NATIONAL UNIVERSITY OF SINGAPORE

UNDERGRADUATE THESIS

Machine Learning For Genomics

Author:

Indrik WIJAYA

Supervisor:

Assoc. Prof Vincent TAN &

Dr. Huili GUO

A thesis submitted in fulfillment of the requirements

for the degree of Bachelor of Science with Honours

April 27, 2018

NATIONAL UNIVERSITY OF SINGAPORE

Abstract

Faculty of Science

Department of Mathematics

Bachelor of Science with Honours

Machine Learning For Genomics

by Indrik WIJAYA

This report will explore the performance of different unsupervised learning algorithms, particularly on clustering, on short time-series data from gene expression values. Many biological data are in the form of short time-series, yet there are not many studies done on this area. Many standard machine learning algorithms normally work well on longer time-series. These algorithms tend to fail to separate different short time-series data into meaningful clusters as the data are not long enough to develop distinct and clear patterns. As such, data that are not supposed to be clustered together, may be clustered together. In this report, we will explore a few algorithms: Short Time-series Expression Miner (STEM), Gaussian Mixture Model, K-means and Hierarchical Clustering. STEM was specifically developed to address the problem of clustering short time-series data, whereas the other three algorithms are the standard machine learning algorithms that are still widely used to cluster time-series data.

Acknowledgements

First and foremost, I would like to thank my **Lord, Jesus Christ** for giving me the strength, knowledge, ability and opportunity to undertake this project under two very amazing supervisors. Without His blessings and steadfast love that continues sustaining me and sending me with people who also support me, I would not be able to finish this properly.

I would like to express my heartfelt gratitude to my supervisors: **Assoc. Prof Vincent Tan** who has sparked further my interest in machine learning and **Dr. Huili Guo** who has patiently and passionately taught me biology and introduced me to bioinformatics, without whom this honours thesis would not have been possible. I am also thankful for them who have indirectly rekindled my passion in academics due to their passion and ability to explain complicated and complex matters in a very easily understandable and simple manner. Their advice and constant guidance have helped me tremendously throughout the writing process of this thesis. I could not have asked for more patient and knowledgeable mentors.

Finally, my sincere thanks goes to all my family members and friends, especially my fellow batchmates who have gone through the journey in NUS since year one, for their continued support and encouragement throughout the writing process of this thesis. I would particularly like to thank the people in **HG lab** who have encouraged me and supported me along the way. Last but not least, I would like to thank **Nancy**, **William** and **Darryl** for the weekly prayer meeting, the listening ears and the patience to bear with me whenever I complained about life.

Contents

List of Figures

List of Tables

1 Introduction and Preview

1.1 Machine Learning

Machine learning is a method of data analysis that automates analytical model building. Using algorithms that learn from data, machine learning allows computers to find hidden insights or patterns without being explicitly programmed where to look. In short, it involves the development of self-learning algorithms to gain knowledge from that data.

Instead of requiring humans to manually derive rules and build models from analyzing large amounts of data, machine learning offers a more efficient alternative for capturing the knowledge in data to gradually improve the performance of predictive models, and make data-driven decisions. Not only is machine learning becoming increasingly important in computer science research but it also plays an ever greater role in our everyday life. Thanks to machine learning, we enjoy robust e-mail spam filters, convenient text and voice recognition software, reliable Web search engine, challenging chess players, and, hopefully soon, safe and efficient self-driving cars.

Biology is also another field that has enjoyed tremendous development due to the rampant use of machine learning algorithms [1] and increasing capability to generate biological data with lower cost and shorter time. Analysis of genome sequencing data sets, including the annotation of sequence elements and epigenetics, proteomic and metabolomic data is one example on how machine learning has helped provide more insights in Biology to understand our lives better.

1.1.1 Unsupervised Learning, Supervised Learning, Semi-supervised Learning

Machine Learning systems can be classified according to the amount and type of supervision they get during training. There are three major categories: supervised, unsupervised learning, semi-supervised learning. In *supervised learning*, the training data we used in the algorithm include the desired solutions, called *labels*. A typical supervised learning task is *classification*. The spam filter is a good and common example of this: it is trained with many example emails along with their *class* (spam or ham), and it must learn how to classify new emails.

In contrast, in *unsupervised learning*, the training data are unlabelled. The machine learning algorithm uses only the unlabelled data and the desired number of different labels to assign as input. This approach requires an additional step in which semantics must be manually assigned to each label, but it provides the benefits of enabling training when labelled examples are unavailable and has the ability to identify potentially novel types of patterns.

A mixture of both *supervised learning* and *unsupervised learning* is called *semi-supervised learning* where the algorithm receives a collection of data points, but only a subset of these data points have associated labels. For this report, we will only focus on *unsupervised learning* as the data are unlabelled.

1.2 Basic Genomics

This section will briefly cover basic ideas, definitions and processes in genomics which are relevant to the project. It hopes to provide a clear understanding to the whole project better. Some of the things that will be covered include:

- Key components in molecular biology: DNA, Transcription, RNA, Translation.
- Mitochondrion and the translational coordination between nuclear and mitochondrial genome during its production.

1.2.1 Molecular Biology

The fundamental idea of molecular biology describes how genetic information is stored and interpreted in the cell: The genetic code of an organism is stored in DNA, which is transcribed into RNA, which is finally translated into protein. Proteins carry out the majority of cellular functions such as motility, DNA regulation, and replication. Though this holds true in most situations, there are a number of notable exceptions to the model. For instance, retroviruses are able to generate DNA from RNA via reverse-transcription.

DNA molecule stores the genetic information of an organism. DNA contains regions called **genes**, which encode for proteins to be produced. Other regions of the DNA contain regulatory elements, which partially influence the level of expression of each gene. **Transcription** is the first step of gene expression, in which a particular segment of DNA is copied into RNA (especially mRNA) by the enzyme RNA polymerase. DNA undergoes transcription to become RNA. **RNA** serves as both the information repository (like DNA today) and the functional workhorse (like protein today) in early organisms. **Translation** is the process in which ribosomes in the cytoplasm or ETR synthesize proteins after the process transcription of DNA to RNA in the cell's nucleus. This project will focus more on **translation**, particularly during mitochondrial biogenesis which will be explained further in the following subsection.

1.2.2 Mitochondria

Mitochondrion, an organelle which is formed by the process above, will be the main focus of our report. Mitochondrion [2] is an important organelle that acts as a key regulator of the metabolic activity of the cell. Mitochondria are best known as the powerhouses of the cell as they produce approximately 90% of the energy required for cellular functions. They are also involved in many other processes and functions such as controlling the concentration of calcium ions within the cell and hormones production. Because mitochondria perform so many functions in different tissues, mitochondrial dysfunction is associated with a large number of diseases, such as neurodegenerative disorders, cardiovascular diseases, cancer, diabetes and obesity.

Mitochondria are produced from the transcription and translation of genes both in the nuclear genome and in the mitochondrial genome. The majority of mitochondrial protein comes from the nuclear genome, while the mitochondrial genome encodes parts of electron transport chain along with mitochondrial rRNA and tRNA. In this report, we want to study particularly on mitochondria which are formed abundantly during mitochondrial biogenesis. Mitochondrial biogenesis is the process by which cells increase their individual mitochondrial mass and copy number to increase the production of ATP as a response to greater energy expenditure.

1.3 Structure of Report

The report will be structured as follows:

- Chapter 1: Introduction and Preview
- Chapter 2: Objective and Literature Review
- Chapter 3: Theories and Tools Used
- Chapter 4: Proposed Algorithms
- Chapter 5: Empirical Study
- Chapter 6: Conclusion and future directions

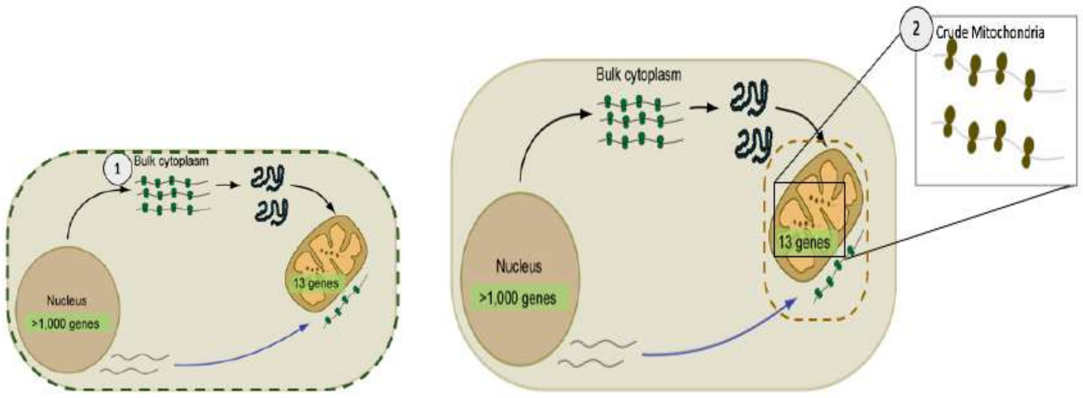
2 Objective and Literature Review

2.1 Objective

The objective of this report is then to apply machine learning algorithms to study an important biological process that happens during mitochondrial biogenesis. Mitochondrial biogenesis and function require the coordinated expression of both the nuclear and mitochondrial genomes [3]. It is estimated that more than 1000 proteins make up the functional mitochondria. However, only 13 of these proteins are encoded by the mitochondrial genome and synthesized within the mitochondrial matrix.

More importantly, the oxidative phosphorylation (OXPHOS) or respiratory complexes, which are responsible for energy production are encoded by both nuclear and mitochondrial genomes. To avoid an imbalance of these subunits, the cell has to precisely coordinate the two genomes. We then want to find out how does the nuclear and mitochondrial genomes communicate with each other. There are 2 sets of data (Figure 2.1) in which each set has another subset of data (mitochondrial genes):

- RNA Bulk
- RNA Bulk Mito (only mitochondrial genes)
- RNA Crude
- RNA Crude Mito (only mitochondrial genes)



Bulk Cytoplasm Crude Mitochondria

FIGURE 2.1: Datasets

By studying genes from these two different pools, we hope to understand the translational coordination between the nuclear and mitochondrial genomes during mitochondrial biogenesis. Specifically, the data will be in the form of expression values (numeric) of each gene at different time points.

2.1.1 Clustering Gene Expression Values

Clustering is an exploratory data analysis technique that allows us to organize a pile of information into meaningful subgroups (*clusters*) without having any prior knowledge of their group memberships. Each cluster that may arise during the analysis defines a group of objects that share a certain degree of similarity but are more dissimilar to objects in other clusters, which is why clustering is also sometimes called 'unsupervised classification'.

Clustering has many applications to computational biology. For example, we consider expression profiles of many genes taken at various developmental stages. Clustering may show that certain sets of genes line up (i.e. show the same expression levels) at various stages. This may indicate that this set of gene has common expression or regulation and we can use this to infer similar function. Furthermore, if we find a uncharacterised gene in such a set of genes, we can reason that the uncharacterised genes also has a similar function through guilt by association.

For this project, we apply this principle which means we are working with the common assumption that genes with similar biological properties will fall under the same cluster in terms of their expression values [4]. Thus, we hope that after getting different clusters or groups of genes, each cluster may exhibit similar biological properties and distinct biological properties from one another. Afterwards, we will then be able to provide an additional information to what kind of communication or mechanism that occurs between nuclear and mitochondrial genomes.

2.2 Literature Review

Time-series expression experiments are used to study a wide range of biological systems. More than 80% of all time series expression datasets are short (8 time points or fewer). These dataset present unique challenges. On account of the large number of genes profiled (often tens of thousands) and the small number of time points many patterns are expected to arise at random. Most clustering algorithms are unable to distinguish between real and random patterns [5].

Clustering algorithms have been extensively applied in the study of time-series gene expression data. Hierarchical clustering [4] and other standard clustering methods such as K-means [6] are often used for this task. Furthermore, time-series data by nature or structure tends to be correlated, which can normally be dealt with the use of multivariate Gaussian mixture model [7]. Although these clustering algorithms have yielded many biological insights, they are not designed for time-series data, especially for short time-series. Specifically, all these methods assume that data at each time point are collected independent of each other, ignoring the sequential nature of time-series data.

Some algorithms have been specifically modified or created to handle time-series biological data. These algorithms include clustering based on the dynamics of the expression patterns [8], clustering using the continuous representation of the profile [9] and clustering using a Hidden Markov Model (HMM) [10, 11]. Spectral clustering [12] has also been developed to deal with time-series data. Nonetheless, there has not been many studies done for spectral clustering and it remains an unpopular clustering algorithm for time-series data.

Furthermore, mixed-effect models or fitting cubic splines [13, 14, 15] to time-series expression values, which are similar to mixture model and Bayesian approach to mixture model [16] have also been developed to improve the clustering of time-series data. While these algorithms work well for relatively long series dataset (10 time points or more), they are not appropriate for short time-series [5]. On account of the noise and the small number of points for each gene, most clustering algorithms cannot distinguish between patterns that occur because of random chance and clusters that represent a real response to the biological experiment. Short Time-series Expression Miner (STEM) [5] is one commonly used algorithm to deal with this problem due to the way it is constructed and its success in producing biologically significant clusters.

Nevertheless, the study of clustering algorithm for time-series data is still inconclusive which means that it may still ultimately depend on the nature of the data itself. For instance, hierarchical clustering remains a popular choice for clustering gene time-series expression values despite its failure for some cases. Hence, other than applying STEM in our report, we will still study popular clustering algorithms such as Gaussian Mixture Model, K-means and Hierarchical Clustering.

3 Theories and Tools Used

This chapter will cover some theories and tools used in the various machine learning algorithms and our analysis.

3.1 Expectation Maximisation (EM) Algorithm

EM is an iterative method that attempts to find the maximum likelihood estimator of a parameter θ of a parametric probability distribution [17]. As a general procedure, EM is used to estimate parameters for probabilistic models with hidden or latent variables. EM algorithm is used both in Gaussian Mixture Model (refer to Chapter 4.2) and K-means (will not be covered in this report).

3.1.1 Outline

Consider a probabilistic model in which we have: observed data y, a parametric density $p(y|\theta)$, a description of some complete data x that we wish we had, and the parametric density $p(x|\theta)$. We assume that the complete data can be modelled as a continuous random vector X with density $p(x|\theta)$, where $\theta \in \omega$ for some set ω. We do not observe X directly; instead, we observe a realization y of the random vector Y that depends on X.

Given that we only have y, we want to find the maximum likelihood estimate (MLE) of θ:

$$\theta_{\text{MLE}} = \arg\max_{\theta\in\omega} \log p(y|\theta)$$

However, for some problem it is difficult to solve the above optimization problem and here where EM comes to help. We make a guess about the complete data X and solve for the θ that maximizes the (expected) log-likelihood of X. And once we have an estimate for θ, we can make a better guess about the complete data X, and iterate. Here are the steps in summary:

- Let $m = 0$ and make an initial estimate $\theta^{(m)} = \theta$.
- Given the observed data y and pretending for the moment that our current guess $\theta^{(m)}$ is correct, formulate the conditional probability distribution $p(x|y, \theta^{(m)})$ for the complete data x.
- Using the conditional probability distribution $p(x|y, \theta^{(m)})$ calculated in Step 2, form the *conditional expected log-likelihood*, which is called the Q-function:
$$Q(\theta|\theta^{(m)}) = \int_{\omega(y)} \log p(x|\theta) p(x|y, \theta^{(m)}) dx = \mathbb{E}_{X|y,\theta^{(m)}} \log p(X|\theta)$$
- Find the θ that maximizes the Q-function; the result is our new estimate $\theta^{(m+1)}$.
- Let $m := m + 1$ and go back to Step 2. The EM algorithm does not specify stopping criterion; standard criterion is to iterate until the estimate stops changing: $||\theta^{(m+1)} - \theta^{(m)}|| < \epsilon$ for some $\epsilon > 0$, or to iterate until the log-likelihood $l(\theta) = \log p(y|\theta)$ stops changing: $|l(\theta^{(m+1)}) - l(\theta^{(m)})| < \epsilon$ for some $\epsilon > 0$.

