Single Cell RNA-Seq Analysis Pipeline Overview

Guilherme de Sena Brandine

May 11, 2017

1 Upstream Analysis

The upstream analysis is done differently whether the data comes from the traditional drop-seq protocol (**?**) from 10X genomics (**?**). In the traditional protocol we use FastQC (**?**) on the second end read to assess the Illumina read quality and overall sequence biases. If necessary we use Trim Galore (**?**) to trim out possible adapter contamination. The second-end reads are then mapped to the NCBI *Coturnix japonica* reference genome using the STAR(**?**) mapper to obtain a BAM file. The package developed by the McCarroll lab (**?**) is used to tag, for every mapped read, the cell/molecule barcode from the first end read as metadata. The first 12 base pairs of read 1 are tagged as cell barcodes, and the last 8 base pairs are tagged as UMIs. Reads are subsequently merged if they have matching cell barcodes and if the UMIs have a Hamming distance of at most 1. Finally, BAM alignments are matched with the annotation from the appropriate reference genome to create a digital gene expression matrix that counts the number of reads mapped to each gene in each barcode.

The CellRanger software from 10X proceeds in a similar fashin as described above, but their barcode standards are different. They have adapters between cell and molecular barcodes that overcome potential ambiguities in barcode sequencing errors and have a list of valid barcodes as reference. Each sequenced cell barcode is associated with a whitelist barcode based on sequence similarity.

2 Downstream Analysis

Downstream analysis is done using the implementation available in the Seurat(**?**) package unless the subsection is marked with an asterisk (*) . Below is a brief explanation of the mathematical assumptions and models at each step of the analysis. The figures shown are from the 10X dataset of Human Fetal Kidney in 16 Weeks Developmental Time.

2.1 Count Table Quality Control

2.1.1 Good-Turing Estimate*

When reads from a particular cell are mapped to a reference genome, a natural question that arises is: What proportion of the total RNA in the library is sequenced?

Each successfully mapped and annotated read can be thought of as sampling a random fragment of RNA from a population with an unknown number of different genes and an unknown number of fragments per gene. Good & Turing (**?**) have shown that, as long as the each fragment has nonzero probability of being sequenced, the asymptotic expected value of the number of fragments S_i not sequenced in cell i is given by:

$$\mathbb{E}[S_i] = n_{i1}/N_i$$

Where n_{i1} is the number of genes in cell i with only one read mapped to it and N_i is the total number of reads in cell i. The most interesting thing is that this result doesn't require any assumption on the distribution of our sampling! Only that every read has a nonzero probability of being sequenced. We'll call the formula above the **estimated coverage**. This approximation is better the larger the value of n_1 is.

When the number of cells in the dataset is large, we do a density plot of the estimated coverage of all cells. Those whose value approaches 1 indicate a very low coverage. Pollen (**?**) has shown that, even when the coverage of all cells is very low, we are still capable of identifying subpopulations accurately, so for the purposes of clustering, this criteria is not used to exclude cells.

