Clustering and Generative Modelling

Ngan Thi Le

Submitted for the Degree of Master of Science in

Computational Finance
with a year in Industry

Department of Computer Science
Royal Holloway University of London
Egham, Surrey TW20 0EX, UK

June 16, 2014

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 13166

Student Name: Ngan Thi Le

Date of Submission: 14 September 2015

Signature: NganLe

Abstract

Cluster Analysis or Clustering is an important practice in data analysis that helps us understand the structure or pattern of unlabelled data. It aims to partition a collection of objects into smaller distinct clusters, where objects in the same groups are more closely related to one another than those assigned to different groups. Many algorithms have been developed and proposed to achieve this task. In this report, three main algorithms: K-means, Hierarchical clustering and Mixture of Gaussians were implemented and compared. And to optimise the clustering performance, I also consider, define and apply different techniques. In particular, the research was conducted on selecting initialisation parameters, (dis)similarity measure, number of clusters and cluster validation methods. Toward the conclusion, the project aims to recommend the right clustering techniques to get the best insight of data.

Table of Contents

1 Introduction

While Classification is predicting labels for new observations according to existing labelled observations (supervised learning), Cluster Analysis identifies the structure of data without the observations' label (unsupervised learning). Generally, the unlabelled data set could be inhomogeneous. Its structure and characteristics are difficult to understand. Clustering the data into homogeneous groups allows us to easily construct a statistical model of the data set, or explain the relationship between observations or attributes.

The ease of constructing statistical models is one of the reasons why clustering has become one of the most important practices in data analysis, and is widely applied across different industries such as biology, business, marketing, social networking, data mining and others.

Hastie et al. [2] identify cluster analysis as the process of partitioning a collection of objects into smaller clusters, where objects in the same group are more closely related to one another than those assigned to different groups. In other words, it aims to achieve high intra-cluster and low inter-cluster similarity. Varieties of clustering algorithms have been developed in order to solve un-known data structures, as different clusters can discover different characteristics of data while accomplishing the task.

As mentioned in the Unsupervised Learning lecture by Zhiyuan Lou, cluster analysis is divided into two types, parametric clustering and non-parametric clustering. For non-parametric clustering, there is no need to define any underlying density function, because it is more concerned with discovering the natural structure of the data set. For parametric clustering, an underlying density function such as Gaussian distribution should be defined in advanced. Each of these methods is further broken down into hierarchical (tree-based) or partitioning methods.

Specifically, agglomeration/divisive hierarchical clustering algorithms are well-known non-parametric hierarchical methods. Manpreet [25] summarised that **hierarchical clustering** produces a sequence of clustering, in which each cluster is nested into the next cluster, and presents the result on a binary tree called a dendrogram. Following this, the dendrogram can be cut at any point to achieve the desired number of clusters.

K-means clustering is a basic and popular non-parametric partitioning method. This algorithm partitions the data set into a set of a predefined number of clusters. Each object is assigned to the closest cluster, which is represented by its centroid.

Mixture of Gaussians is an important parametric partitioning method, and a probabilistic model. It comprises a predefined number of Gaussian density components, where each component corresponds to a cluster. The identity of the

mixture component of each observation is specified by a set of latent (hidden) variables.

Zhiyuan [24] emphasised three main steps in clustering. First of all, we need to define a measure of (dis)similarity, including distance and similarity, between observations. Euclidean, Manhattan distance and Pearson correlation measure are discussed in this report. Secondly, an evaluation criterion for clustering is conducted. The internal validation is common for clustering because it is based on the information intrinsic to the data alone. However, if there is any previous knowledge about the data, external validation can be applied. Finally, an algorithm to minimize or maximize the evaluation function is chosen. This report will demonstrate and compare K-means, Hierarchical and Mixture of Gaussian clustering algorithms. Besides, selecting the most accurate number of clusters and their initialisation parameters is extremely important if K-means or Mixture of Gaussian method is chosen.

In order to achieve the clustering's objectives, all of the parameters and algorithms will be discussed and experimented carefully in this report. The first section presents a background research of K-means, Hierarchical and Mixture of Gaussian algorithms. The second section describes how to optimise clustering algorithms according to different defined numbers of clusters, (dis)similarity methods and evaluation methods. It then follows by simulating and presenting the results generated from those functions/algorithms/methods to both artificial and real data sets.

2 Background Research

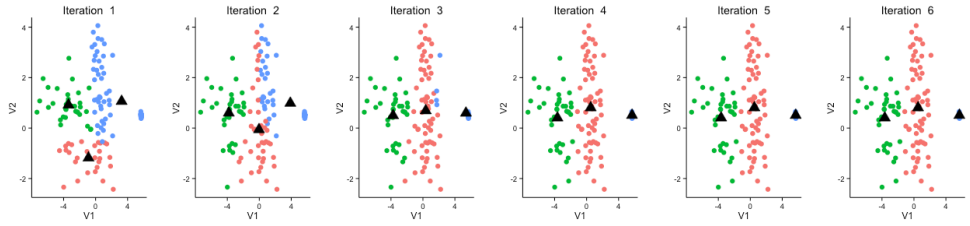
2.1 K-means

- **Original K-means**

K-means is one of the simplest non-parametric partitioning clustering algorithms using iterative methods. K-means aims to subset the data points into k groups such that the distance from points to their assigned cluster centre is minimized. Though many books and papers have described the K-means algorithm, my references are based upon those from Hastie et al [2]. First, a K number of clusters, a K number of random initial centroids (centre points), a distance method and a cluster validation method to measure distortion value are defined in advance. In the first phase, each observation is assigned to the closest cluster centroid according to the defined distance. The distortion measure is also computed and recorded at this step. In the second phase, the centroid, or mean of the observations in each cluster is re-calculated and will be treated as the new cluster centroids. These two steps are iterated until the cluster assignments stop changing, or a defined maximum number of iterations is exceeded. At this point,

the cluster centroids should remain the same or only slightly change for any extra iteration. The K-means algorithm is visualised in example 1 below.

K-means may cause an expensive calculation with a high dimension data set, and can be relatively slow because it is necessary to re-compute the distance between every cluster centroids and every data point in the first step. Therefore, many scientists have developed and proposed improved methods. One of them, Bishop et al. [3] mentioned a few approaches to speed up the K-means algorithm, such as "re-computing the data structure as a tree where nearby points are in the same subtree", or by "making use of the triangle inequality for distances which thereby avoids unnecessary distance calculations". Both Hastie et al [2] and Bishop et al [3] illustrate that K-means is limited to quantitative variables as well as a lack of robustness against great distance outliers. Moreover, K-means only works well with distance measure and can be difficult to reach the convergence with similarity measurement. Therefore, they suggested using K-medoids to be more robust and apply similarity measure.



Example 1: *The progress of K-means with K = 3 on the simulated data set with 150 observations in 2 dimensions. At iteration 1, initial centroids are randomly selected. Starting from iteration 2 -5, each observation is assigned to its closest centroids and the new centroids are re-calculated until the assignment stops changing.*

- **K-medoids**

As described in Hastie et al [2], K-medoids restrict the cluster centroid to be one of the observations assigned to the cluster. Instead of explicitly computing the mean of the cluster, an observation that minimises the total distance to other points in that cluster is chosen as the medoid of the cluster. In other words, a medoid is the most representative point in a cluster. The calculation of clusters' medoids makes the algorithm computational expensive. Also a more appropriate dissimilarity measure can be implemented, K-medoids can be more complex than the original k-means algorithm. The next step is assigning each observation to the closest cluster medoids in order to minimise the distortion value. Like K-means, these two steps are iterated until the assignments no longer change. Obviously, K-medoids is more robust than K-means because a medoid is less manipulated by outliers than a mean. For example, a 1-dimension data set is {1, 3, 5, 7, 1005}, the medoid is 5 but the mean is 204.

- **CLARA (Clustering LARge Applications)**

However, due to expensive computation, K-medoids is only efficient for a small data set. A concept of CLARA is introduced as an extension of K-medoids. As mentioned in Wiki book [5] "CLARA relies on the sampling approach to handle large data sets. Instead of finding medoids for the entire data set, CLARA draws a small sample from the data set and partitions the data into k clusters "around medoids". The quality of resulting medoids is measured by the average dissimilarity between every observation in the entire data set and the medoid of its cluster."

- **Initial centroids/medoids problem**

Whether applying K-means or K-medoids or CLARA, they are sharing the same problem. Hastie et al [2] notice that the distortion value (sum of square error) decreases with each iteration and the algorithms always converge no matter what the initial cluster centres. However, this minimum distortion value is a local rather than global optimum because the final results are sensitive to the chosen initial cluster centres. In other words, partition of data could be completely different by a small change in initial centres as seen in example 2. Witten et al [4] note the chance of finding a global optimum, which is infeasible in all practical clustering techniques, is increased by running the algorithm several times with different random initial centres. Witten et al [4] also recommend using k-means++ to improve speed and accuracy. At first, it chooses a random initial centre from among the observations with a uniform probability distribution. The next centre is selected with a probability that is proportional to the distortion value from the first one. This process is repeated until all K centres are chosen. It then proceeds with the standard k-means clustering above. Another common method mentioned in Hammerly and Elkan [21] is Forgy, which suggests choosing k random observations in the data set to be initial means. This method helps to spread the means out in the data set. Bishop [3] suggested applying the Robbins-Monro procedure that updates the closest cluster centre of each observation by adding the learning rate parameter. It is typically made to decrease monotonically as more observations are considered.

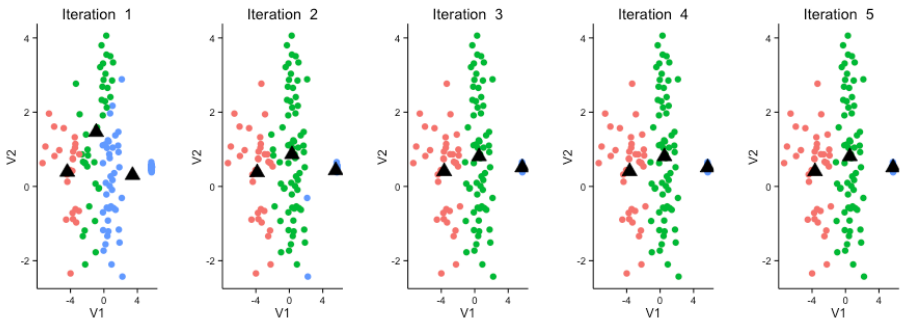
$$\mu_k^{new} = \mu_k^{old} + \eta_n(x_n - \mu_k^{old})$$

where

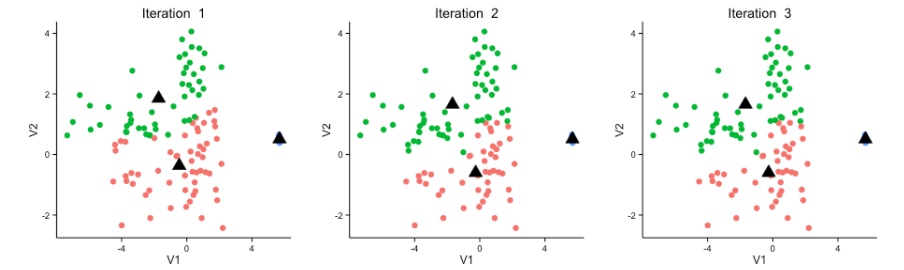
x is a data point

μ is the cluster centre

η is the learning rate parameter



a. *Perform K-medoids with K = 3 on the same simulation data set with example 1*



b. *Re-perform K-medoids with K = 3 on the same simulation data set with example 1*

Example 2: *K-medoids results with different initial centroids*

- **Advantages and Disadvantages**

K-means clustering algorithm is fast and easy to understand as well as practical for many different data sets. However, the results of K-means are heavily dependent on the chosen initial centroids and the pre-specified number of K clusters. Also, a distortion value of a cluster could be very high with the existence of outliers since each observation is assigned to only one cluster. For example, outliers can be observations that lie in the middle between cluster centres, then K-means will randomly assign them to a cluster. Or outliers are observations that quite different from each other and from all other observations.

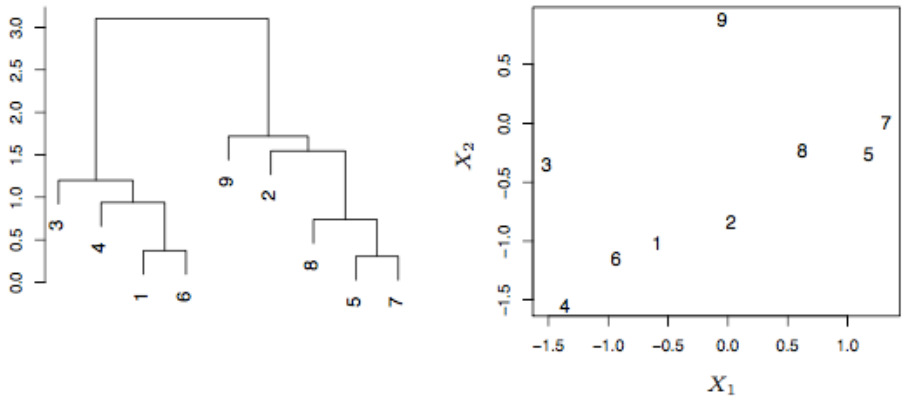
2.2 Hierarchy

As discussed above, the final clustering assignment of K-means or its development algorithms (k-medoids or CLARAs) depends on the pre-specified K clusters and the chosen initial cluster centres. In contrast, Hierarchical Clustering algorithms do not require those specifications; yet instead require the dissimilarity measure (which will be discussed later) between groups of observations. James et al [7] mention hierarchical clustering represents the data with clusters and sub-clusters in a tree-based diagram called a dendrogram. Hastie et al [2] also stated that the clusters at each level of the hierarchy are created by merging those at the next lower level. At the lower level, each cluster contains a single observation, and at the highest level there is only one cluster containing all of the data. Hierarchical

clustering methods are usually subdivided into agglomerative (bottom-up) and divisive (top-down) methods.

