/bioinformatics/btag182
Published: 10 April 2026
Applications Note | Gene expression

OXFORD

Harvesting more reads from single-cell combinatorial barcoding data with *scarecrow*

David Wragg[1], Eunchai Kang[1], Michael D. Morgan[1,*]

[1]Institute of Medical Sciences, University of Aberdeen, Foresterhill, Aberdeen, AB25 2ZD, United Kingdom
*Corresponding author. Institute of Medical Sciences, University of Aberdeen, Foresterhill, Aberdeen, AB25 2ZD, United Kingdom.
E-mail: michael.morgan@abdn.ac.uk
Associate Editor: Xin Gao

Abstract

Summary: **Combinatorial barcoding technologies for single-cell nucleotide sequencing, such as split-pool ligation protocols, involve sequential rounds of cell barcoding to uniquely tag individual cells. The rapid adoption of combinatorial barcoding in recent years is due in part to its scalability across cells and samples. However, small shifts in barcode positions within sequencing reads caused by technical artifacts, e.g. during barcode incorporation or synthesis, can impact the accurate assignment of reads to cell barcodes. Existing processing tools typically assume barcodes contain fixed-length nucleotide sequences located at fixed positions within reads, overlooking any positional variability. Consequently, reads containing truncated or mispositioned barcodes are discarded during initial data processing steps leading to significant data loss. To solve this limitation and maximize the retention of sequencing reads from single-cell combinatorial barcoding experiments, we introduce *scarecrow*. This tool screens a subsample of reads to generate position-specific barcode profiles, which are then used to flexibly identify barcode sequences in each read whilst accounting for positional errors, a phenomenon we refer to as "jitter". Barcode matches are then prioritized to minimize nucleotide mismatches and the degree of jitter. These initial profiles are subsequently used to extract and error correct barcode combinations in high throughput sequencing libraries. By incorporating jitter into barcode error correction, *scarecrow* enables greater data recovery and improved downstream single-cell analyses. *Scarecrow* is fully open access, implemented in Python, and generates output files using standardized sequence file formats for maximal interoperability. A detailed explanation of the *scarecrow* workflow can be found in the supplementary materials.**

Availability and implementation: **Scarecrow is freely available on GitHub https://github.com/MorganResearchLab/scarecrow and Zenodo https://doi.org/10.5281/zenodo.18621784.**

1 Introduction

Single-cell RNA sequencing (scRNAseq) using combinatorial barcoding strategies such as split-pool ligation offer exponential increases in cell indexing capacity compared to single-indexing strategies (Rosenberg *et al.* 2018). For both single- and combinatorial-indexing approaches, faithful transcriptome profiling relies on accurate demultiplexing of sequencing reads to single-cell barcodes. One challenge of this demultiplexing step is variability in barcode positioning (e.g. 1–2 bp shifts) or "jitter" due to technical artifacts. This can result from incomplete barcode synthesis (Sun *et al.* 2025) or be introduced during library preparation and sequencing for example through polymerase slippage during PCR amplification steps, adapter misalignment, or rare indels from enzymatic steps during adapter ligation or tagmentation. Such positional jitter may be more consequential for combinatorial barcoding than single-indexing strategies because of the reliance on multiple barcoding reactions. Existing demultiplexing approaches typically work on the basis that barcodes start either at a fixed position or a position relative to a sequence motif such as a linker (Kuijpers *et al.* 2024). Jitter or truncation will result in the barcode sequence at the expected location not matching a whitelist of known barcodes even if allowing for a small number of nucleotide substitutions. Consequently, these reads are lost to downstream analysis. Furthermore, not accounting for positional jitter of barcodes may result in incorrect barcode assignment compromising transcriptome profile fidelity. For example, a barcode may be identified at the expected position within a sequencing read with a single mismatch, while a perfect match barcode may start one nucleotide upstream or downstream of that position. Efforts to introduce a standardized specification for describing sequencing data, detailing the assay, sequence and library structure, such as *seqspec* (Booeshaghi *et al.* 2024) are commendable,

Received: 17 October 2025. **Revised:** 2 March 2026. **Accepted:** 31 March 2026

however the tools that leverage such specifications are still limited by fixed barcode position assumptions.

