Gene expression clustering applied to scRNAseq: data analysis, gene expression modeling, and benchmark of clustering algorithms

C. REDA

M1 student in Computer Science at ENS Cachan

2016–2017

supervised by G. ILSLEY and N. LUSCOMBE

Okinawa Institute of Science and Technology, Japan

March 1^{st} to July 31^{st} of 2017

Abstract

Genome is the key to understand the mechanisms behind life, that is, how an fertilized egg becomes an embryo, and then a fetus; how, from one single stem cell, thousands of cells can be generated, and each of them have a specific role to play in the organism. Every cell contains a copy of the genome, and now, biologists can get access to the coding sequence contained in one single cell, thanks to recent technologic breakthroughs. When regular RNA sequencing - also called bulk RNA sequencing- could only provide a insight of the average cell activity in one organism, single-cell RNA sequencing (scRNAseq) allows to take a snapshot of the activity in each cell, and to better understand the role of a certain cell, and the inter-gene interactions, in the considered organism.

With such a technique, huge amounts of data about the gene expression in each studied cell can be extracted. How to properly analyze them, in a reasonable time, while taking account of the various error sources in the measurements, still remains a burning question. An interactive and user-friendly online application to analyze single-cell RNA-sequencing data has thus been developed, and allows the exploration of the scRNAseq data from two organisms, *Ciona intestinalis* and *Caenorhabditis elegans*. Other datasets can also be easily added.

More specifically, cell clustering based on gene expression levels -referred as "gene expression clustering" here for short- that is, grouping cells in order to better understand their functions in a certain organism, is a topical issue. Moreover, dozens of algorithms, using several different methods, have been developed to tackle this problem for single-cell RNA data, even through it is still quite a recent field. However, none of them has been selected yet as the reference clustering algorithm. Thus there is a need to perform a benchmark, in order to compare the clustering results, and to check the correctness of the resulting functional cell families found.

Also, modeling the gene expression for single-cell RNA data is of paramount importance, to control the quality of sequencing results for instance. In recent techniques, independence of expression between different genes is still widely assumed, although real data show that this assumption does not stand in practice. A model, leading to a new clustering technique, and an implementation in R are introduced to overcome this issue.

Contents

List of Figures

List of Tables

Chapter 1

Context

This section introduces several main biological concepts, in order to understand the following more computational parts. It is meant to at least give an intuition for the biological phenomena studied here.

1.1 Biological background

From cell to gene expression level

Every cell contains DNA, or **DesoxyriboNucleic Acid**, which is a molecule containing sequences of nucleotides[160] -there are four **bases**: adenine, guanine, cytosine, thymine- that governs life. Several steps are required in order to get a protein -also called **polypeptide**- which is a chemical molecule that performs a certain action in the organism, from a gene-coding sequence present in the genome (see 1.1).

Firstly, during transcription (see Figure 1.2), inside the cell nucleus, some parts of the DNA strand are read by the RNA polymerase -which is a molecule that can produce RNA, or **RiboNucleic Acid**- and converted into a molecule called messenger RNA (**mRNA**)[26]. Note that, as a general rule, the mRNA copy of the DNA fragment does not match perfectly the initial DNA chunk, because only some **introns** (non-coding parts of a sequence associated with a given gene) and **exons** (coding regions) will actually be selected. The set of all RNA molecules (called **transcripts**, mRNA included) is called **transcriptome**.

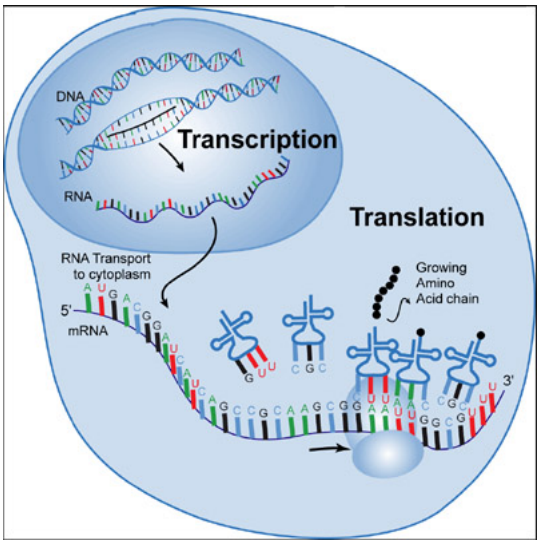


Figure 1.1: From the initial DNA sequence to the protein[38]

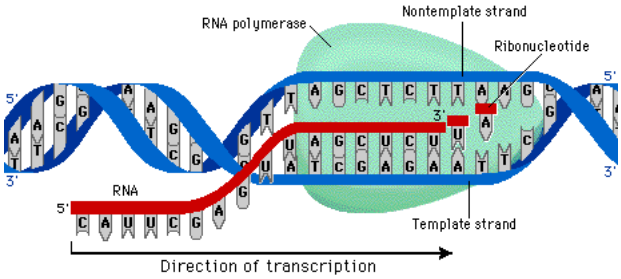


Figure 1.2: Transcription of the DNA strand into mRNA[123]

Once the transcription is performed, this molecule exits the cell nucleus, and later, during the translation (see Figure 1.3), the RNA sequence is read by the ribosomes. The **ribosome** is a macro-molecule which associates an amino acid to each triplet of nucleotides -called **codon**. A protein is then the resulting sequence of amino acids.

It is worth noticing that mRNA is unstable[26] compared to DNA, that is, as a general rule, it cannot be read but a limited number of times by the ribosomes. Hence getting access to the sequence carried by a mRNA molecule provides a good idea of the number of produced proteins by the cell, thus of the cell activity and of the expressed genes; as opposed to the study of the DNA sequence, which can be considered rather as a identifying feature for a certain species or individual when compared to other organisms.

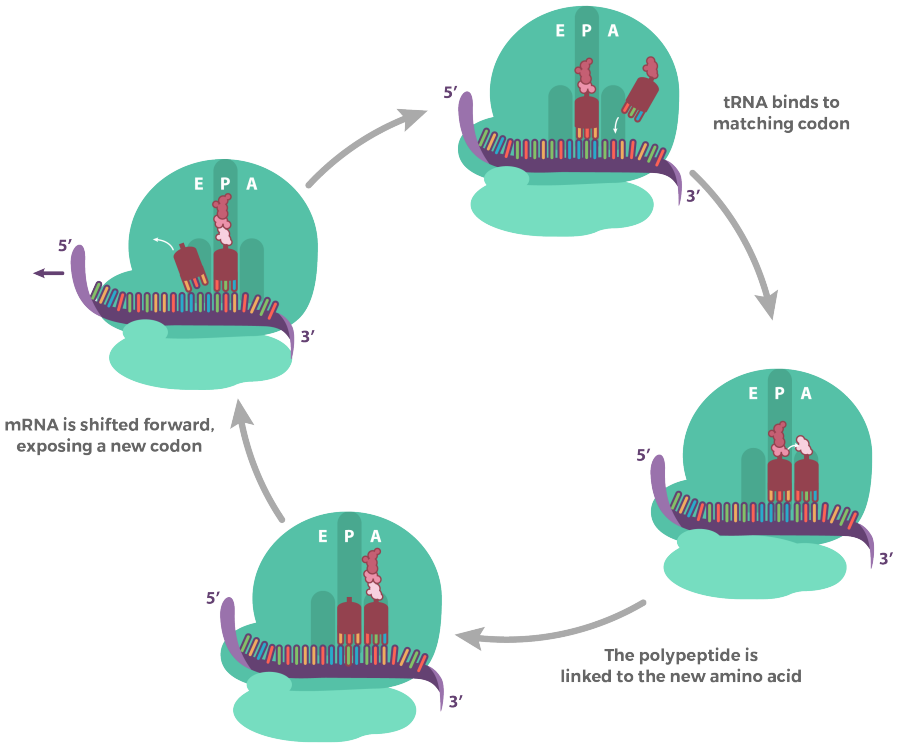


Figure 1.3: Translation of the mRNA into proteins[4]. The green unit is the ribosome, and E, P and A are the three sites where translation occurs.

A gene is considered expressed when the associated coding sequence is transcribed from the initial DNA sequence to the RNA sequence carried by the mRNA, because the protein it codes is likely to be produced by the ribosomes.

Cell clustering based on gene expression levels

It is assumed that the levels of gene expression, for the different genes which coding sequences have been seen in the mRNA sequence extracted from a certain cell, determine the cell function in the organism.

The features which are the most discriminative in the grouping of cells, and which show a strong correlation with a given cell phenotype, are called **informative genes**[103].

For a given cell, the **gene expression pattern** is its profile of gene expression, which is assumed to be specific to a certain functional family of cells, i.e. cells having the same role in the organism (see [14] for a discussion for this hypothesis).

Being able to find the informative genes, and to distinguish a gene expression pattern for each cell, can then provide a better insight of the cell organization in a certain species[150], or of gene expression heterogeneity between different developmental stages[57], or a better understanding of the difference in cell activity between cancer-afflicted and healthy individuals[149], that may lead to the development of more efficient treatments, for instance. Note that gene clustering based on gene expression levels exists too, and may use similar methods as cell clustering[42][79].

Hopefully, when a proper subset of informative genes has been selected, cells may be clustered according to their functional families or their cell type (see Figure 1.4).

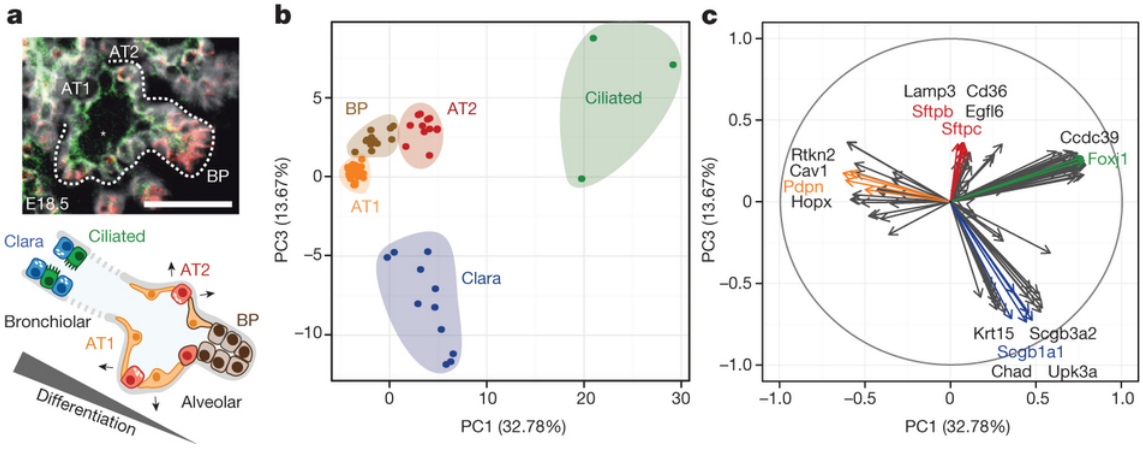


Figure 1.4: Example of cell clustering based on gene expression data. Each cell functional family is represented by a distinct colour[153]

1.2 About RNA sequencing

Purpose

RNA sequencing is aimed at obtaining the **primary structure** of a certain RNA fragment, that is, the ordered series of letters A, G, C and U, which respectively refer to the following nucleotides: adenine, guanine, cytosine and uracil.

As explained above, most of the time, DNA and RNA sequencings serve different purposes. DNA sequencing allows to get access to the code needed for living things to survive, and may be helpful to retrace the evolutionary history of a given species, whereas RNA sequencing can cast a light on the regulatory inter-gene interactions (has gene A an influence on the expression level of gene B? Does it favour/inhibit the transcription of this gene?) and on the developmental phase of an organism, from the fertilized egg to the fetus: how do the different limbs form? How do some stem cells turn into specialized cells, having only a very specific function in the organism? (This phenomenon is called **cell differentiation**.) However, how regulatory systems exactly work is still unknown, and DNA may be useful in the future to understand whole regulatory processes.

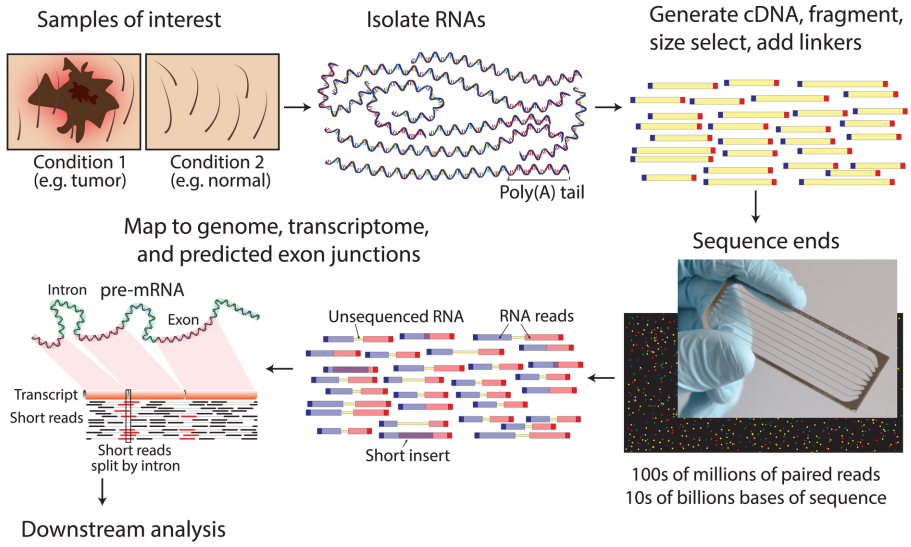


Figure 1.5: General pipeline to get a gene expression matrix from a mRNA fragment[59]

Sequencing biological material requires several steps (see Figure 1.5).

1. First, the mRNA molecule in the samples are chemically isolated.
2. Then, these molecules are used to generate (complementary) DNA (cDNA) via **reverse-transcription**.
3. Later, these cDNA strands are fragmented into **reads**. Reads are short series of A (which stands for adenine), G (guanine), C (cytosine) and T (thymine) letters resulting from the sequencing of the mRNA strand. Then these reads are sequenced, and the resulting reads are aligned to a reference genome (see Figure 1.6), that is available in a online database

such as WormBase[143] or NCBI[2], for instance. The best alignment is the one that preserves most the order of the bases in the reference sequence -but the exact definition of the "best alignment" highly depends on the alignment algorithm used.

4. Then a gene expression matrix M is computed: each time a read from condition j is aligned to a certain reference sequence corresponding to a specific gene i, the associated number $M_{i,j}$ is incremented.

Gene expression value is also called **gene expression level**, or **gene expression count** -when the numbers are integers. The data to analyze is then the integer matrix of gene expression value, that is, as a general rule, a matrix M which rows are indexed by the considered genes -or **features**- and which columns are indexed by the studied cells -called **samples**. Columns can also be called **conditions** (e.g., non-cancer versus cancer cells, or treatment A versus treatment B versus treatment C), depending on the study. Each coefficient $M_{i,j}$ is the number of times the coding sequence for gene i has been recognized in the resulting A, G, U, C-sequence obtained by the sequencing of the mRNA stand in cell j, that is, the number of detected reads in the sequencing of this mRNA strand that refer to this gene.

However, the total number of reads detected may vary from one cell or condition to another. In order to be able to compare gene expression values across several cells, the matrix may be trimmed (to remove outlier cells or genes), and "normalized" to preserve proportionality between gene expression values. There are actually many ways to achieve this normalization[104][37].

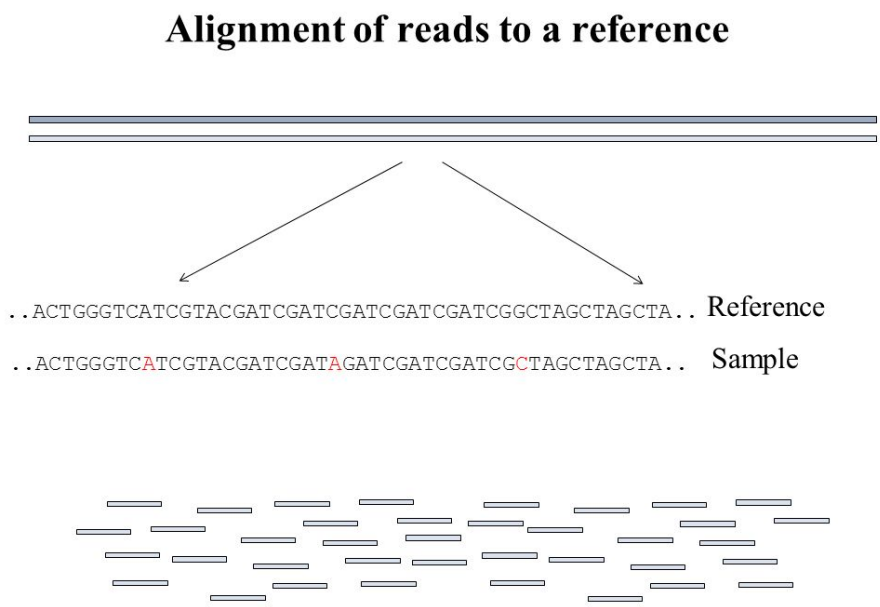


Figure 1.6: Read alignment[155]

My implementation of a pipeline to get a count matrix from a set of bulk RNAseq reads (Illumina sequence data, contained in one or two .fastq file-s), provided a reference genome (.fasta file) and the corresponding annotation file (.gtf), is available there: https://github.com/oist-gene-clustering/RNAseqpipeline

About single-cell RNA sequencing (scRNAseq)

scRNAseq is quite a new technology. At first, studying mRNA became affordable thanks to microarrays[30], that started to be widely used in 1995[134]. This technology used miniaturized and optimized high-throughput screening on arrays of biological data[167] (see Figure). The latter were also used for DNA studies ("DNA microarrays"[140]).

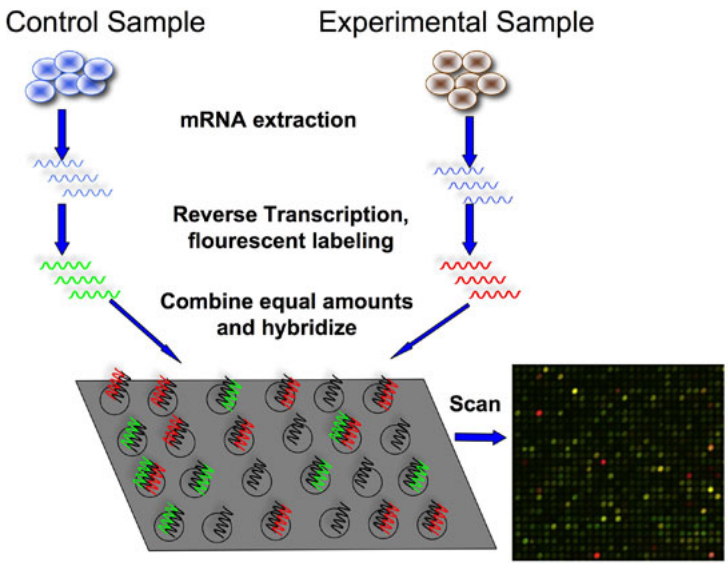


Figure 1.7: Microarray experiment for mRNA data[60]

At this time, tools for improving downstream analysis (that is, the analysis of the resulting gene expression matrix) were needed, and many clustering algorithms[43][40][45][114] and statistical tools[138][85] were designed.

However, in 2007, [15] used RNA-sequencing to detect chromosomal modifications in the transcriptome, via **reverse transcription** (see previous section). Because of its assets, comparing to microarrays -increased data quality[175][158], improved detection of rare transcripts, ...[75]- and the drop in the sequencing price, researchers' interest in regular or bulk RNAseq had soon plummeted in the 2010's and until now[39][101].

Until quite recently, only groups of cells could be isolated and sequenced. Henceforth, the gene expression levels obtained actually were average gene expression values across the cells from one given group. Thanks to scRNAseq, cell activity can now be witnessed at a higher resolution, thus give a more accurate report of the gene expression in a given single cell[142][124]. Nevertheless, there is doubt about the reusability of the previous methods and tools developed for RNA-seq and microarrays, due to the following phenomena being specific to single-cell RNA seq data:

- scRNAseq data are even noisier than bulk RNA seq, because of the low biological material available and procedures which can easily be contaminated[25][61].
- Unlike bulk RNAseq data, scRNAseq results are subject to **dropout** phenomena[86][105]: a lot of gene transcripts cannot actually be detected

(because of an error during reverse-transcription), which results in a gene expression matrix with a lot of zeros, and makes transcriptome profiling harder to perform.

- The higher resolution offered by single-RNA sequence provide a new insight on cells having a same cell type: the gene expression levels of certain genes may vary from one cell to another[124][107].
- It also allows to observe the evolution of cell gene expression profiles during **mitosis** (cell cycle)[151]. The mitosis is the cell division into two new cell copies.

Today, there are lots of techniques[137] for single-cell RNA sequencing, the first one being Tang et al.[147] (2009), see Figure 1.8.

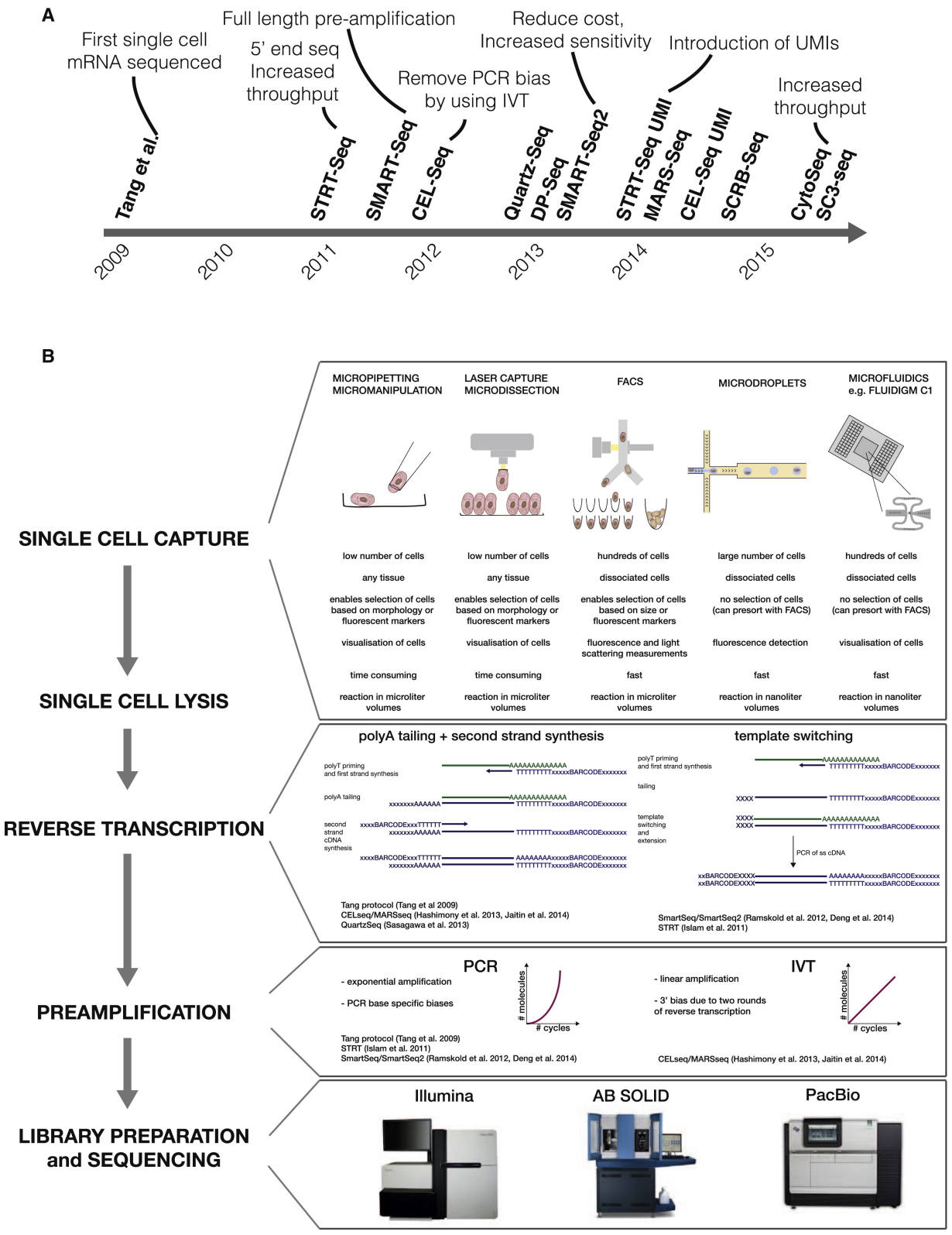


Figure 1.8: History of scRNAseq and general pipeline to get a set of reads from a single-cell RNA fragment[92]

My implementation of a pipeline to get a count matrix from a set of scRNAseq reads (Illumina sequence data, contained in one or two .fastq file-s), provided a reference genome (.fasta file) and the corresponding annotation file (.gtf), is available there: https://github.com/oist-gene-clustering/RNAseqpipeline

It uses softwares from [10][23][98][102][121][81][84][152].

1.3 Cluster analysis

Purpose

Cluster analysis, that is, grouping data samples respect to their similarity -same-group samples are more similar in some sense than to samples from other groups- is one of the primary steps for exploring data, and is meant to guide further analysis. Applied to scRNAseq, it may help discovering new cell functional families[80], control data quality[110] and delete outlier samples[87], or better understand cell differentiation at a given developmental stage[57].

The different groups -called **clusters**- are not known beforehand, neither are the features that can the most reliably discriminate the data samples. As a general rule, data samples are identified to vectors, where each component refers to a certain feature, that may be binary, real-valued, categorical, ... according to the type of data. Thus cluster analysis -also called clustering- boils down to grouping points in a certain mathematical space.

