An Investigation of Clustering Techniques: *k*-means and GMM

Emma Grossman

1 Introduction

The *k*-means algorithm and Gaussian Mixture Models estimated with the Expectation Maximization algorithm are both clustering techniques used to identify relationships between variables. This paper documents the theory behind the two methods, then explores the commonly used functions in R that implement them. Examples of *k*-means and Gaussian Mixture Models are explored in a case study using the `penguins` data set from the `palmerpenguins` package. Then, data were simulated to highlight scenarios in which both methods performed well, as well as scenarios where only one performed well. A discussion on future work relating to this topic concludes this paper.

These two clustering methods are notable because the *k*-means algorithm is more widely known and commonly used, but Gaussian Mixture Models tend to perform better on data structures with groupings that are less distinct or have different variances.

All of the code for this project can be viewed at https://github.com/emmaleda/MS-project-kmeans-and-GMM

2 Background and Methods

2.1 Clustering

Clustering is an unsupervised learning technique used to uncover relationships between variables by classifying existing data into homogeneous groups. Clusters will have high heterogeneity between clusters and high homogeneity within clusters, meaning data within a cluster are similar to one another but very different from data within other clusters.

By creating clusters, new information is generated and the clusters can be used to understand more about the population. For example, suppose a company has data about their customers: purchase history, age, gender, etc. and they want to group customers based on how they might respond to an ad campaign. This could be achieved with clustering. Each customer would be placed into a cluster with other customers who have similar attributes and this cluster would inform the targeted ad campaign (Lantz 2019).

2.2 *K*-means

2.2.1 Algorithm

K-means clustering assumes that points close to each other are more similar, so it groups data points based on distances between observations. The algorithm randomly chooses k rows of data from a data set as the starting cluster centroid points. Then, the remaining data points are assigned to the closest centroid using a distance metric, generally the Euclidean distance. After all points have been grouped into clusters, the centriod points are recalculated as the average of all the points in the

corresponding cluster. Following this, the distances of the points from the centroids are recalculated. Points in which the closest centroid has changed are reassigned to the cluster that contains the closer centroid. This process is repeated until no points are reassigned; the clusters have converged.

This algorithm is highly sensitive to the starting rows that are randomly chosen. If one of the rows that is chosen is an outlier, the algorithm might create a cluster of one point that only contains that outlier. This can be resolved by selecting many starting points and running the *k*-means algorithm for each; the best model is selected from resulting models.

Another issue is choosing how many clusters to create. Ideally, prior knowledge would aid in the process of making an educated guess, but it is not always the case that prior knowledge is available. There are several options if k is unknown: (1) set k equal to the square root of $n/2$, but this can result in too many clusters if n is large, (2) use the elbow method, which plots the explained variation against the number of clusters and chooses the elbow point. Intuitively, this means selecting k at the point in which adding more clusters results in only a small improvement in the fit of the model. Choosing a large k will improve the homogeneity within the clusters but will likely overfit the data (Lantz 2019).

With *k*-means it is important to standardize each feature of a data set. This is because the algorithm decides cluster membership based only on raw distance, so a feature with a particularly large variance compared to other features is likely to become separated into too many clusters (Lantz 2019). Constraining the variance to be equal to one by standardizing each feature reduces this effect.

2.2.2 Implementaiton in R: kmeans{stats}

The *k*-means function commonly used in R is `kmeans` from the `stats` package. The two necessary arguments are `x`, which is a matrix of numeric data, and `centers`, which is the desired number of clusters to find. Another argument of note is the `nstart` argument. As mentioned earlier, *k*-means is very sensitive to the starting rows that are randomly chosen for the centroids; `nstart` combats this by allowing the user to ask for more than one starting row (the default of `nstart` is one). The `kmeans` function then runs its algorithm for each randomly chosen starting row, and the best model is returned. The best model is the one in which the sum of squared distances from observations to their assigned cluster centroid is minimized (R Core Team 2019).

