HHS Public Access

Author manuscript

Nat Methods. Author manuscript; available in PMC 2022 September 11.

Published in final edited form as:

Nat Methods. 2022 March ; 19(3): 316–322. doi:10.1038/s41592-022-01408-3.

Alevin-fry unlocks rapid, accurate, and memory-frugal quantification of single-cell RNA-seq data

Dongze He[1], **Mohsen Zakeri**[2], **Hirak Sarkar**[3], **Charlotte Soneson**[4,5], **Avi Srivastava**[6], **Rob Patro**[*,2]

[1]Department of Cell Biology and Molecular Genetics and Center for Bioinformatics and Computational Biology, University of Maryland

[2]Department of Computer Science and Center for Bioinformatics and Computational Biology, University of Maryland

[3]Department of Biomedical Informatics, Harvard Medical School, Boston, Massachusetts

[4]Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland

[5]SIB Swiss Institute of Bioinformatics, Basel, Switzerland

[6]New York Genome Center, New York City, New York

Abstract

The rapid growth of high-throughput single-cell and single-nucleus RNA-sequencing (sc/snRNA-seq) technologies has produced a wealth of data over the past few years. The size, volume, and distinctive characteristics of these data necessitate the development of new computational methods to accurately and efficiently quantify sc/snRNA-seq data into count matrices that constitute the input to downstream analyses. We introduce the `alevin-fry` framework for quantifying sc/snRNA-seq data. In addition to being faster and more memory frugal than other accurate quantification approaches, `alevin-fry` ameliorates the memory scalability and false-positive

[*]Corresponding author: rob@cs.umd.edu.

Author Contributions Statement

All authors conceptualized the method. D.H., A.S., R.P., M.Z. and H.S. implemented the software. M.Z. and R.P. benchmarked the tools. D.H., R.P. and C.S. analyzed the results. All authors wrote and approved the manuscript.

Competing Interests Statement

RP is a co-founder of Ocean Genomics, inc. The remaining authors declare no competing interests.

Code availability

`Alevin-fry` is written in Rust (https://www.rust-lang.org/), and is available under the BSD 3-Clause license, as a free and open-source tool at https://github.com/COMBINE-lab/alevin-fry. The specific version used in this work (27) has been uploaded to zenodo at https://doi.org/10.5281/zenodo.5799568. The generation of RAD files is implemented as part of the `alevin` command of the `salmon` tool, available at https://github.com/COMBINE-lab/salmon. Both tools are also available through bioconda (28). The roe R package has been developed for the construction of splici reference sequences, and it is available at https://github.com/COMBINE-lab/roe as free and open-source software under the BSD 3-Clause license. Useful scripts and functions for simplifying reference preparation and quantification as well as utilities for reading `alevin-fry` output in Python and R are available at https://github.com/COMBINE-lab/usefulaf. Support for reading `alevin-fry` output (including USA-mode output) has been integrated into the `fishpond` package available at https://github.com/mikelove/fishpond as well as through Bioconductor. The scripts used to perform the analyses in this manuscript are available at https://github.com/COMBINE-lab/alevin-fry-paper-scripts/.

Reporting Summary

Further information on research design is available in the Nature Research Reporting Summary linked to this article.

expression issues that are exhibited by other lightweight tools. We demonstrate how `alevin-fry` can be effectively used to quantify sc/snRNA-seq data, and also how the spliced and unspliced molecule quantification required as input for RNA velocity analyses can be seamlessly extracted from the same preprocessed data used to generate regular gene expression count matrices.

1. Introduction

Both the number and scale of single-cell RNA-seq (scRNA-seq) experiments have been growing rapidly in recent years (1). The data generated by various scRNA-seq technologies have distinct characteristics preventing them from being processed by the otherwise mature and widely-used tools developed for bulk RNA-seq data (2–4). While `Cell Ranger` exists as a commercial solution for preprocessing data generated using popular 10x Genomics technologies, it is both computationally and memory intensive, and since version 3, has been developed as a closed-source product, limiting the transparency of the methods it implements. Further, it does not have built-in support for technologies beyond those developed by 10x Genomics. Therefore, to address the computational challenges that arise in the processing of high-throughput scRNA-seq data, numerous new approaches for efficient preprocessing have been developed.

Srivastava et al. (5) introduced `alevin`, which focused on improving the computational efficiency of tagged-end scRNA-seq quantification and also introduced a novel approach for resolving gene-multimapping UMIs. Likewise, the `raindrop` tool (6) pairs a custom lightweight mapping approach with a reduced index to count UMIs mapping to genes, providing a fast counting approach. Melsted et al. (7) introduced the `kallisto|bustools` pipeline for processing scRNA-seq data; the approach focuses on modularity and speed, using pseudoalignment (3) to the transcriptome to produce intermediate BUS files (8) that are subsequently manipulated using `bustools` commands.

Most recently, Kaminow et al. (9) introduced `STARsolo`, a preprocessing method built directly atop the `STAR` aligner upon which `Cell Ranger` also relies. `STARsolo` focuses on being a fast and easy-to-use solution for processing single-cell and single-nucleus RNA-seq (snRNA-seq) data that can be tuned to mimic `Cell Ranger`, while being much faster and more memory frugal. However, since it performs spliced alignment to the genome, `STARsolo` is more memory and time-intensive than pseudoalignment to the transcriptome (at least for scRNA-seq data).

In this work, we present `alevin-fry`, a configurable framework for the processing of tagged-end scRNA-seq and snRNA-seq data. `Alevin-fry` has been designed as the successor to `alevin`. It subsumes the core features of `alevin`, while also providing important new capabilities and considerably improving the performance profile, and we anticipate that new method development and feature additions will take place primarily within the `alevin-fry` codebase. `Alevin-fry` can preprocess scRNA-seq data more quickly than the next-fastest method, `kallisto|bustools`, while also vastly reducing the considerable number of spuriously expressed genes predicted under pseudoalignment-to-transcriptome approaches (9). Simultaneously, `alevin-fry` exhibits similar accuracy to

STARsolo while processing data appreciably faster and requiring less memory. On snRNA-seq data, where intronic sequences are often included for quantification, alevin-fry and STARsolo are both faster and use less memory than kallisto|bustools. In fact, alevin-fry can process snRNA-seq data with the same speed and memory efficiency with which it processes scRNA-seq data, substantially outperforming both STARsolo and kallisto|bustools. Alevin-fry is an accurate, computationally efficient, and easy-to-use tool that presents a unified framework for preprocessing sc/snRNA-seq data for gene expression or RNA velocity analysis, making it an appealing choice for processing the diverse and growing array of experiments being performed.

2. Results

We demonstrate the performance and accuracy of alevin-fry in a variety of different use cases, and compare its computational resource usage as well as the quality of its results to those provided by the other recently-introduced tools STARsolo and kallisto|bustools. We examine results on simulated data (Section 2.2), on a scRNA-seq dataset where the effect of alignment pipelines has previously been explored (Section 2.3), in the context of preparing count matrices for an RNA velocity analysis (Section 2.4), for the processing of a snRNA-seq dataset (Section 2.5), and finally we investigate the overall runtime and peak memory usage characteristics on this broad array of datasets (Section 2.6).

2.1. Overview of alevin-fry.

Alevin-fry is a configurable framework for the processing of (sc/snRNA-seq) data (see Fig. 1), supporting many tagged-end sc/snRNA-seq protocols. After preparing a reference with respect to which quantification should occur, it makes use of salmon (4) for barcode and UMI parsing and mapping of fragments to the reference index. Accepting as input this mapping information, alevin-fry generates a permit-list for cellular barcodes (CBs) that will be quantified in subsequent steps. Using a multi-threaded approach, it filters and collates the mapping records for permitted CBs to produce a representation optimized for quantification. During quantification, alevin-fry processes the mapping records assigned to the permitted CBs in parallel, and applies one of the available (user-specified) UMI resolution algorithms to estimate a count for each gene within each quantified cell. This result in a gene-by-cell count matrix which can be used for numerous downstream analyses.

2.2. Simulated data.

We first evaluated the different methods on data from a non-parametric simulation first introduced by Kaminow et al. (9). Details about the simulation are provided in Section 4.7. Table 1 displays the results of different methods as evaluated under various metrics on the set intersection of the cells quantified by all methods, where STARsolo was run to perform Cell Ranger-like barcode filtering. Though Cell Ranger was not included in this comparison, we expect it to perform very similarly to STARsolo under default parameter settings, as reported by Kaminow et al. (9). The definitions of these metrics are given in Supplementary Section S8. While no method yields the best performance universally, there are some clear trends that can be observed. First, as noted by Kaminow et al. (9), the methods that perform mapping (either pseudoalignment or pseudoalignment

with structural constraints) directly to the spliced transcriptome alone performed worse than the other approaches — often considerably — under most metrics (the sole exception being the mean per-cell relative false-negative rate). Specifically, these approaches exhibited a markedly reduced cell-level Spearman correlation with the truth, as well as largely inflated relative false-positive expression (27–32%) and increased mean absolute relative deviations (MARD). Among the two evaluated approaches that map only to the spliced transcriptome, `alevin-fry` (in `sketch` mode) performed better than `kallisto|bustools`. On the other hand, the methods that map to expanded references, either the whole genome in the case of `STARsolo` or the *splici* reference (see Section 4.7 and Supplementary Section S2) in the case of `alevin-fry`, all generally performed well under the various metrics. `STARsolo` exhibited the highest cell-level Spearman correlation, as well as the smallest relative false-positive and relative false-negative rate, while `alevin-fry` exhibited the lowest MARD (both when run in `sketch` mode and when using selective-alignment).

To explore the false-positive expression estimates in more detail, we plotted the frequency distribution of the number of cells in which each gene appears, where genes are sorted in descending order, independently per method (see Figure 2a). We observed that `STARsolo` and both variants of `alevin-fry` that make use of the *splici* index followed a very similar frequency distribution, and that this was distinct from the frequency distribution followed by `kallisto|bustools` and `alevin-fry` when mapping only to the spliced transcriptome. This suggests that mapping to the spliced transcriptome alone not only results in hundreds of spuriously expressed genes per cell, but many of these genes themselves are inferred to be expressed across hundreds of cells.

Though there were differences under all metrics reported by the methods mapping to the expanded reference, the magnitude of these differences was generally small, and, in particular, was much smaller than the difference between any of these methods and those methods that map only to the spliced transcriptome. Moreover, we observed that, holding the other variables fixed, selective-alignment yielded a small but consistent accuracy improvement over pseudoalignment with structural constraints. Presumably, this resulted largely from the ability of selective-alignment to discard fragments arising from outside of the spliced or unspliced transcriptome that would otherwise be spuriously assigned to some target. Nonetheless, we observed that pairing the expanded (*splici*) reference with an appropriate UMI resolution strategy that is aware of both spliced and unspliced gene variants allowed for the use of `sketch` mode (pseudoalignment with structural constraints) in a manner that corrected the high number of false-positive expression predictions that were otherwise observed when mapping only to the spliced transcriptome.

2.3. Analysis of a zebrafish pineal gland dataset.

To explore the performance of `alevin-fry` in an experimental sample where the alignment pipeline has previously been shown to have an impact on downstream analysis, we reanalyzed an existing *Danio rerio* (zebrafish) pineal gland dataset (10, 11).

