Tambe and Pachter *BMC Bioinformatics* (2019) 20:32
https://doi.org/10.1186/s12859-019-2612-0

BMC Bioinformatics

SOFTWARE Open Access

Barcode identification for single cell genomics

Akshay Tambe[1] and Lior Pachter[2*]

Abstract

Background: Single-cell sequencing experiments use short DNA barcode 'tags' to identify reads that originate from the same cell. In order to recover single-cell information from such experiments, reads must be grouped based on their barcode tag, a crucial processing step that precedes other computations. However, this step can be difficult due to high rates of mismatch and deletion errors that can afflict barcodes.

Results: Here we present an approach to identify and error-correct barcodes by traversing the de Bruijn graph of circularized barcode k-mers. Our approach is based on the observation that circularizing a barcode sequence can yield error-free k-mers even when the size of *k* is large relative to the length of the barcode sequence, a regime which is typical single-cell barcoding applications. This allows for assignment of reads to consensus fingerprints constructed from k-mers.

Conclusion: We show that for single-cell RNA-Seq circularization improves the recovery of accurate single-cell transcriptome estimates, especially when there are a high number of errors per read. This approach is robust to the type of error (mismatch, insertion, deletion), as well as to the relative abundances of the cells. Sircel, a software package that implements this approach is described and publically available.

Keywords: Single-cell, Barcodes, Barcode identification, de Bruijn graph, Circularization, K-mer counting

Background

Tagging of sequencing reads with short DNA barcodes is a common experimental practice that enables a pooled sequencing library to be separated into biologically meaningful partitions. This technique is in the cornerstone of many single-cell sequencing experiments, where reads originating from individual cells are tagged with cell-specific barcodes; as such, the first step in any single-cell sequencing experiment involves separating reads by barcode to recover single-cell profiles ([7, 20, 22]. For example, in the Drop-Seq protocol, which is a popular microfluidic-based single-cell experimental platform, DNA barcodes are synthesized on a solid bead support, using split-and-pool DNA synthesis [10], and this approach has been applied to obtain single-cell transcriptome profiles from a number of model- and non-model organisms [3, 6, 13, 16, 19, 21]. Similar split-and-pool barcoding strategies are used in other single-cell sequencing assays such as Seq-Well [4] and Split-seq [14]. One consequence of this synthetic technique is that deletion errors are extremely prevalent; by some estimates 25% of all barcode sequences observed contain at least one deletion [10]. Ignoring such errors can therefore dramatically lower the number of usable reads in a dataset, while incorrectly grouping reads together can confound single cell analysis.

Current approach to "barcode calling", the process of grouping reads together by barcode, use simple heuristics to first identify barcodes that are likely to be uncorrupted, and then "error correct" remaining barcodes to increase yields. However the complex nature of errors, that unlike sequencing based error also include deletions, can lead to large number of discarded reads (reads that could not be assigned to a barcode) [10]. Additionally, some current approaches require that the approximate number of cells in the experiment be known beforehand, and in some experimental contexts such information is not easily obtained.

In such experiments, there are two major approaches toward generating barcodes. In the used by 10× Genomics among others, barcodes drawn are from a known 'whitelist' of sequences, and as such this prior knowledge of a whitelist can be used to simplify error-correction and read assignment. On the other hand, the barcodes generated though split-pool synthesis (including

* Correspondence: lpachter@caltech.edu
[2]Departments of Biology and Computing & Mathematical Sciences, California Institute of Technology, 116 Kerckhoff Laboratory, Pasadena, CA 91125, USA
Full list of author information is available at the end of the article

Drop-seq) are random and no prior information can be used for either error-correction or read assignment. The problem of identifying true barcodes from among many sequences corrupted by mismatch and deletion errors seemingly requires a multiple sequence alignment, from which errors can be detected and corrected [24]. However unlike standard biological sequence alignment settings, the single-cell barcode identification problem requires analysis of millions, if not billions of different sequences. On the other hand, the problem is constrained in that the sequences are short (barcodes are typically 10–16 bp long) and the length of each barcode is the same and known.

Here, we present a fast and error-robust k-mer based approach to detecting random barcodes from sequencing data. Our software circumvents the need for complete (and intractable) multiple sequence alignment, by making use of the idea of circularizing the sequences that are to be error corrected, and rather than pursuing a multiple sequence alignment approach, we instead borrow ideas from genome assembly. However unlike assembly methods developed for reconstructing circular genomes [5] our use of circularization is merely a method for adding robustness to the k-mer fingerprinting of barcodes.

Our methods are implemented in software called Sircel whose input is a list of reads and which outputs the number and sequences of cell-barcodes from error-containing datasets in an unbiased manner. Our implementation is robust to insertion, deletion, and mismatch errors, and requires a minimal number of user-inputted parameters. The output is compatible with downstream single-cell analysis tools such as kallisto [1, 11].

