

# The $k$ -means clustering technique: General considerations and implementation in *Mathematica*

Laurence Morissette and Sylvain Chartier

*Université d'Ottawa*

Data clustering techniques are valuable tools for researchers working with large databases of multivariate data. In this tutorial, we present a simple yet powerful one: the  $k$ -means clustering technique, through three different algorithms: the Forgy/Lloyd, algorithm, the MacQueen algorithm and the Hartigan & Wong algorithm. We then present an implementation in Mathematica and various examples of the different options available to illustrate the application of the technique.

Data clustering techniques are descriptive data analysis techniques that can be applied to multivariate data sets to uncover the structure present in the data. They are particularly useful when classical second order statistics (the sample mean and covariance) cannot be used. Namely, in exploratory data analysis, one of the assumptions that is made is that no prior knowledge about the dataset, and therefore the dataset's distribution, is available. In such a situation, data clustering can be a valuable tool. Data clustering is a form of unsupervised classification, as the clusters are formed by evaluating similarities and dissimilarities of intrinsic characteristics between different cases, and the grouping of cases is based on those emergent similarities and not on an external criterion. Also, these techniques can be useful for datasets of any dimensionality over three, as it is very difficult for humans to compare items of such complexity reliably without a support to aid the comparison.

The technique presented in this tutorial,  $k$ -means clustering, belongs to partitioning-based techniques grouping, which are based on the iterative relocation of data points between clusters. It is used to divide either the cases or the variables of a dataset into non-overlapping groups, or clusters, based on the characteristics uncovered. Whether the algorithm is applied to the cases or the variables of the dataset depends on which dimensions of this dataset we want to reduce the dimensionality of. The goal is to produce groups of cases/variables with a high degree of similarity within each group and a low degree of similarity between

groups (Hastie, Tibshirani & Friedman, 2001). The  $k$ -means clustering technique can also be described as a centroid model as one vector representing the mean is used to describe each cluster. MacQueen (1967), the creator of one of the  $k$ -means algorithms presented in this paper, considered the main use of  $k$ -means clustering to be more of a way for researchers to gain qualitative and quantitative insight into large multivariate data sets than a way to find a unique and definitive grouping for the data.

$K$ -means clustering is very useful in exploratory data analysis and data mining in any field of research, and as the growth in computer power has been followed by a growth in the occurrence of large data sets. Its ease of implementation, computational efficiency and low memory consumption has kept the  $k$ -means clustering very popular, even compared to other clustering techniques. Such other clustering techniques include connectivity models like hierarchical clustering methods (Hastie, Tibshirani & Friedman, 2000). These have the advantage of allowing for an unknown number of clusters to be searched for in the data, but are very costly computationally due to the fact that they are based on the dissimilarity matrix. Also included in cluster analysis methods are distribution models like expectation-maximisation algorithms and density models (Ankerst, Breunig, Kriegel & Sander, 1999).

A secondary goal of  $k$ -means clustering is the reduction of the complexity of the data. A good example would be letter grades (Faber, 1994). The numerical grades are clustered into the letters and represented by the average

included in each class.

Finally,  $k$ -means clustering can also be used as an initialization step for more computationally expensive algorithms like Learning Vector Quantization or Gaussian Mixtures, thus giving an approximate separation of the data as a starting point and reducing the noise present in the dataset (Shannon, 1948).

A good cluster analysis is both efficient and effective, in that it uses as few clusters as possible while still capturing all statistically important clusters. Similarity in cluster analysis is usually taken as meaning "proximity", and elements closer to one another in the input space are considered more similar. Different metrics can be used to calculate this similarity, the Euclidian distance  $dE = \sqrt{\sum_i^k (c_i - x_i)^2}$  being the most common. Here  $c$  is the cluster center,  $x$  is the case it is compared to,  $i$  is the dimension of  $x$  (or  $c$ ) being compared and  $k$  is the total number of dimensions. The Squared Euclidian distance  $dE^2 = \sum_i^k (c_i - x_i)^2$ , the Manhattan distance  $dMht = \sum_i^k |c_i - x_i|$  or the Maximum distance between attributes of the vectors  $dMax = \max_{i=1\dots k} \sum_i |c_i - x_i|$  can also be used. Of greater interest are the Mahalanobis distance  $dM = \sqrt{((c - x)^T Cov^{-1}(c - x))}$ , which accounts for the covariance between the vectors, and the Cosine similarity  $dCos = \frac{cx}{||c|| \cdot ||x||}$ , which is a non-translation invariant version of the correlation. By translation invariant, we mean here that adding a constant to all elements of the vectors will not change the result of the correlation while it will change the result of the cosine similarity.<sup>1</sup> Both the correlation and cosine similarity are invariant to scaling (multiplying all elements by a nonzero constant). As the  $k$ -means algorithms try to minimize the sum of the variances within the clusters,  $E = \sum_{i=1}^k \sum_{j=1}^{n_i} \|x_{ij} - c_i\|^2$ , where  $n_i$  is the number of cases included in cluster  $k$  and  $\sum_i^k n_i = n$ , the  $k$ -means clustering technique is considered a variance minimization technique.

Mathematically, the  $k$ -means technique is an approximation of a normal mixture model with an estimation of the mixtures by maximum likelihood. Mixture models consider cluster membership as a probability for each case, based on the means, covariances, and sampling probabilities of each cluster (Symons, 1981). They represent the data as a mixture of distributions (Gaussian, Poisson, etc.), where each distribution represents a sub-population (or cluster) of the data. The  $k$ -means technique is a sub case that assumes that the mixture components (here, clusters) all have spherical covariance matrices and equal sampling probabilities. I also consider the cluster membership for each

case a separate parameter to be estimated.