Shainer and Stemmer (11) reported that for this data, `kallisto| bustools`'s quantifications enable the `FindClusters` function of `Seurat` (12) to recover two distinct

cone photoreceptor (*cPhR*) clusters — the *cPhR* expressing the *red cone opsin* (*red+* cells) and the *cPhR* expressing *parietopsin* (*PT+* cells). Conversely, when using the quantifications from `Cell Ranger`, the *red+* and *PT+ cPhR* clusters are collapsed into a single *cPhR* cluster that expresses the main marker genes for both *cPhR* clusters. Given that the *red+* and *PT+* cells are two distinct types that represent mutually exclusive neuronal fates of photoreceptors in this tissue (13), one would likely view the separate clusters as an important biological signal.

To further investigate the `differences` demonstrated in Shainer and Stemmer (11) under a different set of preprocessing tools, we processed this data with `STARsolo` (with UMI resolution strategies described in Section 4.6), `kallisto|bustools`, and `alevin-fry` using USA-mode and the *splici* reference. To normalize across the cell filtering methods, we ran all tools to produce unfiltered quantifications, then filtered the resulting count matrices using the `DropletUtils` (14) R package. We also evaluated the clustering results of all methods on only the subset of cellular barcodes discovered by `alevin-fry`'s knee-finding method.

We first reproduced the analysis `performed` by Shainer and Stemmer (11), including the additional methods considered in this study. We estimated the high-quality cells using `DropletUtils` and created `Seurat` objects on the filtered count matrices (details in Section 4.6) to find cell clusters (Supplementary Figs. S5.1 and S5.2). As previously reported (11), we found that `kallisto|bustools` exhibited two distinct *cPhR* clusters. Likewise, we found that `STARsolo`, using the default `1MM` UMI resolution strategy, did *not* separate these *cPhR* clusters (as expected, since the default `STARsolo` parameters are designed to mimic `Cell Ranger` (9)). `Alevin-fry`, which was not considered in the original manuscript, yielded quantifications that resulted in two separate *cPhR* clusters. In accordance with previous results (11), we found that under the `kallisto|bustools` quantifications, *col14a1b*, a collagen gene, was detected as the strongest differentially expressed gene (DEG) between the two *cPhR* clusters. This gene was not detected at any appreciable level among any of the cells by `STARsolo` or `alevin-fry` after the prescribed filtering was applied, despite the fact that the `alevin-fry` quantifications still resulted in two *cPhR* clusters. This led us to investigate the `differences` that might be causing the clustering of the `STARsolo` counts to yield a single *cPhR* cluster while both `alevin-fry` and `kallisto|bustools` counts yield two *cPhR* clusters for these cells.

We investigated the effect of changing the default UMI resolution strategy applied by `STARsolo`. When using both the `1MMDir` and `exact` UMI resolution strategies, the `STARsolo` counts yielded two *cPhR* clusters. In this data, the UMI resolution strategy seemed to be an important factor for the signal separating these clusters to be detected by `Seurat`'s clustering algorithm. Notably, even when distinct clusters were not found, the t-SNE embedding computed from `STARsolo` counts placed subsets of the *cPhRs* in different positions in the embedding, and the subsets at these different positions expressed, disjointly, the marker genes for the *red+* and *PT+* cells; the clusters were just not separated by the clustering algorithm, as shown in Supplementary Figs. S5.2e and S5.2f.

Next, we investigated the effect of filtering barcodes using the `emptyDrops` function of `DropletUtils`, rather than the barcode frequency inflection point (details in Section 4.6). When applying this filtering, *none* of the tested methods yielded two *cPhR* clusters at either of the tested resolution parameters (Supplementary Figs. S5.3 and S5.4). Again, while `Seurat` did not separate these *red+* and *PT+* cells, an inspection of the placement of the corresponding cells in the respective t-SNE embeddings, and the genes that they expressed, suggested that a signal distinguishing these cells was present in all tested methods.

Additionally, we evaluated the clustering results for the different quantification methods restricting the set of barcodes to those selected by `alevin-fry`'s knee-distance filtering procedure (details in Section 4.6). Under this filtering approach, all of the tested methods discovered two *cPhR* clusters at the 0.9 and 1.2 resolution parameters (Supplementary Figs. S5.5 and S5.6). This was true even for `STARsolo`'s quantification results when using the `1MM` UMI resolution strategy.

Taken together, these results suggest that the main factors in the separation of these clusters during processing are a combination of (1) the specific filtering parameters used to retain cell barcodes, (2) the UMI deduplication strategy and (3) specific thresholds selected for feature detection and filtering. These results are investigated further in Supplementary Section S6. Overall, we observed a general tendency for more strict filtering to elucidate a signal between these clusters that can be detected by `Seurat`'s clustering algorithm. The signal itself, in terms of the biologically-relevant *opn1lw1* (15) and *parietopsin* (16) marker genes, was quite strong in the quantifications produced by all of the tested methods, if explicitly sought out. This suggests that the specific clustering algorithm used may affect the ability to automatically separate these distinct clusters of cells. We have not investigated this here, but it may be an interesting direction for further work.

Finally, we explored the strong DEG marker signal of the *col14a1b* gene found between the two *cPhR* clusters in the `kallisto|bustools` quantifications, which is absent from the filtered counts of `STARsolo` and `alevin-fry`. To the best of our knowledge, there is no immediate biological mechanism that would lead this gene to be a differential marker between *red+* and *PT+* cells. We performed a detailed, read-level analysis on the expression of this gene to explore the causes of this quantification difference. While this analysis was computationally-intensive, and therefore not feasible at scale across experiments or as a standard part of preprocessing pipelines, it helped elucidate the mechanism at work and why such differences might manifest.

We ran `kallisto` (3) in bulk mode to isolate the reads that were mapped to the constituent transcripts of *col14a1b*. We extracted these reads and attempted to align them to the corresponding transcripts. We found that they almost universally produced poor quality alignments, where the only long contiguous matches between the read and the transcript were stretches of low-complexity sequence close to the indexed k-mer length.

We ran BLAST (17) to query these reads against the NCBI nucleotide database to investigate their potential origins. For reads we examined, the top BLAST hits contained the *pde6hb* (phosphodiesterase 6H, cGMP-specific, cone, gamma, paralog b) gene, which

has biologically plausible expression in this dataset. However, this gene does not appear in the Ensembl 101 *D. rerio* annotation used in this section or the original analysis (11). Thus, in this case, both `STARsolo` and `alevin-fry` avoided seemingly misattributing the large number of reads actually arising from *pde6hb* to other genes within the annotation, while `kallisto|bustools` attributed many of these reads to *col14a1b*, for which there does not appear to be any evidence of expression.

The spurious expression of genes when quantifying with a pseudoalignment-to-transcriptome based approach has been previously reported by Kaminow et al. (9), and has been reported to result in the estimated expression of biologically implausible genes (18). In this dataset, it resulted in the expression of a gene that is detected as the strongest marker between these clusters of interest under `kallisto|bustools` quantifications, and that is almost certainly a spurious result that derived from the use of pseudoalignment-to-transcriptome with no filtering of mapping results. Generally, such occurrences may not be particularly rare, and caution should be applied when interpreting metrics like total gene detection, or median gene or UMI count, particularly among methods that employ different fragment mapping approaches, as larger values of such quantities may indicate reduced precision and not just increased sensitivity.

2.4. RNA Velocity in a mouse pancreas experiment.

The USA-mode of `alevin-fry` generates unspliced (U), spliced(S) and ambiguous (A) counts for each gene within each cell. Those counts can be used to estimate single-cell RNA velocity (19), which represents cellular transcriptional kinetics of cells that are sequenced in scRNA-seq. In a Mus musculus (mouse) pancreas dataset, the ratio of U:S:A is `0.125 : 0.806 : 0.069`. As RNA velocity estimators (19, 20) most often take only spliced and unspliced counts as the input, the ambiguous counts need either to be discarded or to be apportioned toward spliced and unspliced counts. We tested seven different strategies for handling these ambiguous counts and observed that assigning the ambiguous count differently led to distinct velocity and latent time estimates (Supplementary Section S7). Here, we discuss the result of assigning all ambiguous counts as spliced counts, since this coincides with the reasonable prior belief that most reads in this type of experiment should arise from spliced transcripts (see Section 4.8 for details). By doing so, the ratio of U:S is then `0.125 : 0.875`. The streamlines in the velocity graph (Fig. 2b) portray the cycling nature of the Ductal cells and endocrine progenitors, the cellular development process of endocrine progenitors (indicated by the concentration of the transcription factor Ngn3), and the differentiation process of endocrine cells, which ends with Beta cells at the latest time point, as described by Bergen et al. (20).

Setting the corresponding RNA velocity-related flags for `STARsolo (--soloFeatures Gene Velocyto)` and for `kallisto|bustools` (`--workflow lamanno`), returns the counts required by the RNA velocity pipelines. The resulting U:S:A ratio of `STARsolo` counts was `0.122 : 0.834 : 0.044`, and the resulting U:S ratio of `kallisto|bustools` counts was `0.181 : 0.819` (`kallisto|bustools` does not report ambiguous counts). Although the ratios were similar across the results of all methods, the velocity and the latent time estimation were distinct. In the velocity graph produced by `kallisto|`

bustools counts (Supplementary Fig. S7.5a), for example, the streamlines formed a back-flow, and the arrows pointed from the differentiated cells (Epsilon, Beta, and Alpha cells) back towards the pre-endocrine cell cluster, corresponding to the results reported in Soneson et al. (21). The velocity graph derived from STARsolo counts, after assigning ambiguous counts to spliced (Supplementary Fig. S7.3b), avoided such back-flow, but did not reveal the cycling population of Ductal cells, and some streamlines over the Beta cell cluster pointed in the opposite direction, against other streamlines over the same cell population.

Additionally, while the latent time assignments computed by scVelo when using the alevin-fry (Supplementary Fig. S7.2b) and STARsolo (Supplementary Fig. S7.4c) counts matched the streamlines in their respective velocity graph and those provided in the scVelo tutorial, the latent time assignment derived from the kallisto|bustools counts were discordant with those of the other methods as well as with the directions of velocity arrows leading from the Ductal cell cluster and pre-endocrine cell cluster to the differentiated cells. Specifically, when utilizing kallisto|bustools counts, the latent time estimated by scVelo (Supplementary Fig. S7.5b) originated in the cluster of Beta cells, and concorded with the velocity arrows leaving this cluster, but ran opposite to the main flow from the Ductal, Ngn3 and Pre-endocrine clusters into the differentiated cell clusters.

In summary, comparing the velocity graphs generated by all three methods on the endocrine pancreas dataset, the velocity streamlines and latent time assignments derived from alevin-fry counts well delineated the cellular development process of pancreatic endocrinogenesis, and those derived from STARsolo recapitulated most of the expected biology, but differed in some details, while the results derived from the kallisto|bustools counts only recapitulated parts of the expected biology.

2.5. Processing of a mouse placenta snRNA-seq dataset.

Like scRNA-seq, snRNA-seq technology is increasingly used to explore many types of biological questions, particularly in situations where full-cell scRNA-seq would be difficult or dissociation unlikely to succeed. In this section, we analyzed a snRNA-seq dataset from the mouse placenta (22). The details for processing the dataset can be found in Section 4.9 and Supplementary Section S9.

