`Alevin-fry-atac` enables rapid and memory frugal mapping of single-cell ATAC-seq data using virtual colors for accurate genomic pseudoalignment

Noor Pratap Singh[1] [0000−0003−3721−2157], Jamshed Khan[1] [0000−0002−5129−9749], and Rob Patro[1] [0000−0001−8463−1675]

Department of Computer Science, University of Maryland - College Park
`npsingh@cs.umd.edu` `jamshed@cs.umd.edu` `rob@cs.umd.edu`

Abstract. Ultrafast mapping of short reads to transcriptomic and metagenomic references via lightweight mapping techniques such as pseudoalignment has demonstrated success in substantially accelerating several types of analyses without much loss in accuracy compared to alignment-based approaches. The application of pseudoalignment to large reference sequences — like the genome — is, however, not trivial, due to the large size of the references or "targets" (i.e. chromosomes) and the presence of repetitive sequences within an individual reference sequence. This can lead to multiple matching locations for a k-mer within a single reference, which in turn can lead to false positive mappings and incorrect reference assignments for a read when the colors across the k-mer matches for a read are aggregated. Even when the read is determined to map to the appropriate reference, the increased occurrence of k-mer multi-matches within a reference can prevent the determination of the correct approximate position of the read, which is often critical in applications that map short reads to the genome.

We propose a new and modified pseudoalignment scheme that partitions each reference into "virtual colors". These are essentially overlapping bins of fixed maximal extent on the reference sequences that are treated as distinct "colors" from the perspective of the pseudoalignment algorithm. A mapped k-mer is assigned a virtual color id that encodes the combination of the reference and within-reference bin in which the k-mer occurs. When the k-mers across a read are aggregated, the intersection is performed on virtual colors instead of the original colors (references), to determine the compatible set of targets (bins). The virtual colors can then be mapped back to the original references to provide the final mappings. The projection of the original reference sequences into virtual color space, and the corresponding modifications to the pseudoalignment procedure, can be applied dynamically at program invocation and without any modification of the underlying index itself. This makes the setting and modification of instance-appropriate parameters efficient and straightforward and the approach widely applicable.

We apply this modified pseudoalignment procedure to process and map single-cell ATAC-seq data in our new tool `alevin-fry-atac`. We compare `alevin-fry-atac` to both `Chromap` and `Cell Ranger ATAC`. `Alevin-fry-atac` is highly scalable and, when using 32 threads, is approximately 1.78 times faster than `Chromap` (the second fastest approach) while using approximately 3 times less memory and mapping slightly more reads. The resulting peaks and clusters generated from `alevin-fry-atac` show high concordance with those obtained from both `Chromap` and the `Cell Ranger ATAC` pipeline, demonstrating that virtual color-enhanced pseudoalignment directly to the genome provides a fast, memory-frugal, and accurate alternative to existing approaches for single-cell ATAC-seq processing. The development of `alevin-fry-atac` brings single-cell ATAC-seq processing into a unified ecosystem with single-cell RNA-seq processing (via `alevin-fry`) to work toward providing a truly open alternative to many of the varied capabilities of `CellRanger`. Furthermore, our modified pseudoalignment approach should be easily applicable and extendable to other genome-centric mapping-based tasks and modalities such as standard DNA-seq, DNase-seq, Chip-seq and Hi-C.

Keywords: single-cell ATAC-seq · pseudoalignment · virtual colors · genome mapping.

1 Introduction

ATAC-seq [7] is a widely-used assay that enables the profiling of open-chomatin regions within the genome. It has a wide variety of applications, such as providing insight into gene regulation through the identification of promoters and enhancers [38], studying cell differentiation and development [30,21], characterizing disease mechanisms including cancer [9,42], and identification of transcription factors [34] to name a few. As with RNA-seq, it is also now possible to profile chromatin at the level of single cells [8,10], with the number of such samples being generated growing each year. However, the efficient mapping and processing of single-cell ATAC-seq is a computationally-challenging task.

There are currently three main pipelines/methods used for this task. The first one involves aligning reads to the genome using methods such as `Bowtie2` [20], `BWA-MEM` [23], or `HISAT2` [19], followed by demultiplexing into cells and then finally using existing general-purpose programs like `SAMtools` [24] and `Picard` [40] to sort and filter the alignments. The entire process in the pipeline can be quite slow when working with datasets that contain a large number of reads. The mapping process can be sped up to some extent by replacing certain parts of the pipeline (e.g. replacing one of the previously-mentioned alignment tools with `minimap2` [22] as is used in workflows such as `MAESTRO`[41]).

The second major approach is `Cell Ranger ATAC`, which is developed by 10x genomics and performs end-to-end analysis; from read mapping through downstream tasks such as clustering of cells using the called peaks and counts. While this can be substantially faster than the types of pipelines mentioned above, as the components are optimized to work together, and to avoid redundant computation and unnecessary file input and output, there is still much potential for further improvement. Further, `Cell Ranger ATAC` is released under a non-free license, that permits any use or modification with only 10x genomics' own proprietary and patented technologies.

The third, and currently the fastest method is `Chromap` [44]. `Chromap` uses a minimizer-based [31] index and maps the reads by creating candidate anchors based on minimizer hits, subsequently aligning candidate positions against the reference. To further speed up the process, it creates a cache that exploits the property that the reads belonging to peak regions occur frequently, often resulting in the observation of exactly repeated minimizer chains. In addition, `Chromap` also supports other data modalities such as Chip-seq and Hi-C.

However, while certain approaches, like `scATAK` and `snATAK` [4], apply lightweight mapping methods like pseudoalignment to single-cell and single-nucleus ATAC-seq data, they still first rely on traditional read alignment tools upstream to align reads to the genome to produce the accurate genomic mappings necessary to extract peaks, whose sequences are then treated as experiment-specific targets and remapped using pseudoalignment [5]. While lightweight mapping approaches have proven very successful in providing fast read mapping in transcriptomics [5,28] and metagenomics [33,36,25,1,14] without much loss in accuracy compared to alignment-based approaches, no method has yet employed lightweight mapping directly to the genome in the context of single-cell ATAC-seq.

One may expect that such approaches might also help in improving the read mapping efficiency for genomic mapping as well. However, the large reference sequences frequently consisting of repetitive regions make applying pseudoalignment directly to large reference genomes difficult. A k-mer can map to many locations within the same reference which may lead to an incorrect determination of the mapping position, even when the specific reference (e.g. chromosome of origin) is determined correctly. These issues can lead both to reporting spurious mapping positions, as well as missing the most likely approximate mapping location of a read and can also lead to exacerbated multimapping effects.

Recently, the `KMCP` [35] described an approach that uses the a modified COBs [3] index to tackle the metagenomic profiling problem. The COBs data structure is an (approximate) presence/absence index based on the matching of some threshold fraction of k-mers between a query and the reference. To reduce the potential for spurious matching based off of genomically distant k-mers, and to ensure that the evidence supporting the existence and abundance of a taxon isn't due to many reads assigned to only one narrow range of the underlying genome (akin to the strategy employed in `KrakenUniq` [6]), `KMC` divides the metagenomic references into a fixed number of chunks, which themselves (rather than the genomes) become the set of distinct targets represented in the COBs index. Taking the success of such an approach as motivation, we propose a modified pseudoalignment approach for mapping reads to the genome, which partitions the references into "virtual colors" of fixed maximum extent. To determine the set of references (and their approximate positions) from which a read originates, the intersection across the virtual colors to which the k-mers of a read map is performed instead of taking the intersection over the original colors (i.e. references/chromosomes). This also necessitates careful handling of the boundaries between virtual colors (to ensure reads spanning virtual colors are not lost) as well as ensuring that duplicate mappings induced by virtual color overlap are properly identified with only a single position in the original reference space.

