Sequence analysis

Quantifying orthogonal barcodes for sequence census assays

A. Sina Booeshaghi 1, Kyung Hoi (Joseph) Min 2, Jase Gehring 3,*, Lior Pachter 1,4,*

1Division of Biology and Biological Engineering, California Institute of Technology, Pasadena, CA 91125, United States
2Department of Computer Science and Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, United States
3Arcadia Science, Berkeley, CA 94702, United States
4Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125, United States

*Corresponding authors. Arcadia Science, 3100 SAN PABLO AVENUE, SUITE #120, Berkeley, CA 94702, United States. E-mail: jgehring@caltech.edu (J.G.); Lior Pachter@caltech.edu (L.P.)

Associate Editor: Alex Bateman

Abstract

Summary: Barcode-based sequence census assays utilize custom or random oligonucleotide sequences to label various biological features, such as cell-surface proteins or CRISPR perturbations. These assays all rely on barcode quantification, a task that is complicated by barcode design and technical noise. We introduce a modular approach to quantifying barcodes that achieves speed and memory improvements over existing tools. We also introduce a set of quality control metrics, and accompanying tool, for validating barcode designs.

Availability and implementation: https://github.com/pachterlab/kb_python, https://github.com/pachterlab/qcbc.

1 Introduction

Single-cell RNA sequencing assays rely on unique sequences of nucleotides (“barcodes”) to associate sequences from mRNA molecules with individual cells. This strategy has been extended to quantify orthogonal biological or technical information in what is known as “multimodal” genomics. For example, the 10× Genomics Feature Barcoding assay quantifies the expression of cell-surface proteins using targeted antibody barcodes (“Feature Barcode Overview—Software—Single Cell Gene Expression—Official 10× Genomics Support” n.d.). The ClickTag (Gehring et al. 2020), MultiSeq (McGinnis et al. 2019), and Cell Hashing (Stoeckius et al. 2018) assays, on the other hand, use custom sample-specific barcodes to group cells enabling multiplexing and reducing batch effects. Moreover, similar approaches have been used for CRISPR screens (Dixit et al. 2016, Datlinger et al. 2017, Schraivogel et al. 2020), targeted perturbation assays (Gehring et al. 2020, Srivatsan et al. 2020), and recently for spatial genomics assays (Srivatsan et al. 2021).

The processing of data from these assays has required the development of custom tools, resulting in a proliferation of application- and assay-specific software (Roelli et al. 2019, Johnson et al. 2023). We introduce a broadly applicable framework, which we term “kITE” for kallisto Indexing and Tag Extraction, for uniformly quantifying orthogonal barcode-based sequence census assays that has several advantages over assay-specific tools. First, by virtue of providing a single solution to seemingly distinct problems, it serves as a unifying framework for biology x barcode assays and will simplify the development and deployment of novel assays. Second, the software implementation we share is portable, engineered to work across platforms, and is modular, allowing for easy adaptation and customization. It makes use of the kallisto and bustools programs that have been previously extensively validated and benchmarked for quantification of the biological sequences in barcode x biology assays such as single-cell and bulk RNA sequencing (Bray et al. 2016, Melsted et al. 2021). Finally, our method is faster than existing tools, and we identify corner cases in processing that have been missed in previous analyses. The examination of such cases led us to develop a tool, called gcbc, for assessing the quality of barcode sets, and that should prove to be a useful standalone package. It is applicable both during assay development, and during assay quality control.

2 Methods

Feature Barcoding assays use short barcodes and are susceptible to ambiguous or incorrect assignment if errors are introduced during the experimental procedure. Inspired by the CITE-seq-count approach, kITE begins by generating a list of all single-base mismatches of the Feature Barcodes used in an experiment (Fig. 1). The resulting “mismatched FASTA” file is used as input for the “kallisto index” program with k set appropriately (Section 2). Finally, the kallisto | bustools pipeline is used to quantify the dataset using the “mismatch index” generated after running “kallisto index” on the mismatched FASTA file. In this way, kallisto will effectively scan
Sequence can result in possible incorrect barcode assignment. The introduction of errors during Multiseq. These two Multiseq barcodes are short (8 bp in length), yet share long subsequences up to 7 bp in length. BC74, when processing the PDX samples assayed with Multiseq. These two Multiseq barcodes are short (8 bp in length), yet share long subsequences up to 7 bp in length.