The monotonicity of the EM algorithm guarantees that as EM iterates, its guesses will not get worse in terms of their likelihood (the proof can be referred to [18]), but the monotonicity alone cannot guarantee the convergence of the sequence $\{\theta^{(m)}\}$. Hence, in practice, we perform EM algorithm multiple times to get the best fit.

3.2 Determining Optimal Number of Clusters

Here we describe a method to determine the optimal number of clusters for Gaussian Mixture Model.

3.2.1 BIC and AIC

When we want to find optimal number of clusters from various clustering algorithms, too few components are not able to model the distribution appropriately, while having too many components cause issues of overfitting. The number of components can be selected according to Akaike Information Criterion (AIC) [19] or the Bayesian Information Criterion (BIC) [20].

Outline

BIC and AIC impose a penalty on the total number of parameters, scaled by the logarithm of sample size, so as to strike a balance between the goodness-of-fit and the model complexity. Both are expressed as a function of the log-likelihood of the converged mixture model:

$$AIC = -2 \log L(\theta) + 2P$$

$$BIC = -2 \log L(\theta) + \log(N)P$$

where P is the number of free parameters and N is the number of data points. The clustering algorithm (Mixture Model in our case) is run for several different values of K, and the model which minimises the chosen criterion is selected. As $\log(N) > 2$ in most cases, BIC more aggresively penalises an increase in P, generally resulting in a smaller choice for K than by AIC. Lower AIC or BIC values indicates better fitting models. We should also ensure that our choices for k and the covariance matrix structure is appropriate for our application.

3.2.2 Elbow Method Analysis

For *k-means*, since we do not know the definitive answer on the optimal number of clusters, we need a method to quantify the quality of clustering. *Elbow Method* is a graphical tool that helps us to estimate the optimal number of cluster k for our task.

Outline

The elbow method plots the value of the cost function produced by different values of k. This is motivated by the use of intrinsic metrics - such as the within-cluster sum-of-squared-error (distortion). Intuitively, we can say that, if k increases, the distortion will decrease; each cluster will have fewer constituent instances, and the instances will be closer to their respective centroids. However, the improvements to the average distortion will decline as k increases. The value of k at which the improvement to the distortion declines the most is called the elbow (Figure 3.1). Here is the method in summary:

- Find the sum of intra-cluster distances between points $\in D_k$ in a given cluster C_k, $C_k = \sum_{x_i \in D_k} \sum_{x_j \in D_k} \|x_i - x_j\|^2 = 2n_k \sum_{x_i \in D_k} = \|x_i - \mu_k\|^2$
- Find the normalized intra-cluster sums of squares: measure compactness of our clustering, $W_k = \sum_{k=1}^{K} \frac{1}{2n_k} C_k$
- Look at the % of variance explained as a function of the number of clusters

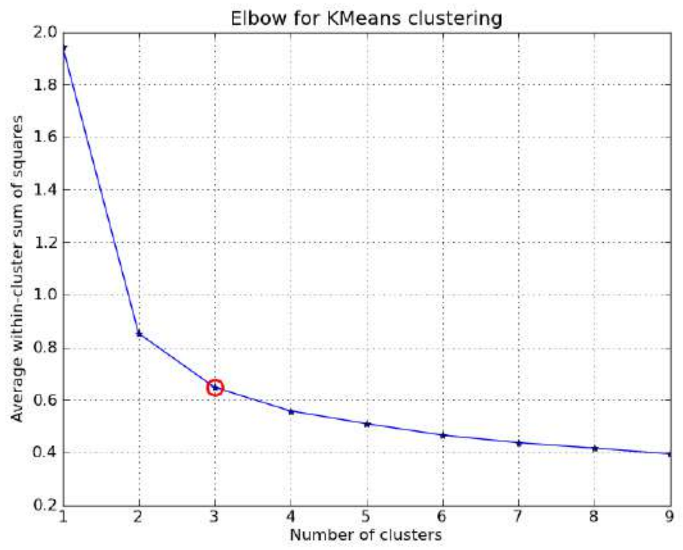


FIGURE 3.1: Ideal elbow curve where it easy to identify the 'elbow' (red circle)

3.2.3 Gap Statistic

Gap Statistic [21] is another popular method used to determine the optimal number of clusters from various clustering algorithms such as *k-means*.

Outline

Given our data $\{x_{ij}\}, i = 1, 2, \ldots, n, j = 1, 2, \ldots, p$, consist of p features measured on n independent observations. Let $d_{ii'}$ denote the distance between observations i and i'. For simplicity, we use euclidean distance $d_{ii'} = \sum_j (x_{ij} - x_{i'j})^2$. Suppose that we have clustered the data into k clusters $C_1, C_2, \ldots, C_k$, with C_r denoting the indices of observations in cluster r, and $n_r = |C_r|$. Let D_k and W_k be the same as that in elbow curve analysis. Then, the idea of gap statistic analysis is to standardize the graph of $\log(W_k)$ by comparing it with its expectation under an appropriate null reference distribution of the data. Our estimate of the optimal number of clusters is the the value of k for which $\log(W_k)$ falls the farthest below this reference curve. Hence, we define $Gap_n(k) = E_n^*\{\log W_k\} - \log W_k$ where E_n^* denotes the expectation under a sample of size n from the reference distribution. Our estimate k^* will be the value maximizing $Gap_n(k)$ after we take the sampling distribution into account. Note that this estimate is very general, applicable to any clustering method and distance measure $d_{ii'}$. Here is the method in summary:

- Compare $\log W_k$ vs null reference distribution of the data, i.e. a distribution with no obvious clustering.
- Find optimal K : $\arg\max_k Gap_n(k) = E_n^*\{\log W_k\} - \log W_k$
- OR Estimate $E_n^*\{\log W_k\} = \frac{1}{B} \sum_{b=1}^{B} \log(W_{kb}^*)$, each of which is computed from a Monte Carlo sample $X_1^*, \cdots, X_n^*$ drawn from our reference distribution.
- Then, Simulation error, $s_k = sd(k)\sqrt{1 + \frac{1}{B}}$, where $sd(k)^2 = \frac{1}{B} \sum_b (\log W_{kb}^* - \frac{1}{B} \sum_{b=1}^{B} \log(W_{kb}^*))^2$
- Find optimal k^* : $\arg\min_k Gap(k) \geq Gap(k+1) - s_{k+1}$. This is a more refined approach for better control of the rejection of the null model.

3.3 Performance Measures

After we have found the optimal number of clusters for our data, we need to evaluate the quality of our clustering. In our project, we have a benchmark algorithm (see 4.1) which will be covered in the next chapter. We assume that this algorithm will yield the most optimal clustering. Then, we want to compare other clustering algorithms to this benchmark algorithm. There are two measures that we use for this project:

3.3.1 Clustering Accuracy (CA)

Let $\{t_i\}$ denote the true classes and $\{c_j\}$ denote the clusters found by a cluster algorithm. We then label all the data in cluster c_j as t_i if they share the most data objects. Note that the number of clusters need not be the same as the number of classes. Then we can calculate the clustering accuracy [12], η as:

$$\eta = \frac{\sum_x I(c_i(x) = t_j(x))}{M}$$

where $I(.)$ is the indicator function, $c_i(x)$ is the label of the cluster which x belongs to, $t_j(x)$ is the true class of x, and M is the size of the dataset. The closer the value to 1, the better the clustering is, and the closer it is to 0, the worse the clustering is.

3.3.2 ME Distance

This method which is first introduced in [22] also computes how far the clusters obtained from any clustering algorithm are from the *true* clusters.

Definition

A *clustering* C of a finite dataset, assumed w.l.o.g to be $\{1, 2, \ldots, n\} = [n]$, is a partition of the dataset into disjoint, non-empty subsets called *clusters*. If the partition has K clusters, we write $C = \{C_1, C_2, \ldots, C_k\}$ and denote by $n_k = |C_k|, \sum_k n_k = n$. A clustering can be represented by a $n \times K$ matrix $\tilde{X}$, whose columns represent the indicator vectors of the K clusters.

$$\tilde{X}_{ik} = \begin{cases} 1 & \text{if } i \in C_k \\ 0 & \text{otherwise} \end{cases}$$

The columns of $\tilde{X}'$ are mutually orthogonal vectors. If we normalize these to length 1, we obtained *normalized* representation X.

$$X_{ik} = \begin{cases} n_k^{-1/2} & \text{if } i \in C_k \\ 0 & \text{otherwise} \end{cases}$$

The Misclassification Error (ME) Distance between Clusterings

The *confusion matrix* of two clusterings $C = \{C_1, C_2, \ldots, C_K\}$ and $C' = \{C'_1, C'_2, \ldots, C'_{K'}\}$ is defined as the $K \times K'$ matrix $M = [m_{kk'}]$ with $m_{kk'} = |C_k \cap C'_k|$. A distance between two clusterings is typically a permutation invariant function of the confusion matrix M. We will assume $K = K'$ for simplicity. The *Misclassification Error (ME)* distance is defined as

$$d(\tilde{X}, \tilde{X}') = 1 - \frac{1}{n} \max_{\pi \in \prod_K} \sum_k m_{k,\pi(k)}$$

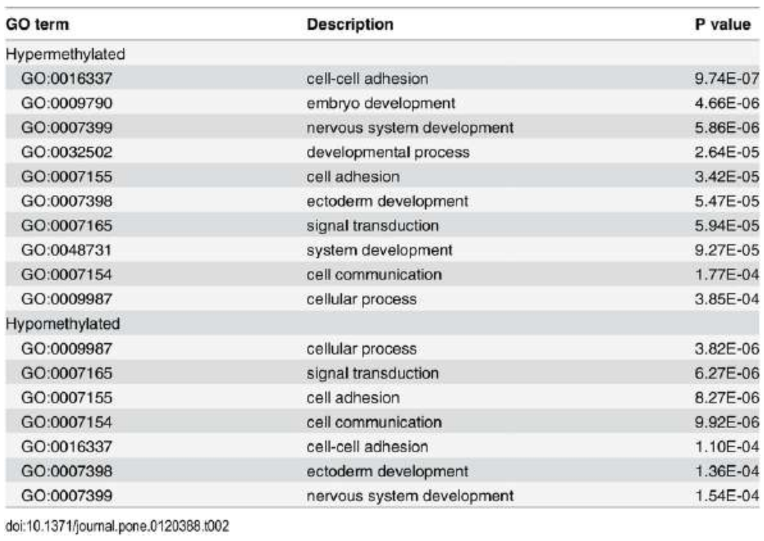
The distance represents the well known cost of classification, minimized over all permutations of the labels $[K]$. d can be computed in polynomial time by a maximum bipartite matching algorithm [23]. Unlike *CA*, we want *ME* distance as close as possible to 1.

3.4 Gene Ontology (GO)

Ultimately, we want to conclude that each cluster obtained from the algorithm contains genes that will share similar biological properties and distinct biological properties with another cluster. We then apply GO analysis on all the different clusters. Ontology on itself means a representation of something we know about. GO then provides an ontology of defined terms representing gene product properties. Each GO term within the ontology has a term name, which may be a word or string of words; a unique alphanumeric identifier; a definition with cited sources; and a name-space indicating the domain to which it belongs. The ontology covers:

- **cellular component**: the parts of a cell or its extracellular environment;
- **molecular function**: the elemental activities of a gene product at the molecular level, such as binding or catalysis;
- **biological process**: operations or sets of molecular events with a defined beginning and end, pertinent to the functioning of integrated living units: cells, tissues, organs, and organisms.

Another category that we will be looking at is KEGG pathway, which tells us a different pathway maps representing our knowledge on the molecular interaction, reaction and relation networks for various biological processes. For this project, we particularly are interested in pathway such as Oxidative Phosphorilation (refer to 2.1). GO helps to perform enrichment analysis on gene sets. For example, given a set of genes that are up-regulated under certain conditions, an enrichment analysis will find which GO terms are over-represented (or under-represented) using annotations for that gene set (Figure 3.2). The results will tell us lists of significant shared GO terms used to describe the set of genes that users entered and *p-value*.



GO term	Description	P value
Hypermethylated		
GO:0016337	cell-cell adhesion	9.74E-07
GO:0009790	embryo development	4.66E-06
GO:0007399	nervous system development	5.86E-06
GO:0032502	developmental process	2.64E-05
GO:0007155	cell adhesion	3.42E-05
GO:0007398	ectoderm development	5.47E-05
GO:0007165	signal transduction	5.94E-05
GO:0048731	system development	9.27E-05
GO:0007154	cell communication	1.77E-04
GO:0009987	cellular process	3.85E-04
Hypomethylated		
GO:0009987	cellular process	3.82E-06
GO:0007165	signal transduction	6.27E-06
GO:0007155	cell adhesion	8.27E-06
GO:0007154	cell communication	9.92E-06
GO:0016337	cell-cell adhesion	1.10E-04
GO:0007398	ectoderm development	1.36E-04
GO:0007399	nervous system development	1.54E-04

doi:10.1371/journal.pone.0120388.t002

FIGURE 3.2: Sample GO

p-value here tells us the probability of seeing at least x number of genes out of the total n genes in the list annotated to a particular GO term, given the proportion of genes in the whole genome that are annotated to that GO term. That is, the GO terms shared by the genes in the user's list are compared to the background distribution of annotation. The smaller the *p-value* is, the more significant the particular GO term associated with the group of genes is (i.e. the less likely the observed annotation of the particular GO term to a group of genes occurs by chance). In other words, when searching the process ontology, if all of the genes in a group were associated with "DNA repair", this term would be significant.

4 Algorithms

4.1 Short Time-series Expression Miner (STEM)

This clustering algorithm has been specifically designed to address the problems with short time-series data of gene expression values [5]. STEM works by assigning genes to a predefined set of model profiles that capture the potential distinct patterns that can be expected from the experiment. It then filters out profiles that are not significant which allows us to obtain the optimal number of clusters. Additionally, it is able to capture the time dependencies of the data. As such, we decided to pick this as our benchmark algorithm. Lastly, the clustering results are verified with GO analysis.

4.1.1 Outline

The following step-by-step list summarizes the STEM algorithm:

- Generate all possible model profiles whereby a user needs to select parameter c to determine the maximum unit of change between time points.
- Select parameter m, the number of distinct model profiles.
- Assign genes to the model profiles obtained previously.
- Determine statistical significance of each of these profiles using *Permutation Test*. Keep the significant profiles.
- Select parameter δ, threshold distance between profiles to group similar significant profiles

This algorithm generates profiles that are actually independent of the data which can be good or bad depending on the data itself. Nevertheless, Friedman [24] argues that this actually reduces the performance of this algorithm. In short, STEM helps in enumerating distinct model profiles that are representative of any expression profile we would probably see.

4.1.2 Generating Model Profiles

The algorithm requires conversion of the raw expression values into log ratios where the ratios are with respect to the expression of the first time point. This means at t_0, all values will be zero. We then define parameter c as the maximum amount of change a gene can have between successive time points. If x_t is the expression value of a gene at time point t, then x_{t+1} can take integer values in $[x_t - c, x_t + c]$. For example, when $c = 2$, a gene can go down either one or two units, remain constant, or go up one or two units.

The number of all possible model profiles will be $|P| = (2c + 1)^{n-1}$, where n is the number of time points in a profile (7 for our data) and P is the set of all possible profiles. As we observe, the number of profiles grows as a high order polynomial in c. For example, with $n = 6$ and $c = 3$, $|P| = 16807$. While, the number goes down to only 3125 when we choose $c = 2$.

Selecting Distinct Model Profiles

Our aim is to select a set of model profiles all of which are distinct from one another, but representative of any expression profile we would probably see. Hence, we want to ensure that the resulting clusters would not be very small. This is done by selecting a set $R \subset P$ with $|R| = m$ such that the minimum distance between any two profiles in R is maximized. Mathematically, this is equivalent to

$$\underset{R\subset P, |R|=m}{\arg\max} \ \underset{p_1, p_2 \in R}{\arg\min} \, d(p_1, p_2) \tag{4.1}$$

where d is a distance metric.