To address the above challenges, we introduce *scarecrow*. First, *scarecrow* screens a subset of reads against barcode whitelists to generate position-specific barcode distributions. These barcode profiles are then used to flexibly match barcodes to an input whitelist whilst accounting for positional jitter, prioritizing barcode matches with the fewest mismatches and smallest jitter. Our benchmarks on Parse Evercode and Scale Biosciences data demonstrate up to a 65% increase in usable reads when accounting for jitter. *Scarecrow* requires no prior knowledge of barcode positions within the library structure, and its outputs are structured for compatibility with standard downstream tools (e.g. *kallisto*, STAR, umi-tools), offering a robust solution for maximizing data yield across diverse single-cell applications.

2 Barcode matching

The standard *scarecrow* workflow consists of three steps performed in the following order: *seed* barcodes, *harvest* barcode profiles, and *reap* the sequences (Fig. 1A; Supplementary Material 1, available as supplementary data at *Bioinformatics* online). While both the *seed* and *reap* steps employ the barcode matching strategy that underpins the utility of *scarecrow*, *seed* only returns exact matches, without accounting for mismatches or jitter, returning a list of barcode matches and their positions. The *harvest* step gathers the barcode positions from the various barcode indices returned by *seed* and generates a profile of expected barcode positions for use with *reap*. Finally, *reap* identifies barcode matches based on the expected positions in the profile, while accounting for jitter and allowing for a specified number of mismatches.

2.1 Set-based approach

Scarecrow creates two complementary sets of barcodes: (i) a "reference" set for each whitelist of barcodes, and (ii) a "mismatches" set of all possible variants of each barcode on each whitelist allowing up to m nucleotide mismatches. Given an expected start position for a barcode of length n, an interval is defined that incorporates the jitter allowing barcoding searching within a range of $\pm x$ nucleotides upstream and downstream. For example, for an expected start position of 10 with jitter = 2, the search range would span position 8 to 12. For each position within this range, the n-length substring is extracted from the sequencing read and stored in a third "query" set. In the above example, the query set would contain five candidate sequences. The query set is compared to the whitelist reference set to first identify barcodes with exact matches. If any are found, these are returned in order of proximity to the expected start position. If no exact matches are found, the query set is then compared to the whitelist mismatches set. In both cases, if any matches are found, these are prioritized in order of fewest mismatches first, followed by closest distance to the expected start position. Accordingly, the *scarecrow* algorithm prioritizes barcode matches in the following order: (i) a perfect match, (ii) a match with the fewest nucleotide substitutions, and (iii) a match closest to the expected start position. If two or more barcodes satisfy all criteria equally (e.g. two candidates with one mismatch located one base from the expected start position), the result is treated as ambiguous, and a null match is returned. If jitter results in the search space range exceeding the start or end positions of a read, then the query sequence is trimmed at one end and padded with N's at the other, enabling the above algorithm to proceed while treating the N's as mismatches. If no valid matches are identified, *scarecrow* returns an invalid barcode—a string of N's of length n.

2.2 Trie-based approach

The set-based matching approach is well-suited to relatively small barcode whitelists, such as 96 barcodes typically used at each indexing step in combinatorial barcoding protocols. However, for larger barcode whitelists, such as those used with single-barcoding protocols, we have also implemented a hybrid trie and index-based algorithm. During the *seed* step, this algorithm generates both a trie and either a k-mer or seed index from each whitelist of barcodes. The trie is used for exact barcode matching whilst the k-mer/seed index supports approximate matching. During the *reap* step, if no exact match is found during trie traversal, an approximate match is sought using the index. The same set of n-length "query" sequences are extracted from the sequencing read, as described in the set-based approach. If the Hamming distance between the query sequence and a k-mer/seed is $\leq$ maximum mismatches (user defined), then the k-mer/seed is added to a list of candidate matches. Final barcode selection follows the prioritization strategy as in the set-based method: exact match, fewest mismatches, and minimal distance from the expected start position. This hybrid trie/k-mer approach is efficient because it significantly reduces the search space and does not require reference and mismatches sets to be held in memory.