3 Results
To evaluate the accuracy and efficiency of kITE across a range of assays, we gathered data from six different assays, and compared kITE quantifications to those produced with assay-specific tools. We first quantified cell surface protein abundance for the 10× Genomics 5266 Peripheral Blood mononuclear cells dataset, which targeted 32 cell surface proteins. We used kallisto bustools and a mismatch index to pseudosalign the target sequences (Appendix A.1.1) and found high concordance between our method and Cell Ranger across varying read depths, with an average Pearson correlation of 0.99 (Fig. 2a–f). kITE was five times faster than Cell Ranger and required 3.5 times less RAM (Appendix A.1.2).

Next, we tested how accurately we could quantify and demultiplex pooled samples tagged with sample-specific barcodes. We processed data from 3795 methanol-fixed mouse neural stem cells from four multiplexed samples generated with the ClickTag assay (Supplementary Fig. S1), and transduced with 90 sgRNAs targeting 45 different genes. We achieve high concordance against Cell Ranger with an average Pearson correlation of 0.99 across all cells for nearly sgRNA’s, with two sgRNAs in the 10× CRISPR screen dataset, EZR-1 and PPIB-2, differing substantially (Supplementary Fig. S5). Both of these two sgRNA’s share a subsequence of 16 bp with EZR-2 and PPIB-1 respectively. The latter two sequences begin with GCC and GGA, both of which are a single hamming distance away from the three bases that precede the target sgRNA in the protospacer sequence (Supplementary Fig. S4b). Therefore, a single mutation two bases away from the start of the EZR-1 sgRNA sequence could result in an ambiguous read. To resolve this, we trimmed the protospacer sequence off of the reads (Appendix A.1.8).

We also processed 10 860 K562 cells expressing dCas9–KRAB from TAP-Seq transduced with 86 sgRNAs targeting 14 promoters or enhancers and 30 control sgRNAs to compare assignment of sgRNAs to individual cells. We found 97.8% of all 10 860 barcodes are fully concurrent with TAP-Seq assignments, 0.53% of cells are partially concurrent, and 1.6% of cells are concurrently discordant. (Appendix A.1.10). The discrepancies are likely the result of differences between mapping procedures producing differences in counts near the threshold (8 UMI counts) resulting in differential cell assignment for cells with low UMI counts (Supplementary Fig. S6).

3.1 Read trimming
Sequencing libraries often contain adapter or spurious sequences that flank the barcode and the presence of these shared sequences can make mapping challenging. Sequencing reads are often trimmed, with tools such as trimmomatic (Bolger et al., 2014), to avoid mapping noninformative regions that flank a barcode, a task that is crucial when barcodes share subsequences at their ends. Trimming helps to...
avoid longer sequence clashes that contain both the shared barcode subsequence and the shared nonbarcode sequence. The 10× CRISPR assay, e.g. contains two barcodes which share a 16-bp subsequence (the ends of which are one hammering distance from a subsequence of the protospacer sequence that flank the barcode) (Supplementary Fig. S4). Similarly, the Multiseq assay produces sequencing reads where the barcodes are flanked by poly-A stretches and two barcodes share a 7-bp subsequence that contains the end of the barcode.

3.2 Barcode design validation

Orthogonal barcode sequences designed for quantifying multimodal data, demultiplexing samples, or performing large-scale perturbation experiments can clash if the sequences are not sufficiently unique. To allow researchers to assess the extent of these issues in datasets they are analyzing, or for assays they are developing, we developed a quality control tool, qcbc, and a simulation framework to assess the quality of a set of barcodes. Our tool is complementary to the simulation framework proposed in (Johnson et al. 2023) and expands on the functionality of DNABarcodes (Buschmann 2017) which produces pairwise barcode distance comparisons. qcbc computes multiple metrics for barcode quality control:

1) diversity of barcodes (subject to distance constraints) (Fig. 3a),
2) number of barcodes that share a subsequence of a given length (Fig. 3b),
3) distribution of Hamming distances between pairwise barcodes (Fig. 3c and d),
4) distribution of homopolymers of a given length (Fig. 3e), and
5) per position and barcode nucleotide distribution (Fig. 3f, Supplementary Fig. S7).