### K-means clustering

We present three  $k$ -means clustering algorithms: the Forgy/Lloyd algorithm, the MacQueen algorithm and the Hartigan & Wong algorithm. We chose those three algorithms because they are the most widely used  $k$ -means clustering techniques and they all have slightly different goals and thus results. To be able to use any of the three, you first need to know how many clusters are present in your data. As this information is often unavailable, multiple trials will be necessary to find the best amount of clusters. As a starting point, it is often useful to standardize the data if the components of the cases are not in the same scale.

There is no absolute best algorithm. The choice of the optimal algorithm depends on the characteristics of the dataset (size, number of variables in the cases). Jain, Duin & Mao (2000) even suggest trying several different clustering algorithms to gain the best understanding possible about the dataset.

#### Forgy/Lloyd algorithm

The Lloyd algorithm (1957, published 1982) and the Forgy's algorithm (1965) are both batch (also called offline) centroid models. A centroid is the geometric center of a convex object and can be thought of as a generalisation of the mean. Batch algorithms are algorithms where a transformative step is applied to all cases at once. They are well suited to analyse large data sets, since the incremental  $k$ -means algorithms require to store the cluster membership of each case or to do two nearest-cluster computations as each case is processed, which is computationally expensive on large datasets. The difference between the Lloyd algorithm and the Forgy algorithm is that the Lloyd algorithm considers the data distribution discrete while the Forgy algorithm considers the distribution continuous. They have exactly the same procedure apart from that consideration. For a set of cases  $[x_1, x_2, \dots, x_n] \in \mathcal{R}^d$ , where  $\mathcal{R}^d$  is the data space of  $d$  dimensions, the algorithm tries to find a set of  $k$  cluster centers  $C = [c_1, c_2, \dots, c_k] \in \mathcal{R}^d$  that is a solution to the minimization problem:

$$E = \sum_{i=1}^k \sum_{j=1}^n d(c_i, x_{ij}), \text{ discrete distribution}$$

$$E = \sum_{i=1}^k \int \rho(x) d(c_i, x_{ij}) dx, \text{ continuous distribution}$$

where  $\rho(x)$  is the probability density function and  $d$  is the distance function. Note here that if the probability density function is not known, it has to be deduced from the data available.

The first step of the algorithm is to choose the  $k$  initial centroids. It can be done by assigning them based on previous empirical knowledge, if it is available, by using  $k$

<sup>1</sup> The correlation is the cosine similarity of mean-centered, deviation-normalized data

random observations from the data set, by using the  $k$  observations that are the farthest from one another in the data space or just by giving them random values within  $R^d$ . Once the initial centroids have been chosen, iterations are done on the following two steps. In the first one, each case of the data set is assigned to a cluster based on its distance from the clusters centroids, using one of the metric previously presented. All cases assigned to a centroid are said to be part of the centroid's subspace  $C(R^d)$ . The second step is to update the value of the centroid using the mean of the cases assigned to the centroid. Those iterations are repeated until the centroids stop changing, within a tolerance criterion decided by the researcher, or until no case changes cluster.

Here is the pseudocode describing the iterations:

- 1- Choose the number of clusters
- 2- Choose the metric to use
- 3- Choose the method to pick initial centroids
- 4- Assign initial centroids
- 5- While metric(centroids, cases)>threshold
  - a. For  $i \leq \text{nb cases}$ 
    - i. Assign case to closest cluster according to metric
  - b. Recalculate centroids

The *k-means* clustering technique can be seen as partitioning the space into Voronoi cells (Voronoi, 1907). For each two centroids, there is a line that connects them. Perpendicular to this line, there is a line, plane or hyperplane (depending on the dimensionality) that passes through the middle point of the connecting line and divides the space into two separate subspaces. The *k-means* clustering therefore partitions the space into  $k$  subspaces for which  $c_i$  is the nearest centroid for all included elements of the subspace (Faber, 1994).

#### **MacQueen algorithm**

The MacQueen algorithm (1967) is an iterative (also called online or incremental) algorithm. The main difference with Forgy/Lloyd's algorithm is that the centroids are recalculated every time a case change subspace and also after each pass through all cases. The centroids are initialized the same way as in the Forgy/Lloyd algorithm and the iterations are as follow. For each case in turn, if the centroid of the subspace it currently belongs to is the nearest, no change is made. If another centroid is the closest, the case is reassigned to the other centroid and the centroids for both the old and new subspaces are recalculated as the mean of the belonging cases. The algorithm is more efficient as it updates centroids more often and usually needs to perform one complete pass through the cases to converge on

a solution.

Here is the pseudocode describing the iterations:

- 1- Choose the number of clusters
- 2- Choose the metric to use
- 3- Choose the method to pick initial centroids
- 4- Assign initial centroids
- 5- While metric(centroids, cases)>threshold
  - a. For  $i \leq \text{cases}$ 
    - i. Assign case  $i$  to closest cluster according to metric
    - ii. Recalculate centroids for the two affected clusters
  - b. Recalculate centroids

#### **Hartigan & Wong algorithm**

This algorithm searches for the partition of data space with locally optimal within-cluster sum of squares of errors (SSE). It means that it may assign a case to another subspace, even if it currently belong to the subspace of the closest centroid, if doing so minimizes the total within-cluster sum of square (see below). The cluster centers are initialized the same way as in the Forgy/Lloyd algorithm. The cases are then assigned to the centroid nearest them and the centroids are calculated as the mean of the assigned data points. The iterations are as follows. If the centroid has been updated in the last step, for each data point included, the within-cluster sum of squares for each data point if included in another cluster is calculated. If one of the cluster sum of square (SSE2 in the equation below, for all  $i \neq 1$ ) is smaller than the current one (SSE1), the case is assigned to this new cluster.

$$SSE2 = \frac{N_i \sum_j \|x_{ij} - c_i\|^2}{N_i - 1} < SSE1 = \frac{N_1 \sum_j \|x_{1j} - c_1\|^2}{N_1 - 1}$$

The iterations continue until no case change cluster, meaning until a change would make the clusters more internally variable or more externally similar.

