Intermediate Data Clustering with k-means

Andy Allinger
24 Dec 2020

The k-means data clustering algorithm is extended to support missing data, mixed data, and to choose the number of clusters. A modified distance calculation finds solutions with a lower total error. Multi-dimensional scaling with missing data support is included. Full source code with no library dependencies is provided in FORTRAN with a full translation to C. Several example problems are discussed.

Contents

Background: Clustering in General

Introduction. K-means is the most commonly used method of data clustering, because it is simple, fast, and usually gives good results. The algorithm alternates between two main steps:

1. Assign each point to its nearest center
2. Let each center be the average of its points

One of the attractions of the method is how readily it may be generalized. A simple implementation [*] defines a "point" as a vector of real numbers, "center" as the arithmetic mean, and "nearest" by distance according to the Pythagorean Theorem. More generally, the "points" can be various kinds of data, the "centers" can be various kinds of averages, "nearest" distance can be generalized to a dissimilarity function, of which there are a bewildering variety to choose from.

Data clustering is a fundamental operation: it joins together things that are alike, and splits apart things that are different. It is a powerful technique applicable to a wide variety of problems. Among previous CodeProject articles, Samir Kunwar has discussed its use in Text Document Clustering, Rodrigo Costa Camargos has explored agglomerative clustering, and Arthur V. Ratz has applied k-means in recommender systems. The present project gives a k-means implementation that can be applied to messy real-world data and is also efficient.

Kinds of data

A more thorough discussion can be found in the chapter "Types of Data and How to Handle Them" in [Kaufman & Rousseeuw, 1990].

Continuous. These are vectors of real numbers. Suppose you have a weather monitoring station that records the temperature, barometric pressure, and humidity. Each observation can be represented by a vector with three components (T, p, h). Each of these measurements is a continuous variable. They do not need to have the same unit of measure.

Categories. Suppose instead that the data has only certain specific values it can take on. Instead of measuring temperature, you might record the weather as sun, rain, or snow. A heart murmur may be either present or absent. A question may be answered yes or no. An animal may be male or female.

Dichotomous. A special case occurs when one of two possible values of a variable is much more informative than the other. Suppose that the weather station records tornadoes as present or absent. Since tornadoes only occur under certain conditions, days that have tornadoes are alike, while days that do not have tornadoes may not be alike. Animals that have feathers have something in common, while animals that do not have feathers may not have anything in common. Dichotomous data is also called "binary asymmetric."

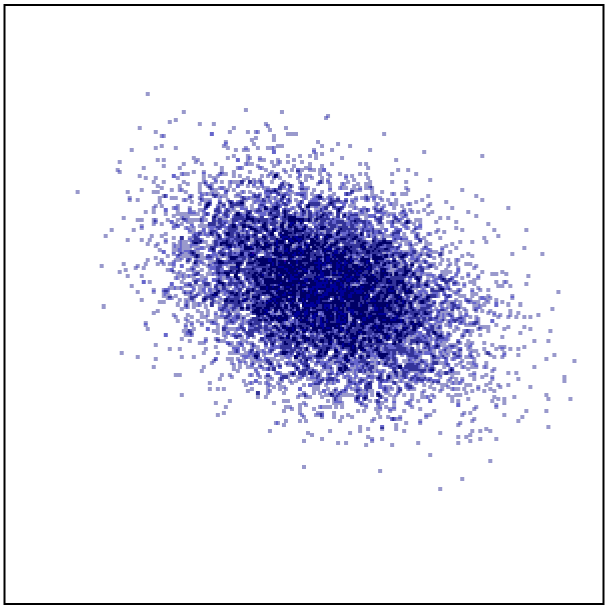
Ordinal. Suppose that instead of having numeric values, the temperature is recorded as hot, warm, cool, or cold. Of runners in a race, who came in first, second and third might be known, but their finish times might be unknown. An ordinal variable is like a categorical variable, but the values can be arranged in a definite order.

Time series. Suppose your weather monitoring station only measures temperature, but you take a measurement every hour for a year. Then the entire record for the year could be considered an observation, and it could be compared to the yearly records of other weather stations. (This program doesn't do time series.)

Histograms. Perhaps instead of a single measurement, the object is a whole collection of measurements forming a statistical distribution. Special dissimilarities, such as the Earth Mover's distance, are appropriate to compare histograms. (This program doesn't do histograms.)

Multimedia. Perhaps you data is audio, images, or video. The problem becomes difficult, but measures have been invented to compare their dissimilarity. [Deza & Deza, 2012] (This program doesn't do multimedia.)

Kinds of Dissimilarities



The dissimilarity function is the heart of data clustering. It is how difference is quantified. A wide variety of dissimilarity functions have been devised over the years. Some have a firm theoretical foundation, some do not. [Podani, 2000] [Sung-Hyuk, 2007] [Boriah, Chandola & Kumar, 2008] [Deza & Deza, 2012]

Foremost is the **Euclidean distance**, the kind taught in geometry, given by the Pythagorean theorem:

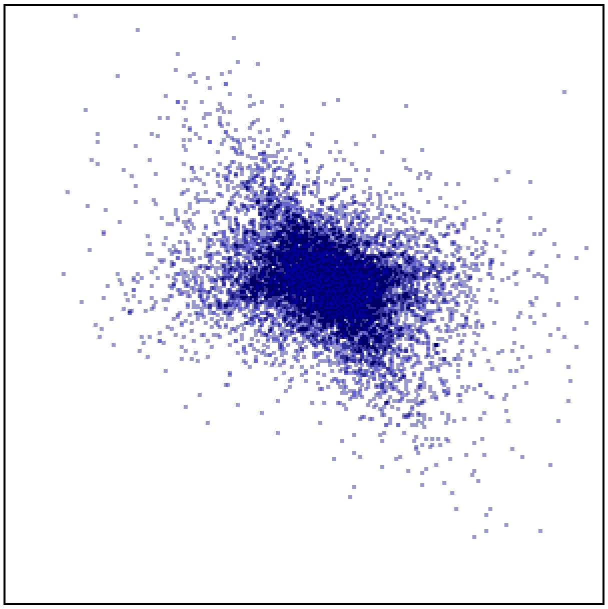
$$d(X,Y)=\sqrt{\sum_i (X_i-Y_i)^2}$$

The reason for its importance is that many real-world probability distributions are ellipsoidally contoured. The Euclidean distance is proportional to the negative logarithm of probability of a *spherically* contoured distribution. Thus, if the input data is suitably scaled so that the variables have similar variance, and if the input variables are mostly independent, then Euclidean distance approximates the unlikelihood that an object belongs to a group.

Manhattan dissimilarity. Some authors [Kaufman & Rousseeuw, 1990] recommend that instead of the Pythagorean Theorem, distance should be calculated according to the L1 metric (sometimes called Taxicab, City-Block, Manhattan):

$$d(X,Y)=\sum|X_i\ \text{-}\ Y_i|$$

The Manhattan metric is less sensitive to extreme (possibly wrong) values because the differences are not squared. However, it corresponds to a parallelogrammatically contoured distribution, which may not exist in nature. [*]



Correlation. Suppose that the data are time series of the discharge of some rivers for several years. Computing the standard correlation would show if there is a relationship between the flow in one river and another, independent of the average size of the river.

The correlation is a similarity measure. It may be converted to a dissimilarity measure by taking the arccosine. [*]

$$d(X,Y)=\arccos(r)$$

Cross-entropy Suppose that the data is some observations of a discrete variable. The usual example is that there are some jars of beans, with different proportions of red, black, and white beans in each jar. Then the cross-entropy between the sample and a particular jar:

$$H(S,M) = -\sum P_s \log(P_m)$$

is the negative log-probability that the sample was drawn from that jar. Here, S stands for sample, and M stands for model. Notice that this dissimilarity is not symmetric:

$$H(S,M)\neq H(M,S)$$

Dissimilarity Functions for Missing Data The simplest way to handle missing data is to compensate for missing values by a simple linear extrapolation from the non-missing data. John Gower proposed a similarity coefficient that was modified into the dissimilarity function [Gower, 1971][Kaufman & Rousseeuw, 1990]:

$$d(X,Y) = \sum\delta(x,y)f(x,y) \ / \ \sum\delta(x,y)$$

where δ(x, y) is an indicator function that has the value 1 if data is present for some variable for both objects X and Y, and has the value of 0 otherwise. The function f(x, y) depends on the kind of data. Continuous and ordinal attributes must be rescaled to the range 0 to 1, then f(x, y), is the absolute value of the difference x-y. For categorical data, f(x, y) is 0 if the values match and 1 if they do not. Dichotomous attributes receive special treatment. Because the {absent, absent} pair is considered completely uninformative, such an instance is treated as if it were missing.

Gower's dissimilarity seems to be the most widely used, but it is not the only possibility. In 1970, Burnaby proposed a measure of similarity that applies factors that gives more importance to independent attributes and to rare values. [Carranza, Feoli & Ganis, 1998] It can handle missing data in the same way as Gower's.

Another way to handle missing data is to replace it with the average for that attribute, as was proposed in [Anderson, 1971]. Still another possibility is to use an average dissimilarity when a value is missing.

Other variations are possible. The differences may be squared, instead of taking the absolute value. This results in a formula resembling the Pythagorean Theorem [Podani, 2000]. It is not generally necessary [*] to scale continuous attributes to 0...1, and other weighting schemes may be introduced, for example [Huang, 1997].

Minimizing Sum-of-Squared Distance. The goal of clustering is to partition the data so that objects that are near each other are in the same group. To assess how well this goal is attained, compute the residual error, defined as the sum of the squared distances from each point to the center to which it is assigned. It is sometimes incorrectly stated that the simple k-means method finds a local minimum of the error. [*] This does not take account of the effect that changing the assignment of the point will have on the center. To find a local minimum of squared error, the dissimilarity function must be the squared Euclidean distance with a correction factor of $N / N-1$

applied if the point belongs to the center, and a correction factor of $N / N+1$

applied if the point does not belong to the center. [Hartigan & Wong, 1979]

Kinds of Averages

Mean. This is the most familiar kind of average. The mean is sensitive to extreme values. It is not defined for categorical attributes.

Median. The median is the value that occurs in the middle of a sorted list. It is not defined for categorical attributes, either.

Mode. The mode is the most common value in a data set. It is defined for categorical attributes. If there are many possible values, then only a few objects in the set may actually have that value.

Medoid. The medoid is an object in the set that has the lowest total dissimilarity to every other object. It is defined for any kind of data for which the dissimilarity is defined. It has the same disadvantage as the mode. Since there can be no fractional categories, it cannot represent a diverse set.