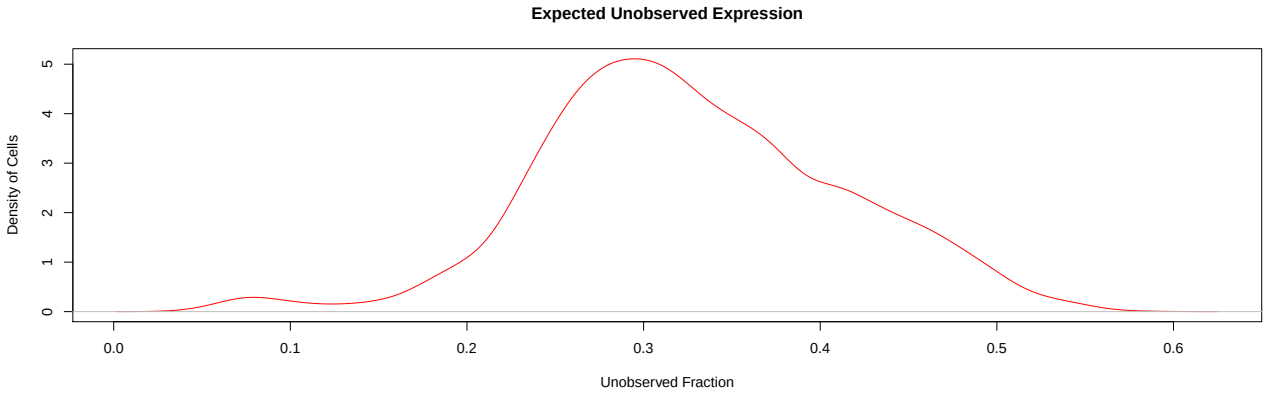

Figure 1: *An example density plot of expected unsequenced reads. Most cells lie on the 0.3 range, meaning that, in average, 70% of the transcriptome of all cells is expected to be sequneced*

2.1.2 Reads from Mitochondrial Genes

Another common QC metrics is the proportion of reads mapped to mitochondrial genes. In higher eukaryotes, if the space of sequenced genes is comprehensive, mitochondrial reads should appear in very low proportion. As a general rule, we exclude cells that have more than 5% of the reads mapped to mitochondrial genes - that is, those with the *MT* prefix in the ensembl annotation -.

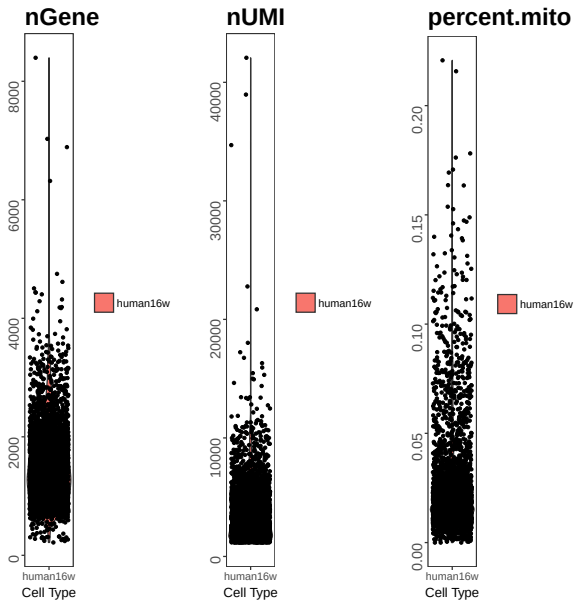

Figure 2: *An example visualization of the content of each cell. Left violin plot is the number of nonzero genes in the transcriptome. Middle plot is the total number of reads. Right plot is the percentage of reads that are mapped to mitochondrial genes. Most cells lie below the 5% threshold of mitochondrial genes but there are also many with a significant expression of MT, which will subsequently be excluded.*

2.1.3 10X Sequencing Saturation

The 10X CellRanger pipeline is also capable of yielding information on the saturation curve of the cells. Figure 3 is an example of a saturation curve, where we plot the number of reads vs the estimated sequence saturation.