2.3 Gaussian Mixture Models

2.3.1 Algorithm

The key difference between *k*-means and Gaussian Mixture Models (GMM) is the assumption of GMM that the data are normal or multivariate normal. The normality assumption allows for a measure of the variance and covariance between covariates within a cluster, unlike *k*-means in which points are assigned based on distance from the closest centroid only and spread is not taken into account.

A mixture model is a probabilistic model used to identify previously unclassified subgroups within a population. The number of subgroups is finite and the goal of creating a mixture model is to estimate the parameters associated with each subgroup. Once subgroups are identified, they are generally used for statistical inference.

Each individual subgroup has an associated mixing probability to account for groups with unequal size. To extend the earlier example, if we were trying to create subgroups based on age, we would not want to assume that there are an equal number of children and adults shopping at a store. There will be far more adults making purchases than children, so it would be inadvisable to assume that there is a fifty-fifty chance that a customer is a child. Rather, we would want to lower that probability to a more likely number. Thus, mixing probabilities take into account that the number of data points in each subgroup may not be equal (Bishop 2006).

A Gaussian Mixture Model is a specific mixture model that assumes the subgroups follow a multivariate normal distribution. The mathematical notation to represent this is:

$$P(X|\mu, \Sigma, \pi) = \sum_{k=1}^{K} \pi_k N(X|\mu_k, \Sigma_k)$$

Where π_k is the mixing probability for cluster k, and $N(X|\mu_k, \Sigma_k)$ is the multivariate Gaussian density for the k^{th} cluster with mean vector μ_k, and variance-covariance matrix Σ_k. K is the number of components, or clusters in the model.

To estimate the parameters of the Gaussian Mixture Models from the data, the Expectation Maximization (EM) algorithm is used to find the maximum likelihood estimates and it works with the same iterative framework as *k*-means. There is an initialization process in which the starting values are chosen for the parameters; next, points are assigned to a cluster, and then the values of the parameters are recalculated based on the points in each cluster. The latter two steps are repeated until convergence.

Rather than assign a hard label such as in *k*-means, where each data point only belongs to one

cluster for any iteration, GMM assigns soft labels: each point belongs to all clusters at once, with an associated probability for the point's membership in each cluster. This is possible because the data are assumed to be normal or multivariate normal and because of Bayes' Rule. Bayes' Rule states that the probability of the i^{th} observation being in cluster k_j based on its observed value can be written in terms of the probability of the observed value, assuming it is in cluster k_j and the mixing proportion.

The first step in the GMM algorithm is initialization. Common practice is to first implement *k*-means on the data set and use the assigned clusters to generate starting values for the parameters. After initial values have been found, the next step (the E step of the EM algorithm) is to cluster the data by determining the probability that a data point belongs to a cluster *k*. This is accomplished using Bayes' Rule:

$$P(x_i \in k_j | x_i) = \frac{P(x_i | x_i \in k_j) * P(k_j)}{P(x_i)}$$

$P(x_i|x_i \in k_j)$ is the posterior probability, $N(X|\mu_k, \Sigma_k)$; $P(k_j)$ is the mixing proportion, π_k, and $P(x_i)$ is the probability density function of a GMM, $\sum_{k=1}^{K} \pi_k N(X|\mu_k, \Sigma_k)$.