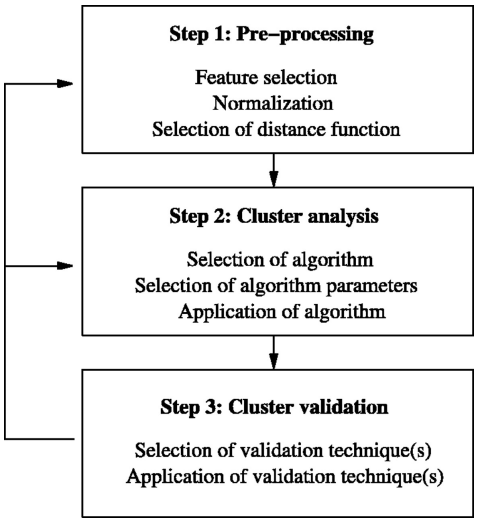


Figure 1.9: Pipeline of cluster analysis, from the raw data to the clustering validation[68]

Steps	**Selection of the following items**	**Corresponding questions**
Pre-processing	Feature selection, distance selection, data points normalization	*- What is the data about?* *- What are the relevant data features, and how to efficiently extract them?* *- Do I need to perform some transformations on the data points before comparing their vectors?* *- How do I compare data vectors?*
Clustering	Choice of the algorithm (and of corresponding parameters)	*- How do I efficiently get the clusters?* *- What is the algorithm input?*
Clustering validation	Internal/external measures	*- How do I check if the clusters are consistent with the original data (external measures)?* *- How do I check if the algorithm went wrong (internal measures)?*

Table 1.1: Cluster analysis steps (according to [68])

Clustering is not all about applying a certain algorithm to the data (see Figure 1.9 and Table 1.1). Each step above is of equal importance. More especially, validation step is of paramount importance, since there is no clustering algorithm that will be able to predict the best grouping of points for any kind of data -with more than 2 points[90]. The choice of the algorithm may determine the shape of the resulting clusters (e.g. spherical for K-means), their size and their density, for instance. Thus having an idea of the clustering error can be helpful.

Moreover, feature selection, especially for gene expression clustering, is also a critical step. Since only a handful of genes are really informative, that, provide insight on the cell functional family, the remaining of the features is but noise, thus useless, and may also prevent an algorithm to find the patterns. There is currently no common definition of what an informative gene should be, and many methods exist[71][82][99]. Most of the time, a gene is considered informative if the variance of its expression across the cells is high, and if they are considered independent -this leads to the use of PCA on the gene expression matrix. However, PCA actually limits the number of informative genes to the minimum of the dimensions of the gene expression matrix M (because the loading vectors it computes are the eigenvectors of $M^T M$, in increasing corre-

sponding eigenvalue order), and assumes a linear relationship between the gene expression values

Definition 1.3.1 (Kendall's τ)**.** *Let $n \in \mathbb{N}^*$, $X = \{x_1, x_2, ..., x_n\}$ and $Y = \{y_1, y_2, ..., y_n\}$, such as X and Y have unique orderable values.*

Let a denote the number of pairs $(x_i, y_i), (x_j, y_j), i, j \in \{1, 2, ..., n\}$ such as $x_i < x_j \wedge y_i < y_j$ OR $x_i > x_j \wedge y_i > y_j$, and b denote the number of pairs such as $x_i < x_j \wedge y_i > y_j$ OR $x_i > x_j \wedge y_i < y_j$. Then:

$$\tau = \frac{a-b}{\frac{1}{2} \times n \times (n-1)}$$

State-of-the-art

Several different methods have been designed for clustering (sc)RNAseq (see Tables 1.2 and 1.3).

Category	**Description**	**Implementations**
Connectivity-based	These algorithms build a hierarchy of the samples (i.e. a tree which leaves are the samples), according to the chosen distance measure. Given a height, the final clustering is the set of trees in the forest obtained by deleting edges higher than the selected height.	Hierarchical clustering (SINCERA[65]), ...
Centroid-based	Given the expected number of clusters k, these algorithms compute iteratively k points, such as each of them is the centroid of a set of similar points. The final clustering is obtained by assigning each point to the cluster of the nearest centroid. These steps are repeated until convergence of the clusters.	K-means[106][154], Fuzzy C-means[44][47], Nonnegative Matrix Factorization[100] (e.g. NIMFA[136]), Nearest Neighbor-like (e.g. SNN-Cliq[169]), ...
Density-based	Clusters are areas of higher density (in the number of data points) compared to the remainder of the data points. These algorithms try to detect local differences in density in order to find the clusters.	DBSCAN[46], ...

Table 1.2: Some categories of clustering algorithms with examples from gene expression clustering (1) (adapted from [166])

Distribution-based	A mixture of k same-model distributions (where k is the expected number of clusters) is fitted to the data, that is, the different parameters of the mixture are such as they maximize the likelihood of the observed data. Then the probability of membership of each point to every cluster is computed, and the final clustering is the one maximizing the membership probabilities.	Gaussian Mixture Models (e.g. EMCluster[112], MCLUST[49]), *ad hoc* models (e.g. Seurat[131]), ...
Discriminative feature computation	These algorithms first compute a somehow summary of the most relevant features of each point, either by representing them by the restrictions of their associated vectors to the components having the maximum of variance (PCA), or by mapping the data vectors to 2D (resp. 3D) vectors such as the 2D (resp. 3D) representations of two similar vectors in the original space are close in the new space (t-SNE). Then one of the algorithms above is applied to the resulting representations.	Principal Component Analysis (e.g. pcaReduce[172]), t-Stochastic Neighbor Embedding (e.g. t-SNE+K-means[62], t-SNE+DBSCAN), ...
Consensus clustering	After having selected one of the algorithms described, and iterated it several times with different values of parameters, these algorithms construct a consensus matrix M such as: $M_{i,j}$ is the (mean) number of times points i and j are located in the same cluster. M is then used to determine the final clustering.	e.g. SC3[87], ...

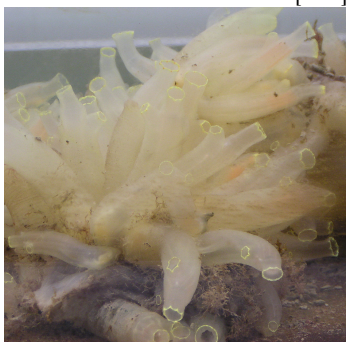
Table 1.3: Some categories of clustering algorithms with examples from gene expression clustering (2) (adapted from [166])

Chapter 2

scRNAseq analysis application

Along with scRNAseq, huge amounts of gene expression data are available. Especially for newly-scRNA-sequenced species, such as *Ciona intestinalis*[146] or *Caenorhabditis elegans*[150], one would like to *explore* the data, that is, being able to visualize the genes and cells that are the most informative about a certain condition, observe the cell correlation depending on certain selected genes of interest, distinguish genes that are the most discriminative between cell functional families or conditions (this is **differential expression analysis**), etc. This is a part of the downstream analysis that can be performed on the gene expression matrix obtained after sequencing.

Figure 2.1: *Ciona intestinalis*[165] (left) and *Caenorhabditis elegans*[164]



I thus implemented an interactive application in order to perform these tasks, to which other datasets can easily be added. The code (in Shiny, a R framework) is available here:

Packages: R[125], shiny[33], scater[109], rPython[18], RColorBrewer[116],

NMF[54], MAST[111], ggplot2[163], genefilter[55], gplots[159], calibrate[58], shinyBS[13], shinythemes[31], shinydashboard[32], shinyjs[12], DT[168], shinycssloaders[129], shinyDND[73], ggfortify[148], ggbiplot[157], plotrix[77] and shinyWidgets[120].

2.1 Objectives

After obtaining the gene expression matrix, one may be interested in knowing the number of subpopulations of cells, that is, of groups of similar cells, but also which genes are expressed differently from one cell group to another (**differential expression analysis**), or visualizing the whole cell population in the observed data (see Figure 2.2). One may also would like to know the **gene expression pattern** for each feature, that is, the sets of most expressed and less highly expressed genes in the cell.

There is already an application, called SAKE[72], performing clustering (with Non-negative Matrix Factorization, see below for a description) and differential expression analysis. It is a really complete and interesting application. However, there are a lot of options, thus making the application not really intuitive, and does not display the gene expression patterns. Moreover, it uses DESeq2[104] to perform differential expression analysis, which is only a slightly modified version of a differential expression analyzer[8] for bulk RNA seq data, thus not taking into account some characteristics of sRNAseq data, such as the expected high number of null gene expression values (DESeq2 models gene expression as only following a negative binomial distribution, which is the one bulk RNAseq is assumed to follow).

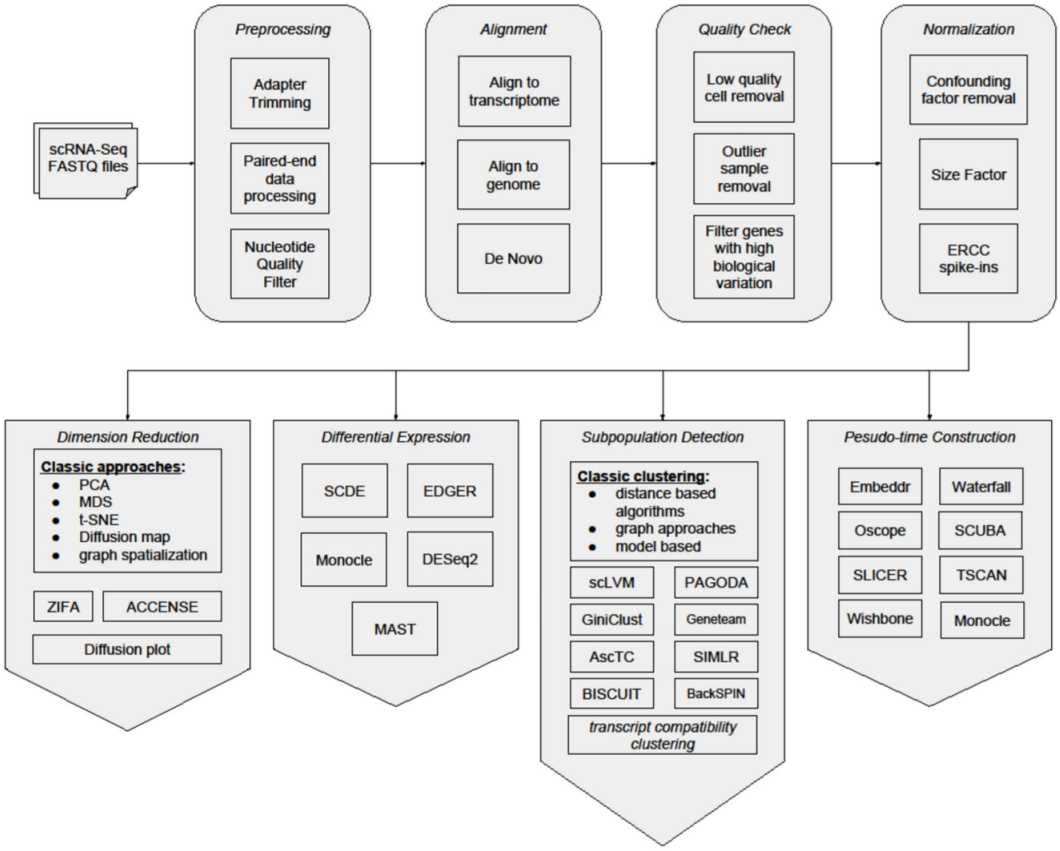


Figure 2.2: Possible downstream analysis pipeline after alignment and quality check of the resulting gene expression matrix[124]

2.2 Implementation

This is an example about how to use the application with the *Ciona intestinalis* dataset[146] (the dataset can be modified using the button on the top left of the application front page). I will compare the results obtained with the application (for cell B5.2 against B5.1, A5.1, A5.2, a5.3, a5.4 and b5.3, b5.4, and genes KH.C9.853 and KH.C13.27) and Figure D, page 382, in [108].

In the **Data visualization** tab (see Figure 2.3), the front page displays a Principal Component Analysis (PCA) plot, indicating each cell type and embryo by a certain shape and colour, which is meant to give a visualization of all cells in the dataset, and help selecting the relevant cell groups.

On the right side, the heatmap is coloured according to the number of raw

counts in each cell for each of the $min(10, p)$ most highly-expressed genes (where p is the number of genes), in order to visualize the potentially gene outliers[124]. On the sides can be seen hierarchical clusterings of the cells (on top), and of the genes (on the left).

The two groups (or conditions) to compare can be selected by drag and drop (see Figure 2.4).

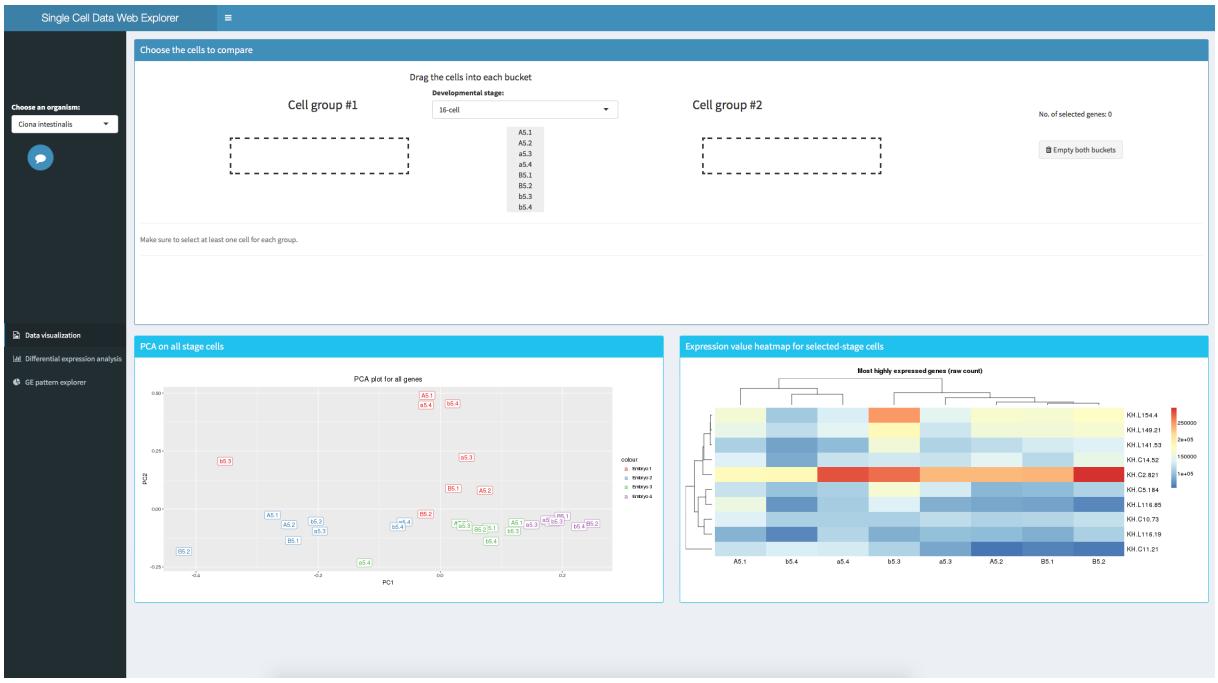


Figure 2.3: Menu

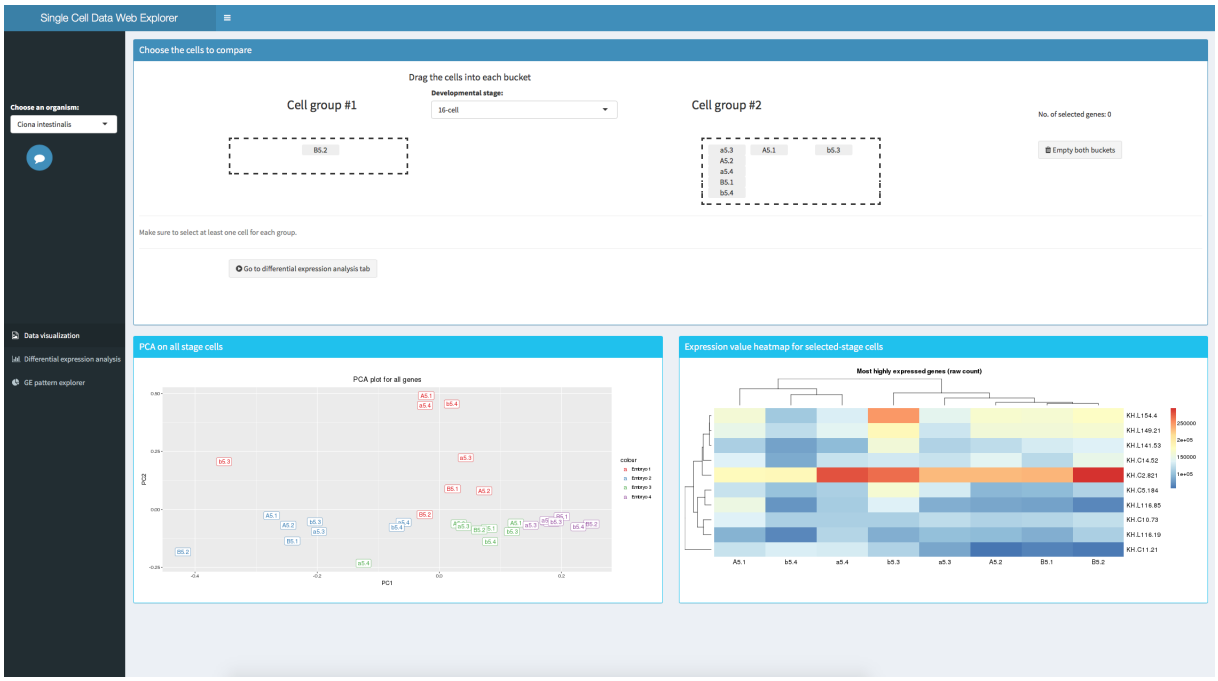


Figure 2.4: Cell group selection

At first, the **Differential expression analysis** tab (see Figure 2.5) allows the user to view the raw counts of every gene in every cell of the dataset. Then genes can be selected directly from the table (or using the provided buttons). The user can search for a given gene with the search bar on the top-right corner of the table. The PCA biplot can be updated: the PCA is then performed on the currently selected genes.

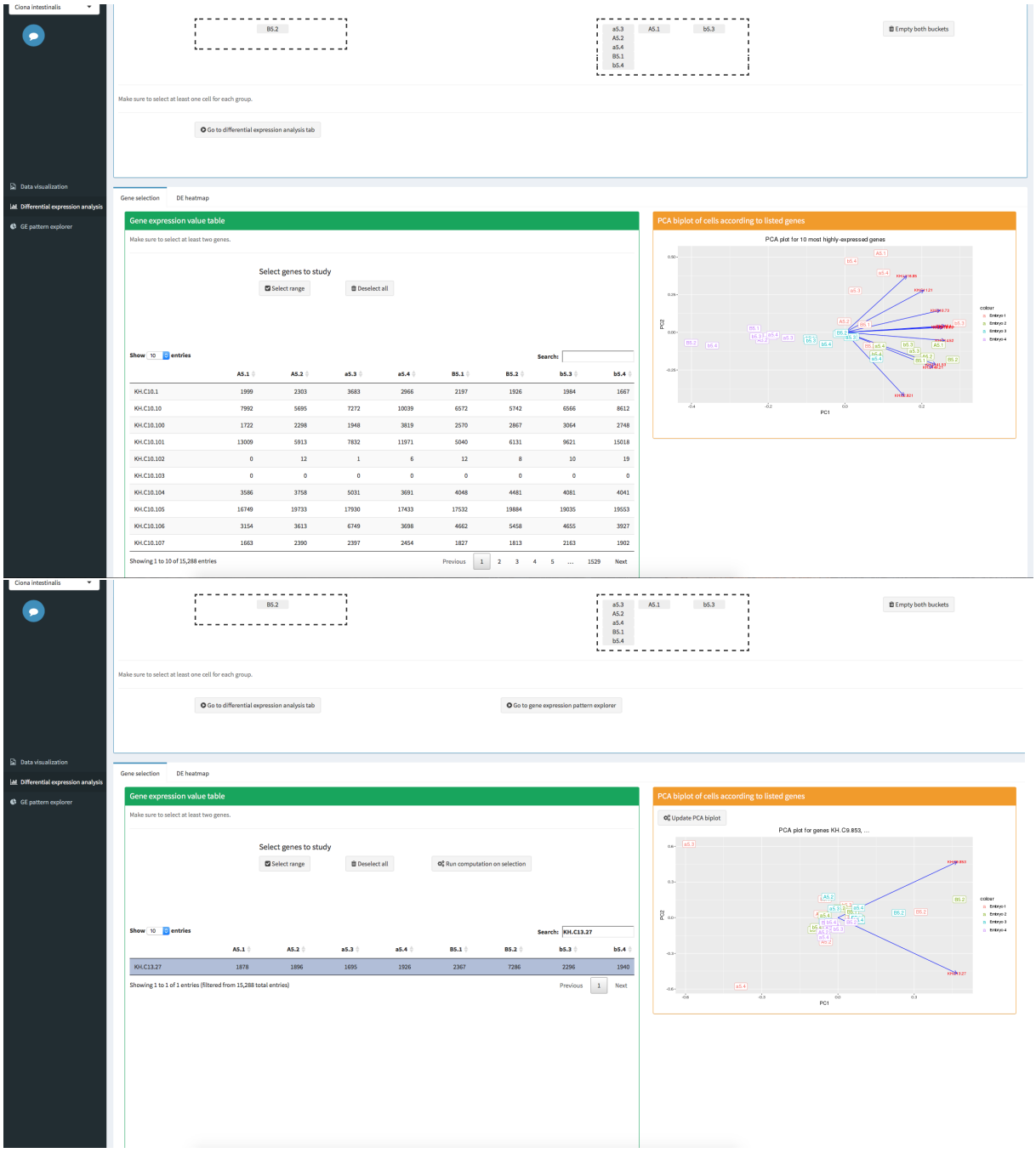


Figure 2.5: Gene selection panel

Differential expression analysis pipeline (from MAST package[48]) is performed on the selected genes and cell groups when clicking on "Run computation on selection" button (see Figure 2.5). After a few seconds of computing, the front page displays the table with genes ranked in increasing **p-value** order (see Figure 2.6). The p-values are a way to quantify the probability that a certain gene can be differentially expressed *only by chance* between the two cell groups: the lower a p-value is, the more the associated gene is likely to be differentially expressed in practice.

In the other tab, there is a heatmap with the selected cells and the most **differentially-expressed genes** (the genes which expression level strongly differs according to the considered cell group), with associated hierarchical clusterings (on genes on the left, on cells on top of the heatmap (see Figure 2.6).

Moreoever, the application offers now the opportunity to select genes according to their **p-values**. The cutoff value is user-selected with the slider bar (see Figures 2.7).