Based on this approach, we also introduce our new tool `alevin-fry-atac`, for processing single-cell ATAC-seq data that integrates the memory frugal `piscem` [13] index along with the modified pseudoalignment algorithm for mapping reads. This enables `alevin-fry-atac` to map reads efficiently while keeping a low memory footprint. `Alevin-fry-atac` after mapping allows for cell-barcode correction and deduplication to produce a BED file that contains the reference name, the start and end positions of the mapped fragment along with their cell barcode and the number of duplicates. This BED file can then be used with peak callers such as `MACS2` 2 [45,15] and used by other downstream tools to analyze single-cell ATAC-seq data.

We evaluated `alevin-fry-atac` on both simulated and experimental datasets and it achieved an accuracy and mapping rate comparable to the other alignment-based methods. We also observed a high concordance between the peaks and clusters generated on the mapped fragment file produced by the other methods to those generated on the mapped fragment file produced by `alevin-fry-atac`. `Alevin-fry-atac` uses 300% less memory compared to `Chromap` while taking 178% less time when mapping the reads. We have previously developed `alevin-fry` to enable efficient preprocessing of single-cell RNA-seq. With the development of `alevin-fry-atac`, we provide a unified open-source framework that can help to ease and facilitate multimodal analysis.

2 Methods

2.1 Preliminaries

The `piscem` [13] index is a modular index which enables efficient retrieval of a set of references (colors), as well as all of the associated positions and orientations for a given k-mer x. Like the `pufferfish` [2] index that preceded it, the `piscem` index is constructed over the compacted colored reference de Bruijn graph, permitting the same set of operations. However, the `piscem` index is substantially smaller than that of `pufferfish`. For a given set of reference sequences, `piscem` uses `cuttlefish` [18] to efficiently construct the compacted colored reference de Bruijn graph and the associated tiling of the reference by unitigs. It then uses `SSHash` [29] to create a k-mer to unitig mapping $\mathcal{K} \rightarrow \mathcal{U}$, which maps each k-mer to a unitig, relative offset, orientation triplet (u_k, p_k, o_k). `piscem` builds a dense inverted index to represent the unitig-to-reference tiling $\mathcal{U} \rightarrow \mathcal{R}$, which allows recalling, for each unitig the reference sequences, and relative positions and orientations of the unitig's occurrences in the original reference. The k-mer to unitig $\mathcal{K} \rightarrow \mathcal{U}$ and unitig-to-reference $\mathcal{U} \rightarrow \mathcal{R}$ can be combined to correctly and efficiently determine the mapping position (and orientation) of a k-mer in the reference, which might be essential for genome-centric mapping tasks. The `piscem` index can be constructed on different types of reference sequences(genome, transcriptome, metagenome, etc.) and the associated software can currently support the processing and mapping of single-cell RNA-seq and bulk RNA-seq. With the introduction of `alevin-fry-atac`, single-cell ATAC-seq is now also supported. A complete manuscript describing the `piscem` index, the key details of the software implementation, and its applications to various indexing tasks is currently under preparation.

Pseudoalignment is a lightweight mapping procedure, which aims to assess the "compatibility" between a read and a collection or references without specifically finding the coordinates of each base of the read in the reference. The term was originally coined in the context of RNA-seq quantification [5] where the transcripts to which a read maps were found by doing an intersection across the colors (transcripts) mapped by a set of selected k-mers of a read. Certain characteristics of this mapping process, however, predate the coining ([43], [46]). Several approaches and algorithms for pseudoalignment have been proposed, that vary from how and which set of k-mers of a read is selected for mapping, to how the colors corresponding to the selected k-mers are aggregated to get the final color set denoting the references to which a read maps [1,14].

The pseudoalignment strategy that has been employed in the current manuscript is the "hybrid" method described in `Themisto` [1] or the "threshold-union" method described in `Fulgor` [14]. Let $\mathcal{R} = \{R_1, R_2, ..., R_N\}$ denote the set of reference sequences we want to map a query sequence Q. Let $K(Q)$ denote the set of k-mers for which a mapping $\mathcal{K} \rightarrow \mathcal{U}$ exists in the index. Given the threshold parameter $\tau \in (0, 1]$, the output of pseudoalignment for a read is a subset $R_C \subset \mathcal{R}$ and the relative read mapping position within a reference, such that each reference in the output R_C appears as a mapped hit (i.e. k-mer match) across at least $\tau * K(Q)$ k-mers. The mapping position of the leftmost k-mer hit of Q to the reference is assigned as the mapping position of the read to reference.

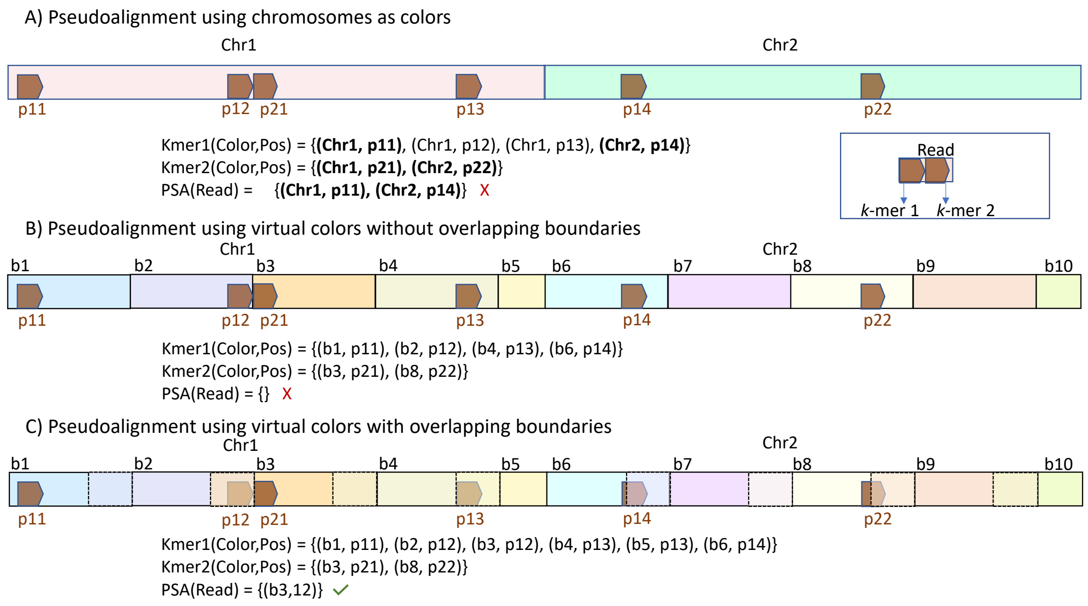
A) Pseudoalignment using chromosomes as colors
Chr1
Chr2
p11
p12
p21
p13
p14
p22
Kmer1(Color,Pos) = {(Chr1, p11), (Chr1, p12), (Chr1, p13), (Chr2, p14)}
Kmer2(Color,Pos) = {(Chr1, p21), (Chr2, p22)}
PSA(Read) = {(Chr1, p11), (Chr2, p14)} X
Read
k-mer 1
k-mer 2
B) Pseudoalignment using virtual colors without overlapping boundaries
b1
b2
b3
b4
b5
b6
b7
b8
b9
b10
Kmer1(Color,Pos) = {(b1, p11), (b2, p12), (b4, p13), (b6, p14)}
Kmer2(Color,Pos) = {(b3, p21), (b8, p22)}
PSA(Read) = {} X
C) Pseudoalignment using virtual colors with overlapping boundaries
Kmer1(Color,Pos) = {(b1, p11), (b2, p12), (b3, p12), (b4, p13), (b5, p13), (b6, p14)}
Kmer2(Color,Pos) = {(b3, p21), (b8, p22)}
PSA(Read) = {(b3,12)} ✓