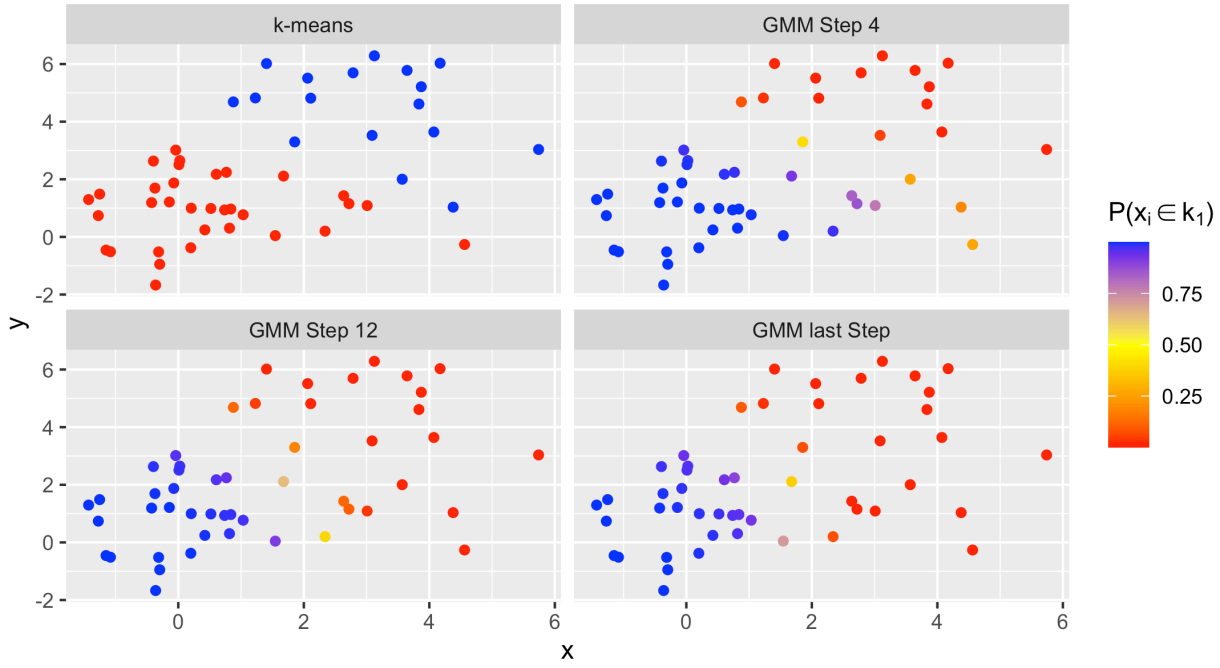
The resulting probability is our "soft cluster" assignment. If we have a bivariate GMM and the probabilities for a particular point are 0.25 for Cluster 1 and 0.75 for Cluster 2, then the point belongs 25% to Cluster 1 and 75% to Cluster 2.

Now that we have the soft cluster assignment, which is the posterior probability for each observation, we can re-estimate the parameters for each cluster. The parameters are recalculated by replacing N_k, the number of observations in the k^{th} component, with the posterior probability $P(x_i|x_i \in k_j)$, in our parameter estimates (the M step of the EM algorithm) (Chan 2016).

Parameter	Initial Estimates	M-step Estimates
μ_k	$\frac{\sum_i^{N_k} x_{i,k}}{N_k}$	$\sum_i^N \frac{P(x_i \in k_j \mid x_i) x_i}{P(x_i \in k_j \mid x_i)}$
Σ_k	$\frac{\sum_i^{N_k} (x_{i,k} - \mu_k)^2}{N_k}$	$\sum_i^N \frac{P(x_i \in k_j \mid x_i)(x_{i,k} - \mu_k)^2}{P(x_i \in k_j \mid x_i)}$
π_k	$\frac{N_k}{N}$	$\frac{\sum_i^N P(x_i \in k_j \mid x_i)}{N}$

After the parameters are re-estimated with the M step, the E step is repeated with the new parameters, generating new posterior probabilities, then the M-step recalculates the parameters with the updated posterior probabilities, and so on until convergence.

We can see this process visually in the graph below.



Since it is common to use *k*-means as a starting point for GMM, the first graph shows how *k*-means allocated the observations, which were simulated to visualize the EM process. In the *GMM Step 4* graph, which is four iterations of both the E and M steps, we can see that the EM algorithm introduced the soft clusters – some of the points have become orange, yellow and light purple. In the next graph, *GMM Step 12*, we can see that the EM algorithm has classified some points as blue

while other points show more uncertainty in their cluster grouping, particularly the observations that are yellow and purple. Finally, in the *GMM last Step* graph, most of the points belong to one of the two groups, though there are a few left in the middle of the graph where the soft clustering is still apparent. Once convergence is reached, the *k* probabilities (produced by the soft clustering) for each observation are assessed, and the cluster that produces the largest probability determines final hard cluster membership for the observation.