- **Dendrograms**

Dendrograms are a tree-structured graph used to visualise the structure of a hierarchical clustering's result. James et al [7] have a clear interpretation of the dendrogram. Each leaf represents one of the observations. Similar observations will start merging into branches (the horizontal axis) and these branches later can be merged with leaves or other branches. The earlier (the lower in the tree) the merges occur, the more similar the groups of observations are to each other. Thus, the observations that merge at the very bottom of the tree are quite similar to each other, whereas observations that merge close to the top of the tree will tend to be quite different. However, the (dis)similarity of the two observations will be indicated by the height of the merging. Hastie et al [2] show that when the dendrogram is cut horizontally at any height, it partitions the data into disjoint clusters. In other words, the height of the cut controls the number of clusters obtained. The number of clusters can be between 1 (if there is no cut) and N (if the cut is at height 0 where each observation is in its own cluster). An example 3 of the dendrogram below is taken from James et al [7].



Example 3: *An illustration of how to properly interpret a dendrogram with nine observations in two-dimensional space.* Left*: A dendrogram generated using Euclidean distance and complete linkage. Observation 5 and 7 are quite similar to each other, as are observation 1 and 6. However, observation 9 is no more similar to observation 2 than it is to observation 8, 5, 7, even though observations 9 and 2 are close together in terms of horizontal distance. This is because observations 2, 8, 5, 9 all fuse with observation 9 at the same height, approximately 1.8.* Right*: The raw data used to generate the dendrogram can be used to confirm that indeed, observation 9 is no more similar to observation 2 than it is to observation 8, 5, and 7.*

- **Agglomerative method**

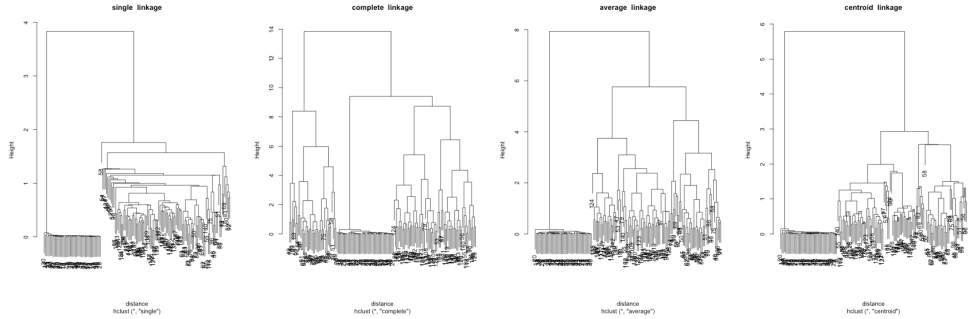
According to Hastie et al [2], agglomerative clustering algorithm starts at the bottom, which means each observation is treated as its own cluster. Then at

each level recursively merge a selected pair of clusters into a single cluster, which produces one less cluster at the next higher level. The pair chosen for merging consists of the two observations or groups of observations with the smallest inter-cluster dissimilarity. It is important to address these pairs are merged into a single cluster based on a pre-specified linkage method.

The four most common types of linkage are single, complete, average and centroid as described in James et al [7]. The first three methods need to compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, then:

- *Single* linkage or minimum distance records the smallest of these dissimilarities
- *Complete* linkage or maximise distance records the largest of these dissimilarities
- *Average* linkage or average distance records the average of these dissimilarities

The *centroid* linkage or mean distance will compute the dissimilarity between the centroid of cluster A and the centroid for cluster B. James et al [7] also specify that average and complete are more preferable than single linkage because they tend to yield more balanced dendrograms. Centroid linkage is less popular because it can often result in inversion "whereby two cluster are merged at a height below either of the individual clusters in the dendrogram". The difference can be seen in the example 4 below. In summary, for each step, the two observations or clusters is chosen based on their closest cluster yet merged based on the specified linkage method, hence, the dendrogram performance deeply depends on the chosen type of dissimilarity measure and linkage.



Example 4: *Agglomerative clustering with four different linkages applying to the simulated data set with 150 observations in 2 dimensions (same data set in example 1). From left to right: single, complete, average and centroid linkage*

- **Divisive method**

As determined by Hastie et al [2], the Divisive method, in contrast to the Agglomerative, starts at the top. It means the entire data set is treated as a single cluster. Then at each level recursively it splits one of the existing clusters at that level into two new clusters. The split is selected to generate two new clusters with

the largest between-group dissimilarity or less similarity. This process continues until all clusters have only one observation remaining. However, the divisive method is not popular due to the expensive computations.

- **Advantages and Disadvantages**

Hierarchical clustering algorithms are easy to understand since they present the data observations in an easy view tree-like format according to their similarity. Unlike K-means, the number of clusters does not need to be pre-specified because the tree can be cut at any level to generate the desired number of clusters. However, deciding how many clusters to obtain is indistinct. In addition, once observations are assigned to clusters, they can't be changed. Similar to K-means, because one observation can be assigned to only one cluster, the distortion value of the cluster containing outliers is higher than it is supposed to be. Further more, hierarchical clustering algorithms heavily depend on the dissimilarity measure and linkage method, thus, one might have to carefully consider and choose appropriately.

2.3 Mixture of Gaussians

As discussed above, both K-means and Hierarchical Clustering will force all outliers into a cluster, which increases the distortion value of that cluster. They also have no underlying density function to explain how the observations were produced, and are only concerned with finding the natural grouping in the data set. Hence, mixture models, particularly Mixture of Gaussians are a more appealing method. It helps reducing outliers, and introducing a clear density function to analyse/cluster the data set.

Zhiyuan [24] indicated that the mixture model is a probabilistic model that mixes the K base probability density functions. If the density estimation is parametric, meaning a particular form of the density is identified (e.g. Gaussian), the Maximum Likelihood method can be used to seek the necessary parameters (e.g. mean and variance). Otherwise, if the density estimation is non-parametric, which assumes knowledge about the density, then Prazen windows, Histograms or Kernel density estimation can be used to estimate the density, directly from the data. According to Murphy [8], Mixture models are used in two main applications, which are black-box density models, and clustering. A black-box density model is useful for a variety of tasks, such as data compression, outlier detection and creating generative classifiers. However, clustering is a more common application. The basic idea of mixture models clustering is to fit the mixture model at first, and then compute the posterior probability that an observation x belongs to cluster k. One of the most widely used mixture models is Mixture of Gaussians, as described below.

2.3.1 Mixture of Gaussians

Mixture of Gaussians (MOG), also be referred as a Gaussian mixture model (GMM), is the most widely used parametric mixture model. Douglas [9] define MOG as a parametric probability density function, represented as a weighted sum of K component Gaussian densities by a distribution π in the form:

$$p(x) = \sum_{k=1}^{K} \pi_k \, N(x|\mu_k, \Sigma_k)$$

where:

x is a D-dimensional continuous-valued data vector (i.e. observation)

K is a number of component Gaussian densities

$\mathrm{N}(x|\mu_k, \Sigma_k)$ is a component Gaussian density with its own mean μ_k and covariance matrix Σ_k

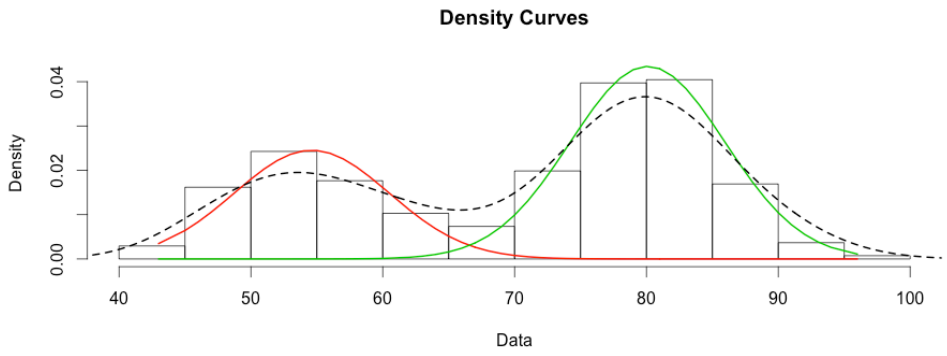
$\pi_k$is called mixing coefficient or can be interpreted as prior probability with a conditions that $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{K} \pi_k = 1$

As mentioned above, for parametric density estimation, Maximum Likelihood can be used to seek the solution that best explains the data set. In the case of MOG which is governed by the parameters $\pi_k, \mu_k \; and \; \Sigma_k$ where k = 1, 2…,K, the log of the likelihood functions is used to find the value of these parameters as follows:

$$\log p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \log \left\{ \sum_{k=1}^{K} \pi_k \, N(x|\mu_k, \Sigma_k) \right\}$$

where $X = \{x_1, \ldots, x_N\}$

As pointed out in Bishop et al [3], the maximum likelihood application to MOG is much more complex than a single Gaussian, due to the presence of the summation over k inside the logarithm. Because of this, the logarithm function no longer acts directly on the Gaussian. When the derivatives of the log likelihood are set to zero, a closed form solution will no longer be obtained. Besides, Murphy [8] also affirmed that computing the Maximum Likelihood estimate becomes increasingly difficult if there is missing data and/or latent variables. In both books, the authors mention an alternative approach called Expectation Maximisation. Murphy [8] defined it specifically "This is a simple iterative algorithm, often with closed-form updates at each step. Furthermore, the algorithm automatically enforces the required constraints". Supplementary details are shown below.



Example 5: *EM for MOG results with K = 2 applied to Faithful waiting data. This graph was generated following the tutorial in [27].*

2.3.2 Expectation Maximisation (EM)

Bishop et al [3] describe EM as an elegant and powerful method for finding maximum likelihood solutions for models with latent variables. Bilmes [10] outlines two main applications of the EM algorithm. The first is when the data has missing values. The second happens when optimizing the likelihood function is analytically difficult, but by assuming the existence of and values for additional but hidden (missing) parameters, the likelihood function can be simplified.

Lets assume that $X = \{x_1, \dots, x_N\}$ is the set of all observed variables, $Z = \{z_1, \dots, z_N\}$ is the set of hidden or missing variables and θ is the set of all model parameters, thus the log likelihood is given by

$$\log p(X|\theta) = \log\{\sum_Z p(X, Z|\theta)\}$$

The general EM algorithm is summarised in Bishop et al [3] as the following:

1. Choose an initial setting for parameters θ_{old}
2. **E-step**: Using the current parameters to calculate the posterior distribution of the latent variables given by $p(Z|X, \theta_{old})$.
3. **M-step**: Using the current posterior distribution to get the new parameter estimate θ_{new} by maximizing the expectation of the log likelihood $Q(\theta, \theta_{old})$

$$\theta_{new} = \arg max_\theta \; Q(\theta, \theta_{old})$$

$$where \; Q(\theta, \theta_{old}) = \sum_Z p(Z|X, \theta_{old}) \log p(X, Z|\theta)$$

4. Check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then let $\theta_{old} \leftarrow \theta_{new}$ and return to step 2.

In general, the EM is an iteration algorithm that alternates between performing an expectation (E) step, which finds the posterior distribution using the current parameters estimate, and a maximization (M) step, which revises the parameters estimate by maximizing the expected log likelihood found on the E-step. These parameters are then used in the next E-step.

2.3.3 Expectation Maximisation for Mixture of Gaussian

As described above, EM is a better approach to solve problems related to MOG. The goal is to maximise the log likelihood function with respect to the parameters (mixing coefficents π_k, mean μ_k and covariance Σ_k). Together with the defined syntax in session 2.3.1, the latent variables z is introduced as in Bishop [3]. Assuming a K-dimension binary random variable z having a 1-of-K representation, in which a particular $z_k$is equal to 1 and all other elements are equal to 0. The value of $z_k \in \{0,1\}$ and $\sum_k z_k = 1$, so there are K possible states for the vector z according to which element is nonzero.