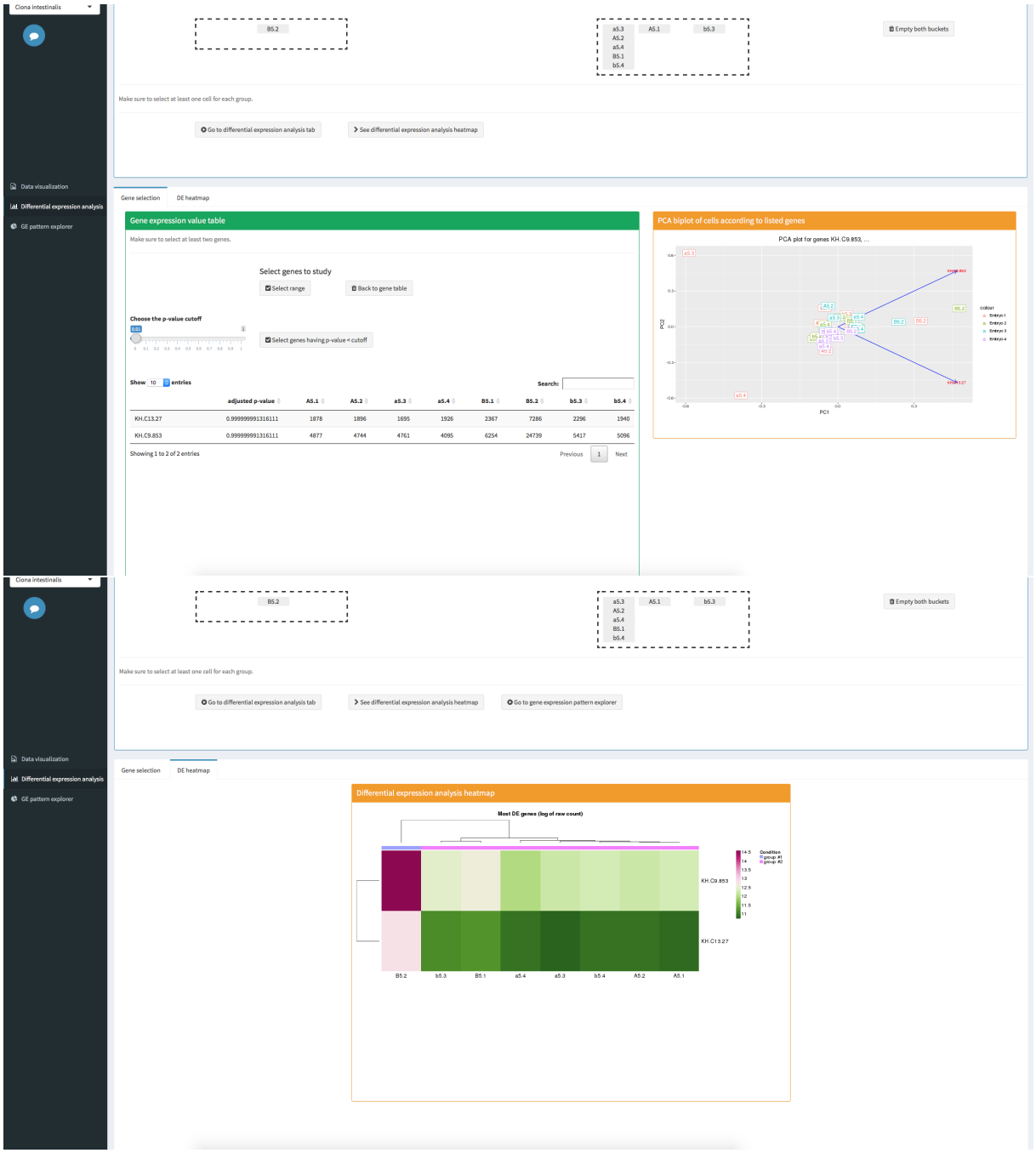


Figure 2.6: Differential expression analysis results

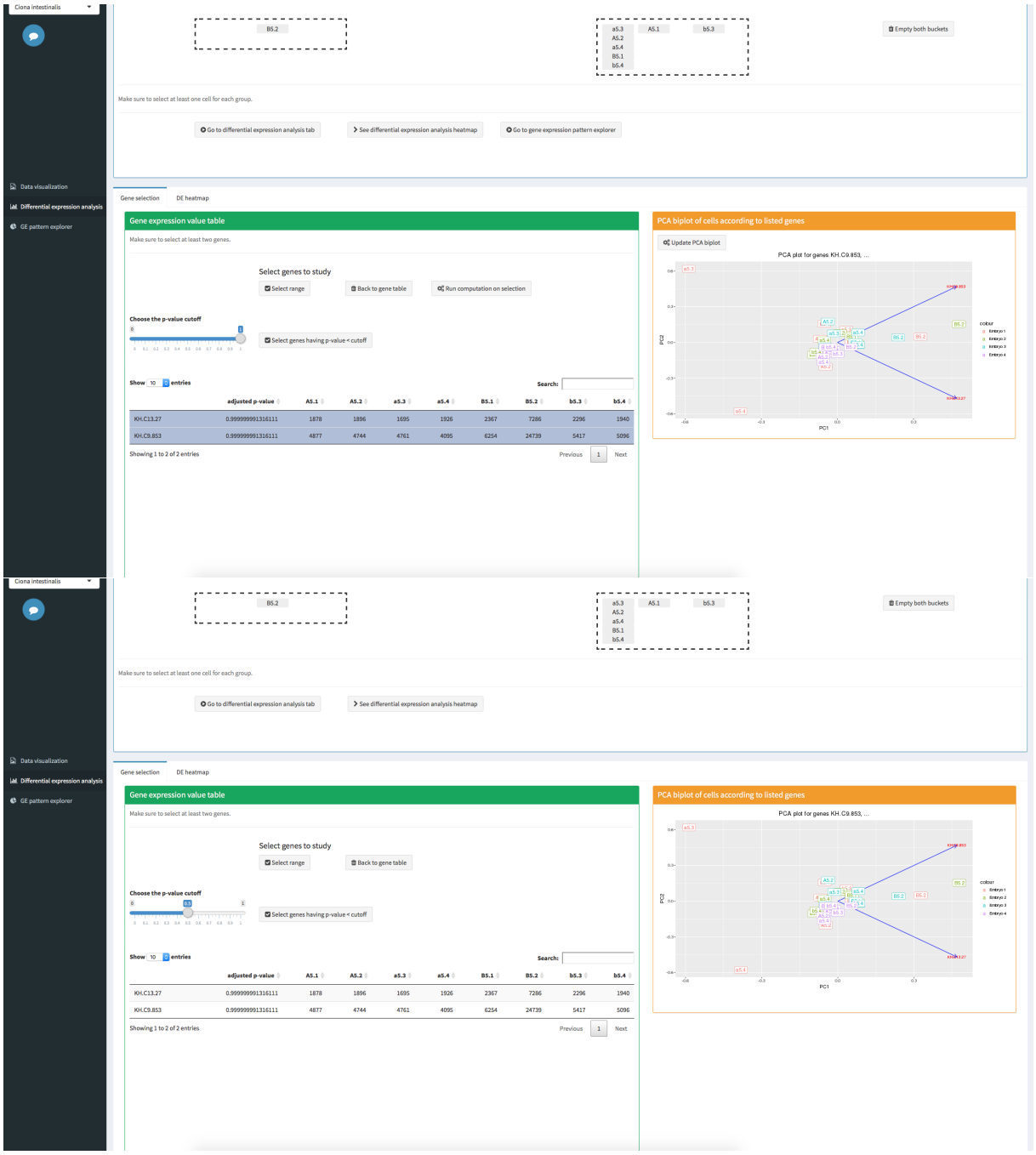


Figure 2.7: Gene selection with p-value cutoff

In the **Gene expression pattern visualization** tab (see Figure 2.8), when the selected threshold type is set to "Binary", the front page displays a schema of the selected cells in each group, where each rectangle represents one of the selected genes.

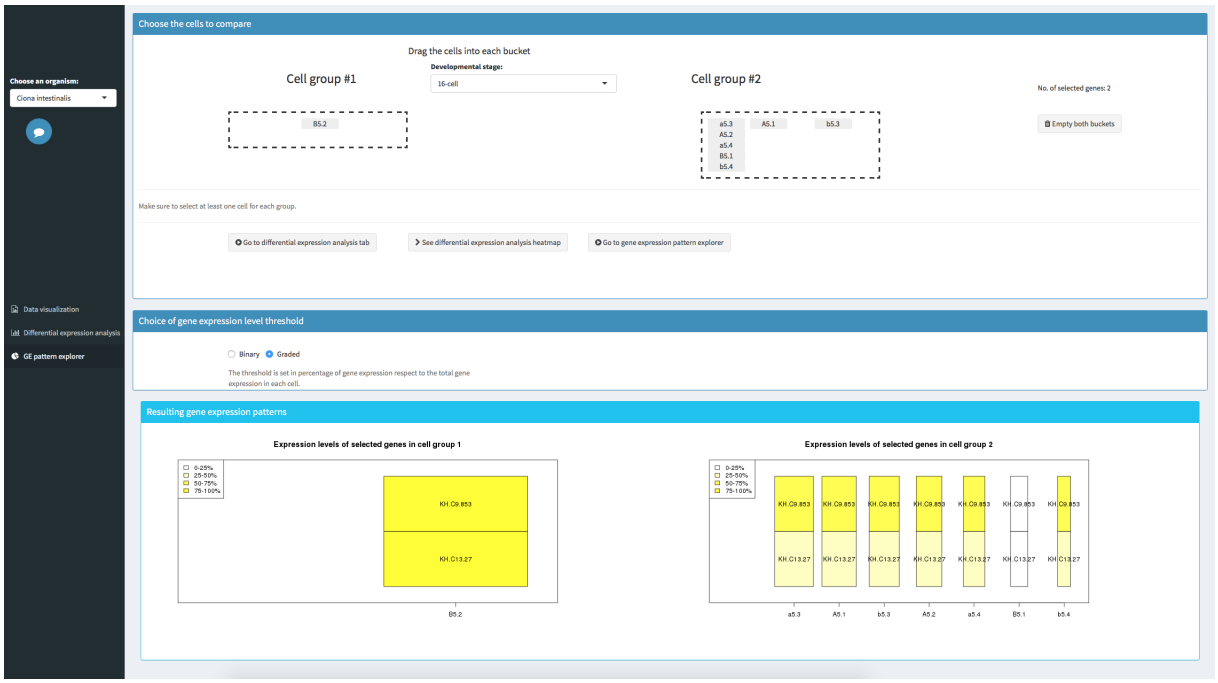
Given the threshold value (in percentage of the total raw count expression value of a given gene in a given cell), the rectangle is colored if and only if the gene expression raw count value of the gene in the cell is over this threshold. For instance, if the threshold is fixed to 10% of the total expression value, the rectangle associated with gene i in the set of rectangles corresponding to cell j will be colored if and only if (M being the gene expression matrix) $M_{i,j} > 10\% \times \sum_k M_{k,j}$. It should be kept in mind that cell gene patterns cannot be compared directly, since the expressiveness of a given gene is only determined by the ratio gene expression value/ total gene expression value in the considered cell.

The threshold value is user-selected with a slider bar (see Figure 2.8).

When the threshold type is set to "Graded", the gene expression patterns are colored according to the legend displayed on the left, the same way as with binary threshold (see Figure 2.9).

Eventually, a script is also provided to make adding new datasets to the application easier. Source code and other tests can be found at: https://github.com/oist-gene-clustering/scwebexplorer

Figure 2.8: Visualization of gene expression pattern with selection of a binary gene expression threshold



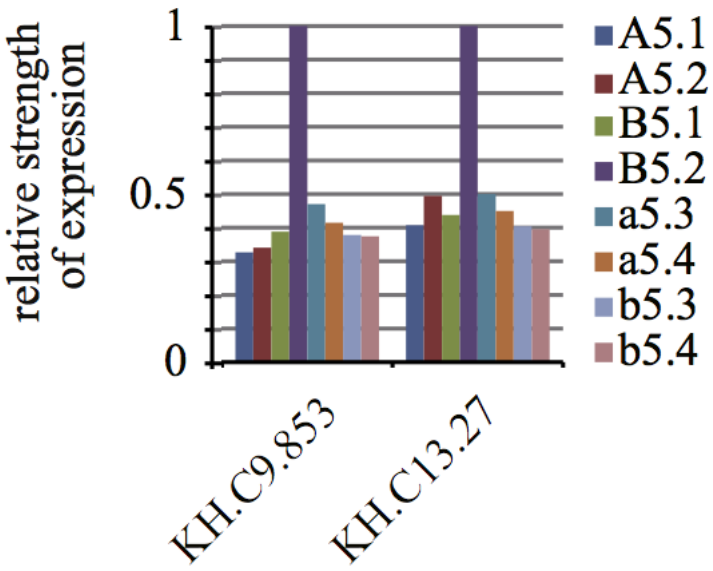


Figure 2.9: Visualization of graded gene expression pattern and comparison with Figure B from [108]

Chapter 3

Modeling of single-cell gene expression

What determines the gene expression level for a given gene, in a given cell, for a given individual, who belongs to a given species?

Because of the different factors (gene, cell, sequencing errors, etc.), trying to predict the absolutely exact gene expression count is indeed vain. However, being able to predict a relatively accurate gene expression value is enough for research purposes, and it has been studied intensively for data quality control softwares[86][48][104][37]. The model of gene expression is also implicitly chosen in the clustering algorithms.

However, such hypotheses as independence of gene expression between two genes, even in the same cell, are still widely assumed[14], even though they are clearly dismissed by the systems biology theory: genes *do* interact with each other, favoring or inhibiting the expression of the other genes through regulatory paths. This assumption is also implicitly made by most of the algorithms introduced in Chapter 1 -except for Seurat[131].

To my knowledge, only [14] (for microarray data), [174] (for bulk RNAseq data) and [131] (for scRNAseq data) have tried to tackle this issue.

However, [14] models the gene dependency by **optimizing a cost function respect to** a certain set of intercorrelated feature classes denoted G (that is, finding a set G that minimizes a certain function), which is then used to generate posterior probabilities on the gene expression level of a certain gene (associated with random variable X) $Prob(X|\text{class}= k)$, which are computed differently depending on whether k belongs to G, and the local structure associated with this class. This model seems hard to adapt to the specificities of scRNAseq data (see previous sections), and computationally expensive, since

G, the local structures for each feature class in G, and the parameters of the posterior probability distributions need to be learned beforehand.

[131], although it is targeted at scRNAseq data, does not take into account the dropout phenomena, and needs to select a small subset of "landmark" (informative) genes (which are actually high-variance genes), and to compute "a spatial reference map of gene expression" for these genes. Moreover, the role of the "resolution" parameter is not intuitive, thus choosing a value for this parameter is done by trial and error. Sometimes, choosing the value in the interval [1,2], as suggested by the authors, does not give good results.

In [174], based on the work of [70], the use of copulas, despite of not being directly adaptable to scRNAseq, is a more elegant solution, that allows to select separately the feature dependence model and each feature probability distribution. The flexibility of this sort of model offers the opportunity to better control and list the underlying simplifying hypotheses.

What I want to find a model of gene expression associated to each condition of the gene expression matrix, that takes into account gene correlation. The set of expression values of all genes in a given cell can be aggregated into a real-valued vector. I will thus try to find a suitable copula model, associated with each condition, that is as close as possible to the multivariate cumulative distribution function giving the probability of the gene expression vector associated to this cell to be smaller (coordinate-wise) to some constant vector. The main point that justifies this approach is that little is known about the whole regulatory system between genes in one cell, but approximations of expression for one given gene have already been studied. Hence distribution models for each variable of the gene expression vector associated with one cell are known.

3.1 Mathematical background

Copulas are a efficient and simple tool to model correlation between random variables.

Notations

Let $I = \{1, 2, ..., p\}, p \in \mathbb{N}^*$, and let $(X_i)_{i \in I}$ be the set of real-valued random variables which joint probability distribution $\mathbb{P}$ is to find. For each $i \in I$, let f_{X_i} denote the **cumulative distribution function** (CDF) associated with the random variable X_i, i.e.:

Definition 3.1.1 (Cumulative distribution function)**.** *The CDF f_{X_i} associated with random variable X_i is defined such as:*

$$\forall x \in \mathbb{R}, \forall i \in I, f_{X_i}(x) = Prob(X_i \leq x)$$

Let eventually $f_{X_1,X_2,...,X_p}$ be the CDF of the random vector $(X_1, X_2, ..., X_p)$. $f_{X_1,X_2,...,X_p}$ is called the **joint CDF** of the variables $(X_i)_{i\in I}$, while each f_{X_i}, for $i \in I$, is the **marginal CDF** (or **margin**) associated with variable X_i.

Warning 3.1.1. *As a general rule:*
$\forall x = (x_1, x_2, ..., x_p) \in \mathbb{R}^p, f_{X_1,X_2,...,X_p}(x) \neq f_{X_1}(x_1) \times f_{X_2}(x_2) \times ... \times f_{X_p}(x_p)$

Let |S| denote the cardinal of the set S. Let us recall that **mutual independence** in statistics, for a set of variables $(C_i)_{i\in I}$, means that the following equalities hold:

Definition 3.1.2 (Mutual independence)**.**

$$\forall J \subseteq I, \forall c = (c_j)_{|J|} \in \mathbb{R}^{\#J}, f_{(C_j)_{j\in J}}(c) = \prod_{j\in J} f_{C_j}(c_j)$$

Mutual independence is distinct from regular (pairwise) independence, which means:

Definition 3.1.3 (Pairwise independence)**.**

$$\forall i, j \in I, \forall c_1, c_2 \in \mathbb{R}, f_{C_i,C_j}(c) = f_{C_i}(c_1) \times f_{C_j}(c_2)$$

If a random variable R follows the (continuous) standard uniform distribution $\mathbb{U}$, it means that:

Definition 3.1.4 (Standard uniform distribution)**.**

$$R \sim \mathbb{U}[0, 1] \text{ } \textit{iff.} \text{ } \forall r \in [0,1], f_R(r) = \mathbb{U}(R \leq r) = r$$

Copulas

Generally speaking, a **copula**[139] is a multivariate joint CDF whose margins are all continuous standard uniform distributions (an interesting explanation of copulas can be found in [5]). The interest in copulas is motivated by Sklar's theorem[139] (see [28] for a proof):

Theorem 3.1.1 (Sklar's copula theorem for real-valued random variables)**.** *Let F be a multivariate cumulative distribution function (CDF), $p \in \mathbb{N}^*$ and $(X_i)_{i\in\{1,2,...,p\}}$ a set of random variables respectively following univariate distributions having CDF $(F_i)_{i\in\{1,2,...,p\}}$.*

Then (if the $(F_i)_i$ are continuous) there exists (an unique) copula C such as: $\forall x_1, x_2, ..., x_p \in \mathbb{R}$, $F(x_1, x_2, ..., x_p) = C(F_1(x_1), F_2(x_2), ..., F_p(x_p))$.

Conversely, given a copula C, $H : (\mathbb{R}^+)^p \to [0, 1]$, $x \to C(F_1(x_1), ..., F_p(x_p))$ is a CDF.

This theorem means that for any (unknown) multivariate CDF, one can generate values of this function using the marginal distributions of the random variables, provided a/the suitable copula has been selected, which is often easier than searching for the real multivariate function. The converse part means that any copula used with some univariate CDFs is a multivariate CDF, which allows to select a copula model and try to get a satisfying approximation of the target CDF.

I will use two lemmas:

Lemma 3.1.1 (Mixture of copulas)**.** *A mixture of copulas is a copula, that is, if* $C_1, C_2, ..., C_n$*, where* $n \in \mathbb{N}^*$*, are copulas, and* $\alpha_1, \alpha_2, ..., \alpha_n$ *are positive real-valued numbers such as* $\sum_{i \leq n} \alpha_i = 1$*, then* $C : [0,1]^n \to [0,1]$*,* $x \to \sum_{i \leq n} \alpha_i \times C_i(x_i)$ *is a copula.*

Since a copula is nothing but a multivariate CDF, the proof is easily deduced from elementary proofs in probability. The following lemma is also well-known:

Lemma 3.1.2 (Probability integral transform)**.** *Let* R *be a random variable with a continuous distribution, and* f_R *be its continuous CDF. Then* $U = f_R(R)$ *follows a standard uniform distribution.*

Since copulas are essentially used to model dependence, they are usually associated with a set of random variables. Let $p \in \mathbb{N}^*$ and $I = \{1, 2, ..., p\}$. Given a set of random variables $(X_i)_{i \in I}$, the corresponding copula is a multivariate function C such as:

$$\mathrm{C} : \mathbb{R}^p \to [0,1], x \to C(f_{X_1}(x_1), f_{X_2}(x_2), ..., f_{X_p}(x_p))$$

This is well-defined thanks to 3.1.2. [34] explains the importance of having standard uniform marginal distributions. The variables inside the copula are called **latent variables** associated with $(X_i)_{i \in I}$. Some plots about copulas and the relationship between the observed variables and the corresponding latent variables are be found in Appendix.

Hence the main purpose here is to make an assumption on the type of dependency between the variables, that is, the copula distribution (Gaussian, etc.) and then try to fit the chosen distribution to the data[17]. Copulas allows the separate modeling of the joint correlation and of the distributions of each vector component -however, modeling separately the distributions does not mean that their respective parameters should be chosen separately[35].

Gaussian copulas

Gaussian copulas are one of the simplest copulas. Let C_ρ denote a Gaussian copula of parameter ρ, $\phi_{p(\rho)}$ being the multivariate normal distribution of zero-mean and standard deviation ρ, and ϕ being the standard normal distribution (of zero-mean and standard deviation 1). Then:

Definition 3.1.5 (Gaussian copula)**.** *Let* $(U_i)_{i\in I} = (f_{X_i}(X_i))_{i\in I}$ *be a set of random variables following a standard uniform distribution, and* C *be the copula associated with the variables* $(X_i)_{i\in I}$*. Let* $u \in [0,1]^p$*. Then:*
$C_\rho(u_1, u_2, ..., u_p) = \phi_{p(\rho)}(\phi^{-1}(U_1) = \phi^{-1}(u_1), \phi^{-1}(U_2) = \phi^{-1}(u_2), ..., \phi^{-1}(U_p) = \phi^{-1}(u_p))$

Thus I perform another variable transformation, as done in [174]: I denote from now the latent variables $(Z_i)_{i\in I}$, such as $\forall i \in I, Z_i = \phi(f_{X_i}(X_i))$.

Estimation of the parameters

I would like to compute the values of the parameters for the margin and copula functions -for instance, the mean vector and the covariance matrix for Gaussian copulas. In order to find their values, the (log) **likelihood function** of the copula can be maximized, respect to the set of all parameters, but for Gaussian copulas, it has no closed form. There are many strategies to avoid this issue; for instance, a heuristic estimation method, called Inference Functions for Margins (IFM), suggested by [83], is to find the best estimators of the margin parameters, by maximizing the likelihood function of each of the margins, and then use them in the copula likelihood function to find the copula parameters with maximum likelihood estimation. This method is computationally less costly, and is considered a good heuristic method[24], although it does not give the exact solution. Other methods also exist, see [170].

Mainly in order to have a reasonable runtime, and avoid the problem of initialization for optimization methods, I decided to estimate the margin parameters and the copula parameters with provided non-optimization methods[173][133] (see Appendix for further information). These estimates are also proven to be more robust, and avoid the problem to deal with singularities when fitting the covariance matrix. This issue cannot actually be ignored, because gene expression matrices have usually a lot of null values, due to dropout events, for instance.

3.2 Model

Given a cell j, it models the probability distribution of expression level for a set of genes; let us number the set of considered genes. Let $X_j \in (\mathbb{R}^+)^p = (X_{1,j}, X_{2,j}, ..., X_{p,j})$ be the random vector associated to the cell, where $X_{i,j}$ is the random variable associated to the expression level of gene i in cell/condition j, and p the cardinal of the set of genes. The probability distribution $\mathbb{P}_j$ of expression level in this cell is:

$$\begin{aligned} \forall x \in (\mathbb{R}^+)^p, \mathbb{P}_j(X_j = x) &= Prob(X_j = x) \\ &= Prob(X_{1,j} = x_1, X_{2,j} = x_2, ..., X_{p,j} = x_p) \end{aligned} \tag{3.1}$$

Main idea

Considering their expression levels across all cells, genes can be divided in two groups: whether they have been subject to a dropout phenomenon. Considering their *relative* expression values in a given cell, they can be divided into mildly-expressed and highly-expressed genes. I call *cell pattern* the binary vector, having as many components as features in the dataset, where a case is set to 1 if the associated feature is considered highly-expressed in this cell, 0 otherwise.

The main assumption is, as written in one of the first sections, cell functions depends on the gene expression profile of the cell; thus I introduce the concept of cell pattern, which is close -in spirit- to the gene pattern implementation in [131]. The gene expression probability distributions, for each cell, will depend on the cell patterns found, which is inspired by the probability distributions conditioned on gene classes like in [14].

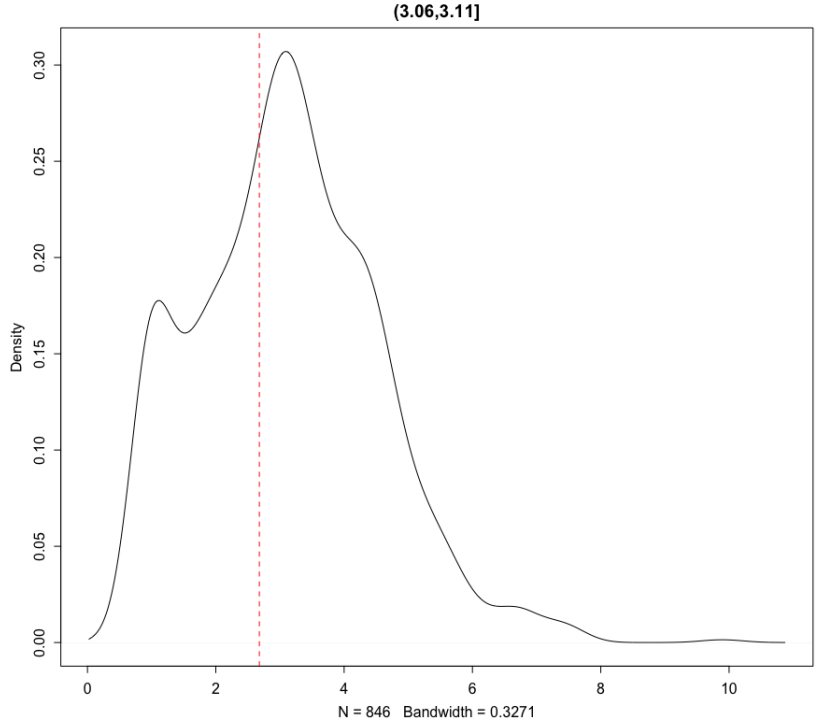


Figure 3.1: Result of kernel density estimation[119] (on n=846 points) applied on log-normalized gene expression values (that belong to the interval (3.06,3.11]) from the Ciona[146] scRNAseq dataset. Kernel estimation is a method used to estimate the density function of a random variable (here, the expression value of a gene in a given cell) from a sample. The red vertical line is the threshold found by the MAST package (see next section).

The joint probability distribution of the expression levels for all genes in a given cell will be modeled with a copula specific to this cell, as done in [174]. In order to tackle the problems specifically raised by scRNAseq data, I will use mixture probability distributions inspired by the works of [86], and [48] to model the marginal probability distributions for each gene in a given cell, depending on whether this gene is considered as dropout.

Hypotheses

1. **Mutual independence of gene expression values between different cells**:

$$\forall x, x' \in (\mathbb{R}^+)^p, \forall k, l,$$
$$Prob(\mathrm{X}_k = x, X_l = x') = \mathbb{P}_k(X_k = x) \times \mathbb{P}_l(X_l = x')$$

Note that this does not actually stand in practice (see plots in Appendix). But since the correlation between cell types is actually the point of gene expression clustering, there is no convincing prior hypothesis on cell correlation that does not imply using the cell labels.

2. **Non-independence of gene expression values between genes in a same cell**.

3. **Idea of a relationship between "cell pattern" and cell functions**: given a certain "activation threshold" and a certain cell, the set of genes that are expressed above this level (*highly-expressed genes*, denoted by a 1 in the cell pattern) and genes that are expressed below this level (*mildly-expressed genes*, denoted by a 0) is supposed to characterize the functional family of this cell.