Dissimilarity between Clusters

If two clusters can be represented by their means or medians, then the distance between them is easy to find. The situation for categorical variables is less clear. The mode or medoid may not represent the cluster accurately. If we take pairs of objects, one from each cluster, and compute the average dissimilarity, as does [Kaufman & Rousseeuw, 1990], some strange effects are possible. Consider two flocks of animals each half male and half female. Then the average dissimilarity between the flocks is 50%. Now consider two flocks of animals, one flock half male and half female, the other flock all male. The average dissimilarity between the flocks is again 50% !

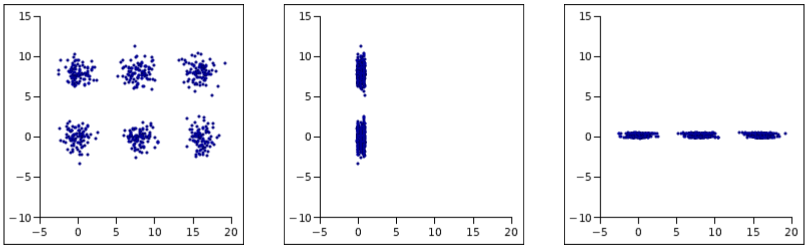
Determining the Number of Clusters

A significant weakness of k-means is that it requires the number of clusters, *k*, to be specified beforehand. The usual approach is to run k-means for several values of *k*, and then inspect the results. The residual error may be measured as the total distance of each point to its center. As *k* is increased, this residual decreases. The intuition is that there will be a significant decrease in the residual when the right value of *k* is reached, and that further additional clusters will not lead to significant improvements. This general approach is sometimes called the Elbow Method. Many variations of this idea have been proposed over the years. A survey of 30 methods for choosing the number of clusters was done by Milligan & Cooper in 1985. A software package for the GNU "R" statistical language was released [Charrad et al., 2014] implementing 30 methods, some of them the same. There does not seem to be any consensus at present on the best variant of the elbow method.

Another approach is to guess an initial value for *k*, then run a variation of k-means that provides for changing the number of clusters. [Ball & Hall, 1965] [Pelleg & Moore, 2000] [Kulis & Jordan, 2012] A disadvantage of this approach is that more parameters must be specified to decide when a cluster should be split, and when clusters should be merged.

It has been proposed that k-means may be run several times with different values of *k*, and a value of *k* should be chosen that yields the most consistent results. This approach can fail badly, as is shown in [Ben-David, von Luxburg, & Pál, 2005].

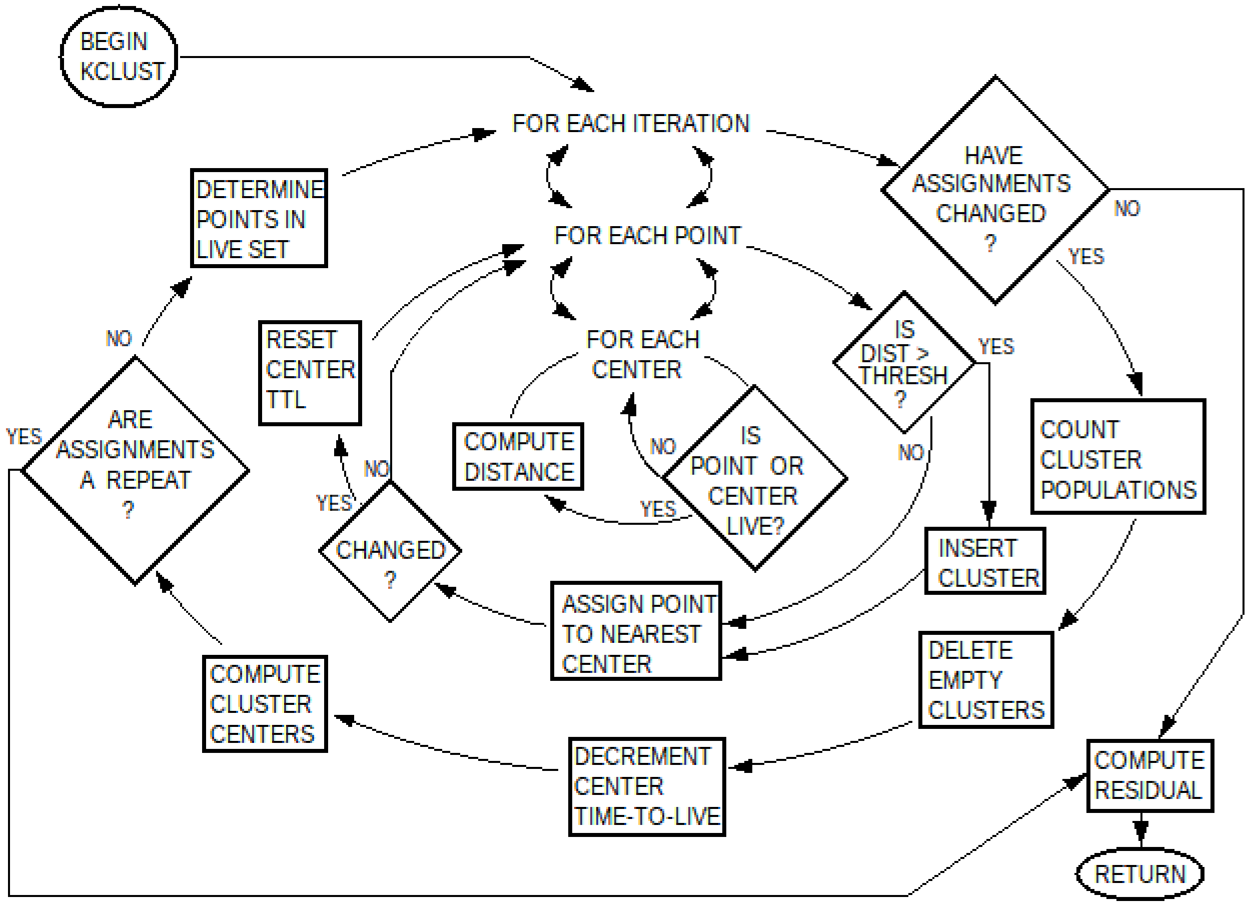
The number of clusters depends on the importance weights of the variables. To see this, suppose that initially there are six clusters. Reducing the importance of the horizontal direction makes two clusters. Reducing the importance of the vertical direction makes three clusters.



History

Much of this information comes from the review article by [Bock, 2008].

1950 Dalenius proposes a method to partition a 1-dimensional data set.
1953 Thorndike informally describes a clustering method using average distances. [Thorndike, 1953]
1956 Steinhaus proposes a k-means algorithm for the continuous case.
1957 Lloyd proposes a 1-dimensional k-means algorithm; his paper is not widely published until 1982.
1965 Forgy presents the basic k-means algorithm in a lecture and an abstract.
1965 Ball & Hall publish a more complex algorithm with provisions to merge and split clusters. [Ball & Hall, 1965]



Package Cluela

`CLUELA` is CLUstering with ELAborations. It extends the basic k-means algorithm to tackle the problems of missing data, mixed data, and choosing the number of clusters. It brings together original and legacy code in plain, honest FORTRAN, with a complete C translation. It is intended to be efficient for use in applications, and clearly written enough that it can be modified by other programmers.

Math is done in single precision. In statistics, the question is not whether the numbers are precise enough, but whether they mean anything at all.

Missing data support

We need some way to represent whether a datum is present or missing. It is possible to set aside some special value ("sentinel value") to represent missing data, but this leads inevitably to collisions. To really adequately indicate what is present requires another array. (Similar to the implementation in the *C Clustering Library.* [de Hoon, Imoto, & Miyano, 2010]) Each element in integer matrix NX should be set to 1 if the corresponding element in data matrix X is present, and 0 if it is missing.

The dissimilarity function is that of Gower, as modified by Kaufman & Rousseeuw, and extended to support importance weights for each variable [*] and generalized to support the Manhattan and Euclidean metrics:

$$d(X,Y)=\sqrt[L]{\frac{\sum_j \delta(x_j,y_j)w_j|x_j-y_j|^L}{\sum_j \delta(x_j,y_j)w_j}}$$

where δ(x,y) is an indicator function that has the value 1 if data is present for both X and Y, and is 0 otherwise.
L is 1 or 2, for the Manhattan or Euclidean metric, respectively.

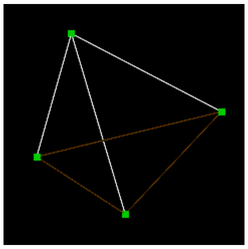
If option ROBUST is set, then the Manhattan metric is used, otherwise the Euclidean metric is used. Option ROBUST also computes centers as medians instead of means, so that CLUELA implements the **k-medians** algorithm.

Mixed data

The dissimilarity function as written accepts only continuous data. This was done not because there is any difficulty in implementing Gower dissimilarity directly, but rather so that averages could always be clearly defined. CLUELA implements k-means rather than the k-medoids algorithm of Kaufman & Rousseeuw because the medoid may not accurately represent the cluster.

To handle categorical data, subroutine RSCODE is included to recode a categorical variable as several continuous variables. The most obvious and most common way to do this is with **dummy coding**. A categorical variable with *c* possible values is rewritten as *c* continuous variables. The continuous variable that corresponds to the category of the object is set to 1, and the rest of the continuous variables are set to 0. Dummy coding contains redundant information, because if *c* - 1 of the dummy-coded values are known, the last value is fully determined. This is slightly inefficient, and can cause numerical problems if we wish to compute the covariance matrix (for example, to run PCA). It is preferable therefore to represent a categorical variable that takes on *c* possible values by *c* - 1 continuous variables. A **regular simplex** is an N-dimensional generalization of a regular polyhedron.

[McCane & Albert, 2007] A regular simplex of *c* vertices lies in a space of *c* - 1 dimensions. For example, the regular simplex with four vertices is a tetrahedron, and it is an ordinary 3-dimensional object. [*] If option ROBUST is set, then the data is embedded in a Manhattan simplex, such that the vertices are at distance 1 according to the Manhattan metric, otherwise the data is embedded in ordinary Euclidean space. Since the recoding increases the number of variables, the importance weights must be scaled down. The recoded data then must be accordingly scaled up to keep the dissimilarities the same as if simple matching of the categorical variables had been used.



Subroutine RSCODE requires input in the form of REAL approximations of small consecutive integers, that is, 1.0, 2.0, 3.0 ... The raw data to be studied will be in a much less convenient format. Two subroutines are provided for this conversion: JSCODE to recode arbitrary integers to consecutive small integers, and IREID to recode CHARACTER data as consecutive small integers.