Summarising from Bishop [3], the joint distribution $p(x|z)$ can be defined in terms of a marginal distribution p(z) and a conditional distribution p(x|z). While the marginal distribution over z is specified in term of the mixing coefficients π_k, the conditional distribution of x given a particular value for z is Gaussian, such that:

$$p(z_k = 1) = \pi_k$$

where

$$0 \leq \pi_k \leq 1 \text{ and } \sum_{k=1}^{K} \pi_k = 1$$

because z uses 1-of-K representation, the marginal distribution can be written in the form:

$$p(z) = \prod_{k=1}^{K} \pi_k^{z_k}$$

similarly

$$p(x|z_k = 1) = N(x|\mu_k, \Sigma_k)$$

the conditional distribution can be written in the form:

$$p(x|z) = \prod_{k=1}^{K} N(x|\mu_k, \Sigma_k)^{z_k}$$

Thus, the marginal distribution x, which is also a Gaussian form, is obtained by summing the joint distribution over all possible states of z to give:

$$p(x) = \sum_{z} p(z)p(x|z) = \sum_{k=1}^{K} \pi_k \, N(x|\mu_k, \Sigma_k)$$

Now, for every observation x_N, there is a corresponding latent variable z_N and as introduced above, EM is the powerful method for finding maximum likelihood solutions for models with latent variables. Bishop [3] had already summarised it as follows:

1. For each cluster k, initialising the means μ_k covariances Σ_k and mixing coefficients π_k as well as evaluate the initial value of the log likelihood.
2. **E-step**: evaluate the posterior distribution using the current parameters.

$$\gamma(z_{nk}) = P(z_k = 1 \mid x) = \frac{\pi_k N(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x|\mu_j, \Sigma_j)}$$

3. **M-step**: re-estimate the parameters using the current posterior distribution

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n-1}^{N} \gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$$

$$\pi_k^{new} = \frac{N_k}{N}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

4. Evaluate the log likelihood and check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then return to step 2.

$$\log p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \log \left\{ \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k) \right\}$$

5. Each observation is then assigned to the cluster according to its most likely posterior probability.

When comparing the K-means algorithm with EM for MOG, there is a close similarity. The two steps in K-means are, updating the assigned clusters and updating the mean of the cluster corresponding respectively to the E (expectation) and M (maximisation) steps of the EM algorithm. Whereas, the K-means algorithm assigns observation uniquely to a cluster in respect to a cluster's mean. EM for MOG allows overlap clusters based on the posterior probabilities. Again, as noticing in Bishop [3], EM algorithm takes many more iterations to reach convergence compared with K-means, and each iteration requires significantly more computation. Therefore, they suggest running K-means or Hierarchical clustering in order to find a suitable initialisation for EM for MOG. Each cluster in K-means then can be used as an initial cluster in EM for MOG. Cluster's mean does not need to recalculate. Cluster's covariance matrix can be computed easily from the set of observations in that cluster, and mixing coefficients can be set to the

proportion of data observations assigned to the respective clusters. Like K-means, EM for MOG is an iterative model, which increases the log likelihood on the next iteration until it is converged. However, it is common practice to stop the algorithms before the convergence in order to reduce computational time. Abbi et al [20] has experienced and suggested some stopping criteria that make the algorithms faster. In their paper conclusion, they recommended to stop when the log likelihood value changes no more than 10^{-8}, or the mixing coefficient, means and covariance change no more than 10^{-8}, 10^{-6}, 10^{-4} respectively.

- **Advantages and Disadvantages**

EM for MOG is computational expensive due to its slow convergence; however, it's a soft assignment where one observation can be assigned to a different cluster based on their posterior probabilities. This helps reducing the effect of data set's outliers. Besides, having other properties like K-means, the initial parameters and the predefined number of clusters significantly impact the EM of MOG performance.

3 Optimisation Clustering

3.1 The number of clusters

Both K-means and EM for MOG are needed to pre-specify the number of clusters; even the hierarchical clustering doesn't require this parameter at the beginning, but at the end when cutting the dendrogram. Thus, the choice for K number of clusters is a critical issue. A number of different methods have been introduced over time.

- **Elbow Method**

As mentioned in Kodinariya et al [6], the Elbow method is a visual method, where clustering's distortion value is plotted against its number of clusters. Particularly, the higher the number of cluster, the lower the distortion value. This is because observations are partitioned into smaller groups; thus, it is getting closer to their cluster centroids. However, at some point, adding another cluster will not give much better distortion value. Hence, it creates an angle in the graph, and the number of cluster is chosen at this point. Nevertheless, Kodinariya et al [6] also state that this method might not be efficient in case there is no elbow or several elbows in a graph.

- **Gap statistic**

R.Tibshirani, G.Walther and T.Hastie [14] developed the gap statistic in 2000 for estimating the number of clusters in a data set. The gap statistic is computed by comparing the change in distortion value (within-cluster dispersion) with its expectation under an appropriate reference null distribution. Their

research [14] concluded that this technique could be applied to any clustering algorithm, such as K-means, Hierarchical and Mixture of Gaussian. The computation of gap statistic proceeds as follows:

1. Cluster the data with various number of clusters from k = 1, 2, …,K giving the distortion value W_k

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} D_r$$

where

- n_r is the number of observations in cluster r
- D_r is the sum of the pairwise distances for all observations in cluster r

2. Generate B samples of the original data set computed from a Monte Carlo sample drawn from the reference distribution and cluster each sample giving distortion value W_{kb}^*. Then the gap statistic is computed by

$$Gap(k) = \frac{1}{B} \sum_b \log W_{kb}^* - \log W_k$$

3. Let $\bar{l} = (\frac{1}{B}) \sum_b \log W_{kb}^*$, compute the standard deviation

$$sd_k = [(\frac{1}{B}) \sum_b \{\log W_{kb}^* - \bar{l}\}^2]^{1/2}$$

and define $s_k = sd_k \sqrt{(1 + \frac{1}{b})}$

Finally, the optimal number of clusters is the smallest k such that

$$Gap(k) \geq Gap(k+1) - s_{k+1}$$

Regarding to Tibshirani et al [14], the gap statistic is outperformed on well-separated clusters. But when there are overlapping or not well separated clusters, estimating the number of clusters is no longer well defined. Thus, I moved on to another method, Bayesian Information Criterion.

- **Bayesian Information Criterion**

Murphy et al [8] cite that Bayesian Information Criterion (BIC) was first developed by Gideon E. Schwarz in 1978. It is a penalised log-likelihood function that can be used for model selection and has the following form:

$$BIC = \log p(X \mid \hat{\theta}_{ML}) - \frac{dof(\hat{\theta})}{2} \log(N)$$

where

$\hat{\theta}_{ML}$ is the maximum likelihood estimation for the model
$dof(\hat{\theta})$ is the number of free parameters in the model

N is the number of observations

Though, BIC has been widely applied for model identification in time series, linear regression etc., it is broadly used to choose the number of clusters in clustering analysis. Steele et al [13] is one of the papers that proved BIC outperforms when compared to other methods such as Maximum A Posteriori (MAP), Integrated Complete-data Likelihood (ICL), and Akaike's Information Criterion (AIC).

Let $X = \{x_1, \dots, x_N\}$ with d-dimension is the data set that is needed to be cluster and $C_k = \{c_i : i = 1, \dots, k\}$ is the clustering which has k clusters. Chen and Gopalakrishan [11] demonstrate how to apply BIC to K-means and Hierarchical Clustering as follows.

- For each cluster that generates from K-means algorithm, it is treated as a multivariate Gaussian distribution $N(\mu_i, \Sigma_i)$ where mean μ_i and covariance matrix Σ_i can be estimated. Thus, the number of free parameters for each cluster is $d + \frac{1}{2}d(d+1)$. Let n_i is the number of observation in cluster c_i, the best K-means clustering is which maximises the BIC criterion:

$$BIC(C_k) = \sum_i^K \left\{-\frac{1}{2} n_i \log|\Sigma_i|\right\} - Nk\left(d + \frac{1}{2}d(d+1)\right)$$

- For Hierarchical clustering, Chen and Gopalakrishan [11] proposed a termination criterion which only merges two node if the merging increases the BIC value. So if the below calculation is negative, those two nodes shouldn't be merged:

$$-n \log|\Sigma| + n_1 \log|\Sigma_1| + n_2 \log|\Sigma_2| + N(d + \frac{1}{2}d(d+1))$$

  where

  n_1, n_2 is the number of observations in node 1 and 2

  Σ_1, Σ_2 Are the covariance matrixes of node 1 and 2

  $n = n_1 + n_2$ is number of observations of the merged node

  Σ is the covariance matrix of the merged node

- For the Mixture of Gaussian, Steele et al [13] assumes the number of parameters for each cluster is

$$dof(\hat{\theta}) = k * \left(1 + d + d * \frac{d+1}{2}\right) - 1$$

The maximum likelihood function, $\log p(X \mid \hat{\theta}_{ML})$, can be computed by the EM algorithm that introduced above. Thus, the BIC of each $C_k$is formed as:

$$BIC_{C_k} = \log p(X \mid \hat{\theta}_{ML}) - \frac{1}{2} * (k * \left(1 + d + d * \frac{d+1}{2}\right) - 1) * \log(N)$$

The best number of clusters is the one that maximises BIC.

BIC performs well in determining the number of components, which is also identified as a number of clusters, of a mixture model. Although, Baudry et al

[12] argued that BIC selects the number of mixture components to provide a good approximation to density, rather than the number of the clusters. This is true in the situation where Gaussian distribution poorly fits individual clusters, and model-based clustering tends to represent one non-Gaussian cluster, by a mixture of two or more Gaussian distributions. At the end, it can lead to overestimation of the number of clusters, so that the number of mixture components is not necessary the same as the number of clusters. Thus, Baudry et al [12] proposed and proved the privilege of the new method in Combining Mixture Components for Clustering published in the Journal of Computational and Graphical Statistics. This method takes advantage of the Gaussian model-based clustering, but can avoid the overestimation of the number of clusters. It is also computationally efficient because only the conditional membership probabilities are used, that allows this method applying to any other mixture model without modification.

Baudry [12] clearly described and suggested the implementation procedure for this algorithm in his research paper and appendix, so I have summarised and linked it to this project. The idea is to build a sequence of possible clustering solutions that fits the data well. Especially, a sequence of K mixture of Gaussians is generated from Expectation Maximisation with $\forall K \in \{K_{min}, \dots, K_{max}\}$. Then the best number of components is chosen using BIC.

From the best K-cluster solution, two mixture components might be chosen to merge so as to minimise the entropy of the finial clustering. Let set $t_{i,1}^K \dots t_{ik}^K$ be the conditional probabilities that observation x_i arises from cluster 1,…,K. If two clusters k and k′ are combined, the $t_{i,j}^K$ remain the same for every j except for k and k' because the newly joined cluster has this combined conditional probability

$$t_{i,k\cup k'}^K = t_{i,k}^K + t_{i,k'}^K$$

Then the result entropy is

$$-\sum_{i=1}^{n}\left(\sum_{j\neq k,k'} t_{ij}^K \log t_{ij}^K + \left(t_{ik}^K + t_{ik'}^K\right)\log\left(t_{ik}^K + t_{ik'}^K\right)\right)$$

Thus, among all possible pairs of cluster (k, k'), the pair that maximise the following criterion are chosen to combine:

$$-\sum_{i=1}^{n}\left(t_{ik}^K \log t_{ik}^K + t_{ik'}^K \log t_{ik'}^K\right) + \sum_{i=1}^{n} t_{i,k\cup k'}^K \log t_{i,k\cup k'}^K$$

As soon as two clusters are combined, the $t_{i,k}^K$ becomes the conditional probability that observations x_i belongs to one of the combined components in cluster k.

This algorithm will change the number and definition of cluster while the log likelihood remains the same with any combined solution since it always based on the same Gaussian mixture. Baudry et al [12] also suggested that if a more

automated procedure is desired for choosing a single solution, one can apply Elbow rule on the graphic displaying the entropy variation against the number of clusters or select the solutions providing the same number of clusters as the Integrated complete likelihood (ICL) method which proposed by Biernacki, Celeux and Govaert (2000).

3.2 (Dis)similarity methods

Clustering is about grouping similar observations based on a defined measure method, which determines whether two observations are similar or dissimilar. Thus, (dis)similarity is one of the main factors used to decide the quality of the clusters. Maimon and Lior [19] define two main types of measures: distance and similarity measures.

- **Distance measure**

Maimon and Lior [19] indicate that the valid distance should be symmetric and obtains its minimum value (usually zero) in case of identical vectors and has the following properties:

a. Triangle inequality

$$d(x,y) \leq d(x,z) + d(z,y) \;\; \forall x,y,z \in S$$

b. $d(x,y) = 0 => x = y \;\; \forall x,y \in S$

where d(x,y) is the distance between two vector x and y.

Few common distance measures:

- **Euclidean distance**

$$d(x,y) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$$

- **City-block distance or Manhattan**

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

- **Similarity measure**

Maimon and Lior [19] again specified that this function should be symmetrical and have a large value when two observations are somehow "similar" and constitute the largest value for identical vectors. One of the most popular methods is:

- **Pearson Correlation Measure**

$$s(x,y) = \frac{(x - \bar{x})^T (y - \bar{y})}{\|x - \bar{x}\| \|y - \bar{y}\|}$$

where

$\bar{x}, \bar{y}$ are the average feature value of x and y over all dimensions.

s(x,y) is similarity function compare two vector x and y

3.3 Cluster validation

After choosing the clustering algorithms and their parameters, it is important to justify how well the clusters perform. In order to measure the quality of the cluster, two types of clustering validations are introduced which are internal and external criterion.

- **Internal criterion**

As mentioned previously, all clustering algorithms are trying to achieve high intra-cluster similarity and low inter-cluster similarity. In other words, this is an internal criterion for the quality of a clustering. Here describes several popular methods for Internal criterion.

- **Sum of Square Error (SSE)**

This is the most common and easy method to compute. The author of [16] recommended that SSE could be used to measure both cohesion and separation of clusters. Cohesion is simply the sum of the squared differences between each observation and its cluster's centres. Since it tries to seek for similarity amongst observations in a cluster, it is also called within-cluster sum of squares (WSS). Separation is measured by the sum of the squared difference between clusters. It tends to figure out the dissimilarity between clusters, so they are referred to between-cluster sum of squared (BSS).

$$WSS = \sum_{k} \sum_{x \in C_k} (x - m_k)^2$$

$$BSS = \sum_{k} |C_k|(m - m_k)^2$$

where $|C_k|$ is the size of cluster k

Clustering is perfect if there is no error, so that the lower the WSS and BSS value, the better the clustering.

- **Silhouette width**

Rendon et all [17] also refer to Silhouette width, which combines ideas of both cohesion and separation. Specifically, it measures how similar those observations to others in its own cluster are, when compared to observations in other clusters:

$$s(i) = \frac{(b(i) - a(i))}{Max\{a(i), b(i)\}}$$

where

s(i) is silhouette value of the *ith* observation of cluster X_k

a(i) is average distance between the *ith* observation and all observations included in X_k

b(i) is minimum average distance between the *ith* observation and all observations included in other cluster

The silhouette value ranges from -1 to +1. The high silhouette value indicates a high value of similarity to its cluster and high dissimilarity to the neighbour clusters.