Implementation

K-mer counting is a fast and well-established technique that has previously been used to dramatically speed up the assignment of reads to transcripts for RNA-seq (N. L. [1, 12]; Z. [23]) and metagenomics[17] and as such might be applicable to barcode calling. We reasoned that by counting k-mers we could rapidly identify error-free subsequences within the context of a larger error-containing read. [8, 18] The intuition behind our approach lies in the fact that while many copies of the same barcode may contain a different profile of errors, pairs of such barcodes may share some overlapping subsequence that is error free. However as the barcode errors are expected to be random, it is unlikely that several reads will share the exact same error pattern. Thus, multiple error-containing reads will share k-mers only in their error-free regions where the overlap, while the error-containing k-mers are expected to be unique. As such, frequently occurring k-mers would arise from error-free regions of barcodes, while much less overlap would be expected from error-prone k-mers. Similar reasoning has been previously used to rapidly detect and reject error-containing reads from RNA-seq and DNA assembly [9, 18].

One difficulty associated with error-correcting barcodes using this technique lies in the fact that barcodes are typically very short: for example Drop-Seq barcodes are 12 base pairs long. Conversely in order for a k-mer counting approach to be feasible we must pick a moderately large value for k, typically $k = 8$. As a result there are many positions on a barcode where a single error would ensure that none of its k-mers are shared with an error-free barcode. To circumvent this problem we circularize the barcode sequences before counting k-mers; this ensures that barcodes containing a single mismatch error still share k-mers with the error-free sequence, independent of where the error occurred within the barcode (Fig. 1a).

Furthermore this approach with a small modification allows for addressing the possibility of insertion or deletion error. In a Drop-seq style experiment, if a barcode did contain a deletion error (at an unknown position), then nucleotide at the expected last position of the barcode actually arises from the first nucleotide of either an adaptor sequence or a molecular identifier. As such, the circularized barcode sequence will contain a single incorrect nucleotide, resulting in a large number of incorrect k-mers. However, if before circularizing, the sequence is truncated by one nucleotide, then incorrect nucleotide is removed, allowing the circularization to contain a majority of correct k-mers. This provides the same robustness to positional errors, but additionally allows for robustness to deletion errors. A similar operation can be performed to handle the possibility of insertion errors: in this case, the sequence that is circularized is the observed barcode, extended by one nucleotide (into the adaptor / molecular identifier sequence).As every read contains unknown mutation type(s), we perform all three circularization operations before counting k-mers. Thus, we obtain a set of error-free subsequences that derive from the 'true' barcodes. This procedure guarantees that all reads with either zero or one error contribute some error-free k-mers, while reads with two or more errors sometimes contribute error-free k-mers, depending on the spacing between the errors.

We use these k-mer counts to identify and error-correct complete barcodes. To do this we build and traverse a directed, weighted de Bruijn graph [2]. In this graph, nodes represent subsequences of length $k - 1$, and an edge represents two nodes that directly adjacent to each other in at least one k-mer. The weight of these edges relates to how many times each edge (k-mer) was observed in the entire dataset. Additionally as the barcode portions of these reads are stranded, these edges are directed by the order of their appearance in the read (5′ to 3′). In this graph, which originates from circularized barcode sequences, a cyclic path of length l represents a possible barcode sequence of the same length. We define the capacity of a path to be the weight of the lowest edge within that path. Thus, high-weight paths represent possible