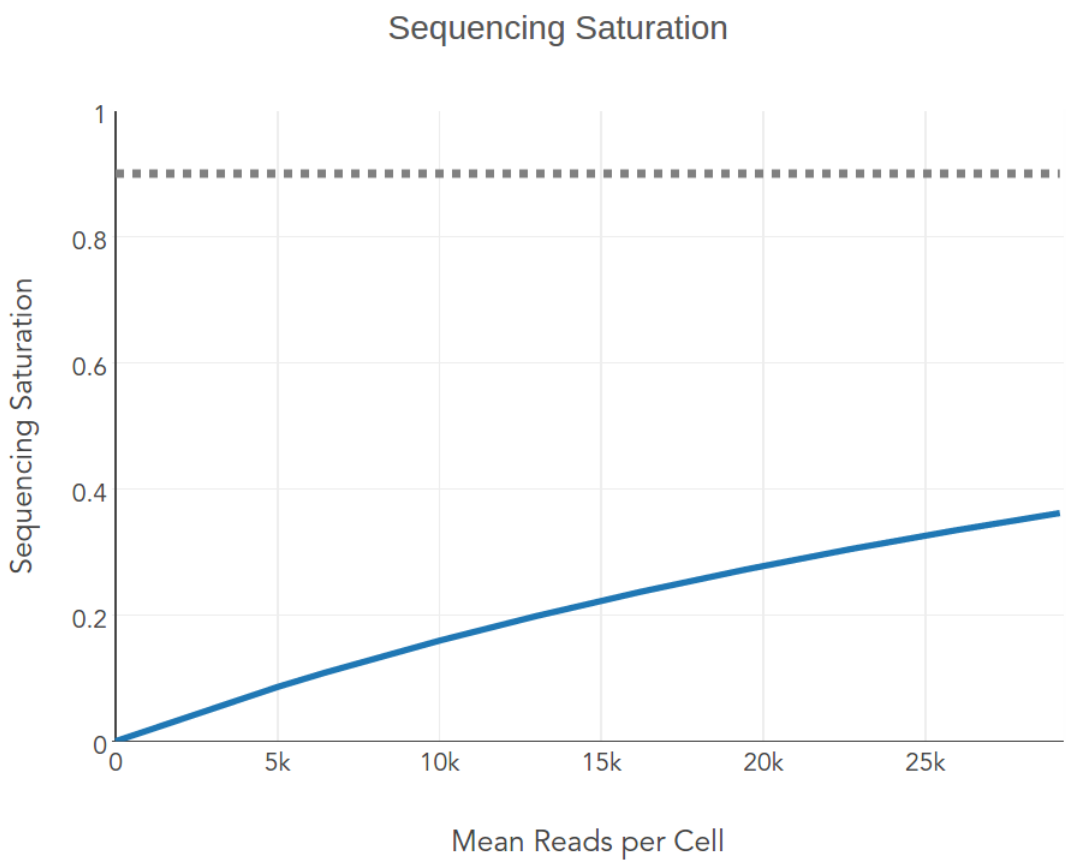

Figure 3: *A plot of the estimated library sequence saturation vs mean number of reads per cell*

2.2 Normalization

Several studies have shown a monotonic relationship between the variance of a particular gene across technical replicates of the same biological RNA-Seq experiment and its average value, both in bulk () and single cell RNA-Seq(). This is problematic since we expect the observed variance across cells to be exclusively due to phenotype heterogeneity. For this reason, we apply a variance stabilizing transformation that turns the mean and variance independent. Specifically, for a random variable X such that $Var[X] \sim E[X]^2$, the random variable $Y = \log(1 + X)$ has approximately constant variance regardless of its mean. We thus do a log transform on the dataset before proceding with further analysis.

2.3 Variable Gene Selection

After normalization, we analyze the subsequent mean-variance relationship across all genes. We calculate the mean and variance of all genes and group them into bins of similar variance. We then calculate the z-score of each gene (ie, how many standard deviations away from its bin the gene is) and keep genes with high absolute z-score.

2.4 PCA and Choice of Principal Components

2.4.1 Overview of the PCA Algorithm

With a (hopefully) smaller list of genes with high variance, we usually will have genes that have similar expression patterns across cells, most likely due to them being co-expressed or biologically dependent. Principal Component Analysis is a dimension reduction method that creates new features (called principal components) as linear combinations of genes such that the variances of these components are maximized. More formally, suppose we have N variable genes and their expression values on cell i are $g_{i1}, g_{i2}, \ldots, g_{iN}$. We create K principal components (and hopefully K is much smaller than N) such that the j-th principal component value for cell i is given by:

$$PC_{ij} = \alpha_{j1} g_{i1} + \ldots + \alpha_{jn} g_{iN}$$

The PCA algorithm chooses the values of the α's under the constraint that, for all j, $\alpha_{j1}^2 + \ldots + \alpha_{jN}^2 = 1$ (ie, the rotation has unit length) and if $j \neq l$ then $\alpha_{j1}\alpha_{l1} + \ldots \alpha_{jN}\alpha_{lN} = 0$ (ie, the principal components are orthogonal to each other).

