Comparison of scRNA-seq Pseudo-Alignment Algorithms

Matthew Dukeshire, Sawyer Lehman, Nicholas Dibley, Cvete Karamacoski

November 29, 2022

Abstract

Single-cell sequencing has become a very popular field within bioinformatics due to the vast amount of information that can be obtained compared to bulk sequencing methods, such as characterizing the heterogeneity of cells. The overall pipeline for data analysis is quite similar to bulk-sequencing methods; however, each step must be tweaked to work for single-cell data. In addition to alignment and quantification, single-cell alignment tools must perform cell calling, aligning reads to individual genes and cells, and remove PCR duplicates. There are plenty of tools developed to perform this task, two of which are Alevin and Kallisto. Comparing the alignment accuracy, time, and memory requirements of these tools revealed some major differences between them as well as a few similarities. Overall, Alevin performed better in terms of ease of use and accuracy, but Kallisto was faster and slightly more memory efficient. This comparison should allow for subsequent studies to select the tool which best matches their needs.

1 Introduction

This project aims to measure the efficiency of single-cell RNA sequencing (scRNA-seq) alignment algorithms to determine which tool is better suited for certain purposes. Compared to bulk sequencing alignment tools, there are not as many present for scRNA-seq as it is a relatively new field. For this report, the scRNA-seq alignment algorithms that we will be comparing are Kallisto (0.46.0) [1] and Alevin (integrated with Salmon 1.9.0 [2]) [3].

scRNA-seq is a transcriptome sequencing technique to observe more information about the heterogeneity of cells than bulk RNA-seq is able to provide, as proven by the first scRNA-seq study in 2009 [4]. scRNA-seq allows for the comparison of the transcriptomes of individual cells and so it allows researchers to study the transcriptional similarities and differences of a population of cells. scRNA-seq can also be used to study characteristics of gene expression such as splicing patterns and monoallelic gene expression. scRNA-seq is ideal for studying cells individually such as the differentiating cells of an early-stage embryo,

or the neurons of the brain. scRNA-seq can also be used to trace the lineage and relationships between heterogeneous cellular states in areas such as cancer, embryonic development, and lung epithelium differentiation [5].

Both Kallisto and Alevin are so-called pseudo-aligners in that they don't directly align reads to the genome, but each does so with different techniques. Kallisto uses a more traditional pseudo approach where it compares k-mers of reads with the transcriptome, thus no direct alignment to reference is computed. Additional strategies have been implemented to decrease false-positive rates in updates to the tool. Alevin uses a selective alignment that promises greater specificity but sacrifices run time. There are a scarce amount of benchmark studies, especially by third-party researchers, involving Alevin, Kallisto, and other alignment tools such as Cell Ranger 6 or STARsolo. Benchmark studies that have been performed have shown contradictory results to those performed by the researchers that developed the tools [6].

With scRNA-seq alignment, there are many more functions for the alignment tool other than simply aligning the reads, like in bulk sequencing. In addition to alignment, these tools remove PCR duplicates, perform cell calling via barcodes, and assign reads to the individual genes in cells. Kallisto and Alevin also differ in their cell calling strategies. Cell calling is done by looking for specific barcodes unique to a cell that is added during sequencing; however, often times there are sequencing errors in these barcodes. So, rather than discard a majority of the reads from the final analysis, these tools can correct the barcodes [6]. These tools also differ in their method for assigning multi-mapped reads as well as in their strategy for removing PCR duplicates.

In order to compare these differing strategies for scRNA-seq alignment, we will measure both Kallisto and Alevin on a series of metrics using real-world single-cell data. This will lead to an improved, practical view as to which alignment tool is better under certain constraints, such as accuracy, run-time, and memory efficiency. The effect on downstream analysis was not investigated due to time constraints.

2 Data

Single-cell RNA sequencing data used was generated from male house mouse (Mus Musculus) Thymus tissue during fetal growth restriction, with a control sample included. The dataset was released in 2018 through the single-cell expression atlas. The data set contains 100nt single-end reads in the form of fastq files, which were down-sampled to 5million and 400k reads for the analysis of the time/space complexity. The 400k sample was used for cell and gene counts as well as downstream processing. Sequencing was done using short-read Illumina NovaSeq 6000. The down-sampled dataset is available from the links below, which include a fastq reads file and a fastq barcode file.
control.fastq
lgf2$_K D.fastq$

The full dataset is available from single-cell expression atlas: $\text{single}_{c}ell_{a}tlas_{E} - MTAB - 6945$ [7] [8].