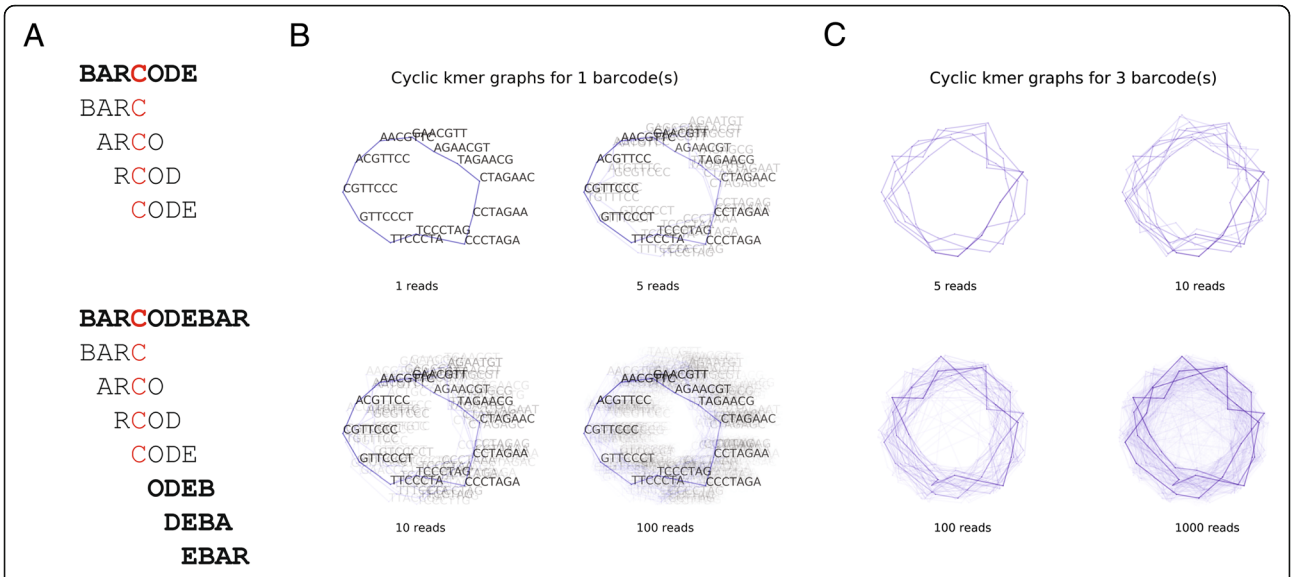

Fig. 1 A strategy to use k-mer counting to identify sequence barcodes. **a** Circularizing barcodes ensures robustness against single mismatches. An example sequence 'BARCODE' contains an error (highlighted in red). When the barcode sequence is short relative to k, all k-mers from this sequence will contain the mutated base. Circularizing the sequence (bottom) ensures that there will be some error-free k-mers from a sequence independent of the position of the error. **b** An example circular k-mer graph containing one barcode. Error-containing reads were simulated from a ground-truth barcode. Reads were circularized and k-mers were counted. The resultant k-mer graph is plotted here. Nodes in this graph are represented as gray dots, and edges as blue lines. Edges weights are represented by shading (dark = high edge weight). Despite a fairly high rate of error (Poisson 3 errors per 12 nucleotide barcode), the true barcode path is visually discernable with a modest number of reads. **c** An example circular k-mer graph containing three barcodes. Same as above

barcodes that contain frequently observed k-mers, while low-weight paths likely represent cycles that formed spuriously. This is depicted in Fig. 1b and c. We emphasize here that we do not need any single read to contain all k-mers in a high-weight path / error-corrected barcode; it is the overlap of many k-mers that likely originate from a number of reads that gives rise to such a path.

To rapidly identify cyclic paths from this graph we use a greedy depth-first recursive search **(Algorithm 1a)**. Briefly, this algorithm works by first (randomly) picking a node from the graph to initialize the search. Each of the outgoing edges that connect to this node is checked, in descending order of the edge weights. This step is repeated for each of the children nodes, for a fixed number of steps given by the length of the barcode (a user-supplied parameter). If at the end of these steps the procedure returns to the same node where it began, a cycle has been found. We use a similar procedure to identify multiple cycles within a graph. After the first cycle has been found we decrement the edge weights of all edges in that cycle by its capacity (the weight of the lowest weight edge in that cycle). This has the effect of removing one edge from the cycle (thereby breaking it) while removing any contribution that cycle had to any other edges. We then repeat the procedure in **Algorithm 1a** until there are no more cycles present in the subgraph. We then repeat this process, starting from a new node, and new nodes are selected from the common k-mers in the dataset. This is described in some more detail in **Algorithms 1a** and **1b.**

This approach identifies several cyclic paths from the barcode de Bruijn graph, and the depth of this search is determined by user-supplied parameters. As only a subset of these paths represents a true error-corrected barcode sequence, we filter the paths based on their path weight. We hypothesized that paths representing a true error-corrected barcodes would have a higher capacity than paths that contained errors, or paths formed by spurious k-mer overlap between [barcode-wise] unrelated sequences. To verify this hypothesis we plotted the cumulative distribution of path capacities, and observed a clear inflection point, corresponding to a subset of paths that had a significantly higher capacity than the rest of the population. We computationally identified this inflection point as a local maximum in the first derivative of the cumulative distribution function. This was facilitated by first smoothing the CDF.

Paths are then thresholded at this inflection point, and paths with capacities higher than the threshold value are deemed error-corrected barcodes, while the rest of the paths are rejected. We then assign each read in our dataset to one of the error-corrected barcodes based on either k-mer compatibility or Levenshtein distance. When assigning a read by k-mer compatibility, a read is assigned to the consensus barcode with which it shares the most k-mers. A read is only assigned to a consensus barcode if it shares a minimum number of k-mers with it (a user-specified parameter). With this protocol, we can vary the length of k (a user parameter) to affect the

output data in a predictable manner. Assigning reads using larger k than that used to assign reads enables us to call error-free barcodes with the higher stringency, while assigning a large number of reads **(Algorithm 2)**. In general, in order for this approach to use information contained in reads with one or more errors, we require that k be smaller than the length of the barcode. Additionally reads with two or more errors can sometimes contribute error-free k-mers (depending on the relative spacing of these errors in the circularized sequence), and the frequency with which this occurs increases as k decreases. However, the number of nodes in the de Bruijn graph is also reduced as k decreases leading to an increase in spurious edges between nodes, obfuscating the cyclic paths that represent true barcodes. As such, for a typical drop-seq barcode of length 12 we recommend using a value of k that is at least 8 nucleotides in length.

Finally, to improve performance we make a small modification to the protocol outlined above. Rather than building a de Bruijn graph of the entire barcode dataset, we instead build a new subgraph for each new node we initialize the search with. This subgraph is built by first extracting all the reads which contain the start k-mer (in any of its circular permutations), and then building a de Bruijn graph from these reads. As the subgraph was built only from reads that contain the start node, this subgraph only contains nodes that are indirectly connected (within a fixed number of steps) to the start node. This substantially simplifies the search procedure while leaving performance unaffected. To rapidly build these subgraphs we prepare a k-mer index of the input dataset, which maps a k-mer to a list of reads that contains that k-mer. When performing a search from a random start node, we query the k-mer index for the start node and prepare a de Bruijn graph from only the subset of reads returned by the query.