- **Pearson Gamma**

Pearson Gamma, as discussed by Montarani and Laura, maximises the correlation coefficient between the vector of pairwise dissimilarities and the binary vector that is 0 for every pair of observations in the same cluster and 1 for every pair of observation in different cluster.

- **External criterion**

Even though the internal criterion provides a good review of the intra and inter cluster, and the most direct evaluation, it sometimes can be a pricey calculation, and can't always prove the clustering's performance. Rendon et al [17] define external criterion as an alternative method, which evaluates the clustering algorithm, based on external information that is not contained in the data set. They suggested a few applications as follows:

- **Purity**

The purity of clustering is calculated by counting the number of correctly assigned observations, then dividing the total number of observations. The defined function is as follows:

$$P_j = \frac{1}{n_j} max_i(n_j^i)$$

where

P_j is purity of cluster j where the number of observations in cluster j with class label i.

n_j is size of cluster j

The overall purity of the clustering solution is obtained as a weighted sum of individual cluster purities and given as

$$Purity = \sum_{j=1}^{m} \frac{n_j}{n} P_j$$

where

m is the number of clusters

n is total number of observations

The purity values range from 0 to 1. A bad cluster has purity value close to 0 and a perfect cluster has a purity value of 1.

- **Entropy**

Entropy computes the purity of the clusters so if a cluster consists of only a single class, the entropy is 0. At first, a class distribution of observations in each cluster is needed to calculate:

$$E_j = \sum_i p_{ij} \log(p_{ij})$$

Then, the total entropy of clusters is the weighted sum of the entropies of all clusters:

$$E = \sum_{j=1}^{k} \frac{n_j}{n} E_j$$

where

n_j is the size of cluster j

k is the number of cluster

n is total number of observations

- **Rand index**

Rand index is described in Manning and Prabhakar [18] by penalising both false positive and false negative decision during clustering. The calculation is as follows:

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

where

RI is rand index

TP is true positive decision assigns two similar observations to the same cluster

TN is true negative decision assigns two dissimilar observations to different clusters

FP is false positive decision assigns two dissimilar observations to the same cluster

FN is false negative decision assigns two similar observations to the different cluster

Rand index measures the percentage of decisions that are correct, the higher the value RI, the better the clustering.

4 Experiment

4.1 Knowledge acquisition

The purpose of this experiment was to implement and compare K-means, Agglomerative Hierarchical and Expectation Maximisation for Mixture of Gaussians (EM for MOG), as well as the parameters that help to enhance the clustering performance on both artificial, and real world data. The implementation is summarised below:

Algorithms	**K-means**	**Agglomerative Hierarchical**	**EM for MOG**
Initialisations	X		X
Dissimilarity			
- Euclidean Distance	X	X	X
- Manhattan Distance	X	X	X
- Pearson Correlation	X	X	X
Number of clusters			

– Elbow method	X	X	X
– Gap Statistic	X	X	X
– BIC	X	X	X
– Combined Mixture Components for Clustering			X
Evaluation			
– Sum of Square Error	X	X	X
– Silhouette width	X	X	X
– Pearson Gamma	X	X	X
– Entropy	X	X	X
– Rand index	X	X	X

For this project, R Programming language (R) was utilised to implement the algorithms and graphics. R is practiced by many statisticians and data miners around the world and has been rising in popularity for its use in developing statistical software and graphics, as well as data analysis. Together with the availability of many statistical functions and excellent graphics, using R enables me to write my own functions, modified the available function and easily visualise the data that I gather.

I wrote my own functions for some of the principal concepts, such as K-means, K-medoids and Expectation Maximisation for Mixture of Gaussian, and most of the others were either applied directly or modified with the available packages in R.

4.2 Data set

- **Artificial data**

The experiment was conducted with three different artificial data sets. The first data set consists of samples drawn from five well-separated Gaussian distributions, where each cluster contains 100 observations in 2 dimensions. I purposely set the means within a wide range, so that there is no overlap cluster. With the same data set size, each of the second and third data sets is composed of a five-component mixture of Gaussians, where their means and covariance are deliberately generated by a MixSim function. I used this function because it allows me to set the overlap rate among the mixture components. Furthermore, the third data set has an extra cluster containing 100 outliers. So, I combined MixSim and some others into a function to easily generate the artificial data sets. Figure 1 represents my artificial data sets.

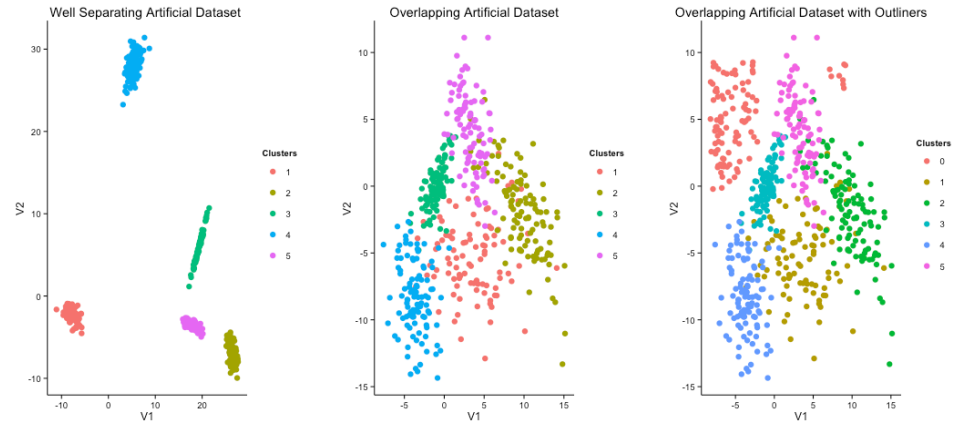


Figure 1: *Artificial Data set*

- **Real data**

The real data set is a Gene Expression data set provided by Dr. Alberto Paccanaro for the coursework on K-means Clustering, in our Programming for Data Analysis class. The data set is comprised of 3006 observations with 8 attributes where each observation is a gene, the first attribute is the gene's name and each of the other attributes represents a time point or a condition. So, each entry is the expression of a gene at time (or condition).

4.3 Simulation process

I have written my own function to implement the K-means, K-medoids and Expectation Maximisation for Mixture of Gaussian algorithms in R. These functions allow me to flexibly identify the initialisation methods and (dis)similarity methods. However, I used the available hclust function for Hierarchical clustering in order to conveniently cut and plot the dendrogram. The simulation starts with implementing and comparing different initialisation, dissimilarity distance and number of clusters methods. It then follows by generating, evaluating, and visualising the three different algorithms with the best choices of parameters. I used the current function cluster.stats within R's packages "fpc" to quickly collect values of some internal criterion of the clustering, such as Sum of Square Error, Silhouette Width and Pearson Gamma. Additionally, using the same function, I computed some external criterion, such as Entropy and Rand Index for artificial data set because it has a pre-specified structure.

- **Initialisation methods**

The first experiment is to measure the performance of K-means with several different initialisation methods. In particular, they are a traditional initialisation method that uses a random set of numbers, a Forgy that randomly selects a set of observations within the data set, and K-means++ that randomly select observations based on specific rules of probability. As all the initialisation

methods have to randomly select samples, K-means algorithm was run 20 times for each method to check the performance stability and increase the chance of finding the global optimum . The following data sets and parameters are set as follows for this experiment:

+ Data sets:
  - Overlapping Artificial Data set with K = 5
  - Well-separated Artificial Data set with K = 5
  - Gene expression real Data set with K = 15
+ Dissimilarity distance: Euclidean
+ Distortion value: Sum of Square Error (SSE)

The result is then plot into a column chart as seen in figure 2 where x-axis and y-axis are index of trial runs and SSE respectively with the random method in red, Forgy method in green and k-means++ in blue on the legend. The summary of average SSE is presented in table 1. We can see that K-means++ returns the best result on both overlapping artificial data set and real data set while the Random method is better with well-separated artificial data set. Looking further into the graphs, for both artificial data sets, you can see the performance of the three initialisation methods are not that different and are quite stable at most of the time. However, in the real data set, K-means++ outperforms the Random and Forgy method. Overall, it can be seen that K-means++ is a better solution.

	Overlapping Data	Well-Separated Data	Real Data
Random	4681.857	7108.199	8e+11
Forgy	4699.622	7320.535	8e+11
K-means++	4601.806	7493.592	5e+11

Table 1: *Average SSE of data sets with different initialisation methods*

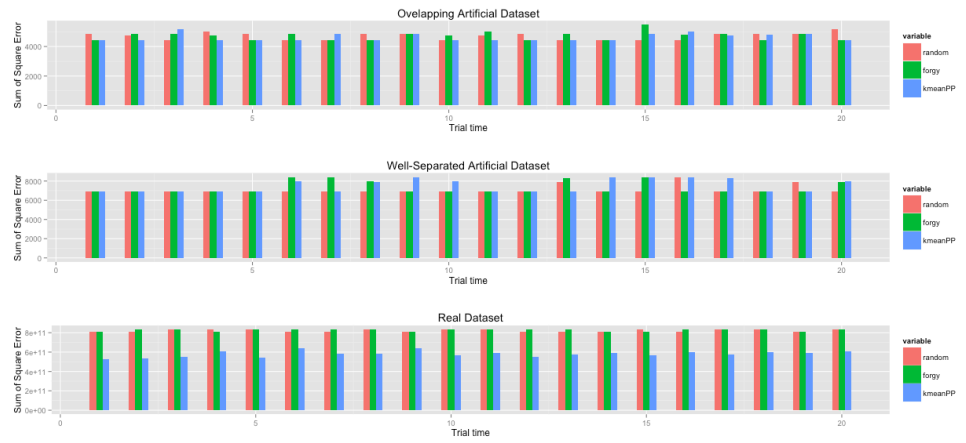


Figure 2: *K-means algorithms with different initial methods on Artificial and Real Data set*

I applied the same concept to Expectation-Maximisation for Mixture of Gaussian (EM for MOG) algorithm but used Log Likelihood as the distortion value along with the following initialisation methods:

1) Randomly selecting the mixing of coefficient, means and covariance
2) Running K-means++ to get means and covariance matrixes from the clusters found in K-means, and computing the mixing coefficient from the fraction of observations assigned to the respective clusters.

Like the K-means's experiment, the results from artificial and real data sets are plotted into the column chart as seen in figure 3, where the x-axis and y-axis are the index of trial runs and the SSE respectively, with the random method in red and k-means in blue on the legend. As we can see easily on the graph, the performance of both initialisation methods is similar and constant throughout 20 trials. In the real data set, the Random method is missing because it can't return any cluster solution after 100 iterations. Thus, I can conclude that the Random method is not sufficient for Expectation Maximisation for Mixture of Gaussians algorithm.

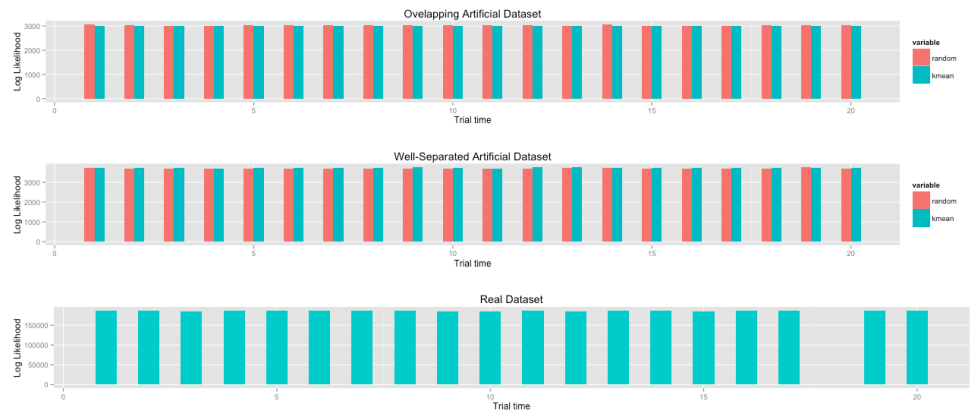


Figure 3: *EM for MOG with different initial methods on Artificial Datsets and Real Data set*

The second experiment aims to discover the difference in clustering when using the same algorithm, but different (dis)similarity methods for the same data set. The distance measures will use Euclidean and City-block (also called Manhattan), and the similarity measurement will be Pearson Correlation. As mentioned in section 3.2, the similarity measure doesn't compute the distance between two observations instead compares the similarity between those. To compare, I visualise the results in figure 4 and 5, as well as compute the Sum of Square Error. In addition, the experiment will use both K-means and Agglomerative Hierarchical clustering whose performance heavily depends on dissimilarity methods.