3 Methods

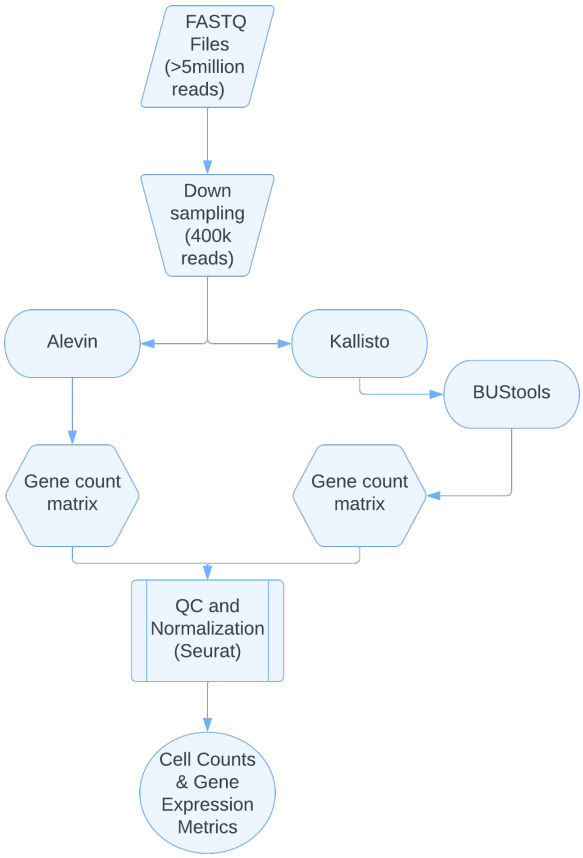


Figure 1: Flowchart of Experimental Design

The scRNA-seq alignment tools will be run on the single-cell RNA sequencing data from Mus Musculus Thymus tissue. The dataset contains over 32 million 100nt single-end reads obtained using Illumina NextSeq 2000. Down-sampled read files will be passed as inputs to each of the alignment tools and will be aligned to the most recent Mouse genome, mm39. Kallisto is used to represent pseudo-alignment algorithms that use de-Bruijn graphs as the index. Alevin will be used to represent quasi-alignment algorithms, which use both selective and pseudo-alignment strategies to align the reads. The percent of reads

aligned for each algorithm will be calculated (see Figure 1).

The run-time and memory requirements will be measured using the jobs report, which is outputted after a job completes on Carbonate. The use of a real-world practical data set, and not a simulated data set, will give insight into the true time and memory requirements.

The final metric used for comparing these algorithms will be the effects on the downstream processing. While each tool has been used for functional and exploratory research, the impact of using one tool over another is unknown. Following the alignment to the reference genome, pre-processing will result in normalized transcript counts. Kallisto also requires the use of BUStools [9] to further process the output and obtain a count matrix. These expression values will be used to compare the effect each algorithm has on the resulting downstream processing. The metrics measured were total cells, total genes, and genes per cell. The highest expressed gene will also be recorded, however further downstream analysis such as identifying markers for the knockdown samples was not performed due to time constraints.

The normalization and quality control (QC) will be performed by Seurat (4.2.1) [10] using the SCTransform function, which utilizes negative binomial regression [11]. First, the filtering cutoff point will be determined using knee-plots. Then, this cutoff point will be utilized in the principle component analysis (PCA) and uniform manifold approximation and projection (UMAP). These two dimension-reduction procedures will be done to provide different ways of visualizing the resulting normalized gene count matrices.

3.1 Alignment Algorithms

It is worth noting there are other aligners that map the reads exactly to the reference genome with a suffix array, but for the purposes of this report, we will only focus on aligners that implement the de-Bruijn graph method as the index. The advantage of pseudo-alignment tools is the use of k-mer comparisons directly to the transcriptome instead of using an uncompressed suffix array to the genome. Other pre-processing steps include barcode sequence correction and unique molecular index (UMI) sequence correction. The barcode identifies the cell type, and the correction of sequencing errors in this identifier is imperative for downstream clustering analysis. The UMI is to identify a specific molecule of RNA, and the correction of UMIs is necessary for de-duplicating any biases associated with the PCR process.