As this index can be quite large (for Drop-seq, which uses 12mer barcodes, each read produces 36 circularized and truncated / extended k-mers to be indexed), which results in an extremely large index. We further simplify this protocol by preparing the index from a subset of the reads. This approximation also does not affect performance, as long as the subset is representative of the entire dataset. We can identify when we have obtained a representative sampling of the data when the k-mer counts distribution has become stable. The exact parameters for this depend on the sequencing depth, number of barcodes, error rate and likely other parameters; however in our tests simply indexing ~ 0.5 m reads is sufficient (Table 1).

Results

To benchmark our algorithms' performance and establish it's performance limits, we performed a large number of simulations, under a wide variety of scenarios. We also compared Sircel's performance against a naïve pipeline based on simple k-mer counting that is not be able to handle insertion / deletion errors. We produced a fixed number of 'true' barcodes, and produced reads by adding a Poisson number of errors to each read. Error positions were selected uniformly at random, and separate datasets were produced for insertion, deletion, mismatch, and all errors. We also varied the barcode abundance distributions between normal, uniform and exponential. For each condition we produced 3 separate datasets and evaluated our algorithms' performance on each. Finally, we evaluate the effect of assigning reads to barcodes using either kmer compatability and Levenshtein distance.

As shown in Additional file 1: Figure S1, our algorithm is able to identify the error-free barcode sequences independent of the number of Poisson errors per read. However we do see a dependence on the specific error type: mismatch errors are better tolerated than insertion or deletion errors. Additionally we find that the barcode abundance distribution strongly affects our ability to detect and error correct barcodes as the error rate increases, with normally distributed barcode abundances being far easier to handle than exponentially distributed abundances. We also used our simulations to evaluate how well we could assign reads to error-corrected barcodes (Additional file 1: Figure S2). Here we see a similar trend with exponentially distributed barcode abundances causing a higher rate of incorrect read assignment with both the naïve pipeline and our approach. Importantly however our algorithm outperforms the naïve approach in these circumstances. In these experiments we observe that using Levenshtein distance to assign reads to consensus barcodes results in a higher rate of correct read assignment. However, these two approaches differ significantly when evaluating the number of unassigned reads for each of the workflows (Additional file 1: Figure S3). Together these results show that assigning reads based on Levenshtein distance results in fewer reads being incorrectly assigned, but a higher number of reads being unassigned to any single barcode. This conservative approach to barcode assignment is likely more useful in a real sc-RNA-seq dataset, and as such is the default behavior of the program.

We next validated our approach on real data, by attempting to identify and error-correct barcode sequences in multiple real datasets. We re-analyzed a previously published species-mixing Drop-Seq experiment

Table 1 Run time for downsampled Macosko et al., datasets

Number of reads in dataset	Number of cells detected	Time
1,000,000	562	6 m 39 s
10,000,000	575	51 m 08 s
100,000,000	574	360 m 31 s

published by Macosko et al., as well as a similar experiment from the unrelated SeqWell protocol. Although the methods differ, both experiments involved single-cell sequencing of a mixture of human and mouse cells, and as such it served as a useful control for barcode calling: if the calling performs well, cells should only contain human, or mouse reads but not both. We used our algorithm on this data and as expected found a clear inflection point in the cumulative distribution of barcode paths. We could readily identify this inflection by its smoothed first derivative, and thresholding the paths at this inflection point yielded 582 barcodes, each of which had accounted for approximately the same number of reads (Fig. 2a and b). These values were consistent with previously reported values from the same dataset. We then quantified single-cell expression profiles using combined human/mouse transcriptome, once again using an algorithm derived from k-mer counting [kallisto]. As seen in Fig. 2c, 'cells' that represent collections of reads clustered by similar barcode k-mers have an fairly even number of reads, as is expected from a single-cell RNA seq experiment. Additionally, these cells exhibit distinct expression profiles, and in nearly every case cells appear to consist of reads deriving entirely from one species (Fig. 2d and Additional file 1: Figure S5). This result indicates that our k-mer counting approach can be used to group reads into single-cell datasets. Finally, we note that deletion errors do pose a real problem in these datasets. To evaluate the extent to which deletion errors in the split-pool bead barcode synthesis protocol affect these datasets we calculated either the Levenshtein distance or the Hamming distance between a consensus barcode and each read that is assigned to it (Additional file 1: Figure S4). We see that the Hamming distance is systematically higher than the Levenshtein distance indicating that at least some insertion/deletion errors are present in and affect the majority of cells in a Drop-seq experiment.