Distance Measure

STEM uses **correlation coefficient** $\rho(x, y)$ as the distance measures as it has been yielding positive results in computational biology, especially when used for clustering [4]. More specifically, STEM uses $gm(x,y) = 1 - \rho(x,y)$ as $\rho(x,y)$ can be negative and does not satisfy the triangle inequality and hence is not a metric.

Even though, $gm(x,y)$ is still not a metric since it also does not satisfy the triangle inequality, it satisfies a generalized version of the triangle inequality. This inequality actually also proves that the correlation coefficient is a transitive measure, which means when using it, two highly dissimilar profiles cannot be very similar to a third profile.

$$gm(x,z) \leq 2(gm(x,y) + gm(y,z))$$

For a set R let $b(R) = \min_{p_1,p_2 \in R} d(p_1, p_2)$, i.e. $b(R)$ is the minimum distance between model profiles in R, which is the quantity we want to maximize. Let R' be the set of profiles that maximizes equation 4.1. Thus, $b(R')$ is the optimal value we can achieve. Unfortunately, this is a NP-hard problem. Moreover, an approximation that guarantees that a solution that is better than $\frac{b(R')}{2}$ is also NP-hard. Here, STEM uses a greedy algorithm that is guaranteed to find such a set, i.e. algorithm finds a set of profiles R, with $b(R) \geq \frac{b(R')}{2}$.

Algorithm 1: Select Distinct Profiles (d, P, m)

```
Let $p_1 \in P$ be the profile that always goes down 1 unit between time points
    1. $R \leftarrow \{p_1\}; L \leftarrow P \backslash \{p_1\}$
    2. for $i \leftarrow 2$ to $m$ do
    3. let $p \in L$ where $p = \arg\max_{p_1 \in R} d(p, p_1)$
    4. $R \leftarrow R \cup \{p\}; L \leftarrow L \backslash \{p\}$
    5. return $R$
```

TABLE 4.1: Greedy Approximation Algorithm to choose a set of m distinct profiles

The following theorem (proof found here [5]) proves the optimality of this algorithm:

Theorem 4.1.1. *Let d be a distance metric. Let $R' \subset P$ be the set of profiles that maximizes 4.1. Let $R \subset P$ be the set of profiles returned by the Table 4.1, then $b(R) \geq \frac{b(R')}{2}$.*

4.1.3 Finding Significant Model Profiles

This step will return different profiles together with the genes assigned to them. Furthermore, this step also tells us the *p-value* of the model profiles, where a small value means that the number of data profiles belonging to the cluster represented by the model profile is significantly large. This involves two steps: assigning the genes according to their expression values to the different model profiles generated earlier and performing *Permutation Test* to identify significant model profiles.

Assigning Genes to Model Profiles

Given a set M of model profiles and a set of genes G, each gene $g \in G$ is assigned to a model expression profile $m_i \in M$ such that $d(e_g, m_i)$ is the minimum over all $m_i \in M$, where e_g is the temporal expression profile for gene g. If the above distance is minimized by $h > 1$ model profiles (i.e. we have ties) then we assign g to all of these profiles, but weigh the assignment in the counts as $\frac{1}{h}$. We count the number of genes assigned to each model profile and denote this number for profile m_i by $t(m_i)$.

Permutation Test

We perform *Permutation Test* to identify model profiles that are significantly enriched for genes in our experiment. The null hypothesis will be that the data are memoryless, i.e. the probability of observing a value at any time point is independent of past and future values. Hence, under H_0, any profile we observe is a result of random fluctuation in the measured values for genes assigned to that profile. Model profiles that represent true biological function deviate significantly from H_0 since many more genes than expected by random chance are assigned to them. *Permutation Test* is used here to quantify the expected number of genes that would have been assigned to each model profile if the data were generated at random.

For each possible permutation we assign genes to their closest model profile. If we let s_i^j be the number of genes assigned to model profile $i = 1, \cdots, m$ in permutation

$j = 1, \cdots, n!$, then $E_i = \frac{\sum_j s_i^j}{n!}$ is the expected number of profiles assigned to the cluster, when considering a random permutation.

Since each gene is assigned to one of the profiles, we can assume that the number of genes, N in each profile $\sim \text{Bin}(|G|, E_i/|G|)$. Thus, the algorithm computes the *p-value*$=$ $P(N \geq t(m_i))$ of seeing $t(m_i)$ genes assigned to profile p_i. Moreover, we need to apply Bonferroni corrections as we are testing more than one model expression profile. Thus, the critical region will be $P(N \geq t(m_i)) \leq \frac{\alpha}{m}$ for α% level of significance.

4.1.4 Grouping Significant Profiles

The idea of grouping significant profiles is motivated by noise. Due to noise, it is impossible to rule out close profiles (even if not the closest) as being the true profile for individual genes. Then, the logical and natural step is to split the set of significant profiles into groups of similar profiles. If we have a measurement of the noise (e.g. from repeat experiments) it is possible to determine a distance threshold δ below which two profiles are considered similar (the difference between genes assigned to these two may be attributed to noise). Such model profiles represent similar enough expression patterns and thus should be grouped together.

We transform this problem of grouping significant profiles into a graph theoretical problem. Define the graph $G(V, E)$ where V is the set of significant model profiles and E the set of edges. Two profiles $v_1, v_2 \in V$ are connected with an edge iff $d(v_1, v_2) \leq \delta$. Cliques in this graph correspond to sets of significant profiles which are all similar to one another. Here we are interested in identifying large cliques of profiles which are all very similar to each other. This naturally leads to a greedy algorithm to partition the graph into cliques and thus to group significant profiles.

In summary, the greedy algorithm grows a cluster C_i around each statistically significant profile p_i. Initially, $C_i = \{p_i\}$. Next, we look for a profile p_j such that p_j is the closest profile to p_i that is not already included in C_i. If $d(p_j, p_k) \leq \delta$ for all profiles $p_k \in C_i$, we add p_j to C_i and repeat this process, otherwise we stop and declare C_i as the cluster for

p_i. After a temporary group has been built for each profile, a final grouping is formed by selecting the largest temporary groups one by one. Since each significant profile must reside in a single final group, the profiles that belong to a selected group are removed from every other temporary group. A new group is added to the list of final groups until there are no profiles left to be grouped.

4.1.5 Challenges

Since the set of potential profiles are not selected from the data, they might contain profiles that do not represent true biological responses. An additional obstacle is that the clustering is done on the noisy and missing expression patterns of genes which might lead to false assignments to clusters [24].

Furthermore, identifying groups of similar model profiles seems like a good idea, yet there are some problems. Aforementioned above, it is not trivial to assign a meaningful value to δ. In the biological results of [5], a value of $\delta = 0.3$ is used.

The grouping of model profiles also complicates the analysis of the clustering results, or at least raises some questions. Should the results of the grouping be integrated into the clustering of the data profiles by combining the clusters that belong to the same group, or would some kind of separate treatment of the grouping results be more appropriate?

4.2 Mixture Model

In this model [18], we are assuming that our data are generated i.i.d. by a mixture of k Gaussians, and we want to estimate the mean and covariances of the k Gaussians. Yet, we do not know which of the observed points came from which Gaussian. Our goal is to estimate the k means and k covariances, and k weights that specify how likely each Gaussian is to be chosen; this entire set of parameters is θ.

4.2.1 Outline

Given n i.i.d. samples $y_1, y_2, \dots, y_n \in \mathbb{R}^d$ drawn from a GMM with k components, the goal is to estimate the parameter set $\theta = \{(w_j, \mu_j, \Sigma_k)\}_{j=1}^k$. The GMM has density

$$p(Y_i = y_i|\theta) = \sum_{j=1}^{k} w_j N(y_i|\mu_j, \Sigma_j), w_j > 0, \sum_{j=1}^{k} w_j = 1$$

Let $\gamma_{ij}^{(m)}$ be the estimate at the mth iteration of the probability that the ith sample was generated by the jth Gaussian component, that is,

$$\gamma_{ij}^{(m)} = P(Z_i = j|y_i, \theta^{(m)}) = \frac{w_j^{(m)} N(y_i|\mu_j, \Sigma_j)}{\sum_{l=1}^{k} w_l^{(m)} N(y_i|\mu_l, \Sigma_l)}$$

which satisfies $\sum_{j=1}^{k} \gamma_{ij}^{(m)} = 1$.

We summarize the whole algorithm in Table 4.2 below.

Algorithm 2: EM for GMM

1. Initialization: Choose initial estimates $w_j^{(0)}, \mu_j^{(0)}, \Sigma_j^{(0)}, j = 1, \dots, k$, and compute the initial log-likelihood, $l^{(0)} = \frac{1}{n} \sum_{i=1}^{n} \log(\sum_{j=1}^{k} w_j^{(0)} N(y_i|\mu_j^{(0)}, \Sigma_j^{(0)}))$.

2. **E-step** For $j = 1, \dots, k$, compute
$\gamma_{ij}^{(m)} = P(Z_i = j|y_i, \theta^{(m)}) = \frac{w_j^{(m)} N(y_i|\mu_j, \Sigma_j)}{\sum_{l=1}^{k} w_l^{(m)} N(y_i|\mu_l, \Sigma_l)}, i = 1, \dots, n$, and $n_j^{(m)} = \sum_{i=1}^{n} \gamma_{ij}^{(m)}$

3. **M-step** For $j = 1, \dots, k$, compute the new estimation

$$w_j^{(m+1)} = \frac{n_j^{(m)}}{n}, \mu_j^{(m+1)} = \frac{1}{n^{(m)}} \sum_{i=1}^{n} \gamma_{ij}^{(m)} y_i,$$

$$\Sigma_j^{(m+1)} = \frac{1}{n_j^{(m)}} \sum_{i=1}^{n} \gamma_{ij}^{(m)} (y_i - \mu_j^{(m+1)})(y_i - \mu_j^{(m+1)})^T,$$

4. Convergence check: Compute the new log-likelihood
$l^{(m+1)} = \frac{1}{n} \sum_{i=1}^{n} \log(\sum_{j=1}^{k} w_j^{(m+1)} N(y_i|\mu_j^{(m+1)}, \Sigma_j^{(m+1)}))$.
Return to Step 2 if $|l^{(m+1)} - l^{(m)}| > \epsilon$ for a present threshold ϵ, otherwise end.

TABLE 4.2: Summary of EM Algorithm for GMM

As the algorithm tends to occasionally converge to sub-optimal solutions, the procedure can be repeated to find the best fit. This model also allows for different types of covariance matrices for each mixture component, which is summarized at Table 4.3 below. For this report, we only consider the two simple cases: Diagonal and Spherical.

Covariance	**Description**	**# parameters**
Full	Each component has its own general covariance matrix (correlated predictors)	$kd(d+1)/2$
Tied	All components share the same general covariance matrix	$d(d+1)/2$
Diagonal	Each component has its own diagonal covariance matrix (uncorrelated predictors)	kd
Spherical	Each component has its own single variance	k

TABLE 4.3: DIfferent Covariance Matrices

4.3 K-means

This searches for a specific number of clusters k maximizing a global target function [18]. Clusters are defined by their center. Iteratively, genes are assigned to the best-matching cluster and then the clusters' center values are updated. In our context and other real-world applications in general, we do not have any ground truth category information about our data. Thus, our goal is to group the samples based on their feature similarities (euclidean distance in this case), which can be achieved using the k-means algorithm that can be summarized by the following four steps (Figure 4.1):

- Randomly pick k centroids from the sample points as initial cluster centers.
- Assign each sample to the nearest centroid $\mu^{(j)}, j \in \{1, \ldots, k\}$
- Move the centroids to the center of the samples that were assigned to it.
- Repeat steps 2 and 3 until the cluster assignments do not change or a user-defined tolerance of a maximum number of iterations is reached.

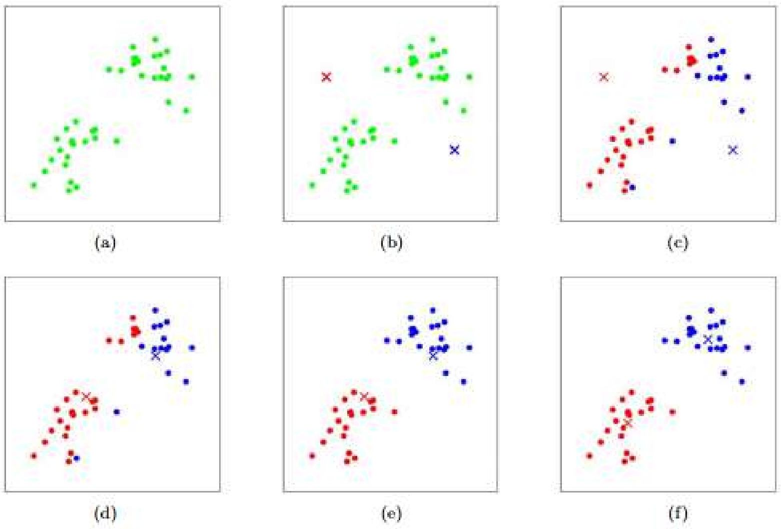


FIGURE 4.1: Iterative steps for k-means

4.3.1 Outline

Mathematically, the steps above can be written as: Given a dataset $D = \{x_t\}_{t=1}^n$ and fix a number of clusters $1 \leq k \leq n$, we want to minimize the cost function

$$J = \sum_{j=1}^{k} \sum_{x \in D_j} \|x - \mu_j\|^2$$

This cost function is the *similarity measure* between our data. D is partitioned into $D_1, D_2, \cdots, D_k$

- Step 0 Initialize the centroids $\mu_j^{(1)}, j = 1, \cdots, k$ randomly
- Step $l \in \mathbb{N}$ (E-step) Assign each point x_t to its closest centroid,

$$j_t = \underset{j=1,\cdots,k}{\arg\min} \|x_t - \mu_j^{(l)}\|^2 \tag{4.2}$$

- Step $l \in \mathbb{N}$ (M-step) Recompute centroids,

$$\mu_j^{(l+1)} = \frac{1}{\|\{t : j_t = j\}\|} \sum_{t:j_t=j} x_t \tag{4.3}$$

Remark: After E-step, we have a partition of $\{1, \cdots, n\}$,

$$D_j^{(l)} = \{t = 1, \cdots, n, j_t^{(l)} = j\},\ \mu_j^{(l+1)} = \frac{1}{\|D_j^{(l)}\|} \sum_{t \in D_j^{(l)}} x_t$$

K-means ++

K-means can sometimes result in bad clusterings or slow convergence if the initial centroids are chosen poorly. One way to address this issue is to run the k-means algorithm multiple times on a dataset and choose the best performing model in terms of the *sum of squared errors*. Another strategy is to place the initial centroids far away from each other via the k-means++ algorithm [25], which leads to better and more consistent results than the classic k-means.

The initialization in k-means++ can be summarized as follows:

- Initialize an empty set M to store the k centroids being selected.
- Randomly choose the first centroid $\mu^{(j)}$ from the input samples and assign it to M.
- For each sample $x^{(i)}$ that is not in M, find the minimum squared distance $d(x^{(i)}, M)^2$ to any of the centroids in M.
- To randomly select the next centroid $\mu^{(p)}$, use a weighted probability distribution equal to $\frac{d(\mu^{(p)},M)^2}{\sum_i d(x^{(i)},M)^2}$
- Repeat steps 2 and 3 until k centroids are chosen.
- Proceed with the classic k-means algorithm.

Challenges

One problem that *k-means* has is that it gives slightly different clusters when it is run repeatedly for time-series data. As such, it is difficult to determine which run is the optimum.

4.4 Hierarchical Clustering

Hierarchical Clustering can be divided into two: divisive and agglomerative. Here, we will discuss more on agglomerative hierarchical (HAC) which has also been extensively used to cluster gene expression values [4].