With qcbc we are able to identify the 16-bp shared subsequence with EZR-2 and PPIB-1 sgRNAs in the 10× CRISPR that result in improper quantification. qcbc is also able to identify the 7-bp shared subsequence between and BC74 in the Multiseq assay.

These metrics are important for understanding the impact that barcode error correction has on rescuing sequencing reads. A large diversity of unique barcodes with shorter shared subsequences and larger pairwise distances between barcodes make it easier to disambiguate sequencing reads with pseudoalignment. Additionally, long homopolymer runs and low nucleotide content diversity could contribute to sequencing errors (“Illumina: What Is Nucleotide Diversity and Why Is It Important?” 2016, Sina Booeshaghi and Pachter 2022).

To evaluate the impact that base errors have on error correction, we developed a simulation framework that reports the fraction of lost barcodes for a given error rate in the observed sequences (Supplementary Fig. S8) (Appendix A.1.13). Short barcodes, such as those in the Multiseq assay, can yield insufficient barcode diversity that results in a greater fraction of barcodes that cannot be unambiguously assigned (Supplementary Fig. S9). Barcodes with small minimum-pairwise/edit distance (Fig. 3d and e) and long shared subsequences (Fig. 3b) such as those in the Multiseq and 10× CRISPR assay can result in more clashes (Appendix A.1.12).

4 Discussion

We have demonstrated a broadly applicable approach to align, error-correct, group, and count sequences from orthogonal barcodes to quantify multimodal data such as cell surface protein abundance, to demultiplex samples, and to perform large-scale perturbation experiments. Our approach is fast, memory efficient, and extensible: orthogonal barcode quantifications can be obtained for numerous assays with variable sequence
structure such the recently developed spatial RNA-seq assays (Srivatsan et al. 2021) and massively parallel reporter assays (Gordon et al. 2020). This allows future assay developers to avoid producing, validating, and maintaining software implementing ad hoc solutions. Moreover, some current methods are slow, have high memory requirements, and are specifically tailored for specific assays, which can result in batch effects when performing integrative analysis. Methods such as CITE-seq-Count offer extensibility features for different barcode strategies, but are slow and memory-intensive. The speed and efficiency of our approach enables reproducible single-cell analysis in the cloud with tools such as Google Colab. We anticipate that our framework for quantifying orthogonal barcoding assays will be useful as multimodal assays proliferate (Packer and Trapnell 2018), and that our quality control tool will help assay developers avoid pitfalls in barcode design and help practitioners quality control orthogonal barcode assays before generating data. Finally, while our approach is currently engineered to correct mismatch errors, robustness to other types of sequencing errors are possible via other methods that could readily be incorporated in our modular framework (Zorita et al. 2015, Sina Booeshaghi et al. 2020, Johnson et al. 2023).

Supplementary data
Supplementary data are available at Bioinformatics Advances online.

Conflict of interest
None declared.

Funding
None declared.

Appendix

A.1 Methods
All code, parameter options, and data can be found in the paper’s GitHub repository: https://github.com/pachterlab/BMGP_2020. kite is implemented in kb_python and can be found here: https://github.com/pachterlab/kb_python. The qcbe barcode validator can be found: https://github.com/pachterlab/qcbe.

A.1.1 Mismatch index, pseudoalignment, and counting
To quantify against a known set of target sequences we first build an alignment index containing all Hamming distance = 1 variants of each target. Restricting the alignment index to Hamming distance = 1 variants allows for recovering more reads while keeping the index size small (Supplementary Fig. S10). The hamming-1 index for the Clicktag assay is 72.8 KB and the hamming-0 index is 2.1 KB. Note that we generate the kallisto index (de Bruijn Graph) for all orthogonal barcodes, including all hamming 1 distance variants of each barcode (one base-pair mutation). We do not include entries that collide as they will produce ambiguities in mapping. Additionally, we set the k-mer size to be the length of the orthogonal barcode, if the barcode length is odd, or one minus the length of the orthogonal barcode, if the barcode length is even, since the k-mer length must be odd (Bray et al. 2016).