Ordinal data may be left as it is, and treated as continuous. Alternatively, it may be preferable to suppose that rare, extreme values are more different than common values. Subroutine ORDNAL replaces ranks with their quantiles, scaled 0 to 1, so that the difference between "Cold" and "Cool" is more when "Cold" has few occurrences in the data set.

Dichotomous data cannot be represented with dummy coding or the regular simplex. It is necessary to embed it with MDS as described below. [*] The use of dichotomous variables will be demonstrated in the Soybean problem only.

Choosing the number of clusters

The user must specify KMIN, the fewest acceptable clusters; KMAX, the most acceptable clusters; and K, the default number of clusters. The subroutine runs k-means for *k* from KMIN-1 to KMAX and notes the improvement in the residual at each step for *k* from KMIN to KMAX. Of the many methods that have been proposed to determine the number of clusters, the **gap statistic** is well-regarded, but computationally expensive. A quick approximation was developed by [Pham, Dimov, & Nguyen, 2005]. The value of *k* which yields the minimum of this statistic is chosen, unless a critical value is not reached, in which case the default number of clusters is accepted.

Other Features

The initial centers are chosen by the usual k-means++ method. For each value of *k*, k-means is run ten times with different initial centers. The result with the least residual is kept.

Setting option SOSDO enables the correction factor of $N/N{-}1$ or $N/N{+}1$

that finds a local minimum of the sum of squared distance objective. This option conflicts with option ROBUST. It also conflicts with building a binary tree (see below), and so is not the default.

External Validation. Sometimes you may want to compare a partition generated with k-means against some other partition. For example, you might wish to test the clustering algorithm on some problem with a known answer. Two subroutines are supplied for this purpose. ARIND computes the Adjusted

Rand Index (ARI), which is a measure of accuracy that has a maximum of 1. It is 0 when the agreement between the clustering and the external standard is what would be expected by chance. It can take negative values, if the agreement is even worse than would be expected by chance. `VINFO` computes the Variation of Information. [Meilă, 2002] It measures distance between the clustering and the external standard, so lower values mean better agreement. Its minimum value is 0; its maximum depends on the size of the problem.

Multi-dimensional scaling: Bosses love graphs

If you have to communicate the results of your analysis, you will need some visual aids so that the audience will have something to look at. To visualize multivariate data, it must first be projected down to two dimensions. That can be done by Multi-Dimensional Scaling (MDS) [Laub, 2004], Singular Value Decomposition, or Principal Component Analysis. Subroutines `MDS` and `PCA` are included.

Multi-dimensional scaling takes a matrix of *squared* dissimilarities and gives coordinates of points in Euclidean space that are the proper distances apart. The technique is general and powerful. It was famously used by [Tobler & Wineburg, 1971] to locate ancient cities in Anatolia, using only mention of their names on the same clay tablet as a similarity. Apply the double centering formula to the matrix:

$$C_{ij} = -\left(\frac{1}{2}\right)\left(A_{ij} - \frac{1}{N}\sum_{k=1}^{N} A_{ik} - \frac{1}{N}\sum_{k=1}^{N} A_{kj} + \frac{1}{N^2}\sum_{k=1}^{N}\sum_{l=1}^{N} A_{kl}\right)$$

Factor the matrix, and multiply each eigenvector by the square root of its corresponding eigenvalue.

`MDS` as implemented in `CLUELA` is not so general; it is tied to the Gower dissimilarity function. It therefore supports mixed and missing data, and can be used to fill in missing values as a preprocessing step for routines that do not have missing data support, such as `KMNS` and `NEUGAS`. It is also necessary to use `MDS` if there is any dichotomous data, which cannot be handled directly by `CLUELA`.

PCA does not support missing data, but it may be much faster than MDS. It only needs to factor the PxP covariance matrix, instead of the NxN dissimilarity matrix which is usually larger. For PCA, we must first compute the covariance matrix:

Find the eigenvalues and eigenvectors, factoring the covariance matrix $V \Lambda V^T$. The projection is $R = X^T V$, using only the columns of V corresponding to the M largest eigenvalues. Notice that R is transposed relative to X.

Options are to use the covariance matrix or the comedian matrix. [Falk, 1997] The comedian

$$com(X,Y) = med(X - med(X))\ (Y - med(Y))$$

is a generalization of the median absolute deviation. It is a highly robust alternative to the covariance that is easy to compute.

Efficiency Improvements

k-means is usually described as fast, or at least faster than some other clustering algorithms. The computational cost of basic k-means is NPKi operations, where N is the number of objects, P is the number of variables, K is the number of clusters, and i is the number of iterations required for convergence. It is possible to modify the algorithm to be faster than this, and these efficiency improvements involve no loss of accuracy.

The bottleneck step of k-means is the computation of the distance from each point to each center. It is possible to find the nearest center to a point without measuring the distance to every center, by putting the centers into a binary tree data structure. [Chávez, Novarro, & Marroquín, 1999] `CLUELA` uses a generalized hyperplane tree. Any data set which has a dissimilarity between each datum may be split by choosing two objects, and assigning the remaining objects to a partition by whether they are nearer the first or the second. If the data and dissimilarity function satisfy the requirements of a metric space, then the triangle inequality may be used to efficiently search the tree for nearest neighbors. (see Appendix) There are four requirements of a metric space:

1. Dissimilarities are not negative
2. Dissimilarity is zero if and only if the objects are identical
3. Dissimilarity between objects is symmetric
4. The triangle inequality: The sum of dissimilarity of A to B plus the dissimilarity of B to C must be greater than or equal to the dissimilarity of A to C.

The Manhattan dissimilarity and the Euclidean dissimilarity are metrics. Some other dissimilarity functions of present interest are not metrics. In particular, *squared* Euclidean distance is *not* a metric. The Gower dissimilarity is not a metric when there is missing data. The dissimilarity function that minimizes the sum of squared distance is not a metric. The generalized hyperplane tree is most valuable when K is large and P is small. `CLUELA` builds a generalized hyperplane tree when the requirements of a metric space are satisfied. Otherwise, it exhaustively compares each point to each center.

In a metric space, if the distance to a point from its nearest center is less than or equal to half the distance of that center to any other center, the point can belong in no other cluster. In that case, no distances need to be computed. [Elkan, 2003] (See Appendix)

Hartigan & Wong introduced the concept of "live sets" of points and of centers to eliminate needless distance calculations. [Hartigan & Wong, 1979] A center is in the live set if any points were added or removed from it on the last iteration. A point is in the live set if the distance to its nearest center has increased since the last iteration. For each point, live centers may have become nearer, and so each point must be compared to all live centers. For a live point, its present center is farther, and so the point must be compared to all centers dead and alive. If neither the point nor the center is in its respective live set, then no change is possible and the computation may be skipped. This reasoning does not depend on the properties of a metric space. `CLUELA` always makes use of the live sets.

It is possible with some kinds of data and dissimilarity functions for k-means to get stuck in a loop and never converge. The assignment array `Z` is hashed, and the hash is stored in a ring buffer. On each iteration, the present hash is compared against the previous hash values to detect if the algorithm is stuck in a loop. If it is stuck, it exits without an error. (It may not be possible to get stuck with the scalar data in `CLUELA`. The loop detection feature has been retained for the sake of programmers who may modify the subroutine to handle less nicely behaved data.)

To compute the median of each cluster, the data must be repacked into a work array. To avoid making K passes over the data matrix to fill the work array, passive sorting is used on the assignment array. Then it is possible to access the points for each cluster without referencing other objects. Since the assignment array is small integers, it can be sorted especially efficiently with the **unstable radix sort**, at a computational cost of N log(K).

By using the binary tree, the computational expense of the assignment step is reduced from NPK to perhaps as low as NP log(K) depending on the effectiveness of the tree. By sorting the data, the cost of computing the centers is NP, the size of the data matrix; or N log(K), the cost of sorting. That leaves as the bottleneck step the k-means++ initialization. An alternative initialization is to use hyperplane partitioning. This is much like putting the data into a Generalized Hyperplane Tree, except forgetting all the details of the tree structure. Using this quick initialization, the whole clustering process is less than O(NPK).

About the translation

The original code was all in "Legacy" `FORTRAN` -- that is, whatever will f2c will accept. That's standard `FORTRAN 77` with a few extensions. f2c translated it, and then it was edited to remove dependencies on `libf2c` and make the matrices doubly subscripted. Double deference may or may not run faster, but it's a lot easier to read. There is no random.c or machine.c because machine constants and random numbers are part of standard `C99`. Likewise, there is no array.f because there is nothing special about declaring a matrix in `FORTRAN`. The arguments lists are often different. Some arguments in `FORTRAN` are needless in `C` and vice versa. There are no header files.

Using the Code

The solution to most clustering problems will require a pre-processing phase, the clustering itself, and then some additional post-processing.

Subroutine `CLUELA` accepts as input only vectors of real numbers. Data in the form of text labels or integers can be converted to small consecutive integers with calls to `IREID` or `JSCODE`, respectively. Ordinal data optionally may be rescaled with `ORDNAL`. If you decide to standardize some or all of the variables, the mean and standard deviation may be computed with `AVEDEV`, or the median and median absolute deviation may be computed with `MEDMAD`. The range may be found with `LEAGRE`. Standardizing shrinks the variables of large variance, so it may be counterproductive. If the variables are in different units some kind of transform will be necessary. Rescaling to the range 0 to 1 may be a better choice:

$$x'_i = (x_i - x_{min}) \;/\; (x_{max} - x_{min})$$

The classical k-means implementation `KMNS` and the alternative Neural Gas clustering algorithm `NEUGAS` are included in the distribution for the sake of comparison. Subroutine `CLUSTE` is a variation of `CLUELA` that skips repeats and uses a quick initialization for faster theoretical efficiency.

In the post-processing phase, `MDS` or `PCA` may be called to project the data a lower dimension so that it can be graphed. `MDS` may also be used in the preprocessing phase. It accepts dichotomous data, which `CLUELA` and `CLUSTE` do not. It also fills in missing data, so that `KMNS` or `NEUGAS` may be used. Write the 2-D results to a spreadsheet with `WCSVSP`.

Other subroutines of interest to the applications programmer are:

`MEDIAN`: the median of a vector of real numbers, by Alan Miller
`SAFDIV`: check whether division will overflow, based on a program by Julio
`ALMEQ`: check whether two real numbers are almost equal
`SHUFFL`: make a random permutation of integers 1...N
`OSIMAN`: optimize a set of parameters by simulated annealing
`DINIT`: choose initial cluster centers according to the k-means++ method
`HCLUST`: choose initial point assignments by hyperplane partitioning
`KCLUST`: k-means or k-medians clustering, with missing data support and the possibility to insert or delete clusters

For calling sequences of theses subroutines, please refer to the source code. All the important subroutines have prologues that explain how to use them.