4.4.1 Outline

The algorithm starts from the trivial clustering that has each data point as a single-point cluster. Then, repeatedly, it merges the "closest" clusters of the previous clustering, as shown in Figure 4.2. Consequently, the number of clusters decreases with each such round. If kept going, it would eventually result in the trivial clustering in which all of the domain points share one large cluster. We then need to choose two parameters to define such an algorithm clearly: the distance (similarity measure or linkage) between clusters and stopping criterion for merging. The most common linkage measures are:

- **Single**: between-clusters distance is defined by the minimum distance between members of the clusters, namely: $D(A, B) = \min\{d(x, y) : x \in A, y \in B\}$
- **Average**: between-clusters distance is defined to be the average distance between a point in one of the clusters and a point in the other, namely: $D(A, B) = \frac{1}{|A||B|} \sum_{x \in A, y \in B} d(x, y)$
- **Max / Complete**: between-clusters distance is defined by the maximum distance between their elements, namely: $D(A, B) = \max\{d(x, y) : x \in A, y \in B\}$
- **Ward**: this involves Ward variance minimization algorithm that deals with an error function. The error function is the average distance of each data point in a cluster to the centroid in the cluster. Between-clusters distance is defined as the error function of the unified cluster minus the error functions of the individual clusters.

4.4.2 Challenges

Unlike other clustering methods, hierarchical clustering does not have any principled way to determine the optimal number of clusters. Thus, we have to determine which of the subtrees are clusters, and which subtrees are only a part of a bigger cluster. If one wishes to turn a dendrogram (lower part of Figure 4.2) into a partition of the space (a clustering), one needs to employ a *stopping criterion*. Common stopping criterions include:

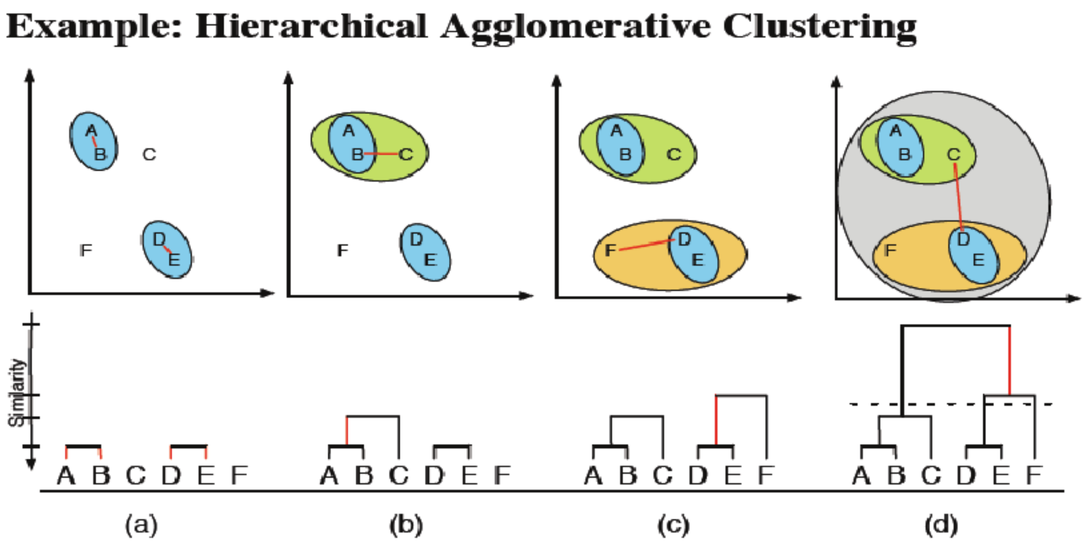


FIGURE 4.2: Iterative steps for hierarchical agglomerative clustering

- Fixed number of clusters: fix some parameter, k, and stop merging clusters as soon as the number of clusters is k.

- Distance upper bound: fix some $r \in \mathbb{R}^+$. Stop merging as soon as all the between-clusters distances are larger than r. We can also set r to be $\alpha \max\{d(x,y) : x, y \in \mathbb{X}\}$ for some $\alpha < 1$. In that case the stopping criterion is called "scaled distance upper bound."

The second stopping criterion basically suggests that we want to stop merging when the cost of merging increases a lot and thus it most likely loses a lot of structure. As such, this leaves us to decide how big a merging cost is acceptable and there is no proven theory to say that this method will often or even usually lead to good choices. In addition, we may also face difficulty in picking the right similarity measure (or linkage) for our data.

5 Empirical Study

In this chapter, we will apply the algorithms mentioned previously on the different datasets to study the coordination between nuclear and mitochondrial genomes. Before that, we will describe the data in more detail.

5.1 Description of Data

The data that we have is **unlabelled** mouse's gene expression values which are in numeric. The values have also been transformed to $\log_2$, which is a standard practice in computational biology. Since it is unlabelled, and from the plots we cannot identify distinct clusters, we have to apply unsupervised learning (clustering in this case). It is a short time-series, with only 7 time points. Each time point has around $8,000 - 10,000$ samples. On top of that, these $8,000 - 10,000$ genes may not exhibit separable or significant clusters so we will look deeper into a special set of genes which are the mitochondrial genes (around $1,100$ samples). We would expect to see more significant separation for this $1,100$ samples. We would also expect that this $1,100$ genes to be highly expressed (higher value) at later time points since they become more abundant in the mitochondrial biogenesis process.

For this report, we will only present results mainly from RNA Crude and RNA Crude Mito as we would expect these pools to be more biologically significant. Furthermore, we will not display all the plots of the different clusters. Instead, we will only present those plots with significantly enriched GO terms. The rest of the results can be found in the appendix A.

5.1.1 Exploratory Analysis

Here we present some basic analysis for our data. From the figures below, we may expect that Gaussian Mixture Model (refer to Section 4.2) will be useful in clustering our time-series gene expression values since all the plots show that the data follow Gaussian normal distribution. Nonetheless, from these figures, we also see that our hypothesis is not totally true as the mitochondrial genes at later time points are not that highly expressed as compared to the earlier time points. The distribution of the values, the maximum and minimum values are generally similar for all time points. This may pose some problems when we cluster the mitochondrial genes as they may not exhibit distinct properties as we expect.

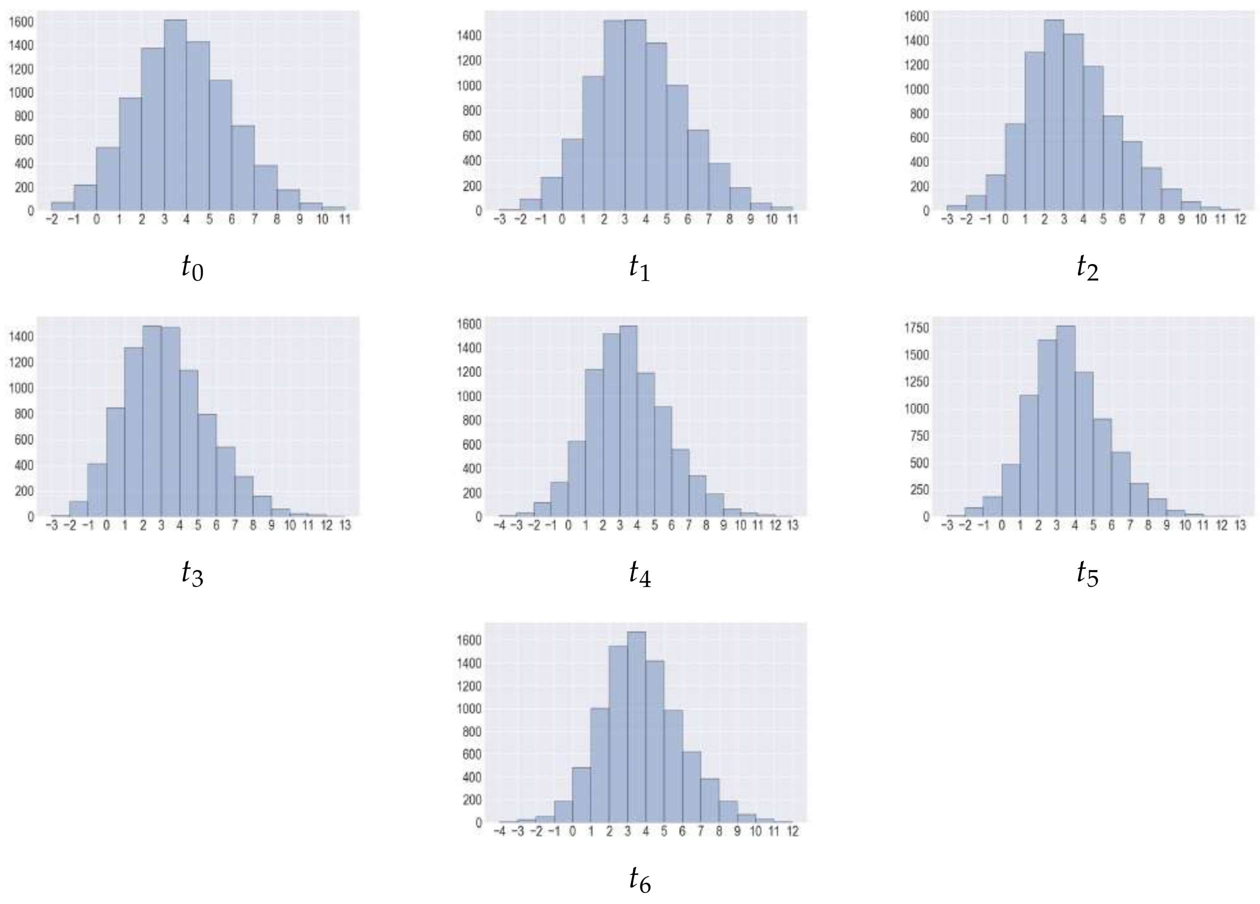


FIGURE 5.1: Distribution of $\log_2 RPKM$ values for RNA Crude

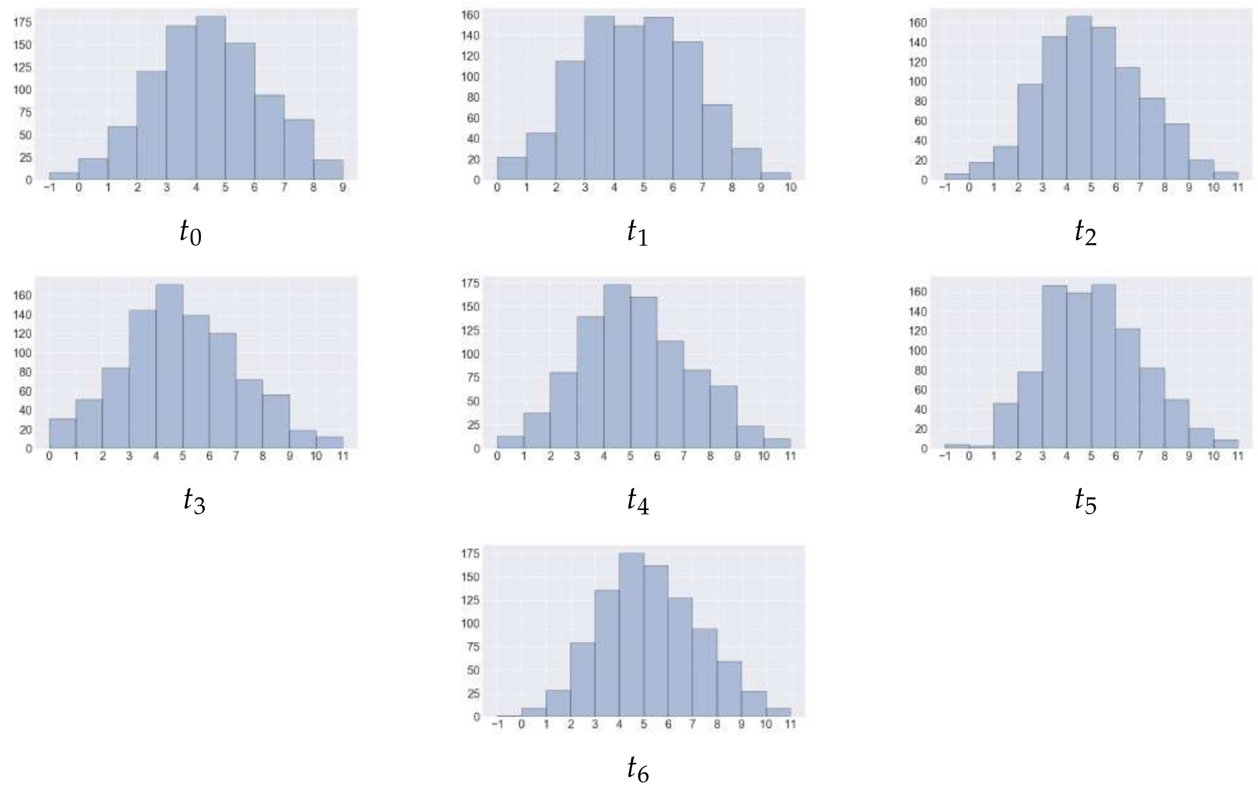


FIGURE 5.2: Distribution of $\log_2 RPKM$ values for RNA Crude Mitochondrial Genes

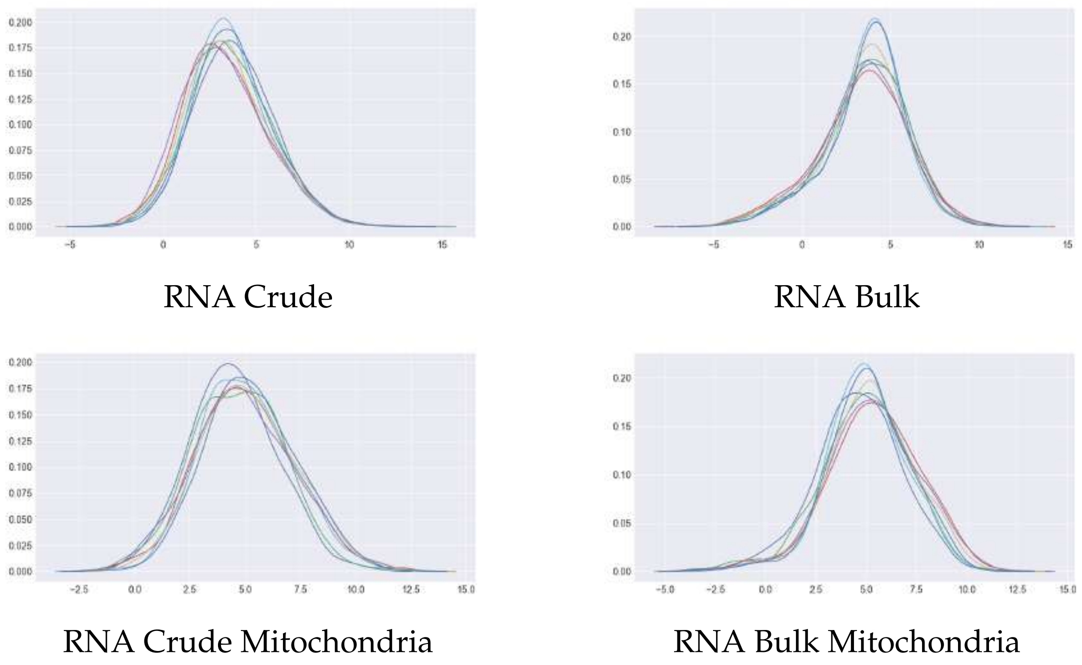


FIGURE 5.3: Comparison of the distribution between different datasets

5.2 STEM

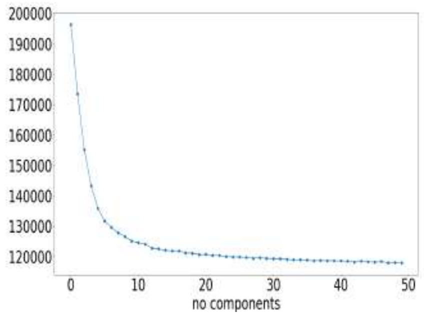
5.2.1 Results

Here, we use $m = 100$ and $c = 3$. Using different values for m and c do not give us significant differences for the clustering results. We use $\delta = 0.3$ as the minimum threshold to group similar significant clusters (refer to Section 4.1). Unfortunately, STEM excludes about 40% of the original pool of genes in its clustering. For RNA Crude, the number of genes considered in the significant clusters are 4894 genes (before cluster is 8675 genes). Similarly, for RNA Crude Mito, the number is 420 genes (before cluster is 911 genes). Due to this result, we decide that we only take the optimal number of clusters from STEM ($k = 19$ for RNA Crude and $k = 11$ for RNA Crude Mito) and then use this for the other algorithms. Moreover, we have also transformed the data into log ratios where the ratios are with respect to the expression of the first time point. We still use the original pools (8675 and 911 genes for Crude and Crude Mito, respectively) of genes for the other algorithms.