Targeted sequencing reads are then pseudoaligned to the custom index using kallisto to generate a Barcode, UMI, Set (BUS) file (Melsted et al. 2019). UMI counts are then
aggregated per cell and quantified per feature barcode using `bustools` to generate a target count matrix.

**A.1.2 Speed and memory comparison**

Preprocessing steps were benchmarked with GNU time `-v` option, and the elapsed (wall clock) time and maximum resident set size (kbytes) were used to perform the time and memory comparisons. All code was benchmarked on Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0–136-generic x86_64) with Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70 GHz processors.

**A.1.3 Barcode design evaluation**

qcbc allows users to evaluate potential orthogonal barcode designs. The tool reports multiple metrics that may help users understand potential error modes. Each of the metrics is explained below along what various values mean:

1) diversity of unique barcodes (subject to distance constraints) (Fig. 3a),
   a) If the diversity of barcodes is high, relative to the maximum possible number of barcodes (constrained by a given pairwise distance), then the design space of barcodes was used more effectively.
   b) Opting for a shorter barcode length at a given pairwise Hamming distance constraint may save reagents and assay complexity.

2) number of barcodes that share a subsequence of a given length (Fig. 3b),
   a) If the number of barcodes that share a subsequence of a given length is high, then k-mer based mapping may yield multimapping reads. For example, if the barcode length is 16 bp and multiple barcodes share a 15-bp subsequence, then ambiguity in k-mer based mapping will arise.
   b) Opting for barcodes with shorter ambiguous subsequences enable effective barcode discrimination using k-mer based mapping.

3) distribution of Hamming distances between pairwise barcodes (Fig. 3c and d)
   a) If the distribution of pairwise Hamming distances between barcodes is low, then the mismatched index may yield barcodes that share a common subsequence giving rise to multimapping reads.
   b) Opting for larger pairwise distances in barcode design will enable effective discrimination of barcodes at a given Hamming distance, rescuing more reads.

4) distribution of homopolymers of a given length (Fig. 3e)
   a) If the distribution of homopolymers of a given length is high, then sequencing with technologies such as Ultima Genomics (Sina Boosheghai and Pachter 2022) which uses mostly natural synthesis for sequencing, may yield errors.
   b) Opting for fewer homopolymers may help alleviate this issue.

5) per position and barcode nucleotide distribution (Fig. 3f, Supplementary Fig. S7)
   a) If the per position and barcode nucleotide distribution is not uniform, then sequencing errors may arise during basecalling.
   b) Opting for a more uniform distribution of nucleotides may help improve base calling during sequencing with technologies such as Illumina.

**A.1.4 10x feature barcode**

FASTQ files for 5266 Peripheral blood mononuclear cells dataset targeting 32 cell surface proteins were downloaded from [https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.2/5k_pbmc_protein_v3.3](https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.2/5k_pbmc_protein_v3.3). A mismatch index was created using the Feature Reference file and `kb` ref with the `kite` workflow. The reads were pseudoligned and a feature count matrix was made using `kb` count with the `kite`: 10x FB technology option. `Cell Ranger`, the Feature Reference, and refdata-gex-GRCh38-2020-A was used to process the FASTQ files into a feature count matrix.

All preprocessing steps are runnable via a Google Colab notebook: [https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/10xB/10xB_preprocess.ipynb](https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/10xB/10xB_preprocess.ipynb). All matrix comparisons are runnable via a Google Colab notebook: [https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/3.0.2/5k_pbmc_protein_v3.3](https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/3.0.2/5k_pbmc_protein_v3.3).