Here is the pseudocode describing the iterations:

- 1- Choose the number of clusters
- 2- Choose the metric to use
- 3- Choose the method to pick initial centroids
- 4- Assign initial centroids
- 5- Assign cases to closest centroid
- 6- Calculate centroids
- 7- For  $j \leq \text{nb clusters}$ , if centroid  $j$  was updated last iteration
  - a. Calculate SSE within cluster
  - b. For  $i \leq \text{nb cases in cluster}$ 
    - i. Compute SSE for cluster  $k \neq j$  if case included
    - ii. If SSE cluster  $k < \text{SSE cluster } j$ , case change cluster

### Quality of the solutions found

There are two ways to evaluate a solution found by  $k$ -means clustering. The first one is an internal criterion and is based solely on the dataset it was applied to, and the second one is an external criterion based on a comparison between the solution found and an available known class partition for the dataset.

The Dunn index (Dunn, 1979) is an internal evaluation technique that can roughly be equated to the ratio of the inter-cluster similarity on the intra-cluster similarity:

$$DI = \min_{i=1 \dots m} \left\{ \min_{j=1 \dots m, i \neq j} \left\{ \frac{d(c_i, c_j)}{\max_{k=1 \dots m} \Delta k} \right\} \right\}$$

where  $d(c_i, c_j)$  is the distance between cluster centroids and can be calculated with any of the previously presented metrics and  $\Delta k$  is the measure of inner cluster variation. As we are looking for compact clusters, the solution with the highest Dunn index is considered the best.

As an external evaluator, the Jaccard index (Jaccard, 1901) is often used when a previous reliable classification of the data is available. It computes the similarity between the found solution and the benchmark as a percentage of correct classification. It calculates the size of the intersection (the cases present in the same clusters in both solutions) divided by the size of the union (all the cases from both datasets):

$$J = \frac{|A \cap B|}{|A \cup B|}$$

### Limitations of the technique

The  $k$ -means clustering technique will always converge, but it is liable to find a local minimum solution instead of a global one, and as such may not find the optimal partition. The  $k$ -means algorithms are local search heuristics, and are therefore sensitive to the initial centroids chosen (Ayramo & Karkkainen, 2006). To counteract this limitation, it is recommended to do multiple applications of the technique, with different starting points, to obtain a more stable solution through the averaging of the solutions obtained.

Also, to be able to use the technique, the number of clusters present in your data must be decided at the onset, even if such information is not available *a priori*. Therefore, multiple trials are necessary to find the best amount of clusters. Thirdly, it is possible to create empty clusters with the Forgy/Lloyd algorithm if all cases are moved at once from a centroid subspace. Fourthly, the MacQueen and Hartigan methods are sensitive to the order in which the points are relocated, yielding different solutions depending on the order.

Fifthly,  $k$ -means clustering has a bias to create clusters of equal size, even if doing so doesn't best represent the group

distributions in the data. It thus works best for clusters that are globular in shape, have equivalent size and have equivalent data densities (Ayramo & Karkkainen, 2006). Even if the dataset contains clusters that are not equiprobable, the  $k$ -means technique will tend to produce clusters that are more equiprobable than the population clusters. Corrections for this bias can be done by maximizing the likelihood without the assumption of equal sampling probabilities (Symons, 1981).

Finally, the technique has problems with outliers, as it is based on the mean, a descriptive statistic not robust to outliers. The outliers will tend to skew the centroid position toward them and have a disproportionate importance within the cluster. A solution to this was proposed by Ayramo & Karkkainen (2006). They suggested using the spatial median instead to get a more robust clustering.

### Alternate algorithms

#### Optimisation of the algorithms usage

While the algorithms presented are very efficient, since the technique is often used as a first classifier on large datasets, any optimisation that speeds the convergence of the clustering is useful. Bottou and Bengio (1995) have found that the fastest convergence on a solution is usually obtained by using an online algorithm for the first iteration through the entire dataset and an off-line algorithm subsequently as needed. This comes from the fact that online  $k$ -means benefits from the redundancies of the  $k$  training set and improve the centroids by going through a few cases (depending on the amount of redundancies) as much as would a full iteration through the offline algorithm (Bengio, 1991).

#### For very large datasets

For very large datasets that would make the computation of the previous algorithms too computationally expensive, it is possible to choose a random sample from the whole population of cases and apply the algorithm on the sample. If the sample is sufficiently large, the distribution of these initial reference points should reflect the distribution of cases in the entire set.