4. All technical replicates (that is, cells from the same condition/embryo/stage/...) are supposed to follow the same gene expression value probability distribution.

5. Cells with a common functional role are supposed to have similar gene expression patterns (due to gene co-regulation[14]).

6. **Hypotheses on the marginal and joint probability distribution models** (see following section "The distribution model").

7. **Mutual independence of the dropout events (resp. mild or high gene expressiveness events)**: this hypothesis is -implicitly- made about the dropout events in [86], when using binary logistic regression to determine the dropout probability for a given gene (see discussion for further insight on this hypothesis).

Notations

- Let $|S|$ be the *cardinal*, that is, the number of elements in the set S.
- Let M be the input gene expression matrix of dimensions $p \times m$ (with $p >> m$), with features as rows, and conditions as columns. The set of rows for column j is $(M_{i,j})_i = \{M_{1,j}, M_{2,j}, M_{3,j}, ..., M_{p,j}\}$. The set of columns for row i is $(M_{i,j})_j = \{M_{i,1}, M_{i,2}, M_{i,3}, ..., M_{i,m}\}$.
- As a general rule, the letter i refers to a gene; the letters j and k refer to a condition or cell.

- Let $(X_{i,j})_{i,j} = \{X_{1,1}, X_{2,1}, ..., X_{p,1}, X_{1,2}, X_{2,2}, ..., X_{p,2}, ..., X_{1,m}, X_{2,m}, ..., X_{p,m}\}$ be the set of nonnegative real-valued random variables associated to the counts $(M_{i,j})_{i,j} = \{M_{1,1}, M_{2,1}, ..., M_{p,1}, M_{1,2}, M_{2,2}, ..., M_{p,2}, ..., M_{1,m}, M_{2,m}, ..., M_{p,m}\}$.
- Let $\phi(\mu, \rho)$ be the multivariate cumulative function of a normal distribution with parameters (mean = μ, variance = ρ), and ϕ the univariate standard normal distribution (of mean 0 and variance 1).
- Let P_k be the gene expression pattern for cell k, and P be the matrix of all the gene expression patterns.
- For a certain cell k, let Y_k be such as:

$$\forall is.o.(P_k)_i = 1, (Y_k)_i = Y_i k = -(X_k)_i$$
$$\forall is.o.(P_k)_i = 0, (Y_k)_i = Y_i k = (X_k)_i$$

- Let $f_{i,k} = f_{Y_{i,k}}$, and $f_k = f_{Y_{1,k},...,Y_{p,k}}$.
- Let $Z_{i,k}$ the so-called latent random variable associated with $X_{i,k}$: $Z_{i,k} = \phi(f_{i,k}(Y_{i,k}))$, and Z_k the latent random variable associated with Y_k: $Z_k = (Z_{1,k}, ..., Z_{p,k})$.
- Let $T_{i,j}$ denote the *activation threshold* for cell j and gene i, that is, one of the lowest points of the valley in Figure **??**, computed on normalized gene expression values, and let Q, called *pattern-oriented threshold matrix* be defined such as $Q_{i,k} = T_{i,k}$ if $P_{i,k} = 0$, and $Q_{i,k} = -T_{i,k}$ otherwise.

The distribution model

The flexibility of the copulas allows the separate selection of the joint and the marginal distribution models. In my model, each cell is modeled by a copula, which is the joint probability distribution for the gene expression for all features in this cell. That is, the margins for each copula are associated to the expression value of each gene in the dataset. So here, each copula, associated with a given cell j, is the cumulative joint probability distribution function of gene expression value for each gene in cell j.

The marginal probability distribution (for each gene, in a given cell)

Gene expression is subject to dropout phenomena[86], and is intrinsically bimodal[48][19] -and sometimes even multimodal[117].

According to [86], the most probable model for gene expression *count* values (i.e. discrete values) $X_{i,j}$, for a given gene in a given cell, would be a mixture of a low-magnitude Poisson distribution, to model the gene expression values subject to dropout, and negative-binomial distribution, to take into account the overdispersion of gene expression values for a successful amplification, using as mixing rate the dropout probability for this gene in this cell:

Definition 3.2.1 (Discrete gene expression value model)**.** *Let $p_{i,j}$ be the dropout probability for gene i in cell j. Then:*

$$\forall x \in \mathbb{N}, Prob(X_{i,j} = x) = p_{i,j} \times Poisson(\lambda_{i,j}) + (1 - p_{i,j}) \times NegBin(r_{i,j}, p_{i,j})$$

where the density functions for Poisson and negative-binomial distributions are respectively:

Definition 3.2.2 (Poisson distribution)**.**

$$\forall X \sim Poisson(\lambda), \forall k \in \mathbb{N},\ Prob(X = k) = \frac{\lambda^k \times e^{-\lambda}}{k!}$$

Definition 3.2.3 (Negative binomial distribution)**.**

$$\forall X \sim NegBin(r, p), \forall k \in \mathbb{N},\ Prob(X = k) = \binom{k+r-1}{k} \times (1-p)^r \times p^k$$

When a gene is subject to a dropout phenomenon, only low background noise is detected. As done in [86], instead of using constant zero function, I take into account random background noise by setting the Poisson parameter to a low constant:

$$\forall i, j \in \{1, 2, ...p\} \times \{1, 2, ..., m\}, \lambda_{i,j} = \lambda_0 < 1$$

Instead of using r and p parametrization for the Negative Binomial distribution, I use the mean-overdispersion (μ, δ) parametrization for negative-binomial[127], which is easier to interpret (see Appendix for the equivalence between the two parametrizations).

However, the marginal distributions need to be continuous in order to use a copula -and to estimate its parameters more easily. It is not continuous now since the Poisson and Negative binomial distributions are discrete. Furthermore, I need to model also the gene expression values when the values are normalized, so not necessarily integers anymore. And the Poisson distribution (thus the Negative Binomial one, since the latter is a mixture of Poisson and Gamma distributions) is sensible to scale modifications.

Thus I use a standard Gamma distribution, that is, with shape parameter λ, and rate parameter 1, to model the Poisson distribution, and a regular Gamma distribution, with parameters α (shape) and β (rate) for the Negative Binomial distribution (Gamma distribution has been actually widely studied as an approximation for this distribution[63][20]), where the density function for a Gamma distribution is:

Definition 3.2.4 (Gamma distribution)**.**

$$\forall X \sim \Gamma(\alpha, \beta), \forall x \in \mathbb{R}^+,\ Prob(X = x) = \frac{\beta^\alpha \times x^{\alpha-1} \times e^{-\beta \times x}}{\Gamma(\alpha)}$$

Again, the selection of the mean and the (over)dispersion factor (μ, δ) boils down to select parameters (α, β) (see Appendix). Henceforth the marginal distribution for gene expression and variable $X_{i,j}$ is:

Definition 3.2.5 (Continuous gene expression value model)**.** *Let $p_{i,j}$ be the dropout probability for gene i in cell j. Then:*

$$\forall x \in \mathbb{N}, Prob(X_{i,j} = x) = p_{i,j} \times Gamma(\lambda_0, 1) + (1 - p_{i,j}) \times Gamma(\mu_{i,j}, \delta_{i,j})$$

I will call this probability distribution CPNB$(\lambda_0, p_{i,j}, \mu_{i,j}, \delta_{i,j})$ (continuous -approximation of- Poisson-Negative Binomial mixture).

The copula probability distribution (for each cell)

According to the general assumption about the relationship between cell functions and gene expression profile, and the plots in Appendix, I can choose as starting distribution a bimodal Gaussian copula, that is, a mixture of two Gaussian copulas (such a mixture is a copula, see Lemma 3.1.1). The one with the highest mean is most likely associated with the most highly-expressed genes in the cell, while the second one seems to match the less expressed genes, which agree with my idea of cell gene expression pattern. I use the same assumption as [174] (which will be discussed later), that is, the covariance matrices represent the regulatory interactions between the genes, thus I assume that both copulas in a mixture have got the same covariance matrix, and that this matrix is independent of the considered cell.

I denote such a bimodal Gaussian copula of means μ_1, μ_2, with mixing parameter α and covariance matrix Σ, $C_{\mu_1,\mu_2,\alpha,\Sigma}$.

Please keep in mind that the variables associated with the copulas are $(Y_{i,j})_{i \leq p, j \leq m}$ (see notations), because what I want to compute is the probability of each gene to have an expression value above or below their associated activation threshold (see the following sections). Since for all $i, j \in \{1, 2, ..., p\} \times \{1, 2, ..., m\}$, $Y_{i,j} = X_{i,j}$ if $P_{i,j} = 0$, and $Y_{i,j} = -X_{i,j}$ otherwise, the associated marginal cumulative distributions $f_{Y_{i,j}}$ are easy to deduce from the previous section (see Appendix). The latent variables associated are thus, for all $k \leq m$, $(Z_{i,k})_{i \in I} = (\phi(f_{Y_{i,k}}(Y_{i,k})))_{i \in I}$.

The final model

For each cell, the final copula distribution model is:

Definition 3.2.6 (Final gene expression distribution model)**.** *Let j be the index of a given cell in the dataset. If q is a vector of gene expression in this cell ($q \in \mathbb{R}^{+p}$):*

$$Prob(Y_j \leq q) = \\ C_{j\mu_{1j},\mu_{2j},\alpha_j,\Sigma}(Z_{1,j} \leq \phi(f_{Y_{1,j}}(q_1)), Z_{2,j} \leq \phi(f_{Y_{2,j}}(q_2)), ..., Z_{p,j} \leq \phi(f_{Y_{p,j}}(q_p)))$$

To know more about the parameter value estimation, see Appendix.

Cell comparison

Now that I have defined a gene expression value probabilistic model, that focuses on the *relative* gene expression levels, to tackle the issue highlighted in the introduction, I would like to be able to compare gene expression profiles between cells, to perform cluster analysis. Intuitively, I consider two cells to be similar if they have the same cell pattern, as computed in matrix P, and the more the probability -implicitly conditioned on the parameters of the distribution model- of both gene expression profiles having this pattern is high, the more they are similar (see last part of this section for a discussion of this hypothesis). This intuition arises from the fact that cells are usually considered similar if they exhibit gene expression "trend similarity"[78], that is, similar gene expression profiles (i.e. the proportions of gene expression are roughly the same). That would mean that they have similar functions.

Definition 3.2.7 (Gene expression profile and cell pattern)**.** *A gene expression profile G -a p-sized nonnegative real-valued vector- suits a given pattern P -p-sized binary vector- respect to threshold vector T -a p-sized nonnegative real-valued vector- if and only if:*

$$\forall i \in \{1, 2, ..., p\},\ \textit{if } P_i = 1,\ \textit{then } G_i > T_i,\ \textit{and otherwise } G_i \leq T_i$$

Definition 3.2.8 (Cell similarity)**.** *Let k, l be two cells. k and l are similar iff:*

1. $k = l$
2. *The probability of the event "*$\forall i \in \{1, 2, ..., p\}, P_{i,k} = P_{i.l}$ *and*
   - *either* $P_{i,k} = 1$ *and* $X_{i,k} > T_i, X_{i,l} > T_i$,
   - *or either* $P_{i,k} = 0$ *and* $X_{i,k} \leq T_i, X_{i,l} \leq T_i$ *" is "high enough".*

Let us try to figure out how to compute similarity value in the second case. Let $I = \{1, 2, ..., p\}$, $U_0 =$ "$\forall i \in \{i \in I | P_{i,k} = 0\}, X_{i,k} \leq T_i, X_{i,l} \leq T_i$", and $U_1 =$ "$\forall i \in \{i \in I | P_{i,k} = 1\}, X_{i,k} > T_i, X_{i,l} > T_i$". These events can be rewritten with random variables $(Y_{i,k})_{i,k}$ and real numbers $(Q_{i,k})_{i,k}$ (see previous section about the notations). I use the pairwise coordinate order on vectors:

$$\begin{aligned} Prob(\text{U}_1 \text{ AND } U_0) &= Prob(\forall i \in I, Y_{i,k} \leq Q_{i,k} \text{ AND } Y_{i,l} \leq Q_{i,l}) \\ &= Prob(Y_k \leq Q_k \text{ AND } Y_l \leq Q_l) \\ &= Prob(Y_k \leq Q_k) \times Prob(\text{Y}_l \leq Q_l) \end{aligned} \tag{3.2}$$

using assumed cell mutual independence.

Definition 3.2.9 (Cell similarity measure)**.** *Let k, l be two cells. Then their similarity value $sim(k, l)$ (0 being complete dissimilarity, and 1 being identity) is defined as such:*
- if $k = l$, then $sim(k, l) = 1$
- else if $P_k \neq P_l$ (i.e. $\exists i \in \{1, 2, ..., p\}, P_{i,k} \neq P_{i,l}$), then $sim(k, l) = 0$
- else $sim(k, l) = Prob(Y_k \leq Q_k) \times Prob(Y_l \leq Q_l)$

3.3 Clustering method

The algorithm I suggest is a model-based hierarchical clustering algorithm[50], that is, it tries to infer the similarities between elements by modeling the distribution generating each group. Copulas have actually already been used for clustering/classification[144][94], and recently were applied to RNA-seq data classification[174].

Mine is divided in five parts, which details can be found in Appendix. I will use the same notations defined in the last section.

Input

This algorithm takes as input:

- The (un)normalized gene expression matrix.
- A cell-wise similarity threshold d: it is meant to be the threshold for the probability of two cells to be similar, in the sense I described above. The default value is 0.5.

There are also other values used for clustering and feature selection, that can be modified:

- $d_{neighbors}$, which is a threshold for merging cell pattern clusters (default: d): the cell-neighbor similarity threshold required for agreement of the neighbor (see Appendix). This value can modified to allow double thresholding, according to the importance of having homogenous clusters.
- d', which is the threshold for gene correlation: two genes are considered truly correlated if their correlation number is above d', and their mean gene expression value is roughly similar (see Appendix). Default value is 0.9, but 0.75 is the minimum generally allowed[27].
- f, such as if a gene appears in more than $f \times 100$ % of the conditions, then it is considered ubiquitous and removed. Default value is 0.7, which is a rough guess based on experimental data.

Outlines

Algorithm 1 Clustering technique: **Mine** algorithm

Input: M, gene expression matrix of size $p \times m$
Parameters:
- d, probability threshold for merging cell patterns (default: 0.5)
Hidden parameters:
- $d_{neighbors}$, threshold for merging cell pattern clusters (default: d)
- d', threshold for gene correlation (default: 0.9)
- f (default: 0.7)
Output: clustering of the conditions (columns) in matrix M

——————**Pre-processing**——————
$M \leftarrow$ filter(M)
$M \leftarrow$ featureSelect(M)
$M \leftarrow$ frequenceTrim(M)
——————**Finding pattern**——————
$res \leftarrow$ thresholdComputation(M)
$modelDropout, modelMild, P, Q \leftarrow$ getPatterns(M, res)
——————**Distribution computation**——————
$margin_means, margin_deltas, margin_rates \leftarrow$estimateMarginParameters($M, P, modelDropout$)
$copulas \leftarrow$ **NULL**
for $j \in \{1, 2, 3, ..., m\}$ **do**
  $copulas_j \leftarrow$createBimodalCopula($M, modelMild$)
end for
——————**Merging pattern**——————
$distMatrix \leftarrow$ null matrix of order m
\# **The elements in the max heap are sorted by similarity value** \#
$mergeablePairs \leftarrow$ initializeHeap()
for $k \in \{1, 2, 3, ..., m-1\}$ **do**
  for $l \in \{k+1, k+2, k+3, ..., m\}$ **do**
    $sim \leftarrow$computeSimilarity($k, l, copulas\_k, copulas\_l, P, Q$)
    $distMatrix[k][l] \leftarrow$ 1-sim
    $distMatrix[l][k] \leftarrow$ 1-sim
    if $sim > d$ **then**
      pushHeap(k, l)
    end if
  end for
end for
——————**Clustering**——————
$clusters \leftarrow$ initializeClusters()
while no cluster convergence **do**
  $queue \leftarrow$ mergeablePairs
  while *queue* is not empty **do**
    $k, l \leftarrow$ extractMinHeap($queue$)
    if k and l are in distinct clusters **then**
      if there is agreement between the neighbors of k and l **then**
        merge the clusters of l and k
      end if
    end if
  end while
end while
return *clusters*

Discussion of the model

- Instead of the regular mixture model I have chosen, I could have used a zero-inflated model (ZIM)[127]. A ZIM is, given a density distribution function f, a mixture rate $0 < \alpha < 1$, and a nonnegative real-valued random variable R, the associated ZIM is defined such as:

  Definition 3.3.1 (General zero-inflated model)**.**

  $$Prob(R = 0) = \alpha + (1 - \alpha) \times f(0)$$

  $$\forall r \in \mathbb{R}^+, r > 0 \; Prob(R = r) = (1 - \alpha) \times f(r)$$

  Note that it is in fact a mixture model with a given probability distribution (Negative Binomial alike for instance) and the null distribution. But, as written previously, contrary to the null distribution, the Poisson distribution allows to account for low background noise[86], if its parameter value is low enough.

  Another issue of my model is that I have to compute the best value for the mixture rate parameter. The hurdle (most of the time set at zero) model is a way to overcome this problem; given two density distribution functions f_1 and f_2 to mix, and a nonnegative real-valued random variable R, the associated hurdle model is[11]:

  Definition 3.3.2 (General hurdle model)**.**

  $$Prob(R = 0) = f_1(0)$$

  $$\forall r \in \mathbb{R}^+, r > 0 \; Prob(R = r) = \frac{1-f_1(0)}{1-f_2(0)} \times f_2(r)$$

  For instance, f_1 would be the Poisson density function, and f_2 be the Negative Binomial density function. But note that this model assumes that the dropout part (that is, the Poisson component of the mixture) is the only one source of null gene expression values. But I consider that transcriptionally silent genes, that is, whose sequence does not appear in the mRNA molecule, also give quasi-zero values[86].

- I could have used the two continuous counterparts of Poisson and Negative Binomial built by [74][29][3], but they are computationally expensive, and since I just want to fit the data, I do not need the asymptotic limits of the probability distributions, but only the assumptions made on variance and mean (and not median, for instance). Estimation of the parameters

for copulas is easier when the marginals are continuous, especially for the computation of the gradient needed for optimization procedures.

Gamma distribution is also quite close to Poisson and Negative Binomial distributions, and has already been used as an approximation for both of them, see [20][63]. I cannot assume that the gene expression values of the two gene groups (dropout and non-dropout) are independent, thus the mixture of the two Gamma distributions is not necessarily a Gamma distribution.

- The Gaussian copula for one-cell gene dependence is probably a really rough assumption. But it is the most recommended for general models[76], and the real gene correlation model is not known. Moreover, since the number of dimensions is huge, the Gaussian copula is the easiest model to fit to the data, and there are efficient and accurate estimators of its parameters.

- The cell comparison measure suggested is not a distance -the given definition does not satisfy transitivity condition- and it may look a bit clumsy since it does not comply with the reflexivity property *per se*. The main purpose of this definition is to be intuitively close to what I would expect in a cell clustering.

  The problem lay upon the computation of the probability of having such a gene pattern for each of the cells. The model actually assumes that the true distribution of the gene expression is a mixture of Gamma distributions, and if the empirical distribution from the data does not abide by this assumption, the gene pattern will be considered unlikely to appear, thus a cell can be considered not similar to itself, and no clustering will actually make sense. This discrepancy may either come from bad quality data -but, with new technologies developing, it will be hopefully unlikely to happen- or either from the fact that the commonly used model for real gene expression is in fact not suitable.

- Another issue is that the parameters are fitted on the very same data on which the clustering part will be applied. Ideally, the data would have lots of replicates of each condition/cell, and cross validation procedures could be performed. It though requires data annotation work. I tried to solve this issue by generating data using kernel density estimation, or by applying a Gaussian random noise to the input data.

  A possible improvement would be the possibility to select a given number of cell groups, and to perform the clustering on these groups, e.g., cells from healthy people, from cancer-afflicted non-treated patients, and from treated patients. But the algorithm is meant to contribute to an

exploratory analysis, thus seeing similarities between cells without prior hypothesis nor knowledge.

- I assumed the covariance matrix was the same for every cell/copula, as done in [174]. This covariance matrix is meant to represent the gene regulatory system network, as highlighted by [133]. This system is not supposed to be different from one cell to another. However, it might be different between a group of highly-expressed genes and a group of mildly-expressed genes.

3.4 Implementation

The algorithm has been implemented in R, and the source code can be found here: https://github.com/oist-gene-clustering/mine

Packages: R[125], MAST[111], doParallel[6], mvtnorm[56], caret[52], Matrix[16], corpcor[132], fastcluster[115] and foreach[7].

Chapter 4

Benchmark on clustering algorithms

There exist currently lots of algorithms, each of them using various methods, developed to tackle cell clustering for single-cell RNA data, even through it is still quite a recent field. However, none of them has been selected yet as the reference clustering algorithm. I performed a benchmark, including the most known algorithms in R targeted at scRNAseq data, in order to compare the clustering results, and to check the correctness of the resulting functional cell families found.

The tested algorithms in the following benchmark are SC3, pcaReduce, tSNE+K-means, tSNE+DBSCAN, K-means, SNN-Cliq, SINCERA, SEURAT, NMF and Fuzzy C-means, and have been briefly introduced in the section **Cluster analysis**.

4.1 Pipeline

Method

This benchmark tests the ability of one algorithm to find consistent clusters of cells/conditions (**accuracy test**), respect to the biological data, the time needed to compute the answer in function of the dimensions of the gene expression matrix (**time complexity test**), and the stability of the resulting clustering (**stability test**). For more insight on the datasets, see Appendix.

For each dataset, I looked up in the corresponding article the expected number of clusters, and tried to find a "reference clustering", that is, the annotation of the subpopulations in the dataset. Sometimes the number of clusters

does not match the number of subpopulations, but I assumed that the annotation into subpopulations is still either a subclustering or a overclustering of the real reference clustering, and thus could be helpful to determine if the algorithm gave consistent results with the data. For more details about the data, please check the Excel file at the following link: https://github.com/oist-geneclustering/geneclusteringbenchmark

To be able to compare fairly the different algorithms, I tried to find the best parameters for each dataset and each algorithm (which can be found in the Excel file previously quoted), that is, which seemed to give the best results according to the PCA and tSNE plots, and giving a number of clusters as close as possible to the expected one. Of course, these plots can only be considered indications and have bias, but since I could not get access to the real reference clustering, I thought it would be wiser not to rely too much on the latter.

Eventually, I ran again several times all the algorithms on one dataset (the one having the highest mean accuracy measure), in order to observe the variation of accuracy between runs.

Accuracy test

I tested on each dataset each algorithm. For indeterministic algorithms, I used 100 runs and kept the clustering giving the highest Adjusted Rand Index[126] (ARI, computed with MClust[49]) with the reference clustering I fixed for each dataset. Except for Seurat, SINCERA and SNN-Cliq, all algorithms are indeterministic.

Definition 4.1.1 (Adjusted Rand Index (ARI))**.** *Let $X = \{X_1, X_2, ..., X_r\}$ and $Y = \{Y_1, Y_2, ..., Y_s\}$ be two clusterings of the same dataset $\{x_1, x_2, ..., x_m\}, m, r, s \in \mathbb{N}^*$. For all $k, l \in \{1, 2, ..., r\} \times \{1, 2, ..., s\}$, let $n_{k,l} = |X_k \cap Y_l|$ be the number of common points in clusters X_k and Y_l, and $a_k = \sum_{l \leq s} n_{k,l}$, and $b_l = \sum_{k \leq r} n_{k,l}$. Then:*

$$ARI(X, Y) = \frac{\sum_{i,j}\binom{n_{i,j}}{2} - (\sum_i \binom{a_i}{2} \times \sum_j \binom{b_j}{2})/\binom{m}{2}}{\frac{1}{2}(\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}) - (\sum_i \binom{a_i}{2} \times \sum_j \binom{b_j}{2})/\binom{m}{2}}$$

The ARI may take negative values, meaning that the resulting accuracy respect to the reference clustering is less than the expected index when the compared clustering has been generated randomly. The closer it gets to 1, the more the two clusterings are similar. This measure is commonly used for clustering comparison, as done in [87][78], for instance.

Time complexity

The benchmark takes into account the time (in seconds, using base R function *system.time*) needed to compute the solution with the highest ARI.

Stability test

Indeterministic algorithms have been iterated 100 times on the Biase dataset. Then I computed a stability measure, which is the frequency among the 100 runs of the most frequent solution. It may show the difference of performance between two iterations of the same algorithm on a given dataset.

4.2 Results

See Appendix for more details about the ranking.

Table 4.1: Ranking of algorithms (the "average" value is computed using the weighted mean described in Appendix)

Rank	**Accuracy (ARI)**	**Accuracy (absolute(no. clusters-expected no.))**	**Stability**	**Time**
#1	Fuzzy C-Means (avg. ARI: 0.77)	SC3 avg. abs. diff: 0)	SC3 (stability: 1.00)	+DBSCAN (avg. time: 4.97s)
#2	K-Means (avg. ARI: 0.74)	+K-means (avg. abs. diff: 0)	PCAReduce (stability: 0.29)	SINCERA (avg. time: 6.00s)
#3	SC3 (avg. ARI: 0.70)	SINCERA (avg. abs. diff: 0)	K-means (stability: 0.29)	SC3 (avg. time: 6.31s)
#4	PCAReduce (avg. ARI: 0.67)	Fuzzy C-means (avg. abs. diff: 0)	+DBSCAN (stability: 0.04)	+Kmeans (avg. time: 6.77s)
#5	+DBSCAN (avg. ARI: 0.52)	PCAReduce (avg. abs. diff: 1)	+K-Means (stability: 0.02)	K-means (avg. time: 6.77)
#6	SINCERA (avg. ARI: 0.52)	Seurat (avg. abs. diff: 1)	Fuzzy C-means (stability: 0.01)	Fuzzy K-means (avg. time: 10.55s)
#7	SEURAT (avg. ARI: 0.52)	+DBSCAN (avg. abs. diff: 2)		PCAReduce (avg. time: 12.99s)
#8	+Kmeans (avg. ARI: 0.46)	K-means (avg. abs. diff: 6)		Seurat (avg. time: 13.74s)
#9				
#10				

SNN-Cliq, SINCERA and Seurat algorithms are deterministic, thus computing the stability measure for these algorithms is not required.