3 Workflow

We recommend that *seed* is run independently for each barcode whitelist, instructing it to sample 10 000 reads each time to generate well-supported barcode profiles. The outputs from *harvest* can be used to diagnose any potential issues with barcode incorporation or assumptions about library structure. For example, if a barcode returns a low read fraction (Fig. 1, available as supplementary data at *Bioinformatics* online) this can indicate that it was already demultiplexed by the sequencing provider. In such cases, the barcode should instead be recovered from the FASTQ header for each read using the *weed* command. To avoid redundant processing time, the corresponding barcode should be removed from the input barcode positions CSV before running *reap*. To ensure efficient barcode matching, barcode whitelists should contain sequences in the same orientation as they appear in the read. This is because it is computationally more efficient to match barcodes to reads in the same orientation than to reverse complement and re-search if no match is found. Finally, we recommend adapter trimming after running *scarecrow* to ensure consistent read lengths. This is because adaptor removal may lead to highly variable barcode positions, causing positional jitter to miss genuine barcodes outside of the

expected range (e.g. ±2 nucleotides), resulting in loss of valid barcodes. The optimal jitter and mismatch settings depend on the scRNA-seq chemistry, and are impacted by barcode start position, barcode length, and whitelist size (refer to online documentation for details). Barcodes starting at the beginning of a sequencing read or terminating at the end require a higher mismatch value to match barcodes due to potential base-clipping. Longer barcodes can accommodate larger mismatch values than shorter barcodes as there is a reduced risk of collision. For comparison, existing proprietary algorithms, including Cell Ranger and Trailmaker accommodate up to 1 and 2 mismatches, respectively.

4 Examples

We illustrate the utility of *scarecrow* with publicly available data generated from Parse Biosciences Evercode and Scale Biosciences QuantumScale library preparations derived from (De Simone *et al.* 2025) (Supplementary Material 2, available as supplementary data at *Bioinformatics* online). These data are chosen because they profiled peripheral blood mononuclear cells, a diverse combination of immune cells, from the same donor across different commercial and widely used single-cell RNA-sequencing protocols. Considering the QuantumScale data, we show that applying jitter of 1 returns 65% more reads with matched (valid) barcodes compared to applying no jitter ($n = 311\ 847\ 144$ vs. 189 333 595; Fig. 1B; Table 1, available as supplementary data at *Bioinformatics* online). This also results in 67% more reads with perfectly matched barcodes (293 542 244 vs. 175 732 492), 65% more reads for alignment after sifting to remove reads with unmatched barcodes (311 768 984 vs.189 286 390), and 64% more uniquely mapped reads (214 478 807 vs. 130 506 898). These data gains propagate to downstream processing steps. After quantifying single-cell gene expression with *kallisto* and applying basic quality filtering, using jitter of 1 returned 50% more cells (12 789 compared to