#### Fuzzy $k$ -means clustering

In fuzzy  $k$ -means clustering (Bezdek, 1981), each case has a set of degree of belonging relative to all clusters. It differs from previously presented  $k$ -means clustering where each case belongs only to one cluster at a time. In this algorithm, the centroid of a cluster ( $c_k$ ) is the mean of all cases in the dataset, weighted by their degree of belonging to the cluster ( $w_k$ ).

$$c_k = \frac{\sum_i w_k(x_i) \times x_i}{\sum_i w_k(x_i)}$$

The degree of belonging is a function of the distance of the case from the centroid, which includes a parameter controlling for the highest weight given to the closest case. It iterates until a user-set criterion is reached. Like the  $k$ -means clustering technique, this technique is also sensitive to initial clusters and local minima. It is particularly useful for dataset coming from area of research where partial belonging to classes is supported by theory.

### **Self-Organising Maps**

Self-Organizing Maps (Kohonen, 1982) are an artificial neural network algorithm that aims to extract attributes present in a dataset and transcribe them into an output space of lower dimensionality, while keeping the spatial structure of the data. Doing so clusters similar cases on the map, a process that can be likened to the  $k$ -means algorithm clustering centroids. This neural network has two layers, the input layer which is the initial dataset, and an output layer that is the self-organizing map, which is usually bi-dimensional. There is a connection weight between each variable (or attribute) of a case and the map, thus making the connection weights matrix of the dimensionality of the input multiplied by the dimensionality of the map. It uses a Hebbian competitive learning algorithm. What is obtained at the end is a map where similar elements are contiguous, which also give a two dimensional representation of the data. It is therefore useful if a graphic representation of the data is advantageous to its comprehension.

### **Gaussian-expectation maximization (GEM)**

This algorithm was first explained by Dempster, Laird & Rubin (1977). It uses a linear combination of  $d$ -dimensional Gaussian distributions as the cluster centers. It aims to minimize

$$GEM = - \sum_{i=1}^n \log \left( \sum_{j=1}^k p(x_i|c_j)p(c_j) \right)$$

where  $p(x_i|c_j)$  is the probability of  $x_i$  (the case), given that it is generated by a Gaussian distribution that has  $c_j$  as its center, and  $p(c_j)$  is the prior probability of said center. It also computes a soft membership for each center through Bayes rule:

$$m(c_j x_i) = \frac{p(x_i|c_j)p(c_j)}{p(x_i)}$$

### **The Mathematica Notebook**

There exists a function in Mathematica, "FindClusters",

that implements the  $k$ -means clustering technique with an alternative algorithm called  $k$ -medoids. This algorithm is equivalent to the Forgy/Lloyd algorithm but it uses cases from the datasets as centroids instead of the arithmetical mean. The implementation of the algorithm in Mathematica allows for the use of different metrics. There is also a function in Matlab called "kmeans" that implements the  $k$ -means clustering technique. It uses a batch algorithm in a first phase, then an iterative algorithm in a second phase. Finally, there is no implementation of the  $k$ -means technique in SPSS, but an implementation of hierarchical clustering is available. As the goals of this tutorial are to showcase the workings of the  $k$ -means clustering technique and to help understand said technique better, we created a Mathematica Notebook where the inner workings of all three algorithms are open to view (available on the TQMP website).

The Notebook has clearly labeled sections. The initial section contains all of the modules used in the Notebook. This is where you can see the inner workings of the algorithms. In the section of the Notebook where user changes are allowed, you find various subsections that explicit the parameters the user needs to input. The first one is used to import the data, which should be in a database format (.txt, .dat, etc.), and should not include the variable names. The second section allows to standardize the dataset variables if need be. The third section put a label on each case to keep track of cases as they are clustered. The next sections allows to choose the number of clusters, the stop criterion on the number of iterations, the tolerance level between the cluster solutions, the metric to be used (between Euclidian distance, Squared Euclidian distance, Manhattan distance, Maximum distance, Mahalanobis distance and Cosine similarity) and the starting centroids. To choose the centroids, random assignation or farthest vectors assignation are available. The following section is the heart of the Notebook. Here you can choose to use the Forgy/Lloyd, MacQueen or Hartigan & Wang algorithm. The algorithms iterate until the user-inputted criterion on the number of iterations or centroid change is reached. For each algorithm, you obtain the number of iterations through the whole dataset needed for the solution to converge, the centroids vectors and the cases belonging to each cluster. The next section implements the Dunn index, which evaluates the internal quality of the solution and outputs the Dunn index. Next is a visualisation of the cases and their centroids for bidimensionnal or tridimensionnal datasets. The next section calculates the equation of the vector/plan that separates two centroids subspaces. Finally, the last section uses Mathematica's implementation of the ANOVA to allow the user to compare clusters to see for which variables the clusters are significantly different from one another.

	Performance	Information	Verbal Expression	Age		Performance	Information	Verbal Expression	Age
1	87	5	31	6.4	1	-0.628365	-1.42134	-0.528701	-1.00737
2	97	7	36	8.3	2	0.0785456	-0.781736	0.264351	1.14588
3	112	9	42	7.2	3	1.13891	-0.142134	1.21601	-0.100737
4	102	16	45	7.	4	0.432001	2.09647	1.69184	-0.327394
5	85	10	38	7.6	5	-0.769747	0.177667	0.581571	0.352578
6	76	9	32	6.2	6	-1.40597	-0.142134	-0.370091	-1.23402
7	120	12	30	8.4	7	1.70444	0.817269	-0.687311	1.25921
8	85	8	28	6.3	8	-0.769747	-0.461935	-1.00453	-1.12069
9	99	9	27	8.2	9	0.219928	-0.142134	-1.16314	1.03255

Table 1. Data used left) non standardised right) standardised

**Example**

1a. Let's take a toy example and use our Mathematica notebook to find a clustering solution. The first thing that we need to do is activate the initialisation cells that contain the modules. We'll use a dataset (at the beginning of the Mathematica Notebook) that has four dimensions and nine cases. Please activate only the dataset needed. As the variables are not on the same scale, we start by standardizing the data, as seen in Table 1.

