Cory Williams

Programmer

Project 4

<u>Method/Results:</u>

Precomputed Alevin data was obtained from the BF528 project folder. To import and convert the Alevin output to be pipelined into R, the package tximport version 1.12.7 was utilized. The Alevin output file used contained gene expression levels, gene identification code, and cell sample barcodes for around 30,000 individual cells. For filtering and further downstream analysis, the package Seurat version 2.7 was used. Tximport converts Alevin output files into a matrix that the Seurat can turn into an object to apply multiple analysis tools easily.

The raw alevin file contained varying gene counts per cell with the highest gene count for a single cell being in the upper 7000's.[figure first plot} A gene count that high is problematic and usually points into the compilations of doublets. Per the project, guidelines provided cross-referenced by information in the supplementary article, the range of genes per cell should be 4000 or lower. With those complications and parameters in mind, furthering filtering had to be conducted before normalization.

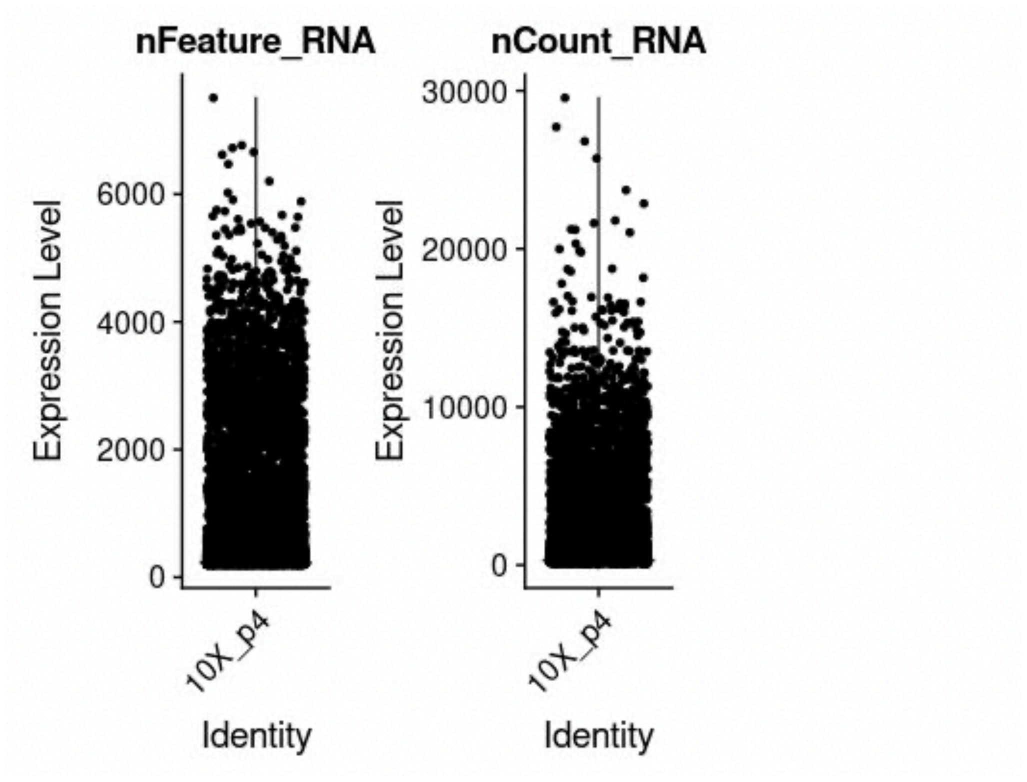


Figure 9. Plot of Alevin output cell and gene count before filtering. On the right (nCount_RNA) represents the total cell count hitting 30,000 displayed in an expression level format. Upon first glance nCount_RNA implies that cells have different "expression levels" however just for nCount_RNA plot this actually represents the total number of cells recorded on the y axis with the highest point at 30,000 representing that is the max. On the left plot (nFeature_RNA is gene expression) represents a true expression level plot with the y axis scale representing gene expression per cell with only a few cells having over 6000 genes recorded and a majority of the cells having 4000 genes or less recorded.

Filtering:

The first filtering that was conducted before normalization was removing cells that have a gene count (feature count) of more than 2700. This filtering step was done with a command from the Seurat package with parameters set for keeping all cells with gene counts of above 200 and below 2700.