Kallisto (0.46.0) [1] is a pseudo-aligner and implements a de-Bruijn graph method as the index for alignment to the reference transcriptome. The alignment is done before the barcode correction, and there is no correction done to the UMIs. The barcode correction step is done with a "whitelist" of barcodes provided by the sequencing platform. If there is a sequencing issue in the barcode, those with the error are placed one Hamming distance away to the nearest whitelisted barcode. Any multi-mapped reads are discarded from the final gene count matrix that is generated with an external tool.

Alevin [3] is an improved version of a pseudo-aligner, and can be described as

a quasi-aligner in that it utilizes the selective alignment featured in the internal mapper, Salmon (1.9.0) [2]. The barcode correction is done first, and a putative barcode list based on frequency from the data itself is used in lieu of the whitelist of barcodes provided by the sequencing platform. The barcodes with errors are then replaced with an edit distance calculation according to the generated list. Similar to Kallisto, a de-Bruijn graph method is done for the alignment of the reads to the reference transcriptome. After alignment, UMI correction is done. For this step, a graph is generated of the UMIs and de-duplicated by replacing smaller nodes of UMIs with the most similar and larger nodes. Alevin handles the multi-mapping of reads through an expectation-maximization (EM) algorithm and distributes the read counts accordingly. The output of this tool is the final gene count matrix.

4 Results and Discussion

Each of the 5 million read and 400k read data sets was run using Alevin and Kallisto. Kallisto outputs require the use of BUStools in order to generate the gene count matrix. Following completion of each run, the time taken and memory used reported by SLURM on carbonate were recorded. The alignment, number of barcodes, number of genes, and genes per cell were all reported in json files by each of the tools. For Alevin cell counts, the final filtered set of barcodes was used. Finally, the most expressed gene was obtained from Seurat. In terms of ease of use, the input format for Alevin allowed for much more user customization and required just one command. Kallisto was easy to run; however, customization requires the use of regex expressions and the output must undergo further processing by BUStools.

As seen in Table 1, the alignment percent, or mapping rate, was slightly higher for Alevin. This is likely due to Alevin being able to handle multi-mapped reads as well as utilizing selective alignment rather than Kallisto which is a true pseudo-aligner. The number of genes detected by each tool is relatively similar, with Kallisto identifying slightly more genes. This was relatively surprising because it had a lower mapping rate. It could be due to Kallisto lacking a UMI correction/de-duplication step, allowing for PCR duplicates to be erroneously mapped.

Finally, the biggest observed difference between the two tools was the number of cell barcodes identified, thus affecting the number of genes per cell. While the number of cells in the full experiment is known, rarely are they all used for analysis. Both Alevin and Kallisto have a form of barcode identification and filtering, thus the cell counts couldn't be compared to a true value. However, the cell counts can be compared to one another. Alevin identified more than 3 times more cells than Kallisto. This was very surprising, despite the major differences between the two barcode identification algorithms. The stark difference may be due to Alevin identifying too many cells, but it is more likely that Kallisto was unable to find many of the barcodes within the internal whitelist. Further investigation into the differences in the number of cells identified by scRNA-

seq alignment tools is needed to better understand this inconsistency. Due to this disparity in cell count, the downstream QC and analysis will be affected. Interestingly, despite the data being down-sampled, it was observed that both tools reported the average highest expressed gene to be Hbb-bs, which is also reported as the highest expressed gene in the original study. Further effects on downstream analysis beyond QC and normalization were not investigated.

The run-time for two sized data sets on each tool is shown in Figure 2a. While both tools were able to run the smaller 400k read file within similar time frames (less than 1.5 min), Alevin took nearly 3 hours to deal with the larger 5 million read file, whereas Kallisto took less than 30 minutes. This is likely due to the different alignment strategies implemented, where Kallisto is a true pseudo-aligner which is much faster. Additionally, the barcode identification technique of Alevin is much more involved while Kallisto simply checks them against a list.