Now, as we have no prior information on the dataset, we chose to pick three clusters and to choose the cases that are the furthest from one another, as seen in Table 2.

We chose the Lloyd method to find the clusters with a Euclidian distance metric. We then run the main program, which iterates in turn to assign cases to centroids and move the centroids. After 2 iterations, the centroids have attained their final position and the cases have changed clusters to their final position. The solution found has one cluster containing cases 1, 6 and 8, one cluster containing case 4 and one cluster containing the remaining cases (2,3,5 and 7). This is illustrated in Figure 1.

1b. It is often interesting to see which variables contributed to the clustering the most. While not strictly part of the *k*-means clustering technique, it is a useful step when the variables have meaning assigned to them. To do so, we use the ANOVA with post-hoc test found at the end of the Notebook. We find that cluster 1 contain cases with a higher age than cluster 3 ( $F(2,6) = 14.11, p < .01$ ) and that cluster 2 is higher than both the other two clusters on the information variable ( $F(2,6) = 5.77, p < .05$ ). No clusters differ significantly on the verbal expression and performance variables.

1c. It is also possible to obtain the equation of the boundary between the clusters' neighborhood. To do so, use

	Performance	Information	Verbal Expression	Age
	-0.628365	-1.42134	-0.528701	-1.00737
	0.0785456	-0.781736	0.264351	1.14588
	1.13891	-0.142134	1.21601	-0.100737
	0.432001	2.09647	1.69184	-0.327394
	-0.769747	0.177667	0.581571	0.352578
	-1.40597	-0.142134	-0.370091	-1.23402
	1.70444	0.817269	-0.687311	1.25921
	-0.769747	-0.461935	-1.00453	-1.12069
	0.219928	-0.142134	-1.16314	1.03255

Table 2. Farthest cases

the "Finding the equation" section of the Notebook. The output obtained is shown in Table 3.

If we read this output, we find that the equation of the vector that separates cluster 1 from cluster 2 is  $-2.28 -1.37p - 2.77i - 2.33ve - .79a$ , where *p* is performance, *i* is information, *ve* is verbal expression and *a* is age. So any new cases within each of the boundaries could be classified as belonging to the corresponding centroid.

2a. Let's briefly present a visual example of clusters centers moving. To do so, we used a dataset with 2 attributes (named xp1-xp3.dat in the Notebook). The dataset is very asymmetrical and as such would not be an ideal case to apply the algorithm on (since the distribution of the clusters is unlikely to be spherical), but it will show the moving of the clusters effectively. We chose a four clusters solution and used the Forgy/Lloyd algorithm with Euclidian distance again.

We can see in Figure 2 that the densest area of the dataset has two clusters, as the algorithm tries to form equiprobable clusters. The data that is really high on the first variable belong to a cluster while the data that is really high on the second variable belong to a fourth cluster.

2b. The next simulation compares the different metrics for the same algorithm. We used the Forgy/Lloyd algorithm with random starting clusters.

	Performance	Information	Verbal Expression	Age
1	-0.628365	-1.42134	-0.528701	-1.00737
2	0.0785456	-0.781736	0.264351	1.14588
3	1.13891	-0.142134	1.21601	-0.100737
4	0.432001	2.09647	1.69184	-0.327394
5	-0.769747	0.177667	0.581571	0.352578
6	-1.40597	-0.142134	-0.370091	-1.23402
7	1.70444	0.817269	-0.687311	1.25921
8	-0.769747	-0.461935	-1.00453	-1.12069
9	0.219928	-0.142134	-1.16314	1.03255

Iterations: 2

Cluster centers:  $\begin{pmatrix} -0.934693 & -0.675136 & -0.634441 & -1.12069 \\ 0.432001 & 2.09647 & 1.69184 & -0.327394 \\ 0.474415 & -0.0142134 & 0.0422961 & 0.737895 \end{pmatrix}$

Cluster belonging:  $\{\{6, 1, 8\}, \{4\}, \{9, 5, 2, 7, 3\}\}$

Figure 1. Upper section) Cases assignation to clusters Lower section) Output from the Notebook giving the number of iterations, the final centroids and the cases belonging to each clusters

	Cluster 1	Cluster 2	Intercept	Performance	Information	Verbal Expression	Age
1	2		-2.28171	-1.36669	-2.77161	-2.32628	-0.793301
1	3		1.10819	-1.40911	-0.660922	-0.676737	-1.85859
2	3		3.3899	-0.0424146	2.11069	1.64955	-1.06529

Table 3. Plans defining the limits of each centroid subspace

We can see in Figure 3 that depending on what we want to prioritize in the data, the different metrics can be used to reach different goals. The Euclidian distance prioritizes the minimisation of all differences within the cluster while the cosine similarity prioritizes the maximization of all the similarities. Finally, the maximum distance prioritizes the minimisation of the distance of the most extreme elements of each case within the cluster.

3a. This example demonstrates the influence of using different algorithms. We used the very well-known iris flower dataset (Fisher, 1936). The dataset (named iris.dat in the Notebook) consists of 150 cases and there are three classes present in the data. Cases are split into equiprobable groups (group 1 is from cases 1 to 50, group 2 is from 51 to 100 and group 3 is from 101 to 150). Each iris is described by three attributes (sepal length, sepal width and petal length).