Fig. 1. Toy example demonstrating how traditional pseudoalignment on genomic references could lead to spurious mapping or completely missing the correct mapping. In the example, we have two k-mers, which map to multiple references, with k-mer 1 mapping to multiple locations within chromosome 1. The individual mapped positions within a reference are denoted by pij, with i denoting the k-mer id and j denoting the order of the positions globally for that k-mer. PSA(Read) denotes the reference ids and the relative positions on the reference along which Read maps using pseudoalignment. The true mapping for the read in the example is (Chr1, p12). A) Using traditional pseudoalignment both Chr1 and Chr2 are selected as references from which the read originates. The position to which a read maps in pseudoalignment is governed by where the left-most k-mer of a read maps. Thus, p14 is picked as a position for the hit, representing mapping to Chr2. If a k-mer maps to multiple positions within a reference, then by design the position with the smallest coordinate is picked, which for Chr1 is p11. We thus have two false positive mappings (Chr1, p11), (Chr2, p14) and a false negative since (Chr1, p11) is not reported. B) When the references are projected onto a set of virtual colors whose boundary spaces do not overlap, we miss the mapping if the read spans across the boundaries (b2, b3), creating an $\varnothing$ set when taking the intersection across the colors. C) When the boundaries of virtual colors overlap, each k-mer is assigned two colors if it falls within the overlapping region. This ensures that the read gets mapped after pseudoalignment.

2.2 Pseudoalignment using virtual colors

The accuracy of compatibility information generated by pseudoalignment may be similar to alignment-based approaches when the reference sequences are comparably short, which is the case for transcriptomics and metagenomics (and when the sequenced sample is devoid of non-indexed references that are sequence-similar to the indexed references [37]). However, when directly applying pseudoalignment for mapping to a genome, where the reference sequences are large, comprising of chromosomes that often contain repetitive sequences, we observe a marked drop-off in mapping accuracy using various traditional pseudoalignment approaches. This can happen due to multiple matching locations for a k-mer within a single reference that can lead to false positive and false negative mapping positions during read mapping, as demonstrated in Figure 1 A.

Construction of the virtual colors — We propose modifying the pseudoalignment approach by projecting the original reference sequences (i.e. the original colors) onto a set of "virtual colors", that themselves do not exist as individual elements in the index, but which will represent the targets to which queries can map. The virtual colors, conceptually, consist of a set of overlapping bins, each of fixed maximal extent, on the reference sequences. This overlap is needed since the matching k-mers of a read can span the boundary of multiple disjoint virtual colors (and therefore become lost) (Figure 1 B, C). The virtual colors are designed not to extend over the individual reference sequences. Thus a virtual color b_i belonging to reference R_i will not overlap with any virtual color b_j belonging to reference j, $\forall i \neq j$. The construction of the virtual colors is controlled by two parameters namely ℓ_{vcol} and *ov_length*. The parameter *ov_length* controls how many bases a virtual color will overlap with the virtual color on its left. It should be at least equal to the read length to ensure that the entire read can fit within it. The parameter ℓ_{vcol} controls the total number of bases that will be spanned by a virtual color with most virtual colors on a reference spanning length of $(ov\_length + \ell_{\text{vcol}})$ bases. The leftmost virtual color on a reference will cover a total of ℓ_{vcol} bases. The rightmost virtual color on the reference will cover $ov\_length + rl$ bases such that $(B[R_i]-1)\cdot\ell_{\text{vcol}}+rl = L[R_i]$, where $B[R_i]$ denotes the total number of virtual colors contained within the reference sequence R_i and $L[R_i]$ denotes the length of the reference sequence. $B[R_i]$ can be computed as $ceil\left(\frac{L[R_i]}{\ell_{\text{vcol}}}\right)$. The virtual colors are created dynamically independent of the index.

Assigning virtual color to a k-mer— Before starting the mapping process, we create an array CB of length $|\mathcal{R}|$, such that:

$$CB[R_i] = \begin{cases} 0, & \text{if } R_i = 0 \\ CB[R_i-1] + B[R_i-1] & \text{otherwise} \end{cases}$$

where $CB[R_i]$ stores the cumulative number of virtual colors up to reference R_i. To assign a virtual color to a k-mer x for which there exists a mapping in the index, we extract the reference id, R_i, and p^x_{ij} — the relative position of the j-th occurrence of the k-mer x within reference i in the index. Corresponding to each occurrence of k-mer x on reference R_i. The color id for the k-mer x is computed as $V(x) = CB[R_i-1] + \frac{p^x_{ij}}{\ell_{\text{vcol}}}$. Further, we check if the k-mer x lies in the overlapping region between the bins. If that is the case, then it is assigned an additional color id $V(x)+1$. This ensures a read gets mapped if it entirely (or partly) lies within the overlapping regions.

Obtaining the final color set — Since a k-mer can be assigned two virtual colors corresponding to the same matching position, it might happen that a read after pseudoalignment also gets assigned two virtual colors for that same reference position. This is not a problem for mapping single-end reads, as removing mappings with duplicate color ids and positions is straightforward. However, individual mate mappings must be merged for paired-end reads to get the final set of mapped references and the corresponding starting positions. To do this, we first remove duplicate mappings for the individual mates. Two mappings a, b are deemed duplicate $\leftrightarrow$: $(\text{ori}(a) = \text{ori}(b))$, $(\text{ref}(a) = \text{ref}(b))$, and $(\text{pos}(p) = \text{pos}(b))$, where $\text{ori}(\cdot), \text{ref}(\cdot)$, and $\text{pos}(\cdot)$ denote the orientation, reference id (original color) and position for a mapped hit, respectively.

By design, a mapped hit will always have *at most one other duplicate* under the above criteria. These hits are deduplicated by keeping the hit with the smaller virtual color id. After removing the duplicate mapping hits for each mate, hits a and b are merged, with a being the mapping for mate 1 and b being the mapping for mate 2, if they meet the following criteria :

$$\begin{aligned} \text{ref}(a) &= \text{ref}(b) \\ |\text{vid}(a) - \text{vid}(b)| &\leq 1 \\ \text{ori}(a) &\neq \text{ori}(b) \\ -\text{dovetail}_{\max} \leq \text{pos}(a) - \text{pos}(b) &\leq \text{ins}_{\max}, \text{if ori}(a) = r \\ -\text{dovetail}_{\max} \leq \text{pos}(b) - \text{pos}(a) &\leq \text{ins}_{\max}, \text{if ori}(a) = f \end{aligned}$$

Here $\text{vid}(\cdot)$ represents the virtual color id and orientation f, r represents the forward and reverse orientation respectively. We use the values $\text{dovetail}_{\max} = 20$ and $\text{ins}_{\max} = 1000$ in this work.

The mapped hits supported by different numbers of k-mers can arise when the threshold $\tau < 1$. In such cases, we want to pick the hits for which we have the highest confidence, provided by the number of k-mer hits. Thus, we keep track of all valid mappings for the read pair, and let τ' denote the largest fraction of k-mer hits supporting any reported mapping. We subsequently filter out all merged mappings having $< \tau'$ fraction of valid mapped k-mers.

Merging paired-end reads before mapping — Before mapping the individual ends of a paired-end read, we try to see if we can merge the reads to create a single fragment before mapping. This idea has been suggested and adopted in several existing read-mapping tools (e.g. `STAR` [12] attempts to merge overlapping paired-end reads before mapping, and `Chromap` uses this approach for adaptor trimming [44]). We first try to merge the reads in dovetail orientation and, if that is not possible, then attempt to find an overlap merge. For a dovetail merge, the prefix of one mate should match with the suffix of the reverse complement of the other mate. For an overlap merge, the suffix of one mate should intersect with the prefix of the reverse complement of the other mate. For the merge to be successful, there should be at least an overlap of a certain number of bases between the mates (we use a default of 30). If the mates can be merged, then the merged fragment, and not the individual mates are mapped to the reference.