The memory requirements of each tool, shown in Figure 2b, are much more similar than the time requirements. Alevin uses slightly more memory for the same input files, but it doesn't scale with input size. Overall, memory complexity is the same for both tools however Kallisto is much more time efficient and thus could be a good first choice, especially for exploratory analysis.

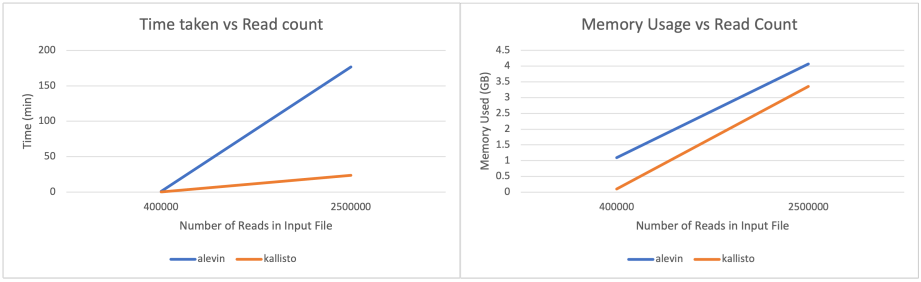
In Figure 3, the knee-plots are shown for both Alevin 3a and Kallisto 3b. Knee-plots simply plot the total RNA molecules, or UMIs, against the barcode rank. The barcode rank is derived from the frequency at which each barcode appears and can be interpreted as a confidence to whether the barcode is associated with a real cell. The two plots look relatively similar, other than where Seurat placed the inflection point. It is worth noting that because the reads are down-sampled, the knee-plot does not give an ideal inflection point which would likely be present in a high-quality full data set. The Kallisto knee-plot does show a few more high-ranked barcodes with more UMIs, however, it drops quicker. The inflection points were used to filter the cells for PCA and UMAP. Surprisingly, Alevin showed a much lower barcode rank cutoff. This may indicate that while Kallisto found much fewer cells it may have also found higher-quality cells, but further investigation is needed to prove this.

Using Seurat, PCA was performed on the outputs from Alevin and Kallisto, shown in Figure 4. Due to the difference in cell count and the different inflection points, it was expected that the PCA knee-plots look at least slightly different. Due to all cells being of the same cell type (Thymus) and within the control sample, we do not expect to see a big drop off for a component due to low variance between the cells. This hypothesis holds true for both the Alevin and Kallisto PCA knee-plots. Overall, the difference in the number of cells didn't seem to influence PCA greatly due to all cells being of the same type.

Finally, UMAP dimension reduction was performed and each cell was plotted based on the top two components, seen in Figure 5. Because only one cell type is present, it was expected that there would be no discernible clusters present. This holds true for both Alevin and Kallisto, providing further evidence that the filtering performed by Seurat following the initial cutoff provided by the knee-plots made the two outputs much more similar.

Table 1: Comparison of Basic Metrics for Alevin and Kallisto

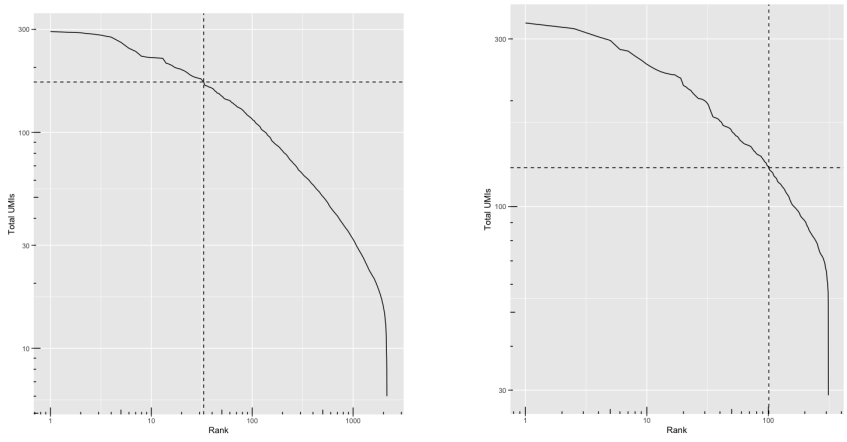
Metric	Alevin	Kallisto-Bustools
Time (400k reads/2.5m reads)	1.17 min / 176.58 min	0.3 min / 23.93 min
Memory Used (400k reads/2.5m reads)	1.1GB / 4.07GB	0.1GB / 3.36GB
Alignment % (unique %)	24.68	21.9
Num Barcodes (Cells)	1009	313
Num Genes total	34433	35387
Genes per cell (Avg)	93	250
Most Expressed Gene	Hbb-bs	Hbb-bs



(a) Time taken for two data sets.