We chose random starting vectors from the dataset for the simulations centroids and used the same for all three algorithms:

$$\begin{pmatrix} 1.27607 & 0.10609 & 0.760212 \\ -0.897674 & -1.27769 & -0.429981 \\ -0.535384 & 1.95113 & -1.16677 \end{pmatrix}$$

We also used the Euclidian metric to calculate the distance between cases and centroids. Table 4 summarizes the results.

We can see that all algorithms make mistakes in the classification of the irises (as was expected from the characteristics of the data). For each, the greyed out cases are misclassified. We find that the Forgy/Lloyd algorithm is the one making the fewer mistakes, as indicated here by the Dunn and Jaccard indexes, but the graphs of Figure 4 shows that the best visual fit comes from the MacQueen algorithm.

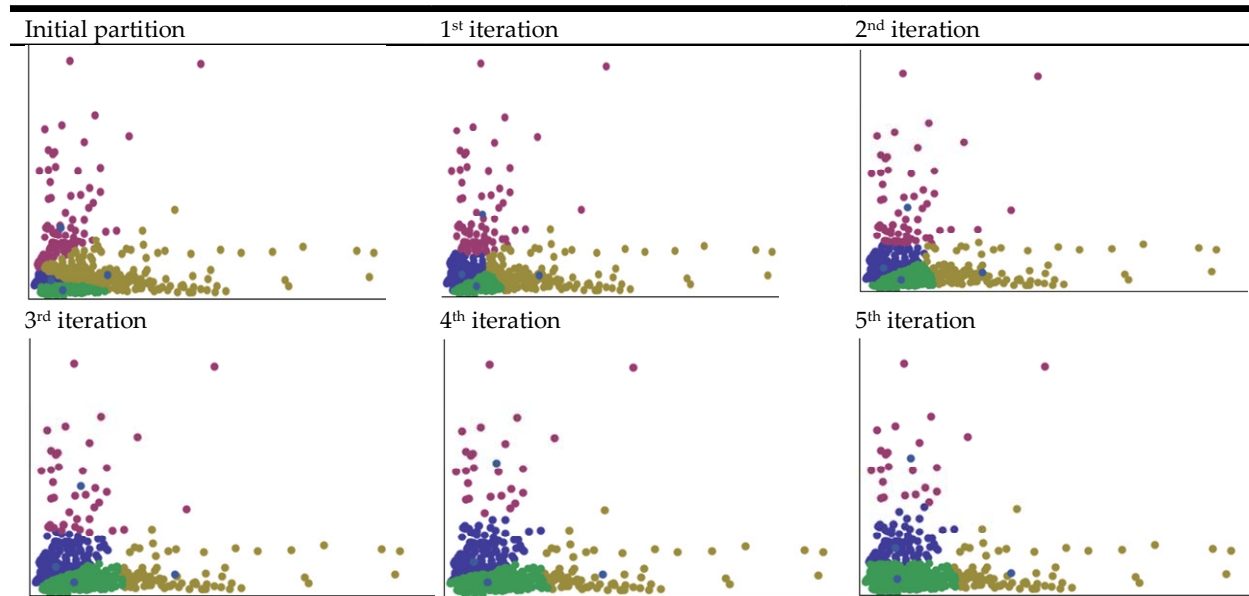


Figure 2. Cases assignation changes during iterations

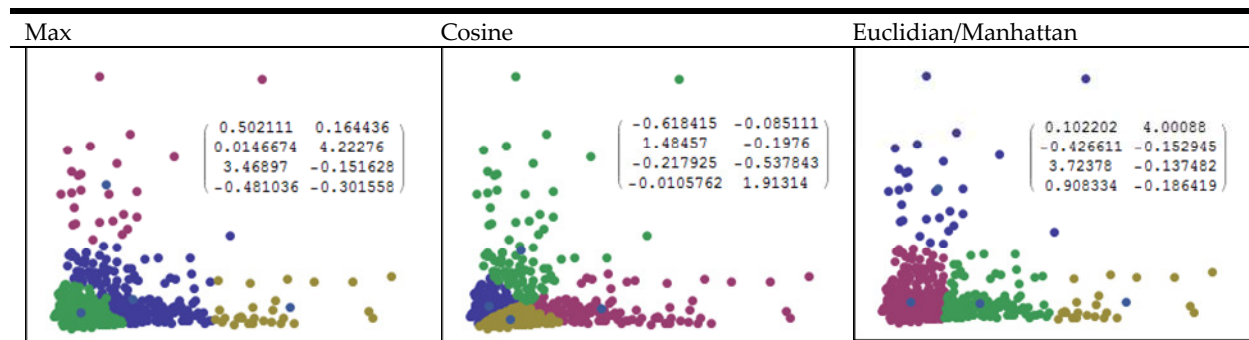


Figure 3. Effect of metric in the Forgy/Lloyd algorithm

Table 4. Summary of the results

Technique (iterations)	Cluster	Cases included	Dunn index	Jaccard index
Forgy/Lloyd (2)	$\begin{pmatrix} 1.08563 & 0.0484324 & 0.945498 \\ -0.153378 & -0.943513 & 0.295239 \\ -0.998721 & 0.892116 & -1.29862 \end{pmatrix}$	51, 52, 53, 55, 57, 59, 66, 71, 75, 76, 77, 78, 86, 87, 92, 101, 103, 104, 105, 106, 108, 109, 110, 111, 112, 113, 116, 117, 118, 119, 121, 123, 125, 126, 128, 129, 130, 131, 132, 133, 134, 136, 137, 138, 140, 141, 142, 144, 145, 146, 148, 149 42, 54, 56, 58, 60, 61, 62, 63, 64, 65, 67, 68, 69, 70, 72, 73, 74, 79, 80, 81, 82, 83, 84, 85, 88, 89, 90, 91, 93, 94, 95, 96, 97, 98, 99, 100, 102, 107, 114, 115, 120, 122, 124, 127, 135, 139, 143, 147, 150 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 43, 44, 45, 46, 47, 48, 49, 50	.042	.813