2.3.2 Mclust{mclust}

The `Mclust` function from the `mclust` package is a commonly used function to fit Gaussian Mixture Models to both univariate and multivariate data.

In the univariate case, the `mclust` package fits two models, one in which the variances are constrained to be equal across all clusters and the other in which variances are allowed to differ between the clusters, then compares the BIC values and chooses the best model.

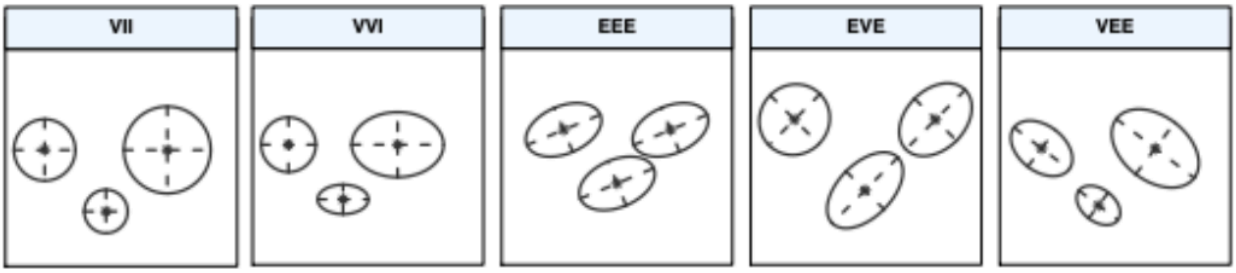
For the multivariate case, covariances are determined in addition to variances, and thus provide more flexibility for the structure of each cluster. Fourteen total models result from constraining aspects of the variance-covariance matrix. The variance-covariance matrix for cluster k is represented using the form $\Sigma_k = \lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T$. λ_k determines the volume of each cluster: if there is only one λ_k, i.e. $\lambda_k = \lambda$, then all of the clusters will have the same volume; if λ_k is allowed to vary, clusters can have different volumes. $\mathbf{A}_k$ is a diagonal matrix that controls the shape of each cluster with the requirement that det($\mathbf{A}_k$) = 1. If constrained to be equal across clusters, $\mathbf{A}_k$ ensures that each cluster has the same shape, e.g. they are all circular. If allowed to vary, then each cluster can have a different shape. Lastly, $\mathbf{D}_k$ determines the orientation of each cluster, so whether the

clusters are aligned with the coordinate axes, not aligned with coordinate axes, or allowed to vary in their orientation; $\mathbf{D}_k$ is an orthogonal matrix. $\mathbf{A}_k$ and $\mathbf{D}_k$ are both square matrices with dimensions equal to the number of covariates.

For example, one possible model is called "VEE" which indicates clusters that are "ellipsoidal, equal shape and orientation". The variance-covariance matrix is of the form $\Sigma_k = \lambda_k \mathbf{DAD}^T$: the volumes are allowed to vary by cluster but the shapes and the orientations are constrained to be the same for all clusters (Scrucca et al. 2016).

Below are a few of the model names and their specifications, along with some images to visualize:

Model	Σ_k	Distribution	Volume	Shape	Orientation
VII	$\lambda_k I$	Spherical	Variable	Equal	–
VVI	$\lambda_k \mathbf{A}_k$	Diagonal	Variable	Variable	Coordinate axes
EEE	$\lambda \mathbf{DAD}^T$	Ellipsoidal	Equal	Equal	Equal
EVE	$\lambda \mathbf{DA}_k\mathbf{D}^T$	Ellipsoidal	Equal	Variable	Equal
VEE	$\lambda_k \mathbf{DAD}^T$	Ellipsoidal	Variable	Equal	Equal