Example problems

- The soybean demonstrates how to call `CLUELA` to cluster mixed and missing data with dichotomous variables. The results are checked using `daisy` and `pam` in `R`.
- The wine problem shows that the number of clusters depends on the importance weights of the variables.
- The credit approval problem shows how to call `KCLUST` directly.
- The round problems shows that `CLUSTE` is faster than `KMNS` for many clusters are few dimensions.
- The horse colic problem demonstrates a way to optimize the importance weights. It also shows how to call `NEUGAS`.
- The votes problem is a case where the robust principal components are very different from the standard principal components.
- The forest fire problem considers the case of cyclic variables, and shows how to use embedded components that are imaginary numbers.

Most of these problems come from the University of California at Irvine Machine Learning Repository. [Lichman, 1987] Soy, wine, credit, and votes were originally posed as classification problems. The known classes can be used as an external validation measure of the clustering result.

Training sets and test sets have been combined as soybean.both and horse-colic.both. This is permissible, because clustering does no training. All the prior assumptions are built into the dissimilarity function and the importance weights.

Importance weights for the Wine problem came from came from optimizing a spectral clustering residual, and did not involve use of the class labels. The round, credit, votes and fire problems use equal weights. The weights for the horse colic problem are determined thru optimization by simulated annealing. For the soybean data, weights were obtained by some kind of linear transform, and I can't remember if it involves looking at the class labels, so please don't take them too seriously.

All the data sets are included in the distribution.

The Soybean problem

Some ill plants are suffering from 19 diseases, manifested by 35 symptoms, which are a mixture of continuous, categorical, and ordinal variables. The symptoms and the diseases have been identified by human experts, which should make us suspicious, since this is not a double-blind experiment. It is possible that the agronomist's diagnoses have influenced his description of the symptoms. Much of the data is missing. Twelve variables are binary, and have the value of "present" in less than 25% of the instances. These are reasonable candidates to treat as dichotomous variables. We will compare the results of treating these variables as categorical or dichotomous, and then check these results by re-running the problem in GNU `R` calling `daisy` and `pam`.

Combining training and test sets, the number of objects `N` is 683. The number of classes `G` is 19. Consider possible numbers of clusters from `KMIN`=1 to `KMAX`=50. The number of original variables `O` is 35. Many of these will have to be recoded. The hard-coded array `CAT(O)` gives the number of categories for each input variable. For continuous and ordinal variables, write zero. For dichotomous variables, write *one*. Set `CATMAX` to the largest value in `CAT`. Each categorical variable will have to be recoded with a number of continuous variables one fewer than the number of categories. The data matrix that is input to the clustering routine is `X(P,N)`. To find `P`, start with `O` and add `CAT(H)-2` for each categorical variable. The subroutine does *not* do this automatically; it must be hard-coded in the main program. It is important for the analyst to examine the data and think about the problem.

Studying the input, we find that P=54. The attribute values in soybean.both are integers 0, 1, 2... for available data and "?" for missing data. `RSCODE` requires categories to be numbered starting with 1.0. For available data, `X` is set to 1.0 + the input and `NX` is set to 1. For missing data, `X` is set to `XBOGUS`, and `NX` is set to 0.

The first `O` rows of `X` are filled with the input. The continuous and ordinal variables are rescaled 0 to 1 to match the treatment in `daisy`. Call `MDS`, treating the dichotomous variables as categorical, and obtain embedding matrix `VC`. Then change the dichotomous variables from {1.0, 2.0} to {0.0, 1.0}, because by convention this is the coding for the dichotomous {absent, present}. Call `MDS`, recognizing dichotomous variables, and obtain embedding matrix `VD`.

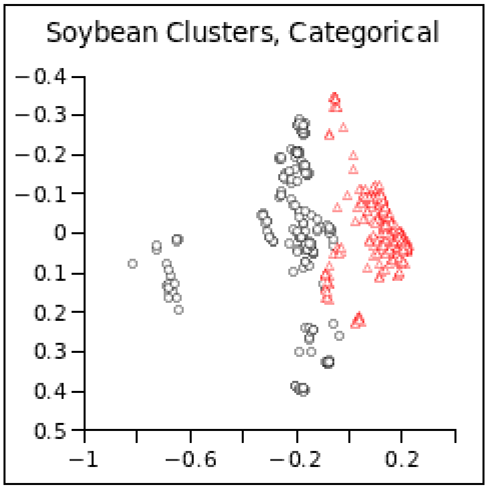
Call `RSCODE` to recode the categorical variables with vertices of the regular simplex. This takes up more memory. `RSCODE` starts at the last row and works backwards, moving the data to the end of the array, returning `X(P,N)` of all continuous data. The class labels were given as text strings; they must be converted to integers 1, 2, 3... with a call to `IREID`. Set a default `K = 20` and we are finally ready to call the main subroutine. On return, the results are compared to the class labels, and the projected data is written to soy_clusters.csv and soy_classes.csv. The F statistic of [Pham, Dimov & Nguyen, 2005] is returned in array `F()` for each possible number of clusters `KMIN`...`KMAX`. Small values of F are supposed to indicate a good number of clusters.

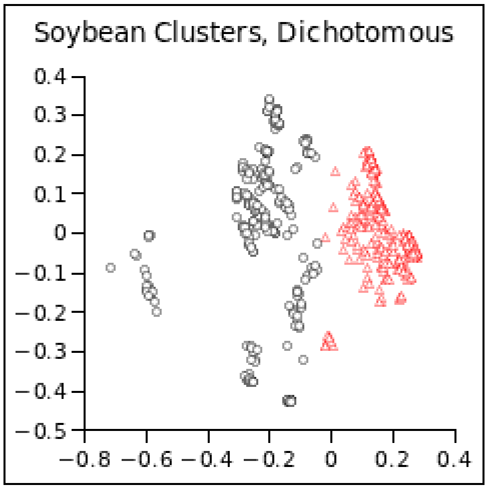
Next, repeat the process using dichotomous variables by calling `CLUELA` with embedded matrix `VD` in place of `X`. Print the output as before. As a check, call `MDS` again on the recoded `X`, taking all variables to be continuous, obtaining embedding matrix `VR`. Comparing `VR` to `VC` ensures that the coordinates returned by `RSCODE` represent objects that are equally dissimilar to each other as the original data.

Running the program prints to the terminal:

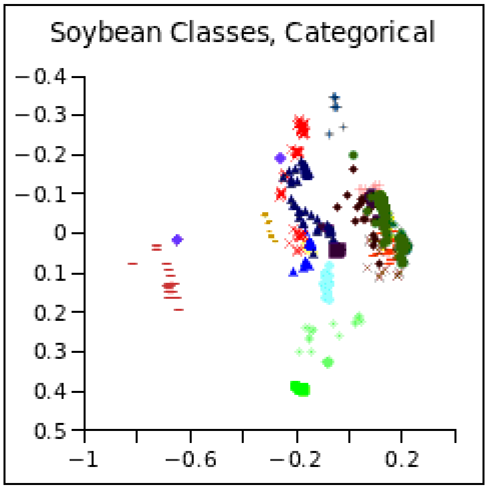
```
cluela returns #0, made 2 clusters
Returned residual:    1.566e+04
Adjusted Rand Index: 0.154
Variation of Information:   3.025
cluela returns #0, made 2 clusters
Returned residual:        124.6
Adjusted Rand Index: 0.164
Variation of Information:   3.004
Greatest difference between original and recoded embeddings:  5.60e-04
program complete
```

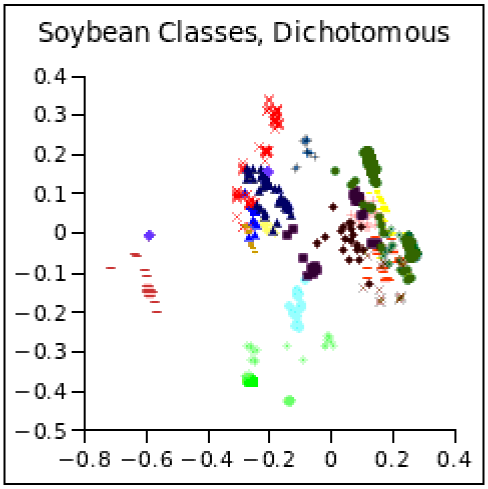
This value of the Adjusted Rand Index (ARI) indicates that the clustering is only 15% better than guessing. Examining the graph of the clustering results, we see that `CLUELA` has produced a reasonable-looking partition of two clusters. The orientation of the axes of an embedding produced by MDS or PCA is arbitrary. Flipping the vertical direction, the differences between categorical and dichotomous is minor.



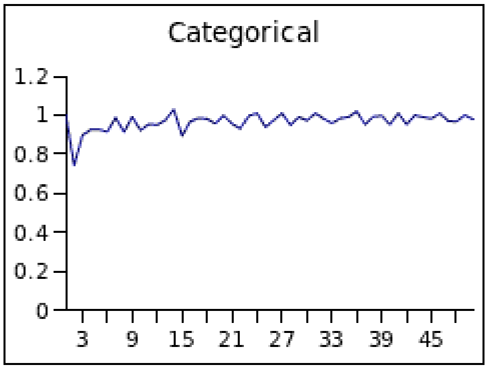


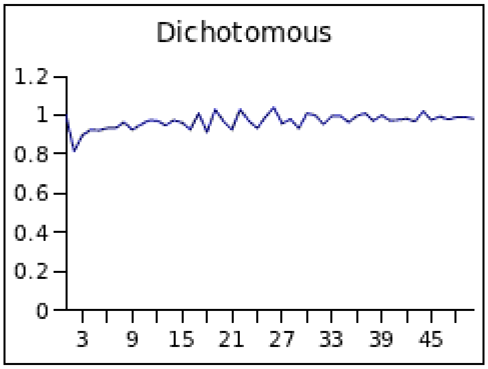
The classes are distributed in a much more complex way:





A plot of the F values shows a strong tendency for two clusters, and nothing at 19.





This is grounds for us to either be suspicious of our approach to the Soybean data, or of the F statistic. We will see in the next problem that the F statistic is helpful sometimes but not always.