An informative subproduct of PCA is the set of genes that contribute the most to each PC, that is, for a specific PC j, the genes that have the largest absolute values of α_j in the above formula. To visualize such genes, we build heatmaps for each PC j. We choose the cells with highest values of PC_j and the genes with highest α_j and plot them on a heatmap. we usually see a bimodal distribution of genes with both positive and negative correlation (Figure 4), and we may further interrogate if these genes have similar biological functions by testing for enrichment (see *Ontology Enrichment Analysis*)

2.4.2 Choosing the number of PCs to retain

Dimensionality reduction usually comes with the cost of information loss. If we wanted to retain as much variance between cells as in our original dataset, we would have to have as many PCs as we have genes, which wouldn't be very useful. Many PCs, however, have a very small variance and contribute to, at most, noise. Storey (**?**) has proposed a statistical method called JackStraw to measure the significance of a principal component by analyzing how the coefficients α_{jg} of each gene g in PC j changes when we shuffle the expression value of all cells. We use this statistic to retain pcs such that most genes still have a large α when permutated.

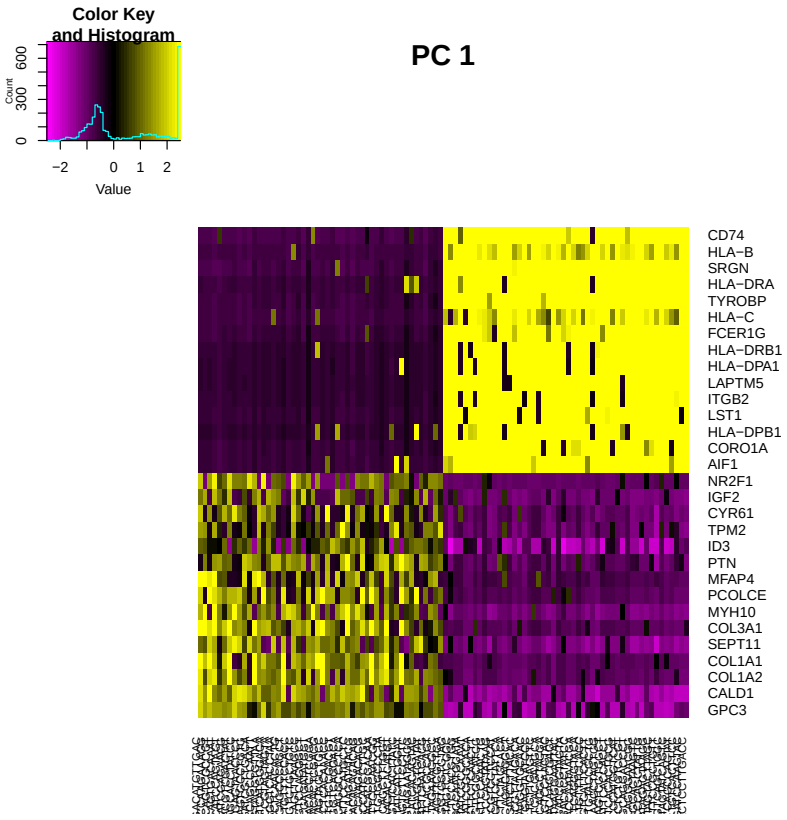

Figure 4: *An example of PC Heatmap: rows are genes with highest absolute value of α coefficients in Principal Component 1, and columns are cells with highest value of PC1. Rows and columns are clustered by similarity.*

2.5 Clustering and Subpopulation Discovery

2.5.1 The K Nearest Neighbors Graph

The goal of the aforementioned steps is to summarize the large volume of data into a much smaller number of principal components. We use the principal components to further group the cells into subpopulatins with similar expression. We usually don't know a priori how many subpopulations exist, whence we'd like to use an unsupervised approach which would also give us that information. Shekhar(**?**) has analyzed the robustness and correctness of several different clustering methods on datasets with a large number of cells with shallow coverage and has shown that the Louvain-Jaccard method yields best results.