MacQueen (15)	$\begin{pmatrix} -0.0254945 & -0.980435 & 0.357184 \\ 0.940124 & 0.0389975 & 0.890051 \\ -1.01119 & 0.839494 & -1.30052 \end{pmatrix}$	54, 56, 58, 60, 61, 63, 65, 68, 69, 70, 73, 74, 76, 77, 79, 80, 81, 82, 83, 84, 86, 88, 90, 91, 93, 94, 95, 96, 97, 98, 99, 100, 102, 107, 109, 112, 114, 115, 120, 127, 128, 134, 135, 143, 147 51, 52, 53, 55, 57, 59, 62, 64, 66, 67, 71, 72, 75, 78, 85, 87, 89, 92, 101, 103, 104, 105, 106, 108, 110, 111, 113, 116, 117, 118, 119, 121, 122, 123, 124, 125, 126, 129, 130, 131, 132, 133, 136, 137, 138, 139, 140, 141, 142, 144, 145, 146, 148, 149, 150 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50	.033	.787
Hartigan (6)	$\begin{pmatrix} 0.908368 & -0.0970024 & 0.863413 \\ -0.611517 & -0.821445 & -0.218062 \\ -0.884618 & 1.19691 & -1.29237 \end{pmatrix}$	51, 52, 53, 55, 57, 59, 62, 64, 66, 71, 72, 73, 74, 75, 76, 77, 78, 79, 84, 86, 87, 92, 98, 101, 103, 104, 105, 106, 108, 109, 110, 111, 112, 113, 115, 116, 117, 118, 119, 121, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 144, 145, 146, 147, 148, 149, 150 2, 4, 9, 10, 13, 14, 26, 31, 35, 38, 39, 42, 46, 54, 56, 58, 60, 61, 63, 65, 67, 68, 69, 70, 80, 81, 82, 83, 85, 88, 89, 90, 91, 93, 94, 95, 96, 97, 99, 100, 102, 107, 114, 120, 122, 143 1, 3, 5, 6, 7, 8, 11, 12, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 32, 33, 34, 36, 37, 40, 41, 43, 44, 45, 47, 48, 49, 50	.036	.72

### Mathematica K-medoid algorithm

Mathematica's implementation gives the clustering of the cases, but not the cluster centers or the tag of the cases. It makes it quite confusing to actually keep track of individual cases. Here is the graphical solution obtained, which is very

similar to the MacQueen solution (Figure 5).

### Conclusion

We have shown that  $k$ -means clustering is a very simple and elegant way to partition datasets. Researchers from all fields would gain to know how to use the technique, even if



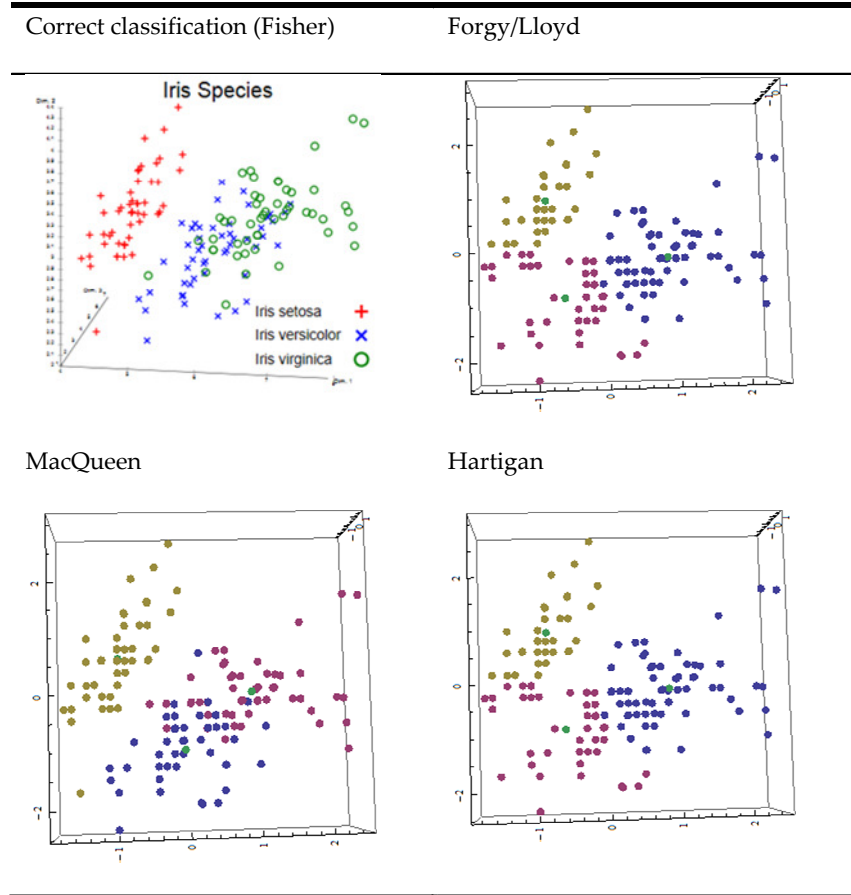


Figure 4. Final classification of different algorithms

it's only for preliminary analyses. The three algorithms combined with the six different metrics available should allow tailoring the analysis based on the characteristics of the dataset and depending on the goals to be achieved by the clustering (maximizing similarities or reducing differences). We present in Table 5 a summary of each algorithm.