Let's compare these results to a solution in GNU R. Start R in interactive mode, change directory, and load the package "cluster":

```
R
```

```
setwd('~/cluela')
```

```
library(cluster)
```

Read the file into data frame x, set variables 3, 4, 6, 8, and 10 as ordinal and review the input:

```
x <- read.table('soybean.both', header=FALSE, sep=",", na.strings="?",
colClasses=list("character", "numeric", "factor", "factor", "factor", "factor",
"factor", "factor", "factor", "factor", "factor", "factor", "factor", "factor",
"factor", "factor", "factor", "factor", "factor", "factor", "factor", "factor",
"factor", "factor", "factor", "factor", "factor", "factor", "factor", "factor",
"factor", "factor", "factor", "factor", "factor", "factor"))[2:36]
```

```
x[3] <- ordered(x[,3])
```

```
x[4] <- ordered(x[,4])
```

```
x[6] <- ordered(x[,6])
```

```
x[8] <- ordered(x[,8])
```

```
x[10] <- ordered(x[,10])
```

```
str(x)
```

This produces:

```
'data.frame':    683 obs. of  35 variables:
 $ V2 : num  6 4 3 3 6 5 5 4 6 4 ...
 $ V3 : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ V4 : Ord.factor w/ 3 levels "0"<"1"<"2": 3 3 3 3 3 3 3 3 3 3 ...
...
```

Set the weights, call `daisy` to compute the matrix of dissimilarities between each object. Review the dissimilarities:

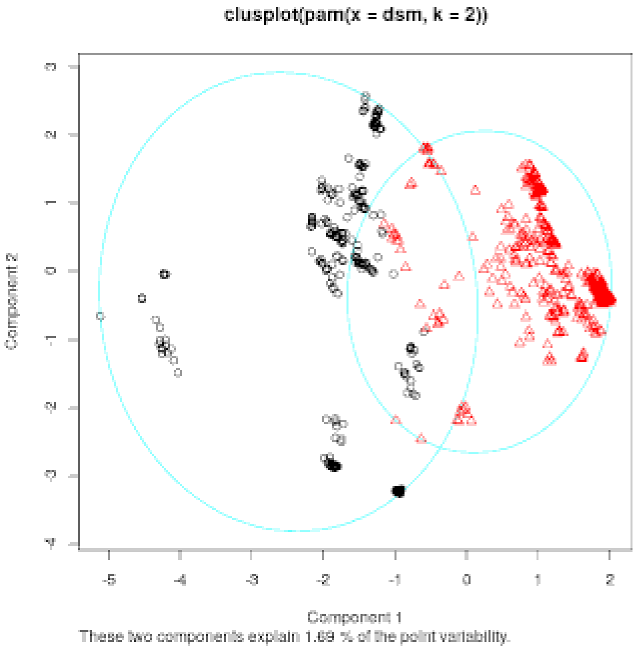
```
w <- c(2.742, 1.944, 6.565, 2.532, 2.200, 2.227, 1.386, 2.997, 1.378, 2.186, 3.578,
4.254, 3.271, 4.313, 4.405, 2.795, 5.595, 12.57, 3.092, 3.847, 2.163, 2.124, 3.362,
2.623, 12.17, 12.57, 12.57, 3.162, 2.544, 4.050, 4.626, 4.764, 4.301, 5.215, 3.464)

dsm = daisy(x, metric="gower",
type=list(asymm=c(5,16,17,20,23,25,27,30,31,32,33,34)), weights=w)

summary(dsm)
```

This produces:

```
232903 dissimilarities, summarized :
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.2910  0.4212  0.4139  0.5327  1.0000
Metric :  mixed ;  Types = I, N, O, O, A, O, N, O, N, O, N, N, N, N, N, A, A, N, N,
A, N, N, A, N, A, N, A, N, N, A, A, A, A, A, N
Number of objects : 683
```

Call `pam` to make 2 clusters by k-medoids, and plot the results:

```
result <- pam(dsm, 2)
clusplot(result, col.p=result$clustering)
```



That looks like the output produced by `CLUELA`. Check this by printing the embedded coordinates:

```
cmdscale(sqrt(dsm), add=FALSE)
```

This produces:

```
                [,1]          [,2]
  [1,] -0.218908503 -2.884794e-02
  [2,] -0.214924352 -2.570253e-02
  [3,] -0.205159906 -1.364315e-02
...
```

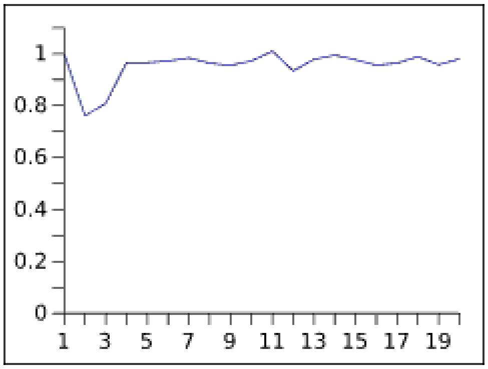
which confirms the result. Quit `R`.

```
q()
```

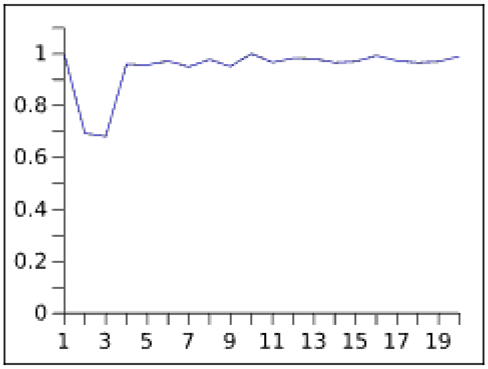
The Wine problem

The Wine data set is all continuous variables. `N`=178, `P`=13, `K`=3. Let us focus on trying to find the correct number of clusters. In [Pham, Dimov & Nguyen, 2005] it was reported that their proposed method chose the correct number of clusters.

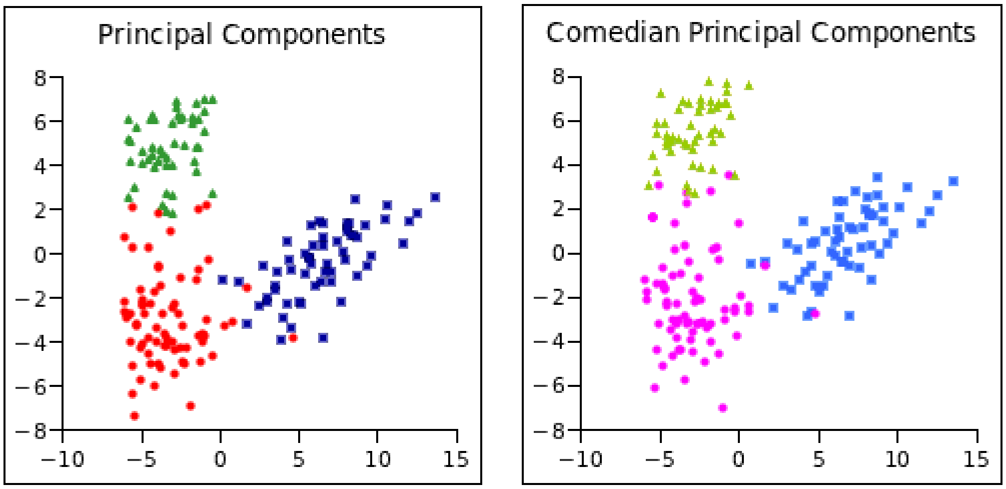
As a first attempt, running `CLUELA` on the raw data returns pretty lousy accuracy. Reading the journal article a little more carefully, we notice that we need to standardize the input. The plot of the F statistic now shows a minimum at 2, not 3.



Reading the journal article considerably more carefully, Pham et. al. were running their own peculiar variant of k-means that we do not care to replicate. Instead, we will use special (*very* special) importance weights. These were obtained by optimizing the residual of the Bach-Jordan spectral clustering algorithm [Bach & Jordan, 2003] with subroutine PRAXIS by Richard Brent. This doesn't use the class labels, but it sneaks in a bias, because the weights have been optimized under the assumption that K=3. The F values now show a minimum at 3 that is slightly less than the value at 2.



Moral: a good statistician can prove anything. And an evil one can too. There is a tiny difference between the ordinary and the robust PCA projections.



The Credit approval problem

This is a two-class classification problem, of 15 attributes which have been deliberately obscured. The data is read and recoded much as before. N=690, O=15, G=2, P=38. Continuous variables are rescaled 0 to 1. The weights are set equal.

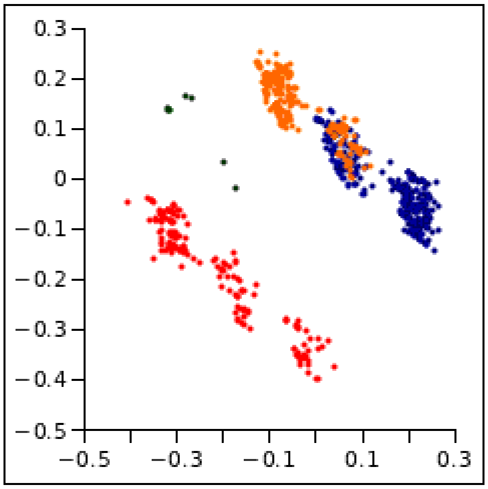
When the CLUELA interface is used, THRESH is set to a very large number, and KCLUST is repeated 10 times for every number of clusters from KMIN to KMAX. Calling KCLUST directly skips the repeats and allows for a different way of determining the number of clusters. Set K = 2. Call DINIT to apply the kmeans++ initialization, then call KCLUST using the value of THRESH returned by DINIT. A point which lies farther away than THRESH from its nearest center is made a new center. This produces:

```
kclust returns #0, made 2 clusters
```

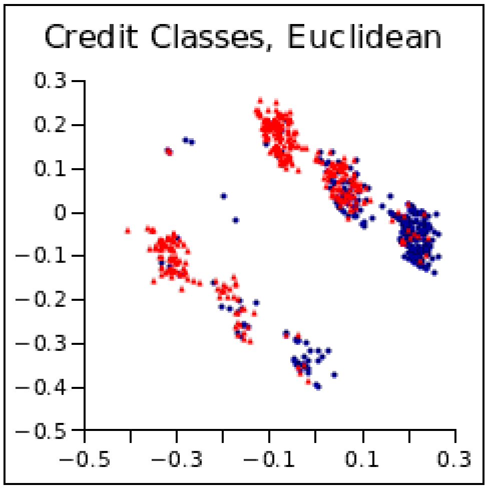
Next call DINIT, divide THRESH by 2, and call KCLUST. This produces:

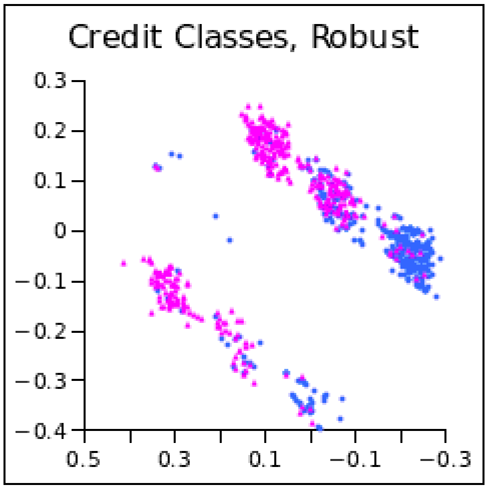
```
with half the threshold distance, kclust returns #0, made 4 clusters
```

A run of the program found these clusters:



but there is a considerable element of chance! The classes may be plotted by MDS on their Euclidean or Manhattan distances:



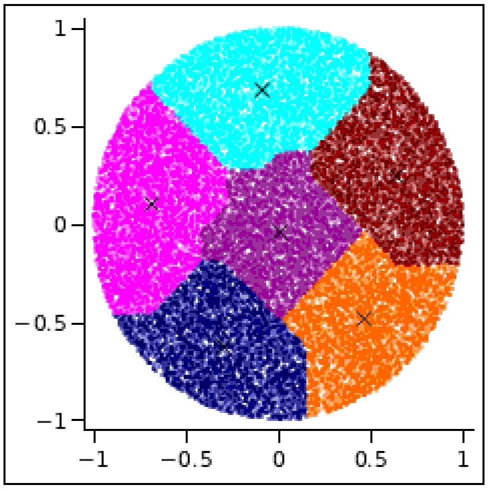


Allowing for flipping the horizontal axis, the difference between the graphs is tiny.