4.3 Discussion on the benchmark

The files used for the benchmark can be found at: https://github.com/oist-gene-clustering/geneclusteringbenchmark

Packages: R[125], SC3[88], pcaReduce[176], pcaMethods[141], scater[109], mclust[51], Rtsne[95], SINCERA[64], rPython[18], Seurat[130], pheatmap[91], reshape2[162], e1071[113], dbscan[67], NMF[54], ggplot2[163] and RColorBrewer[116].

Generally speaking, the main issue with this benchmark is the fact I don't know the true reference clustering. But the ARI may still be relevant since the chosen clusterings are subclusterings or overclusterings. It is also important to take into account the resulting number of clusters compared to the real one. I also learnt that actually the tsne package has unsolved documented bugs (that are explained at [22]), and since I used tSNE visualization when looking for the best parameters for each algorithm, it may had a influence on the results. For algorithms using tSNE (such as tSNE+K-means and tSNE+DBSCAN), I used the Rtsne package, and no bug in this package has been reported yet.

It can also be seen that, although K-means and Fuzzy K-means give good results when iterated, those two algorithms are quite instable, thus they are not suitable for an exploratory analysis. It can also noticed that most of the algorithms need in input the expected number of clusters. This number can be guessed using gap statistics[69] or the silhouette measure[128]. For Seurat and SNN-Cliq which do not need this input, it is quite hard actually to find the best parameter value, and nothing but trial-and-error strategy and visualization (with PCA or t-SNE, for instance) can be helpful. Unfortunately, visualization tools can also be biased[161].

From Table 4.1, SC3 and SINCERA algorithms seem to be the best tradeoff between accuracy, little computation time and stability. Of course, this benchmark needs replication in order to check the results I have obtained.

Chapter 5

Conclusion

This internship focused on gene expression, from the sequencing pipeline of the mRNA molecule to the following downstream analysis, including clustering to detect cell functional families. It gave me an insight on this quite wide research field, and I have learnt more about the biological phenomena involved, the computational approaches developed to handle the huge amount of data obtained by sequencing, statistics, the copula theory, and also about reactive programming and parallelism, while designing the application. I could also apply what I learnt about optimization on real-life data, and I had to deal with the issue of feature selection, noise removing and data generation.

5.1 Overview of the internship

These are the outcome of this internship:

- A possible pipeline in Python to clean and obtain a gene expression matrix out of raw RNA and single-cell RNA sequencing data.
- An online, user-friendly, application in Shiny R for analyze single-cell RNA sequencing data, and perform differential expression analysis, that is crucial for exploratory analysis on the cells and genes involved. Contrary to the other existing applications, the latter allows also to visualize gene expression patterns.
- A benchmark performed on several clustering algorithms and single-cell RNA seq datasets. As far as I know, only the SC3 article did such a benchmark.
- A new clustering algorithm, focused on gene expression modelling with copulas, while taking into account the characteristics of single-cell seq data.

5.2 Outlook

What could be done:

- I could give a nicer user interface to the application, and gene expression patterns could be represented on a drawing of the true single cell.
- Ideally, I would test the clustering algorithms on the true clusterings. I could also add other sorts of algorithms, such as model-based clustering techniques for instance.
- The algorithm I designed should be compared to the other algorithms in the benchmark. It should also be made faster to be useful.

Bibliography

[1] about boxplot. http://onetipperday.blogspot.jp/2012/06/about-boxplot.html. Accessed: 2017-07-26.

[2] NCBI database. https://www.ncbi.nlm.nih.gov. Accessed: 2017-07-03.

[3] Salah H Abid and Sajad H Mohammed. On the continuous poisson distribution. *International Journal of Data Envelopment Analysis and* Operations Research**, 2(1):7–15, 2016.

[4] Khan Academy. https://www.khanacademy.org/science/biology/gene-expression-central-dogma/translation-polypeptides/a/translation-overview. Accessed: 2017-06-08.

[5] M. Alice. Modelling dependence with copulas. https://datascienceplus.com/modelling-dependence-with-copulas/. Accessed: 2017-05-29.

[6] Revolution Analytics and Steve Weston. *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*, 2015. R package version 1.0.10.

[7] Revolution Analytics and Steve Weston. *foreach: Provides Foreach Looping Construct for R*, 2015. R package version 1.4.3.

[8] Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Genome biology*, 11(10):R106, 2010.

[9] Shakesha Anderson. Logistic Regression. http://schatz.sju.edu/multivar/guide/logistic.pdf. Accessed: 2017-06-21.

[10] Simon Andrews et al. Fastqc: a quality control tool for high throughput sequence data, 2010.

[11] Barry C Arnold, N Balakrishnan, Jose-Maria Sarabia Alegria, and Roberto Minguez. *Advances in Mathematical and Statistical Modeling*. Springer Science & Business Media, 2009.

[12] Dean Attali. *shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds*, 2016. R package version 0.9.

[13] Eric Bailey. *shinyBS: Twitter Bootstrap Components for Shiny*, 2015. R package version 0.61.

[14] Yoseph Barash and Nir Friedman. Context-specific bayesian clustering for gene expression data. *Journal of Computational Biology*, 9(2):169–191, 2002.

[15] W Brad Barbazuk, Scott J Emrich, Hsin D Chen, Li Li, and Patrick S Schnable. Snp discovery via 454 transcriptome sequencing. *The plant journal*, 51(5):910–918, 2007.

[16] Douglas Bates and Martin Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2017. R package version 1.2-8.

[17] Rihab BEDOUI and Makrem BEN DBABIS. Copules et mesures du risque bidimensionnelles: application pratique aux hedge funds. *université Paris X*, 2008.

[18] Carlos J. Gil Bellosta. *rPython: Package Allowing R to Call Python*, 2015. R package version 0.0-6.

[19] Marina Bessarabova, Eugene Kirillov, Weiwei Shi, Andrej Bugrim, Yuri Nikolsky, and Tatiana Nikolskaya. Bimodal gene expression patterns in breast cancer. *BMC genomics*, 11(1):S8, 2010.

[20] DJ Best and PG Gipps. An improved gamma approximation to the negative binomial. *Technometrics*, 16(4):621–624, 1974.

[21] Fernando H Biase, Xiaoyi Cao, and Sheng Zhong. Cell fate inclination within 2-cell and 4-cell mouse embryos revealed by single-cell rna sequencing. *Genome research*, 24(11):1787–1796, 2014.

[22] Babraham Institute Bioinformatics Group. TSNE test. http://www.bioinformatics.babraham.ac.uk/tsne/. Accessed: 2017-07-26.

[23] Anthony M Bolger, Marc Lohse, and Bjoern Usadel. Trimmomatic: a flexible trimmer for illumina sequence data. *Bioinformatics*, page btu170, 2014.

[24] Eric Bouyé, Valdo Durrleman, Ashkan Nikeghbali, Gaël Riboulet, and Thierry Roncalli. Copulas for finance-a reading guide and some applications. 2000.

[25] Philip Brennecke, Simon Anders, Jong Kyoung Kim, Aleksandra A Kołodziejczyk, Xiuwei Zhang, Valentina Proserpio, Bianka Baying, Vladimir Benes, Sarah A Teichmann, John C Marioni, et al. Accounting for technical noise in single-cell rna-seq experiments. *Nature methods*, 10(11):1093–1095, 2013.

[26] Sydney Brenner, François Jacob, and Matthew Meselson. An unstable intermediate carrying information from genes to ribosomes for protein synthesis. *Nature*, 190(4776):576–581, 1961.

[27] J. Brownlee. Feature Selection with the Caret R package. http://machinelearningmastery.com/feature-selection-with-the-caret-r-package/. Accessed: 2017-07-26.

[28] H Carley and MD Taylor. A new proof of sklar's theorem. In *Distributions with given marginals and statistical modelling*, pages 29–34. Springer, 2002.

[29] Nimai Kumar Chandra and Dilip Roy. A continuous version of the negative binomial distribution. *Statistica*, 72(1):81, 2012.

[30] Tse-Wen Chang. Binding of cells to matrixes of distinct antibodies coated on solid surface. *Journal of immunological methods*, 65(1-2):217–223, 1983.

[31] Winston Chang. *shinythemes: Themes for Shiny*, 2016. R package version 1.1.1.

[32] Winston Chang and Barbara Borges Ribeiro. *shinydashboard: Create Dashboards with 'Shiny'*, 2017. R package version 0.6.0.

[33] Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson. *shiny: Web Application Framework for R*, 2017. R package version 1.0.3.

[34] A. Charpentier. Copula estimation and influence of margins. http://freakonometrics.hypotheses.org/2432. Accessed: 2017-06-09.

[35] A. Charpentier. Modeling the marginals and the dependence separately. http://freakonometrics.hypotheses.org/tag/copula. Accessed: 2017-05-29.

[36] Yilun Chen, Ami Wiesel, and Alfred O Hero. Shrinkage estimation of high dimensional covariance matrices. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 2937–2940. IEEE, 2009.

[37] Yunshun Chen, Aaron TL Lun, and Gordon K Smyth. Differential expression analysis of complex rna-seq experiments using edger. In *Statistical analysis of next generation sequencing data*, pages 51–74. Springer, 2014.

[38] AJ Consulting Company. http://www.ajconsultingcompany.com/transcription-to-translation.html. Accessed: 2017-06-08.

[39] Ana Conesa, Pedro Madrigal, Sonia Tarazona, David Gomez-Cabrero, Alejandra Cervera, Andrew McPherson, Michał Wojciech Szcześniak, Daniel J Gaffney, Laura L Elo, Xuegong Zhang, et al. A survey of best practices for rna-seq data analysis. *Genome biology*, 17(1):13, 2016.

[40] Alan R Dabney. Classification of microarrays to nearest centroids. *Bioinformatics*, 21(22):4148–4154, 2005.

[41] Qiaolin Deng, Daniel Ramsköld, Björn Reinius, and Rickard Sandberg. Single-cell rna-seq reveals dynamic, random monoallelic gene expression in mammalian cells. *Science*, 343(6167):193–196, 2014.

[42] Patrik D'haeseleer. How does gene expression clustering work? *Nature biotechnology*, 23(12):1499, 2005.

[43] Jin Hwan Do, D Choi, et al. Clustering approaches to identifying gene expression patterns from dna microarray data. *Molecules and cells*, 25(2):279, 2008.

[44] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[45] Michael B Eisen, Paul T Spellman, Patrick O Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[46] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[47] Maurizio Filippone, Francesco Masulli, and Stefano Rovetta. Unsupervised gene selection and clustering using simulated annealing. In *International Workshop on Fuzzy Logic and Applications*, pages 229–235. Springer, 2005.

[48] Greg Finak, Andrew McDavid, Masanao Yajima, Jingyuan Deng, Vivian Gersuk, Alex K Shalek, Chloe K Slichter, Hannah W Miller, M Juliana McElrath, Martin Prlic, et al. Mast: a flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell rna sequencing data. *Genome biology*, 16(1):278, 2015.

[49] Chris Fraley and Adrian E Raftery. Mclust: Software for model-based cluster analysis. *Journal of classification*, 16(2):297–306, 1999.

[50] Chris Fraley and Adrian E Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association*, 97(458):611–631, 2002.

[51] Chris Fraley, Adrian E. Raftery, T. Brendan Murphy, , and Luca Scrucca. mclust version 4 for r: Normal mixture modeling for model-based clustering, classification, and density estimation technical report no. 597. Technical report, University of Washington, 2012.

[52] Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan, and Tyler Hunt. *caret: Classification and Regression Training*, 2017. R package version 6.0-76.

[53] Jim Frost. Why You Need to Check Your Residual Plots for Regression Analysis: Or, To Err is Human, To Err Randomly is Statistically Divine. http://blog.minitab.com/blog/adventures-in-statistics-2/why-you-need-to-check-your-residual-plots-for-regression-analysis, 2012. Accessed: 2017-06-21.

[54] Renaud Gaujoux and Cathal Seoighe. A flexible r package for nonnegative matrix factorization. *BMC Bioinformatics*, 2010.

[55] R. Gentleman, V. Carey, W. Huber, and F. Hahne. *genefilter: genefilter: methods for filtering genes from high-throughput experiments*, 2016. R package version 1.56.0.

[56] A. Genz, F. Bretz, T. Miwa, X. Mi, F. Leisch, F. Scheipl, and T. Hothorn. *mvtnorm: Multivariate Normal and t Distributions*, 2017. R package version 1.0-6.

[57] Mubeen Goolam, Antonio Scialdone, Sarah JL Graham, Iain C Macaulay, Agnieszka Jedrusik, Anna Hupalowska, Thierry Voet, John C Marioni, and Magdalena Zernicka-Goetz. Heterogeneity in oct4 and sox2 targets biases cell fate in 4-cell mouse embryos. *Cell*, 165(1):61–74, 2016.

[58] Jan Graffelman. *calibrate: Calibration of Scatterplot and Biplot Axes*, 2013. R package version 1.7.2.

[59] Malachi Griffith, Jason R Walker, Nicholas C Spies, Benjamin J Ainscough, and Obi L Griffith. Informatics for rna sequencing: a web resource for analysis on the cloud. *PLoS computational biology*, 11(8):e1004393, 2015.

[60] Yevgeniy Grigoryev. Introduction to DNA Microarrays. http://bitesizebio.com/7206/introduction-to-dna-microarrays/. Accessed: 2017-06-09.

[61] Dominic Grün, Lennart Kester, and Alexander Van Oudenaarden. Validation of noise models for single-cell transcriptomics. *Nature methods*, 11(6):637–640, 2014.

[62] Dominic Grün, Anna Lyubimova, Lennart Kester, Kay Wiebrands, Onur Basak, Nobuo Sasaki, Hans Clevers, and Alexander van Oudenaarden. Single-cell messenger rna sequencing reveals rare intestinal cell types. *Nature*, 525(7568):251–255, 2015.

[63] William C Guenther. A simple approximation to the negative binomial (and regular binomial). *Technometrics*, 14(2):385–389, 1972.

[64] Minzhe Guo. *SINCERA: An R implementation of SINCERA pipeline for processing single-cell RNA-seq data.*, 2017. R package version 0.99.0.

[65] Minzhe Guo, Hui Wang, S Steven Potter, Jeffrey A Whitsett, and Yan Xu. Sincera: a pipeline for single-cell rna-seq profiling analysis. *PLoS Comput Biol*, 11(11):e1004575, 2015.

[66] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[67] Michael Hahsler and Matthew Piekenbrock. *dbscan: Density Based Clustering of Applications with Noise (DBSCAN) and Related Algorithms*, 2017. R package version 1.1-1.

[68] Julia Handl, Joshua Knowles, and Douglas B Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212, 2005.

[69] Trevor Hastie, Robert Tibshirani, and Guenther Walther. Estimating the number of data clusters via the gap statistic. *J Roy Stat Soc B*, 63:411–423, 2001.

[70] Esther Hee Lee. Copula analysis of correlated counts. In *Bayesian Model Comparison*, pages 325–348. Emerald Group Publishing Limited, 2014.

[71] Zena M Hira and Duncan F Gillies. A review of feature selection and feature extraction methods applied on microarray data. *Advances in bioinformatics*, 2015, 2015.

[72] Yu-Jui Ho Ho, Toby Aicher, and M. Hammell Lab. SAKE gene expression data analysis application. https://github.com/naikai/sake. Accessed: 2017-04-26.

[73] Aaron Hoffer. *shinyDND: Shiny Drag-n-Drop*, 2016. R package version 0.1.1.

[74] Andrii Ilienko. Continuous counterparts of poisson and binomial distributions and their properties. *arXiv preprint arXiv:1303.5990*, 2013.

[75] Illumina. RNA-Seq Microarray Technology Comparison. https://www.illumina.com/technology/next-generation-sequencing/mrna-seq.html. Accessed: 2017-06-13.

[76] David I Inouye, Eunho Yang, Genevera I Allen, and Pradeep Ravikumar. A review of multivariate distributions for count data derived from the poisson distribution. *Wiley Interdisciplinary Reviews: Computational Statistics*, 9(3), 2017.

[77] Lemon J. Plotrix: a package in the red light district of r. *R-News*, 6(4):8–12, 2006.

[78] Pablo A Jaskowiak, Ricardo JGB Campello, and Ivan G Costa. On the selection of appropriate distances for gene expression data clustering. *BMC bioinformatics*, 15(2):S2, 2014.

[79] Daxin Jiang, Chun Tang, and Aidong Zhang. Cluster analysis for gene expression data: A survey. *IEEE Transactions on knowledge and data engineering*, 16(11):1370–1386, 2004.

[80] Lan Jiang, Huidong Chen, Luca Pinello, and Guo-Cheng Yuan. Giniclust: detecting rare cell types from single-cell gene expression data with gini index. *Genome Biology*, 17(1):144, 2016.

[81] Peng Jiang, James A Thomson, and Ron Stewart. Quality control of single-cell rna-seq by sinqc. *Bioinformatics*, 32(16):2514–2516, 2016.

[82] Thanyaluk Jirapech-Umpai and Stuart Aitken. Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes. *BMC bioinformatics*, 6(1):148, 2005.

[83] Harry Joe and James Jianmeng Xu. The estimation method of inference functions for margins for multivariate models. 1996.

[84] Shintaro Katayama, Virpi Töhönen, Sten Linnarsson, and Juha Kere. Samstrt: statistical test for differential expression in single-cell transcriptome with spike-in normalization. *Bioinformatics*, 29(22):2943–2945, 2013.

[85] M Kathleen Kerr, Mitchell Martin, and Gary A Churchill. Analysis of variance for gene expression microarray data. *Journal of computational biology*, 7(6):819–837, 2000.

[86] Peter V Kharchenko, Lev Silberstein, and David T Scadden. Bayesian approach to single-cell differential expression analysis. *Nature methods*, 11(7):740–742, 2014.

[87] Vladimir Yu Kiselev, Kristina Kirschner, Michael T Schaub, Tallulah Andrews, Andrew Yiu, Tamir Chandra, Kedar N Natarajan, Wolf Reik, Mauricio Barahona, Anthony R Green, et al. Sc3-consensus clustering of single-cell rna-seq data. *bioRxiv*, page 036558, 2016.

[88] Vladimir Yu. Kiselev, Kristina Kirschner, Michael T. Schaub, Tallulah Andrews, Andrew Yiu, Tamir Chandra, Kedar N Natarajan, Wolf Reik, Mauricio Barahona, Anthony R Green, and Martin Hemberg. Sc3 - consensus clustering of single-cell rna-seq data. *Nature Methods*, 2017.

[89] Allon M Klein, Linas Mazutis, Ilke Akartuna, Naren Tallapragada, Adrian Veres, Victor Li, Leonid Peshkin, David A Weitz, and Marc W Kirschner. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, 161(5):1187–1201, 2015.

[90] Jon Kleinberg. An impossibility theorem for clustering. In *NIPS*, volume 15, pages 463–470, 2002.

[91] Raivo Kolde. *pheatmap: Pretty Heatmaps*, 2015. R package version 1.0.8.

[92] Aleksandra A Kolodziejczyk, Jong Kyoung Kim, Valentine Svensson, John C Marioni, and Sarah A Teichmann. The technology and biology of single-cell rna sequencing. *Molecular cell*, 58(4):610–620, 2015.

[93] Aleksandra A Kolodziejczyk, Jong Kyoung Kim, Jason CH Tsang, Tomislav Ilicic, Johan Henriksson, Kedar N Natarajan, Alex C Tuck, Xuefei Gao, Marc Bühler, Pentao Liu, et al. Single cell rna-sequencing of pluripotent states unlocks modular transcriptional variation. *Cell Stem Cell*, 17(4):471–485, 2015.

[94] Ioannis Kosmidis and Dimitris Karlis. Model-based clustering using copulas with applications. *arXiv preprint arXiv:1404.4077*, 2014.

[95] Jesse H. Krijthe. *Rtsne: T-Distributed Stochastic Neighbor Embedding using a Barnes-Hut Implementation*, 2015.

[96] Max Kuhn. Caret package. *Journal of Statistical Software*, 28(5):1–26, 2008.

[97] Hemberg lab. https://hemberg-lab.github.io/scRNA.seq.datasets/. Accessed: 2017-06-30.

[98] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357–359, 2012.

[99] Tomasz LATKOWSKI and Stanisław OSOWSKI. Feature selection methods in application to gene expression: autism data. *Przegląd Elektrotechniczny*, 90(8):199–204, 2014.

[100] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[101] Luis Leon-Novelo, Claudio Fuentes, and Sarah Emerson. Bayesian estimation of negative binomial parameters with applications to rna-seq data. *arXiv preprint arXiv:1512.00475*, 2015.

[102] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, et al. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[103] Bing Liu, Chunru Wan, and Lipo Wang. An efficient semi-unsupervised gene selection method via spectral biclustering. *IEEE Transactions on nanobioscience*, 5(2):110–114, 2006.

[104] Michael I Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology*, 15(12):550, 2014.

[105] Aaron TL Lun, Karsten Bach, and John C Marioni. Pooling across cells to normalize single-cell rna sequencing data with many zero counts. *Genome biology*, 17(1):75, 2016.

[106] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[107] Celia Pilar Martinez-Jimenez, Nils Eling, Hung-Chang Chen, Catalina A Vallejos, Aleksandra A Kolodziejczyk, Frances Connor, Lovorka Stojic, Timothy F Rayner, Michael JT Stubbington, Sarah A Teichmann, et al. Aging increases cell-to-cell transcriptional variability upon immune stimulation. *Science*, 355(6332):1433–1436, 2017.

[108] Terumi Matsuoka, Tatsuro Ikeda, Kotaro Fujimaki, and Yutaka Satou. Transcriptome dynamics in early embryos of the ascidian, ciona intestinalis. *Developmental biology*, 384(2):375–385, 2013.

[109] Davis McCarthy, Quin Wills, and Kieran Campbell. *scater: Single-cell analysis toolkit for gene expression data in R*, 2016. R package version 1.2.0.

[110] Davis J McCarthy, Kieran R Campbell, Aaron TL Lun, and Quin F Wills. scater: pre-processing, quality control, normalisation and visualisation of single-cell rna-seq data in r. *bioRxiv*, page 069633, 2016.

[111] Andrew McDavid, Greg Finak, and Masanao Yajima. *MAST: Model-based Analysis of Single Cell Transcriptomics*, 2016. R package version 1.0.5.

[112] Volodymyr Melnykov, Wei-Chen Chen, and Ranjan Maitra. Model-based semi-supervised clustering. *Some theoretical contributions to the evaluation and assessment of finite mixture models with applications*, page 123, 2009.

[113] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2017. R package version 1.6-8.

[114] Stefano Monti, Pablo Tamayo, Jill Mesirov, and Todd Golub. Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. *Machine learning*, 52(1):91–118, 2003.

[115] Daniel Müllner. fastcluster: Fast hierarchical, agglomerative clustering routines for R and Python. *Journal of Statistical Software*, 53(9):1–18, 2013.

[116] Erich Neuwirth. *RColorBrewer: ColorBrewer Palettes*, 2014. R package version 1.1-2.

[117] Anna Ochab-Marcinek and Marcin Tabaka. Bimodal gene expression in noncooperative regulatory systems. *Proceedings of the National Academy of Sciences*, 107(51):22096–22101, 2010.

[118] Stack Overflow. Cutting SciPy hierarchical dendrogram into clusters via a threshold value. https://stackoverflow.com/questions/28687882/cutting-scipy-hierarchical-dendrogram-into-clusters-via-a-threshold-value. Accessed: 2017-07-8.

[119] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[120] Victor Perrier and Fanny Meyer. *shinyWidgets: Custom Inputs Widgets for Shiny*. R package version 0.3.1.110.

[121] Mihaela Pertea, Geo M Pertea, Corina M Antonescu, Tsung-Cheng Chang, Joshua T Mendell, and Steven L Salzberg. Stringtie enables improved reconstruction of a transcriptome from rna-seq reads. *Nature biotechnology*, 33(3):290–295, 2015.

[122] David Peters. Logit, probit ordinal models (lecture 1). Accessed: 2017-06-21, 2016.

[123] phschool.com. http://www.phschool.com/science/biology_place/biocoach/transcription/tcproc.html. Accessed: 2017-06-08.

[124] Olivier B Poirion, Xun Zhu, Travers Ching, and Lana Garmire. Single-cell transcriptomics bioinformatics and computational challenges. *Frontiers in Genetics*, 7, 2016.

[125] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016.

[126] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[127] Germán Rodrıguez. Models for count data with overdispersion. 2013.

[128] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[129] Andras Sali. *shinycssloaders: Add CSS Loading Animations to 'shiny' Outputs*, 2017. R package version 0.2.0.

[130] Rahul Satija. *Seurat: Seurat : R toolkit for single cell genomics*. R package version 1.4.0.14.