Discussion

We have shown how a de Bruijn graph formulation of the barcode calling problem based on circularization of input sequences is a useful approach to identify and error-correct barcode sequences. Our approach simplifies the problem of sequence error correction by rephrasing it as a k-mer counting question, and as such is simple and relatively fast. Furthermore it does not rely heavily on user-supplied parameters or any prior knowledge about the exact nature of the sequencing errors; as such we expect it to be applicable to a number of different single-cell barcoding techniques that differ in the exact nature of the barcode generation chemistry. We also show that our approach produces usable data from real-world datasets, and that our integrated pipeline using kallisto and transcript compatibility counts is an effective approach for rapid and accurate analysis of Drop-Seq single-cell RNA-Seq data.

We benchmarked our algorithm using an extensive set of simulations that systematically varied the error rate per read, the error type, and the abundance of each barcode [single-cell] within the dataset. From these simulations we observe that the barcode abundance distribution makes a significant difference to performance, with normally- and uniformly- distributed barcode abundances being far better tolerated than exponentially distributed barcodes. This behavior is expected; with exponentially distributed barcode abundances, the inflection point in the CDF of cyclic path weights is obscured, making it difficult to distinguish between a true barcode path with low abundance, and an error-containing path with relatively high weight. Notably, exponentially distributed barcode abundances are not expected (and indeed not observed) in real data: the total RNA content from any given single cell in an experiment are typically approximately uniformly distributed.

These simulations also demonstrated that although our method is tolerant of errors when identifying and error-correcting barcode sequences, errors lower its ability to assign individual reads to error-corrected consensus sequences. This is not surprising, because reads with a large number of errors are unlikely to contain any error-free k-mers that are required to assign a read. Simulations also revealed that our algorithm is more tolerant to mismatch errors over insertion or deletion errors. We postulate that this is because in the barcode de Bruijn graph, reads that contain only mismatches form a cyclic path of the correct length, whereas reads containing insertion or deletion errors form paths with incorrect length, complicating the cyclic-path search protocol. This effect is most pronounced at the error rates that are higher than typical Drop-seq datasets.

Conclusions

Single-cell genomics is a dynamic field that encompasses a large and growing number of techniques that measure a variety of biological properties. However one commonality in these workflows is that experiments mark reads originating from distinct cells with single cell barcodes. Correctly identifying and grouping reads by their barcodes in the presence of experimental and sequencing errors is an essential first step in any single-cell analysis pipeline. The software presented here addresses this universal problem, and as such it should be useful for a variety of single cell sequencing based genomics experiments. Our approach presented here uses the novel concept of kmer-circularzation, which enables the fast and efficient operation of k-mer counting to be expanded to problems that potentially include insertions and deletions. Although we have not