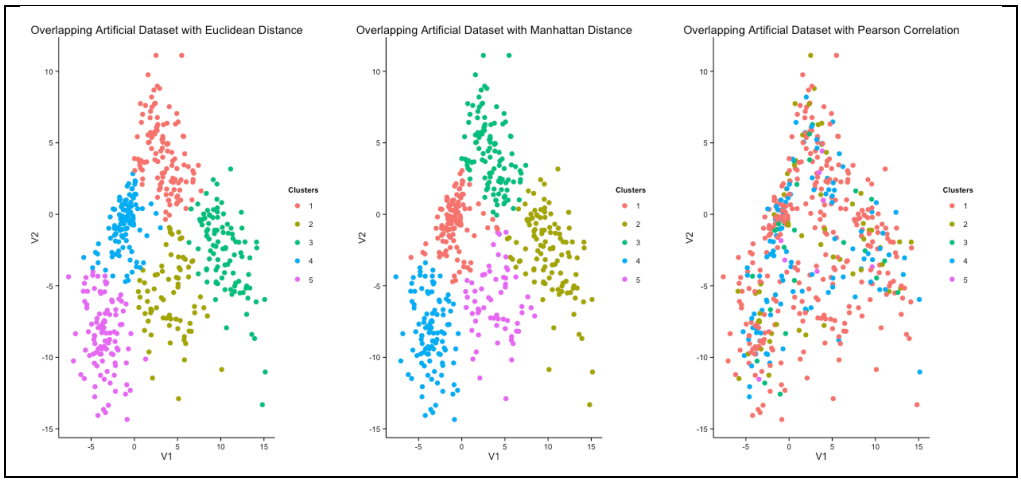
Figure 4 below illustrates the performance of K-means++ clustering with three different distance methods, Euclidean distance, Manhattan and Pearson correlation, applying to overlapping and well-separated artificial data sets with k = 5. Each colour in the graph represents a unique cluster. As we can see Euclidean

and Manhattan partition both overlapping and well-separated data sets in 5 clear separated clusters while Pearson correlation doesn't create any clusters. Obviously, the Pearson correlation isn't a good solution for these particular artificial data sets. In contrast, the result of K-means++ on the real data set, which tried to find 5 clusters, is totally different. Figure 5 demonstrates that Pearson correlation is a better choice, because it groups the identical movement pattern of the gene while the others try to compute and find the closest distance to the clusters' centres. Moreover, Pearson correlation also partitions the data set into similar cluster's size, but Euclidean and Manhattan have both sparse and dense clusters. In order to confirm the distance performance, I recorded the sum of square error and the results are as follows:

	Overlapping Data set	Well Separated Data set	Real Data set
Euclidean	4409	1076	2.068696e+12
Manhattan	7139	1783	8.695095e+12
Pearson correlation	247.5	247.5	36.7

Table 2: *Sum of Square Error of K-means++ clustering with different dissimilarity methods*

As we can see, the Euclidean measure is better than Manhattan in both of the artificial data sets. By looking at the graph we can see that Pearson correlation is not a sufficient solution here. However, the SSE of Euclidean and Manhattan on the real data set are extremely high that have no clear figure to compare to, while Pearson correlation yields its sum of square error (or correlation) at 36.7 that is relatively good for 3006 observations.



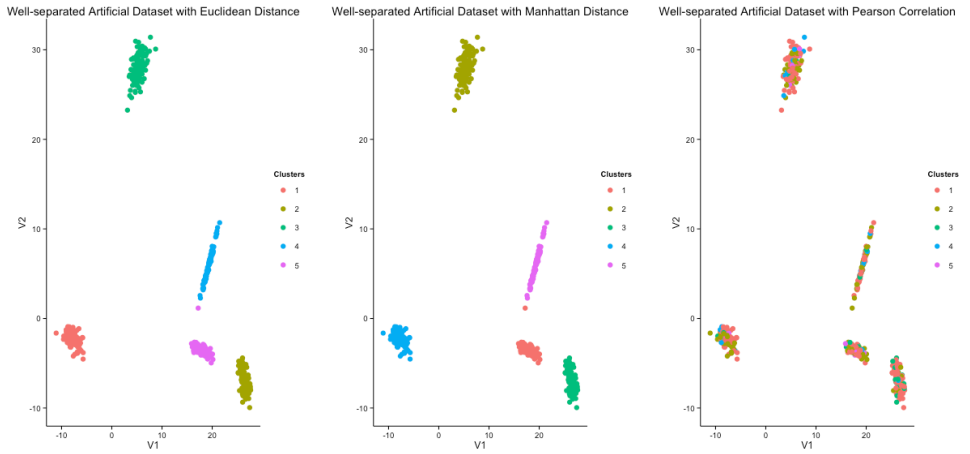
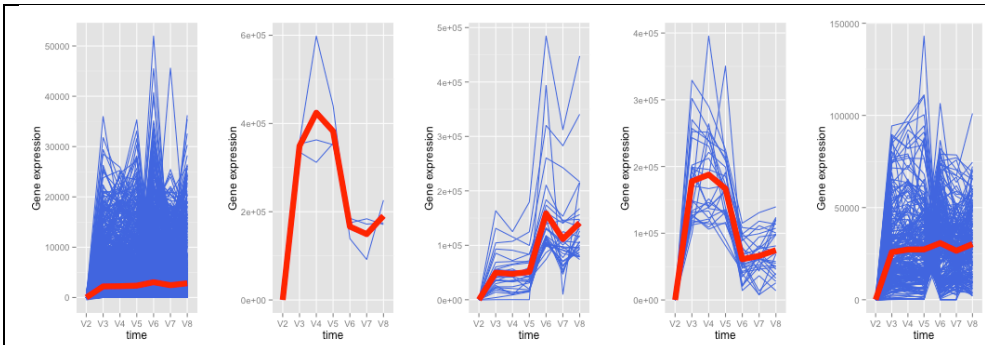
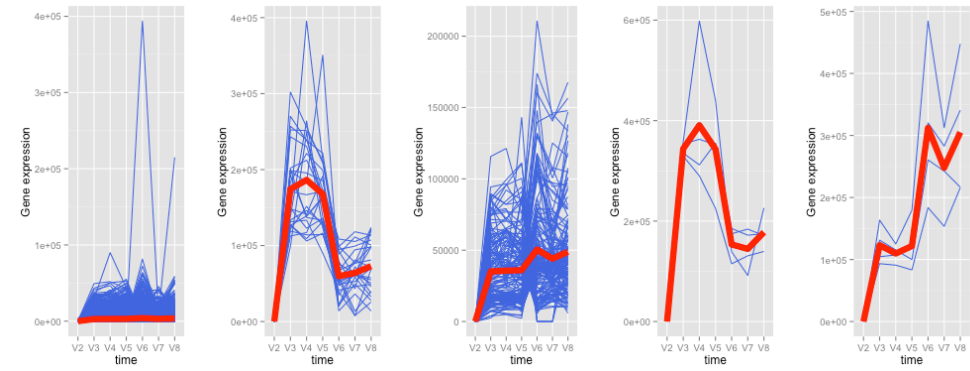


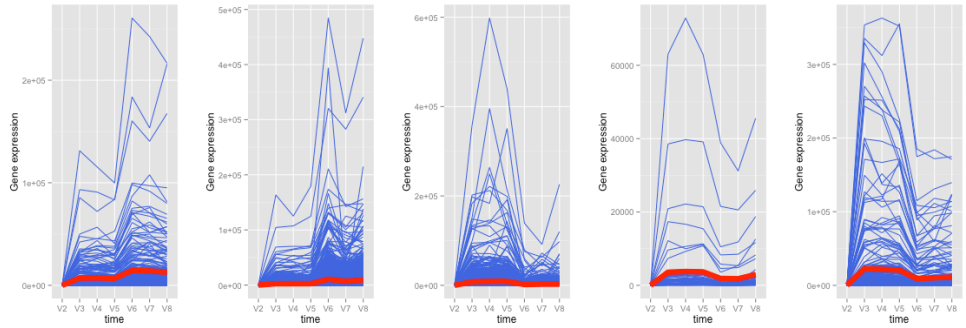
Figure 4: *K-means with different distance methods on Overlapping and Well Separated Artificial Data sets*



a. *K-means with Euclidean distance*



b. *K-means with Manhattan*



c. *K-means with Pearson Correlation*

Figure 5: *K-means with different distance methods on Real data set*

As Hierarchical clustering depends not only on the (dis)similarity distance but also on the linkage, I first ran the algorithms with complete linkage and cut the tree for 5 clusters on both of the artificial and real data sets using Euclidean, Manhattan and Pearson Correlation distance in order to compare the dissimilarity distance. It follows by comparing and seeking the best solution for linkage. The results of dissimilarity distance are presented in table 3 and it's not surprising that they are similar to K-means's experiment. Artificial data sets work well with Euclidean while Real data set is best with Pearson correlation.

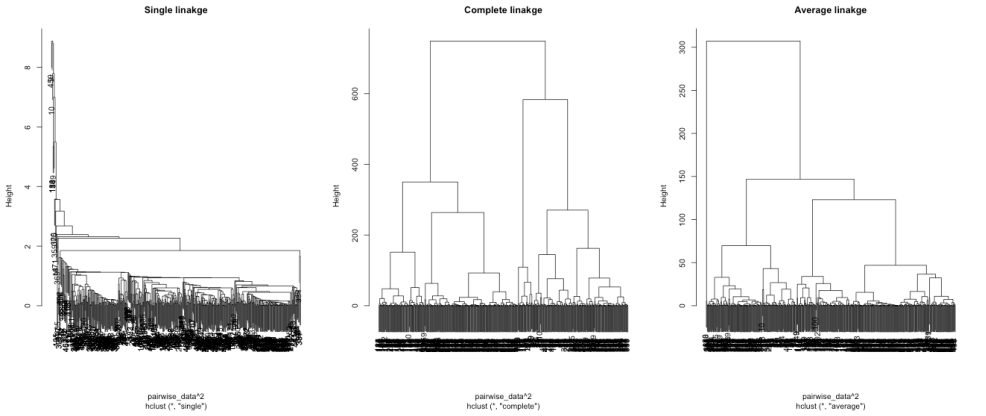
	Overlapping Data set	Well Separated Data set	Real Data set
Euclidean	6039	1082	2.485049e+12
Manhattan	9531	2577	1.324446e+13
Pearson correlation	247.5	246.5	587

Table 3: *Sum of Square Error of Agglomerative Hierarchical clustering with different dissimilarity methods*

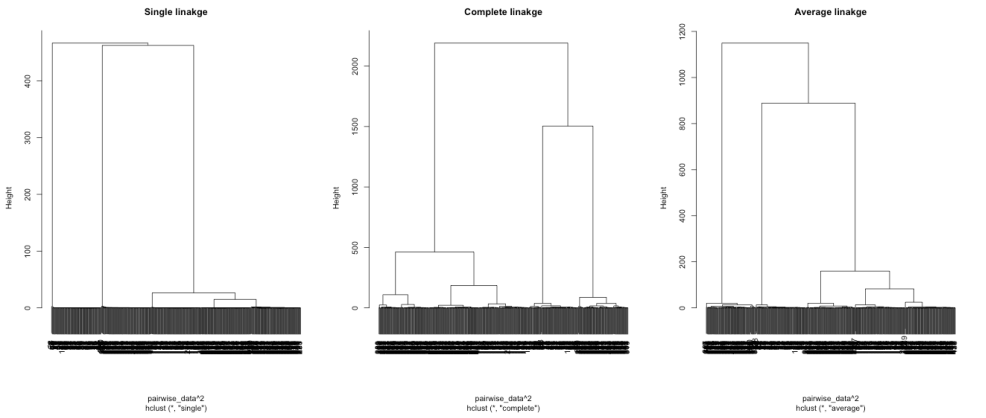
Next, the dendrograms in figure 6 and table 4 show the results from different linkages. First, lets look at the results for the overlapping artificial data set on figure 6.a. The graphs show that single linkage creates an unbalanced dendrogram, and it tends to fuse to the right of the dendrogram, while the others yield a more balanced tree. Second, the well-separated artificial data set can implement any linkage methods since all of the dendrograms in figure 6.b are similar. Finally, figure 6.c demonstrates the results for the real data set. Single linkage again creates an unbalanced dendrogram, where most observations are fused at the bottom, then, the last few branches are merged at a very high value. Also, average linkage tends to fuse on the right of the dendrogram, which make it unbalanced. Complete linkage returns a more balanced dendrogram as predicted. Looking further to the numerical figures in table 4, we can see that complete linkage is well performed compared to others in the Overlapping and Real Data sets.

	Overlapping Data set	Well Separated Data set	Real Data set
Single	25091.781	1082.386	588.2685
Complete	6039.027	1082.386	586.9527
Average	6336.187	1076.458	588.0135

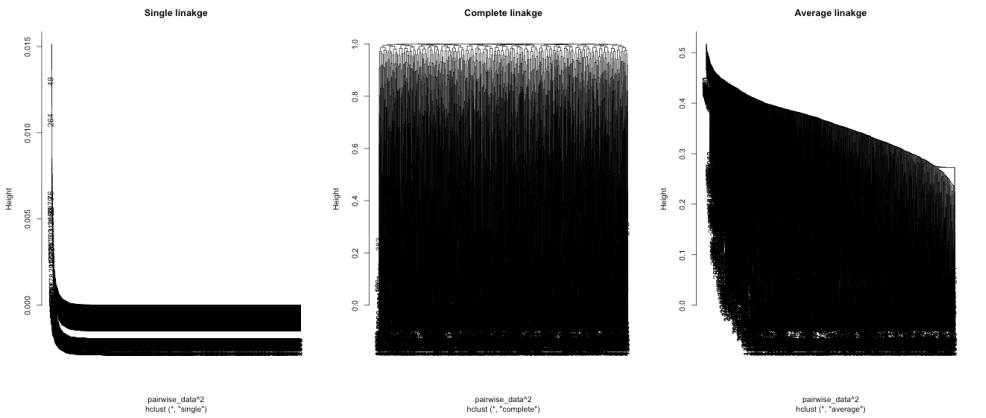
Table 4: *Sum of Square Error of Agglomerative Hierarchical clustering with different linkage methods*



a. *Agglomerative Hierarchical Clustering on Overlapping artificial data set*



b. *Agglomerative Hierarchical Clustering on Well-Separated artificial data set*



c. *Agglomerative Hierarchical Clustering on Real data set*

Figure 6: *Dendrogram of Agglomerative Hierarchical Clustering with different linkages*

Overall, Euclidean distance is the best solution for artificial data sets, while Pearson correlation is the best for real data set while using complete linkage. However, from this experimental, the choice of dissimilarity significantly affects the clustering performance. Thus, it is important to run several different methods and visualise the result, if possible, to have an overview and select the best fit.