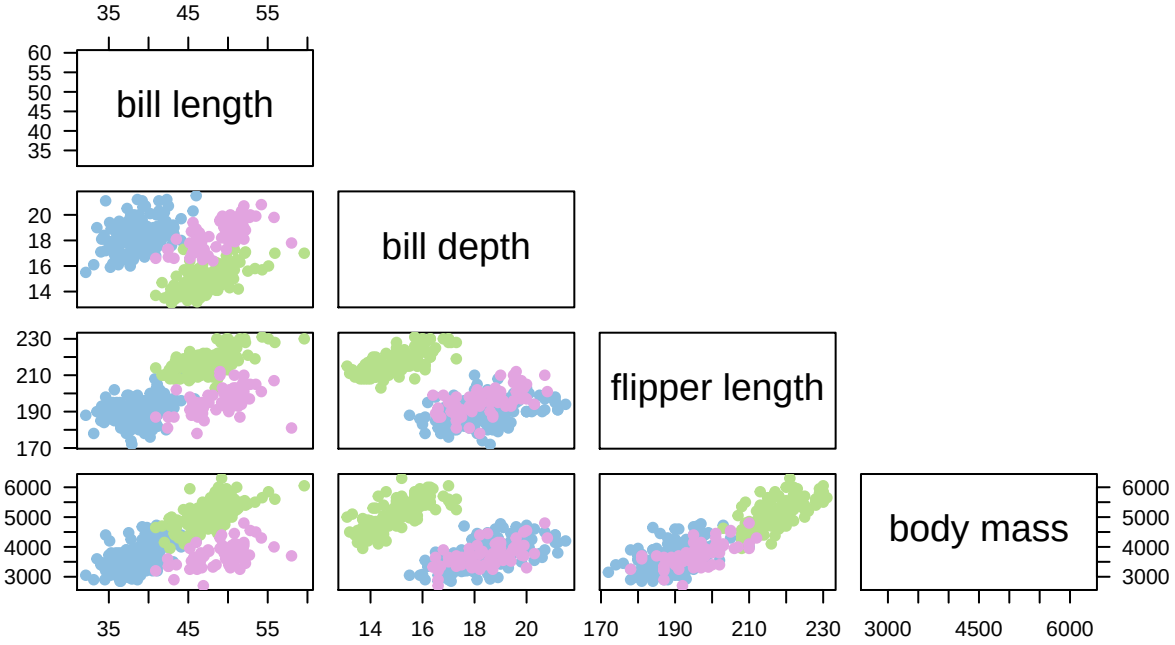


(Scrucca et al. 2016)

3 Case study: Penguins data

In order to better understand *k*-means and GMM, a case study was conducted on the `penguins` data set from the `palmerpenguins` package. The `palmerpenguins` package includes size measurements from three penguin species from Antarctica that were recorded from 2007 to 2009. Three clusters were chosen for this data since there are three species of penguins to identify (Horst, Hill, and Gorman 2020).

Here is the scatterplot matrix of the size measurements of the three penguin species:



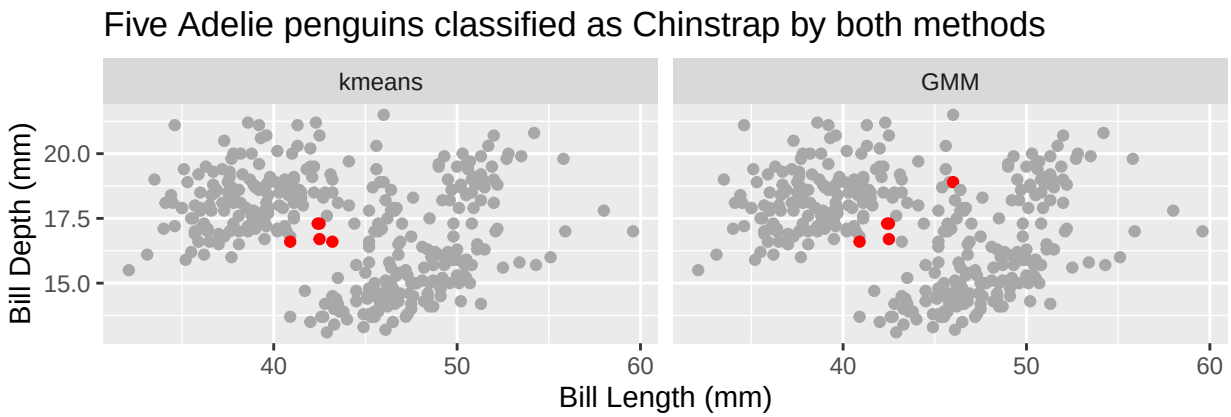
The Gentoo penguins (green) generally seem to be separate from the other two species. Adelie (blue) and Chinstrap (pink) penguins, however, overlap significantly when comparing body mass and flipper length, body mass and bill depth, and flipper length and bill depth.