(b) Memory used for two data sets.

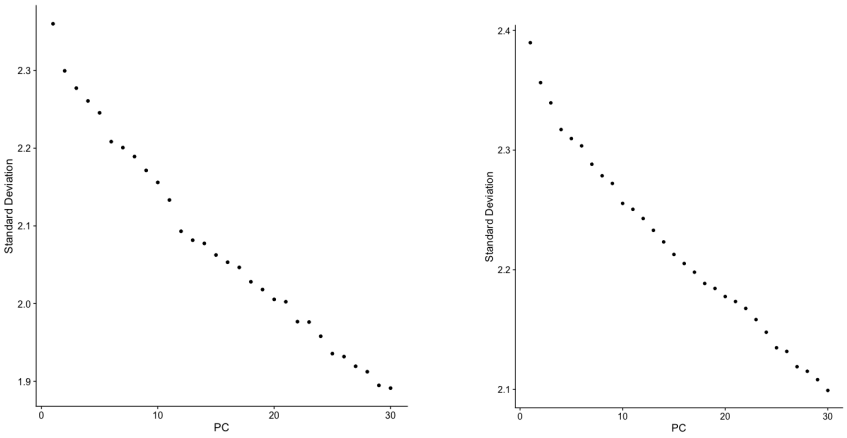
Figure 2: The two sized down-sampled data sets were: a 5 million read data set, and a 400,000 read data set. Metrics were gathered from the Carbonate job report. Alevin is shown in blue, and Kallisto in orange.



(a) Alevin KneePlot

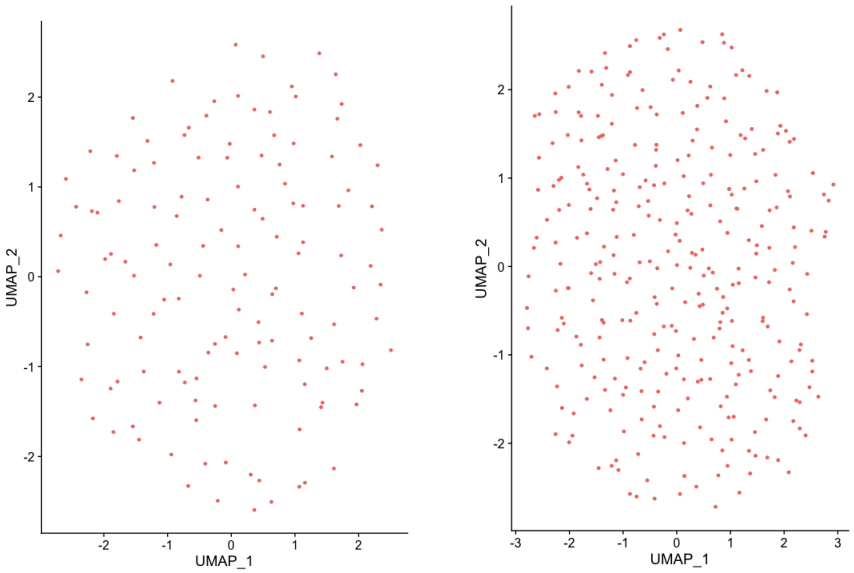
(b) Kallisto KneePlot

Figure 3: KneePlots generated by Seurat. The intersection of the two lines indicates the filtering cutoff point for further analysis.



(a) Alevin PCA (b) Kallisto PCA

Figure 4: PCA plots generated by Seurat.



(a) Alevin UMAP (b) Kallisto UMAP

Figure 5: UMAP plots generated by Seurat.