2.3 `Alevin-fry-atac` pipeline

We now describe the `alevin-fry-atac` pipeline for preprocessing single-cell ATAC-seq data. The first step is to map all the reads using the modified pseudoalignment algorithm described above. The default output of this phase is a RAD (reduced alignment data) format file. The RAD format is a chunk-based, binary file optimized for machine parsing that encodes the relevant information necessary for subsequent data processing. Further details of this format are described in [16], though it worth noting that the single-cell ATAC-seq variant of such files carries some distinct information from the single-cell RNA-seq variant (e.g. it includes positions for each mapping, but doesn't include UMI information).

Cell barcode correction — After mapping, the next step is cell barcode correction. While various approaches to barcode detection and correction are possible (`alevin-fry` implements several distinct modes), for `alevin-fry-atac` we currently only support correcting to an unfiltered permit list. The permit list file contains a list of experiment-independent barcodes that are a superset of the barcodes that should be observed in any given sample. We first scan through the RAD file and count the number of times a barcode corresponding to the mapped record appears (exactly). If a cell or barcode has at least `min-reads` records (default 10) mapping to it, and is also in the whitelist, it is marked as present. For all the other mapped records corresponding to cells not currently marked as present, a nearest neighbor search is performed on the barcode against the list of present cells. If a barcode has a unique (i.e. only one) present neighbor at edit distance 1, that barcode is corrected to that of the neighbor. All other records are ignored.

Sorting and Deduplication — In this step, a coordinate-sorted BED file is produced. First, a collection of temporary files are created, each corresponding to contiguous ranges of the underlying genome. The input RAD file is parsed and all records are routed to their appropriate temporary file based on their starting mapping location. The temporary files are then individually sorted (in parallel) and the sorted files are merged (in genomic coordinate order) to produce a BED file. During the sorting process, the duplicate entries for a record (i.e., entries with the same reference, start position, end position, and barcode) are collapsed, and the corresponding number of duplicate entries is stored. For each mapped fragment, the BED file contains the reference name, start position, end position, cell barcode and the total number of duplicates (including the fragment itself). The BED file can then be passed to a peak caller such as `MACS2` 2 [15], the output of which, along with the original BED file containing the counts, forms the input for downstream analysis.

3 Experimental Setup

3.1 Datasets

We evaluated and compared the performance of the different methods on three simulated and three experimental datasets. `Mason` [17] was used to generate the simulations for read lengths $50, 100, 150$ bases. One million paired-end fragments for each read length were created with a probability mismatch 0.25, according to the Illumina model in the simulator. The experimental single-cell ATAC-seq datasets were downloaded from the 10x website with the URLs provided in Table S1. The experimental datasets include the `Human 10K PBMC`, `Mouse 8K Cortex` and `Human 3K Brain` respectively. `Human 3K Brain` dataset unlike the other two, is a multi-omic dataset, for which both single-cell RNA-seq and single-cell ATAC-seq data are available, and, in this manuscript, we have worked with the single-cell ATAC-seq data.

3.2 Analysis pipeline

The analysis pipelines used to carry out the experiments in this manuscript were created using `Snakemake` [26]. Below, we describe how the individual experiments were carried out and evaluated.

Simulated data — `Chromap` was run without any `--preset` arguments. For `Bowtie2`, reads were aligned by setting the maximum fragment length (`X`) to 2000. These settings were taken from `Chromap`'s experiment repository on GitHub. `Alevin-fry-atac` always maps all the records irrespective of the whitelist file. The accuracy of the mappings was evaluated using a `Python` script that was originally used for evaluation of mapping accuracy in the stobemers [32] paper.

Experimental datasets — `Chromap` was run with `--preset atac` argument using a whitelist file on all the datasets. We could run `Cell Ranger ATAC` only on the `Human 10K PBMC` and `Mouse 8K Cortex` datasets, since `Human 3K Brain` is a multiomic dataset and required running `Cell Ranger Arc`, which we did not run (and could not easily compare). The downstream pipelines for analyzing ATAC-seq data such as `Signac` [39] require as input a `tabix`-indexed fragment file as produced in the `Cell Ranger ATAC` pipeline. Thus, a tabix index was created for the fragment files produced by both `alevin-fry-atac` and `Chromap`. We then use `MACS2` 2 [15] as a peak caller on the fragment files produced by all the methods. The peaks and the fragment files are then passed to `Signac` for quality control and downstream analysis, such as clustering into cell types.

Benchmarking — We benchmark the read mapping and the process of generating deduplicated, barcode-corrected fragment files separately using the `/usr/bin/time` command. `Chromap` was benchmarked with and without using the `--preset atac` arguments for the read mapping process. We ran both `Chromap` and `alevin-fry-atac` for a different number of threads, with each method being run three times for a given thread count. The average across the three runs was computed to obtain the time and memory usage for that thread. To benchmark mapping for `Chromap` run using `--preset atac`, we subtracted the time taken for sorting, deduplication and writing, which was output to `stdout` from the total time taken to run the method. We ran `Cell Ranger ATAC` only once with 32 cores and 100 Gb of allowed RAM, since it was not possible to benchmark the individual steps. `Cell Ranger ATAC` has previously been shown to be substantially slower than `Chromap`.

Software versions — The following software versions were used for the analyses in this manuscript: `Cell Ranger ATAC` version 2.1.0, `Chromap` version $0.2.7 - r493$, `Bowtie2` version 2.5.4, `mason` version 2.0.9, `MACS2` version 2.2.9.1, `R` version 4.3.2, `Signac` version 1.13.0, `tabix` version 1.16, and `Snakemake` version 8.18.0. Mapping for `alevin-fry-atac` was done using `piscem` version 0.11.0 and subsequent processing was done using the `alevin-fry-atac` branch of the `alevin-fry` repository.

4 Results

4.1 Performance on simulated data

Evaluating the effect of different parameters for `alevin-fry-atac`— We first varied the parameter ℓ_{vcol} (which controls the number of bases a virtual color will span, with most virtual colors spanning $\ell_{\text{vcol}} + ov\_length$ bases), for a given k-mer size k and pseudoalignment threshold τ and observed its impact on mapping accuracy on the simulated data for the different read lengths (Figures S1 to S4). We find that accuracy decreases continuously as ℓ_{vcol} increases, irrespective of the values of the other parameters. Importantly, the sharpest decrease is observed when the chromosomes themselves are used as the colors for pseudoalignment (removing the effect of virtual colors on the procedure entirely). The accuracy drops to $84 - 86\%$ from being $92 - 96\%$ (depending on the read length) when using virtual colors. In general, we observe a drop in accuracy of $\sim 10\%$ or more when disabling virtual colors. Longer reads generally yield higher accuracy across most settings, except in the specific parameter configuration where $\tau = 1$ and virtual colors are disabled, where reads with length 100 have the highest accuracy.

We next varied the threshold τ while keeping ℓ_{vcol} and k fixed (Figures S5 to S8). The accuracy decreases as the threshold decreases from 0.6 to 1. The decline seems to be sharpest from 0.8 to 1, with the magnitude of decrease in accuracy in general being much higher for larger read lengths. For $\tau = 1$, at all the ℓ_{vcol} values, the accuracy of reads with length 150 is smaller than reads with length 100. However, when the chromosomes are used as colors, the accuracy for reads with length 150 increases when τ increases from $0.6 - 0.8$ and falls again at 1.

Finally, we varied k, while keeping ℓ_{vcol}, τ fixed (Figures S9 to S12). We observe that for reads of length 50, accuracy increases from $k = 23$ to $k = 25$ and starts decreasing across most virtual color extends and thresholds. However, when chromosomes are used as colors, accuracy increases with an increase in k. For the reads with larger lengths, a larger k is preferred as the accuracy increases uniformly across the parameters.