- **Number of clusters**

The first experiment is trying to select the right number of clusters. There are four different methods that will be implemented in this project. These are elbow method, gap statistic, Bayesian Information Criterion and Combined Mixture Components for Clustering. According to the results that I had with initialisation and dissimilarity distance, I ran K-means++, Agglomerative Hierarchical and EM for MOG with a sequence of number of clusters, K = {3…15}, using Euclidean distance on artificial data set and Pearson correlation on the real data set.

The first implementation is the **elbow method**. Figure 7 demonstrated the Sum of Square Error for K-means and Agglomerative Hierarchical Clustering and Log Likelihood for Expectation Maximisation for Mixture of Gaussian on the overlapping artificial data set. As mentioned in the background review, the elbow method is easy to apply and visualise, but sometimes there is no elbow, like the SSE of K-means and Agglomerative Hierarchical Cluster, or several elbows in a graph like the log likelihood value of the EM for MOG.

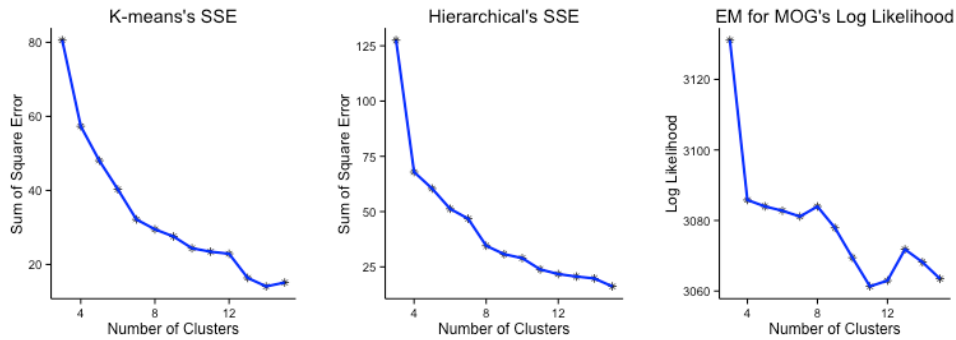
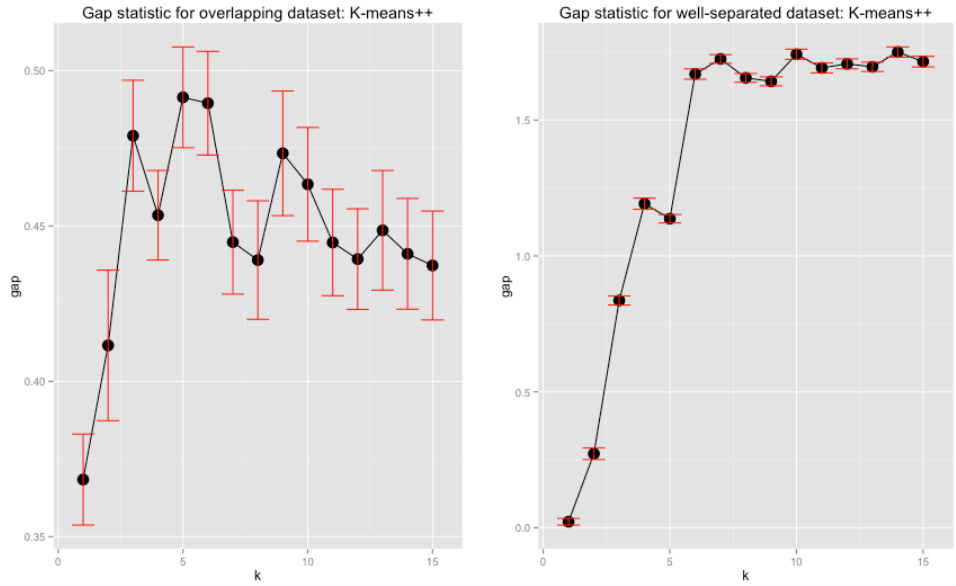


Figure 7: *Cluster validation on Artificial Data set*

The second implementation is to achieve the optimal number of clusters using **Gap statistic** methods. As stated in section 3.1, Gap statistic works extremely great with well-separated data sets, but not with overlapping data sets. Therefore, I ran the test on both overlapping and well-separated artificial data sets in order to prove the above point. I implemented the "clusGap" function, which is developed in R's package "cluster" based on Tibshirani [14]. The settings is as follows:

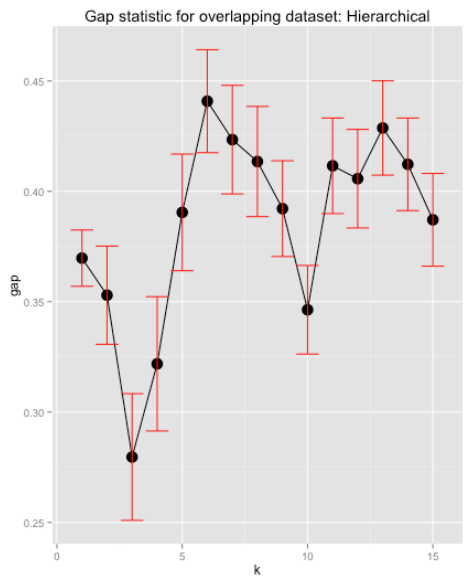
- Functions: K-means, Complete linkage for Agglomerative Hierarchical Clustering and EM for MOG using Euclidean distance.
- 100 Monte-Carlo bootstrap samples
- The maximum number of clusters to consider is 15

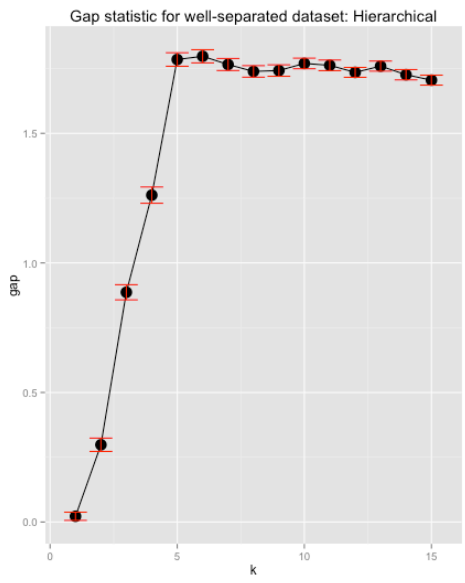
The graphs in Figure 8 present the differences of Gap statistic's behaviour on the different kind of data set. It can be clearly seen that Gap statistic recommends 5 clusters for well-separated data set in K-means and Agglomerative Hierarchical, as they create an angle at 5. Any points after 5 is constant or slightly change. However, for the overlapping data, the gap values are inconsistent and their standard deviations (represented in the error bar) are very large. Therefore, if the data sets are not well separated, Gap statistic is not highly recommended.



a.1. *Overlapping data set* a.2. *Well-separated data set*

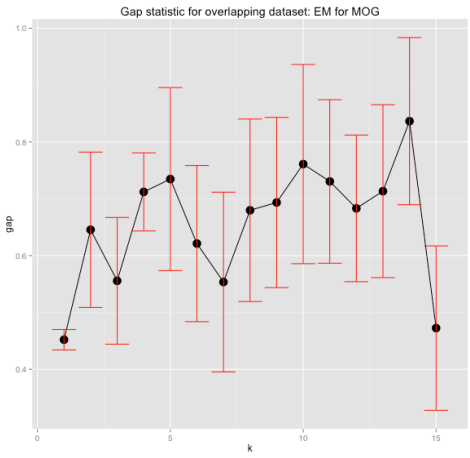
a. *Gap statistic for K-means++*

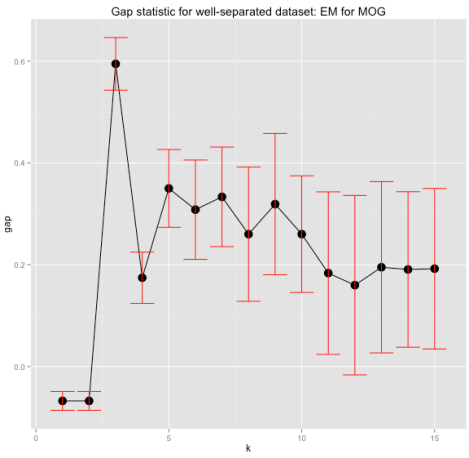




b.1. *Overlapping data set* b.2. *Well-separated data set*

b. *Gap statistic for Agglomerative Hierarchical Clustering*





c.1. *Overlapping data set* c.2. *Well-separated data set*

c. *Gap statistic for EM for MOG*

Figure 8: *Gap statistic*

The third implementation is **BIC**. Due to the expensive calculation in Agglomerative Hierarchical clustering, I only computed the BIC for K-means and EM for MOG. I set up the same experiment procedure as Gap statistic, which ran K-means and EM for MOG for K = {3..5} on Overlapping Artificial using Euclidean distance and Real Data sets using Pearson correlation, and recorded the results for further calculation.

To find BIC value for K-mean, I worked out the log likelihood for each cluster, and then add the penalty parameter from the equation in section 3.1. BIC is the best at k = 3 for artificial data set. Afterward, I used the current function, Mclust, in the package mclust to find the BIC of EM for MOG. The best result is k = 5 for artificial data set. Figure 9 visualises the result of K-means and EM for MOG

clustering from the best BIC. It shows that the original data set and the result from EM are quite similar.

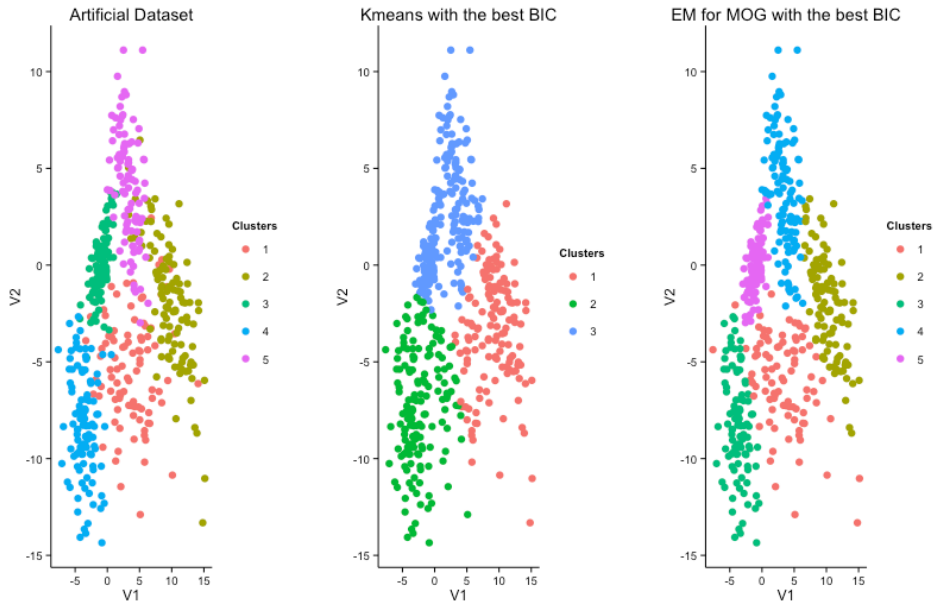
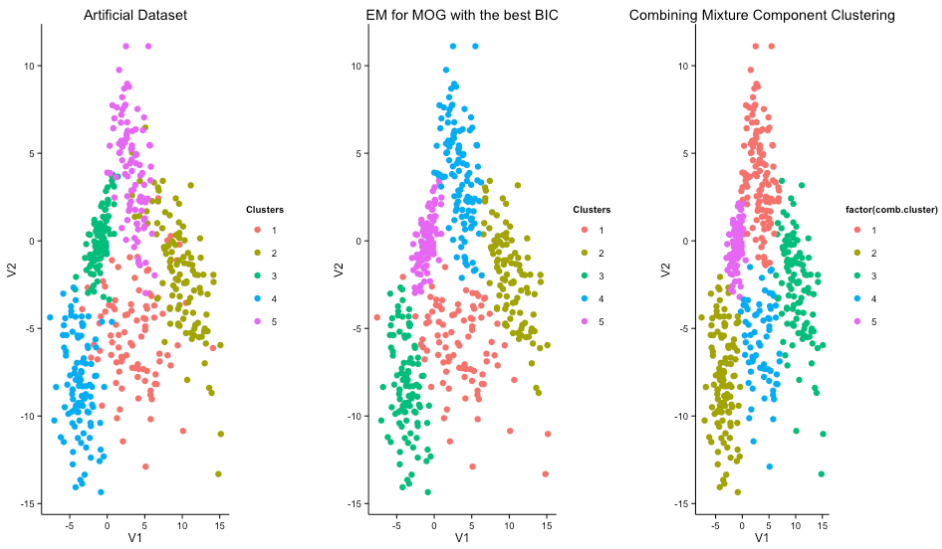


Figure 9*: K-means and EM for MOG's best results with BIC on Overlapping Artificial Data set*

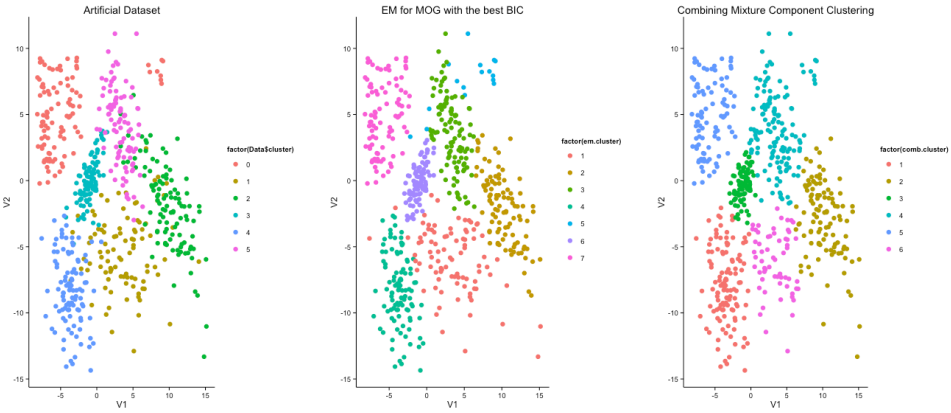
Next, the Combined Mixture Components for Clustering (CMC) was implemented to EM for MOG. The graphs in figure 10.a represent that both BIC and CMC return the same number of clusters with only a slight difference in assigning observations to clusters. This means that BIC and CMC have similar performance for data sets where all clusters are drawing from the same distribution function. As revealed in section 3.1, the main feature of CMC is to prevent the overestimation of the number of clusters when there is an extra non-Gaussian cluster. Hence, I applied the same concept to the artificial data set with an extra cluster. BIC returns 7 clusters and CMC returns 6 clusters. As shown in Figure 10.b, CMC has merged the small cluster into the nearby larger cluster. The table 5 below summaries the numerical review. The log likelihood of BIC is not that different from the CMC, but the Silhouette value is improved when applying those methods to a new data set. Therefore, it is better to use the CMC methods when not all the data is drawn from the same distribution function.