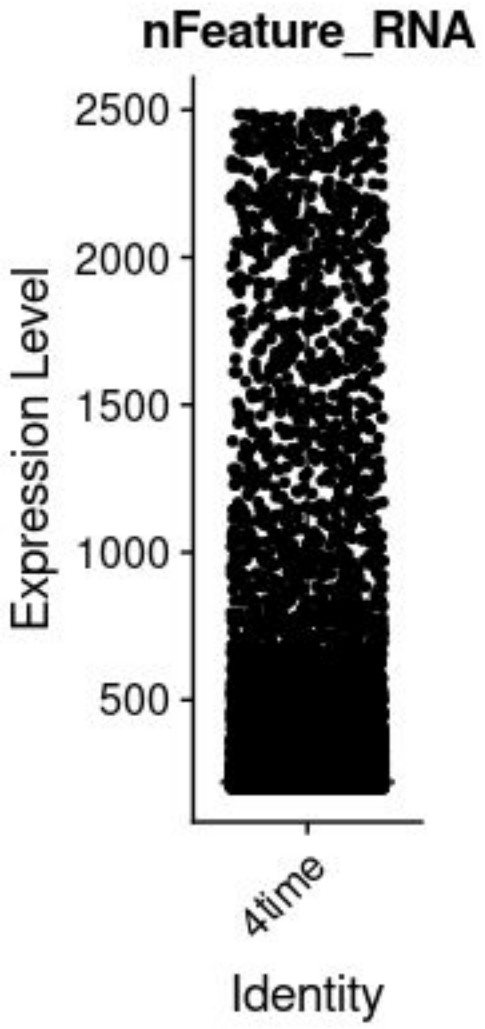


Figure 10. Post filtering gene expression plot. This plot represents cells retained after feature count filtering of less than 2700 but more than 200. This is pre normalization so the cells that appear to have expressions levels of less than 200 should be adjusted for automatically post normalization.

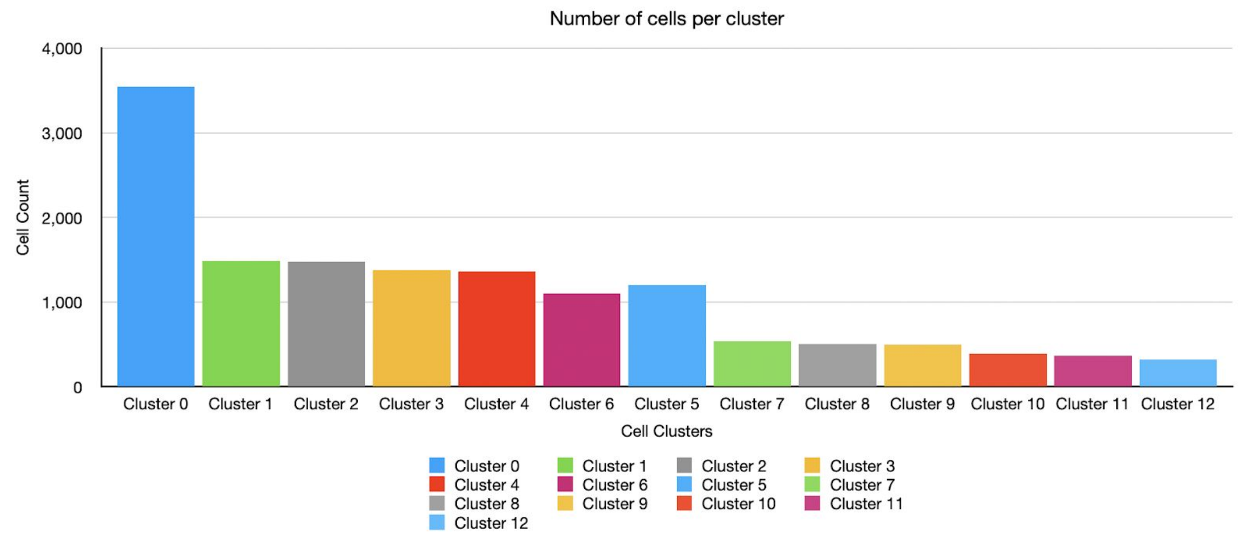
Normalization:

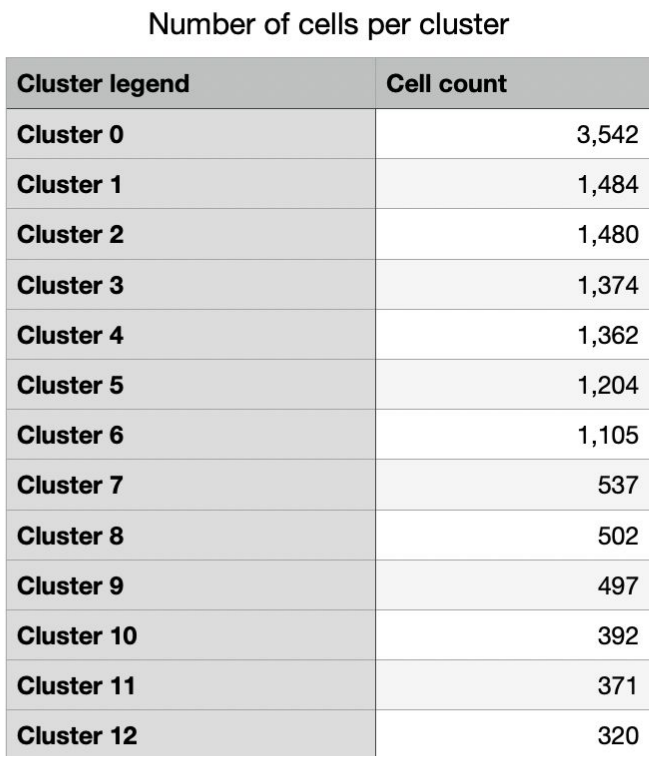
After this filtering step, normalization was done using the normalize function of the Seurat package with the specific normalization method being “LogNormalize with a scale factor of 10,000.

Post Normalization Filtering:

Post normalization filtering was done by identifying genes with high cell variability (variability filtering). This filtering step looks across all normalized cells and their genes and removes the peripheral cells that have gene abnormally low and or high expression of specific genes. This ensures that the cells kept for further downstream analysis do not have a large group of cells that are uncommonly expressing too high or too low levels of a particular gene. Variability filtering was followed by linear transformation. Linear transformation was used to shift the expression of each gene so that the mean expression across all cells is 0 in addition to scaling the expression of each gene so that the variance across cells is 1. This step gives equal weight in downstream analyses so that highly expressed genes do not dominate.

Initial filtering, normalization, and post normalization filtering dramatically dropped the cell count from around 30,000 to around 14,000 cells of higher quality. Clustering was done using the Seurat clustering function and produced a total of 13 clusters. Clusters had an average of 1000 cells per cluster. [figure#chart]




Number of cells per cluster

Cluster legend	Cell count
Cluster 0	3,542
Cluster 1	1,484
Cluster 2	1,480
Cluster 3	1,374
Cluster 4	1,362
Cluster 5	1,204
Cluster 6	1,105
Cluster 7	537
Cluster 8	502
Cluster 9	497
Cluster 10	392
Cluster 11	371
Cluster 12	320

Figure 11. Post filtering and normalization clustering. (A) Color assigned bar chart of the 13 produced clustering after filter and normalization. Cluster 0 appears to have the most cells while cluster 12 having an alarming low count. **(B)** Table containing the raw cell count number records per cluster, can also be used as a legend for above chart.

__Discussion:__

There were two main filtering methods of choice used for pre analyses of the alevin output data. The two main filtering methods accounted for removing cell doublets and lowering gene expression. The threshold selected for doublet removal was chosen with the knowledge of cells expressing lower than 200 genes are usually projected to be insignificant to analysis along with cells expressing over 2700 to be abnormally high given the cell type looked at in the article.   The lowering gene expression variation filtering was broken into two sub filtering methods were the first removed outlier cells with abnormally low or abnormally high gene expression relative to a majority of the cells. The second sub filtering, linear transformation, shifted the expression of each gene so that the mean expression across cells is 0 and to scale the expression of each gene so that the variance across cells is 1. The LogNormalization is a standard normalization method that is applied to many other single-cell output applications. When applied to this dataset, it ran relatively quickly and as expected. This compound filtering combined with normalization resulted in producing 13 cluster groups that not only made sense but seemed to be reasonably similar to the number of clustered groups that were present in the precomputed data for the analyst role.

__Conclusion:__

Package versions were the most challenging aspect of this experiment, with the provided version being severely out of date, leading to multiple installations and updating of the required packages. Computing power also played a significant factor when importing the Alevin output file into R. This took over an hour when ran on the

Boston university clustering computing network with four cores selected; however, when the cores were increased to 16 that time was cut in half. Future directions would lead to making sure a sufficient amount of cores is selected from the start to dramatically reduce load times.