As it's not implemented in regular statistics package software like SPSS, we presented a Mathematica notebook that should allow any researcher to use the technique on their data easily.

(References follow)

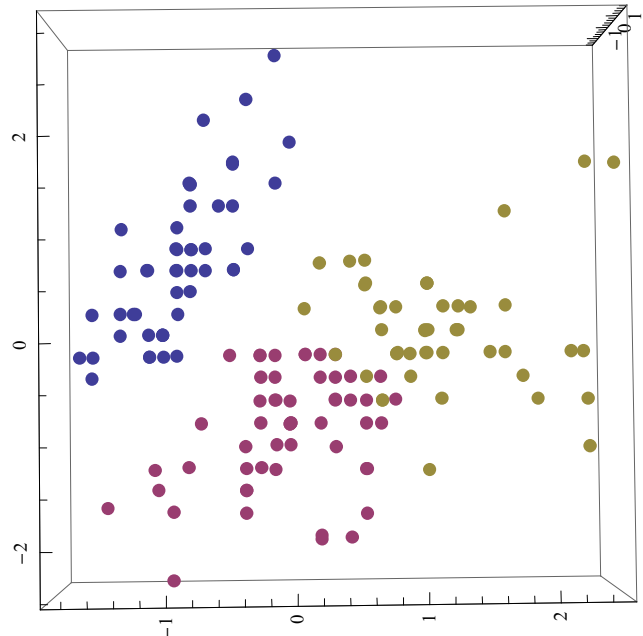


Figure 5. K-medoid clustering solution

Table 5: Summary of the algorithms for  $k$ -means clustering

Algorithm	Advantages	Disadvantages
Lloyd	- For large data sets - Discrete data distribution - Optimize total sum of squares	- Slower convergence - Possible to create empty clusters
Forgy's	- For large data sets - Continuous data distribution - Optimize total sum of squares	- Slower convergence - Possible to create empty clusters
McQueen	- Fast initial convergence - Optimize total sum of squares	- Need to store the two nearest-cluster computations for each case - Sensitive to the order the algorithm is applied to the cases
Hartigan	- Fast initial convergence - Optimize within-cluster sum of squares	- Need to store the two nearest-cluster computations for each case - Sensitive to the order the algorithm is applied to the cases

### References

- Ankerst, M., Breunig, M.M., Kriegel, H.-P. & Sander, G. (1999). OPTICS: Ordering Points To Identify the Clustering Structure. ACM Press. pp. 49–60.
- Ayramo, S. & Karkkainen, T. (2006) Introduction to partitioning-based cluster analysis methods with a robust example, *Reports of the Department of Mathematical Information Technology; Series C: Software and Computational Engineering*, C1, 1-36.
- Bezdek, J.C. (1981) Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York.
- Bengio, Y. (1991) Artificial Neural Networks and their Application to Sequence Recognition, PhD thesis, McGill University, Canada
- Bottou, L. & Bengio, Y. (1995) Convergence properties of the K-means algorithms, in *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky & T. Leen, eds., 7, The MIT Press, 585–592.
- Dempster, A.P., Laird, N.M. & Rubin, D.B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39 (1), 1–38.
- Faber, V. (1994). Clustering and the Continuous k-means Algorithm, *Los Alamos Science*, 22, 138-144.
- Fisher, R.A. (1936) The use of multiple measurements in taxonomic problems. *Annals Eugenics*, 7(II), 179-188.
- Forgy, E. (1965) Cluster analysis of multivariate data: efficiency vs. interpretability of classification, *Biometrics*, 21, 768.
- Hastie, T., Tibshirani, R. & Friedman, J. (2000) The elements of statistical learning: Data mining, inference and prediction, Springer-Verlag, 2001.
- Hartigan, J.A. & Wong, M.A. (1979) Algorithm AS 136 A K-Means Clustering Algorithm, *Applied Statistics*, 28(1), 100-108.
- Jaccard, P. (1901) Étude comparative de la distribution florale dans une portion des Alpes et des Jura, *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37, 547–579.
- Jain, A.K., Duin, R.P.W. & Mao, J. (2000) Statistical pattern recognition: A review, *IEEE Trans. Pattern Anal. Machine Intelligence*, 22, 4–37.
- Kohonen, T. (1982). Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, 43(1), 59–69.
- Lloyd, S.P. (1982) Least squares quantization in PCM, *IEEE Transactions on Information Theory*, 28, 129–136.
- MacQueen, J. (1967) Some methods for classification and analysis of multivariate observations, *Proceedings of the Fifth Berkeley Symposium On Mathematical Statistics and Probabilities*, 1, 281-296.
- Shannon, C.E. (1948), A Mathematical Theory of Communication, *Bell System Technical Journal*, 27, pp. 379–423 & 623–656.
- Symons, M.J. (1981) Clustering Criteria and Multivariate Normal Mixtures, *Biometrics*, 37, 35-43.
- Voronoi, G. (1907) Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *Journal für die Reine und Angewandte Mathematik*. 133, 97-178.

Manuscript received 16 June 2012

Manuscript accepted 27 August 2012