The idea of this method is to join every cell to its k nearest neighbors as measured by Euclidean distance. If we retain m PCs, the Euclidean distance between cell i and cell j is given by:

$$d_{ij} = \sqrt{(PC_{i1} - PC_{j1})^2 + \ldots + (PC_{im} - PC_{jm})^2}$$

We build a network of cells in which every cell is connected by the k cells with smallest distance. A subsequent step is to assign an edge weight between any two connected cells given by their jaccard index,

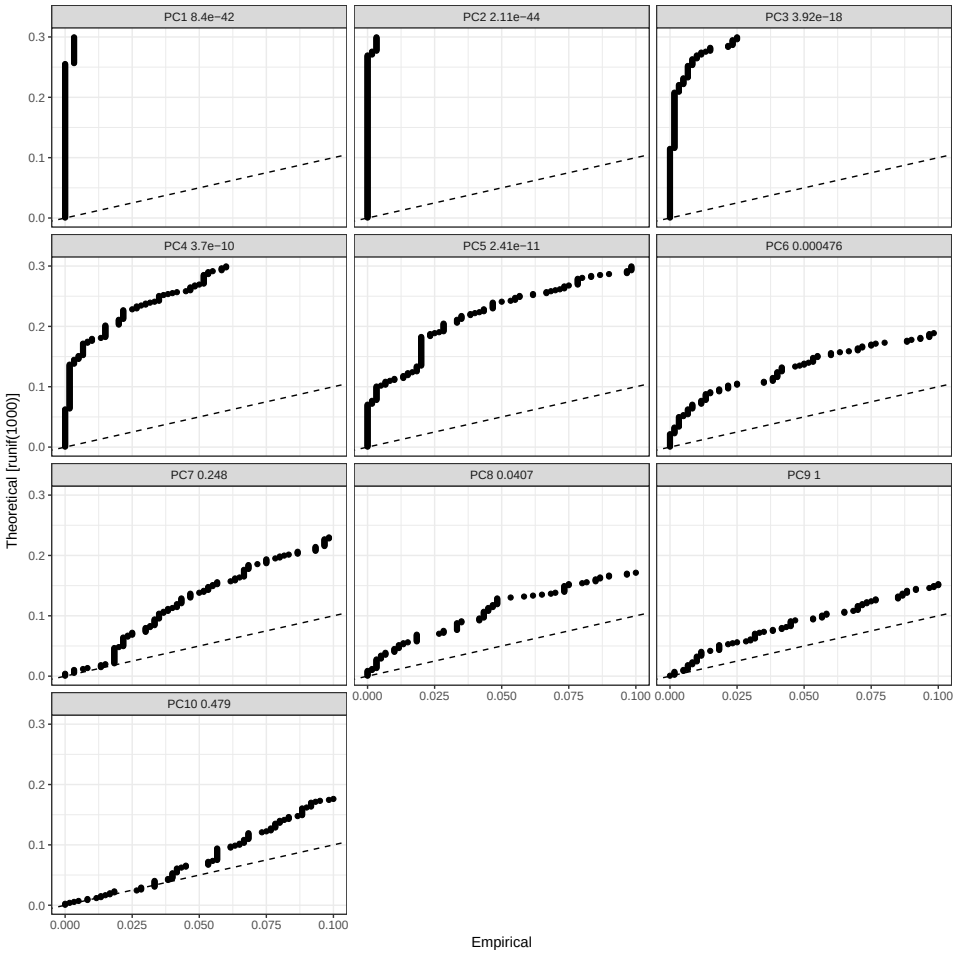

Figure 5: *An example of a JackStraw plot. In each PC plot, each dot is a gene and its x and y coordinates are the p-value of the permutation test and the theoretical p-value if the gene expression was randomly uniformly selected, respectively. PCs that have many genes above the diagonal line have the most statistically significant genes and should be retained. We usually set a cutoff of $p < 10^{-4}$ as a standard rule to choose which PCs to keep.*

given by:

$$A_{ij} = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} = \frac{|S_i \cap S_j|}{2k - |S_i \cap S_j|}$$

Which is to say the edge weight is the ratio between the number of common nearest neighbors between the two cells and the total number of edges of the two (which is at most $2k$). Figure 6 shows a practical example of how to calculate the edge weights. Naturally, if any two cells have no common nearest neighbors, their edge weight will be 0 and the edge will not be added to the graph.

