Learn more about alevin and alevin-fry!

# Accurate, efficient, and uncertainty-aware expression quantification of single-cell RNA-seq data

Hirak Sarkar[1], Avi Srivastava[2], Mohsen Zakeri[1], Scott K Van Buren[3], Naim U Rashid[3], Michael I Love[3], Rob Patro[1]

[1]University of Maryland, Computer Science, Maryland, MD, [2]New York University, Center for Genomics and Systems Biology, New York City, NY, [3]University of North Carolina-Chapel Hill, Department of Biostatistics, Chapel Hill, NC, University of North Carolina-Chapel Hill, Department of Genetics, Chapel Hill, NC

## Background

The rapid growth in the generation of single-cell RNA-seq (scRNA-seq) data highlights the need for scalable computational platforms to extract useful information from this data, such as gene expression estimates and the corresponding uncertainty information, that can be used in downstream applications. We present a flexible time- and memory-efficient framework for processing various types of scRNA-seq data that accounts for multi-mapping sequencing reads and that estimates the quantification uncertainty inherent in the gene counts. This uncertainty arises from gene-ambiguous UMIs that are particularly problematic, as they tend not to arise randomly, but instead arise preferentially from sequence-similar gene families. Alevin, and the new extension, alevin-fry, support a principled approach for estimating this expression uncertainty using a cell-level bootstrapping procedure. Alevin-fry is a framework that processes the mapping information generated by alevin, constructs an intermediate cell-level representation of the interactions among reads, UMIs, and genes called parsimonious UMI graphs (PUGs), and exposes multiple strategies, ranging from trivial to sophisticated, for resolving PUGs into gene-level counts. We observe that alevin and alevin-fry are capable of processing tagged-end single-cell data accurately, quickly and with very low memory requirements (usually ~2GB). Alevin is written in C++14 and is available as part of salmon at https://github.com/COMBINE-lab/salmon. Alevin-fry is written in Rust and is available at https://github.com/COMBINE-lab/alevin-fry.

## Efficient single cell Data Storage format

A typical single-cell preprocessing pipeline generates a cell-by-gene count matrix which is used for virtually all downstream analyses, for example differential expression studies, pseudotime analysis etc. Matrix market format (mtx) is a standard format to store the sparse count matrices. However, being a textual representation, the mtx is not particularly compact, though compressing the file helps. The characteristics of these matrices — sparsity and typically low magnitude counts — can be exploited to design more efficient formats for storage and transfer. We introduce the EDS (efficient data storage) format which is not only smaller in size compared to mtx, it is also faster to load and requires less memory. The EDS format matrices can be read into R directly using the fishpond [5] package.

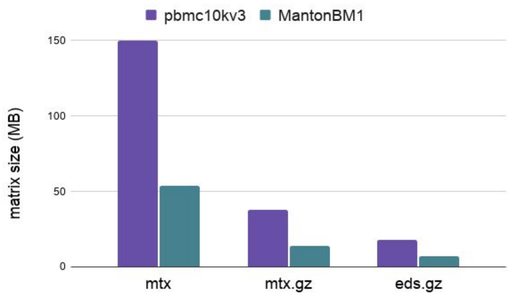


**Figure 1:** The size of the the gene x cell count matrix for the MantonBM1 (~4k cells, chromimum v2 chemistry) and pbmc 10k (~10k cells, chromium v3 chemistry) datasets in matrix market (coordinate) format, gzipped matrix market (coordinate) format and EDS (efficient data storage) format.

## Alevin-fry pipeline overview

Alevin-fry (Figure 2) provides an efficient foundation for memory-constrained, highly-parallel, raw data processing and UMI resolution. It exposes multiple (simple & complex) resolution algorithms that can be paired with alevin or other upstream alignment tools via BAM conversion.