Among the 10, 483 high-quality cells in the quantifications processed by alevin-fry, a total of 17 clusters were found with a clustering resolution parameter of 0.6. To assign cell types for each cluster, a preprocessed Seurat object (Supplementary Section S9) was used as the reference for cell-type classification using Seurat's anchor transfer functionality. In this Seurat object, cells were classified as belonging to 5 major cell types: blood cells, decidual stroma, endothelial, fetal mesenchyme, and trophoblast. Those cell types correspond to the basic structure of the placenta, which consists of the maternal decidua, the junctional zone, and the labyrinth zone (22, 23).

By transferring the cell type annotations from the reference Seurat object to the alevin-fry result, all five clusters were detected, and the t-SNE embedding of the alevin-fry counts was similar to that of the reference object (Fig. 2c). This process was

also performed for the result of STARsolo (Supplementary Fig. S9.2b) and kallisto|bustools (Supplementary Fig. S9.2c), and the five essential cell types were also detected. In conclusion, all three methods were able to retain the most significant biological signals captured in the snRNA-seq experiment, and subsequently produced similar cell type assignments and t-SNE embeddings.

Subsequently, the 7, 027 nuclei that were assigned as trophoblast in the alevin-fry result were selected to analyze refined trophoblast subclusters. As some cell types had only a few corresponding nuclei, we set the clustering resolution very high (as 2.5) to detect the detailed clustering assignments; 27 clusters were found. Referring to anchors from the reference result (22), which defined 13 cell types, 12 of them (all but *SynTII precursor*) were assigned to these 27 clusters. After applying the same procedure, the 6, 837 trophoblast nuclei assigned under the kallisto|bustools counts resulted in the discovery of 11 cell types (all but *SynTII precursor* and *SynTI precursor*) and the 6, 631 trophoblast nuclei assigned under the STARsolo counts resulted in 10 cell types being found, all but *SynTI precursor*, *LaTP* and *JZP1*. The reference labels not assigned across methods generally had low barcode counts in the reference dataset. Just as with the cluster analysis explored in Section 2.3, the “absence” of a cluster depends on the details of the filtering approach, intermediate processing, and clustering parameters, and so the lack of a distinct cluster annotated via reference transfer does not necessarily indicate that the relevant biological signal was not present in the counts produced by a method.

In summary, all tools demonstrated robust recapitulation of the major expected biological signals from this snRNA-seq experiment, with alevin-fry recovering slightly more known cell types when sub-clustering trophoblast nuclei.

2.6. Speed and memory usage.

Finally, we assessed the speed and memory requirements of the three tools tested in this manuscript across the datasets explored in the previous sections as well as using the PBMC10k dataset (24) with the latest 10x reference annotation. We exclude Cell Ranger from this analysis, as it has previously been demonstrated that STARsolo can produce results that are almost identical to those of Cell Ranger, but that it is much faster and requires less RAM (9) (see Section 4.10).

Among the methods tested, alevin-fry, when using sketch mode, was the fastest (Fig. 2d). When processing scRNA-seq data and indexing only the spliced transcriptome, kallisto|bustools was the second-fastest tool. When both alevin-fry and kallisto|bustools are configured to use the spliced transcriptome alone as the mapping target, alevin-fry exhibited the lowest memory usage, followed by kallisto|bustools. The speed of STARsolo matched that of kallisto|bustools as the number of threads was increased (often at around 16 to 20 threads depending on the specific details of the hardware configuration being used), but, by virtue of aligning against the entire genome, it consumed more memory when performing a standard (spliced) scRNA-seq analysis. As expected, when alevin-fry was configured to use the *splici* reference rather than just the spliced transcriptome, there was a moderate increase in the memory usage (e.g. to ~ 10 GB

in dense mode and ~ 6.5 GB in sparse mode for the most recent 10x Genomics annotation of the *Homo sapiens* (human) transcriptome). The runtime saw little effect when mapping against the *splici* reference compared to the spliced transcriptome, and there also appeared to be little consistent difference in the mapping speed of `alevin-fry` when using the sparse rather than the dense index. Thus, while mapping against the *splici* reference required more memory, it had little effect on the runtime and yielded markedly more accurate counts, as it avoided the pitfalls of pseudoalignment-to-transcriptome described by Kaminow et al. (9).

When processing snRNA-seq data, `alevin-fry` was the fastest and most memory-frugal method (Fig. 2d and 2e). Since `STARsolo` and `alevin`-fry indices already contained the relevant intronic sequence, their index size did not grow when processing snRNA-seq samples or preparing RNA velocity inputs. However, when processing snRNA-seq data, there was a notable performance inversion between `STARsolo` and `kallisto|bustools`. The size of the `kallisto|bustools` index grew much larger than those of the other tools, and the speed decreased substantially. Thus, depending upon the specific organism and annotation complexity, when processing snRNA-seq samples, `STARsolo` was the second-fastest and second-most memory-frugal tool (even when using its dense suffix array index). On the dataset examined here, compared to `alevin-fry` (sparse, unfiltered), `STARsolo` took ~ 2.6 times as long and used ~ 6.3 times as much memory while `kallisto|bustools` took ~ 4.1 times as long and used ~ 13.1 times as much memory.

In summary, `alevin`-fry was the fastest method, on average completing in under half the time required by the next fastest method. It also exhibited tightly-controlled peak memory requirements, with processing using the sparse index completing in less than 8 GB of memory for all the different organisms and datasets processed in this paper. Among `STARsolo` and `kallisto|bustools`, which method was faster or which required less memory depended on the specific type of data being processed and the details of the reference being used.

3. Discussion

We have introduced `alevin-fry` as an accurate, computationally efficient, and lightweight framework for the processing of both single-cell and single-nucleus RNA-seq data. Compared to both `STARsolo` and `kallisto|bustools`, `alevin`-fry is consistently the fastest of these tools and can process datasets, on average, in less than half the time taken by the other tools. At the same time, when taking advantage of its sparse index, `alevin`-fry can process both single-cell and snRNA-seq data using less than 8 GB of RAM. The *splici* reference, that we propose to use for all types of quantifications covered here, allows the application of a fast mapping method (pseudoalignment (3) with structural constraints) while largely avoiding the estimation of spurious gene expression that is observed when such approaches are applied only to the spliced transcriptome (9). This allows `alevin-fry` to quantify expression with considerably increased precision compared to other lightweight tools, while using appreciably less memory than `STARsolo`.

Moreover, coupling the *splici* reference with a UMI resolution method that is aware of the splicing status of different indexed targets, we introduce USA-mode quantification. This

unifies scRNA-seq, snRNA-seq and RNA velocity preprocessing using `alevin-fry`. At the same time, `alevin-fry` is highly configurable, providing flexibility to users at many stages of the preprocessing pipeline. For example, at the expense of a higher runtime (though not substantially increased peak memory usage), even more precise quantifications can be obtained by performing selective-alignment (25) instead of pseudoalignment (3) with structural constraints. Similarly, multiple options are provided for barcode (*i.e.*, cell) permit-list generation and UMI resolution. `Alevin-fry` can also be used for processing other types of experiments, such as spatial scRNA-seq data and feature barcoded scRNA-seq data, and we are maintaining a growing suite of tutorials at https://combine-lab.github.io/alevin-fry-tutorials/.

As sc/snRNA-seq technologies continue to rapidly develop, improving methods used to analyze the resulting data will require ongoing benchmarking of methods to identify the strengths of existing techniques and areas for improvement in future approaches. For example, a recent study by You et al. (26) evaluated many different pipelines for the preprocessing of UMI-based scRNA-seq data. Concordant with the current paper, You et al. (26) found `alevin-fry`'s performance to be excellent, both computationally and in terms of the accuracy and robustness of the resulting counts. However, they report that `alevin-fry` — at least when using pseudoalignment with structural constraints — and `kallisto|bustools` demonstrate a left-skew in the count distribution of pseudogenes and therefore may underestimate the abundance of transcripts labeled with this biotype. Studies such as these will help guide improvements to existing tools and the development of improved methods. Similarly, broad evaluations should be carried out for the quantification of snRNA-seq data and the evaluation of spliced and unspliced count estimates for purposes such as RNA velocity inference. Likewise, in addition to evaluating tools across various experimental samples, it will be useful for future studies to incorporate simulated data into their analysis (9). However, the current paucity of sequence-level simulators for UMI and droplet-based technologies (27) makes the extensive use of simulated data challenging.

While `alevin-fry` provides an efficient and flexible framework for processing many types of sc/snRNA-seq data, some current implementation limitations, and benchmarking studies such as that performed by You et al. (26), motivate future work. For example, the existing mapping and UMI resolution algorithms are likely not well-suited to long-read scRNA-seq data, though we would like to support such protocols in the future. Additionally, it will be useful to investigate what other reference sequences can be incorporated into the index, and what modifications to the mapping and UMI-assignment algorithms can be made to further improve quantification accuracy and robustness, specifically among challenging transcript biotypes such as pseudogenes. Finally, we believe there is likely room to improve UMI resolution methodologies further, to infer more accurate cell-level molecule counts by, for example, modeling biases in the data, accounting for the likelihood with which different complex UMI and gene-mapping scenarios may arise, and by sharing information across similar cells within a sample or even across distinct data modalities.

We believe that `alevin-fry` strikes a remarkable balance between the often-competing criteria of accuracy, performance, and flexibility, and that these characteristics make it

an appealing choice for preprocessing the rapidly-growing collection of high-throughput sc/snRNA-seq data.

4. Methods

`Alevin-fry` is a configurable framework for the processing of single-cell and single-nucleus RNA-sequencing sc/snRNA-seq data. It makes use of `salmon` (1) for basic barcode and UMI parsing and the mapping of the reads to the constructed reference index. The output of `salmon`, when configured to produce output for `alevin-fry`, is a RAD (Reduced Alignment Data) format file, which is a chunk-based, binary file optimized for machine parsing, that encodes the relevant information necessary for subsequent (post-mapping) processing of the data (Supplementary Section S1). `Alevin-fry` consumes the `salmon` output directory — containing the RAD file and other relevant meta-information about the sample — and processes the data in a number of steps. The main processing steps correspond to permit-list generation, RAD file collation, and finally, quantification of the collated RAD file. We describe the options provided by `alevin-fry` and further details of these specific steps below.

4.1. Constructing a reference index.

The `alevin-fry` workflow quantifies scRNA-seq data based on a reference index created by `salmon`. Here, discuss two types of reference sequences that can be used to construct such an index, and describe the relative advantages and disadvantages of these options. Regardless of the reference over which one decides to build an index, `salmon` makes use of the pufferfish (2) index, and a dense or sparse index variant can be constructed.

First, at least for the processing of scRNA-seq (not snRNA-seq) data, one might consider building a reference index over the spliced transcriptome. The main benefits of this approach are that it is simple, and the resulting index tends to be very small. For example, when using the spliced transcriptome extracted from the latest 10x Genomics version of GRCh38, the (dense) reference index is only ~ `700` MB, and the entire mapping and quantification procedure can be performed in ~ `3` GB of RAM.