The above results can be summarized in Figure 2. `Alevin-fry-atac` has higher accuracy for larger read lengths. It is clear from the simulations that virtual colors, irrespective of the ℓ_{vcol} value, *substantially* improve the mapping accuracy compared to using chromosomes as colors for pseudoalignment. For reads of length 50, ℓ_{vcol} seems to be the most important parameter, with accuracy dropping from 93.7% to 92.7% when ℓ_{vcol} is increased from $1,000$ to $1,000,000$. For reads of length 100 and 150, while increasing ℓ_{vcol} reduces the accuracy, the threshold τ is also important, with accuracy decreasing from 96.73% to 94.45% when the threshold is increased from 0.6 to 1.

Comparing performance with the other methods — We next compare the mapping accuracy of `alevin-fry-atac` with `Bowtie2` [20] (which does full read alignment) and `Chromap` on the simulated dataset in Table 1. As seen above for `alevin-fry-atac`, other methods also have higher accuracy for longer read lengths. `Bowtie2` has the highest accuracy followed closely by `Chromap` while `alevin-fry` has marginally lower accuracy across the evaluated read lengths. For the reads of length 50, the mapping accuracy of `alevin-fry` is $\sim 2\%$ lower than `Bowtie2` and around $\sim 1\%$ lower for the reads of length 150.

4.2 Experimental datasets

Mapping Rate — We also looked at the impact of varying the different parameters of `alevin-fry-atac` on the overall mapping rate for the experimental datasets, since mapping accuracy itself can't be measured due to a lack

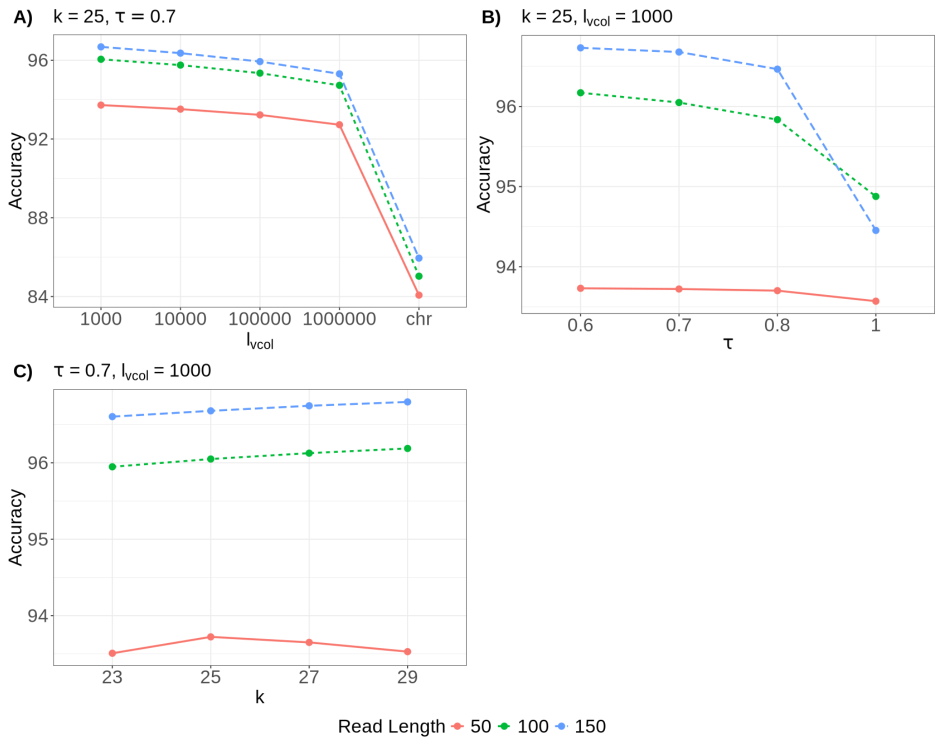

Fig. 2. Evaluating accuracy across different read lengths for `alevin-fry-atac` by varying A) ℓ_{vcol} for $k = 25, \tau = 0.7$, B) τ for $k = 25, \ell_{\text{vcol}} = 1000$, C) k for $\tau = 0.7, \ell_{\text{vcol}} = 1000$. The x-axis label *chr* in panel A implies mapping is done using chromosome sequence as colors (removing the impact of virtual colors on mapping procedure).

Table 1. Mapping accuracy on the simulated data for the different methods

Method	Read Length		
	50	100	150
`alevin-fry-atac` ($k = 25, \tau = 0.7, \ell_{\text{vcol}} = 1,000$)	93.72	96.05	96.68
`Chromap` (w=7, k=17)	95.48	97.11	97.61
`Bowtie2`	95.71	97.16	97.65

of known ground truth. We varied the parameters $\ell_{\text{vcol}}(1,000, 10,000, 100,000, 1,000,000, \text{chr})$, $\tau(0.7, 1)$ and $k(23, 25, 31)$ for the `Human 10K PBMC`, `Mouse 8K Cortex` and `Human 3K Brain` datasets respectively (Figure S13). As with the simulated dataset, we also find a decrease in mapping rate as ℓ_{vcol} increases for different k and τ values, with the fall being the sharpest when chromosomes are used as colors (i.e. when virtual colors are disabled). The highest mapping rate is observed for $k = 25$, followed closely by $k = 23$ for the human datasets, while for the mouse dataset the highest mapping rate is achieved at $k = 23$ followed by $k = 25$, with the lowest values observed at $k = 31$ across all datasets. A slightly lower mapping rate is also observed when *tau* is increased from 0.7 to 1. However, the above results change when chromosomes are used as colors, with $k = 31$ and $\tau = 1$ having a higher mapping rate compared to other k and τ values. The overall mapping rate for `alevin-fry-atac` for $k = 25, \tau = 0.7, \ell_{\text{vcol}} = 1,000$ is $(98.27\%, 97.73\%$ and $96.99)$ for the `Human 10K PBMC`, `Mouse 8K Cortex` and `Human 3K Brain` datasets respectively while for `Chromap` is $(95.31\%, 95.56\%$ and $94.50\%)$, without using the `--preset atac` argument.

Comparing peaks — We next compare the peaks obtained for the different methods across the different datasets. We find that the peaks produced by the `Cell Ranger ATAC` output cover the largest number of bases across the datasets (Table S2). The peaks produced by the `Cell Ranger ATAC` custom peak caller cover more bases than those produced by `MACS2`. On the human datasets, peaks produced by `alevin-fry-atac` cover more bases compared to `Chromap`, while for the mouse datasets, `Chromap` covers more bases. The peak overlap between the methods is greater than 90% across the datasets and in most cases more than 95% (Table 2). The overlap between `Chromap` and `Cell Ranger ATAC` is slightly larger than between `alevin-fry-atac` and `Cell Ranger ATAC`— though this difference disappears almost entirely when using the peak range of `Cell Ranger ATAC` as the denominator.

Comparing the clusters — We next do a pairwise comparison of the clusters obtained for the different methods, both qualitatively and quantitatively. Overall, a large overlap is observed between the clusters on the different datasets (Figures 3 and S14). There is less crossover (i.e. more concordance) between the clusters obtained for `Cell Ranger ATAC` and `alevin-fry-atac` compared to other pairwise comparisons on the `Human 10K PBMC` dataset. Similarly, the UMAP plots (Figures S15 to S17) show very similar cluster projections. Quantitatively, we use Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) to evaluate the clusters (Table 3). A high concordance is observed between the clusters across the datasets. For the `Human 10K PBMC` dataset, both ARI and NMI between `alevin-fry-atac` and `Cell Ranger ATAC` are higher than the other pairwise comparisons. On the `Mouse 8K Cortex` dataset, ARI and NMI for pairwise comparison between `alevin-fry-atac` and `Chromap` is highest. Similarly, NMI > 0.9 is observed between `alevin-fry-atac` and `Chromap` for the `Human 3K Brain` dataset. Though the methods are all concordant and the values are all close, we always observe that the clusters produced with `alevin-fry-atac` peaks have equal or higher concordance with those produced with `Cell Ranger ATAC` than those produced with `Chromap` do with those produced with `Cell Ranger ATAC`.