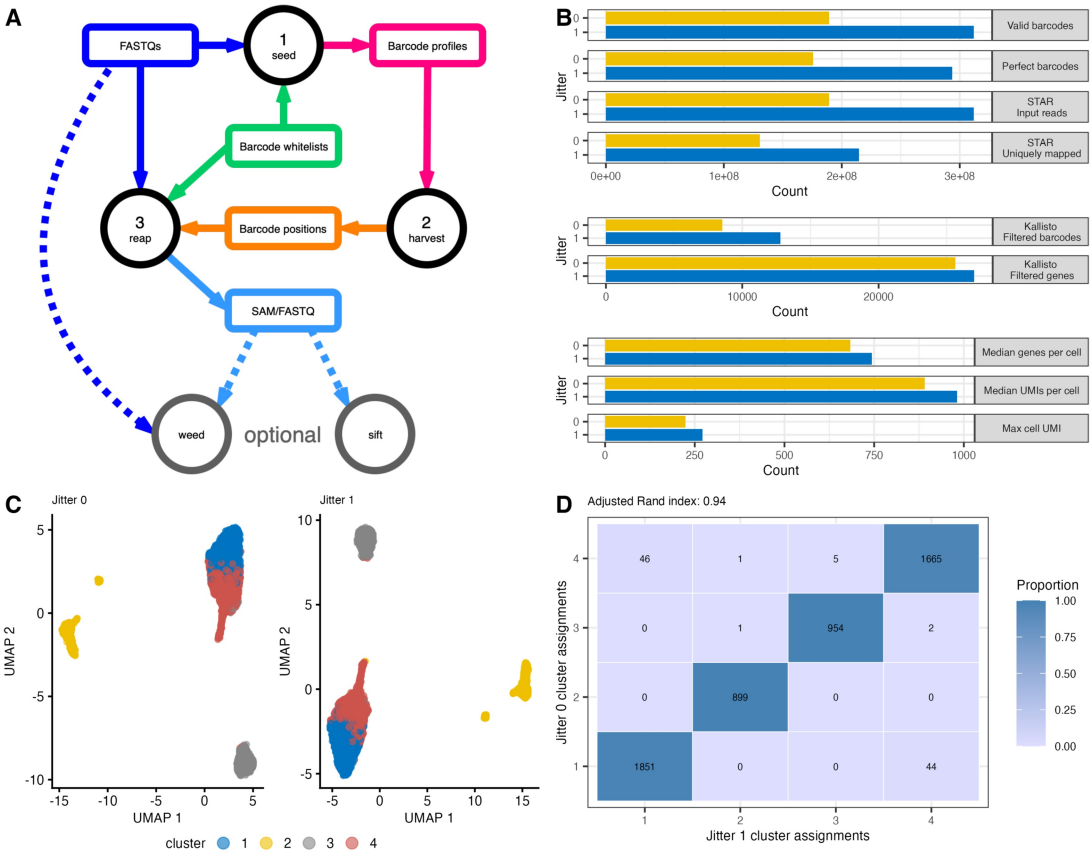

Figure 1 (A) Schematic diagram of the *scarecrow* workflow. This begins with FASTQ files (dark blue box) and barcode whitelists (green box) being passed to *seed*. The resulting barcode profiles (pink box) are passed to *harvest* which outputs a file of expected barcode positions (orange box). These positions, together with the FASTQ files and barcode whitelists are passed to *reap*, which returns either a SAM or interleaved FASTQ file (light blue box) containing the target sequence and barcode information. Optional steps, denoted by dashed lines, include extracting a barcode from the FASTQ read header with *weed*, if the reads have already been demultiplexed on one barcode, and removing any reads containing unmatched barcodes with *sift*. (B) Summary counts from processing Scale Bio data with no jitter and with jitter = 1. This illustrates under each condition: the number of raw reads with valid barcodes, the number of reads with perfectly matching barcodes, the number of reads post-*sift* passed to STAR for alignment, the number of uniquely mapped reads, the number of genes and barcodes remaining post-filtering on a *kallisto* count matrix, the median genes and UMIs per cell in the filtered count matrix, and the max UMI observed across the matrix. (C) Uniform manifold approximation and projections generated from processed and quality controlled ScaleBio profiling of PBMC with no jitter (left) or jitter = 1 (right). Points are colored by cluster assignment. (D) Heat map indicating agreement of cell cluster assignment from (C) between jitter values with adjusted Rand index shown.