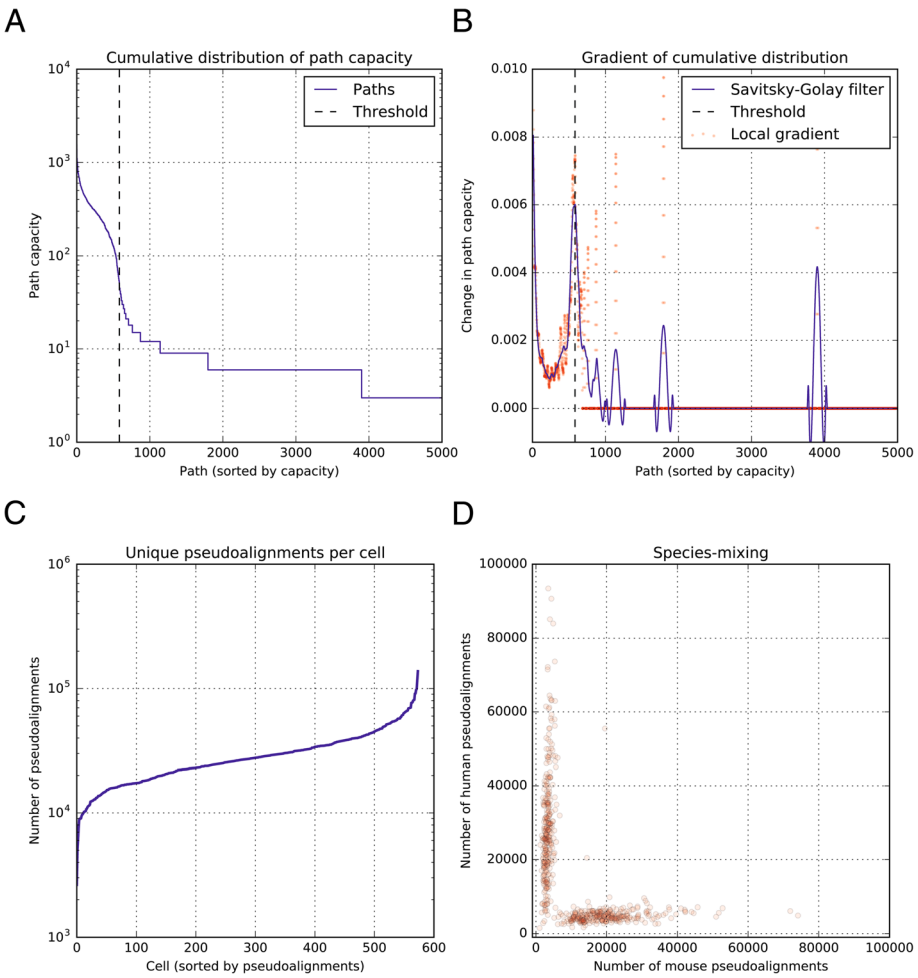

Fig. 2 Identifying barcodes and splitting reads from Macosko et al., species mixing experiment. **a** Circular paths were identified in the circular barcode k-mer graph from a published Drop-seq dataset. The distribution of circular path weights versus path rank clearly shows an inflection point. Paths with weight higher than this inflection point are deemed to be true barcodes. **b** This inflection point can be identified as a local maximum in the first derivative of the path-weight distribution. A Savitskiy Golay filter facilitates in this identification by smoothing the data. **c** Reads were grouped into cells by assigning them to to thresholded paths based on k-mer compatibility alone. This assignment results in a flat distribution in the number of pseudoalignments per cell. **d** Reads that were split based on barcode k-mer compatibility alone also segregate by their number of pseudoalignments to different transcriptiomes. This indicates that assigning reads based on k-mer compatibility produces distinct and biologically relevant groupings

explored this here, we believe this approach might therefore have applications in other areas of genomics; one potential application might be to use k-mer circulariation to obtain k-mer fingerprints of metageomic datasets in an indel sensitive manner.

Finally although we focus here on the specific Drop-seq protocol, there are a number of related single-cell experiments that rely on split-pool combinatorial synthesis of barcodes [14], as well as other massively parallel single-cell sequencing experiments that measure other genomic and transcriptomic properties [4, 15]. As error correcting and clustering barcodes is central to these assays as well, we believe that these methods will also benefit from our software.

Availability and requirements

Project name: Sircel

Project home page: https://github.com/pachterlab/sircel

Operating system: platform independent

Programming language: python3

Other requirements:

python3 (version 3.5 or higher),

numpy,

scipy,

scikit-learn,

Redis (https://redis.io/)

License: MIT

Any restrictions to use by non-academics: no

Example

The example data set (supplementary data) shows the workflow to identify and split barcoded reads from a published Drop-seq dataset [SRR1873277]. This dataset derives from a species-mixing experiment, where human and mouse cells were mixed prior to single-cell RNA-seq. As such reads grouped by their barcodes should also segregate by which species they [pseudo] align with. We can therefore evaluate the performance of Sircel by how frequently reads from the two species appear to derive from the same cell.

Methods

Raw sequencing data for Drop-seq and Seqwell species mixing was obtained from the sequence read archive (SRR1873277 and SRR5250839 respectively) and converted to fastq format using SRA-toolkit. Subsamples of these datasets were obtained using standard command line tools:

```
zcat INFILE.fastq.gz | head –n NUM_READS*4 |
gzip > OUTFILE.fastq.gz
```

We used this data without any further processing, or read filtering. Sircel was then used to identify barcodes and assign reads with the following parameters: k-mer length of 7, search breadth of 1000 subgraphs, search depth of 5 paths per subgraph. All results presented here were processed with 32 threads.Output from Sircel was then fed, into a single-cell analysis pipeline based on kallisto (N. L. [1]) and transcript compatibility counts [11]. Our integrated pipeline, as well as ipython notebooks to visualize the data is available on Github.

Simulations were performed by first randomly generating a 500 ground truth barcode sequences of length 12. Each barcode was assigned a relative abundance drawn from one of three pre-defined distributions (normal, uniform and exponential). Reads were generating by selecting a barcode according to the barcode abundance, and adding a Poisson number of errors given by user-defined rate. Error type (insertion, deletion, mismatch or any) was also varied systematically during this step. Each simulation consisted of 100,000 reads generated in this manner. For each condition (combination of barcode abundance distribution, Poisson error rate and error type), we produced three separate simulations for a total of 180 datasets. The Poisson error rate was varied between 0 and 3 errors per read. The number of cells in each simulation was generated from a normal distribution with mean of 500 and standard deviation of 50. Exponentially distributed cell abundances were generated with scale parameter 0.2, and normally distributed cell abundances were generated with a mean of 200 reads per cell, and standard deviation of 20 reads per cell.

Our naïve pipeline was based on that of [11]. It was implemented in python.

These simulated datasets were then fed into Sircel to identify error-free barcodes. For each simulation we compared the output of Sircel to the ground-truth barcodes, identifying true positives as barcode sequences that were present in both the Sircel output and the ground-truth, false positives as barcode sequences that were found in the Sircel output but not the ground truth, and false negatives as barcode sequences that were not found in the Sircel output but not the ground truth. For each true positive barcode identified by Sircel, we additionally evaluated whether the reads assigned to that barcode were correctly assigned. Reads that derived from the ground truth barcode were deemed correctly assigned, and all other reads were labeled as incorrectly assigned.

Ipython notebooks to reproduce this analysis are available on Github.