4.3 Memory and Running Time

We finally benchmarked the memory and running time of `alevin-fry-atac` and `Chromap` for mapping single-cell ATAC-seq reads on the `Human 10K PBMC` and `Mouse 8K Cortex` datasets respectively. We first compared how these metrics change when the number of threads is increased (Figure 4 A, Figure S18 A). For `Chromap`, we evaluated two different modes, one designed only for mapping (which we call “Map Only”) (i.e. not using the `preset` argument) and one designed for including the preprocessing along with the mapping of single-cell ATAC-seq data (which we call “Whitelist”) (i.e. using the `preset atac` argument). For `Chromap` (Whitelist) we subtract the time taken to sort and duplicate from the total time taken to run. This was done to evaluate the effect of entirely skipping reads whose barcodes reside outside the set of those in the permitlist, as `Chromap` (Whitelist) simply does not map or process these reads. We find that across both the datasets, using 4 and 8 threads, `Chromap` is faster than `alevin-fry-atac`. However, starting at 16 threads, the performance across the methods becomes comparable. At higher thread counts, the running time for `alevin-fry-atac` becomes substantially lower than that of `Chromap`. Specifically, we observe that while it is quite fast, `Chromap` does not seem to scale well beyond 8 threads, with little corresponding decrease in run time, though substantially faster mapping is possible (as evidenced by `alevin-fry-atac`’s performance). The maximum resident memory consumed by `alevin-fry-atac` is substantially smaller that that of `Chromap`, using approximately 6.3 and 5.5 Gbs for the `Human 10K PBMC` and `Mouse 8K Cortex` datasets respectively (Figure 4 B, Figure S18 B). `Chromap` (Whitelist) used 20.6, 19.7 Gbs and `Chromap` (Map Only) used 41.2, 36.3 Gbs for the above datasets respectively. There is a slight increase in memory usage across the methods when the number of threads is increased. We also summarize the above metrics in Table 4, where the results for `alevin-fry-atac` and `Chromap` are reported on 32 threads. Though both `Chromap` and `alevin-fry-atac` use similar but somewhat different methods for correcting for barcodes and deduplication, in both of these tools the correction and duplication steps are fast, and are not the main bottleneck in the pipeline. We also report the results for `Cell Ranger ATAC` for the entire analysis (including mapping, peak calling and clustering), since it is not possible to run and benchmark each step individually.

5 Discussion

We have introduced `alevin-fry-atac`, which provides a computationally efficient, memory-frugal, scalable and accurate framework for processing and mapping single-cell ATAC-seq data. It uses only $1/3$ of the memory of `Chromap` and less than $1/2$ of the memory of `Cell Ranger ATAC` when processing single-cell ATAC-seq (and `alevin-fry-atac` utilizes only $\frac{1}{6}$ of the memory of `Chromap` when the latter is not provided with a permit list during mapping). Since `Chromap` only maps the reads corresponding to the barcodes present in the permit list file, this could be one reason why we observe an increased time usage in the mapping only mode. Internally, `alevin-fry-atac` uses the `piscem` index, which is small and versatile and enables mapping reads from other technologies such as bulk RNA-seq, single-cell RNA-seq and now single-cell ATAC-seq.

`Alevin-fry-atac` is also computationally fast and scalable. When using 16 threads, the runtime of

[2] For `Cell Ranger ATAC` it is not possible to run and thus benchmark each step individually, thus the time and memory consumption also include other tasks such as peak calling and clustering.

Table 2. Percentage overlap between the peaks pairwise for the different methods across the datasets. Depending on which method is used as a denominator, percentage overlap can change. The values in the parenthesis denote the overlap when the second method was used as the denominator. N/A entries denote that `Cell Ranger ATAC` was not run on that dataset.

Method / Dataset	`alevin-fry-atac` vs `Chromap`	`alevin-fry-atac` vs `Cell Ranger ATAC`	`Chromap` vs `Cell Ranger ATAC`
`Human 10K PBMC`	96.59 (97.97)	96.84 (97.02)	99.14 (97.92)
`Mouse 8K Cortex`	97.99 (98.31)	97.56 (94.35)	98.37 (94.82)
`Human 3K Brain`	92.23 (97.80)	N/A	N/A

Table 3. Evaluating the concordance between the clusters obtained for the different methods pairwise across the datasets. The two entries per cell separated by comma denote Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI). N/A entries denote that `Cell Ranger ATAC` was not run on that dataset.

Method / Dataset	`alevin-fry-atac` vs `Chromap`	`alevin-fry-atac` vs `Cell Ranger ATAC`	`Chromap` vs `Cell Ranger ATAC`
`Human 10K PBMC`	0.87, 0.91	0.93, 0.94	0.87, 0.91
`Mouse 8K Cortex`	0.95, 0.96	0.94, 0.94	0.93, 0.94
`Human 3K Brain`	0.89, 0.91	N/A	N/A

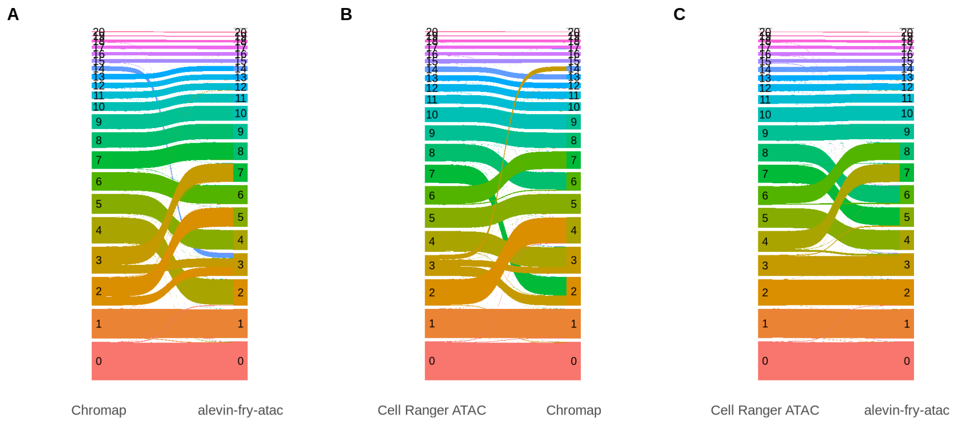

Fig. 3. Pairwise Sankey plot on the clusters obtained across the different methods for `Human 10K PBMC` dataset.