	Overlapping Artificial Data set		Overlapping Artificial Data set with Outlier	
	BIC	CMC	BIC	CMC
Log likelihood	-2911.362	-2934.836	-3519.876	-3564.515
Silhouette	0.387	0.365	0.389	0.487

Table 5: BIC vs CMC on Overlapping and Overlapping with Outlier Artificial Data set



a. *Artificial Data set where all clusters are generated from multivariate Gaussian distribution*



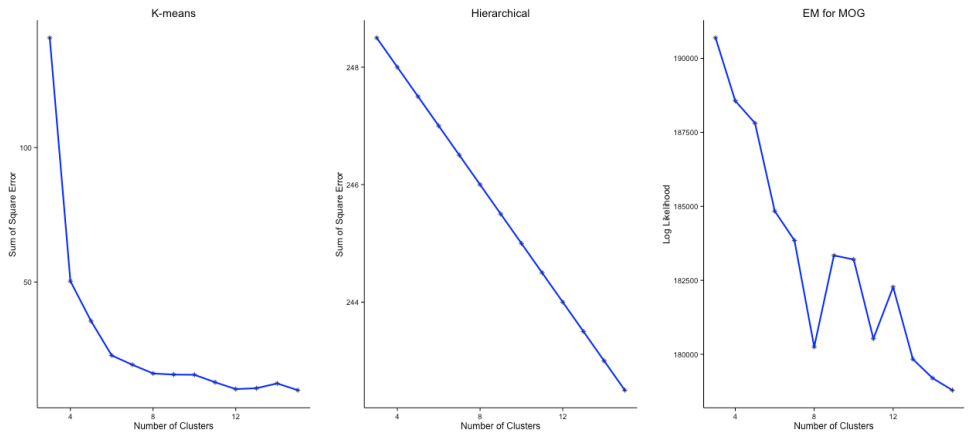
b. *Artificial Data set with an extra non-Gaussian cluster*

Figure 10: *EM for MOG with the best BIC vs Combining Mixture of Gaussian*

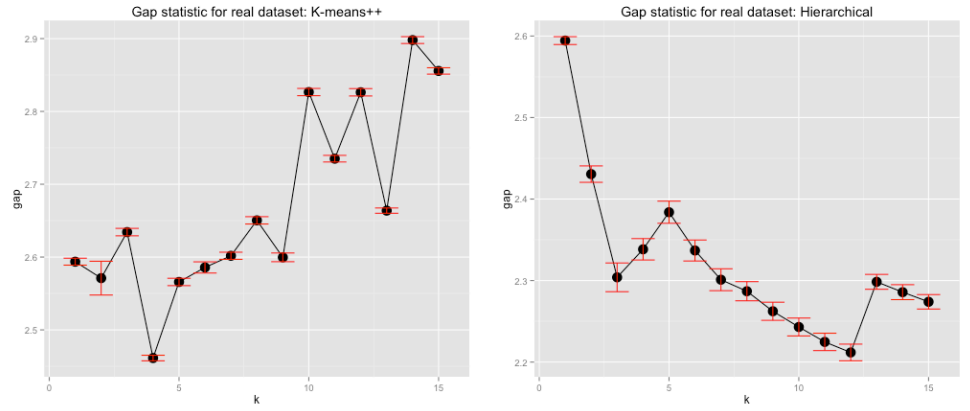
Lastly, I implemented the above methods to the real data set and the results are presented in figure 11. With the elbow method on figure 11.a, K = 8 returns the best result when using K-means, however, there is no conclusion for the other methods. Following is the gap statistic. Since it is computational expensive, it returned the value for K-means and Agglomerative Hierarchical but not EM for MOG. I couldn't find the optimal solutions here since the Gap values are inconsistent and there are many angles in a graph.

The further implementation is on BIC and CMC, the best result is k = 3 with BIC for K-means, k = 10 with BIC for EM for MOG and K = 15 for CMC. Although BIC and CMC are recommended to use in the artificial data set, I think this is not the best solution. When I proceed the next session, Mixture of Gaussian

isn't a suitable clustering algorithm for this real data set. Thus, using the Elbow method with K = 8 for the real data set is a better choice.



a. *Elbow method for K-means, Agglomerative Hierarchical and EM for MOG*



b. Gap statistic method for K-means ++ and Agglomerative Hierarchical Clustering

Figure 11: *Choose the best number of clusters for Real data set using Elbow and Gap statistic methods*

- **Cluster validation**

Last but not least, I experimented with different methods for cluster validation. The following parameters for the clustering algorithm are set:

- Overlapping artificial data set with k = 5 using Euclidean distance and complete linkage.
- Real data set with k = 8 using Pearson correlation distance and complete linkage.

In particular, the clustering performance on both artificial and real data sets will be judged based upon internal criterion, including Sum of Square Error, Silhouette width and Pearson Gamma. Additionally, with the availability of labels in the artificial data set, I have also computed the external criterion, which are entropy and rand index. All calculations use the current function, "cluster.stats".

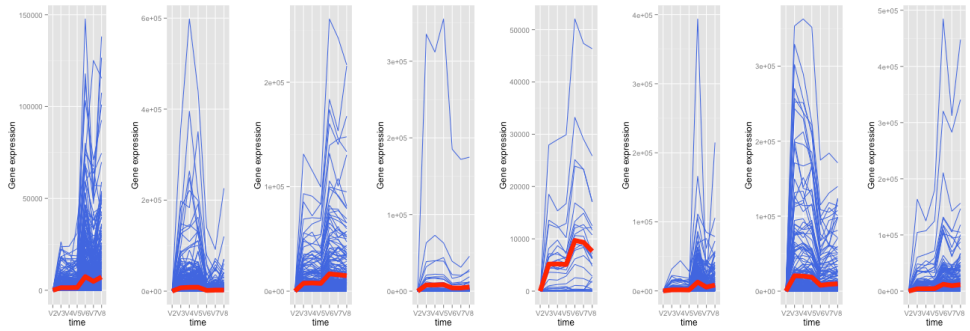
Even though the EM for MOG is based on Log Likelihood, I have recorded the SSE, in order to have an equivalent comparison to other methods.

		Overlapping Artificial Data set			Real Data set		
		K-means	Hierarchical	EM for MOG	K-means	Hierarchical	EM for MOG
Internal Criterion	SSE	4842	6039	5252	70.93	1215	953
	Silhouette Width	0.407	0.379	0.415	-0.01	- 0.01	-0.7
	Pearson Gamma	0.576	0.584	0.562	0.48	-0.001	0.59
External Criterion	Entropy	1.6	1.47	1.6			
	Rand Index	0.496	0.556	0.754			

Table 6: *Overall cluster validation figures on Artificial and Real Data sets*
Note: The best value is highlight in yellow

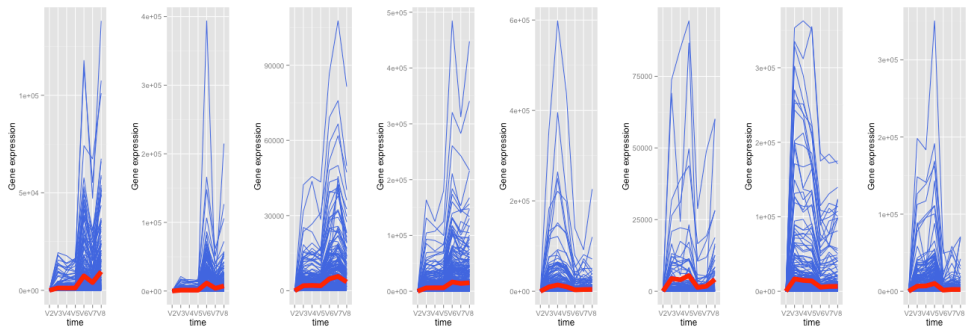
Each validation index represents a different characteristic of the data, so it is not sufficient to evaluate the cluster quality according to one index. To recap, the cluster is better with the lower SSE, the higher Silhouette Width, the lower Pearson gamma, the lower entropy and the higher Rand index. Moving ahead, as illustrated in table 6, on the experimental with overlapping artificial data set, K-means's SSE gave the smallest value while EM for MOG returned better indexes on the others. Also while comparing K-means and EM for MOG, we can see the difference in SSE and Silhouette width is not too large, and the entropy is the same. However, the Rand index in EM for MOG is much higher than K-means. Therefore, we can conclude that EM for MOG is a better solution for this particular data set. Looking at the figures for real data set, K-means seems to be a better result when its SSE is lower and Pearson Gamma is higher than others. Because the real data set is using Pearson correlation as a dissimilarity measurement, Silhouette width is not the right choice to compare as it is purely designed for distance method. According to Montanari and Anderlucci [23], "Pearson Gamma gives a good approximation of the dissimilarity structure in the sense that when observations are in different clusters they should strongly be correlated with large dissimilarity". Therefore, K-means is a more appropriate method for this gene expression data set.

Since K-means using Pearson correlation distance is selected to be a clustering algorithm for the real data set, I took a further step to compare the performance of K-means and K-medoids with K = 8. As shown in figure 12.a, K-means returns some dense clusters, for instance cluster 1 and 2 have 633 and 949 observations respectively. Besides, the sparse cluster like cluster 4, which only has 89 observations, has a similar shape to cluster 2. Now, considering the graphs in 11.b, the numbers of observations in every cluster are similar and their means are also in different shapes. From numerical point of view, the K-medoids' SSE and Pearson Gamma are lower than K-means. In details, the K-medoids' SSE is only 16.56, so it is really low and near to 0 for 3006 observations. However, K-medoids consumed a lot of time to execute and compute the final output. Thus, if we are not concerned about time, K-medoids is a better choice.



a. *K-means with K = 8 on the real data set*

SSE = 70.93 and Pearson Gamma = 0.48



b. *K-medoids with K = 8 on the real data set*

SSE = 16.56, Pearson Gamma = 0.46

Figure 12: *K-means vs K-medoids on the real data set*

5 Conclusion

The experiments above are on both the artificial and real gene expression data sets using K-means, Hierarchical clustering and EM for MOG. The final result is using EM for MOG with Euclidean distance and BIC to find the best number of clusters for the artificial data set. The real data works well with K-means, or even better with K-medoids using Pearson correlation with the elbow method to select the best cluster. As all algorithms are starting with random parameters, the numerical figures in this report could be changed when you re-run the program. However, the fundamental concept won't change.

The first intention of the project was determining how to choose the right initialisation parameters for K-means and Expectation Maximisation for Mixture of Gaussians (EM for MOG). From the experiment, I discovered that K-means++ is a better solution than the others as it randomly selects a set of observations within the data set based on a specific rule of probability, instead of generating a random vector or assigning a random set of observations to be initial cluster centroids.

The second approach was distance and similarity measures. The quality of clustering is significantly affected by the (dis)similarity measure. Still, there is no specified right or wrong answer because the data's structure is unknown. Therefore, it is very important to run several methods and visualise the result to get the best fit.

The third implementation was how to choose the optimal number of clusters. Generally, the elbow method is quick and easy, but does not work all the time. Gap statistic is slow but can apply to any clustering algorithm and compares the change in distortion value; it also only works well with well-separated data. BIC is computational expensive but a more efficient solution for any kind of data, especially when using with EM for MOG. Also, using the proposal from Combined Mixture Components for Clustering paper solves the over-estimating limitation of BIC.

The last application was about cluster validation. Internal criterion such as SSE and Silhouette Width work well with distance measure, while Log likelihood and Pearson Gamma are used with density or similarity measure. When the previous knowledge of data is available, external criterion, such as entropy and Rand index are good tools to validate the clustering.

Finally, to choose the best clustering algorithms, the time and quality of clusters should be taken into consideration. K-means can give a swift result but K-medoids is considered to be more robust. Yet neither of them can find the global optimal at convergence. Hierarchical clustering is quick and easy to visualise in a dendrogram. It also does not require initialisation parameters or number of clusters at the start. EM for MOG takes more time to seek the final cluster, but it assumes an underlying density function for data, and reduces the influence of outliers. Additionally, a good choice of parameters makes a difference in clustering

performance. The clustering should be repeated several times in order to get the best result, because K-means and EM for MOG start with random parameters.

This report was not concerned about reducing data's dimensions to make the computation quicker and more efficient, as well as improve visualisation. Further studies may focus on those. The code I wrote in R for this report can't be used for the large-scale data sets, because it is not designed for that level of scale. A better implementation either on the algorithms or a different programming system, would improve the clustering performance.

6 Professional issues

As a student of Computational Finance with an ambition of a becoming data scientist, it is important for me to commit to ethical and professional conduct, as well as be responsible for my work and projects, as it will has major effects on my future stakeholders.