**A.1.5 ClickTag**

FASTQ files for 3795 methanol-fixed mouse neural stem cells from four multiplexed samples were downloaded from [https://caltech.box.com/shared/static/zzqam7yul7ujetqyhnhd4lvf8hvqojyg.g](https://caltech.box.com/shared/static/zzqam7yul7ujetqyhnhd4lvf8hvqojyg.g). A mismatch index was created using the list of targeted barcodes file and `kb` ref with the `kite` workflow. The list of barcodes can be found here: [https://github.com/pachterlab/BMGP_2020/tree/main/references/clicktag](https://github.com/pachterlab/BMGP_2020/tree/main/references/clicktag). The reads were pseudoligned and a feature count matrix was obtained using `kb` count with the `kite` technology option. `CITE-seq-Count` and the targeted barcodes were used to process the FASTQ files into a feature count matrix.

All preprocessing steps are runnable via a Google Colab notebook: [https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/clicktag/clicktag_preprocess.ipynb](https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/clicktag/clicktag_preprocess.ipynb). All matrix comparisons are runnable via a Google Colab notebook: [https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/clicktag/clicktag_preprocess.ipynb](https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/clicktag/clicktag_preprocess.ipynb).

We generated a `kallisto` index with no mismatches by running `kb` ref with the—no-mismatches option and compared the resultant count matrix generated with `kb` count to the count matrix generated with mismatches.

**A.1.6 Multiseq**

FASTQ files for 9551 primary patient-derived xenograft (PDX) samples from nine multiplexed samples were downloaded from [https://caltech.box.com/shared/static/0scoe38xvcoxfd6znnm848pipm2r1ip.g](https://caltech.box.com/shared/static/0scoe38xvcoxfd6znnm848pipm2r1ip.g). A mismatch index was created using the list of targeted barcodes file and `kb` ref with the `kite` workflow. The list of barcodes can be found here: [https://github.com/pachterlab/BMGP_2020/tree/main/references/multiseq](https://github.com/pachterlab/BMGP_2020/tree/main/references/multiseq). The reads were pseudoligned and a feature count matrix was obtained using `kb` count with the `kite` technology option. `CITE-seq-Count` and the targeted barcodes were used to process the FASTQ files into a feature count matrix.

All preprocessing steps are runnable via a Google Colab notebook: [https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/multiseq/multiseq_preprocess.ipynb](https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/multiseq/multiseq_preprocess.ipynb). All matrix comparisons are runnable via a Google Colab notebook: [https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/multiseq/multiseq.ipynb](https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/multiseq/multiseq.ipynb).

The FASTQ reads were trimmed using `seqtk` trimfq with the -e option set to the length of the read minus the 8 such that only the 8-bp barcode was retained.
A.1.7 Cell hashing

FASTQ files for 10807 PBMCs from eight multiplexed samples were downloaded from https://caltech.box.com/shared/static/0scoe38xvcoif62xnn848pimh2zr1ip.gz. A mismatch index was created using the list of targeted barcodes file and *kb* ref with the *kite* workflow. The list of barcodes can be found here: https://github.com/pachterlab/BMGP_2020/tree/main/references/multiseq. The reads were pseudoaligned and a feature count matrix was obtained using *kb* count with the *kite* technology option. CITE-seq-Count and the targeted barcodes were used to process the FASTQ files into a feature count matrix. All preprocessing steps are runnable via a Google Colab notebook: https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/cellhash/cellhash_preprocess.ipynb. All matrix comparisons are runnable via a Google Colab notebook: https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/cellhash/cellhash.ipynb

A.1.8 10× CRISPR

FASTQ files for 2799 A549 lung carcinoma cells expressing nuclease-dead Cas9–Krüppel-associated box (dCas9–KRAB) from 10× Genomics CRISPR screen transduced with 90 sgRNAs targeting 45 different genes were downloaded from https://support.10xgenomics.com/single-cell-gene-expression/datasets/4.0.0/SC3_v3_NextGem_DL_Crispr_10k. A mismatch index was created using the Feature Reference file and *kb* ref with the *kite* workflow. The reads were pseudoaligned and a feature count matrix was obtained using *kb* count with the kite: 10×FB technology option. Cell Ranger, the Reference, and refdata-gex-GRCh38-2020-A was used to process the FASTQ files into a feature count matrix. All preprocessing steps are runnable via a Google Colab notebook: https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/10xCRISPR/10xCRISPR_preprocess.ipynb. All matrix comparisons are runnable via a Google Colab notebook: https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/10xCRISPR/10xCRISPR.ipynb.