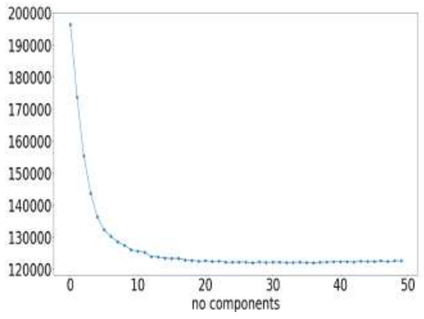
5.3 Mixture Model

5.3.1 Results

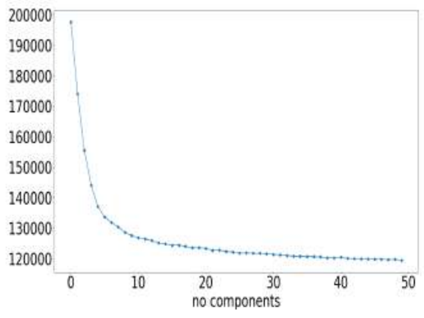
From the AIC and BIC analysis below (Figure 5.4), we are unable to determine the optimal number of clusters as it seems that the number of clusters keeps increasing whereas from STEM, we find that the number of clusters is actually not that high. Even when we perform this analysis with maximum number of components of > 100, we are still unable to find the optimum point. For mitochondrial genes which has at most 1000 data, AIC and BIC analysis still could not help us extract the optimal number of clusters. The reason is probably because of the nature of the data that is not rich enough to develop obvious separable clusters (refer to 3.1). Another possible reason is that the data simply cannot be modelled as a mixture of Gaussians. As such, we initialize the number of clusters for GMM obtained from STEM and then perform GO analysis for the two covariance matrices. We then perform GO analysis on each of the resulting clusters. Nonetheless, this result may correspond to the inability of STEM to cluster most of the genes (it only clusters $\approx$ 60% of the total genes) which means that indeed the data can be divided into many clusters.



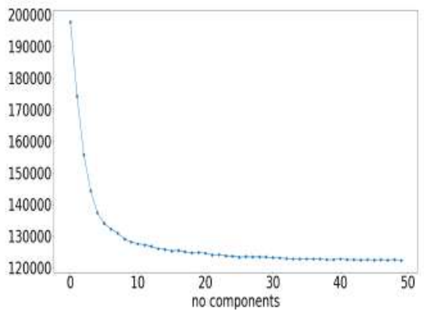
AIC Curve(Diagonal)



BIC Curve(Diagonal)



BIC Curve(Spherical)



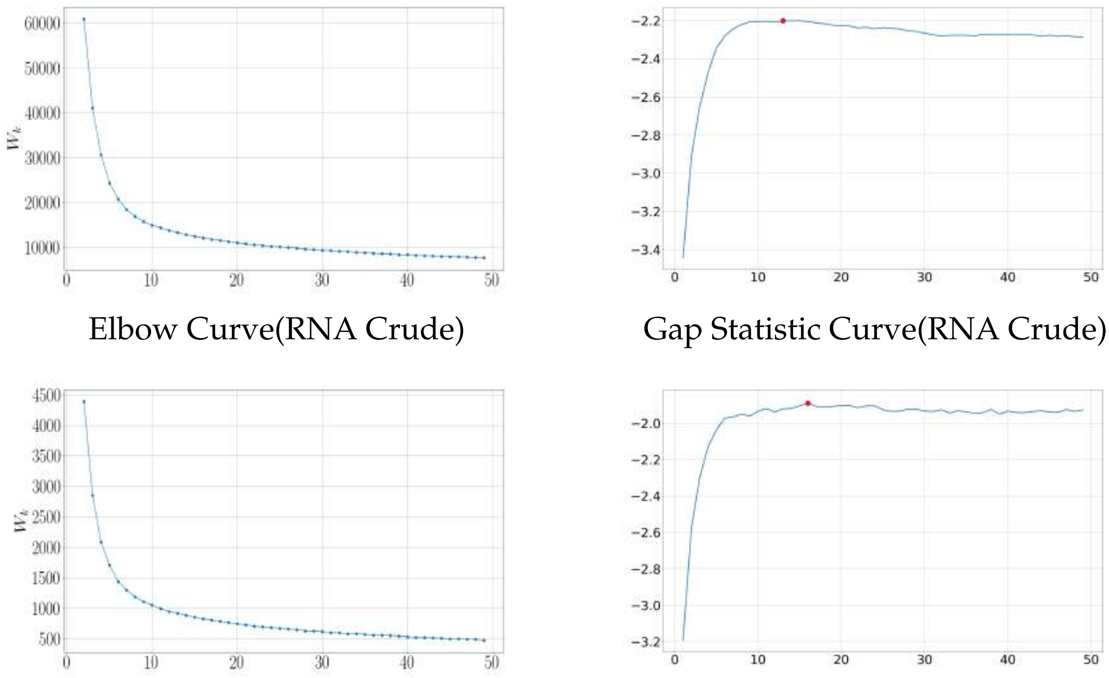
BIC Curve(Spherical)

FIGURE 5.4: AIC and BIC results for GMM with Diagonal and Spherical covariance matrices for RNA Crude data

5.4 K-means

5.4.1 Results

Similar to GMM, we are unable to determine the optimum number of clusters for k-means algorithm from both the elbow curve and gap statistics analysis (Figure 5.5). *K-means* may face the same problem like GMM that the data may not be rich enough to develop separable clusters for *k-means*. Another possible reasons is that euclidean distance may not be a good measure. As such, we will proceed with using optimal number of clusters obtained from STEM and the perform *k-means* algorithm to the data with this initial number of clusters. Then, we perform GO analysis on the resulting clusters.



Elbow Curve(RNA Crude) Gap Statistic Curve(RNA Crude)

Elbow Curve(RNA Crude Mito) Gap Statistic Curve(RNA Crude Mito)

FIGURE 5.5: Finding Optimal *k* for *k-means*

5.5 Hierarchical Clustering

5.5.1 Results

As mentioned in the previous chapter (Section 4.4), we will not be able to determine the optimal number of clusters from the dendrograms. Similar to the other algorithms, we will use *k* obtained from STEM. In this report, we will only show the plots for HAC with Complete linkage as it gives the best results (found in Appendix A).

5.6 Comparison of Algorithms

In this section, we use $k = 19$ and $k = 11$ for RNA Crude and RNA Crude Mito respectively when we apply the other clustering algorithms. From the results below (Table 5.1 and Table 5.2), we can see that the number of genes that fall into each cluster for different algorithms is actually quite similar (except for HAC (Comp)), especially when the total number of genes in the whole dataset is small (for RNA Crude Mito).

Cluster	**K-means**	**GMM (Sph)**	**GMM (Comp)**	**HAC (Comp)**
1	791	947	851	2123
2	756	925	814	978
3	751	897	805	872
4	678	720	712	832
5	666	619	667	756
6	658	554	636	475
7	625	544	523	456
8	573	480	503	430
9	532	457	489	362
10	492	455	473	357
11	469	410	408	188
12	460	355	376	181
13	324	346	371	173
14	276	243	334	110
15	227	225	300	104
16	178	212	175	97
17	96	122	96	89
18	69	100	82	47
19	35	45	41	45

TABLE 5.1: Gene count in each cluster for the different algorithms ($k = 19$), total number of genes = 8675

Cluster	K-means	GMM (Sph)	GMM (Diag)	HAC (Comp)
1	134	140	121	357
2	126	122	110	98
3	111	102	107	84
4	99	102	100	76
5	82	99	100	76
6	82	78	89	69
7	77	77	88	41
8	69	57	87	40
9	55	56	56	32
10	54	48	35	28
11	20	30	18	10

TABLE 5.2: Gene count in each cluster for the different algorithms ($k = 11$), total number of genes = 911

5.6.1 GO Analysis

In this section, we will only show the results for RNA Crude Mito. For each of the 11 clusters we get from the various algorithms, we perform GO analysis (we focus on *biological process* (bp), *cellular compartment* (cc) and *KEGG pathway* (KEGG)). We use *clusterprofiler* [26] to perform our GO analysis. Our GO analysis shows that majority of the clusters indeed contain genes that share similar biological properties. This proves the earlier proposition that genes with similar biological properties may be found in the same cluster in terms of their expression values. Then, we mostly only include GO terms that are related to mitochondrial biogenesis. Furthermore, this confirms our earlier belief that traditional clustering algorithms such as GMM, K-means and HAC can still be useful in clustering time-series data, even when it is short.

When we perform GO analysis on clusters from STEM algorithm($\approx$ 50% data) we do not get as many enriched clusters as we get from the other algorithms. This may show that

STEM removes important genes that should have been clustered into one of the significant clusters. Interestingly, some enriched GO terms appear in multiple clusters, such as **mitochondrial inner membrane, OXPHOS, mitochondrial gene expression**. This means that there may be interaction between clusters. Yet, it may also be expected that these are common mitochondrially-related GO terms. By analyzing these enriched GO terms with the genes involved in each GO term further such as by incorporating additional biological data, we may then be able to make better conclusion on our objective.

Here is a brief summary of the GO analysis for RNA Crude Mito:

- K-means
  - Cluster 3 (110 genes): cc (mitochondrial inner membrane, organelle inner membrane), KEGG (carbon metabolism, TCA cycle, OXPHOS)
  - Cluster 5 (81 genes): bp (mitochondrial transport, mitochondrial gene expression), cc (mitochondrial inner membrane, mitochondrial matrix), KEGG (ribosome, OXPHOS, Parkinson's disease)
  - Cluster 7 (76 genes): bp (cellular respiration, mitochondrial respiratory chain complex I assembly), cc (mitochondrial inner membrane, organelle inner membrane), KEGG (OXPHOS, Parkinson's disease, Alzheimer's disease)
  - Cluster 11 (19 genes): cc (inner mitochondrial membrane protein complex, Mitochondrial protein complext), KEGG (OXPHOS, Parkinson's disease, Thermogenesis)
- GMM (Diagonal)
  - Cluster 4 (99 genes): cc (mitochondrial inner membrane, organelle inner membrane, mitochondrial membrane), KEGG (OXPHOS, carbon metabolism, Huntington's diseases)
  - Cluster 5 (99 genes): bp (mitochondrial transport, mitochondrial gene expression), cc (mitochondrial matrix, mitochondrial inner membrane, organnelle inner membrane), KEGG (OXPHOS, Thermogenesis)

  - Cluster 6 (88 genes)8: cc (mitochondrial inner membrane, organelle inner membrane, mitochodrial protein complex), KEGG (OXPHOS, Parkinson's disease, Alzheimers' disease)
  - Cluster 8 (86 genes): bp (mitochondrial transport, mitochondrial gene expression), cc (mitochondrial matrix, mitochondrial inner membrane)
  - Cluster 10 (34 genes): cc (inner mitochondrial membrane, mitochondrial protein complex), KEGG (OXPHOS, Parkinson's disease, Alzheimers' disease)

- GMM (Spherical)
  - Cluster 3 (101 genes): cc (mitochondrial inner membrane, organelle inner membrane), KEGG (OXPHOS, carbon metabolism, Huntington's disease, Parkinson's disease)
  - Cluster 5 (98 genes): bp (mitochondrial transport, mitochondrial gene expression), cc (mitochondrial matrix, organelle ribosome, mitochondrial inner membrane), KEGG (ribosome, Huntington's disease)
  - Cluster 7 (76 genes): cc (mitochondrial inner membrane, mitochondrial protein complex), KEGG (OXPHOS, Parkinson's disease, Alzheimer's disease)
  - Cluster 10 (47 genes): cc (mitochondrial protein complex, organelle inner membrane), KEGG (OXPHOS, Parkinson's disease, Alzheimer's disease)

- HAC (Complete)
  - Cluster 1 (357 genes): cc (mitochondrial inner membrane, organelle inner membrane), KEGG (OXPHOS, thermogenesis, carbon metabolism)
  - Cluster 2 (98 genes): bp (purine ribonucleoside triphosphate metabolic process, ribonucleoside triphosphate metabolic process), cc (organelle inner membrane, mitochondrial inner membrane, mitochondrial matrix), KEGG (Parkinson's disease, OXPHOS)

– Cluster 3 (84 genes): bp (mitochondrial RNA metabolic process, mitochondrial respiratory chain complex assembly), cc (mitochondrial matrix, mitochondrial inner membrane)

– Cluster 4 (76 genes): bp (mitochondrial respiratory chain complex assembly, cellular respiratory), cc (mitochondrial inner membrane, organelle inner membrane), KEGG (OXPHOS, thermogenesis)

Similar enriched GO terms and other terms also appear when we perform GO analysis for RNA Crude (results are exluded). The cluster plots for the above summary can be found below:

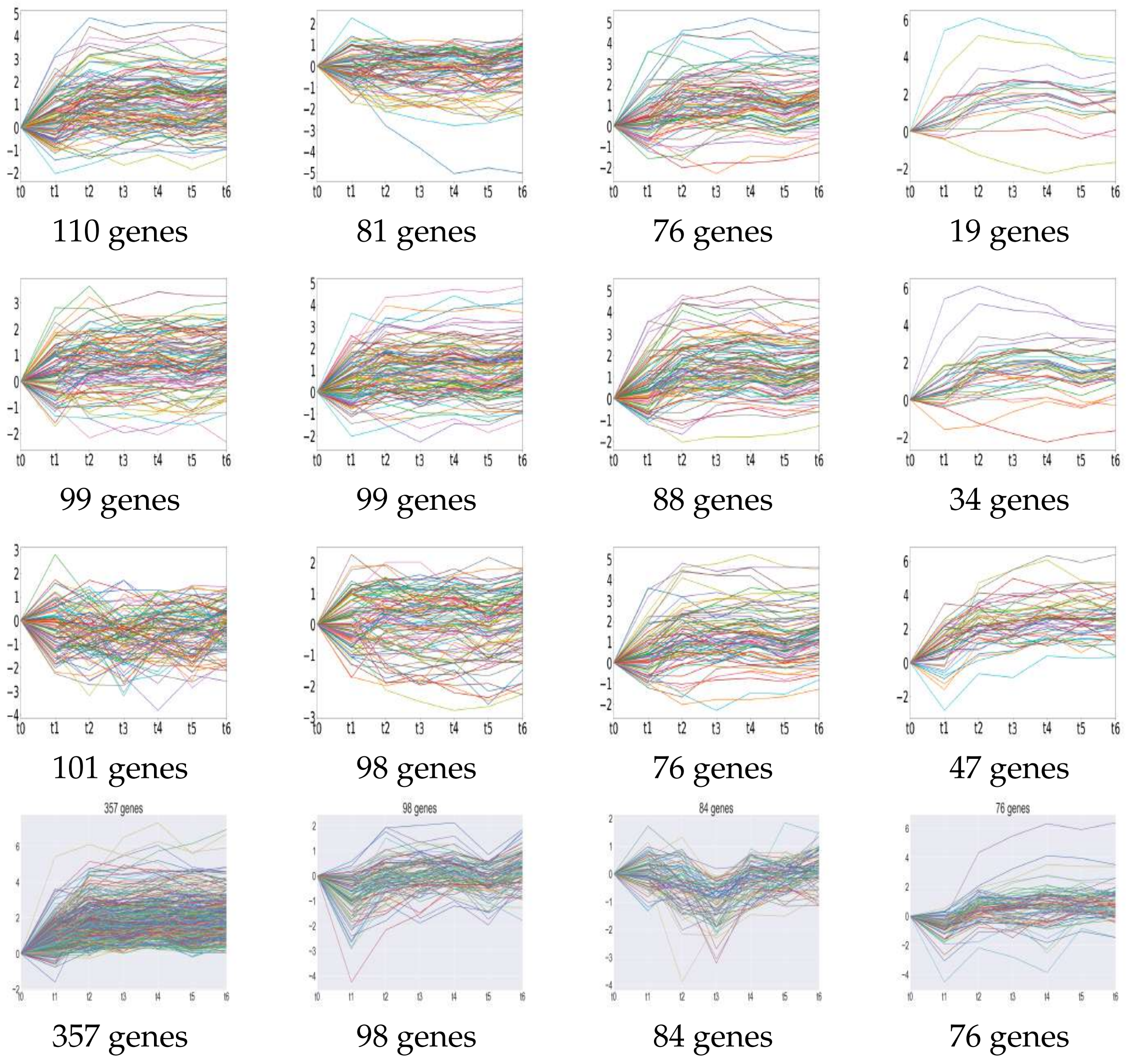


FIGURE 5.6: Plots comparison for the 4 enriched clusters
In order of appearance: K-means, GMM (Diagonal), GMM (Spherical), HAC (Ward)

5.7 Performance Evaluation

In this section, we compare the performances of the different clustering algorithm with the assumption that STEM gives the optimal clustering. Here, we cannot use the whole pool of genes as STEM does not consider about half of these genes to be significant. Thus, we pick the genes that are considered significant by STEM and then apply the other clustering algorithms to these sets of genes. Then, we compare the performances of each clustering algorithm with using the two metrics described earlier (*clustering accuracy* and *ME distance*, refer to Section 3.3).