However, while the frugal resource use of an index restricted to only the spliced transcriptome is appealing, it comes with potential drawbacks. The most significant drawback, perhaps, is that it results in substantial false-positive rates (*i.e.*, spuriously detected genes) (3). One likely mechanism is that in typical scRNA-seq experiments, some fraction of reads (in cases, up to ~ `25`%) derive from intronic or intergenic sequences rather than from exons (3). When these true sequences of origin are absent from the index, reads deriving from them may sometimes be spuriously assigned to a spliced transcript that shares some local sequence similarity with the true locus of origin. The degree to which such spurious assignment occurs also depends on the specifics of the algorithm used for mapping; for example, the problem is most pronounced when using pseudoalignment (4), followed by pseudoalignment with structural constraints, and is somewhat (but not fully) mitigated when using selective-alignment (3).

One alternative is to map to the genome directly, as is done by `Cell Ranger` and `STARsolo`. This allows consideration of all genomic loci when determining the appropriate mapping location for a read, and results in the elimination of the false positives that are induced by forcing reads to map only against the annotated transcriptome. While building such an index is quite comprehensive, the associated costs are that the index is inevitably larger, and the common alignment approaches for scRNA-seq data (both `Cell Ranger` and `STARsolo` are based on STAR (5) as their underlying aligner) require considerably more RAM during alignment. Further, these approaches require solving the spliced (rather than contiguous) alignment problem; while good solutions (like STAR and HISAT2 (6)) exist, this problem is more computationally intensive and lightweight approaches like quasi-mapping (7) and pseudoalignment (4) have not yet been adapted to the problem of spliced mapping.

We propose here an alternative middle ground, which is to align against a reference that indexes both the spliced transcriptome and the set of (collapsed) intron sequences that are likely to generate reads in a typical sc/snRNA-seq experiment. We employ a reference preparation algorithm to produce what we refer to as a *splici* (spliced + intronic) reference, representing a slight modification of references previously used for RNA velocity preprocessing (8, 9). Further details explaining how this reference is constructed are provided in Supplementary Section S2, and we have developed an R package, named `roe`, to automate this construction process. Unlike the spliced transcriptome alone, this index contains the intronic sequences that are likely to give rise to a non-trivial fraction of reads in a scRNA-seq experiment, and including these sequences allows one to properly resolve read origin and avoid the spurious mapping associated with mapping against the spliced transcriptome alone, similar to what is accomplished by decoy sequences in bulk RNA-seq quantification (10) (though different in execution, as the quantification method itself, and not just the mapping algorithm, is aware of these sequences). On the other hand, by indexing the spliced transcriptome and introns (with flanking sequence) directly, this reference does not require spliced alignment, and is therefore amenable to both fast contiguous alignment algorithms like selective-alignment (10) as well as lightweight approaches like pseudoalignment (4). Throughout this manuscript, we append the mitochondrial genes to the splici reference (see Supplementary Table 1). We also set the flank length as the read length minus 5, though the quantification results appear very robust to the specific flank length chosen (Supplementary Section S3). While the size of this reference is considerably larger than the spliced transcriptome alone, it is still smaller than the genome. For the index used by `alevin-fry`, a dense index for a recent human reference constructed in such a manner requires ~ 10 GB of RAM for mapping, while the sparse index requires only ~ `6.5` GB. We demonstrate below how mapping against this index addresses the shortcomings of mapping against just the spliced transcriptome, while retaining modest memory requirements.

4.2. Fragment mapping.

As with constructing a reference against which to map reads, multiple choices can be made as to `exactly` how fragments should be mapped to the reference. In `alevin-fry` there are two main options available, selective-alignment (10), and pseudoalignment (4) with structural constraints. The mapping of reads is performed using the `salmon alevin`

command with the `--rad` or `--sketch` flags, which instructs the program to produce a RAD file and other auxiliary files for subsequent processing with `alevin-fry`, rather than to quantify the data directly with `alevin` (11). Broadly, among the two mapping approaches, selective-alignment is more accurate but more computationally intensive. Fragments are mapped against the index using maximal `exact` matches between reads and indexed unitigs (uniMEMs) as seeds, which are then chained to determine a putative mapping score. Low-scoring putative mappings are discarded, and high-scoring mappings are validated using alignment scoring via dynamic programming, based on the banded, parallel implementation of minimap2 (12). All best-scoring alignments that are above a user-defined threshold are reported as valid alignments for the fragment. The explicit alignment scoring avoids the reporting of mappings where the locus having the best set of seed matches is not the locus having the best alignment. Likewise the discarding of alignments below the user-defined threshold ensures that fragments arising from some other origin that have no high-quality alignment in the indexed reference will not be reported and processed as valid mappings.

On the other hand, pseudoalignment with structural constraints, exposed via the `--sketch` flag, is very fast, but it does not validate mapping locations via alignment scoring. This approach first uses a custom implementation of pseudoalignment (4) to determine which k-mers from the fragment match different targets. Subsequently, the implied mappings are subjected to filtering by structural constraints requiring that the matches supporting the pseudoalignment are in a consistent orientation, are co-linear with respect to the read and the reference, and that the stretch (maximum distance between any pair of k-mers comprising the mapping) is not too large. While using a *splici* reference largely eliminates the problem of false-positive expression that has previously been reported (3) when using pseudoalignment-to-transcriptome approaches, enabling accurate quantification using this rapid approach, there are still some false-positive mappings that can only be properly eliminated with alignment scoring (*i.e.*, using selective-alignment).

4.3. Permit-list generation.

After the reads have been mapped to the target index, either using selective-alignment or pseudoalignment with structural constraints, the resulting RAD file is inspected to determine the set of cellular barcodes (CBs) that should be used for quantification. In scRNA-seq experiments, cell capture is imperfect, and thus some fraction of barcodes may correspond to droplets that failed to properly capture a cell (13). In this case, the fragments associated with these barcodes usually exhibit many fewer distinct Unique Molecular Identifiers (UMIs) mapped to target sequences in the index than barcodes corresponding to properly captured cells. Likewise, errors that occur during PCR amplification and sequencing can “corrupt” the sequence of a cellular barcode, so that the barcode observed in the sequenced fragment is different from that which was originally attached to the underlying molecule prior to sequencing.

`Alevin-fry`’s `generate-permit-list` command works to determine the set of CBs that will eventually be quantified, as well as to perform correction of likely-corrupted barcodes to the “true” barcode from which they derived. It exposes a number of different

strategies to determine the set of CBs that should be quantified. The currently supported strategies are `--force-cells`, `--expect-cells`, `--knee-distance`, `--unfiltered-pl` and `--valid-bc`. A description of all available methods is provided in the alevin-fry documentation (https://alevin-fry.readthedocs.io/en/latest/generate_permit_list.html). Here we briefly describe the `--knee-distance` and `--unfiltered`-pl strategies, since they are likely to be the most commonly employed by users of `alevin-fry`. This step is also used to apply orientation filtering to the mapped records. So, for example, in protocols where all fragments are expected to map to the reference in the forward orientation, fragments (and their associated barcodes) are only considered valid if at least one forward strand mapping exists. Finally, while `alevin-fry` does keep track of the number of unmapped reads corresponding to each barcode for quality control reporting purposes, only the mapped reads displaying each barcode are considered for the purposes of generating the permit-list.

Knee distance permit-list generation.—The knee distance filtering implemented in `alevin-fry` is a modified implementation of the strategy that is provided in the `UMI-tools` (14) software. It is an iterative knee-finding strategy that attempts to automatically determine the number of barcodes corresponding to high-quality cells by examining the frequency histogram of observed barcodes. Briefly, this method first counts the number of reads associated with each barcode, and then sorts the barcodes in descending order by their associated read count. It then constructs the cumulative distribution function (CDF) from this sorted list of frequencies. Finally, it applies an iterative algorithm to attempt to determine the optimal number of barcodes to include by looking for a “knee” in the CDF graph. The algorithm considers each barcode in the CDF where its x-coordinate is this barcode’s rank divided by the total number of barcodes (*i.e.*, its normalized rank) and the y-coordinate is the (normalized) cumulative frequency achieved at this barcode. It then computes the distance of this barcode from the baseline (defined by the start and end of the CDF). The initial knee is predicted as the point that has the maximum distance from the baseline. The algorithm is iterative, because experiments with many low-quality barcodes may predict too many valid barcodes using this method. Thus, the algorithm is run repeatedly, each time considering a prefix of the CDF from index 0 through the previous knee’s index times `5`. Once two subsequent iterations of the algorithm return the same knee point, the algorithm terminates. Once the set of “permitted” barcodes has been determined by this method, the reads that have barcodes not within this set are corrected against it by checking if they are within one edit of some barcode in the list; if so, they are attributed to that barcode.

Correcting to an unfiltered permit-list.—Some technologies, like 10x Chromium, provide a set of specific known and experiment-independent barcodes that will be a superset of the barcodes that should be observed in any given sample. This list of “possible” barcodes can be treated as a set of barcodes against which the observed barcodes can be corrected. The `--unfiltered-pl` option accepts as an argument a list of possible barcodes for the sample. When using this argument, the user may also pass the `--min-reads` argument to determine the minimum frequency with which a barcode must be seen in order to be retained. The algorithm used in this mode passes over the input records (mapped reads) and counts how many times each of the barcodes in the unfiltered permit-list occurs

exactly. Any barcode occurring at least min-reads times will be considered as a present cell. Subsequently, all barcodes that did not match a present cell will be searched (at an edit distance of up to 1) against the barcodes determined to correspond to present cells. If an initially non-matching barcode has a unique neighbor among the barcodes for present cells, it will be corrected to that barcode, but if it has no 1-edit neighbor, or if it has two or more 1-edit neighbors among that list (*i.e.*, its correction would be ambiguous), then the record is discarded. Of course, unfiltered count matrices constructed in this manner will contain many barcodes not corresponding to properly captured cells, and should be subjected to subsequent filtering prior to analysis.

In all cases, the result of the generate-`permit-list` step of `alevin-fry` is the creation of a correction map that specifies which barcodes are to be quantified, and how barcodes are to be corrected against this quantified set, as well as a census of the number of observed and valid fragments corresponding to each corrected barcode. The census information is used in the subsequent collation step to enable an efficient partitioning strategy for collating the records by corrected barcodes.

4.4. Collation of RAD files.

Once the permit-list and correction map have been generated, the initial RAD file must be *collated* by the corrected cellular barcodes; this is done using `alevin-fry`'s `collate` command. In this phase, all fragments to be quantified are grouped together such that those sharing the same barcodes appear contiguously in the file. This step of processing serves an analogous purpose as the sorting of BUS (15) files as done in `kallisto|bustools` (9). However, there are a few technical `differences` that relate to the way in which `alevin-fry` processes the collated RAD files and to the way in which RAD files are structured differently from BUS files.