The FASTQ reads were trimmed using `seqtk` trimfq with the `-e` option set to the length of the read minus the 19 plus overhang such that only the 19-bp barcode plus the set overhang was retained.

A.1.9 10× CRISPR barcode discordance

We took the kite and Cell Ranger count matrices and looked for the cell barcode that was the most discordance in the sum of counts between the two quantifications (CTGAGCGC ATAAGATG). We then looked for the guide RNA that exhibited the most discordant counts (DPP4-1, GGCAGGCGAG AGTCAGCG). We filtered the Cell Ranger BAM file for reads that were unaligned and then further filtered for the specific cell barcode that exhibited the most discordance. Then we grep’ed for the sequence of the specific guide RNA, that was the most discordant in the cell of interest, counted the number of reads and UMIs for which there was a direct match.

A.1.10 TAP-Seq

FASTQ files for 10860 K562 cells expressing dCas9–KRAB from TAP-Seq transduced with 86 sgRNAs targeting 14 promoters or enhancers and 30 control sgRNAs were downloaded from https://www.ncbi.nlm.nih.gov/sra/?term=SRR9916613. A mismatch index was created using the list of targeted barcodes file and *kb* ref with the *kite* workflow. The list of barcodes can be found here: https://github.com/pachterlab/BMGP_2020/tree/main/references/tapseq. The reads were pseudoaligned and a feature count matrix was obtained using *kb* count with the standard technology option. Perturbation status for each cell was downloaded from https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM4012688 using `fastq GSM4012688 (Galvez-Merchan et al. 2023).` All preprocessing steps are runnable via a Google Colab notebook: https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/tapseq/tapeq_simulation.ipynb. All matrix comparisons are runnable via a Google Colab notebook: https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/tapseq/tapeq.ipynb

A.1.11 Datasets and code

The following table lists all datasets and code used to evaluate kite.

| Assay       | ncells | nfeatures | barcodes | fastqs | preprocess | analysis |
|-------------|--------|-----------|----------|--------|------------|----------|
| 10×FB       | 5266   | 32        | link     | Link   | link       | link     |
| Clicktag    | 3795   | 4         | Link     | link   | link       | link     |
| Multiseq    | 9531   | 9         | Link     | Link   | link       | link     |
| Cell Hashing| 10 807 | 8         | Link     | Link   | link       | link     |
| 10× CRISPR  | 2799   | 90        | Link     | Link   | link       | link     |
| TAP-Seq     | 10 860 | 86        | Link     | Link   | link       | link     |

A.1.12 Barcode validation

For a given list of barcodes we report the maximum number of barcodes that could be made for the given barcode length, the number of barcodes that share a subsequence of length k, and the pairwise hamming distance between all pairs of barcodes. A Google Colab notebook reproducing the results of this simulation can be found here: https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/barcode_valuator.ipynb.

The `qcbc` tool for barcode validation can be found here: https://github.com/pachterlab/qcbc

A.1.13 Barcode simulation

We assumed a per-base error rate that is constant for every base in a barcode and simulated barcode mutants from a set of given barcodes. Then, for multiple error rates, we estimated the number of barcode mutants that are given a hamming distance from the correct barcode that became ambiguous after introducing sequencing errors. A Google Colab notebook reproducing the results of this simulation can be found here: https://github.com/pachterlab/BMGP_2020/blob/main/analysis/notebooks/barcode_sim.ipynb

A.2 Software versions

Anndata 0.6.22.post1
bustools 0.41.0
Cell Ranger 6.0.1
CITE-seq-Count 1.4.4
awk (GNU awk) 4.1.4
grep (GNU grep) 3.1
kallisto 0.48.0
kb_python 0.26.3
Matplotlib 3.0.3
Numpy 1.18.1
Pandas 0.25.3
Scipy 1.4.1
References

Bolger AM, Lohse M, Usadel B. Trimmomatic: a flexible trimmer for illumina sequence data. *Bioinformatics* 2014;30:2114–20.