Below are the resulting confusion matrices for both *k*-means and GMM. The rows represent the true values of the penguin species and the columns are the three clusters that *k*-means created.

	k-means			GMM		
	Cluster 1	Cluster 2	Cluster 3	Cluster 1	Cluster 2	Cluster 3
Adelie	0	127	24	0	151	0
Chinstrap	0	5	63	0	5	63
Gentoo	123	0	0	123	0	0

The *k*-means algorithm correctly classifyed all of the Gentoo penguins. Perhaps this is not unexpected considering the scatterplot matrix; Gentoo penguins were fairly easy to distinguish from the other two species. *k*-means did not classify Adelie or Chinstrap quite as well. Twenty-four Adelie penguins were classified as Chinstrap and five Chinstrap penguins were classified as Adelie. Similar to the *k*-means algorithm, none of the Gentoo penguins were classified incorrectly by GMM. Additionally, only five Chinstrap penguins were misclassified and no Adelie penguins were misclassified.

If we add up the number of penguins that were misclassified and divide by the total number of penguins, a misclassification metric can be obtained. 100*(24+5)/342 = 8.48% of the data were misclassified by *k*-means and 100*(5/342) = 1.46% by GMM. In this case, GMM did much better than *k*-means, misclassifying about 7 percentage points fewer penguins. Both methods misclassified five of the Chinstrap penguins as Adelie, so it might be informative to understand if the five points were the same for both *k*-means and GMM.



Though both *k*-means and GMM misclassify five Adelie penguins as Chinstrap, only four are the same penguins, as seen in the graph above. It is a little hard to distinguish, but the observation with a bill depth of just below 17.5 mm is actually two observations. So, both methods mistakenly classified four Adelie penguins as Chinstrap, and each method had one additional point that it also misclassified.

4 Simulation study

4.1 Goal of the Simulation

The goal of simulating data was to understand the conditions in which each method does well, poorly, or when both do equally well.

4.2 Simulation set up

The simulation generates 50 data sets with two bivariate normal variables. The true clusters are recorded at the time the data sets are created.

Both *k*-means (`kmeans`) and GMM (`Mclust`) are run on each of the 50 data sets and the parameters

are recorded (cluster means, confusion matrices, and for GMM the selected model and estimated variance-covariance matrix). The result is 50 sets of parameters for each method.

The function created to generate the data and run the analysis takes six parameters: μ_1 and μ_2, which are vectors containing the two mean parameters for each cluster, Σ_1 and Σ_2, which are the variance-covariance parameter vectors for each cluster, and n_1 and n_2, which are the sample sizes to generate for each respective cluster. Varying the parameters provided and examining the results provides insight into how well each method does at estimating the true parameters. The parameters chosen are listed in Tables 1 and 2.

Table 1: Scenario 1 - Model VII with $\Sigma_k = \lambda_k I$

Parameter	Cluster 1	Cluster 2
$\mu_k = \begin{bmatrix} \mu_{kx} \\ \mu_{ky} \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 5 \\ 5 \end{bmatrix}$
Σ_k	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}$
n_k	250	250