First, the `collate` command, unlike sorting, does not induce a total order on the resulting records. Specifically, while records that pass filtering and have the same corrected barcode are guaranteed to occur contiguously within the resulting collated file, there is no specific or meaningful order between the segments of the file corresponding to individual corrected barcodes. Further, within the set of records corresponding to a corrected barcode, there is no ordering or collation among the UMIs. This is due, in part, to the fact that all records sharing the same corrected barcode will be present in memory at the same time during the quantification phase, as well as the fact that certain UMI resolution strategies apply edit-distance collapsing for which UMI collation is insufficient. This means that, in the general case, collation can potentially be implemented in a more computationally efficient manner than sorting, though in practice, multi-threaded sorting of fixed-sized records is very efficient. Eliminating the requirement of having sorted records also means that the collated records corresponding to each barcode can appear at whichever location in the collated output file is desired. This admits extra flexibility in how collation is performed. Specifically, corrected barcodes are assigned to roughly appear in order of descending frequency in the output collated RAD file. This means that the largest (and potentially slowest to process) cells will appear near the beginning of the collated RAD file. Since quantification itself is multi-threaded, this allows more efficient pipelining of quantification

among multiple threads. Since threading granularity happens at the level of individual cells (i.e., the records for the same cell will never be quantified by multiple threads at the same time), placing the largest cells early in the quantification phase means that one is unlikely to encounter a situation where large and complex cells are encountered late in processing and many threads remain work starved while processing for the large cell completes.

The collation strategy implemented in `alevin-fry` is a two-pass approach. First, each corrected barcode is assigned a bucket index; the input RAD file is parsed (in parallel by many worker threads), and each record is written to the bucket it is assigned based on its corrected barcode. This ensures that all records sharing the same corrected barcode are routed to the same bucket. Further, the bucket sizes are limited by a user-defined maximum record count to ensure that individual buckets can be fully loaded into memory while retaining an overall small memory profile. In a second pass, each bucket is read into memory and its records are locally collated. This is done by constructing an in-memory hash map mapping each corrected barcode in this bucket to the vector of records sharing this barcode. Subsequently, each locally collated chunk is appended to the output collated RAD file (and optionally compressed if the user passes the `--compress` flag). In the resulting RAD file, the number of chunks is equal to the number of cells to be quantified (*i.e.*, the number of corrected barcodes), and all of the records sharing the same corrected barcode appear consecutively in the file.

4.5. Quantification.

With the collated RAD file prepared, `alevin`-fry is able to quantify the count for each gene within each cell separately and in parallel via the `quant` command. As with the mapping and permit-list generation phase, a number of different UMI resolution strategies are implemented in `alevin-fry`. Here, we briefly describe those strategies — `cr-like` and `cr-like-em` — that currently support the unspliced, spliced and ambiguous (USA) quantification mode that is used throughout this manuscript. All results presented in this manuscript were computed using the `cr-like` UMI resolution strategy. As opposed to splice unaware quantification (which `alevin-fry` also supports), the USA quantification mode produces a count for each splicing status of each gene within each quantified cell. Additional resolution strategies are described in Supplementary Section S4.

The quantification for each cell is carried out independently and in parallel, so we explain the procedure, without loss of generality, for the records corresponding to an individual cell. First, read records are collated (in memory) by their corresponding UMI. For each UMI, the set of transcripts to which the read maps are projected onto the corresponding set of genes. This process is aided by the use of a three-element transcript to gene map. Each entry in the map contains the name of an individual target sequence from the *splici* reference, the corresponding gene to which this target belongs, and a splicing status, recorded as ‘S’ if the target derives from a spliced transcript and ‘U’ if it derives from intronic (unspliced) sequence. Each gene is assigned a pair of globally unique identifiers, one corresponding to all “spliced” variants of the gene and the other to the “unspliced” (intronic) sequences for the gene. The gene-level identifiers corresponding to a given record are sorted and deduplicated. All records corresponding to the current UMI are iterated in the same fashion,

and a count is kept of how many times the UMI is associated with a read that maps to each gene identifier (with “spliced” and “unspliced” identifiers treated as distinct).

After all occurrences of the UMI are observed, the UMI is assigned to the gene with the largest frequency. If there is no unique gene with the highest frequency of occurrence, then the UMI is discarded if the `cr-like` resolution strategy is being used. On the other hand, if the `cr-like`-em resolution strategy is being used, a gene-level equivalence class is formed from all gene identifiers having the highest frequency of mapping for this UMI. Each identifier in the label of the equivalence class comprises a gene and a splicing status. The status is ‘U’ if only the unspliced identifier of this gene is among the most frequent mapping targets for this UMI, it is ‘S’ if only the spliced identifier is among the most frequent, and if both the unspliced and spliced identifiers of this gene are among the most frequent mapping targets for this UMI, then the status is ‘A’ (ambiguous). The UMI is attributed to this equivalence class, and an expectation maximization (EM) algorithm, like that employed in `alevin` (11), is subsequently used to probabilistically allocate counts to specific gene and splicing status pairs in the resulting count vector for this cell.

Under both of these resolution strategies, the resulting count matrix contains a count not just for each gene within each cell, but the count is further distributed over each gene’s splicing status (confidently assigned to spliced molecules from the gene, confidently assigned to unspliced molecules from the gene, or ambiguous in splicing status). Depending upon the type of data analysis being performed, this count matrix can then be used to directly extract the counts of interest. For example, if performing a “standard” single-cell gene expression analysis, one can extract the spliced and ambiguous counts for each gene in each cell and sum them together to produce the equivalent of a standard count matrix. If performing quantification on a single-nucleus RNA-seq sample, the counts from all splicing categories can be summed to produce the total UMI count attributed to each gene. For a RNA velocity analysis, the spliced and unspliced counts can be separated into distinct matrices and provided to a downstream RNA velocity computation tool (16, 17).

These resolution strategies thus provide a convenient solution for quantification of gene expression in a variety of different single-cell settings. The same processing approach can be used for the quantification of gene expression in single-cell experiments, or in single-nucleus experiments, or even to provide the input for RNA velocity analysis. At the same time, explicitly accounting for the unexpected origin of reads (*e.g.*, from intronic sequence in single-cell experiments) can also greatly mitigate spurious detection of genes exhibited by methods that restrict mapping or alignment to only the spliced transcriptome. This is possible as these resolution strategies implemented by `alevin-fry` are designed to infer both the gene and splicing status of the underlying fragments, but leave the determination of how to combine or aggregate UMIs arising from different splicing statuses to downstream analysis. We provide the function `loadFry` in the `Fishpond` (18) R package for flexibly processing `alevin-fry`’s result.

Finally, a number of additional and even more sophisticated resolution methods (namely `parsimony` and `parsimony-em`) are present in `alevin-fry` but not yet exposed under USA-mode. These implement variants on the original UMI resolution algorithm introduced

by Srivastava et al. (11) that applies a parsimony condition to approximately determine the minimal set of transcripts that could give rise to the observed set of UMIs. These alternative methods are further described in Supplementary Section S4. We are currently working on adapting these algorithms so that they can also be meaningfully applied in `alevin-fry`'s USA mode.

4.6. Additional preprocessing and filtering details for a *zebrafish* pineal gland experiment.

Different UMI resolution modes of `STARsolo`.—In Section 2.3 we explored the effect of making use of different UMI resolution modes that are exposed by `STARsolo`. Here, we briefly enumerate the modes we evaluated, and summarize their behavior. The default UMI resolution approach of `STARsolo` is `1MM`. This applies an iterative collapse of barcodes mapping to the same gene and separated by a single mismatch; this approach is designed to replicate the behavior of `Cell Ranger`. The second UMI resolution strategy we evaluated is labeled by `STARsolo` as `1MMDir`. This strategy is based on the directional algorithm introduced by Smith et al. (14). It builds a directed graph that takes into account both the mismatch distance between barcodes mapping to the same gene, as well as the relative frequency of these barcodes, and then applies a greedy algorithm to resolve the vertices in the graph into a set of inferred, distinct, UMIs in the original sample prior to PCR amplification and sequencing. Finally, we evaluated the `exact` UMI deduplication strategy, which only deduplicates UMIs that map to the same gene and that have identical UMI sequences. While the first of these strategies is designed to replicate the behavior of `Cell Ranger`, the latter two are not available in `Cell Ranger`, and therefore are not considered by Shainer and Stemmer (19).

Cell and feature filtering to reproduce Shainer and Stemmer (19).—In order to estimate the high-confidence cells from empty droplets, the `barcodeRank` function of `DropletUtils` (13) was applied on the count matrices with a lower bound set to `500` to determine the `inflection` point on the UMI count of barcodes. Barcodes that have a UMI count below the inflection point were regarded as empty droplets, and were filtered from the count matrices. The filtered count matrices were then used to create a `Seurat` object using the `CreateSeuratObject` function with thresholds `min.cells = 3` and `min.feature = 200`. These thresholds were used throughout the subsequent analyses for the zebrafish dataset.

Cell and feature filtering using `emptyDrops`.—The `emptyDrops` function in the `DropletUtils` (13) R package implements a procedure explicitly designed to model the ambient background distribution of expression and select, with some user-selected false discovery rate (FDR), the barcodes corresponding to high-quality cells. To select cells under this filtering scheme, we filtered the barcodes using `emptyDrops` with the default parameter setting. Using these cells, we created a `Seurat` object using the `CreateSeuratObject` with `min.cells = 3` and `min.feature = 200`. Given this `Seurat` object, the subsequent filtering, clustering and marker detection procedures applied were the same as above, replicating the procedure of the Shainer and Stemmer (19) analysis.

Cell and feature filtering using `alevin-fry` knee filtering.—To apply cell barcode filtering using the knee-distance approach of `alevin-fry`, we first quantified the data with `alevin-fry`, using the knee-distance method to determine the permit-list. Subsequently, the unfiltered count matrices for each method were subset to include only the barcodes appearing in this permit-list. This filtering strategy is more conservative than those examined above (*i.e.*, fewer cells passed filtering). Using these cells, we created a `Seurat` object using the `CreateSeuratObject` with `min.cells = 3` and `min.feature = 200`. Given this `Seurat` object, the subsequent filtering, clustering and marker detection procedures applied were the same as above, replicating the procedure of the Shainer and Stemmer (19) analysis.

4.7. Simulated data.

To compare the performance of different tools in terms of quantification accuracy, we selected a non-parametric simulation, which was introduced by Kaminow et al. (3). This simulation is seeded with the PBMC5k experiment (20). We used the simulated data in which reads derive from across the genome at realistic rates (*i.e.*, from introns, spliced transcripts, and intergenic sequences), but without the simulated gene-level multimapping. While this simulation does not tie to any parametric model, and therefore is likely to produce realistic mapping statistics, it is important to recall the caveat that simulated data often fail to recapitulate at least some important aspects of experimental data (10). This suggests that the performance on simulated data likely represents, in some sense, the upper bound of accuracy achievable by these methods on experimental data, and the degradation in performance of different methods may vary as the complexity of the data increases. Nonetheless, analyzing the accuracy of these tools under various metrics on this simulated data provides an important perspective on some potential strengths and shortcomings of different methods in a situation where the ground truth counts are known.

4.8. RNA velocity.

With the development of single-cell RNA-seq technologies, RNA velocity analysis has become increasingly popular. `Velocyto` (16) defines single-cell RNA velocity as the time derivative of the gene expression state, which is determined by the ratio of the spliced and unspliced molecule counts of each individual gene. By modeling transcriptional dynamics, RNA velocity can reveal cellular differentiation dynamics and developmental lineages present in a given single-cell experiment. `scVelo` (17) further enhances RNA velocity computation by eliminating the steady-state assumption made by `Velocyto`, and applying an expectation maximization method to solve the differentiation dynamics according to a series of master equations. The accurate and robust estimation of RNA velocity remains an active and exciting area of research.