	RNA Bulk		**RNA Bulk (Mito)**		**RNA Crude**		**RNA Crude (Mito)**	
Algorithm	**CA**	**ME**	**CA**	**ME**	**CA**	**ME**	**CA**	**ME**
K-means	56.37	68.17	66.77	59.91	48.39	72.60	45.95	68.81
GMM (Full)	50.87	71.41	59.15	58.84	51.70	71.00	53.81	64.76
GMM (Diag)	52.79	72.31	68.90	59.15	44.34	76.24	42.86	71.67
GMM (Sph)	57.23	70.55	66.92	60.52	53.11	72.00	48.10	67.86
HAC (Ward)	53.17	70.75	63.41	60.52	47.10	73.03	43.10	71.43
HAC (Comp)	71.68	49.61	78.51	31.55	67.94	49.94	71.43	44.76
HAC (Ave)	66.40	38.88	73.48	33.84	64.73	44.79	61.43	44.76

TABLE 5.3: Performance evaluation for the different algorithms (in %), the underlined values are the best performing algorithm for that particular class of clustering

From the Table 5.3 above, we can see that there is no algorithm that outperforms the rest for all the datasets. As expected, HAC with Complete and Average as the linkage outperforms the other algorithms since they use correlation coefficient as the distance metric just like STEM. Another interesting result is that some of the algorithms perform worse for RNA Crude (Mito) when the size of the data is smaller.

6 Conclusion

6.1 Summary

6.1.1 STEM To Determine Number of Clusters

We pick STEM as the algorithm that provides the optimum clustering due to a few reasons:

- Only STEM that can give us optimal number of clusters from the statistical test
- STEM takes into account the sequential nature of time-series data
- STEM generates profiles that are independent of the data
- Profiles generated by STEM tells us the dynamic between successive time points

Nonetheless, since STEM also computes the statistical significance of each cluster, it can exclude many genes which could have been clustered. This omits the possibility of having biologically significant clusters yet not statistically significant clusters as shown in the GO analysis for the other algorithms. Hence, our proposal is that to perform STEM and then extract the optimal number of clusters (k) from it. Thereafter, we use k as the initial input for other clustering algorithms and perform GO analysis on the resulting clusters. On the other hand, if the removal of many genes corresponds to removal of noise, STEM may then be used to filter out noise.

6.1.2 No Algorithm Performs Better For All Data

Our results show that no algorithm performs better for all situations. From all the clusters generated by each algorithm, there are always a mix between biologically and statistically meaningful clusters and those that are not. It is also difficult to conclude on what kind of situation or data where one algorithm can perform better than rest. We may need to further preprocess the data to see whether it can improve the performance of each algorithm. As expected, Hierarchical Clustering with Complete and Average linkage performs better than the other algorithms.

6.2 Further Work

6.2.1 Different Algorithms

There are still plenty of clustering algorithms out there that can be used to cluster short time-series data. One possible future plan is to improve the algorithms that we have used previously, such as by incorporating additional data or prior; adjusting the parameters. Here we mention briefly some algorithms that have been improved or modified to cluster short time-series gene expression data:

Explore On Soft Clustering

All the algorithms above are considered hard clustering, which means one gene cannot belong to more than one cluster. In reality, one gene can belong to multiple biological groups and thus, may belong to more than one cluster. For this aspect, we can perform Fuzzy-clustering algorithm, a popular soft clustering algorithm. Fuzzy-clustering, modified for short time-series [27] has also been used to study the dynamic of gene expression values with short time points.

Bayesian Approach

We can also consider to study variational Bayesian estimation of Gaussian Mixture model [16] that may improve the performance of GMM. We implement the technique of variational inference, which is an extension of expectation-maximization that maximizes a lower bound on model evidence (including priors) instead of data likelihood. Due to its Bayesian nature, this model allows us to specify a concentration parameter: weight concentration prior, which helps affect the number of components in the model. Furthermore, high concentration parameter not only leads to more active components in the mixture, but also more uniform weights, which may be useful.

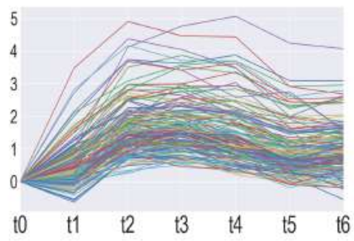
6.2.2 Interaction Between Clusters

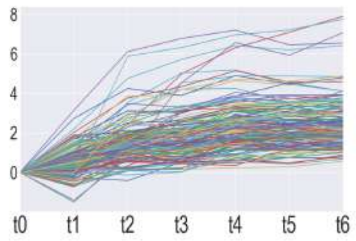
As mentioned in Section 5.6, there may be interaction between clusters which will give more insights on our clustering. One method that has been used is an improvement of HAC, *Fast Optimal Leaf Ordering For Hierarchical Clustering* [28]. Optimal leaf ordering causes input elements that are highly correlated in a cluster to appear in the middle of the linear ordering of the cluster, while marginally related elements are on the borders of the cluster. This improvement reorders the leaves in HAC which gives a more understandable visualizations and hence, helps reveal more biological structures. These relationships are very important in time series data analysis.

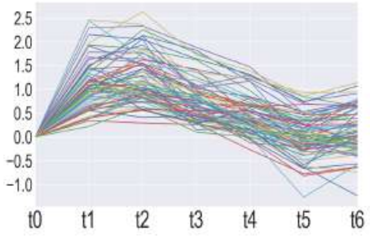
6.2.3 Other Distance Measures

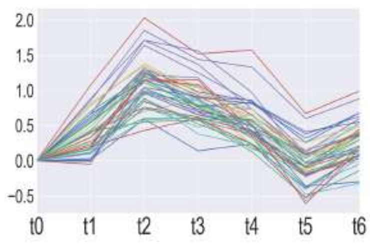
There are also plenty of distance or similarity measures that can be used when we want to cluster time-series data such dynamic time warping distance, short time-series distance developed for Fuzzy-clustering [27] and many other.

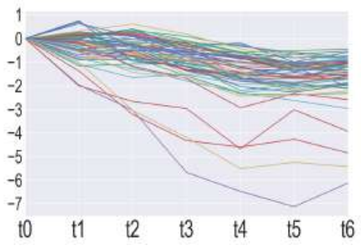
A Plots For RNA Bulk Mito

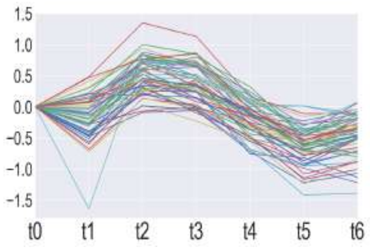

(1) 157 genes

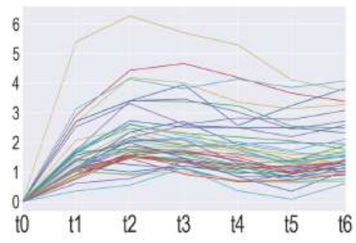

(2) 155 genes

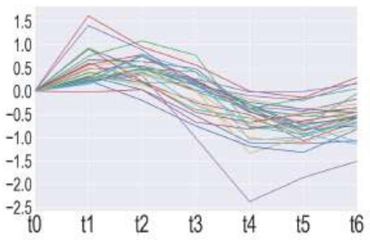

(3) 71 genes

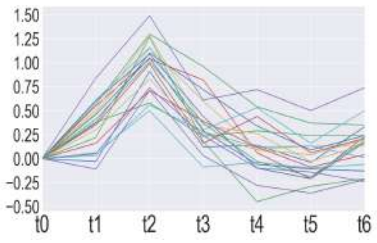

(4) 44 genes


(5) 61 genes


(6) 57 genes


(7) 42 genes


(8) 30 genes


(9) 20 genes

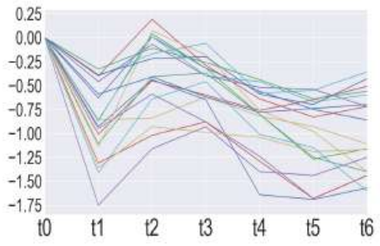
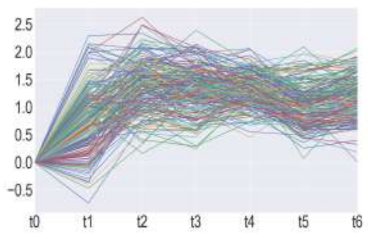
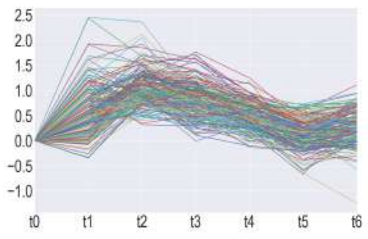
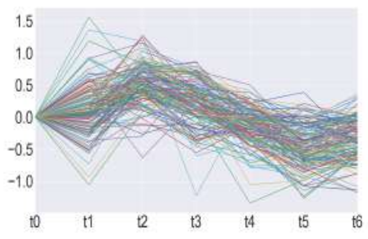
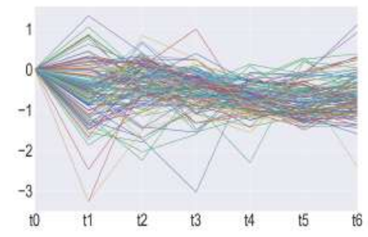

(10) 19 genes

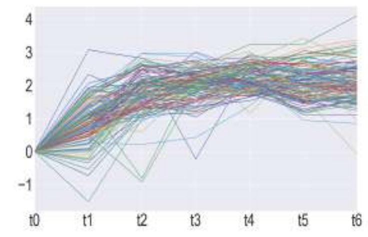
FIGURE A.1: Clustering Results for STEM

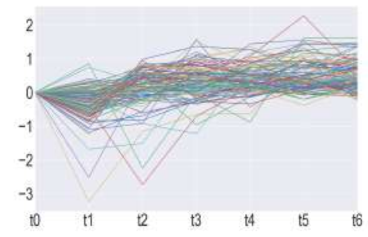

(1) 196 genes

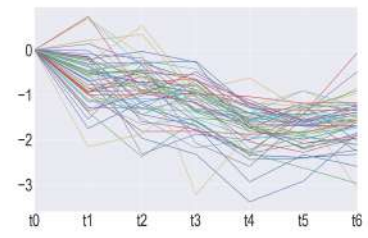

(2) 177 genes

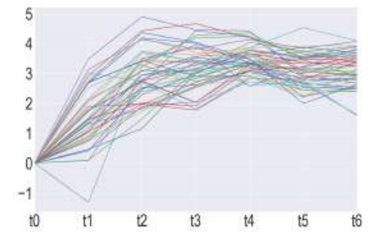

(3) 145 genes

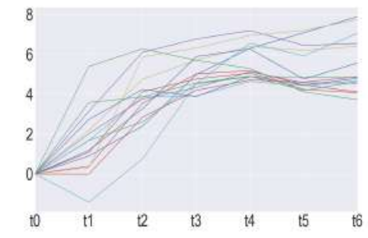

(4) 132 genes


(5) 124 genes


(6) 108 genes


(7) 54 genes


(8) 46 genes


(9) 15 genes

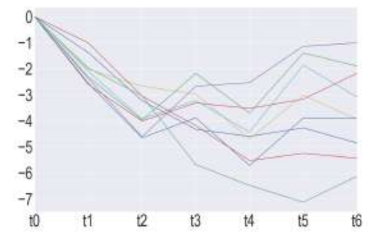
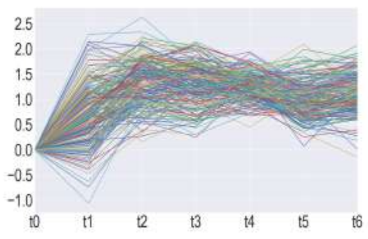
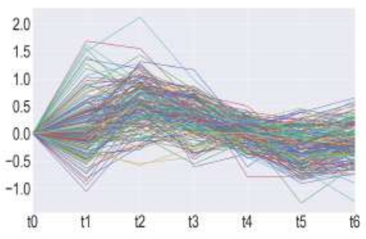
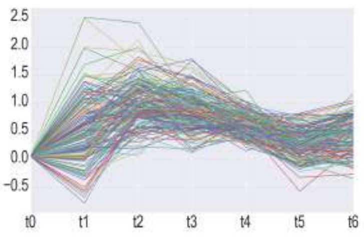
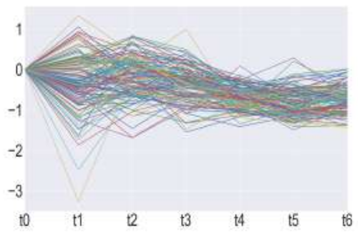

(10) 9 genes

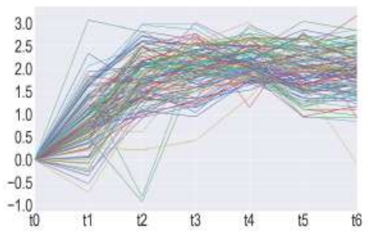
FIGURE A.2: Clustering results for k-means

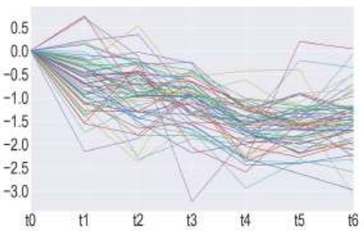

(1) 192 genes

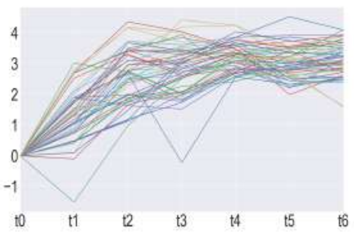

(2) 187 genes

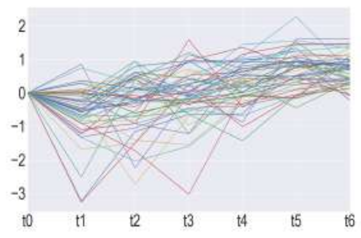

(3) 172 genes

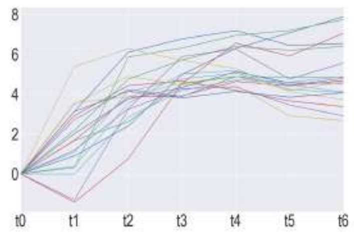

(4) 137 genes


(5) 124 genes


(6) 61 genes


(7) 53 genes


(8) 51 genes


(9) 19 genes

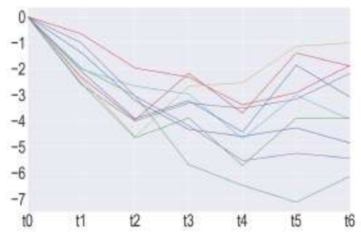
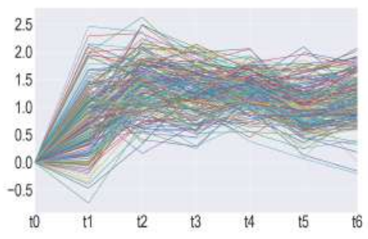
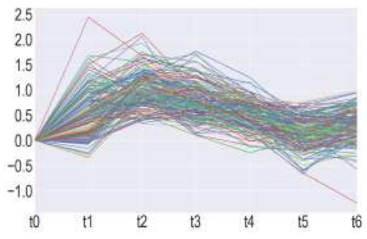
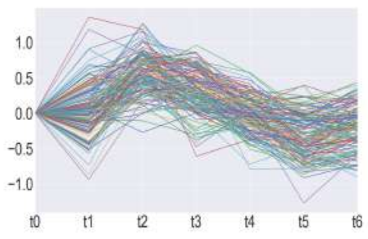
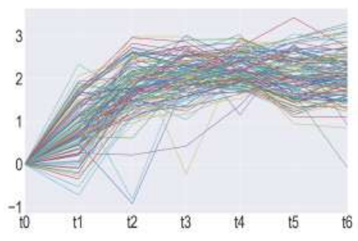