[131] Rahul Satija, Jeffrey A Farrell, David Gennert, Alexander F Schier, and Aviv Regev. Spatial reconstruction of single-cell gene expression data. *Nature biotechnology*, 33(5):495–502, 2015.

[132] Juliane Schafer, Rainer Opgen-Rhein, Verena Zuber, Miika Ahdesmaki, A. Pedro Duarte Silva, and Korbinian Strimmer. *corpcor: Efficient Estimation of Covariance and (Partial) Correlation*, 2017. R package version 1.6.9.

[133] Juliane Schäfer and Korbinian Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical applications in genetics and molecular biology*, 4(1), 2005.

[134] Mark Schena, Dari Shalon, Ronald W Davis, and Patrick O Brown. Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270(5235):467, 1995.

[135] RNA seq blog. RPKM, FPKM and TPM, clearly explained. http://www.rna-seqblog.com/rpkm-fpkm-and-tpm-clearly-explained/. Accessed: 2017-07-31.

[136] Chunxuan Shao and Thomas Höfer. Robust classification of single-cell transcriptome data by nonnegative matrix factorization. *Bioinformatics*, page btw607, 2016.

[137] Ehud Shapiro, Tamir Biezuner, and Sten Linnarsson. Single-cell sequencing-based technologies will revolutionize whole-organism science. *Nature Reviews Genetics*, 14(9):618–630, 2013.

[138] Richard Simon, Amy Lam, Ming-Chung Li, Michael Ngan, Supriya Menenzes, and Yingdong Zhao. Analysis of gene expression data using brb-array tools. *Cancer informatics*, 3, 2007.

[139] M Sklar. *Fonctions de répartition à n dimensions et leurs marges*. Université Paris 8, 1959.

[140] Edwin M Southern. Dna microarrays: history and overview. *DNA Arrays: Methods and Protocols*, pages 1–15, 2001.

[141] Wolfram Stacklies, Henning Redestig, Matthias Scholz, Dirk Walther, and Joachim Selbig. pcamethods – a bioconductor package providing pca methods for incomplete data. *Bioinformatics*, 23:1164–1167, 2007.

[142] Oliver Stegle, Sarah A Teichmann, and John C Marioni. Computational and analytical challenges in single-cell transcriptomics. *Nature Reviews Genetics*, 16(3):133–145, 2015.

[143] Lincoln Stein, Paul Sternberg, Richard Durbin, Jean Thierry-Mieg, and John Spieth. Wormbase: network access to the genome and biology of caenorhabditis elegans. *Nucleic acids research*, 29(1):82–86, 2001.

[144] Youssef Stitou, Noureddine Lasmar, and Yannick Berthoumieu. Copulas based multivariate gamma modeling for texture classification. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 1045–1048. IEEE, 2009.

[145] Charles J Stone. Consistent nonparametric regression. *The annals of statistics*, pages 595–620, 1977.

[146] Ritsuko Suyama, Filipe Tavares-Cadete, and Garth Ilsley. Single cell transcriptome analysis in ciona intestinalis. private communication, 2016.

[147] Fuchou Tang, Catalin Barbacioru, Yangzhou Wang, Ellen Nordman, Clarence Lee, Nanlan Xu, Xiaohui Wang, John Bodeau, Brian B Tuch, Asim Siddiqui, et al. mrna-seq whole-transcriptome analysis of a single cell. *Nature methods*, 6(5):377–382, 2009.

[148] Yuan Tang, Masaaki Horikoshi, , and Wenxuan Li. *ggfortify: Data Visualization Tools for Statistical Analysis Results*, 2016.

[149] David T Ting, Ben S Wittner, Matteo Ligorio, Nicole Vincent Jordan, Ajay M Shah, David T Miyamoto, Nicola Aceto, Francesca Bersani, Brian W Brannigan, Kristina Xega, et al. Single-cell rna sequencing identifies extracellular matrix gene expression by pancreatic circulating tumor cells. *Cell reports*, 8(6):1905–1918, 2014.

[150] Sophia C Tintori, Erin Osborne Nishimura, Patrick Golden, Jason D Lieb, and Bob Goldstein. A transcriptional lineage of the early c. elegans embryo. *Developmental cell*, 38(4):430–444, 2016.

[151] Cole Trapnell, Davide Cacchiarelli, Jonna Grimsby, Prapti Pokharel, Shuqiang Li, Michael Morse, Niall J Lennon, Kenneth J Livak, Tarjei S Mikkelsen, and John L Rinn. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature biotechnology*, 32(4):381–386, 2014.

[152] Cole Trapnell, Adam Roberts, Loyal Goff, Geo Pertea, Daehwan Kim, David R Kelley, Harold Pimentel, Steven L Salzberg, John L Rinn, and

Lior Pachter. Differential gene and transcript expression analysis of rna-seq experiments with tophat and cufflinks. *Nature protocols*, 7(3):562–578, 2012.

[153] Barbara Treutlein, Doug G Brownfield, Angela R Wu, Norma F Neff, Gary L Mantalas, F Hernan Espinoza, Tushar J Desai, Mark A Krasnow, and Stephen R Quake. Reconstructing lineage hierarchies of the distal lung epithelium using single-cell rna-seq. *Nature*, 509(7500):371–375, 2014.

[154] George C Tseng. Penalized and weighted k-means for clustering with scattered objects and prior information in high-throughput biological data. *Bioinformatics*, 23(17):2247–2255, 2007.

[155] Harold Underwood. Short read alignment and assembly: An introduction. http://slideplayer.com/slide/7907109/. Accessed: 2017-06-08.

[156] Dmitry Usoskin, Alessandro Furlan, Saiful Islam, Hind Abdo, Peter Lönnerberg, Daohua Lou, Jens Hjerling-Leffler, Jesper Haeggström, Olga Kharchenko, Peter V Kharchenko, et al. Unbiased classification of sensory neuron types by large-scale single-cell rna sequencing. *Nature neuroscience*, 18(1):145–153, 2015.

[157] Vincent Q. Vu. *ggbiplot: A ggplot2 based biplot*, 2011. R package version 0.55.

[158] Zhong Wang, Mark Gerstein, and Michael Snyder. Rna-seq: a revolutionary tool for transcriptomics. *Nature reviews genetics*, 10(1):57–63, 2009.

[159] Gregory R. Warnes, Ben Bolker, Lodewijk Bonebakker, Robert Gentleman, Wolfgang Huber Andy Liaw, Thomas Lumley, Martin Maechler, Arni Magnusson, Steffen Moeller, Marc Schwartz, and Bill Venables. *gplots: Various R Programming Tools for Plotting Data*, 2016. R package version 3.0.1.

[160] James D Watson and F HC Crick. Molecular structure of nucleic acids. *Resonance*, 9(11):96–98, 2004.

[161] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-sne effectively. *Distill*, 2016.

[162] Hadley Wickham. Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12):1–20, 2007.

[163] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009.

[164] Wikipedia. Caenorhabditis (French). https://fr.wikipedia.org/wiki/Caenorhabditis. Accessed: 2017-06-13.

[165] Wikipedia. Ciona intestinalis. https://en.wikipedia.org/wiki/Ciona_intestinalis. Accessed: 2017-06-13.

[166] Wikipedia. Cluster analysis. https://en.wikipedia.org/wiki/Cluster_analysis. Accessed: 2017-06-07.

[167] Wikipedia. Microarray. https://en.wikipedia.org/wiki/Microarray. Accessed: 2017-06-09.

[168] Yihui Xie. *DT: A Wrapper of the JavaScript Library 'DataTables'*, 2016. R package version 0.2.

[169] Chen Xu and Zhengchang Su. Identification of cell types from single-cell transcriptomes using a novel clustering method. *Bioinformatics*, page btv088, 2015.

[170] Jun Yan et al. Enjoy the joy of copulas: with a package copula. *Journal of Statistical Software*, 21(4):1–21, 2007.

[171] Liying Yan, Mingyu Yang, Hongshan Guo, Lu Yang, Jun Wu, Rong Li, Ping Liu, Ying Lian, Xiaoying Zheng, Jie Yan, et al. Single-cell rna-seq profiling of human preimplantation embryos and embryonic stem cells. *Nature structural & molecular biology*, 20(9):1131–1139, 2013.

[172] Christopher Yau et al. pcareduce: hierarchical clustering of single cell transcriptional profiles. *BMC bioinformatics*, 17(1):140, 2016.

[173] Danni Yu, Wolfgang Huber, and Olga Vitek. Shrinkage estimation of dispersion in negative binomial models for rna-seq experiments with small sample size. *Bioinformatics*, 29(10):1275–1282, 2013.

[174] Qingyang Zhang. Classification of rna-seq data via gaussian copulas. *Stat*, 2017.

[175] Shanrong Zhao, Wai-Ping Fung-Leung, Anton Bittner, Karen Ngo, and Xuejun Liu. Comparison of rna-seq and microarray in transcriptome profiling of activated t cells. *PloS one*, 9(1):e78644, 2014.

[176] Justina Zurauskiene and Christopher Yau. *pcaReduce: Hierarchical Clustering of Single Cell Transcriptional Profiles*, 2015. R package version 1.0.

Appendix A

Physical characteristics

The benchmark has been run with the following hardware configuration:

Table A.1: Testing platform hardware configuration

Devices	**Details**
CPU	Intel Core i5@2.70GHz
Cores	4
Level 2 cache per core	256 KB
Level 3 cache	6 MB

Appendix B

Modeling

I performed some tests on the Tintori, Ciona and Biase datasets (more information about these datasets can be found in the following section), in order to guess the best distribution model for the margin functions and the copula. The first two datasets feature raw count gene expression matrices, whereas the Biase dataset use normalized counts (FPKM).

Marginal distributions

The gene expression values of dropout genes are almost always null, hence can be modelled by the low-magnitude Poisson distribution, thus I focus on non-dropout genes. The functions used for plotting figures B.1, B.2 and B.3 can be found in the file *utils.R* in the **Mine** algorithm code. More plots can be found at: https://github.com/oist-gene-clustering/mine

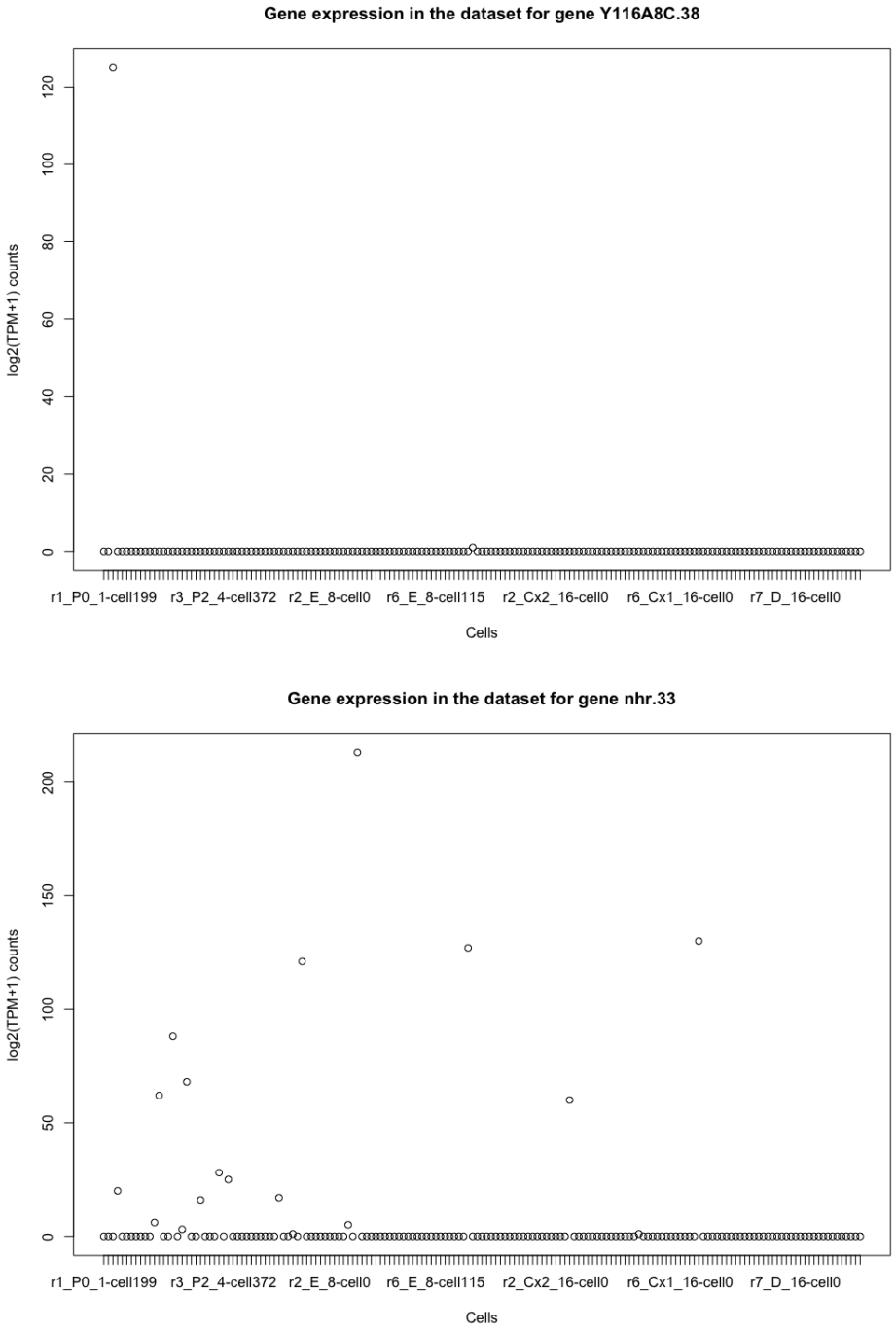


Figure B.1: Gene expression values for two non-dropout genes in the Tintori dataset (cells in x axis, values in y axis)

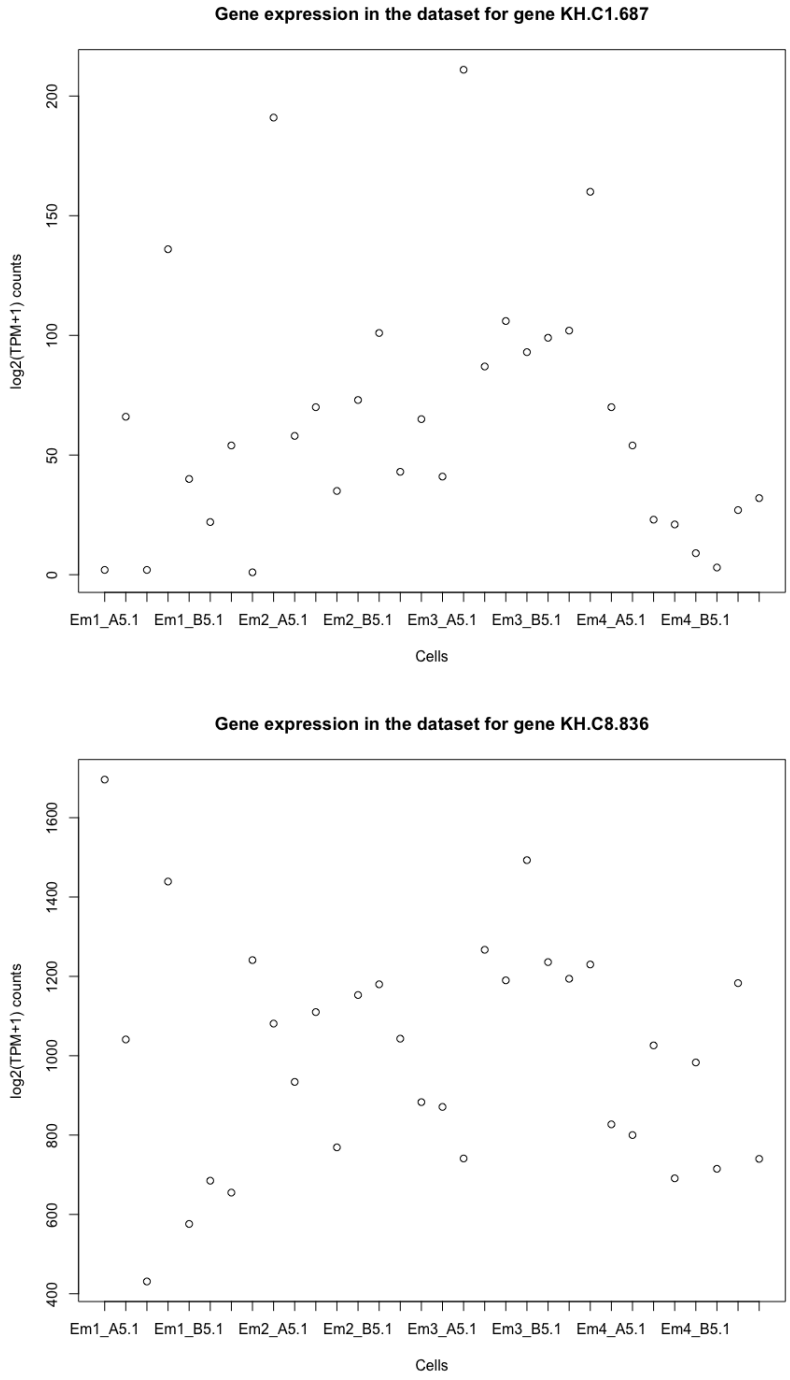


Figure B.2: Gene expression values for two non-dropout genes in the Ciona dataset (cells in x axis, values in y axis)

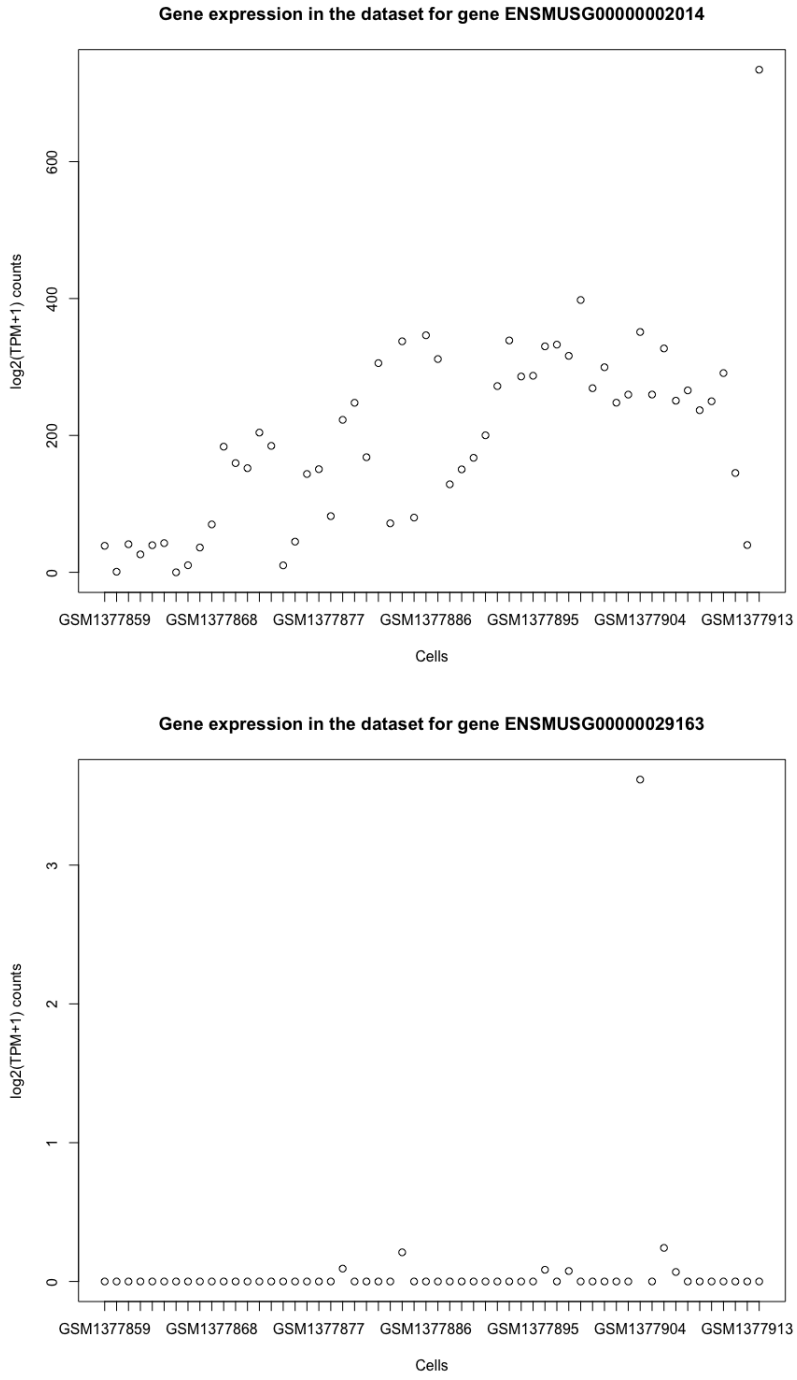


Figure B.3: Gene expression values for two non-dropout genes in the Biase dataset (cells in x axis, values in y axis)

Copula distribution

The following plots (produced by the function *pairs.panels* from package **psych**) represent a histogram of the values found in each cell, the correlation ellipses found between two different cells, the linear model found (the red line) and the correlation number between the two vectors associated with the two cells of the considered pair. Missing values (NA) for dropout genes are due to the null variance of their gene expression values. More plots can be found at: https://github.com/oist-gene-clustering/mine

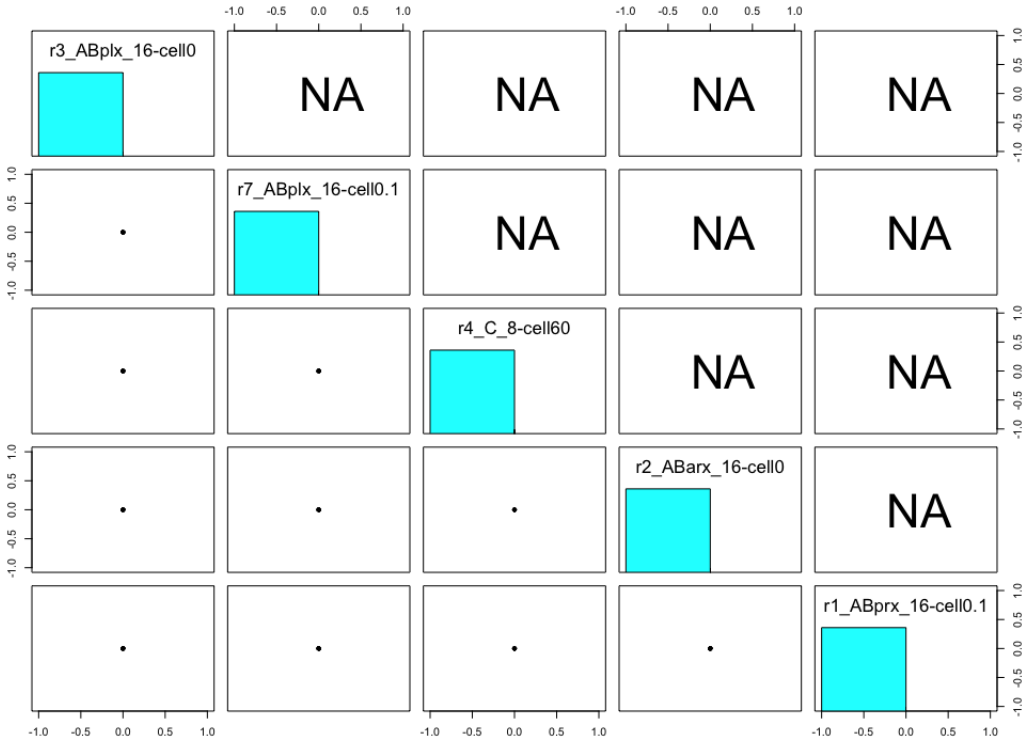


Figure B.4: Correlation between 5 cells (which names are written on the plot) from the Tintori dataset according to the gene expression in 5 randomly-selected **dropout** genes

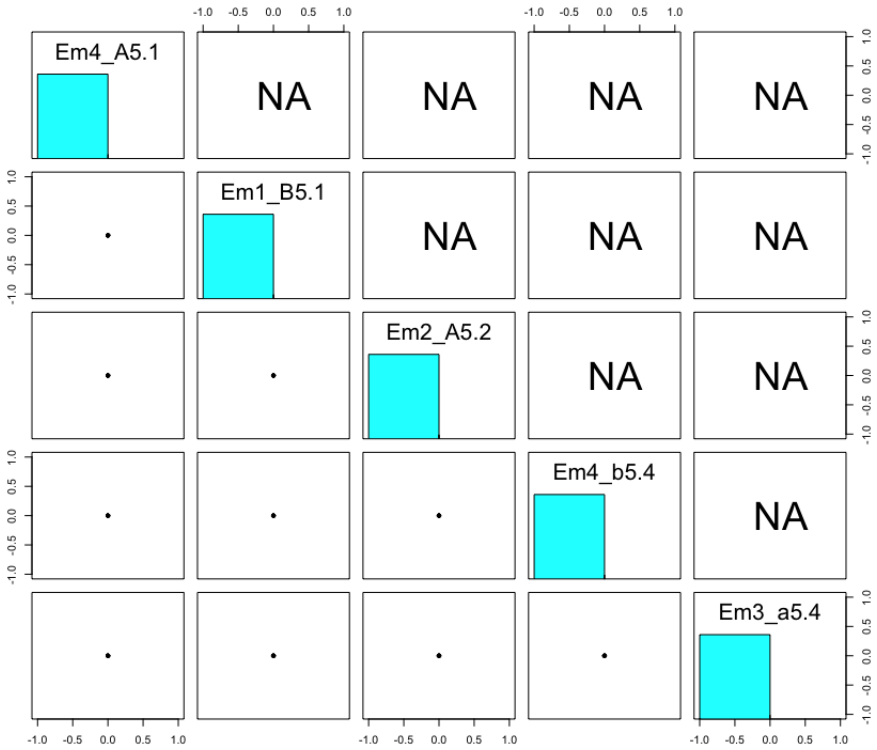


Figure B.5: Correlation between 5 cells (which names are written on the plot) from the Ciona dataset according to the gene expression in 5 randomly-selected **dropout** genes