5 Limitations and Future Direction

Despite Alevin and Kallisto showing some differences, the full extent of these cannot be understood from this analysis. Firstly, real-world data was used and thus true alignment accuracy cannot be determined. However, future studies could include a simulated data set to get true accuracy for alignment. Additionally, further downstream analysis is needed to fully investigate how each tool will affect the analysis of real-world data. Future studies should include cell-type identification by including scRNA-seq data for multiple cells. Also, a disease vs control analysis could be performed as a case study to compare the tools. Finally, further studies should include other single-cell RNA tools such as Cell Ranger and STARsolo to encapsulate more alignment algorithm types.

6 Conclusion

This study aimed to compare the performance of Alevin and Kallisto for the alignment and cell-calling of single-cell RNA sequencing data using various metrics. Each tool uses differing algorithms, with Alevin being a quasi-aligner and Kallisto being a true pseudo-aligner. Each tool also has differing methods for identifying cell barcodes and UMI de-duplication. In the end, it was observed that Kallisto is more time and memory efficient. Additionally, Kallisto will drop out many more cells than Alevin due to Kallisto comparing barcodes directly against a whitelist, while Alevin uses the frequency of each barcode to identify probable cell barcodes. Thankfully, QC and filtering with tools like Seurat tend to mediate many of these differences, proving the importance of QC, especially in meta-analysis where you combine outputs from multiple tools. Future studies should provide further context to this comparison by including more tools and performing additional downstream analysis, such as cell-type identification via gene markers. Overall, each tool has a distinct set of use cases. For instance, because of its time and space efficiency, Kallisto is likely to be used for preliminary data exploration, whereas Alevin would be utilized for more formal research.

References

[1] N. L. Bray, H. Pimentel, P. Melsted, and L. Pachter, "Near-optimal probabilistic rna-seq quantification," *Nature biotechnology*, vol. 34, no. 5, pp. 525–527, 2016.

[2] R. Patro, G. Duggal, M. I. Love, R. A. Irizarry, and C. Kingsford, "Salmon provides fast and bias-aware quantification of transcript expression," *Nature methods*, vol. 14, no. 4, pp. 417–419, 2017.

[3] A. Srivastava, L. Malik, T. Smith, I. Sudbery, and R. Patro, "Alevin efficiently estimates accurate gene abundances from dscrna-seq data," *Genome biology*, vol. 20, no. 1, pp. 1–16, 2019.

[4] F. Tang, C. Barbacioru, Y. Wang, E. Nordman, C. Lee, N. Xu, X. Wang, J. Bodeau, B. B. Tuch, A. Siddiqui, *et al.*, "mrna-seq whole-transcriptome analysis of a single cell," *Nature methods*, vol. 6, no. 5, pp. 377–382, 2009.

[5] A. Haque, J. Engel, S. Teichmann, and T. Lonnberg, "A practical guide to single-cell rna-sequencing for biomedical research and clinical applications," *Genome Medicine*, vol. 9, no. 75, pp. 1–12, 2017.

[6] R. S. Brüning, L. Tombor, M. H. Schulz, S. Dimmeler, and D. John, "Comparative analysis of common alignment tools for single-cell RNA sequencing," *GigaScience*, vol. 11, 01 2022. giac001.

[7] W. A. Bacon, R. S. Hamilton, Z. Yu, J. Kieckbusch, D. Hawkes, A. M. Krzak, C. Abell, F. Colucci, and D. S. Charnock-Jones, "Single-cell analysis identifies thymic maturation delay in growth-restricted neonatal mice," *Frontiers in immunology*, vol. 9, p. 2523, 2018.

[8] P. Moreno, S. Fexova, N. George, J. R. Manning, Z. Miao, S. Mohammed, A. Muñoz-Pomer, A. Fullgrabe, Y. Bi, N. Bush, *et al.*, "Expression atlas update: gene and protein expression in multiple species," *Nucleic Acids Research*, vol. 50, no. D1, pp. D129–D140, 2022.

[9] P. Melsted, V. Ntranos, and L. Pachter, "The barcode, umi, set format and bustools," *Bioinformatics*, vol. 35, no. 21, pp. 4472–4473, 2019.

[10] Y. Hao, S. Hao, E. Andersen-Nissen, W. M. Mauck III, S. Zheng, A. Butler, M. J. Lee, A. J. Wilk, C. Darby, M. Zager, *et al.*, "Integrated analysis of multimodal single-cell data," *Cell*, vol. 184, no. 13, pp. 3573–3587, 2021.