(10) 10 genes

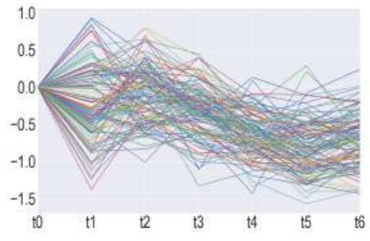
FIGURE A.3: Clustering results for GMM(Diagonal)

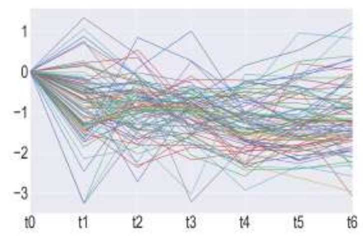

(1) 205 genes

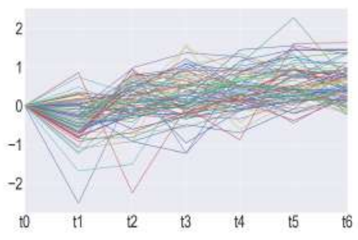

(2) 172 genes

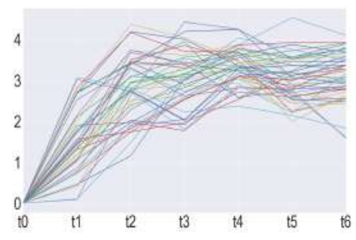

(3) 144 genes

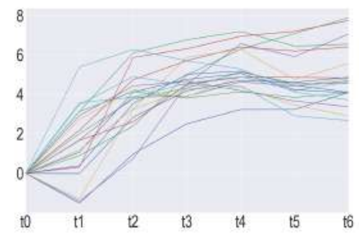

(4) 121 genes


(5) 120 genes


(6) 88 genes


(7) 83 genes


(8) 43 genes


(9) 20 genes

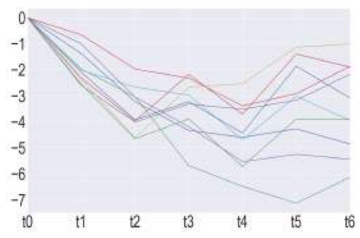
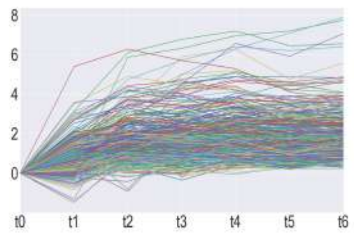
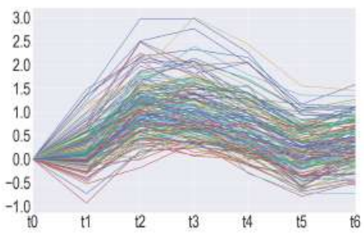
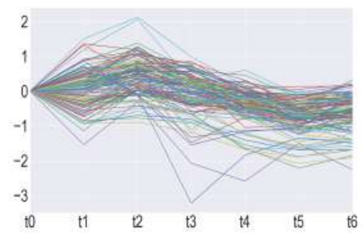
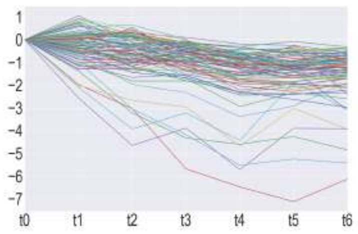

(10) 10 genes

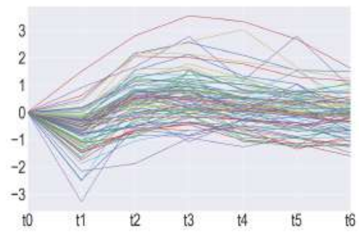
FIGURE A.4: Clustering results for GMM(Spherical)

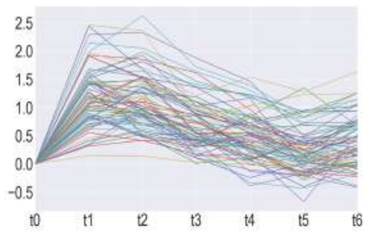

(1) 361 genes

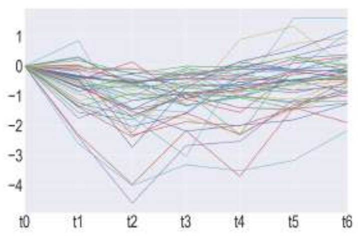

(2) 175 genes

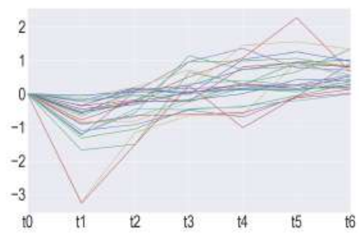

(3) 132 genes

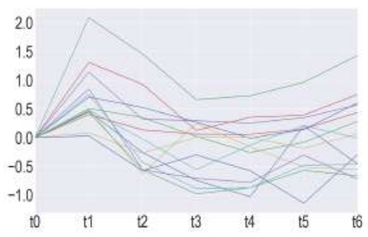

(4) 87 genes


(5) 81 genes


(6) 73 genes


(7) 47 genes


(8) 27 genes


(9) 14 genes

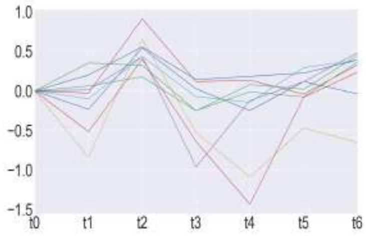
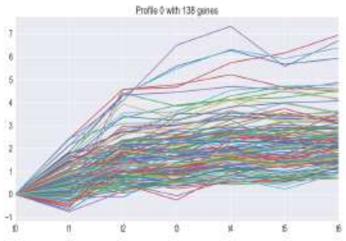
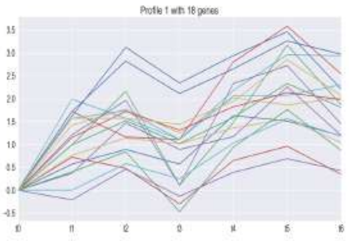

(10) 9 genes

FIGURE A.5: Clustering results for HAC (Complete)

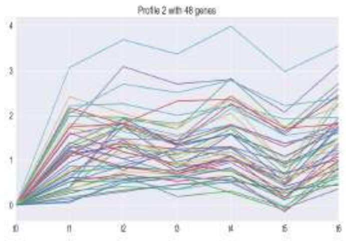
B Plots For RNA Crude Mito



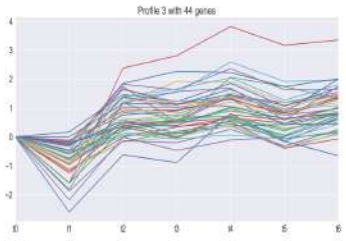
(1) 137 genes



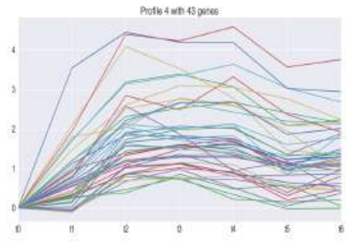
(2) 17 genes



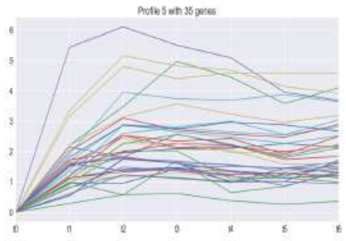
(3) 47 genes



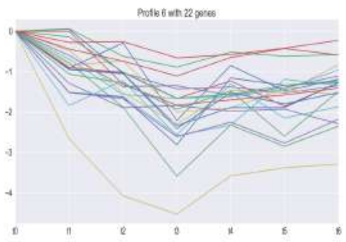
(4) 43 genes



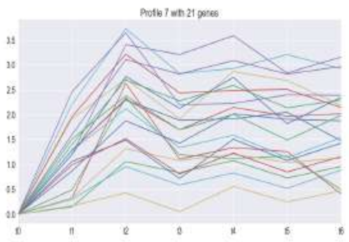
(5) 42 genes



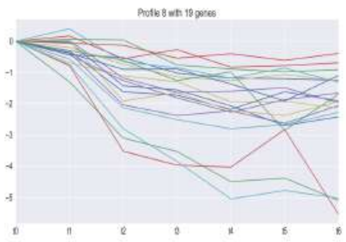
(6) 34 genes



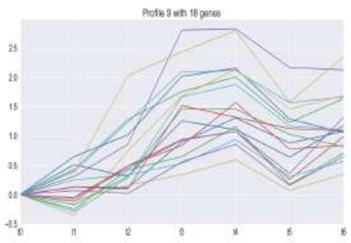
(7) 21 genes



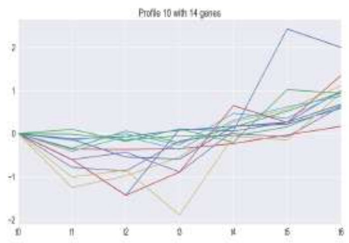
(8) 20 genes



(9) 8 genes

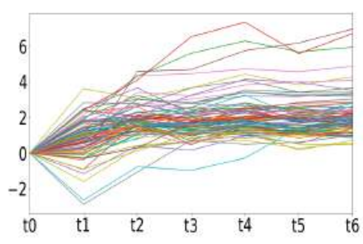


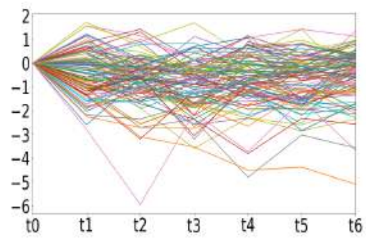
(10) 17 genes

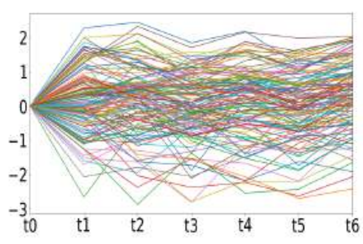


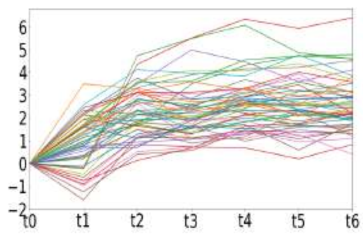
(11) 13 genes

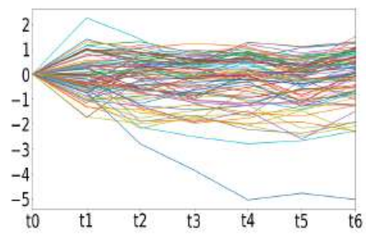
FIGURE B.1: Clustering results for STEM

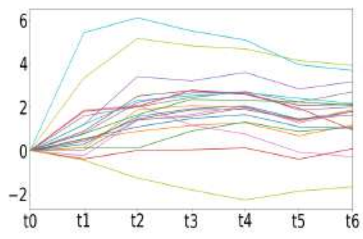

(1) 81 genes

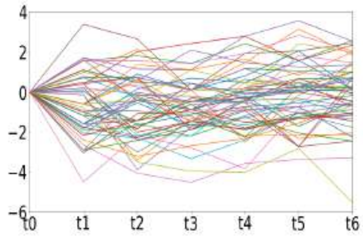

(2) 98 genes

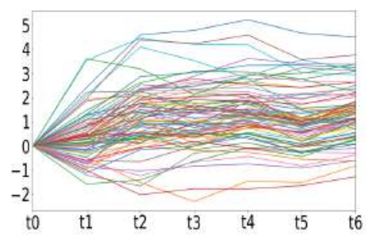

(3) 133 genes

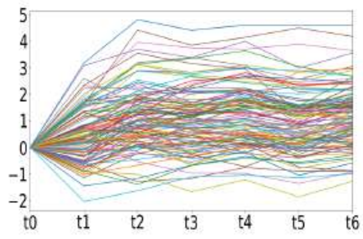

(4) 53 genes

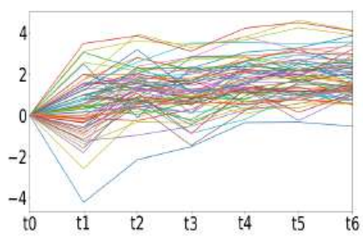

(5) 81 genes


(6) 19 genes


(7) 56 genes


(8) 76 genes


(9) 110 genes


(10) 68 genes

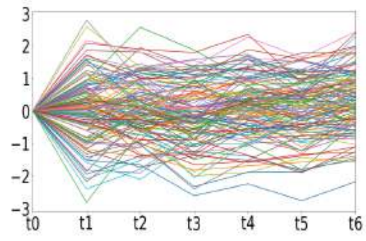
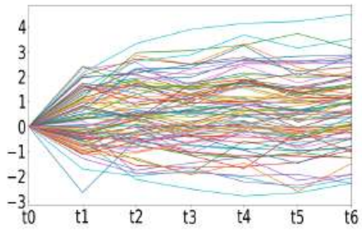
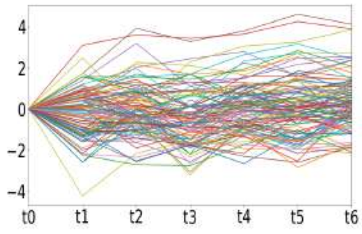
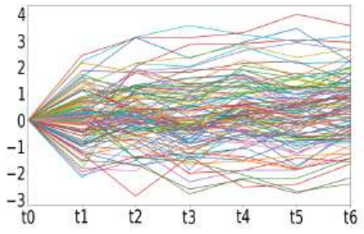
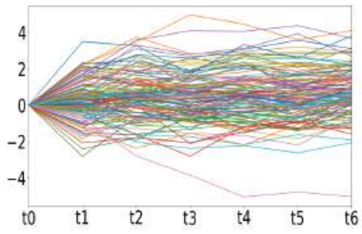

(11) 125 genes

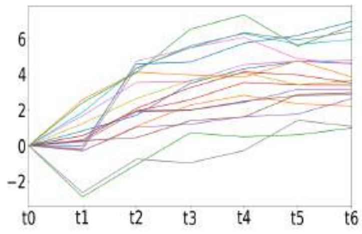
FIGURE B.2: Clustering results for K-means