2.5.2 Clustering by Maximizing the Louvain Modularity

The goal on building such graph is to group the nodes into smaller subgroups (henceforth called *clusters*) such that there are many edges inside the clusters and few edges between cells of different clusters. Of

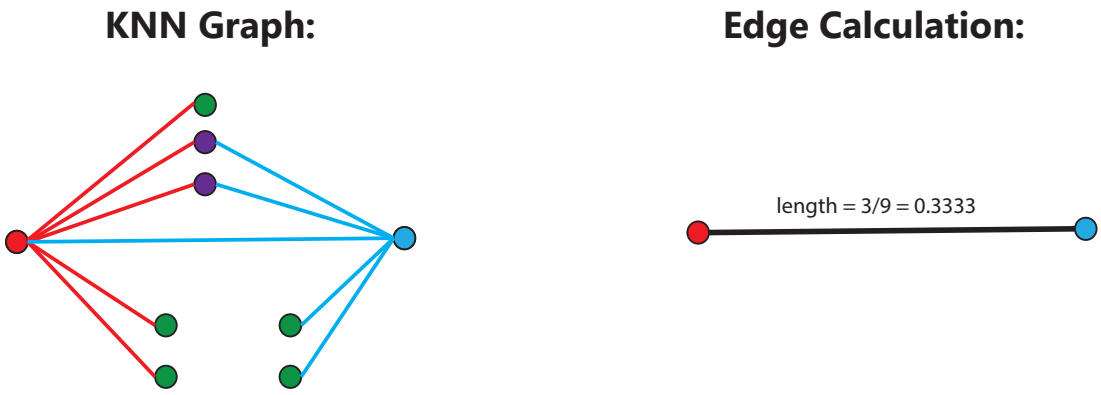

Figure 6: *A schematic illustration on how edges values are calculated based on Jaccard Index. Here* $k = 5$ *and the red and blue cells have their 5 nearest neighbors indicated by the red and blue lines, respectively. The two cells have 3 common neighbors: The two purple cells + the red cell (as, by default, the cell itself is also part of its K nearest neighbors set). There are 9 cells in total in the union of the two KNN sets, so the edge weight between red and blue will be* $3/9 = 0.3333...$

course, if we stated the problem like this, a trivial solution would be to have just one group comprising all the cells, which is undesired. Louvain has proposed a different measure that better fits the problem: Can we divide the cells into clusters such that the sum of the edges inside a cluster is bigger than the sum of the average edge weight of all cells in the cluster?

From this definition we turn clustering into an optimization problem. The formula to maximize is given by:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{w_i w_j}{2m} \right] \delta(c_i, c_j)$$

Where m is the total number of edges, c_i (resp, c_j) is the cluster number to which cell i (resp j) belongs, A_{ij} is the value of the edge between cells i and j as calculated above, w_i (resp, w_j) is the sum of all edges connected to cell i(resp, j) and $\delta(c_i, c_j)$ is a function equal to 1 if $c_i = c_j$ (ie, cells i and j belong to the same cluster) or 0 otherwise.

It is not obvious but we can show that the value of Q always relies between -1 and 1. A value of Q close to 1 indicates a good clustering where all the edges inside clusters are high above the average values of edges of specific nodes.

The goal here is to assign values to all c_i of all cells i to maximize Q. For instance, if all c_i are equal (that is, all cells belong to the same cluster), then $Q = 0$. The same happens if all c_i are distinct (ie, every cell forms a singleton cluster). It is entirely possible that these extreme cases are our optimal solution, but in most single cell analyses, cells naturally have subgroups of phenotypes with matching principal components

that can be discovered upon optimizing Q.

One thing to note is that the parameter k - the number of nearest neighbors we used to build the graph - can change the final solution. If $k = 0$, for example, then there are no edges in the graph and $Q = 0$. Similarly, if k is equal to the total number of cells, then all cells will be connected and, as we've seen above, $Q = 0$ as well. Seurat tries several different values of k that yield the value of Q as close to 1 as possible.

The biological identity of these clusters can further be validated by analyzing specific marker genes known a priori to influence the identity of the cells.