8530) and 5% more genes detected across cells (27 015 vs. 25 616) (Table 2, available as supplementary data at *Bioinformatics* online). We also increased the median number of genes per cell with non-zero expression by 9% (744 compared to 684), a 10% increase in the median number of UMIs per cell (981 compared to 891), and a 21% increase in the maximum cell UMI observed (272 compared to 224) (Table 3, available as supplementary data at *Bioinformatics* online). Therefore, recovering more reads at barcode error correction steps leads to higher-quality single-cell expression profiles for down-stream analysis tasks.

Our analyses indicate that approximately 2% of reads are assigned different barcodes when accounting for jitter (Table 4, available as supplementary data at *Bioinformatics* online). This occurs because reads initially assigned to a barcode with a single mismatch with no jitter can be reassigned to perfect matching barcode with jitter > 0. To ensure that these reassignments did not introduce artifacts due to our advanced error correction, and to confirm the expected gains in data quality, we performed a uniform scRNA-seq analysis using *kallisto* (Melsted *et al.* 2021) to generate count matrices with and without jitter applied on both Evercode and QuantumScale data sets. Following basic quality control to remove low quality cells and potential doublets, we identified common barcodes between the matrices generated with and without jitter. We then performed principal components analysis for dimensionality reduction, visualized the results using uniform manifold approximation and projection, and grouped cells into clusters using separate shared nearest neighbor graphs (Fig. 1C). Comparing cluster membership highlighted the high concordance between both datasets (adjusted Rand index; ARI ≥ 0.94) indicating that our jitter-based barcode error correction did not negatively impact neighbor-finding or cell clustering. Moreover, we found no evidence of hybrid cells caused by barcode reassignment. In summary, *scarecrow* is a library agnostic pre-processing command-line tool that yields more reads and cells from combinatorial barcoding scRNA-seq data, enabling the maximal use of these highly scalable technologies.

Author contributions

David Wragg (Conceptualization [equal], Formal analysis [lead], Investigation [lead], Methodology [lead], Software [lead], Writing—original draft [lead], Writing—review & editing [equal]), Eunchai Kang (Funding acquisition [equal], Supervision [supporting], Writing—review & editing [supporting]), Michael D. Morgan (Conceptualization [equal], Formal analysis [equal], Funding acquisition [equal], Project administration [lead], Supervision [lead], Writing—original draft [equal], Writing—review & editing [equal])

Supplementary material

Supplementary material is available at *Bioinformatics* online.

Conflict of interests

None declared.

Funding

This work was supported by Alzheimer's Society UK (grant number 636 to M.D.M. and E.K.), and the Royal Society (grant number RG/R2/232125 to M.D.M.).

Data availability statement

Data from De Simone *et al.* were downloaded from Sequence Read Archive (accessions SRR28867557 & SRR28867558).

References

Booeshaghi AS, Chen X, Pachter L *et al.* A machine-readable specification for genomics assays. *Bioinformatics* 2024; **40**:btae168.

De Simone M, Hoover J, Lau J *et al* . A comprehensive analysis framework for evaluating commercial single-cell RNA sequencing technologies. *Nucleic Acids Res.* 2025;**53**:gkae1186.

Kuijpers L, Hornung B, van den Hout-van Vroonhoven MCGN *et al.* Split Pool ligation-based single-cell transcriptome sequencing (SPLiT-seq) data processing pipeline comparison. *BMC Genomics* 2024;**25**:361.

Melsted P, Booeshaghi AS, Liu L *et al.* Modular, efficient and constant-memory single-cell RNA-seq preprocessing. *Nat Biotechnol* 2021;**39**:813–8.

Rosenberg AB, Roco CM, Muscat RA *et al.* Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. *Science* 2018;**360**:176–82.

Sun J, Philpott M, Loi D *et al.* Enhancing single-cell transcriptomics using interposed anchor oligonucleotide sequences. *Commun Biol* 2025;**8**:67.

Bioinformatics, 2026, 42, 1–4
https://doi.org/10.1093/bioinformatics/btag182
Applications Note