[11] C. Hafemeister and R. Satija, "Normalization and variance stabilization of single-cell rna-seq data using regularized negative binomial regression," *Genome biology*, vol. 20, no. 1, pp. 1–15, 2019.

7 Appendix

```
salmon alevin --dumpFeatures --dumpMtx -l ISR -1
    real_alevin_attempt/mouse_reads1.fastq -2 real_alevin_attempt/
    mouse_reads2.fastq --dropseq -i salmon_index_files -o
    real_alevin_attempt --tgMap t2g_alevin_fix.txt
```

Listing 1: Alevin command

```
kallisto bus -i mus_musculus/transcriptome.idx -o
    kallisto_output_attempt5 -x DropSeq real_alevin_attempt/
    mouse_reads1.fastq real_alevin_attempt/mouse_reads2.fastq
```

Listing 2: Kallisto command

```
library(dplyr)
library(tidyr)
library(stringr)
library(rstudioapi)
library(data.table)
library(BUSpaRse)
library(Seurat)
library(tidyverse)
library(DropletUtils)
library(biomartr)
library(Matrix)
library(rtracklayer)
library(tximport)


setwd(dirname(getActiveDocumentContext()$path))

yx = readMM('./real_alevin_attempt/alevin/quants_mat.mtx')

dim(yx)

rr = (read.delim('./real_alevin_attempt/alevin/quants_mat_rows.txt'
    , header = F))

cc = read.delim('./real_alevin_attempt/alevin/quants_mat_cols.txt',
     header = F)

rownames(yx) = rr$V1

colnames(yx) = cc$V1

yx = t(yx)



x = readMM('./kallisto_output_attempt5/matrix.mtx')

r = read.delim('./kallisto_output_attempt5/genes2.tsv', header = F)
    [,1]

rownames(x) = r
```

```
39
40 c = read.delim('./kallisto_output_attempt5/barcodes.tsv', header =
       F)[,1]
41
42 colnames(x) = c
43
44
45
46 bc_rank = barcodeRanks(x, exclude.from = 5)
47
48
49
50 knee_plot <- function(bc_rank) {
51
52   knee_plt <- tibble(rank = bc_rank[["rank"]],
53
54                      total = bc_rank[["total"]]) %>%
55
56     distinct() %>%
57
58     dplyr::filter(total > 0)
59
60   annot <- tibble(inflection = metadata(bc_rank)[["inflection"]],
61
62                   rank_cutoff = max(bc_rank$rank[bc_rank$total >
       metadata(bc_rank)[["inflection"]]]))
63
64   p <- ggplot(knee_plt, aes(rank, total)) +
65
66     geom_line() +
67
68     geom_hline(aes(yintercept = inflection), data = annot, linetype
        = 2) +
69
70     geom_vline(aes(xintercept = rank_cutoff), data = annot,
       linetype = 2) +
71
72     scale_x_log10() +
73
74     scale_y_log10() +
75
76     annotation_logticks() +
77
78     labs(x = "Rank", y = "Total UMIs")
79
80   return(p)
81
82 }
83
84 options(repr.plot.width=9, repr.plot.height=6)
85
86 knee_plot(bc_rank)
87
88
89
90 res_mat <- yx[, tot_counts > 30]
91
```

```r
res_mat <- res_mat[Matrix::rowSums(res_mat) > 0,]

dim(res_mat)

summary(Matrix::colSums(res_mat))



seu <- CreateSeuratObject(res_mat, min.cells = 3) %>%

  SCTransform(verbose = FALSE)


VlnPlot(seu, c("nCount_RNA", "nFeature_RNA"), pt.size = 0)



ggplot(seu@meta.data, aes(nCount_RNA, nFeature_RNA, color = 'red')) +

  geom_point() +

  labs(x = "Total UMI counts per cell", y = "Number of genes detected")



seu <- RunPCA(seu, verbose = FALSE, npcs = 30)

ElbowPlot(seu, ndims = 30)

seu <- RunUMAP(seu, dims = 1:20)

UMAPPlot(seu)
```

Listing 3: Seurat R code