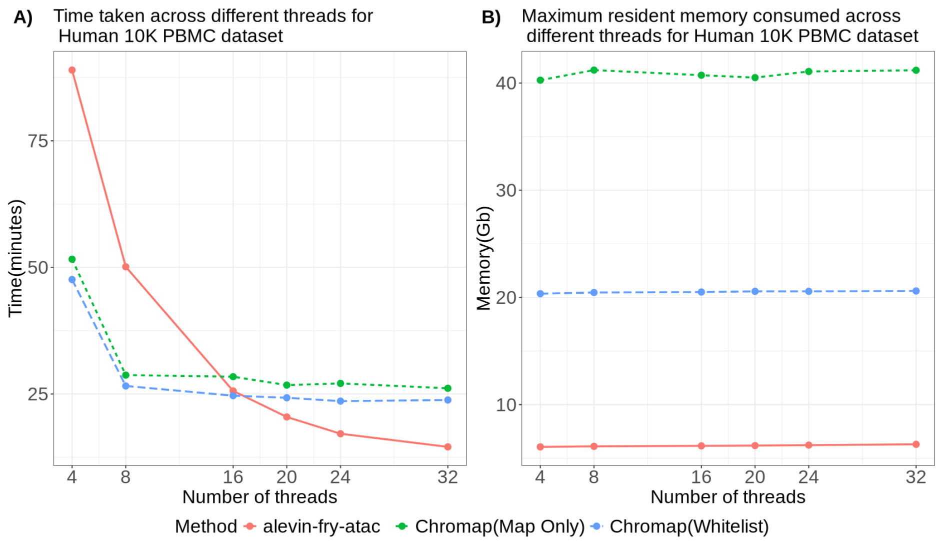

Fig. 4. Comparing the time taken and maximum resident memory consumed by `alevin-fry-atac` and `Chromap` for the `Human 10K PBMC` dataset across the different threads. For `Chromap` (Whitelist), `Chromap` is run with `preset atac` arguments along with the whitelist file. For `Chromap` (Map Only), `Chromap` is run only without the `preset` arguments.

Table 4. Time and maximum resident memory consumed by the different methods for mapping scATAC-seq reads. For both `Chromap` and `alevin-fry-atac` results are reported using 32 threads[2].

Method	Memory (Gb)	Time (minutes)
Human 10K PBMC		
`Cell Ranger ATAC` *	14.9	222
`alevin-fry-atac`	6.3	14.6
`Chromap` (Whitelist)	20.6	26.1
Mouse 8K Cortex		
`Cell Ranger ATAC` *	8.6	190
`alevin-fry-atac`	5.5	11.9
`Chromap` (Whitelist)	19.7	19.1

`alevin-fry-atac` is comparable to `Chromap`, but at higher thread counts it is faster. At 32 threads, `alevin-fry-atac` takes 178% less time for mapping than `Chromap`. At the same time, by introducing the concept of virtual colors, and modifying the pseudoalignment procedure accordingly, we can map reads at a similar accuracy to alignment-based methods. Such accuracy would not have been attainable without the use of virtual colors. We also observe a very high concordance between the peaks and clusters obtained for `alevin-fry-atac` with the other methods tested across experimental datasets. The `alevin-fry` ecosystem already supports the processing of single-cell RNA-seq. The additional support for single-cell ATAC-seq makes it the only fully open-source software directly supporting both modalities. This should ease and facilitate multi-modal analysis. We thus believe that `alevin-fry-atac` is a viable and strong alternative to the other methods.

While in this manuscript, the main focus has been on mapping single-cell ATAC-seq data, it should be trivial to extend and apply pseudoalignment using virtual colors to other genomic-centred mapping-based tasks and modalities, such as DNase-seq, Chip-seq and Hi-C. The accuracy obtained on the simulated datasets provides further confidence in this hypothesis.

While our mapping speed is already very high, it can be improved further by exploiting the property that a large fraction of ATAC-seq reads arise and map to certain fixed regions within the genome. A cache can be built for such reads that would reduce the calls to the main index when mapping. This feature was introduced in `Chromap`, where they report substantial speed improvements due to frequent use of these caches.

Finally, for our pseudoalignment strategy, we have queried all k-mers and utilized the "threshold-union" approach. However, other lightweight mapping strategies such as selective alignment [37] and pseudoalignment with structural constraints[16] can also be explored to see how they impact speed and mapping accuracy. Finally, all pipelines, including `alevin-fry-atac`, currently discard reads that map to more than one location. Such reads constitute $4-10\%$ of the total reads in the datasets analyzed in this manuscript, and might have a non-trivial impact on downstream analysis [27]. One interesting future direction would be to develop methods to resolve the origin of such reads probabilistically using methods such as expectation-maximization [11].

6 Code availability

The `alevin-fry-atac` software is written in Rust and C++ and is freely available under BSD 3-Clause license, as a free and open-source tool at `https://github.com/COMBINE-lab/alevin-fry-atac`. The scripts to produce the results in this manuscript are available at `https://github.com/NPSDC/alevin-fry-atac-paper-scripts`.

7 Funding

This work is supported by the National Institute of Health under grant award numbers R01HG009937 to R.P, and the National Science Foundation under grant award numbers CCF-1750472, and CNS-1763680 to R.P. Also, this project has been made possible in part by grants 252586 and 2024342821 from the Chan Zuckerberg Initiative DAF, an advised fund of Silicon Valley Community Foundation (or the Chan Zuckerberg Initiative Foundation). The funders had no role in the design of the method, data analysis, decision to publish, or preparation of the manuscript.

8 Acknowledgement

We want to thank Dr. Jeremy Simon for helpful discussions during this project. The scripts hosted at `https://jeremymsimon.github.io/chromap_vs_cellranger_scATAC_exploration_10x/index.html` were modified to do downstream analysis.

9 Declarations of interests

RP is a cofounder of Ocean Genomics, Inc.

References

1. Jarno N Alanko, Jaakko Vuohtoniemi, Tommi Mäklin, and Simon J Puglisi. Themisto: a scalable colored k-mer index for sensitive pseudoalignment against hundreds of thousands of bacterial genomes. *Bioinformatics*, 39(Supplement_1):i260–i269, 2023.
2. Fatemeh Almodaresi, Hirak Sarkar, Avi Srivastava, and Rob Patro. A space and time-efficient index for the compacted colored de Bruijn graph. *Bioinformatics*, 34(13):i169–i177, 2018.
3. Timo Bingmann, Phelim Bradley, Florian Gauger, and Zamin Iqbal. *COBS: A Compact Bit-Sliced Signature Index*, page 285–303. Springer International Publishing, 2019.
4. A. Sina Booeshaghi, Fan Gao, and Lior Pachter. Assessing the multimodal tradeoff. *bioRxiv*, 2023.
5. Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal probabilistic RNA-seq quantification. *Nature biotechnology*, 34(5):525–527, 2016.
6. F. P. Breitwieser, D. N. Baker, and S. L. Salzberg. KrakenUniq: confident and fast metagenomics classification using unique k-mer counts. *Genome Biology*, 19(1), November 2018.
7. Jason D Buenrostro, Paul G Giresi, Lisa C Zaba, Howard Y Chang, and William J Greenleaf. Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. *Nature methods*, 10(12):1213–1218, 2013.
8. Jason D Buenrostro, Beijing Wu, Ulrike M Litzenburger, Dave Ruff, Michael L Gonzales, Michael P Snyder, Howard Y Chang, and William J Greenleaf. Single-cell chromatin accessibility reveals principles of regulatory variation. *Nature*, 523(7561):486–490, 2015.
9. M Ryan Corces, Jeffrey M Granja, Shadi Shams, Bryan H Louie, Jose A Seoane, Wanding Zhou, Tiago C Silva, Clarice Groeneveld, Christopher K Wong, Seung Woo Cho, et al. The chromatin accessibility landscape of primary human cancers. *Science*, 362(6413):eaav1898, 2018.
10. Darren A Cusanovich, Riza Daza, Andrew Adey, Hannah A Pliner, Lena Christiansen, Kevin L Gunderson, Frank J Steemers, Cole Trapnell, and Jay Shendure. Multiplex single-cell profiling of chromatin accessibility by combinatorial cellular indexing. *Science*, 348(6237):910–914, 2015.
11. Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.
12. Alexander Dobin, Carrie A Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R Gingeras. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, 29(1):15–21, 2013.
13. Jason Fan, Jamshed Khan, Giulio Ermanno Pibiri, and Rob Patro. Keeping k-mers in check–Building fast, small, and composable indices based on the De Bruijn graph. In *Biological Data Science Meeting, Cold Spring Harbor Laboratory*, 2022.
14. Jason Fan, Jamshed Khan, Noor Pratap Singh, Giulio Ermanno Pibiri, and Rob Patro. Fulgor: a fast and compact k-mer index for large-scale matching and color queries. *Algorithms for Molecular Biology*, 19(1):3, 2024.
15. John M. Gaspar. Improved peak-calling with MACS2. *bioRxiv*, 2018.
16. Dongze He, Mohsen Zakeri, Hirak Sarkar, Charlotte Soneson, Avi Srivastava, and Rob Patro. Alevin-fry unlocks rapid, accurate and memory-frugal quantification of single-cell RNA-seq data. *Nature Methods*, 19(3):316–322, 2022.
17. M. Holtgrewe. Mason ? A Read Simulator for Second Generation Sequencing Data. *Technical Report FU Berlin*, October 2010.
18. Jamshed Khan and Rob Patro. Cuttlefish: fast, parallel and low-memory compaction of de Bruijn graphs from large-scale genome collections. *Bioinformatics*, 37(Supplement_1):i177–i186, 2021.
19. Daehwan Kim, Joseph M Paggi, Chanhee Park, Christopher Bennett, and Steven L Salzberg. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. *Nature biotechnology*, 37(8):907–915, 2019.
20. Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature methods*, 9(4):357–359, 2012.
21. David Lara-Astiaso, Assaf Weiner, Erika Lorenzo-Vivas, Irina Zaretsky, Diego Adhemar Jaitin, Eyal David, Hadas Keren-Shaul, Alexander Mildner, Deborah Winter, Steffen Jung, et al. Chromatin state dynamics during blood formation. *science*, 345(6199):943–949, 2014.
22. Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.
23. Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows–Wheeler transform. *bioinformatics*, 25(14):1754–1760, 2009.
24. Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, and 1000 Genome Project Data Processing Subgroup. The sequence alignment/map format and SAMtools. *bioinformatics*, 25(16):2078–2079, 2009.
25. Tommi Mäklin, Teemu Kallonen, Sophia David, Ben Pascoe, Guillaume Méric, David M Aanensen, Edward J Feil, Samuel K Sheppard, Jukka Corander, and Antti Honkela. High-resolution sweep metagenomics using ultrafast read mapping and inference. *bioRxiv*, page 332544, 2018.
26. Felix Mölder, Kim Philipp Jablonski, Brice Letcher, Michael B Hall, Christopher H Tomkins-Tinch, Vanessa Sochat, Jan Forster, Soohyun Lee, Sven O Twardziok, Alexander Kanitz, et al. Sustainable data analysis with Snakemake. *F1000Research*, 10, 2021.