Figure 7: *The KNN graph from the Nephron Progenitor Cells of the 16 Week Fetal Kidney Dataset. Cells are colored by their inferred clusters as to maximize Q.*

2.6 Differential Expression of Clusters

In the Louvain Modularity clustering, we did not take into account the influence of particular genes or PCs in the identity of the clusters. Instead, each PC weighs equally when calculating the Euclidean distance. Given the identity of the clusters, we may instead be interested in which genes are upregulated in each individual cluster when compared to the cluster outside. For this we use a likelihood ratio test that is suited for zero-inflated samples such as those of single cell RNA-Seq as proposed by McDavid (**?**). If we fix a gene j in cluster i, we model the raw reads of gene g as a zero-inflated log-normal distribution. This distribution has 3 parameters: The probability π_{ij} of drop-out, and the mean and variance μ_{ij} and σ^2_{ij} . Similarly, let's define the same values but estimated from all the cells in all clusters except cluster i, that is: $\pi_{\tilde{ij}}$ is the drop-out probability outside of cluster i, and similarly $\mu_{\tilde{ij}}$ and $\sigma^2_{\tilde{ij}}$. The hypothesis tests at hand here are thus the following:

Alternative Hypothesis: $H_a : \pi_{ij} \neq \pi_{\tilde{ij}}$ and $\mu_{ij} \neq \mu_{\tilde{ij}}$ (ie, there is differential expression)

Null Hypothesis: $H_o : \pi_{ij} = \pi_{\tilde{ij}}$ and $\mu_{ij} = \mu_{\tilde{ij}}$ (ie, there is no differential expression)

A likelihood ratio test based on this model gives us a p-value for every gene and every clusters.

An immediate problem with this approach is the choice of p to reject the hypothesis: If there are c such clusters and g such genes, we are making $c \times g$ hypothesis tests. Let us remind of the meaning of the p-value we calculate above: p is the probability of rejecting the null hypothesis (that the genes actually have the same mean expression both inside and outside of the cluster) when the null hypothesis is actually true. If $p = .01$, for example, we have a 1% of detecting a false positive by chance. This makes the choice of a statistically significant p rather difficult. If, for example, we have $c = 10$ clusters and $g = 2,000$ variable genes, we're doing $c \times g = 20,000$ tests. Of course sometimes we will reject the null hypothesis just by random chance with so many tests!

A possible solution for this problem is to use the *Bonferroni Correction*, and make our p cutoff more restrictive. For example: If we would reject a single hypothesis test if $p < 1\%$ but we're doing instead $20,000$ tests, the idea is to make our p value cutoff smaller by dividing it by the number of tests we do. So we would only consider a gene statistically significant between two clusters if $p < .01/20,000 = 5 \times 10^{-7}$.

There are pros and cons to this approach. The biggest drawback is that it may be sometimes too restrictive, that is, we would be throwing off genes that may be statistically significant and characterize a cluster. Practical experience with the data, however, shows that there are always thousands of genes in each cluster with $p < 5 \times 10^{-7}$, and most of the times we're not worried about a particular gene, but rather sets of genes that may give us insights on the biology regulating the cluster identity. A restrictive cutoff would hence not critically compromise the identity of this set of genes.

2.7 Network Construction*

At this point every individual cluster has a set of genes that are upregulated with good significance. A subsequent question of interest is: Which sets of these genes are highly correlated or anti/correlated to each other? This may seem like a very similar question we answered using PCA, but there's two fundamental differences: First, we're now analyzing individual clusters rather than the entire dataset, and this can certainly cause correlations that existed before not to exist anymore. Second, we do not weigh in on the importance of the variance of individual genes. For instance, if two genes have constant expression across the entire dataset, these would be certainly thrown away in the first steps of the pipeline since they don't yield any important information about the heterogeneity in the data. However, if two genes that have constant expression inside a cluster whilst passing the differential expression test, we would be inclined to believe that they have similar biological function and may belong to the same gene network.