To explore preprocessing for RNA velocity analysis, we make use of a mouse pancreatic endocrinogenesis dataset introduced by Bastidas-Ponce et al. (21) and used as an example dataset in the `scVelo` python package. This experiment is obtained with the 10x Genomics Chromium Single Cell 3’ Reagent Kit v2 and the read length is 151 nt. To utilize the cell state annotation information provided in `scVelo` example dataset, only the `3, 696` cells that are included in the `scVelo` example dataset are included in our analysis. The quantified

cells are all from stage E15.5. The processing was performed on raw FASTQ files retrieved from the Gene Expression Omnibus (GEO), accession number GSM3852755.

Following the preprocessing steps adopted by `scVelo`, we downloaded the pre-built mouse *mm10 v2.1.0* reference from 10x Genomics. To obtain the appropriate input for RNA velocity analysis with `alevin-fry`, we made use of USA-mode quantification, `kallisto|bustools` was run via the `kb_python` tool with the `--workflow lamanno` option, which results in the generation of two separate output matrices corresponding to the spliced and the unspliced counts, and `STARsolo` was run with the `--soloFeatures Gene Velocyto` option.

Depending on the RNA velocity method being used, ambiguous counts (which are output separately by `STARsolo` and `alevin-fry`) should either be provided explicitly, or allocated among the spliced and unspliced counts (or discarded entirely). We tested seven different strategies to process the ambiguous counts, which are for each gene within each individual cell, (i) discarding the ambiguous count, (ii) regarding the ambiguous count as spliced, (iii) regarding the ambiguous count as unspliced, (iv) evenly distributing the ambiguous count to spliced and unspliced, (v) dividing the ambiguous count by the ratio of confidently spliced count to the confidently unspliced count, (vi) dividing the ambiguous count by the ratio of not-unspliced (spliced + ambiguous) to unspliced, and (vii) dividing the ambiguous counts by the ratio of spliced to not-spliced(unspliced + ambiguous). We discuss the results of approach (ii) in Section 2.4, and provide all other results in Supplementary Section S7.

`ScVelo` version 0.2.3 is used to analyze RNA velocity under a python 3.8.5 environment. Cells whose cell barcode is in the `scVelo` example dataset are kept for further analysis. The pre-defined cell type and UMAP representation of each cell are obtained from the `scVelo` example dataset. The count matrices generated by all three methods are processed as described in Bergen et al. (17). Specifically, the count matrices are median normalized, only the top `2, 000` variable genes are kept, the first- and second-order moments of the normalized spliced and unspliced counts of each gene are calculated, and the reaction rates and latent variables are recovered. RNA velocity is estimated using the dynamical mode, and the directional flow of the estimated velocity is visualized in the predefined UMAP (22) embedding.

4.9. Clustering analysis of snRNA-seq data.

To evaluate the process of quantifying an snRNA-seq dataset using these preprocessing tools, we performed a cell type clustering analysis for the E14.5 samples from a snRNA-seq mouse placenta dataset (23) using `Seurat` 4.0.1 (24) under an `R` 4.0.5 environment. The nuclei were captured with the Chromium Single Cell 3’ Reagent V3 Kit from 10x Genomics, and the read length is 150nt. These raw reads were accessed from GEO under accession code GSM4609872. When analyzing single-nucleus RNA-seq data, we sum the unspliced (U), spliced (S), and ambiguous (A) counts returned by `alevin-fry` to get the overall count of each gene within each cell. Likewise, `kallisto|bustools` is run via the `kb_python` tool with the `--workflow nucleus` option specified, and `STARsolo` is run with the `--soloFeatures GeneFull` option.

To compare the results from different quantification tools in the snRNA-seq setting under a consistent and robust barcode filtering approach, we implemented the `emptyDrops_CR` functionality of `STARsolo` in `R`. The `emptyDrops_CR` filtering method is, itself, reverse engineered from the hybrid filtering strategy of `Cell Ranger`, which combines filtering based on various thresholds with the statistical testing method introduced by Lun et al. (13). This functionality is now included in the `DropletUtils` R package as the `emptyDropsCellRanger` function. This method can help to avoid the large number of relatively low-quality barcodes that we observed to pass the filtering of `emptyDrops` in snRNA-seq data. The hybrid method makes use of specific thresholds to control the size of the candidate pool of high-quality cells. We applied this function to remove putative empty droplets from the results of all tools under the same setting, which is the default setting in `STARsolo`. Only barcodes (cells) with FDR-adjusted p-value less than `0.01` of arising from non-empty droplets, with mitochondrial count percentage less than `0.25`%, and with `500 − 4,000` expressed genes, were kept for further clustering analysis.

After filtering empty droplets, the RNA counts of each nucleus were log-normalized. Next, the top `2, 000` variable genes were selected and their gene counts were scaled and used in the following steps. Then, PCA was performed with these variable genes, and a subset of significant PCs was selected using the `JackStraw` algorithm implemented in `Seurat`. Using these significant PCs, t-SNE dimensionality reduction (25) was performed, the nearest neighbor graph was constructed, and clustering was performed. In order to assign a cell type to each cluster, the R object provided by Marsh and Blelloch (23) was used as the reference to transfer the cell type annotation from the reference samples to the query object according to the anchor genes determined by the significant PCs using the `FindTransferAnchors` function of `Seurat` (see Supplementary Section S9).

The nuclei assigned as trophoblast were then selected to explore the trophoblast subclusters. Similar to the previous procedure, clusters were found using a subset of significant PCs determined by a `JackStraw` plot, and cell types were learned from the trophoblast R objected provided in the supplementary file of Marsh and Blelloch (23).

4.10. Details of time and memory benchmarking.

All experiments were conducted on a server with an Intel Xeon CPU (E5–2699 v4) with 44 cores and clocked at 2.20 GHz, 512 GB of memory, and 8 (non-RAID) 3.6 TB Toshiba MG03ACA4 HDDs. Samples were processed using a Nextflow (26) workflow. All tools tested here provide multi-threaded capabilities and were run with 16 threads. Experiments were run using `STARsolo` version 2.7.9a, `salmon` v1.5.1, `alevin-fry` v0.4.0, `kb_python` 0.26.0 (with `kallisto` 0.46.2 and `bustools` 0.40.0). `STARsolo` was run with the `--soloFeatures Gene` option to process single-cell samples, with the `--soloFeatures GeneFull` option to process single-nucleus samples, and with the `--soloFeatures Gene Velocyto` option to process samples for RNA velocity analysis. `STARsolo` offers the ability to use either a dense or sampled suffix array index. Here, all tests were run using the dense suffix array, which provides the fastest runtime but which also requires more memory. If memory is at a greater premium, users can instead choose to use the sparse index, which reduces the index size by a factor of ~ `2` but which requires ~ `1.7` times as long for

processing, on average (3). `Kallisto|bustools` was run using the `kb_python` wrapper with `--workflow standard` used for single-cell samples, `--workflow nucleus` used for single-nucleus samples and `--workflow lamanno` used to process samples for RNA velocity analysis. `Alevin-fry` was run in USA-mode on all samples using the `--cr-like` UMI resolution method and the appropriate counts were extracted from the resulting count matrix depending upon the sample type. `Alevin-fry` was tested with both the sparse and dense index as well as using both the unfiltered permit-list and filtering of barcodes prior to quantification using the knee-distance method.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgements

This work is supported by the National Institute of Health under award number R01HG009937 to R.P. and K99CA267677 to A.S., and the National Science Foundation under award number CCF-1750472 to R.P., and CNS-1763680 to R.P. Also, this project has been made possible in part by grant number CZIF2020–004893 from the Chan Zuckerberg Initiative Foundation to R.P. The funders had no role in the design of the method, data analysis, decision to publish, or preparation of the manuscript.

Data availability

All data analyzed in this paper are publicly available. The mouse pancreas dataset, the mouse placenta dataset and the zebrafish pineal dataset analyzed during the current study are available on NCBI Gene Expression Omnibus, under accession number GSM3852755 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM3852755), GSM4609872 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM4609872) and GSM3511193 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM3511193), respectively. The PBMC5k and PBMC10k dataset are available from 10x Genomics at https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.2/5k_pbmc_v3 and https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.0/pbmc_10k_v3, respectively. The description of the references that were used for mapping the sequencing reads from each dataset can be found in Supplementary Table 1.

References

1. Svensson Valentine, Veiga Beltrame Eduardo da, and Pachter Lior. A curated database reveals trends in single-cell transcriptomics. Database, 2020, 2020.
2. Li Bo, Ruotti Victor, Stewart Ron M, Thomson James A, and Dewey Colin N. RNA-Seq gene expression estimation with read mapping uncertainty. Bioinformatics, 26(4):493–500, 2010. [PubMed: 20022975]
3. Bray Nicolas L, Pimentel Harold, Melsted Páll, and Pachter Lior. Near-optimal probabilistic RNA-seq quantification. Nature Biotechnology, 34(5):525, 2016.
4. Patro Rob, Duggal Geet, Love Michael I, Irizarry Rafael A, and Kingsford Carl. Salmon provides fast and bias-aware quantification of transcript expression. Nature Methods, 14(4):417–419, 2017. [PubMed: 28263959]