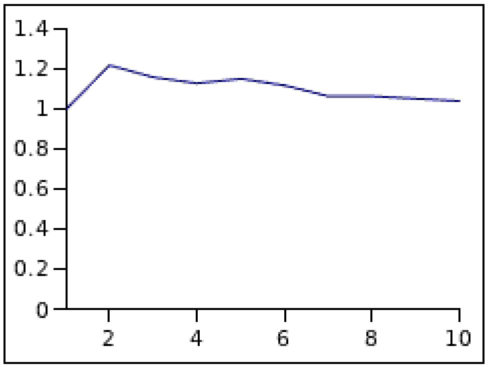
Problem "round" - the unit circle revisited

The `t_round` program generates 30,000 points uniformly at random on the unit circle. We saw this problem in "Beginning Data Clustering" [*] and solved it with the simple k-means algorithm presented there. Now that we have `CLUELA` we can compare those results with the results from using the `ROBUST` and `SOSDO` options.

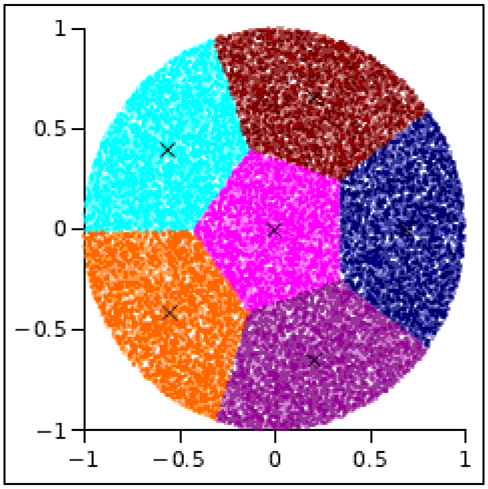
The `ROBUST` option invokes the Manhattan metric. This is a non-Euclidean geometry, and we need to be prepared to accept that the plot of the results in this approximately Euclidean region of space-time on your computer monitor are going to look kind of weird.



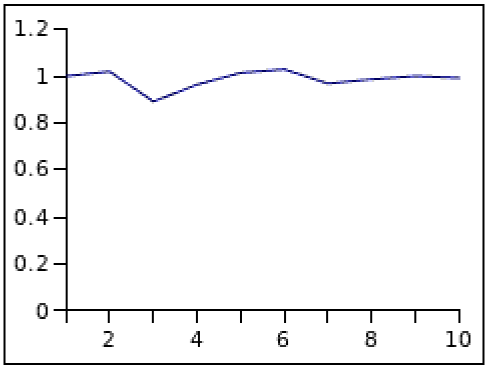
Pham, et al's method of choosing K assumes Euclidean space. [Pham, Dimov & Nguyen, 2005] It is no surprise that it shows no clustering tendency in Manhattan space.



Setting the `SOSDO` option instead returns the familiar result.



The F values show a tendency to form three clusters, which replicates the result of Pham, et. al.



For a large number of clusters, the binary tree is an important time saver. Set K=1000 and race CLUSTE against KMNS.

```
cluste returns #          0  in   0.677361012      seconds. Made         1000
clusters with residual    14.0591612
```

```
kmns returns #          0  in    1.31767201      seconds. Made         1000
clusters with residual    13.8801050
```

So CLUSTE is faster, but KMNS is more accurate.

The Horse Colic Problem

This data set is somewhat difficult. First there are some possible mistakes. The young/adult field was supposed to be coded {1,2} but it is instead {1,9}. That's easy to fix, but still strange. The field "capillary refill" is supposed to be coded {1,2}, but it contains a value of 3. This is probably a mistake, but RSCODE will fail if we don't set CAT=3 for this variable. Some of the ordinal variables are in the wrong order. For example,

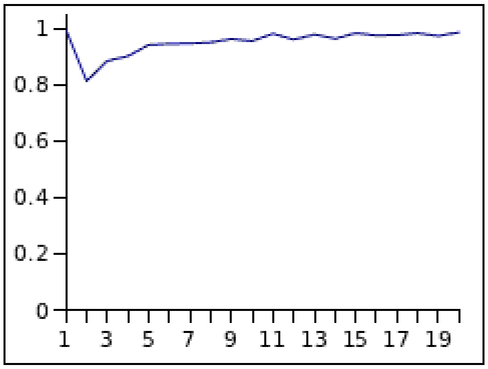
```
!       extremities temp   Ordinal            1 = Normal
!                                             2 = Warm
!                                             3 = Cool
!                                             4 = Cold
```

So we need to swap 1 and 2 so that the sequence is Warm, Normal, Cool, Cold.

Three attributes in the horse-colic.both are diagnosis codes. These are not categorical variables, because diagnoses are not mutually exclusive. A horse can have more than one illness. To represent this for `RSCODE`, it needs to be converted to a categorical variable. First, we have to count the distinct diagnoses. There are 62 of them. Each possible diagnosis is treated as a binary variable. So we use the first 24 attributes, and convert the next three attributes into 62 variables, and then ignore attribute 28. So N=368, O=24+62=86.

The attribute hospital number takes on 346 different values. This is an extravagant number to have to recode into dummy variables. We can see that recoding is not feasible when the number of categories becomes very large. Some analysts choose to combine categories in these situations. It would be reasonable to omit this variable entirely. It is also possible to write a dissimilarity function that handles categorical data directly, and thus have no need to recode it. We shall suffer the consequences of recoding, and get P=440.

Next, we recode the data and call the clustering subroutine much as before. `CLUELA` searches for a good number of clusters from `KMIN`=1 to `KMAX`=20. There appears to be a preference for two clusters.

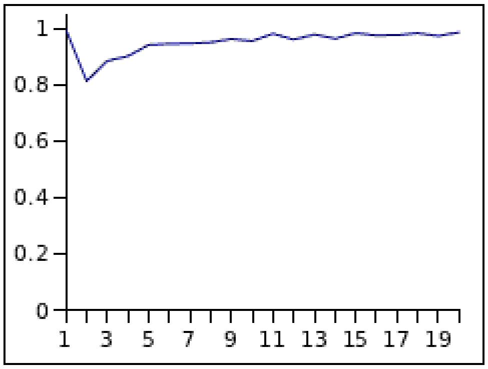


Consider the problem of choosing a set of importance weights in a *principled* way. Assigning a point to one cluster rather than another is an abrupt shift, and so the methods of optimizing continuous functions are not well-suited. **Simulated annealing** is an optimization technique that is versatile, and also simple to apply. The objective is to minimize the total squared distance of points to centers, but there are other factors that must be taken into account.

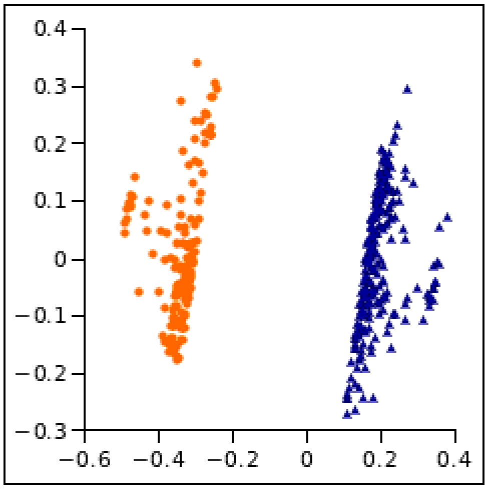
- Weights must be constrained to sum to 1, so that they are not all driven to zero.
- There must be a factor giving preference to equality, to prevent the trivial solution of all the weight on a single variable.
- The difficulty of an optimization problem increases rapidly with the number of variables. The weights to be optimized should be of the 25 input variables, not of the 440 recoded variables.

The objective function is then $h = u \sum w^a$

where u is the residual returned by `KCLUST`, w_j are the weights to be optimized, and a is an exponent chosen by the analyst. First, cluster the unweighted data. Values of the Pham f statistic show a preference for two clusters:



Taking the penalty exponent PEX=2 resulted in a very well-separated clustering:

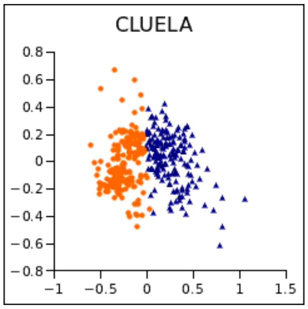


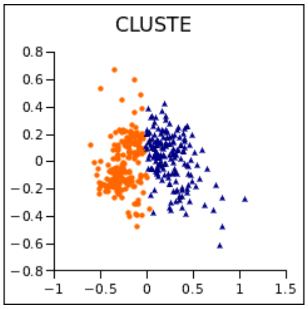
The weights have concentrated on a few categorical variables. There is a clear structure, but much of the data is being ignored. This is not what we want. Try again with a stronger penalty PEX=3.