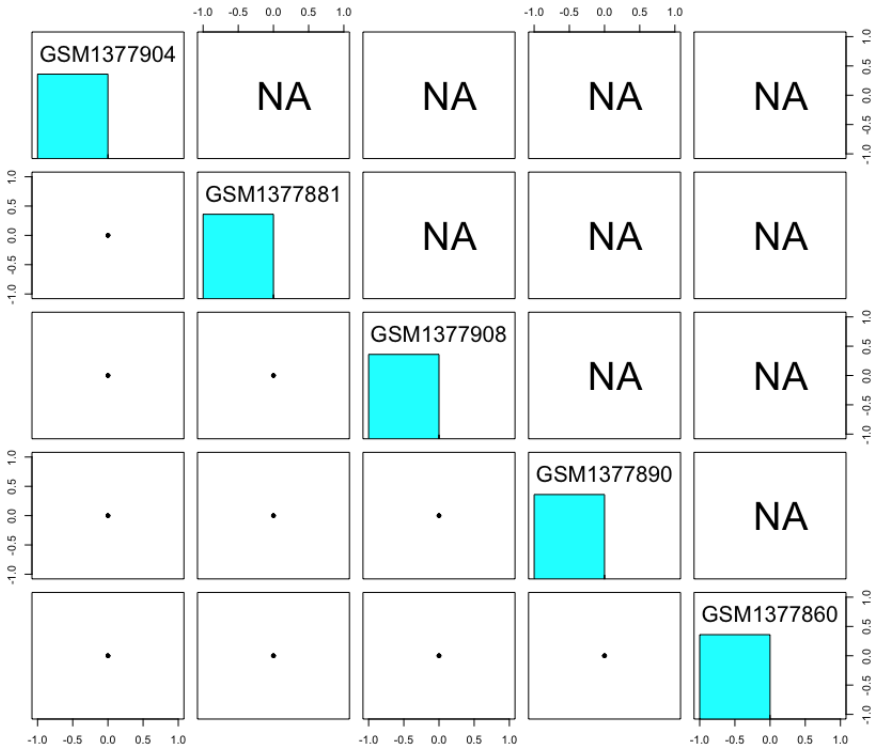


Figure B.6: Correlation between 5 cells (which names are written on the plot) from the Biase dataset according to the gene expression in 5 randomly-selected **dropout** genes

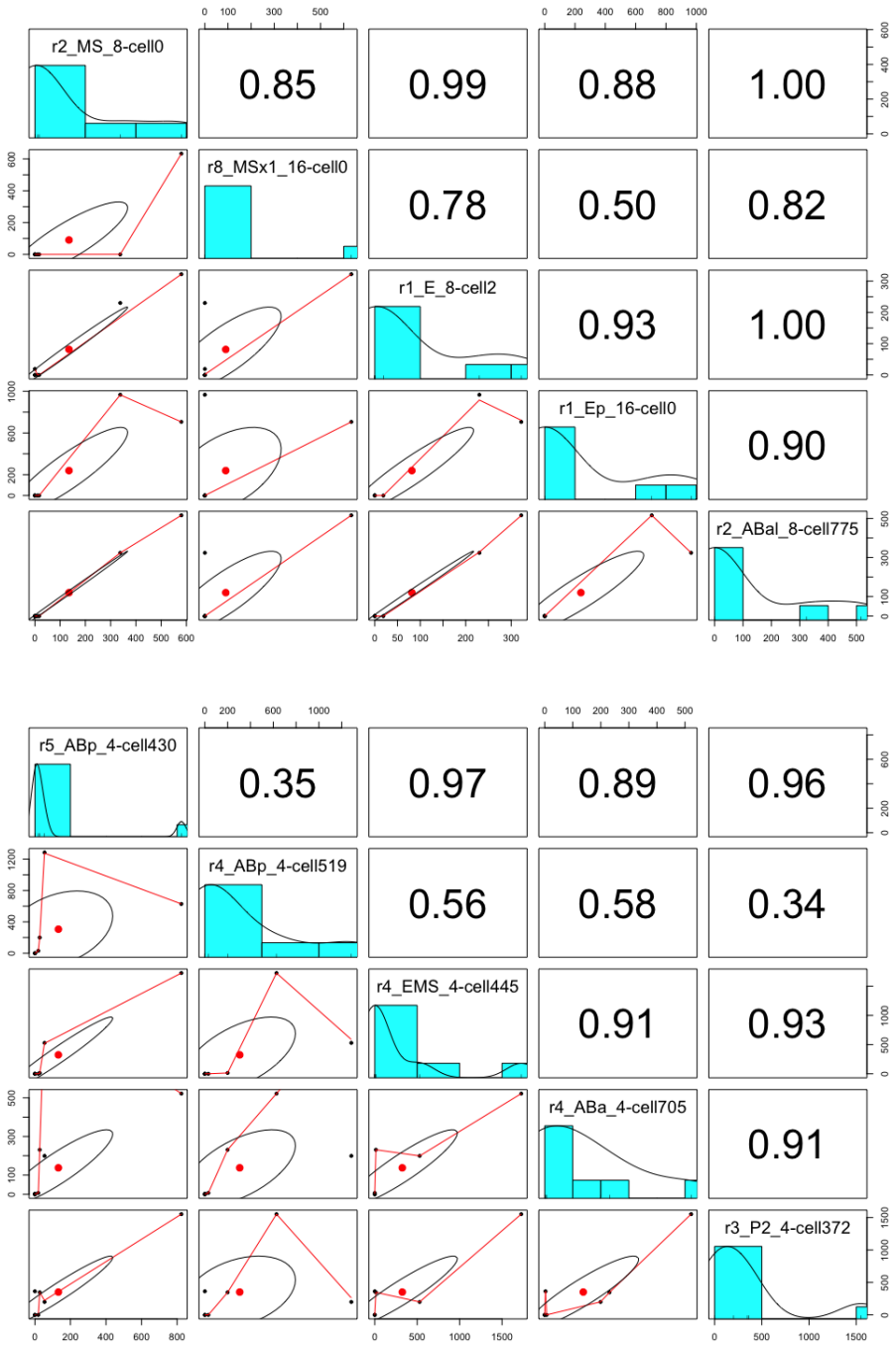


Figure B.7: Correlation between 5 cells (which names are written on the plot) from the Tintori dataset according to the gene expression in 5 randomly-selected **non-dropout** genes (top). Correlation between 5 4-stage cells (which names are written on the plot) from the Tintori dataset according to the gene expression in 5 randomly-selected **non-dropout** genes.

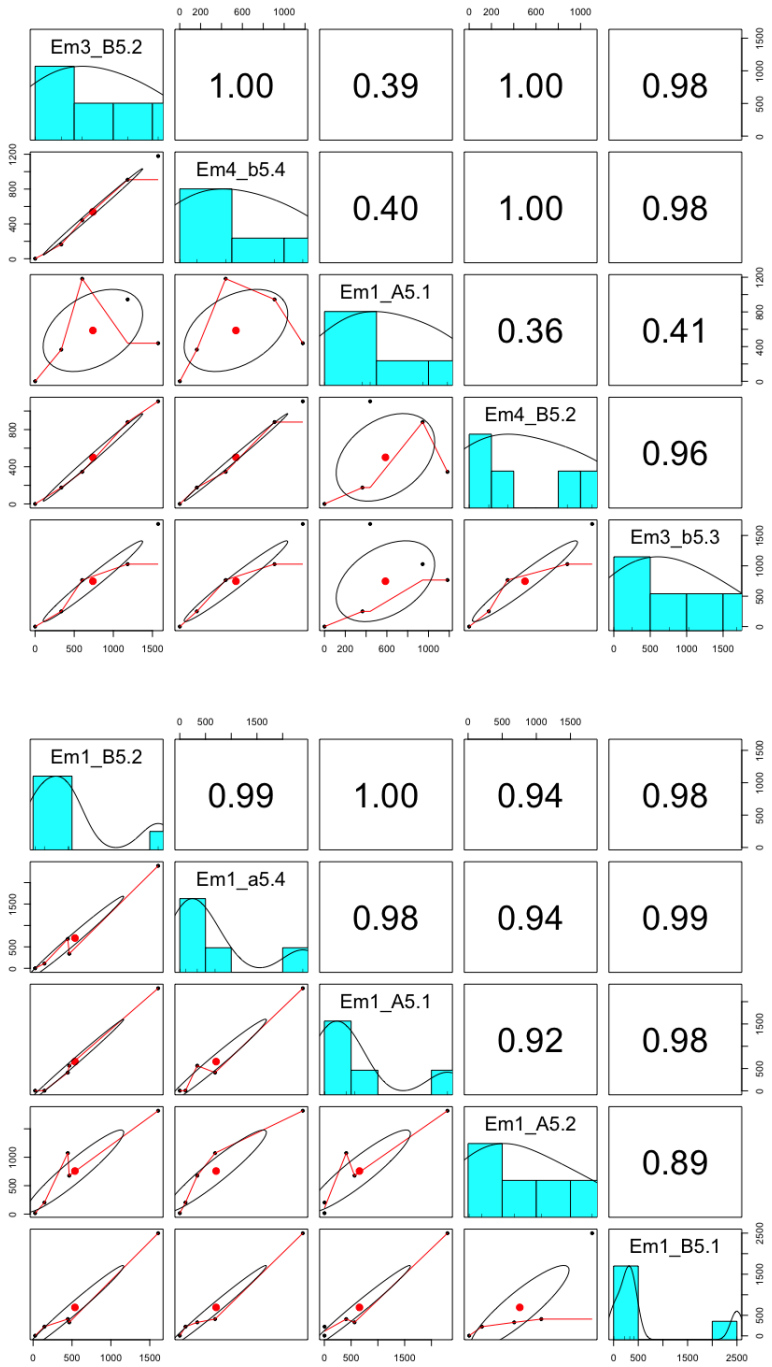


Figure B.8: Correlation between 5 cells (which names are written on the plot) from the Ciona dataset according to the gene expression in 5 randomly-selected **non-dropout** genes (top). Correlation between 5 cells from Embryo 1 (which names are written on the plot) from the Ciona dataset according to the gene expression in 5 randomly-selected **non-dropout** genes.

It can noticed that cells from a same embryo are highly correlated, even if they have different cell types.

Parameter estimation methods

Four sets of parameters need to be estimated: the Poisson-like distribution parameter, the Negative Binomial-like mean and dispersion parameters, the mixture rates for margin and copula distributions, and the copula parameters, that is, the two means of the bimodal copula and the covariance matrix parameter. I used for each of these parameters a non-optimization method, to lessen the computational cost of the algorithm, and to ensure more accuracy and more stability.

Poisson-like parameter

As explained in the previous section, this parameter is set to an arbitrary constant, which is the same as the one used by [86]: $\lambda_0 = 0.1$.

Negative Binomial-like parameters

The mean is found by the method for DESeq2 software, first described by [8], which formula is displayed in the **Modeling** section. The dispersion parameter is found by the shrinkage estimation method described in [173].

Mixture rate parameters

I assumed that the dropout events (resp. mild-high expressiveness events) of different (sets of) genes are independent, thus a logistic regression can be applied to predict the probability of the considered event[9][122][53].

- After application of the MAST function, gene expression values of genes which are considered dropout (i.e. which gene expression value is thought to be background noise) are set to zero.

  Actually, [48] shows the choice of the background noise value does not matter much. I can thus later annotate genes being dropout by marking genes having a null expression value. Dropout events are considered to depend on the mean gene expression value of a feature[86]. However, the absolute value of a gene in a given cell by itself is not useful, the most important part is to know the gene expression value respect to the expression values of other genes in the same cell.

  Hence the values on which the regression will be performed are the following weighted mean values: if C is the set of cells, and G the set of genes, the value associated with a given gene g is: $\sum_{j \in C} \frac{M[g,j]}{\sum_{k \in G} M[k,j]}$.

- The MAST function also computes the "activation" gene expression threshold, such as, if the expression value of a given gene is above this threshold, then this gene is considered highly-expressed, mildly-expressed otherwise. This threshold is the valley -when it exists- between the two peaks in gene expression density, that is, what I see as the two nodes of the bimodal gene expression.

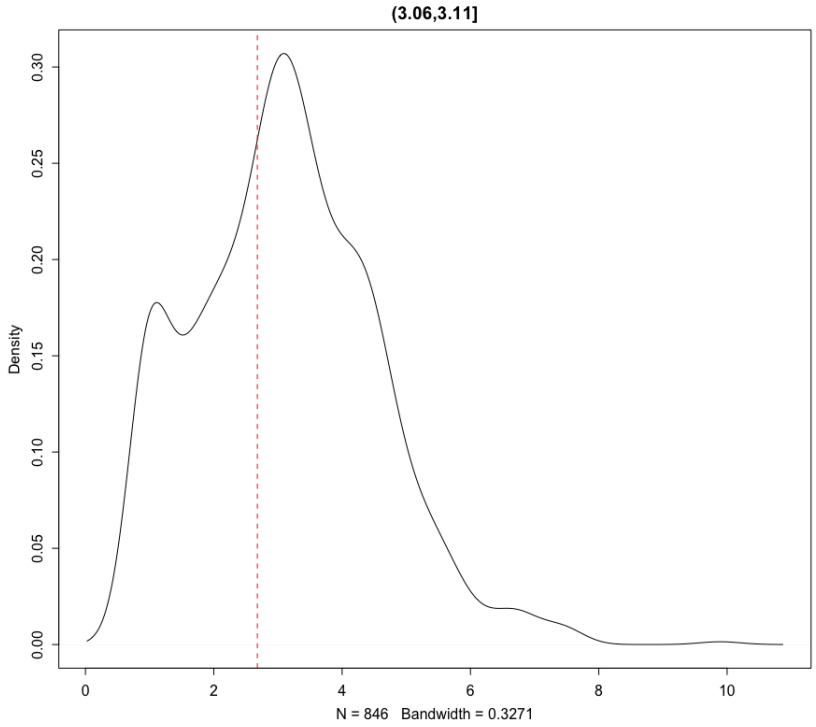


Figure B.9: Result of kernel density estimation[119] (on n=846 points) applied on log-normalized gene expression values (that belong to the interval (3.06,3.11]) from the Ciona[146] scRNAseq dataset. Kernel estimation is a method used to estimate the density function of a random variable (here, the expression value of a gene in a given cell) from a sample. The red vertical line is the threshold found by the MAST package.

Copula parameters

For a given copula, the mean vectors of the bimodal Gaussian copula are the two highest peaks in the kernel density estimation performed by MAST. The covariance matrix is estimated via the corpcor R package[133].

Clustering algorithm

Pre-processing (pre-processing step in cluster analysis)

- For a given positive integer k, $M[k,]$ is the kth row of M, and for i, j positive integers, $M[i, j]$ is the coefficient in ith row and jth column of M.
- TPM (*Transcripts Per Kilobase Million*) is one of the most common normalization methods used for scRNA-seq (see [135] for a review of the different methods). Indeed, as a general rule, the total gene expression count over all the detected genes varies from one cell to another, thus making the raw counts not directly comparable between cells. Also, the length of the coding sequence may change from one gene to another. In order to compare gene expression values of different features in a given sample, counts should also be normalized to take this into account[104].
- Feature selection is an unavoidable step in clustering[103]. I use a quick and intuitive *ad hoc* method, which is similar to [96]. Given my definition of informative genes (see **Modeling** section), I choose a threshold of correlation d', such as if two genes have an absolute correlation value above this threshold, and if they have roughly a similar mean expression value, they are considered truly correlated. Note that the condition on the mean expression values should not be ignored, see [66] for example. Then I compute a distance based on these two conditions on the set of genes, and apply hierarchical clustering on the resulting distance matrix, Then I cut the obtained dendrogram at height $1 - d'$, (see Figure B.10) and keep only one gene of each of the resulting clusters. I chose Spearman's ρ for the correlation function, and an *average-linkage* strategy for the clustering, according to the benchmark performed by [78].
- Null variance rows/features are deleted, for they are non informative of expression changes across the samples
- Rows are also trimmed according to the frequency of expression of a given gene across the set of conditions. If a gene is ubiquitous, it will not be helpful in order to determine the differences between conditions[65][87].

Algorithm 2 Function *filter*

Input: M, gene expression matrix of size $p \times m$
Output: filtered gene expression matrix
Remove rows with missing values
Remove null-variance rows
if M is already normalized **then**
  $\forall i \leq p, \forall j \leq m,\ M_{i,j} \leftarrow \frac{1}{log(2)} \times log(M_{i,j} + 1)$
else
  $\forall i \leq p, \forall j \leq m,\ M_{i,j} \leftarrow \frac{1}{log(2)} \times log(\frac{M_{i,j}}{\sum_{k \leq p} M_{k,j}} \times 10^6 + 1)$
end if
return M

Algorithm 3 Function *featureSelect*

Input: M, gene expression matrix of size $p \times m$
(hidden) parameter: d', threshold of correlation between features
Output: gene expression matrix with trimmed rows
dMatrix $\leftarrow$ **NULL**
$\forall i \leq p, \forall k \leq p$, dMatrix[i,k] $\leftarrow 1 - correlation(M[i,], M[k,])$
Apply hierarchical clustering to the row vectors of M using dmatrix
Cut the hierarchical tree at height $1 - d'$
selectedFeatures $\leftarrow$ clusters from the cut tree
Remove from M rows NOT appearing in selectedFeatures
return M

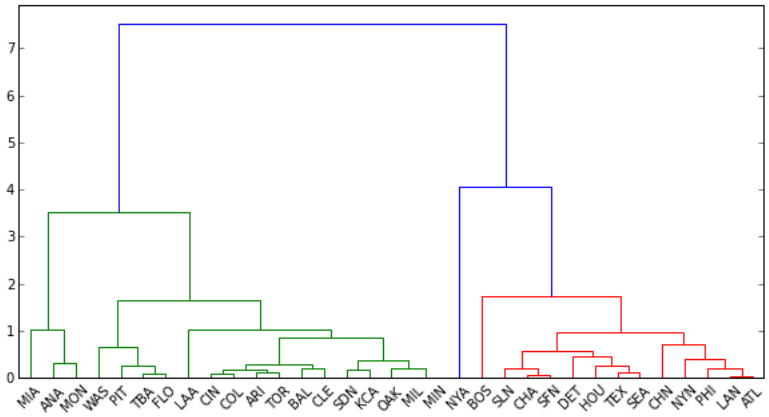


Figure B.10: Example of tree resulting from a hierarchical clustering of the data[118]. The leaves are the clustered elements, and the more the two groups at a given node are similar, the shorter the branches are. If this tree is cut at height 3 (y axis), then there are 4 clusters. If it is cut at height 5, there are two resulting clusters.

Algorithm 4 Function *frequenceTrim*

Input: M, gene expression matrix of size $p \times m$
(hidden) parameter: f, maximum feature frequency allowed in data
Output: trimmed gene expression matrix
Remove in M features present in more than $f \times 100\%$ of the conditions
return M

Find the most probable patterns (pre-processing step)

The idea behind this step is:

1. to classify genes into dropout/non-dropout genes.
2. to find the **activation threshold** for every gene in each cell, that is, if the expression value of a certain gene in a certain cell type is above the so-called activation threshold associated with this gene and this condition, it means that this gene is activated -also called **highly-expressed**- in this cell. Otherwise, the gene is considered **mildly-expressed**.
3. to fit logistic models for dropout and mild/high gene expressiveness events for a given gene, in a given cell.

The *thresholdComputation* function is the *thresholdSCRNACountMatrix* function from MAST R package (described in [48]), which determines adaptively the gene expression value threshold t (using kernel density estimation) such as, if the expression value associated with one gene is below t, then the gene is considered dropout, and its expression value is replaced by a constant value corresponding to background noise (arbitrarily fixed at zero, see [48] for details). It also computes one of the lowest valley points between the two modes of gene expression, which is considered the **activation threshold** (see Figure B.9).

Algorithm 5 Finding pattern part (function *getPatterns*)

Input: M, new dropout-thresholded gene expression matrix of size $p \times m$, and *res*, result of the *thresholdComputation* function
Output: modelDropout and modelMild, logistic models, and P, matrix of cell patterns, Q, pattern-oriented matrix
for $i \leq p$ **do**
  if $\sum_{k \leq p} M[k, j] = 0$ **then**
    $v \leftarrow 0$
  else
    $v \leftarrow \sum_{j \leq m} \frac{M[i,j]}{\sum_{k \leq p} M[k,j]}$
  end if
  if $v = 0$ **then**
    Mark gene i as dropout
  else
    Mark gene i as non-dropout
  end if
end for
thresholds $\leftarrow$ list of threshold values in res (vector of length m)
for $j \leq m$ **do**
  for $i \leq p$ **do**
    if $M[i, j] \leq$ threshold[j] **then**
      Mark gene i in cell j as mildly-expressed
    else
      Mark gene i in cell j as highly-expressed
    end if
  end for
end for
modelDropout $\leftarrow$ logistic regression using k-fold cross validation
modelMild $\leftarrow$ logistic regression using k-fold cross validation for each condition
$P \leftarrow$ **NULL**
$Q \leftarrow$ **NULL**
for $j \leq m$ **do**
  for $i \leq p$ **do**
    if gene i in cell j marked as mildly-expressed **then**
      $P[i, j] \leftarrow 0$
      $Q[i, j] \leftarrow$ threshold[j]
    else
      $P[i, j] \leftarrow 1$
      $Q[i, j] \leftarrow$ -threshold[j]
    end if
  end for
end for
return modelDropout, modelMild, P, Q

Computation of the gene expression distribution for each cell (pre-processing step)

The objectives of this part of the algorithm are:

1. to find the best estimates for the margin parameters.
2. to find the best estimates for the mixtures rates.
3. to find the best estimates for the copula parameters.

The estimateMarginParameters function uses formulas for mean and dispersion parameter estimation from respectively [8] and [173]. The formula for the mean parameter is recalled below (M being the gene expression matrix):

Definition B.0.1 (Size factor)**.** *For cell j:*

$$sizeFactors[j] = median_{i \leq p} \frac{M[i,j]}{\prod_{l \leq m} (M[i,k]+1)^{(\frac{1}{m})}}$$

Definition B.0.2 (Estimated gene expression mean)**.** *For gene i:*

$$margin\_means[i] = \sum j \leq m \frac{M[i,j]}{m \times sizeFactors[j]}$$

The estimated gene expression dispersion vectors margin_deltas are obtained by the shrinkage estimation method described in [173].

The correlation matrix is found by the method described in [133], which allows to find a target correlation matrix T such as T is diagonal: $\exists \nu \in (\mathbb{R}^+)^p, T = diag(\nu)$, which allows for different variances. This is a commonly used model for gene expression covariance matrices, although it is really restrictive, and it is unsure whether it can actually describe properly the true gene regulatory system. This computation is provided by the R package **corpcor**. I could have used the formula suggested by [36], as it is especially targeted at Gaussian distributions, but the resulting matrix is $\beta \times I_d$, where I_d is the identity matrix and $\beta \in \mathbb{R}^+$, which is a even more constrained model. Optimization methods often fail, because the quadratic system to solve is often singular, due to the large number of zeroes in the gene expression matrix, and when it actually works, the resulting matrix is often ill-conditioned -even when a regularization is applied to the optimization problem- so not only not useful in practice, but also probably wrong biologically speaking.

The mixture rate for margin and copula functions are computed with the logistic regression models fitted as described above.

Merge pattern clusters (pre-processing step)

The idea behind this is to obtain a list of pairs of cell types, ordered by decreasing similarity value, that are similar "enough" (in the sense of the cell similarity definition given previously), that I call "mergeable pairs". The function *computeSimilarity* is the implementation of the measure *sim*. *distMatrix*

is the distance matrix deduced from the similarity values.

Clustering (clustering step)

The strategy for merging cell clusters is close to the K-Nearest Neighbor strategy. The main idea is to consider mergeable pairs in decreasing similarity value order, and, if the two elements of the pair are in different clusters, see if there are K points in the same cluster of one of the elements that are similar "enough" to the other element (*K neighbors agreeing to the merging*), and vice-versa, and, if this condition is satisfied, then merge the clusters.

Definition B.0.3 (Neighbor in a cluster)**.** *The set of neighbors of a point k in a given cluster is the set of all points, different from k, that belong to this cluster.*

Definition B.0.4 (Neighbor agreement)**.** *Let d be a probability threshold ($d \geq 0.5$). A neighbor of a point k agrees with the merging of its cluster and a distinct cluster of a point l iff. $sim(k, l) > d$.*

Definition B.0.5 (Same-cluster neighbor agreement)**.** *Let K be a nonzero integer. K neighbors of a point k agree with the merging of their cluster and a distinct cluster of a point l iff. there are K neighbors of k which agree with the merging.*

Definition B.0.6 (Condition for the merging of clusters)**.** *Let K be a nonzero integer. Two clusters can be merged (via a mergeable pair (k, l)) iff. K neighbors of k and K neighbors of l agree with the merging.*

At first, clusters are initialized to m clusters, containing only one condition each.