Algorithm 1a. Recursively identify a single cycle of fixed length in graph

1. Initialize recursion:
    a. Pick a starting edge by edge weight
        i. Edge links node and neighbor
            1. Set the current path to the starting edge
            2. Record the identity of the starting node
2. Recursion:
    a. Get all outgoing edges that emanate from neighbor
    b. Sort outgoing edges by edge weight (descending)
    c. For each outgoing edge
        i. Extend the current path by this edge
        ii. Continue recursion
3. Terminate recursion:
    a. If the current path length is longer than the barcode length
        i. If the path start node and end node are the same, the path is a cycle
            1. Return True
        ii. Else return False

Algorithm 1b. Identify multiple cycles of fixed length within the same graph

1. Initialize
    a. Pick a starting edge by edge weight
2. Repeat the following
    a. Identify a single cycle (see above)
    b. Decrement all edge weights in that cycle by the capacity of that cycle
        i. Lowers the weight of the limiting edge to 0
        ii. All other edges will have some smaller positive value for edge weight
3. Terminate loop when there are no more cycles in the graph

Algorithm 2. Identify barcodes and assign reads

1. Index k-mers
    a. Build a look-up table associating each k-mer with a set of reads that contains that k-mer
2. Identify barcodes
    a. Pick a starting edge (k-mer) in order by edge weight
    b. Build a de Bruijn sub-graph. This is a directed, weighted, de Bruijn graph containing only the subset of reads that contain the starting k-mer
        i. Building only a subgraph greatly speeds up graph traversal
    c. Identify high-weight cycles
        i. See Algorithm 1.
        ii. Remove this cycle from the subgraph by decrementing the weights of all edges in this subgraph by the capacity of this path
    d. Repeat c to find other paths that originate from this node
    e. Repeat a–e to find paths that start at a different node
3. Merge similar paths by Hamming distance
4. Threshold paths to identify true barcodes
    a. Identify an inflection point in the cumulative distribution of path lengths
5. Assign reads
    a. Assign each read to the path with which it shares the most k-mers

Additional file

Additional file 1: Figure S1. Sircel can robustly identify the number of cells present in a dataset. We performed several simulations with error-prone reads. The number of errors per read, the type of errors, and the distribution of barcode abundances were all systematically varied. Performance was compared over three pipelines: a naïve approach (grey), using Sircel and k-mers (red) and using Sircel and Levenshtein distance (cyan). A. Any errors. B. Deletions. C. Insertions. D. Mismatches. **Figure S2**. Assigning reads to consensus barcodes depends on errors rate and barcode abundance distribution. Using the same simulations as before, the fraction of reads that were correctly assigned in each cell was quantified. We find that Sircel using Levenshtein distance performs the best. A. Any errors. B. Deletions. C. Insertions. D. Mismatches. **Figure S3**. Assigning reads to consensus barcodes by k-mer compatibility depends on errors rate and barcode abundance distribution. In the same simulations, the fraction of reads that could not be unambigiously assigned in each cell was quantified. A. Any errors. B. Deletions. C. Insertions. D. Mismatches. **Figure S4**. Indel errors are present in real data. We separated barcodes in a species mixing experiment from Seqwell (SRR5250839), and evaluated the Hamming and Levenshtein distances between each read and its consensus barcode. We find that Hamming distance is systematically larger than Levenshtein distance, indicating that the data contains indels. **Figure S5**. Species mixing with Seqwell data. We separated barcodes in a species mixing experiment from Seqwell and evaluated our ability to split reads by species. **Figure S6**. Circularized de Bruijn graph from real data. A de Bruijn subgraph was prepared from circularized reads that could be assigned assigned to 10 randomly selected barcodes from the Macosko et al. dataset is depicted here. Line transparency is proportional to the weight of each edge. (PDF 2486 kb)

Acknowledgements
We thank Jase Gehring and Vasilis Ntranos for helpful comments and feedback during the development of the method.

Funding
None.

Availability of data and materials
The datasets analyzed here were obtained from previously published datasets, which are available at the NCBI Sequence Read Archive. SRA ascension numbers used in this paper are SRR1873277 and SRR5250839.

Authors' contributions
AT and LP conceived of the project. AT wrote the software and analyzed data. AT and LP wrote the manuscript. All authors read and approved the final manuscript.

Ethics approval
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details
[1]Division of Biology and Biological Engineering, California Institute of Technology, 116 Kerckhoff Laboratory, Pasadena, CA 91125, USA. [2]Departments of Biology and Computing & Mathematical Sciences, California Institute of Technology, 116 Kerckhoff Laboratory, Pasadena, CA 91125, USA.