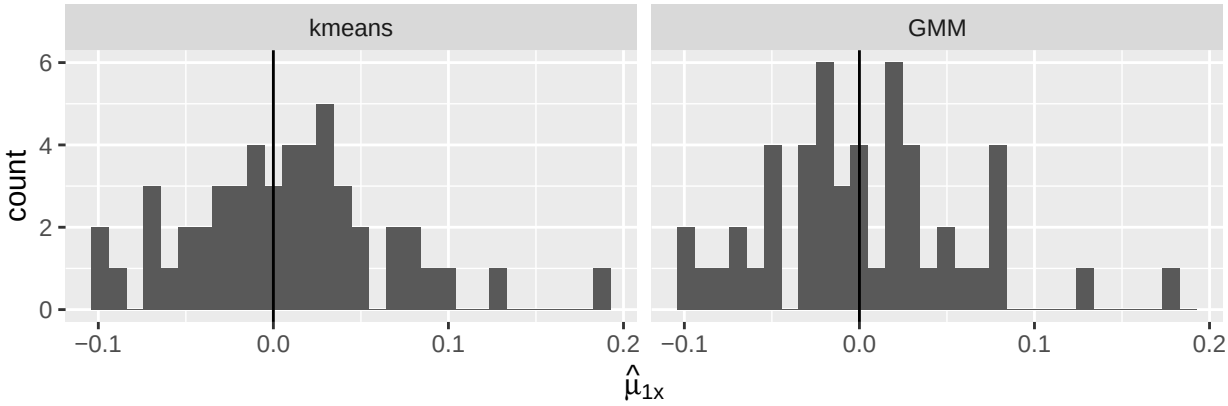
Table 2: Scenario 2 - Model VVI with $\Sigma_k = \lambda_k A_k$

Parameter	Cluster 1	Cluster 2
$\mu_k = \begin{bmatrix} \mu_{kx} \\ \mu_{ky} \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
Σ_k	$\begin{bmatrix} 2 & 0 \\ 0 & 0.1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$
n_k	250	250

4.3 Results

4.3.1 Scenario 1 - Model VII with $\Sigma_k = \lambda_k I$

In the scenario 1, both methods estimate the parameters well and nearly all of the data points are placed in the correct clusters. The histogram depicts the parameter estimates from each method for the μ_{kx} value, which in this case was 0, indicated by the vertical line.



These two histograms look reasonable; we would expect sample means to be distributed normally and both methods produce estimates of μ_{1x}, $\hat{\mu}_{1x}$, that seem to be distributed normally. *K*-means does not directly estimate μ_{1x}, but creates this estimate from the sample average of the x feature for observations in cluster 1.

More importantly, perhaps, is to look at the confusion matrices and the misclassification percentages in order to understand how well each method did at correctly classifying each data point.

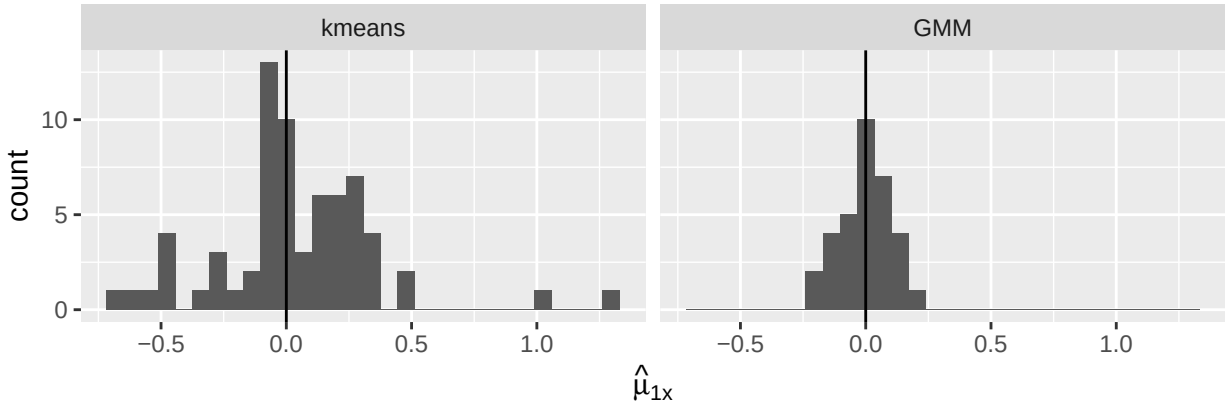
Below are the averages of all 50 confusion matrices for the *k*-means data and the GMM data. As we can see, both performed very well. The average misclassification percentage for *k*-means is 0.164%. The average misclassification percentage for GMM is 0.108%.