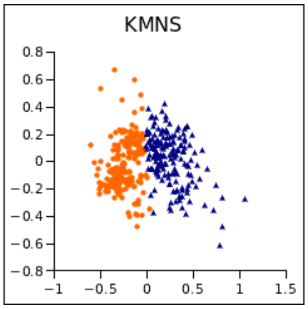
```
penalty exponent 3.000000 and robust=F yield optimized weights:   0.049  0.055
0.031  0.004  0.034  0.004  0.053  0.053  0.042  0.050  0.022  0.054  0.053  0.051
0.051  0.042  0.052  0.045  0.017  0.002  0.047  0.032  0.049  0.050  0.058
```

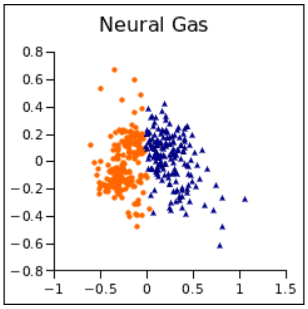
A few of the weights have been shrunk towards zero, but most of the problem data is being taken into account. Using the optimized weights, embed the data with MDS to fill in missing values so that we can compare CLUELA, CLUSTE, KMNS and the rival Neural Gas algorithm. The **Neural Gas** algorithm [Martinetz & Schulten, 1991] is equivalent to data clustering. It was developed with a focus on learning a representation of the data space, rather than partitioning a data set. The output to the terminal is:

```
cluela returns #0  Residual: 115.230087

cluste returns in 0.140965 seconds.  Residual: 115.230087

kmns returns in 0.024483 seconds. Residual: 115.223747

neugas returns in 0.055848 seconds.  Residual: 115.556557

program complete
```









The graphs are often identical. These results show that `CLUELA` is as accurate as `KMNS` when using the `SOSDO` option, but that `KMNS` is faster. For large `P` and small `K`, `CLUSTE` is slower. Neural Gas is computationally expensive, but effective.

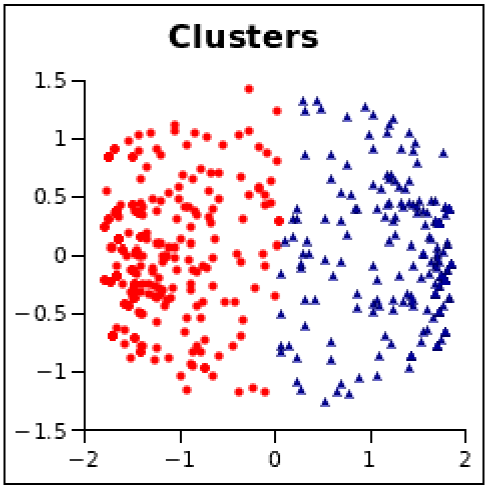
Is the method for determining the weights satisfactory? Well, it's better than nothing. The analyst still has discretion in choosing the penalty exponent, the simulated annealing parameters, how to standardize the continuous attributes, and whether to use robust mode. Simulated annealing is a stochastic method, and so the results are different every run of the program. An unscrupulous statistician may run the program until it returns the answer he is looking for. Also there is no way to disentangle the problem of choosing the weights from choosing the number of clusters.

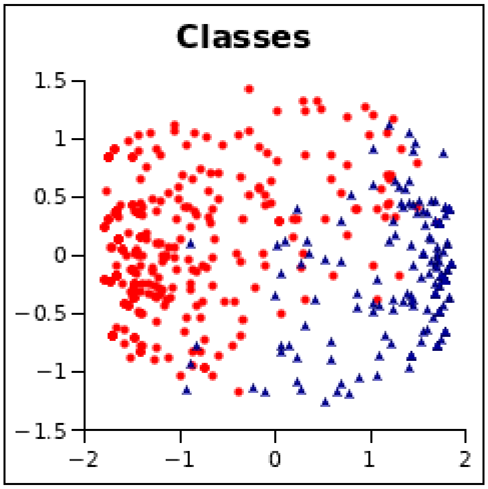
House votes '84

The House Votes data set is a record of the votes cast on some bills by the U.S. House of Representatives during its 1984 session. The goal is to classify each representative as a Democrat or a Republican. N=435, P=16, G=2. The votes are 'y', 'n', and '?'. We first attempt to treat the question marks as missing data. `CLUELA` returns with `IFAULT=10`, because there is a null object in the data. Somebody in the Congress didn't cast a vote on any of these bills. The k-means algorithm has no way to deal with null objects. [*]

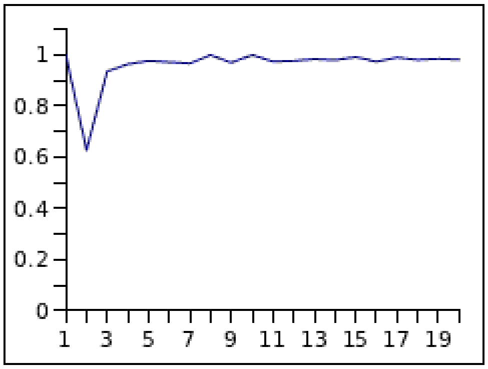
In this case there is an easy solution. Instead of treating the votes as categorical data, treat it as continuous data, and let n⇒0.0 ?⇒0.5 and y⇒1.0

The clustering results somewhat match the classes. The adjusted Rand index is 0.58

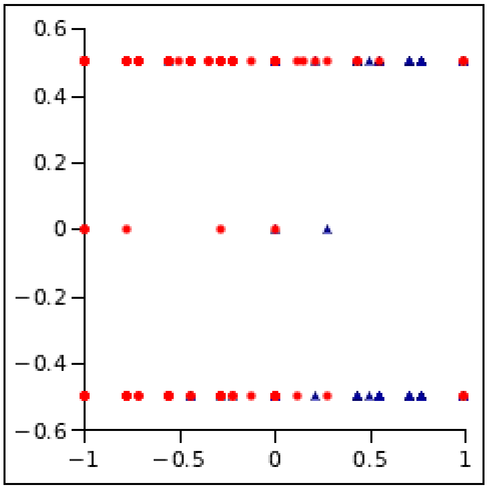




The F values show a strong tendency to form two clusters, as they should.



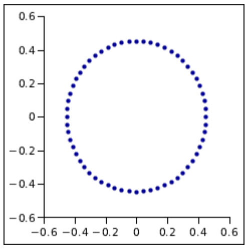
Repeat the problem in robust mode. The adjusted Rand index does not change much. It is 0.53. The embedding is entirely different:



The ordinary embedding depends on the covariance matrix, and the robust embedding depends on the comedian matrix. The covariance matrix of discrete variables is a full matrix, but the comedian matrix may have many zero entries, and so there is no resemblance.

The Forest Fires problem

The forest fires data contains measurements of the area burned in fires in a natural area in Portugal. It records the location of the fire, the month, the day of the week, temperature, humidity, wind speed, rainfall, and some statistics that were devised to predict fires.



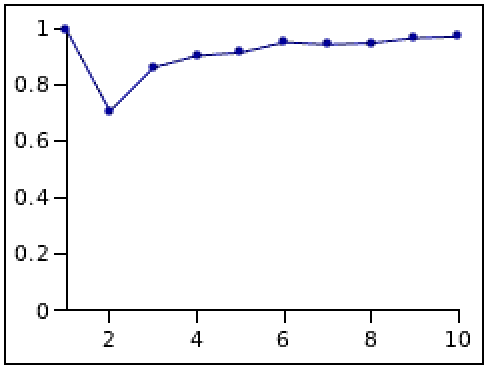
Month and day pose a special challenge, because they are cyclic variables. The difference between December and January is one month, not eleven. `CLUELA` does not have support for cyclic variables, but it can be embedded using `MDS`. The small program t_embed creates a 1D data set of numbers 0 to 1 and embeds it as cyclic data. This produces many dimensions out the single input dimension, but the first two are the most important. The output is exactly what was hoped for. Multi-dimensional scaling is a very powerful tool.

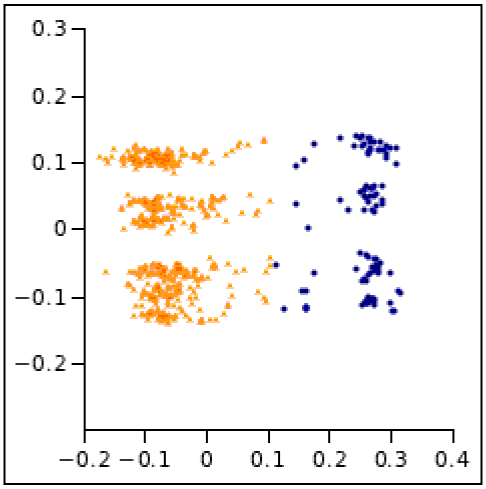
The first directions returned by MDS are the most important, and often an approximation is all that is desired. MDS represents the original distances *exactly*, but all `N` directions must be used. This gets a little bit tricky. To compute the embedded coordinates, an eigenvector of the dissimilarity matrix must be multiplied by the square root of the corresponding eigenvalue. Directions that correspond to positive eigenvalues are real directions. Directions that correspond to negative eigenvalues are *imaginary* directions. The greater the difference between two objects in an imaginary direction, the nearer they are to each other. A data set that includes real and imaginary directions is called pseudo-Euclidean [Laub, 2004] and is definitely not a metric space.

`CLUELA` only accepts real numbers as input, but setting a weight negative will serve to treat the variables as imaginary. The forest fires data will be clustered three times, to compare the effect of treating it as all linear, respecting the cyclic attributes using real directions only, and respecting the cyclic attributes using the real and imaginary directions. Since the area burned is very skewed, take the logarithm of it. Scale the cyclic variables to unit period, and the continuous variables to unit range. Leave the weights equal. Comparing the results against each other:

```
Comparing the linear against pseudo-Euclidean:
Adjusted Rand Index: 0.888586
Variation of Information: 0.274069
Comparing the real embedding against pseudo-Euclidean:
Adjusted Rand Index: 1.000000
Variation of Information: 0.000000
```

So the effect of the cyclic variables was quite small. Using the imaginary directions had no effect whatsoever on the outcome. Considering the pseudo-Euclidean clustering, there seem to be two clusters:





Cyclic data and imaginary coordinates may not be especially useful, but they are certainly interesting.

Notes

1. For an introduction to k-means, please consult Wikipedia, or the earlier article "Beginning Data Clustering" on Codeproject, which gives a simple implementation.
2. A univariate Laplace distribution does have the L1 geometry. What is called a multivariate Laplace distribution [Kotz, Kozubowski & Podgórski, 2001] is elliptically contoured. In one

dimension the Euclidean and Manhattan metrics coincide anyway, so there may not be a good theoretical justification for the Manhattan metric.