Booeshaghi AS, Lubock NB, Cooper AR et al. Reliable and accurate diagnostics from highly multiplexed sequencing assays. *Sci Rep* 2020;10:21759.

Bray NL, Pimentel H, Melsted P et al. Near-optimal probabilistic RNA-Seq quantification. *Nat Biotechnol* 2016;34:525–7.

Buschmann T. DNABarcodes: an R package for the systematic construction of DNA sample tags. *Bioinformatics* 2017;33:920–2.

Datlinger P, Rendeiro AF, Schmidl C et al. Pooled CRISPR screening with single-cell transcriptome readout. *Nat Methods* 2017;14:297–301.

Dixit A, Parnas O, Li B et al. Perturb-Seq: dissecting molecular circuits with scalable single-cell RNA profiling of pooled genetic screens. *Cell* 2016;167:1853–66.e17.

“Feature Barcode Overview—Software—Single Cell Gene Expression—Official 10x Genomics Support.” n.d. https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/feature-bc (9 September 2022, date last accessed).

Gálvez-Merchán A, Min KH, Pachter L, Booeshaghi AS. Metadata retrieval from sequence databases with ftq. *Bioinformatics* 2023;39:btaa667.

Gehring J, Hwee Park J, Chen S et al. Highly multiplexed single-cell RNA-Seq by DNA oligonucleotide tagging of cellular proteins. *Nat Biotechnol* 2020;38:35–8.

Gordon M, Grace F, Inoue B et al. lentiMPRA and MPRAflow for high-throughput functional characterization of gene regulatory elements. *Nat Protoc* 2020;15:2387–412.

“Illumina: What Is Nucleotide Diversity and Why Is It Important?” 2016. https://support.illumina.com/bulletins/2016/07/what-is-nucleotide-diversity-and-why-is-it-important.html (1 July 2016, date last accessed).

Johnson MS, Venkataram S, Kryazhimskiy S. Best practices in designing, sequencing, and identifying random DNA barcodes. *J Mol Evol* 2023;91:263–80.

McGinnis CS, Patterson DM, Winkler J et al. MULTI-Seq: sample multiplexing for single-cell RNA sequencing using lipid-tagged indices. *Nat Methods* 2019;16:619–26.

Melsted P, Booeshaghi AS, Liu L et al. Modular, efficient and constant-memory single-cell RNA-seq preprocessing. *Nat Biotechnol* 2021;39:813–8. https://doi.org/10.1038/s41587-021-00870-2

Melsted P, Ntranos V, Pachter L. The barcode, UMI, set format and BUStools. *Bioinformatics* 2019;35:4472–3.

Packer J, Trapnell C. Single-cell multi-omics: an engine for new quantitative models of gene regulation. *Trends Genet* 2018;34:653–65.

Roelli P, Bhimber BF, Santiagorevale G et al. Hoohm/CITE-Seq-Count: 1.4.2. 2019. https://doi.org/10.5281/zenodo.2590196

Schraivogel Daniel AR, Gschwind JH, Milbank DR et al. Targeted perturb-seq enables genome-scale genetic screens in single cells. *Nat Methods* 2020;17:629–35.

Sina Booeshaghi A, Pachter L. Pseudoalignment facilitates assignment of error-prone ultima genomics reads. bioRxiv, https://doi.org/10.1101/2022.06.04.494845, 2022, preprint: not peer reviewed.

Srivatsan SR, McFaline-Figueroa JL, Ramani V et al. Massively multiplex chemical transcriptomics at single-cell resolution. *Science* 2020;367:45–51.

Srivatsan SR, Regier MC, Barkan E et al. Embryo-scale, single-cell spatial transcriptomics. *Science* 2021;373:111–7.

Stoeckius M, Zheng S, Houck-Loomis B et al. Cell hashing with bar-coded antibodies enables multiplexing and doublet detection for single cell genomics. *Genome Biol* 2018;19:224.

Zorita E, Cusco P, Filion GJ. Starcode: sequence clustering based on all-pairs search. *Bioinformatics* 2015;31:1913–9.