K controls the granularity of the clustering algorithm: the more K is big, the more it will be hard to merge two clusters. Furthermore, I chose to have the number of neighbors K needed for agreement specific to one cluster, and change according to the size of this cluster. I want to make the merging of two small clusters easier, and conversely, the merging of a bigger cluster harder, because smaller clusters are most likely more homogenous than big ones, and I want to keep the clusters balanced respect to the correlation between all points in a same cluster. A "good choice" for the K-Nearest Neighbor strategy (with Euclidian distance) is actually having $K \sim \sqrt{n}$, where n is the total number of elements (see Stone's theorem[145]). Although the underlying "distance" used in the merging of the clusters is **not** an Euclidean distance (because it is not a distance...), this strategy seems to suit the behavior I would like to see for the cluster merging, even though it is not asserted by a consistency theorem.

Henceforth, in my algorithm, at a given iteration i of the inner loop, for a given cluster C_i, the number of neighbors needed for agreement K_{C_i} is $\sqrt{\#C_i - 1}$ (because my definition of neighbor does not include the considered node itself).

Measures (Validation step)

In the future, the user would be able to get access to the error rate for the logistic regressions, and on the estimates for the margin and copula parameters.

Appendix C

Benchmark

Ranking

Let A denote the set of algorithms, and D the set of test datasets. Since the datasets have been produced with different methods, and are of variable quality, I decided to take into account the mean performance of all algorithms on a given dataset for the final ranking.

For each quantity q (that gives the value of the corresponding attribute: *time*, *accuracy*, ... for a given dataset and a given algorithm) listed in the columns of the next table (except for *time*), I used the following score function and weights:

Definition C.0.1 (Overall performance on a dataset)**.** *For a dataset d:*

$$\rho_d = median_{a \in A}(accuracy(d, a))$$

Definition C.0.2 (Weight associated with a dataset)**.** *For a dataset d:*

$$\delta_d = \frac{\rho_d}{\sum_{d' \in D} \rho_{d'}}$$

Thus this function gives more weight to results from datasets on which the tested algorithms give their best outputs, because these datasets are considered more relevant than other datasets where the clustering pattern is less easy to find.

Definition C.0.3 (Score function)**.** *For an algorithm a, evaluated for a given quantity q:*

$$score_{q,d}(a) = \sum_{d \in D} \rho_d \times q(d, a)$$

The ranking for an algorithm and a given quantity is the position of the corresponding score in the ordered vector of scores for all algorithms. For the ranking, I only took into account the results from all datasets except for Treutlein, Ciona, Klein and Usoskin, because I could not get results for the NMF and SNN-Cliq algorithms on these datasets (see next section).

Test datasets

The datasets have been formatted by [97]. Using the classification suggested by [87], I sorted the datasets into **gold standard** and **silver standard** datasets, that is, the clustering described in the corresponding article for gold standard datasets has been found or confirmed by biological means, and not by computation or clustering. This information allows to better interpret the number of clusters found by the different algorithms: the clusterings found for gold standard datasets are expected to be less biased than the silver standard ones.

Table C.1 displays the following info about the test datasets:

- the organism from which the biological data comes from.
- whether the considered dataset is gold or silver standard.
- the expected number of clusters, and the number of clusters for the "reference" clustering.
- the dimensions of the gene expression matrix, that is, the number of features times the number of conditions.

Table C.1: Test datasets for the benchmark

Dataset	**Organism**	**Date / Technology**	**Standard**	**Exp. | Ref.**	**Dimensions**
Biase[21]	Mouse	2014 / SMARTer	Gold	5 | 6	$25,737 \times 56$
Ciona[146]	Ciona	2014 / Tang	Gold	5 | 5	$15,288 \times 32$
Deng[41]	Mouse	2014 / SMARTSeq	Gold	9 | 10	$22,958 \times 317$
Goolam[57]	Mouse	2016 / SMARTSeq2	Gold	8 | 5	$41,388 \times 124$
Kolodziejczyk[93]	Mouse	2015 / Wellcome Trust Sanger	Gold	11 | 9	$38,653 \times 704$
Tintori[150]	C. elegans	2016 / SMARTer	Gold	5 | 5	$31,383 \times 299$
Yan[171]	Human	2013 / Tang	Gold	7 | 7	$20,214 \times 90$
Klein[89]	Mouse	2015 / Droplet	Silver	16 | 4	$24,175 \times 2,717$
Treutlein[153]	Mouse	2014 / SMARTer	Silver	12 | 5	$23,745 \times 196$
Usoskin[156]	Mouse	2015 / Islam	Silver	15 | 11	$25,334 \times 622$

Details about the accuracy test performed

Table C.2: ARI value (respect to the reference labels) and number of clusters found by each algorithm in datasets Biase, Ciona, Deng, Goolam and Kolodziejczyk (1)

Dataset (Expected no.) / Algorithm	**SC3**	**PCAReduce**	**K-means**	**tSNE +DBSCAN**
Biase (6)	0.79 | 6	0.98 | 7	0.99 | 6	0.68 | 4
Ciona (5)	-0.11 | 5	-0.07 | 6	-0.04 | 5	0.01 | 2
Deng (9)	0.44 | 9	0.48 | 9	0.00 | 9	0.65 | 7
Goolam (8)	0.40 | 8	0.58 | 9	0.89 | 8	0.06 | 7
Kolodziejczyk (11)	0.85 | 11	0.72 | 9	0.84 | 9	0.55 | 10

Table C.3: ARI value (respect to the reference labels) and number of clusters found by each algorithm in datasets Biase, Ciona, Deng, Goolam and Kolodziejczyk (2)

Dataset (Expected no.) / Algorithm	**SNN-Cliq**	**SINCERA**	**Seurat**	**Fuzzy K-means**
Biase (6)	0.63 | 3	0.98 | 6	0.61 | 6	0.98 | 6
Ciona (5)	-0.10 | 4	-0.12 | 5	0.00 | 1	0.07 | 5
Deng (9)	0.49 | 9	0.35 | 9	0.79 | 9	0.86 | 9
Goolam (8)	0.14 | 8	0.30 | 8	0.44 | 8	0.78 | 8
Kolodziejczyk (11)	0.49 | 9	0.33 | 11	0.71 | 10	0.57 | 9

Table C.4: ARI value (respect to the reference labels) and number of clusters found by each algorithm in datasets Biase, Ciona, Deng, Goolam and Kolodziejczyk (3)

Dataset (Expected no.) / Algorithm	**NMF**	**tSNE +Kmeans**
Biase (6)	0.48 | 6	0.65 | 6
Ciona (5)	NA | NA	-0.04 | 5
Deng (9)	0.54 | 9	0.85 | 9
Goolam (8)	0.33 | 4	0.21 | 8
Kolodziejczyk (11)	NA | NA	0.44 | 9

Table C.5: Reference number of clusters and number of clusters found for each algorithm for datasets Tintori, Yan, Klein, Treutlein and Usoskin (1)

Dataset (Reference no.) / Algorithm	**SC3**	**PCAReduce**	**K-means**	**tSNE +DBSCAN**
Tintori (5)	0.70 | 5	0.53 | 6	0.62 | 5	0.35 | 4
Yan (7)	0.65 | 7	0.92 | 8	0.91 | 7	0.87 | 7
Klein (16)	0.77 | 16	0.51 | 9	0.53 | 9	0.55 | 10
Treutlein (12)	0.28 | 12	0.45 | 9	0.51 | 9	0.44 | 5
Usoskin (15)	0.90 | 11	0.38 | 9	0.51 | 9	0.16 | 10

Table C.6: Reference number of clusters and number of clusters found for each algorithm for datasets Tintori, Yan, Klein, Treutlein and Usoskin (2)

Dataset (Reference no.) / Algorithm	**SNN-Cliq**	**SINCERA**	**Seurat**	**Fuzzy K-means**
Tintori (5)	0.05 | 5	0.32 | 5	0.48 | 5	0.66 | 5
Yan (7)	0.63 | 7	0.58 | 7	0.66 | 5	0.85 | 7
Klein (16)	NA | NA	0.48 | 16	0.41 | 25	0.91 | 9
Treutlein (12)	0.14 | 7	0.33 | 12	0.32 | 10	0.56 | 9
Usoskin (15)	0.15 | 9	0.30 | 11	0.44 | 8	0.55 | 9

Table C.7: Reference number of clusters and number of clusters found for each algorithm for datasets Tintori, Yan, Klein, Treutlein and Usoskin (3)

Dataset (Reference no.) / Algorithm	**NMF**	**tSNE +Kmeans**
Tintori (5)	0.36 | 5	0.35 | 5
Yan (7)	0.43 | 7	0.71 | 7
Klein (16)	NA | NA	0.34 | 9
Treutlein (12)	NA | NA	0.32 | 9
Usoskin (15)	NA | NA	0.17 | 9

Some of the results could not be obtained because the process has been killed before the end of the computation (the "NA" values).

Plots

Clustering accuracy

The functions used to generate the following plots and perform the computation of the ARI can be found in the benchmark files, at: https://github.com/oist-gene-clustering/geneclusteringbenchmark

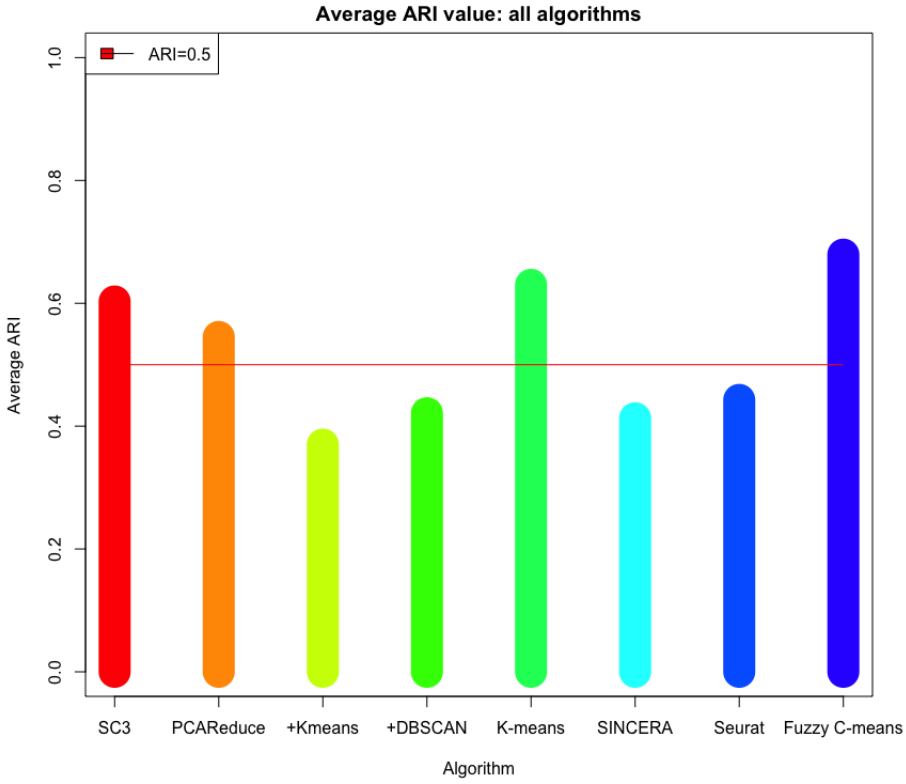


Figure C.1: Average Adjusted Rand Index (ARI) for some algorithms on Biase, Deng, Goolam, Treutlein, Klein, Ciona, Tintori, Kolodziejczyk, Yan and Usoskin datasets

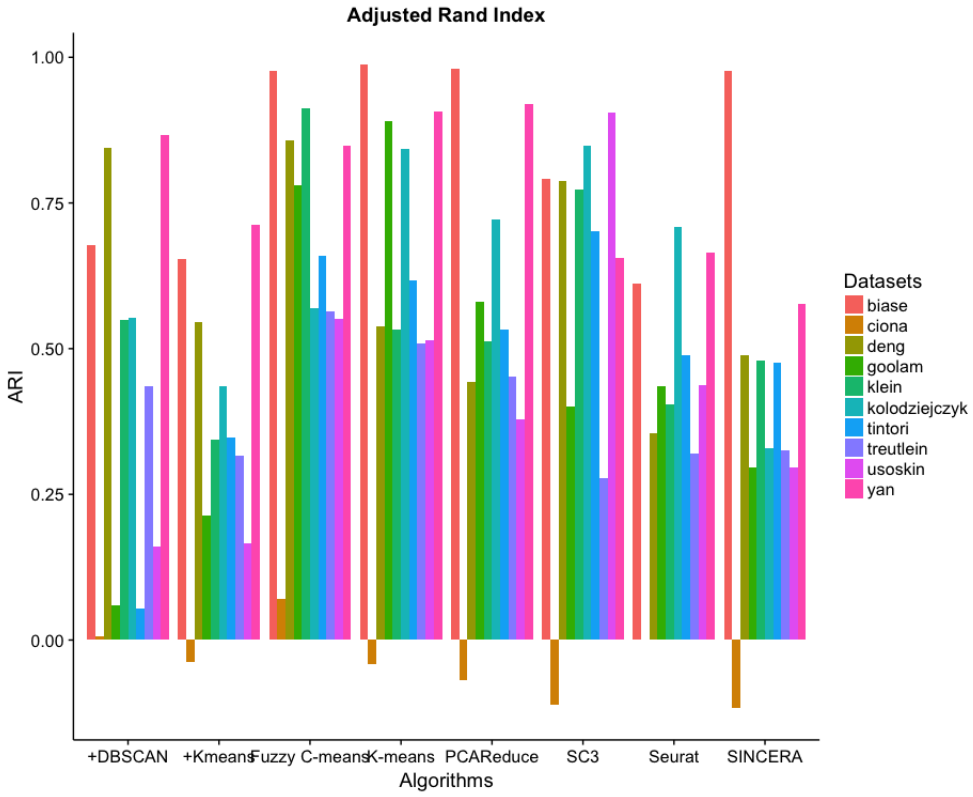


Figure C.2: Average Adjusted Rand Index (ARI) for some algorithms on Biase, Deng, Goolam, Treutlein, Klein, Ciona, Tintori, Kolodziejczyk, Yan and Usoskin datasets

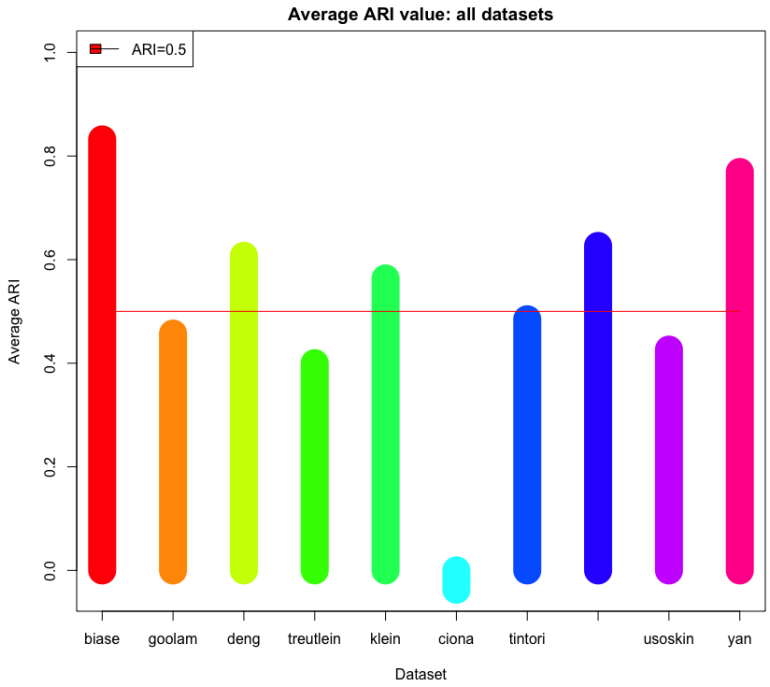


Figure C.3: Average ARI for all tested algorithms per dataset

Stability of the clusterings

The functions used to generate the following plots and compute the stability measure can be found in the benchmark files, at: https://github.com/oist-gene-clustering/geneclusteringbenchmark

The chosen stability measure is the frequency of the most frequent solution, as done in [87].

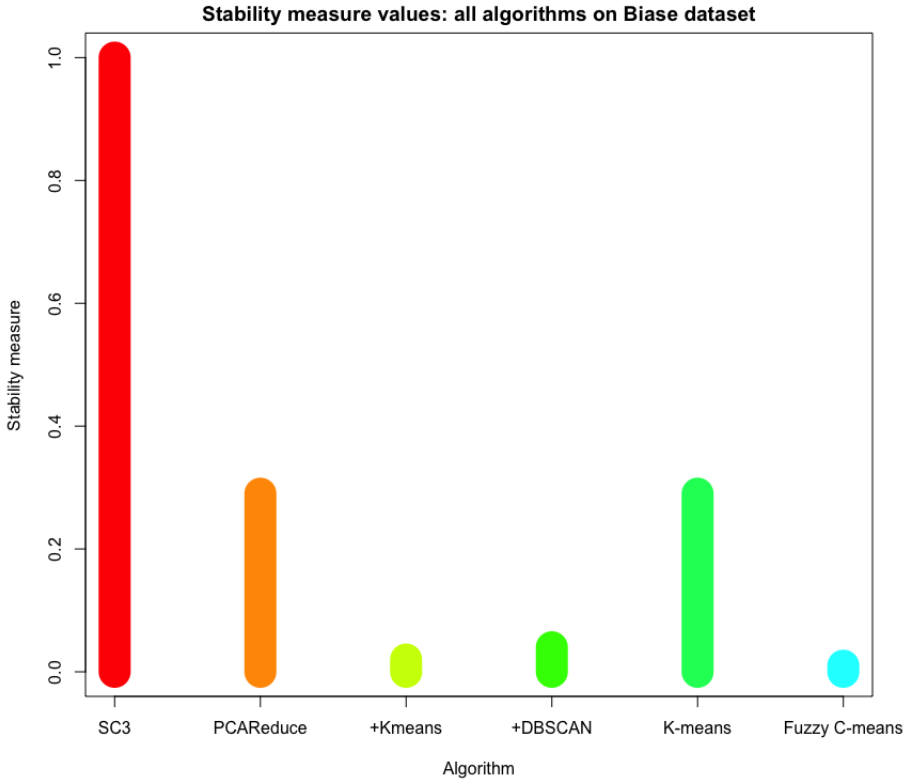


Figure C.4: Stability measure for each algorithm on the Biase dataset, which is the one having the highest mean ARI value overall datasets

But not only it is important to know if the algorithm often returns the same solution, but also knowing how much the returned solutions vary respect to the reference clustering is useful to interpret the correctness of the result. Boxplots help in visualizing quickly the variation of a given measure.

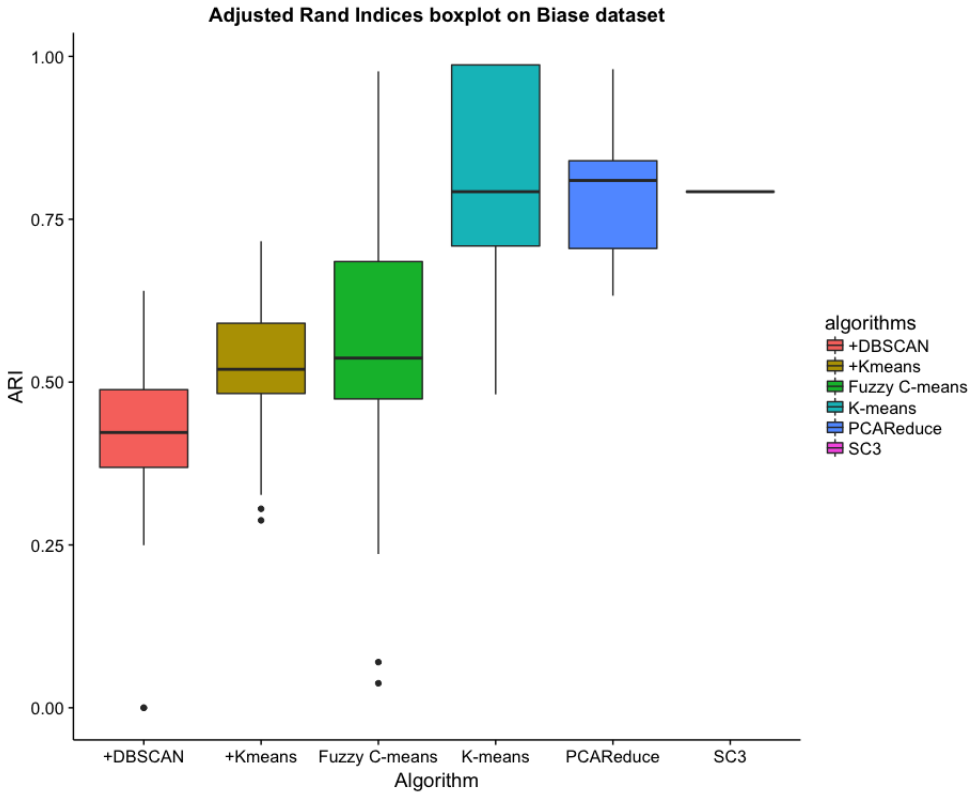


Figure C.5: Adjusted Rand Index (ARI) boxplot for each algorithm on the Biase dataset

According to [1], the thick black horizontal line is the median value (50th percentile) of the measure, and the top of the box is the 75th percentile of the values, the bottom the 25th percentile. The set of percentiles are the 99 values that split the range of the values into 100 equal parts. The top of the upper (resp. the bottom of the lower) black vertical line ("whisker") is, by default, either the maximum (resp. minimum) value, either the 75th percentile (resp. the 25th percentile) plus (resp. minus) $\frac{3}{2}$ of the box length -75th percentile minus 25th percentile.

Time complexity

The functions used to generate the following plots can be found in the benchmark files, at: https://github.com/oist-gene-clustering/geneclusteringbenchmark

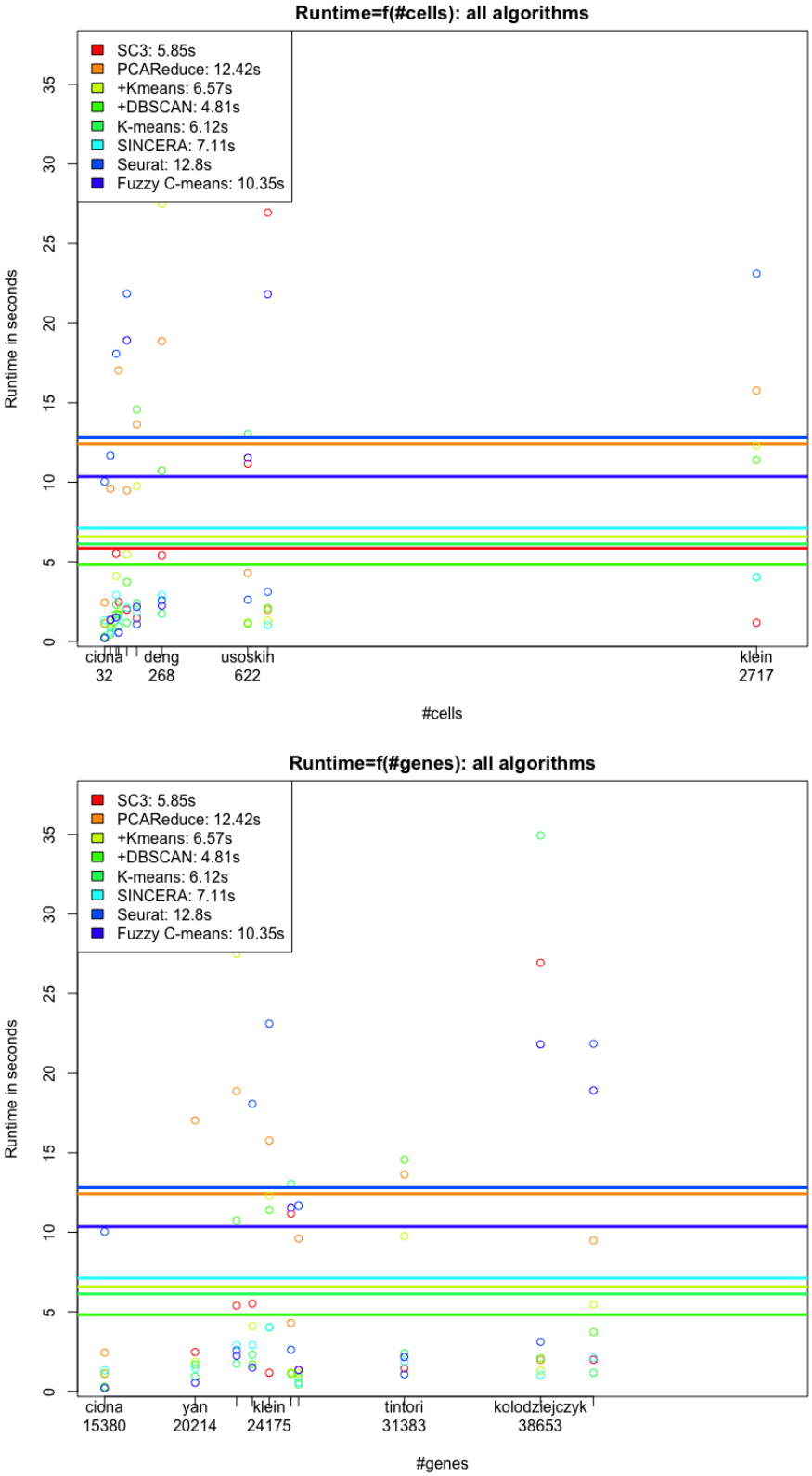


Figure C.6: Time complexity in function of the number of cells (top), of the number of genes (bottom). Lines show the mean runtime for the best solution, for each algorithm across all datasets.

Appendix D

Proofs

Parametrization equivalences

Negative-Binomial distribution

Gamma distribution

Model