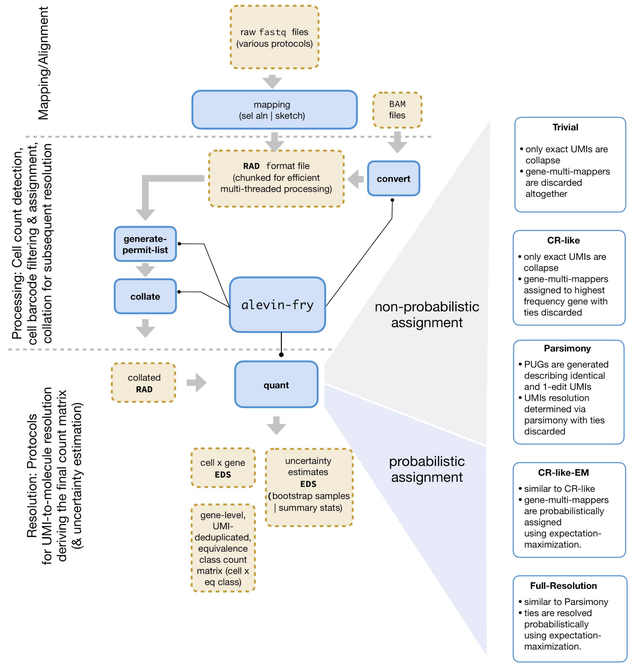


**Figure 2:** An overview of the alevin-fry pipeline showing the various ways in which the data can be processed. Each yellow box shows an input / output (some outputs are inputs to subsequent stages of the pipeline), while each blue box represents a processing component of the pipeline. The pipeline can be roughly divided into the three phases of mapping / alignment, pre-processing and resolution into a quantification matrix. The quantification matrix itself can be just of estimated gene counts, or can also include inferential replicates. It is also possible to produce a count matrix of UMI-deduplicated gene-level equivalence classes by cells, representing the gene-level mapping ambiguity directly, perhaps after parsimony or frequency-based resolution, rather than the estimated per-gene UMI counts.

## Concordance of quantification between methods

There's agreement across methods, yet substantial differences exist. Selective-alignment is most concordant with STARSolo, particularly when paried with simple resolution algorithms. Probabilistic assignment of multimappers — rather than discarding — generally reduces concordance (as may be expected since STARSolo discards these); the effect is particularly pronounced when paired with sketch mapping.

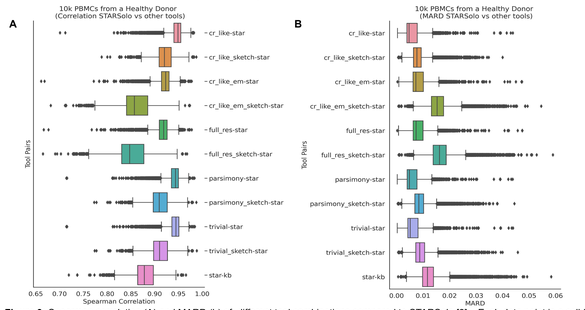


**Figure 3:** Spearman correlation (A) and MARD (b) of different tool combinations compared to STARSolo [3] . Each data point is a cell-level measurement over all genes within that cell. The cells considered were those in the intersection of cells quantified by all approaches.

## Time and memory resource usage of different single-cell methods

The time and memory requirements for different tools when processing the PBMC 10k (chromium v3) dataset (Figure 4 A, B). When all tools are given 16 threads, alevin-fry with selective-alignment takes the most time to complete and alevin-fry in sketch mode the least — STARSolo [3] is the second fastest tool, but at a lower thread count would become slower than kallisto. Alevin-fry (depending on resolution algorithm) uses a maximum of 2-3G of RAM; kallisto uses 9-10G (depending on if the kb-tools wrapper is used), and STARSolo uses the most memory at ~32G.

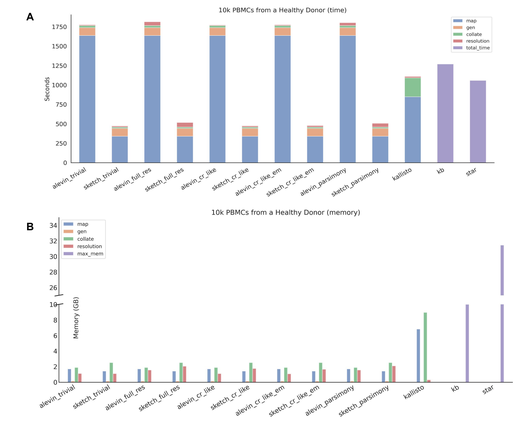


**Figure 4 :** The time (A) and memory (B) resource usage of different pipelines when processing the PBMC10k (chromium v3 chemistry) dataset using 16 threads. In panel B, note the break in the y-axis at 10-26G.

Figure 5 shows how the runtime of different methods scales with respect to the number of threads used for mapping. Since alignment and selective-alignment [6] based methods are compute-bound, they continue to scale to large numbers of threads, eventually approaching the other methods. Kallisto [4] and alevin [2] in sketch mode reach minimal runtimes at ~4 and ~12 threads respectively, with alevin sketch taking about 43% of the time of kallisto.