5. Srivastava Avi, Malik Laraib, Smith Tom, Sudbery Ian, and Patro Rob. Alevin efficiently estimates accurate gene abundances from dscRNA-seq data. Genome Biology, 20(1):1–16, 2019. [PubMed: 30606230]
6. Niebler Stefan, Müller André, Hankeln Thomas, and Schmidt Bertil. RainDrop: rapid activation matrix computation for droplet-based single-cell RNA-seq reads. BMC Bioinformatics, 21(1):1–14, 2020. [PubMed: 31898485]
7. Páll Melsted, Booeshaghi A Sina, Liu Lauren, Gao Fan, Lu Lambda, Min Kyung Hoi Joseph, Beltrame Eduardo da Veiga, Hjörleifsson Kristján Eldjárn, Gehring Jase, and Pachter Lior. Modular, efficient and constant-memory single-cell RNA-seq preprocessing. Nature Biotechnology, pages 1–6, 2021.
8. Melsted Páll, Ntranos Vasilis, and Pachter Lior. The barcode, UMI, set format and BUStools. Bioinformatics, 35(21):4472–4473, 2019. [PubMed: 31073610]
9. Kaminow Benjamin, Yunusov Dinar, and Dobin Alexander. STARsolo: accurate, fast and versatile mapping/quantification of single-cell and single-nucleus RNA-seq data. BioRxiv, 2021. doi: 10.1101/2021.05.05.442755.
10. Shainer Inbal, Michel Maximilian, Marquart Gregory D., Bhandiwad Ashwin A., Zmora Nilli, Livne Zohar Ben-Moshe, Zohar Yonathan, Hazak Adi, Mazon Yael, Förster Dominique, Hollander-Cohen Lian, Cone Roger D., Burgess Harold A., and Gothilf Yoav. Agouti-related protein 2 is a new player in the teleost stress response system. Current Biology, 29(12):2009–2019.e7, June 2019. doi: 10.1016/j.cub.2019.05.021. [PubMed: 31178320]
11. Shainer Inbal and Stemmer Manuel. Choice of preprocessing pipeline influences clustering quality of scRNA-seq datasets. BMC genomics, 22(1):1–13, 2021. [PubMed: 33388042]
12. Hao Yuhan, Hao Stephanie, Andersen-Nissen Erica, Mauck William M., Zheng Shiwei, Butler Andrew, Lee Maddie J., Wilk Aaron J., Darby Charlotte, Zager Michael, Hoffman Paul, Stoeckius Marlon, Papalexi Efthymia, Mimitou Eleni P., Jain Jaison, Srivastava Avi, Stuart Tim, Fleming Lamar M., Yeung Bertrand, Rogers Angela J., McElrath Juliana M., Blish Catherine A., Gottardo Raphael, Smibert Peter, and Satija Rahul. Integrated analysis of multimodal single-cell data. Cell, 184(13):3573–3587.e29, June 2021. doi: 10.1016/j.cell.2021.04.048. [PubMed: 34062119]
13. Cau Elise, Ronsin Brice, Bessière Laurianne, and Blader Patrick. A notch-mediated, temporal asymmetry in BMP pathway activation promotes photoreceptor subtype diversification. PLOS Biology, 17(1):e2006250, January 2019. doi: 10.1371/journal.pbio.2006250. [PubMed: 30703098]
14. Lun Aaron TL, Riesenfeld Samantha, Andrews Tallulah, Gomes Tomas, Marioni John C, et al. EmptyDrops: distinguishing cells from empty droplets in droplet-based single-cell RNA sequencing data. Genome Biology, 20(1):1–9, 2019. [PubMed: 30606230]
15. Crespo Cátia, Soroldoni Daniele, and Knust Elisabeth. A novel transgenic zebrafish line for red opsin expression in outer segments of photoreceptor cells. Developmental Dynamics, 247(7): 951–959, April 2018. doi: 10.1002/dvdy.24631. [PubMed: 29603474]
16. Wada Seiji, Shen Baoguo, Kawano-Yamashita Emi, Nagata Takashi, Hibi Masahiko, Tamotsu Satoshi, Koyanagi Mitsumasa, and Terakita Akihisa. Color opponency with a single kind of bistable opsin in the zebrafish pineal organ. Proceedings of the National Academy of Sciences, 115(44):11310–11315, October 2018. doi: 10.1073/pnas.1802592115.
17. Altschul Stephen F., Gish Warren, Miller Webb, Myers Eugene W., and Lipman David J.. Basic local alignment search tool. Journal of Molecular Biology, 215(3):403–410, October 1990. doi: 10.1016/s0022-2836(05)80360-2. [PubMed: 2231712]
18. Brüning Ralf Schulze, Tombor Lukas, Schulz Marcel H., Dimmeler Stefanie, and John David. Comparative analysis of common alignment tools for single cell RNA sequencing. BioRxiv, 2021. doi: 10.1101/2021.02.15.430948.
19. Manno Gioele La, Soldatov Ruslan, Zeisel Amit, Braun Emelie, Hochgerner Hannah, Petukhov Viktor, Lidschreiber Katja, Kastriti Maria E., Lönnerberg Peter, Furlan Alessandro, Fan Jean, Borm Lars E., Liu Zehua, Bruggen David van, Guo Jimin, He Xiaoling, Barker Roger, Sundström Erik, Castelo-Branco Gonçalo, Cramer Patrick, Adameyko Igor, Linnarsson Sten, and Kharchenko Peter V. RNA velocity of single cells. Nature, 560(7719):494–498, August 2018. doi: 10.1038/s41586-018-0414-6. [PubMed: 30089906]

20. Bergen Volker, Lange Marius, Peidli Stefan, Wolf F. Alexander, and Theis Fabian J. Generalizing RNA velocity to transient cell states through dynamical modeling. Nature Biotechnology, 38(12): 1408–1414, August 2020. doi: 10.1038/s41587-020-0591-3.

21. Soneson Charlotte, Srivastava Avi, Patro Rob, and Stadler Michael B. Preprocessing choices affect RNA velocity results for droplet scRNA-seq data. PLOS Computational Biology, 17(1):e1008585, January 2021. doi: 10.1371/journal.pcbi.1008585. [PubMed: 33428615]

22. Marsh Bryan and Blelloch Robert. Single nuclei RNA-seq of mouse placental labyrinth development. ELife, 9, November 2020. doi: 10.7554/elife.60266.

23. Woods Laura, Perez-Garcia Vicente, and Hemberger Myriam. Regulation of placental development and its impact on fetal Growth—New insights from mouse models. Frontiers in Endocrinology, 9, September 2018. doi: 10.3389/fendo.2018.00570.

24. 10X Genomics. 10k peripheral blood mononuclear cells (PBMCs) from a healthy donor (v3 chemistry) https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.0/pbmc_10k_v3, 2018.

25. Srivastava Avi, Malik Laraib, Sarkar Hirak, Zakeri Mohsen, Almodaresi Fatemeh, Soneson Charlotte, Love Michael I, Kingsford Carl, and Patro Rob. Alignment and mapping methodology influence transcript abundance estimation. Genome Biology, 21(1):1–29, 2020.

26. You Yue, Tian Luyi, Su Shian, Dong Xueyi, Jabbari Jafar S., Hickey Peter F., and Ritchie Matthew E. Benchmarking UMI-based single-cell RNA-seq preprocessing workflows. Genome Biol 22, 339 (2021) [PubMed: 34906205]

27. Sarkar Hirak, Srivastava Avi, Patro Rob, *Minnow*: a principled framework for rapid simulation of dscRNA-seq data at the read level, Bioinformatics, Volume 35, Issue 14, July 2019, Pages i136–i144 [PubMed: 31510649]

References for the Methods Section

1. Patro Rob, Duggal Geet, Love Michael I, Irizarry Rafael A, and Kingsford Carl. Salmon provides fast and bias-aware quantification of transcript expression. Nature Methods, 14(4):417–419, 2017. [PubMed: 28263959]

2. Almodaresi Fatemeh, Sarkar Hirak, Srivastava Avi, and Patro Rob. A space and time-efficient index for the compacted colored de Bruijn graph. Bioinformatics, 34(13):i169–i177, 2018. [PubMed: 29949982]

3. Kaminow Benjamin, Yunusov Dinar, and Dobin Alexander. STARsolo: accurate, fast and versatile mapping/quantification of single-cell and single-nucleus RNA-seq data. BioRxiv, 2021. doi: 10.1101/2021.05.05.442755.

4. Bray Nicolas L, Pimentel Harold, Melsted Páll, and Pachter Lior. Near-optimal probabilistic RNA-seq quantification. Nature Biotechnology, 34(5):525, 2016.

5. Dobin Alexander, Davis Carrie A, Schlesinger Felix, Drenkow Jorg, Zaleski Chris, Jha Sonali, Batut Philippe, Chaisson Mark, and Gingeras Thomas R. STAR: ultrafast universal RNA-seq aligner. Bioinformatics, 29(1):15–21, 2013. [PubMed: 23104886]

6. Kim Daehwan, Paggi Joseph M, Park Chanhee, Bennett Christopher, and Salzberg Steven L. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. Nature Biotechnology, 37(8):907–915, 2019.

7. Srivastava Avi, Sarkar Hirak, Gupta Nitish, and Patro Rob. RapMap: a rapid, sensitive and accurate tool for mapping RNA-seq reads to transcriptomes. Bioinformatics, 32(12):i192–i200, 2016. [PubMed: 27307617]

8. Soneson Charlotte, Srivastava Avi, Patro Rob, and Stadler Michael B. Preprocessing choices affect RNA velocity results for droplet scRNA-seq data. PLOS Computational Biology, 17(1):e1008585, January 2021. doi: 10.1371/journal.pcbi.1008585. [PubMed: 33428615]

9. Páll Melsted A Booeshaghi Sina, Liu Lauren, Gao Fan, Lu Lambda, Joseph Min Kyung Hoi, Beltrame Eduardo da Veiga, Hjörleifsson Kristján Eldjárn, Gehring Jase, and Pachter Lior. Modular, efficient and constant-memory single-cell RNA-seq preprocessing. Nature Biotechnology, pages 1–6, 2021.

10. Srivastava Avi, Malik Laraib, Sarkar Hirak, Zakeri Mohsen, Almodaresi Fatemeh, Soneson Charlotte, Love Michael I, Kingsford Carl, and Patro Rob. Alignment and mapping methodology influence transcript abundance estimation. Genome Biology, 21(1):1–29, 2020.
11. Srivastava Avi, Malik Laraib, Smith Tom, Sudbery Ian, and Patro Rob. Alevin efficiently estimates accurate gene abundances from dscRNA-seq data. Genome Biology, 20(1):1–16, 2019. [PubMed: 30606230]
12. Li Heng. Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics, 34(18):3094–3100, 2018. [PubMed: 29750242]
13. Lun Aaron TL, Riesenfeld Samantha, Andrews Tallulah, Gomes Tomas, Marioni John C, et al. EmptyDrops: distinguishing cells from empty droplets in droplet-based single-cell RNA sequencing data. Genome Biology, 20(1):1–9, 2019. [PubMed: 30606230]
14. Smith Tom, Heger Andreas, and Sudbery Ian. UMI-tools: modeling sequencing errors in Unique Molecular Identifiers to improve quantification accuracy. Genome Research, 27(3):491–499, 2017. [PubMed: 28100584]
15. Melsted Páll, Ntranos Vasilis, and Pachter Lior. The barcode, UMI, set format and BUStools. Bioinformatics, 35(21):4472–4473, 2019. [PubMed: 31073610]
16. Manno Gioele La, Soldatov Ruslan, Zeisel Amit, Braun Emelie, Hochgerner Hannah, Petukhov Viktor, Lidschreiber Katja, Kastriti Maria E., Lönnerberg Peter, Furlan Alessandro, Fan Jean, Borm Lars E., Liu Zehua, Bruggen David van, Guo Jimin, He Xiaoling, Barker Roger, Sundström Erik, Castelo-Branco Gonçalo, Cramer Patrick, Adameyko Igor, Linnarsson Sten, and Kharchenko Peter V. RNA velocity of single cells. Nature, 560(7719):494–498, August 2018. doi: 10.1038/s41586-018-0414-6. [PubMed: 30089906]
17. Bergen Volker, Lange Marius, Peidli Stefan, Wolf F. Alexander, and Theis Fabian J. Generalizing RNA velocity to transient cell states through dynamical modeling. Nature Biotechnology, 38(12):1408–1414, August 2020. doi: 10.1038/s41587-020-0591-3.
18. Zhu Anqi, Srivastava Avi, Ibrahim Joseph G, Patro Rob, and Love Michael I. Non-parametric expression analysis using inferential replicate counts. Nucleic Acids Research, 47(18):e105–e105, August 2019. doi: 10.1093/nar/gkz622. [PubMed: 31372651]
19. Shainer Inbal and Stemmer Manuel. Choice of preprocessing pipeline influences clustering quality of scRNA-seq datasets. BMC genomics, 22(1):1–13, 2021. [PubMed: 33388042]
20. 10X Genomics. 5k peripheral blood mononuclear cells (PBMCs) from a healthy donor (v3 chemistry) https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.2/5k_pbmc_v3, 2019.
21. Bastidas-Ponce Aimée, Tritschler Sophie, Dony Leander, Scheibner Katharina, Tarquis-Medina Marta, Salinno Ciro, Schirge Silvia, Burtscher Ingo, Böttcher Anika, Theis Fabian, Lickert Heiko, and Bakhti Mostafa. Massive single-cell mRNA profiling reveals a detailed roadmap for pancreatic endocrinogenesis. Development, January 2019. doi: 10.1242/dev.173849.
22. Becht Etienne, McInnes Leland, Healy John, Dutertre Charles-Antoine, Kwok Immanuel WH, Ng Lai Guan, Ginhoux Florent, and Newell Evan W. Dimensionality reduction for visualizing single-cell data using UMAP. Nature Biotechnology, 37(1):38–44, 2019.
23. Marsh Bryan and Blelloch Robert. Single nuclei RNA-seq of mouse placental labyrinth development. ELife, 9, November 2020. doi: 10.7554/elife.60266.
24. Hao Yuhan, Hao Stephanie, Andersen-Nissen Erica, Mauck William M., Zheng Shiwei, Butler Andrew, Lee Maddie J., Wilk Aaron J., Darby Charlotte, Zager Michael, Hoffman Paul, Stoeckius Marlon, Papalexi Efthymia, Mimitou Eleni P., Jain Jaison, Srivastava Avi, Stuart Tim, Fleming Lamar M., Yeung Bertrand, Rogers Angela J., McElrath Juliana M., Blish Catherine A., Gottardo Raphael, Smibert Peter, and Satija Rahul. Integrated analysis of multimodal single-cell data. Cell, 184(13):3573–3587.e29, June 2021. doi: 10.1016/j.cell.2021.04.048. [PubMed: 34062119]
25. Maaten Laurens van der and Hinton Geoffrey. Visualizing data using t-sne. Journal of Machine Learning Research, 9(86):2579–2605, 2008.
26. Tommaso Paolo Di, Chatzou Maria, Floden Evan W, Barja Pablo Prieto, Palumbo Emilio, and Notredame Cedric. Nextflow enables reproducible computational workflows. Nature Biotechnology, 35 (4):316–319, 2017.