	k-means		GMM	
	Cluster 1	Cluster 2	Cluster 1	Cluster 2
Group A	249.98	0.02	249.82	0.18
Group B	0.8	249.2	0.36	249.64

Since `Mclust` chose the model it considered to be "best", it is interesting to look at which models each of the 50 simulations chose. The correct model for Scenario 1, VII, was selected 94% of the time. Other models it produced were EII, VEE, and VEI, which were all selected once.

4.3.2 Scenario 2 - Model VVI with $\Sigma_k = \lambda_k A_k$

In the next example, both methods find it much harder to correctly classify the data.



These two histograms look quite different from each other, *k*-means has a much larger spread. The $\hat{\mu}_{1x}$ values calculated by GMM are much more compact around zero, even in the presence of similar μ_{1x} and μ_{2x} values.

Below is the average of all 50 confusion matrices for the *k*-means and GMM data. As we can see, neither method does as well as in the first scenario, but *k*-means does particularly poorly The average misclassification percentage for *k*-means is 28%. The average misclassification percentage

for GMM is 13.56%.

	k-means		GMM	
	Cluster 1	Cluster 2	Cluster 1	Cluster 2
Group A	217.98	32.02	234.7	15.3
Group B	107.98	142.02	52.48	197.52

We can also look at which models were selected for Scenario 2, where the true model is VVI. The correct model was chosen 96% of the time. VVE was the only other model selected and it was chosen twice.

4.4 Simulation conclusion

The simulation found that in cases where the cluster means were relatively dissimilar $\left(\mu_{1x} = \begin{bmatrix} 0 & 0 \end{bmatrix}$ and $\mu_{2x} = \begin{bmatrix} 5 & 5 \end{bmatrix}\right)$ and the variances were small and equal, both methods did well at accurately grouping the data into clusters. In the case when the means were quite similar $\left(\mu_{1x} = \begin{bmatrix} 0 & 0 \end{bmatrix}$ and $\mu_{2x} = \begin{bmatrix} 1 & 1 \end{bmatrix}\right)$ and variances were similar but unequal within clusters, both methods fared worse, but Gaussian Mixture Models classified the data with more accuracy (13.56%) compared to *k*-means (28%).

5 Conclusion and Future Work

K-means clustering and Gaussian Mixture Models are both methods of finding clusters for unclassified data. *K*-means tends to excel when clusters are distinct from each other but does not take into account the variance of clusters. Though standardizing the variables helps with this, the method is limited by its use of distance to classify data. Gaussian Mixture Models is less widely used, but if

there is reason to believe the data follow a normal distribution, it tends to be more accurate than *k*-means clustering.

Some limitations of the simulation are that only two of the fourteen possible models for the variance-covariance matrix were explored in this investigation and that all of the data were simulated to meet the multivariate Gaussian assumption of GMM. Exploring the simulation of non-Gaussian random variables might yield situations in which *k*-means performs more accurately than GMM.

References

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Springer Science + Business Media, LLC.

Chan, Fong Chun. 2016. *Fitting a Mixture Model Using the Expectation-Maximization Algorithm in R*. https://tinyheero.github.io/2016/01/03/gmm-em.html.

Horst, Allison Marie, Alison Presmanes Hill, and Kristen B Gorman. 2020. *Palmerpenguins: Palmer Archipelago (Antarctica) Penguin Data*. https://allisonhorst.github.io/palmerpenguins/.

Lantz, Brett. 2019. *Machine Learning with R: Expert Techniques for Predictive Modeling Third Edition*. Packt Publishing.

R Core Team. 2019. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Scrucca, Luca, Michael Fop, T. Brendan Murphy, and Adrian E. Raftery. 2016. "mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models." *The R Journal* 8 (1): 289–317. https://doi.org/10.32614/RJ-2016-021.