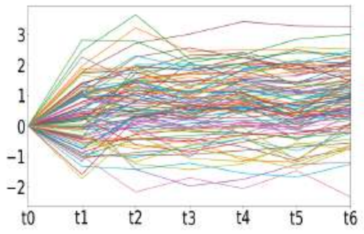

(1) 86 genes

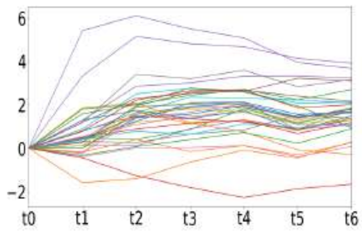

(2) 106 genes

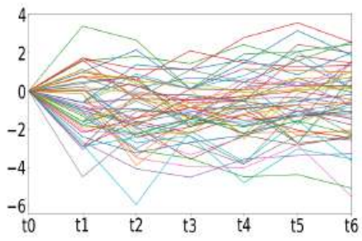

(3) 109 genes

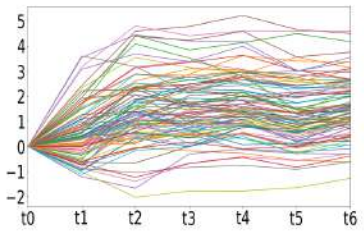

(4) 120 genes

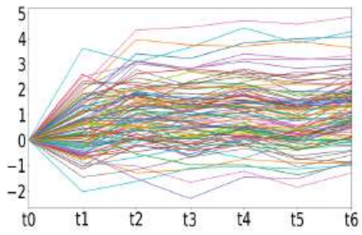

(5) 17 genes


(6) 99 genes


(7) 56 genes


(8) 89 genes


(9) 100 genes


(10) 99 genes

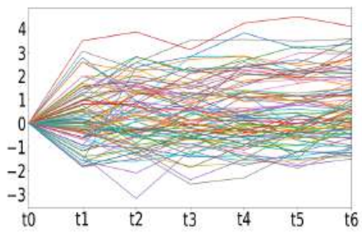
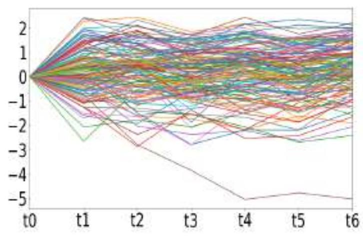
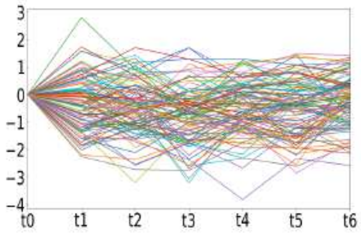
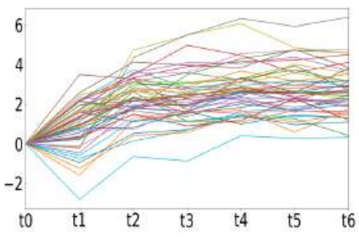
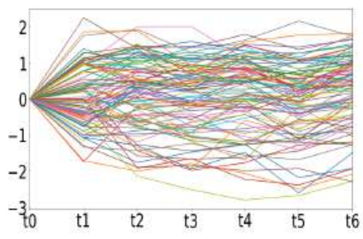

(11) 87 genes

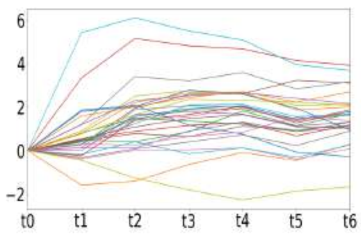
FIGURE B.3: Clustering results for GMM (Diagonal)

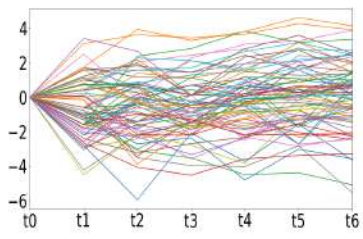

(1) 140 genes

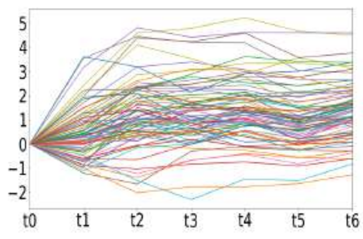

(2) 102 genes

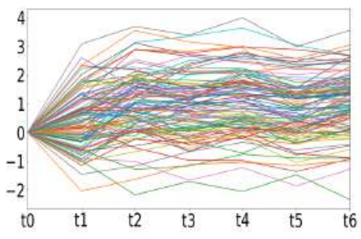

(3) 48 genes

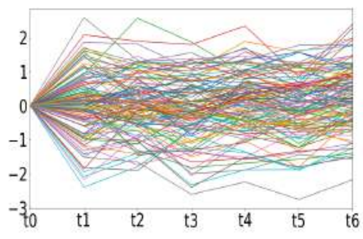

(4) 98 genes

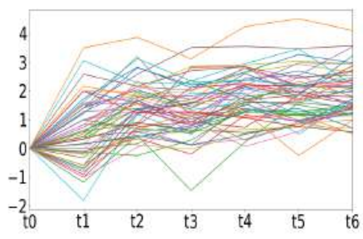

(5) 29 genes


(6) 77 genes


(7) 76 genes


(8) 101 genes


(9) 121 genes


(10) 56 genes

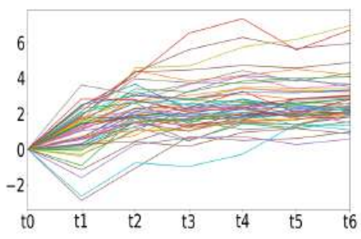
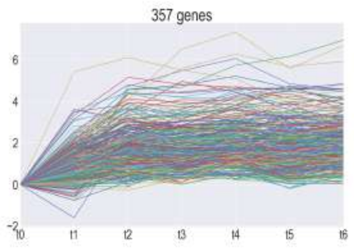
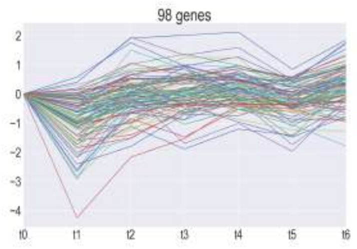

(11) 55 genes

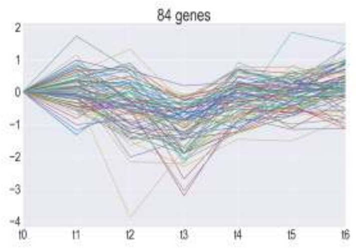
FIGURE B.4: Clustering results for GMM (Spherical)



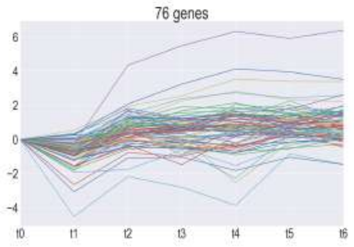
(1) 357 genes



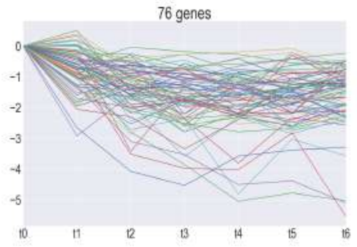
(2) 98 genes



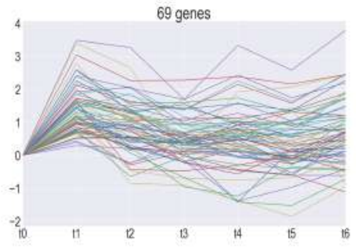
(3) 84 genes



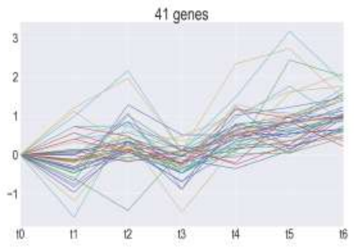
(4) 76 genes



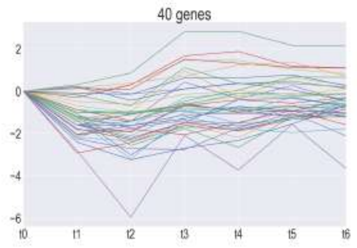
(5) 76 genes



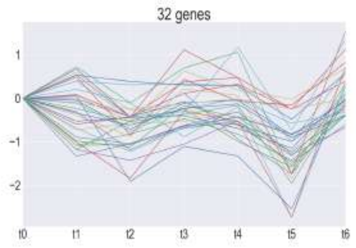
(6) 69 genes



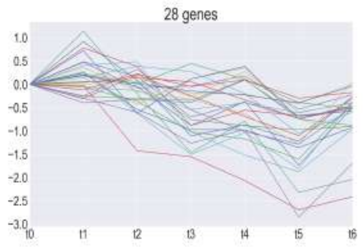
(7) 41 genes



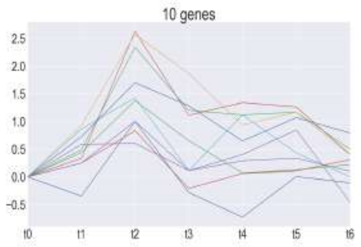
(8) 40 genes



(9) 32 genes



(10) 28 genes



(11) 10 genes

FIGURE B.5: Clustering results for HAC (Complete)

Bibliography

[1] Maxwell W. Libbrecht and William Stafford Noble. “Machine learning applications in genetics and genomics”. In: *Nature Reviews Genetics* 16.6 (2015), pp. 321–332. ISSN: 1471-0056. DOI: 10.1038/nrg3920. URL: http://www.nature.com/articles/nrg3920.

[2] Bruce Alberts et al. “The Genetic Systems of Mitochondria and Plastids”. In: (2002). URL: https://www.ncbi.nlm.nih.gov/books/NBK26924/.

[3] Mary T Couvillion et al. “Synchronized translation programs across compartments during mitochondrial biogenesis”. In: 533.7604 (2016), pp. 499–503. ISSN: 1527-5418. DOI: 10.1038/nature18015. arXiv: 15334406. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4964289/pdf/nihms-779824.pdf.

[4] Michael B. Eisen et al. “Cluster analysis and display of genome-wide expression patterns”. In: *Proceedings of the National Academy of Sciences* 95.25 (1998), pp. 14863–14868. ISSN: 0027-8424. eprint: http://www.pnas.org/content/95/25/14863.full.pdf. URL: http://www.pnas.org/content/95/25/14863.

[5] Jason Ernst, Gerard J. Nau, and Ziv Bar-Joseph. “Clustering short time series gene expression data”. In: *Bioinformatics* (2005). DOI: 10.1093/bioinformatics/bti1022. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1456994/pdf/1471-2105-7-191.pdf.

[6] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. A Wiley Interscience Publication. Wiley, 1973.

[7] Emil Eirola and Amaury Lendasse. “Gaussian Mixture Models for Time Series Modelling, Forecasting, and Interpolation”. In: *Advances in Intelligent Data Analysis XII*. Ed. by Allan Tucker et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 162–173. ISBN: 978-3-642-41398-8.

[8] Marco F. Ramoni, Paola Sebastiani, and Isaac S. Kohane. “Cluster analysis of gene expression dynamics”. In: *Proceedings of the National Academy of Sciences* 99.14 (2002), pp. 9121–9126. ISSN: 0027-8424. DOI: 10.1073/pnas.132656399. eprint: http://www.pnas.org/content/99/14/9121.full.pdf. URL: http://www.pnas.org/content/99/14/9121.

[9] Ziv Bar-Joseph et al. “Comparing the continuous representation of time-series expression profiles to identify differentially expressed genes”. In: *Proceedings of the National Academy of Sciences* 100.18 (2003), pp. 10146–10151. ISSN: 0027-8424. DOI: 10.1073/pnas.1732547100. eprint: http://www.pnas.org/content/100/18/10146.full.pdf. URL: http://www.pnas.org/content/100/18/10146.

[10] Lawrence R. Rabiner. “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition”. In: *Proceedings of the IEEE* (1989). ISSN: 15582256. DOI: 10.1109/5.18626. arXiv: arXiv:1011.1669v3.

[11] Alexander Schliep, Alexander Schönhuth, and Christine Steinhoff. “Using hidden Markov models to analyze gene expression time course data”. In: *Bioinformatics* 19.SUPPL. 1 (2003). ISSN: 13674803. DOI: 10.1093/bioinformatics/btg1036.

[12] Fei Wang and Changshui Zhang. “Spectral Clustering for Time Series”. In: *Pattern Recognition and Data Mining*. Ed. by Sameer Singh et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 345–354. ISBN: 978-3-540-28758-2.

[13] Ping Ma et al. “A data-driven clustering method for time course gene expression data”. In: *Nucleic Acids Research* (2006). ISSN: 03051048. DOI: 10.1093/nar/gkl013.

[14] Monica Golumbeanu and Niko Beerenwinkel. “Clustering time series gene expression data with TMixClust”. In: (2017).

[15] Yafeng Zhang et al. “Comparison of Clustering Methods for Time Course Genomic Data: Applications to Aging Effects”. In: (2014). arXiv: arXiv:1404.7534v1.

[16] Zhenyu Jia and Shizhong Xu. "Bayesian mixture model analysis for detecting differentially expressed genes". In: *International Journal of Plant Genomics* 2008 (2008). ISSN: 16875370. DOI: 10.1155/2008/892927.

[17] Maya R. Gupta and Yihua Chen. "Theory and Use of the EM Algorithm". In: *Foundations and Trends in Signal Processing* 4.3 (2011), pp. 223–296. ISSN: 1932-8346. DOI: 10.1561/2000000034. URL: http://dx.doi.org/10.1561/2000000034.

[18] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Vol. 53. 9. 2013, pp. 1689–1699. ISBN: 978-0-387-31073-2. DOI: 10.1117/1.2819119. arXiv: arXiv:1011.1669v3.

[19] Hirotogu Akaike. "Information Theory and an Extension of the Maximum Likelihood Principle". In: *Selected Papers of Hirotugu Akaike*. Ed. by Emanuel Parzen, Kunio Tanabe, and Genshiro Kitagawa. New York, NY: Springer New York, 1998, pp. 199–213. ISBN: 978-1-4612-1694-0. DOI: 10.1007/978-1-4612-1694-0_15. URL: https://doi.org/10.1007/978-1-4612-1694-0_15.

[20] Gideon Schwarz. "Estimating the Dimension of a Model". In: *The Annals of Statistics* 6.2 (1978), pp. 461–464. ISSN: 0090-5364. DOI: 10.1214/aos/1176344136. arXiv: arXiv:1011.1669v3. URL: http://projecteuclid.org/euclid.aos/1176344136.

[21] Robert Tibshirani, Guenther Walther, and Trevor Hastie. *Estimating the number of clusters in a data set via the gap statistic*. 2001. DOI: 10.1111/1467-9868.00293. URL: http://doi.wiley.com/10.1111/1467-9868.00293.

[22] Marina Meila. "The uniqueness of a good optimum for k-means". In: *ACM INTERNATIONAL CONFERENCE PROCEEDING SERIES* 148 (), p. 2006. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.78.2239.

[23] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1982. ISBN: 0-13-152462-3.

[24] Julia Sivriver, Naomi Habib, and Nir Friedman. “An integrative clustering and modeling algorithm for dynamical gene expression data”. In: *Bioinformatics* 27.13 (2011), pp. 392–400. ISSN: 13674803. DOI: 10.1093/bioinformatics/btr250.

[25] David Arthur and Sergei Vassilvitskii. “K-Means++: the Advantages of Careful Seeding”. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* 8 (2007), pp. 1027–1025. ISSN: 978-0-898716-24-5. DOI: 10.1145/1283383.1283494. arXiv: 1212.1121.

[26] Guangchuang Yu et al. “clusterProfiler: an R Package for Comparing Biological Themes Among Gene Clusters”. In: *OMICS: A Journal of Integrative Biology* 16.5 (2012). PMID: 22455463, pp. 284–287. DOI: 10.1089/omi.2011.0118. eprint: https://doi.org/10.1089/omi.2011.0118. URL: https://doi.org/10.1089/omi.2011.0118.

[27] Carla S. Möller-Levet et al. “Fuzzy Clustering of Short Time-Series and Unevenly Distributed Sampling Points”. In: 2003. ISBN: 978-3-540-40813-0. DOI: 10.1007/978-3-540-45231-7_31.

[28] Ziv Bar-Joseph, David K. Gifford, and Tommi S. Jaakkola. “Fast optimal leaf ordering for hierarchical clustering”. In: *Bioinformatics* 17.suppl$_1$ (2001), S22–S29. DOI: 10.1093/bioinformatics/17.suppl_1.S22. eprint: /oup/backfile/content_public/journal/bioinformatics/17/suppl_1/10.1093/bioinformatics/17.suppl_1.s22/2/17s022.pdf. URL: +http://dx.doi.org/10.1093/bioinformatics/17.suppl_1.S22.