This is the foundation of the theory of network construction proposed by Horvath (). The WGCNA method builds yet another graph, but this time between genes. Each pair of genes i and j is joined by an edge whose weight is given by their correlation coefficient:

$$\rho_{ij} = \frac{\sum_k (g_{ki} - \bar{g_{\cdot i}})(g_{kj} - \bar{g_{\cdot j}})}{\sqrt{(g_{ki} - \bar{g_{\cdot i}})^2}\sqrt{(g_{kj} - \bar{g_{\cdot j}})^2}}$$

Here g_{ki} (resp g_{kj}) is the expression of gene i (resp, j) in cell k and $\bar{g_{\cdot i}}$ (resp $\bar{g_{\cdot j}}$) is the average expression of gene i (resp j) across all cells in the cluster. The value of ρ_{ij} varies between -1 and 1. $\rho_{ij} \approx -1$ means that the two genes are *anti-correlated* (that is, one increases when the other decreases), whereas $\rho \approx 1$ means that they are correlated, that is, they increase and decrease in a similar way. When $\rho_{ij} \approx 0$ it means that the expression of the two is uncorrelated, which we expect to happen with most pairs of genes.

In reality, we do not use ρ_{ij} for this network construction, but rather some power of the correlation. That is, the actual edge value will be given by $s_{ij} = \rho_{ij}^{\alpha}$ for some value of $\alpha > 1$. The reason for this is to "weaken" the influence of genes that may have a positive/negative correlation simply by random chance.

Finally, given that many of the edges will have a value very close to zero - that is, genes with no biological correlation whatsoever - we cut off edges by setting a minimum correlation threshold τ and set the final value of the edges as $a_{ij} = s_{ij} \times \delta_{|s_{ij}|>\tau}$. That is, we only keep edges such that $|s_{ij}| > \tau$.

How do we choose α and τ? This is outside of the scope of the analysis overview, but in short we want α to have the property that any edge s_{ij} can be written as a factor $s_{ij} = s_i \times s_j$. As for τ, we choose it based on a criteria called *scale free topology*, that is, the number of genes $p(k)$ whose sum of edges is k must be given by a power law: $p(k) \sim k^{-\gamma}$, so fitting a linear model on $\log p(k)$ and k and maximizing the R^2 statistic for several different values of τ gives us the optimal solution.

2.8 Ontology Enrichment Analysis*

In many cases it is possible to make safe assumptions about the biological properties of cells solely based on the behavior of marker genes and the networks. An extra validation of those properties can be given by comparing the gene lists to an ontology database. We use the topGO package () that uses the PANTHER database (www.geneontology.org) to test for ontology enrichment.

The Gene Ontology is structured as a directed acyclic graph where the 3 root nodes have very general terms (biological process, cell cycle and molecular function) and each node goes from a more generic to a more specific term. Each term has a set of genes associated with that biological function.

When we have a list of genes (eg, from a differential expression table or from PCA), a common question that arises is: What GO categories have more genes than usual? As a statistics question, this means: What are the categories that have more genes than what we would expect if we were to select the same amount of genes randomly across the genome? We solve this using *Fisher's Exact Test* to find a p-value for every GO category in the organism of interest. We sort the GO terms in increasing order of p-value and assume the smallest p results are a good indication of the underlying biology of the selected genes.

2.9 Pseudotime Analysis*

In the context of cells continuously differentiating from one phenotype into another, it is often desired to be able to reconstruct the differentiation pathway they have undergone. We assume our sample consists of cells randomly sampled from the early to late differentiation pathway, and we use the distances between their transcription profiles to be able to rebuild their transition path. The most common approach for this is the Monocle () algorithm.

Once again in Monocle we build a graph where we connect all pairs of cells by their euclidean distance as definied in Section 2.5.1. The assumption here is that two cells that are adjacent in the differentiation path will have a small distance (ie, a small difference of their expression profiles). Monocle attempts to find an ordering of the cells such that the sum of all the edges between adjacent cells is as small as possible. This is known in the computer science literature as the *traveling salesman problem*. Though the construction of this pathway and subsequent decisions on if and where we should branch the path are rather loaded technically, it suffices to say that Monocle finds a good approximation of the most likely pathway the cells underwent.

It is important to notice that it is crucial that the dataset we use to reconstruct the pseudotime actually makes sense. If we, for instance, apply the method to a set of many unrelated cell types, Monocle will actually ”force” them into a reconstruction pathway, which certainly might be undesired.