Received: 23 May 2017 Accepted: 7 January 2019
Published online: 17 January 2019

References
1. Bray NL, Pimentel H, Melsted P, Pachter L. Near-optimal probabilistic RNA-seq quantification. Nat Biotechnol. 2016;34(5):525–7 https://doi.org/10.1038/nbt.3519.
2. Compeau PEC, Pevzner PA, Tesler G. How to apply de Bruijn graphs to genome assembly. Nat Biotechnol. 2011;29(11):987–91 https://doi.org/10.1038/nbt.2023.
3. Fincher CT, Wurtzel O, de Hoog T, Kravarik KM, Reddien PW. Cell type transcriptome atlas for the planarian Schmidtea mediterranea. Science. 2018;360(6391):eaaq1736–14 https://doi.org/10.1126/science.aaq1736.
4. Gierahn TM, Wadsworth MH, Hughes TK, Bryson BD, Butler A, Satija R, et al. Seq-well: portable, low-cost RNA sequencing of single cells at high throughput. Nat Methods. 2017;14(4):395–8 https://doi.org/10.1038/nmeth.4179.
5. Hunt M. Circlator: automated circularization of genome assemblies using long sequencing reads. Genome Biol. 2015:1–10 https://doi.org/10.1186/s13059-015-0849-0.
6. Karaiskos N, Wahle P, Alles J, Boltengagen A, Ayoub S, Kipar C, et al. The Drosophila embryo at single-cell transcriptome resolution. Science. 2017;358(6360):194–9 https://doi.org/10.1126/science.aan3235.
7. Klein AM, Mazutis L, Akartuna I, Tallapragada N, Veres A, Li V, et al. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. Cell. 2015;161(5):1187–201 https://doi.org/10.1016/j.cell.2015.04.044.
8. Li H. BFC: correcting Illumina sequencing errors. Bioinformatics. 2015:1–3 https://doi.org/10.1093/bioinformatics/btv290/-/DC1.
9. Liu Y, Schroder J, Schmidt B. Musket: a multistage k-mer spectrum-based error corrector for Illumina sequence data. Bioinformatics. 2013;29(3):308–15 https://doi.org/10.1093/bioinformatics/bts690.
10. Macosko EZ, Basu A, Satija R, Nemesh J, Shekhar K, Goldman M, et al. Highly parallel genome-wide expression profiling of individual cells using Nanoliter droplets. Cell. 2015;161(5):1202–14 https://doi.org/10.1016/j.cell.2015.05.002.
11. Ntranos V, Kamath G, Zhang JM, Pachter L, Tse DN. Fast and accurate single-cell RNA-seq analysis by clustering of transcript-compatibility counts. Genome Biol. 2016:1–14 https://doi.org/10.1186/s13059-016-0970-8.
12. Patro R, Mount SM, Kingsford C. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. Nat Biotechnol. 2014;32(5):462–4 https://doi.org/10.1038/nbt.2862.
13. Plass M, Solana J, Wolf FA, Ayoub S, Misios A, Glažar P, et al. Cell type atlas and lineage tree of a whole complex animal by single-cell transcriptomics. Science. 2018;360(6391):eaaq1723–12 https://doi.org/10.1126/science.aaq1723.
14. Rosenberg AB, Roco C, Muscat RA, Kuchina A, Mukherjee S, Chen W, et al. Scaling single cell transcriptomics through split pool barcoding; 2017. p. 1–13. https://doi.org/10.1101/105163
15. Rotem A, Ram O, Shoresh N, Sperling RA, Goren A, Weitz DA, Bernstein BE. Single-cell ChIP-seq reveals cell subpopulations defined by chromatin state. Nat Biotechnol. 2015:1–11 https://doi.org/10.1038/nbt.3383.
16. Saunders A, Macosko E, Wysoker A, Goldman M, Krienen F, de Rivera H, et al. A single-cell atlas of cell types, states, and other transcriptional patterns from nine regions of the adult mouse. Brain. 2018:1–27 https://doi.org/10.1101/299081.
17. Schaeffer L, Pimentel H, Bray N, Mellsted P, Pachter L. Pseudoalignment for metagenomic read assignment. Arxiv Preprint; 2015. p. 1–13.
18. Skums P, Dimitrova Z, Campo DS, Vaughan G, Rossi L, Forbi JC, et al. Efficient error correction for next-generation sequencing of viral amplicons. BMC Bioinformatics. 2012:1–13 https://doi.org/10.1186/1471-2105-13-S10-S6.
19. Stephenson W, Donlin LT, Butler A, Rozo C, Bracken B, Rashidfarrokhi A, et al. Single-cell RNA-seq of rheumatoid arthritis synovial tissue using low-cost microfluidic instrumentation. Nat Commun. 2018:1–10 https://doi.org/10.1038/s41467-017-02659-x.
20. Svensson V, Natarajan KN, Ly L-H, Miragaia RJ, Labalette C, Macaulay IC, et al. Power analysis of single-cell RNA-sequencing experiments. Nat Methods. 2017;14(4):381–7 https://doi.org/10.1038/nmeth.4220.
21. Tosches MA, Yamawaki TM, Naumann RK, Jacobi AA, Tushev G, Laurent G. Evolution of pallium, hippocampus, and cortical cell types revealed by single-cell transcriptomics in reptiles. Science. 2018;360(6391):881–8 https://doi.org/10.1126/science.aar4237.
22. Trapnell C. Defining cell types and states with single-cell genomics. Genome Res. 2015;25(10):1491–8 https://doi.org/10.1101/gr.190595.115.

23. Zhang Z, Wang W. RNA-skim: a rapid method for RNA-Seq quantification at transcript level. Bioinformatics. 2014;30(12):i283–92 https://doi.org/10.1093/bioinformatics/btu288.
24. Zorita E, Cuscó P, Filion GJ. Starcode: sequence clustering based on all-pairs search. Bioinformatics. 2015;31(12):1913–9 https://doi.org/10.1093/bioinformatics/btv053.