27. He Dongze, Zakeri Mohsen, Sarkar Hirak, Soneson Charlotte, Srivastava Avi, and Patro Rob. Alevin-fry v0.4.0 for manuscript "`alevin-fry` unlocks rapid, accurate, and memory-frugal quantification of single-cell rna-seq data"., 2021 https://zenodo.org/record/5806834#.Yio8TxDMKBQ.

28. Grüning Björn, Dale Ryan, Sjödin Andreas, Chapman Brad A, Rowe Jillian, Tomkins-Tinch Christopher H, Valieris Renan, and Köster Johannes. Bioconda: sustainable and comprehensive software distribution for the life sciences. Nature Methods, 15(7):475–476, 2018. [PubMed: 29967506]

29. He Dongze, Zakeri Mohsen, Sarkar Hirak, Soneson Charlotte, Srivastava Avi, and Patro Rob. Additional data for manuscript "`alevin-fry` unlocks rapid, accurate, and memory-frugal quantification of single-cell rna-seq data", 2021 https://zenodo.org/record/5799568#.Yio8ehDMKBQ.

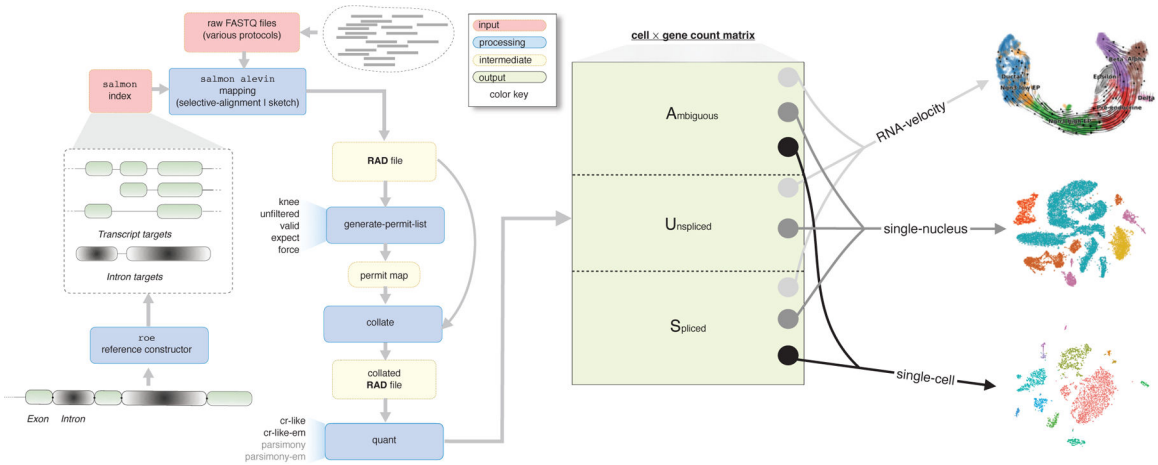

Fig. 1 – 2-column figure. Overview of the `alevin-fry` pipeline

(operating in unspliced, spliced, ambiguous (USA) quantification mode). The arrows highlight the flow of data through the pipeline, whose output is a matrix specifying the expected counts of each of the considered splicing states of each gene within each quantified cell.

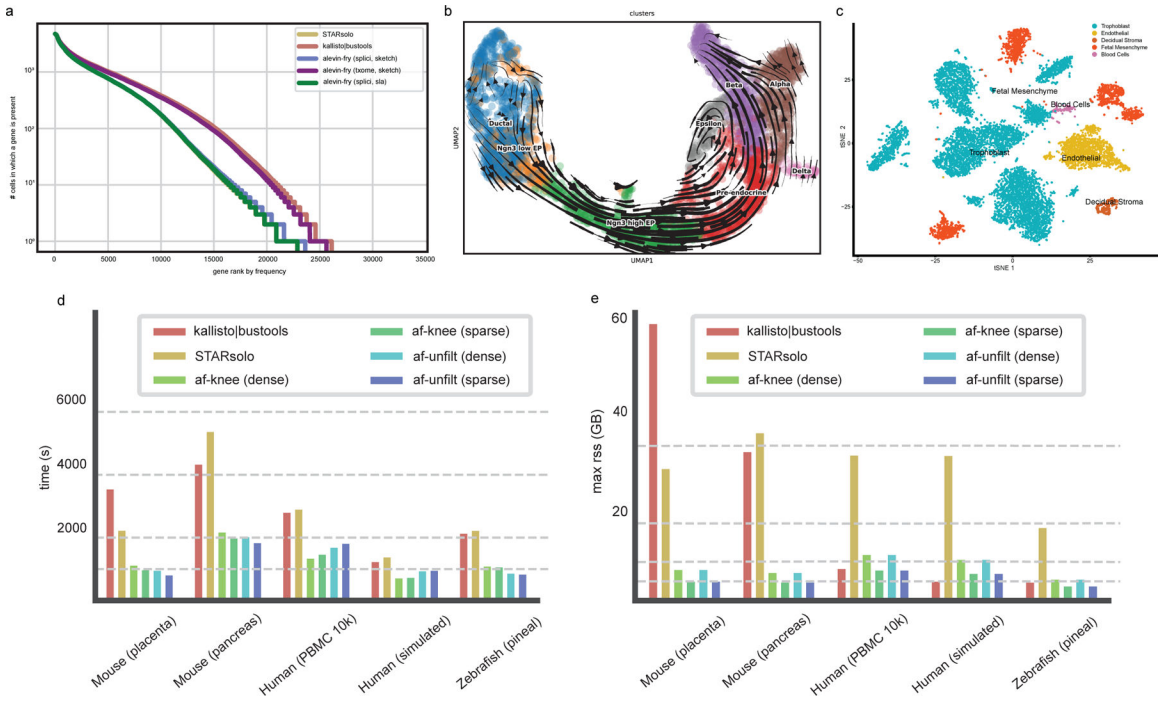

Fig. 2 – 2-column, 5-panel figure. Comprehensive analysis of the performance pf `alevin-fry` on real and simulated datasets.

(a) The frequency distribution of the presence of genes across all shared cells for `STARsolo`, `kallisto|bustools`, and `alevin-fry` (including multiple index types for `alevin-fry`) on the simulated data. Different color lines represent the quantification methods. Within the variants of `alevin-fry, txome` stands for transcriptome reference (i.e., just indexing the annotated, spliced, transcriptome), and `sketch` (pseudoalignment with structural constraints) and `sla` (selective-alignment) label the mapping method used to obtain the result. Due to the similarity of the distributions, the line of `STARsolo` is occluded by the line of `alevin-fry(splici, sla)`.

(b) A visualization of the velocity estimation derived from `alevin-fry` counts in a UMAP-based embedding after assigning all ambiguous counts as spliced; the streamlines represent the direction of RNA velocity estimated by `scVelo`. Points (cells) are colored according to the cell type annotation.

(c) The t-SNE embedding plot of an `alevin`-fry processed mouse placenta snRNA-seq dataset. The color of each nucleus represents the inferred cell-type annotation, which was learned from a reference dataset.

(d) and (e) are timing and peak memory usage for all tools (run with 16 threads) on the different datasets evaluated in this manuscript. The x-axis of (d) and (e) represents the evaluated datasets. The y-axis of (d) represents the run time of each tool, measured in seconds. The y-axis of (e) denotes the peak memory usage — measured as the maximum resident set size (max rss) — during the execution of each tool. Dashed horizontal lines in (d) denote 15 minutes, 30 minutes, 60 minutes and 90 minutes, respectively. Dashed horizontal lines in (e) denote 4GB, 8GB, 16GB and 32GB, respectively.

Table 1:

The performance of the examined tools on the simulated data. Each row lists a different quantification method being evaluated. Among the variants of `alevin-fry, txome` stands for mapping against the spliced transcriptome reference, `splici` stands for mapping against the splici reference, and `sketch` (pseudoalignment with structural constraints) and `sla` (selective-alignment) describe the mapping method. Each column lists a metric. They are, from left to right, the mean cell-level Spearman correlation of gene abundances, the mean absolute relative deviation (MARD) where `NA` values are dropped and treated as zero, and the mean relative false-positive and false-negative expression per cell. Detailed definitions are in Supplementary Section S8. All metrics are measures on the subset of genes and cells defined by all tested methods, and are taken with respect to ground-truth abundances.

method	mean Sp. corr.	MARD (drop NA)	MARD (NA=0)	mean rFP/cell	mean rFN/cell
`STARsolo`	0.997	0.031	0.002	0.001	0.005
`kallisto|bustools`	0.864	0.263	0.024	0.328	0.006
`alevin-fry` (txome, sketch)	0.883	0.226	0.020	0.273	0.006
`alevin-fry` (splici, sketch)	0.988	0.026	0.002	0.011	0.012
`alevin-fry` (splici, sla)	0.992	0.019	0.001	0.004	0.011