There are professional bodies such as the British Computer Society (BCS) and the Association for Computing Machinery (ACM) that have laid out a code of conduct for IT professionals, including myself, to follow. Having read the code and looking into this subject, I aware of the following key issues, which I encounter in my project.

- **Management**

Before starting my project, I needed to ensure I have accessed to appropriate books, research papers and journals. My priority resource is library, and as a student of Royal Holloway, University of London, I am being able to enter a wide range of libraries around London. Whenever I can't find enough information from there, I conducted Internet search to fulfil my requirements.

While conducting my project, I was working 2 days per week at my placement company, thus only leaving 5 days a week to carry out my project and maintain a work/life balance. Therefore, I have to use my time management skill effectively. At the start of the project, I have had to plan carefully for anything that may affect the project, creating a timeline and goals for each part of the work. However, my project is not purely research but also involves coding, thus, I needed some contingency time in case any bugs arose in my program.

- **Plagiarism**

For a project of this scale, deep knowledge of the subject and topic is required. Thus, I have thoroughly researched by reading books, research papers and journals, as well as other relevant resources. Avoiding plagiarism is one of the concerns that I have to deal with. As placed in the project handbook, plagiarism is the practice of taking someone else' work or ideas and using them as one's own, and it is a serious offence in any academic or research project. Since I'm using multiple sources of information and data in my project, I ensure to include and acknowledge them in the context of my project, as well as writing them down in my bibliography. Specifically, I conduct MLA style for my citations.

Besides, there are many different types of code that I have used in my project. Most of them are written or modified by myself; however, there were times that I used the code of others to aid with my code and the visualisation of my graphs. For example, I used Multiplot code, which was developed by Wiston Chang [28] and was not a built-in function in any R packages. Normally, I could plot only one graph at a time with the ggplot2 package, but with the help of Multiplot, I could plot as many graphs as I wanted in a single instance. This was a

great help, as it allowed me to compare and contrast the different methods easier. To ensure that the owners of Multiplot and other code that I used in my project are acknowledged, I ensure they are in my reference list.

- **Licensing**

I acknowledge that to conduct my project ethically, I will have to either purchase a proper license or use open source software. A definition about open source software from wiki page [28]: "Open-source software is software whose source code is published and made available to the public, enabling anyone to copy, modify and redistribute the source code without paying royalties or fees". On the other hands, a software licensing is required when using proprietary software. It is defined in [29]: "is licensed under legal right of the copyright holder, with the intent that the licensee is given the right to use the software only under certain conditions, and restricted from other uses, such as modification, sharing, studying, redistribution, or reverse engineering. Usually the source code of proprietary software is not made available". Nowadays, many people illegally use the proprietary software by hacking or downloading from torrents. This action is intolerant and consider as stealing other intellectual property.

For this project, I have researched about available data analysis software. There is a wide range of available software for data analysis, such as R, Matlab, Python and Weka. However, I purposely use R because it is open source and is one of the most comprehensive statistical analysis packages available, with over 4800 pre-built packages. I mainly code on RStudio, which is a free and open source IDE for R. Besides, my report is written on Microsoft Word, which is a licensed version that available for Royal Holloway students.

The above are some main professional issues that I encountered during my project. However, I commit to act ethically and professionally under the code of conduct from BCS and ACM as a professional data scientist.

7 Self-assessment

While carrying out my project, I encountered a number of problems and issues; however, none of these were things that I could not overcome. The first issue that I faced was to underestimate the amount of work involved, and how long it would actually take to complete everything. When I first started the project, I didn't take into account the placement report. Nevertheless, I planned enough contingency time to successfully complete the project, as well as the placement report. I also misjudged the level of difficulty required to code in R, thus, I struggled a bit to get used to the language at first. Luckily, with the experience in programming that I earned from my placement, I quickly became familiar with R's language.

The most difficult problem I encountered was debugging the EM for MOG function that I wrote. When I first finished coding, the EM for MOG algorithm was running perfectly on the artificial data sets. However, when it came to the real data set, the code kept failing after the first iteration. I spent 3 days debugging every single line in my code. I figured out the problem occurred because the random initialisation parameters generated a Gaussian distribution, which was too far from any data points. It led to the failure when it tried to compute the e-step, where the probability density function (pdf) was 0, and the sum of all pdf is also 0. I realised that this error happened because I wasn't be able to see the whole picture that may occur from the algorithm. Therefore, I decided to go back to the basics to extend my knowledge and fully understand the algorithm before re-coding. Even though I was behind my original plan, the quality of my report was maintained. I also spent some more time to reschedule and adjust my timeline to make sure everything else stayed on track.

From one of my placement projects, I had a chance to experience executing 240 GB of data with R. Therefore this helped me to plan my time efficiently when executing my code on the real data set. Instead of wasting time waiting for the result, I jumped into another tasks while the code was executing. Besides, I had a project about geospatial clustering where I had to group the closest postcodes together, and draw a minimum spanning tree within the clusters. In this project, I applied some simple algorithms, such as K-means and Hierarchical Clustering, thus, I only needed a short time to quickly review these algorithms.

Overall, I think I have managed to complete the project as planned, and that is a great achievement. Although I encountered some difficulties, I believe my work is precise and useful for academic as well as practical use.

8 How to use my project

All of my codes will be run using R. The sequence of codes to execute my program is below. For each section, it is suggested that you keep the same R session.

```
#  Installing R:
https://cran.r-project.org/doc/manuals/r-release/R-admin.html
Note: the code has been tested in R version 3.0.3 on Platform: x86_64-apple-darwin10.8.0 (64-bit)
#  Installing necessary packages in R by executing the following command:
install.packages(c("data.table", "plyr", "mclust","MixSim","ggplot2",  "dplyr", "pdist", "corpcor", "mvtnorm", "proxy", "cluster", "clusterSim", "fpc", "reshape2","RmixmodCombi","clValid", "som"))
```

Section 2 – Background Research

```
#  Changing working directory:
setwd("/Project/Code") ## Or anywhere that you locate the code
#  Load R script for example 1: K-means process
source("Example1.R")
#  Load R script for example 2: K-medoids process
source("Example2.R")
#  Load R script for example 4: Dendrogram for Hierarchical clustering
source("Example4.R")
```

Section 4 – Experiment

```
#  Changing working directory:
setwd("/Project/Code") ## Or anywhere that you locate the code
#  Generate artificial data sets and load real data set
source("LoadData.R")
#  Initialisation methods
source("InitialisationMethods.R")
#  (Dis)similarity methods
source("DistanceMethods.R")
#  Selecting number of clusters methods
source("NumberOfCluster.R")
#  Cluster Validation
source("ClusterValidation.R")
#  K-means vs K-medoids
source("KmeansVSKmedoids.R")
```

9 References

1. Everitt, Brian, Sabine Landau, and Morven Leese. *Cluster Analysis*. 4th ed. London: Arnold, 2001. Print.
2. Hastie, Trevor, Robert Tibshirani, and J. H Friedman. *The Elements Of Statistical Learning: Data Mining, Inference, and Prediction*. Canada: Springer, 2002. Print.
3. Bishop, Christopher M. "Mixture Models and EM." *Pattern Recognition and Machine Learning*. New York: Springer, 2006. Print.
4. Witten, I. H., Eibe Frank, and Mark A. Hall. "Algorithms: The Basic Methods." *Data Mining: Practical Machine Learning Tools and Techniques*. 3rd ed. Burlington: Morgan Kaufman, 2011. Print.
5. "Data Mining Algorithms In R/Clustering/CLARA." - *Wikibooks, Open Books for an Open World*. Web. 29 July 2015. https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/CLARA
6. Kodinariya, Trupti M., and Prashant R. Makwana. "Review on Determining Number of Cluster in K-Means Clustering." *International Journal of Advance Research in Computer Science and Management Studies* Volume 1.Issue 6, November 2013: 90-95. Web. 30 July 2015. http://www.ijarcsms.com/docs/paper/volume1/issue6/V1I6-0015.pdf
7. James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. 0. Print.
8. Murphy, Kevin P. *Machine Learning a Probabilistic Perspective*. Cambridge, Mass.: MIT, 2012. Print.
9. Reynolds, Douglas. "Gaussian Mixture Models." MIT Lincoln Laboratory, USA. Lecture. Wed. 30 July 2015. https://www.ll.mit.edu/mission/cybersec/publications/publication-files/full_papers/0802_Reynolds_Biometrics-GMM.pdf
10. Bilmes, Jeff A. "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models." *International Computer Science Institute* 4.510 (1998): 126. Web. 20 Aug 2015. http://lasa.epfl.ch/teaching/lectures/ML_Phd/Notes/GP-GMM.pdf
11. Chen, Scott Shaobing, and Ponani S. Gopalakrishnan. "Clustering via the Bayesian information criterion with applications in speech recognition." *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference* on. Vol. 2. IEEE, 1998. Web. 15 Aug. 2015. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=675347&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D675347

12. Baudry, Jean-Patrick, Adrian E. Raftery, Gilles Celeux, Kenneth Lo, and Raphaël Gottardo. "*Combining Mixture Components for Clustering*." Journal of Computational and Graphical Statistics 19.2: 332-53. Web. 13 Aug. 2015. https://www.stat.washington.edu/raftery/Research/PDF/Baudry2010.pdf.
13. Steele, Russell J., and Adrian E. Raftery. "Performance of Bayesian Model Selection Criteria for Gaussian Mixture Models." *Technical Report No. 559*. Web. 13 Aug. 2015. https://www.stat.washington.edu/research/reports/2009/tr559.pdf.
14. Tibshirani, Robert, Guenther Walther, and Trevor Hastie. "Estimating the Number of Clusters in a Data Set via the Gap Statistic." *Journal of the Royal Statistical Society: Series B (Statistical Methodology) J Royal Statistical Soc B*: 411-23. Web. 8 Aug. 2015. http://web.stanford.edu/~hastie/Papers/gap.pdf.
15. Halkidi, Maria, Yannis Batistakis, and Michalis Vazirgiannis. "On clustering validation techniques." Journal of Intelligent Information Systems 17.2 (2001): 107-145. Web. 20 Aug 2015. http://web.itu.edu.tr/sgunduz/courses/verimaden/paper/validity_survey.pdf
16. "Cluster Validation." Web. 28 Aug. 2015. http://www.cs.kent.edu/~jin/DM08/ClusterValidation.pdf .
17. Rendon, Eréndira, Itzel M. Abundez, Citlalih G. S. D. Zagal, Alejandra Arizmendi, Elvia M. Quiroz, and Elsa Arzate h. "A Comparison of Internal and External Cluster Validation Indexes." *Applications of Mathematics and Computer Engineering*. Web. 20 Aug. 2015. http://www.wseas.us/e-library/conferences/2011/Mexico/CEMATH/CEMATH-26.pdf .
18. Manning, Christopher D., and Prabhakar Raghavan. *Introduction to Information Retrieval*. New York: Cambridge UP, 2008. Web. 15 Aug 2015. http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html.
19. Maimon, Oded, and Lior Rokach. "Clustering Methods." *Data Mining and Knowledge Discovery Handbook*. New York: Springer, 2005. Web. 16 Aug 2015. http://www.ise.bgu.ac.il/faculty/liorr/hbchap15.pdf.
20. Abbi, Revlin, Elia El-Darzi, Christos Vasilakis, and Peter Millard. "Analysis of Stopping Criteria for the EM Algorithm in the Context of Patient Grouping According to Length of Stay." *2008 4th International IEEE Conference Intelligent Systems*: 3-9. Web. 29 Aug. 2015. http://westminsterresearch.wmin.ac.uk/5640/1/Abbi_El_Darzi_Vasilakis_Millard_2008_as_published.pdf .
21. Hamerly, G. and Elkan, C. 2002."Alternatives to the k-means algorithm that find better clusterings".*Proceedings of the eleventh international conference on Information and knowledge management* (CIKM). Web. 29 Aug. 2015. http://charlotte.ucsd.edu/users/elkan/cikm02.pdf

22. Biernacki, C., G. Celeux, and G. Govaert. "Assessing a Mixture Model for Clustering with the Integrated Completed Likelihood." *IEEE Transactions on Pattern Analysis and Machine Intelligence IEEE Trans. Pattern Anal. Machine Intell.*: 719-25.
23. Montanari, Angela, and Laura Anderlucci. "Cluster Analysis." Web. 2 Sept. 2015. http://www2.stat.unibo.it/montanari/Didattica/Multivariate/CA.pdf.
24. Luo, Zhiyuan. "Unsupervised Learning." Data Analysis Class. Royal Holloway, Egham. 22 Nov. 2013. Lecture.
25. Kaur, Manpreet, and Usvir Kaur. "Comparison Between K-Mean and Hierarchical Algorithm Using Query Redirection." *International Journal of Advanced Research in Computer Science and Software Engineering* 3.7. Web. 24 Apr. 2015. <http://www.ijarcsse.com/docs/papers/Volume_3/7_July2013/V3I7-0565.pdf>.
26. Benaglia, Tatiana, et al. "mixtools: An r package for analyzing finite mixture models." Journal of Statistical Software 32.6 (2009): 1-29. ftp://www.r-project.org/pub/R/web/packages/mixtools/vignettes/mixtools.pdf
27. Chang, Winston. "Graphs with Ggplot2." *R Graphics Cookbook*. Beijing: O'Reilly, 2013. Print.
28. *Wikipedia*. Wikimedia Foundation. Web. 13 Sept. 2015. https://en.wikipedia.org/wiki/Open-source_software.
29. *Wikipedia*. Wikimedia Foundation. Web. 13 Sept. 2015. https://en.wikipedia.org/wiki/Proprietary_software.