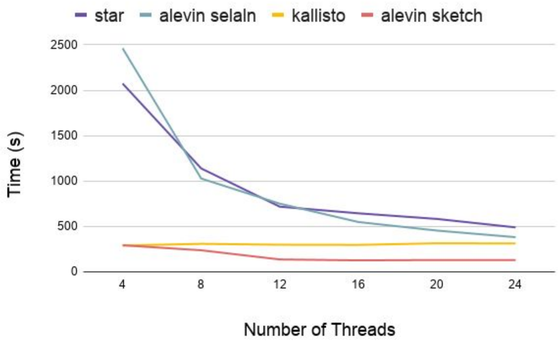


**Figure 5:** The effect of using different numbers of threads on the mapping time of different tools. STAR and alevin using selective-alignment are compute-bound and continue to scale to 24 threads. We start the scaling where the first of the tested tools saturates its performance on this dataset; kallisto mapping time staturates at 4 threads and remains relatively constant at around 300s. In sketch mode, alevin mapping time reduces up to 12 threads, and then remains relatively constant at around 130s, at which point data ingestion limits seem to become the bottleneck. All timing was taken using a warm cache.

## Quantification uncertainty in downstream analysis

Just like accounting for uncertainty in *transcript-level* quantification in bulk RNA-seq can improve transcript-level differential testing, accounting for uncertainty in *gene-level* quantification in tagged-end, single-cell RNA-seq can improve downstream analyses, like differential testing of genes between divergent branches of a trajectory (Figure 6).

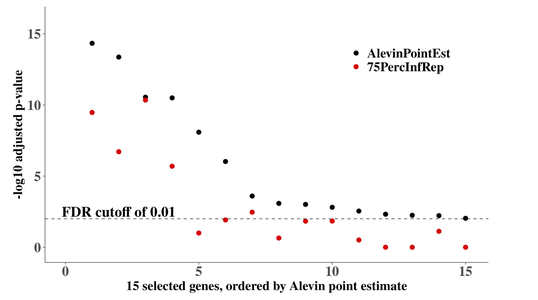


**Figure 6:** The 75 percentile of p-values derived from inferential replicates (red) reduces the false discovery rate of null genes compared to p-values derived from the Alevin point estimates (black). In this analysis the uncertainty-aware p-values of 10 (out of 15) null genes falls below the FDR cutoff of 0.01.

In a single-cell experiment, the burden of propagating inferential replicates can be quite high ( proportional to # inf replicates x # cells x # genes ). Fortunately, propagating just the mean and variance of the inferential replicates allows the fitting of a parametric (Negative Binomial) approximation to the posterior distribution that is both efficient to store and transfer, and that can be sampled from on demand to generate pseudo-inferential replicates. These pseudo-inferential replicates cover the true posterior gene abundances to a similar degree as the inferential replicates themselves (Figure 7) allowing for their use as an efficient proxy to the full inferential replicates [1].

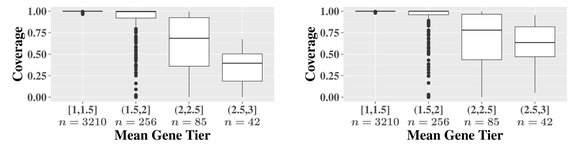


**Figure 7:** This plot demonstrate how the proposed compression of inferential replicates retains the ability to cover the true gene abundance at least as well as the original inferential replicates. The coverage ratio obtained by using all the inferential replicates are displayed in the left plot, while the right plot shows the coverage ratios obtained using draws from the fit Negative Binomial. In both cases, genes are stratified by their average (across cells) "tier" to demonstrate the effect of the level of uncertainty on coverage.

## References

1. Van Buren, Scott, et al. "Compression of quantification uncertainty for scRNA-seq counts." *BioRxiv* (2020).
2. Srivastava, Avi, et al. "Alevin efficiently estimates accurate gene abundances from dscRNA-seq data." *Genome biology* 20.1 (2019): 1-16.
3. Dobin, Alexander, et al. "STAR: ultrafast universal RNA-seq aligner." *Bioinformatics* 29.1 (2013): 15-21.
4. Melsted, Páll, et al. "Modular and efficient pre-processing of single-cell RNA-seq." BioRxiv (2019): 673285.
5. Zhu, Anqi, et al. "Nonparametric expression analysis using inferential replicate counts." Nucleic Acids Research 47.18 (2019): e105-e105.
6. Srivastava, Avi, et al. "Alignment and mapping methodology influence transcript abundance estimation." *Genome biology* 21.1 (2020): 1-29.