27. Alexis Morrissey, Jeffrey Shi, Daniela Q James, and Shaun Mahony. Allo: Accurate allocation of multi-mapped reads enables regulatory element analysis at repeats. *bioRxiv*, 2023.
28. Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. Salmon provides fast and bias-aware quantification of transcript expression. *Nature methods*, 14(4):417–419, 2017.
29. Giulio Ermanno Pibiri. Sparse and skew hashing of k-mers. *Bioinformatics*, 38(Supplement_1):i185–i194, 2022.
30. Sebastian Preissl, Rongxin Fang, Hui Huang, Yuan Zhao, Ramya Raviram, David U Gorkin, Yanxiao Zhang, Brandon C Sos, Veena Afzal, Diane E Dickel, et al. Single-nucleus analysis of accessible chromatin in developing mouse forebrain reveals cell-type-specific transcriptional regulation. *Nature neuroscience*, 21(3):432–439, 2018.
31. Michael Roberts, Wayne Hayes, Brian R Hunt, Stephen M Mount, and James A Yorke. Reducing storage requirements for biological sequence comparison. *Bioinformatics*, 20(18):3363–3369, 2004.
32. Kristoffer Sahlin. Effective sequence similarity detection with strobemers. *Genome research*, 31(11):2080–2094, 2021.
33. Lorian Schaeffer, Harold Pimentel, Nicolas Bray, Páll Melsted, and Lior Pachter. Pseudoalignment for metagenomic read assignment. *Bioinformatics*, 33(14):2082–2088, 2017.
34. Alicia N Schep, Jason D Buenrostro, Sarah K Denny, Katja Schwartz, Gavin Sherlock, and William J Greenleaf. Structured nucleosome fingerprints enable high-resolution mapping of chromatin architecture within regulatory regions. *Genome research*, 25(11):1757–1770, 2015.
35. Wei Shen, Hongyan Xiang, Tianquan Huang, Hui Tang, Mingli Peng, Dachuan Cai, Peng Hu, and Hong Ren. KMCP: accurate metagenomic profiling of both prokaryotic and viral populations by pseudo-mapping. *Bioinformatics*, 39(1):btac845, 2023.
36. Giorgos Skoufos, Fatemeh Almodaresi, Mohsen Zakeri, Joseph N Paulson, Rob Patro, Artemis G Hatzigeorgiou, and Ioannis S Vlachos. AGAMEMNON: an Accurate metaGenomics And MEtatranscriptoMics quaNtificatiON analysis suite. *Genome biology*, 23(1):39, 2022.
37. Avi Srivastava, Laraib Malik, Hirak Sarkar, Mohsen Zakeri, Fatemeh Almodaresi, Charlotte Soneson, Michael I. Love, Carl Kingsford, and Rob Patro. Alignment and mapping methodology influence transcript abundance estimation. *Genome Biology*, 21(1), September 2020.
38. Rebekah R Starks, Anilisa Biswas, Ashish Jain, and Geetu Tuteja. Combined analysis of dissimilar promoter accessibility and gene expression profiles identifies tissue-specific genes and actively repressed networks. *Epigenetics & chromatin*, 12:1–16, 2019.
39. Tim Stuart, Avi Srivastava, Shaista Madad, Caleb A Lareau, and Rahul Satija. Single-cell chromatin state analysis with Signac. *Nature methods*, 18(11):1333–1341, 2021.
40. Picard Toolkit. Broad institute. *(No Title)*, 2019.
41. Chenfei Wang, Dongqing Sun, Xin Huang, Changxin Wan, Ziyi Li, Ya Han, Qian Qin, Jingyu Fan, Xintao Qiu, Yingtian Xie, et al. Integrative analyses of single-cell transcriptome and regulome using MAESTRO. *Genome biology*, 21:1–28, 2020.
42. Jie Wang, Cristina Zibetti, Peng Shang, Srinivasa R Sripathi, Pingwu Zhang, Marisol Cano, Thanh Hoang, Shuli Xia, Hongkai Ji, Shannath L Merbs, et al. ATAC-Seq analysis reveals a widespread decrease of chromatin accessibility in age-related macular degeneration. *Nature communications*, 9(1):1364, 2018.
43. Derrick E Wood and Steven L Salzberg. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome biology*, 15:1–12, 2014.
44. Haowen Zhang, Li Song, Xiaotao Wang, Haoyu Cheng, Chenfei Wang, Clifford A Meyer, Tao Liu, Ming Tang, Srinivas Aluru, Feng Yue, et al. Fast alignment and preprocessing of chromatin profiles with Chromap. *Nature communications*, 12(1):6566, 2021.
45. Yong Zhang, Tao Liu, Clifford A Meyer, Jérôme Eeckhoute, David S Johnson, Bradley E Bernstein, Chad Nusbaum, Richard M Myers, Myles Brown, Wei Li, et al. Model-based analysis of ChIP-Seq (MACS). *Genome biology*, 9:1–9, 2008.
46. Ilya Y Zhbannikov, Samuel S Hunter, Matthew L Settles, and James A Foster. SlopMap: a software application tool for quick and flexible identification of similar sequences using exact k-mer matching. *Journal of data mining in genomics & proteomics*, 4(3), 2013.