3. There are transformations from similarity to dissimilarity which are quicker to compute, but the arccosine is the arc length between two points on the unit hypersphere. This is a metric dissimilarity, which is important for some applications.
4. It is necessary to scale continuous and ordinal variables 0...1 when dichotomous values are in use for the dissimilarities to satisfy the triangle inequality. [Gower, 1971]
5. For instance, Wikipedia said as of 26 August 2017, "The standard algorithm aims at minimizing the WCSS objective, and thus assigns by 'least sum of squares', which is exactly equivalent to assigning by the smallest Euclidean distance."
6. If all the data is continuous, then the importance of each variable can be represented by multiplying the data by a scalar. Since this cannot be done with categorical variables, it is necessary to have a separate array to show the importance of each variable.
7. It has been suggested to achieve dummy coding with *c* - 1 variables by omitting the last continuous dummy variable. [Bowles, 2015] This introduces a bias, since the last category is less different from the other categories than the other categories are from each other.
8. The numerical difficulty is due to a logical inconsistency. If learning the presence of something is informative, then learning the absence of it must also be informative. (See Appendix) Of course, this does not mean that the informational content is equal, and dichotomous variables have been advocated by the foremost authorities in the field. If information content is to be taken into account, it ought to be done in a principled way and inferred from the data and not imposed by the analyst by his prior assumptions.
9. Graph clustering algorithms can handle null objects by assigning them an arbitrary dissimilarity.
10. It is possible to cluster cyclic variables, but it would make the whole package more complicated, so support was deleted.

Apology

Everybody makes mistakes, but not everybody admits to them. Before revision 3, this package contained a version of PCA that claimed to handle missing data, because I thought that a simple linear extrapolation could be made. The problem is, the components of the eigenvectors may be either positive or negative, and so there is no way to make an allowance for omitting some of them. I have replaced this invalid treatment with MDS to handle missing data correctly. The replacement PCA does not accept missing data. The non-Euclidean covariances in the old version have been removed.

The perfect is the enemy of the good, and so I am pushing out this software with some known shortcomings.

- The routines in prep.f and prep.c search for duplicates by sorting and comparing adjacent elements. To be really efficient, this should be done with hash tables.
- The binary tree could be made more efficient if one of the pivots at each level was the pivot at its ancestor level.

- The ancient EISPACK routines are acceptable only for small problems. To apply MDS to data sets of more than a few thousand objects, the dissimilarity matrix must be approximated. Then the Lanczos algorithm should be used to find the eigenvectors.
- HCLUST splits the cluster with the largest population, not the largest diameter.
- Using average dissimilarities to deal with missing data is a possibility left unexplored.

Revision History

3. 24 December 2020 Invalid missing-data PCA replaced by multi-dimensional scaling.
Maximum distance criterion in binary tree search.
Replacement for safe-to-divide function `SAFDIV`.
Remove compensated addition.
Forest Fire problem replaces Maple Leaf Rag problem.

2. 8 November 2017 Fixed mistake in USORTU that returned an invalid index when N equals 1.

1. 7 October 2017 Changed loop in KCLUST for deleting clusters to negative step. This should fix mistakes that could occur when more that one cluster is to be deleted.

0. 29 August 2017 Initial release.

What's Next

The path straight ahead in data clustering leads to Expectation Maximization. The notion of dissimilarity is given a precise meaning as conditional probability. A helpful case study was made by [Miller et al., 1994]. Free programs are EMMIX by McLachlan & Peel, mclust by Fraley et. al., and AUTOCLASS.

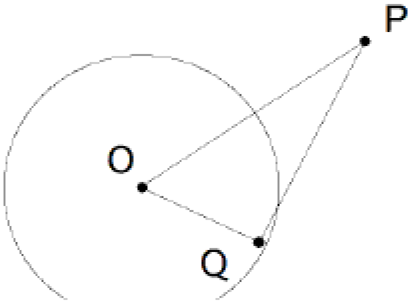
Appendix

Dichotomous variables are illogical: (reduction to an absurdity)
A: something observed
B: something inferred
Given: The presence of A is informative of B, and the absence of A is uninformative of B.
$\Pr(B|A) \neq \Pr(B)$
$\Pr(B|\neg A) = \Pr(B)$

Thus, $\Pr(B|\neg A) = \Pr(B) = \Pr(B \wedge A) + \Pr(B \wedge \neg A)$
$\Pr(B|\neg A) = \Pr(B|A) \Pr(A) + \Pr(B|\neg A) (1 - \Pr(A))$
$\Pr(B|\neg A) = \Pr(B|A)$
$\therefore \Pr(B|A) = \Pr(B)$, which contradicts the assumption.

Half-distance Lemma: [Elkan, 2003] If the distance to a point from its nearest center is less than or equal to half the distance of that center to any other center, the point can belong in no other cluster.

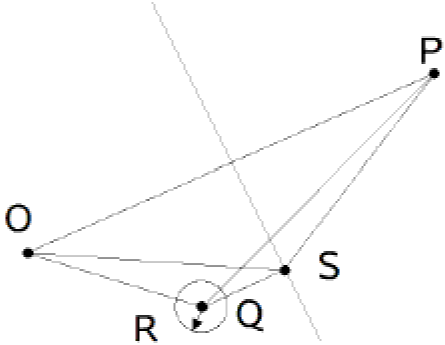
Proof: Let Q be the point. Let O be its current center. Let P be the nearest center to O.



Given:
$|OQ|$ ≤ (½) $|OP|$
$2|OQ|$ ≤ $|OP|$
By the triangle inequality,
$|OP|$ ≤ $|OQ|+|PQ|$
$2|OQ|$ ≤ $|OP|$ ≤ $|OQ|+|PQ|$
$|OQ|$ ≤ $|PQ|$

Generalized Hyperplane tree formula: [Chávez, 1999] QS ≥ ½ |PQ - OQ|

Proof: The pivots are O and P, and a query point is Q. S is hypothetical point on the boundary surface. The distance |QS| must be bounded to compare it to r, the radius of the ball containing the known neighbors of Q.



Construct triangle PQS, then by the triangle inequality:

$$|QS| \geq |PQ| - |PS|$$

Construct triangle OQS, then by the triangle inequality:

$$|QS| \geq |OS| - |OQ|$$

Add

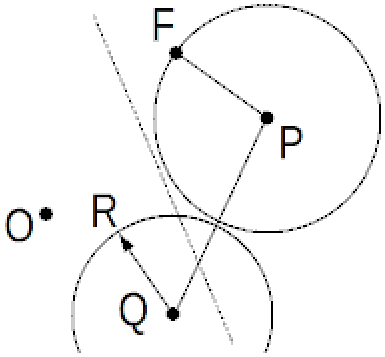
$$2|QS| \geq (|PQ| - |OQ|) + (|OS| - |PS|)$$

|OS| = |PS| by definition, so

$$|QS| \geq (½)\ (|PQ| - |OQ|)$$

Bisector tree formula: [Kalantari, 1983] R < PQ - PF

Pivots are O and P, and a query point is Q. F is the point in the far partition farthest from pivot P. R is the search radius around Q.



If the distance

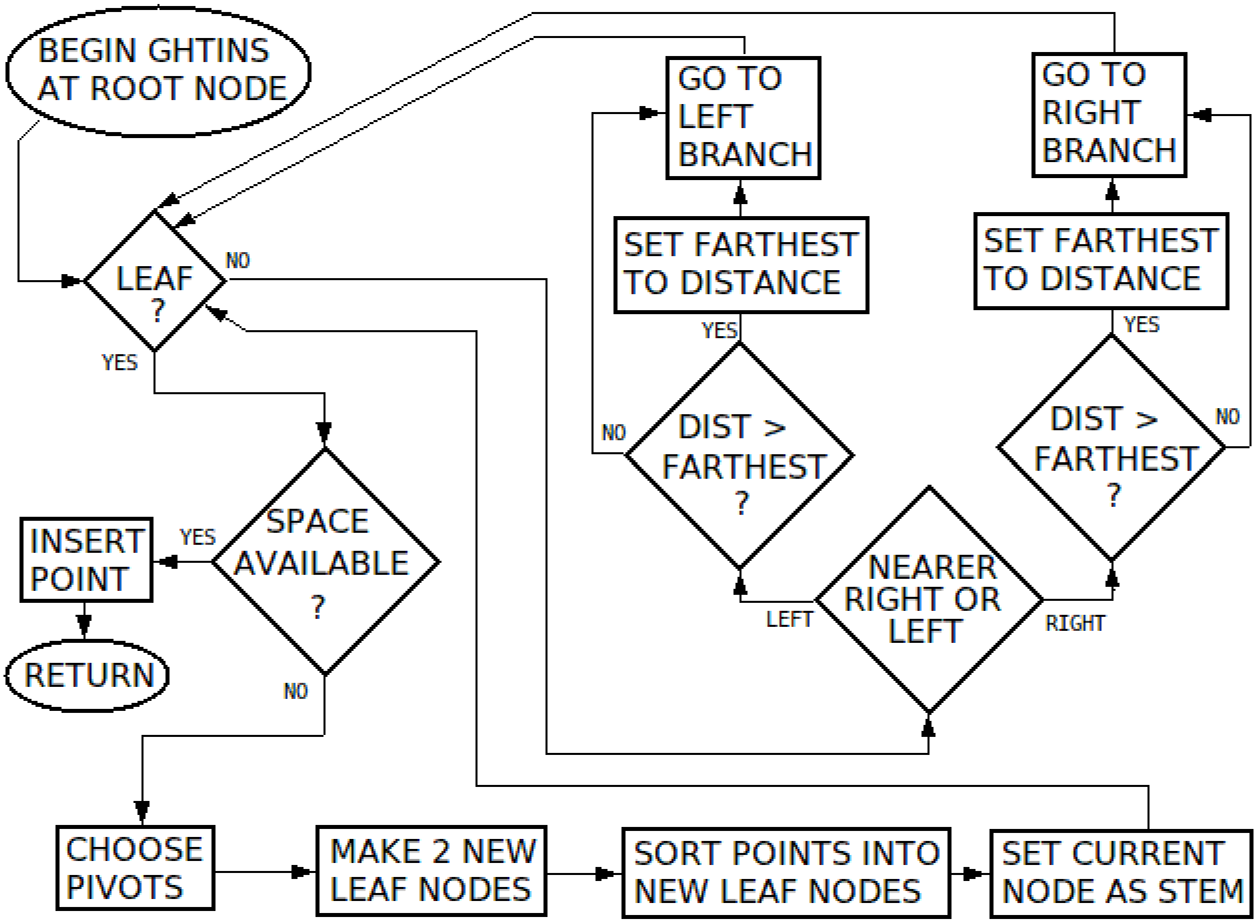
$$|PQ| > |PF| + R$$

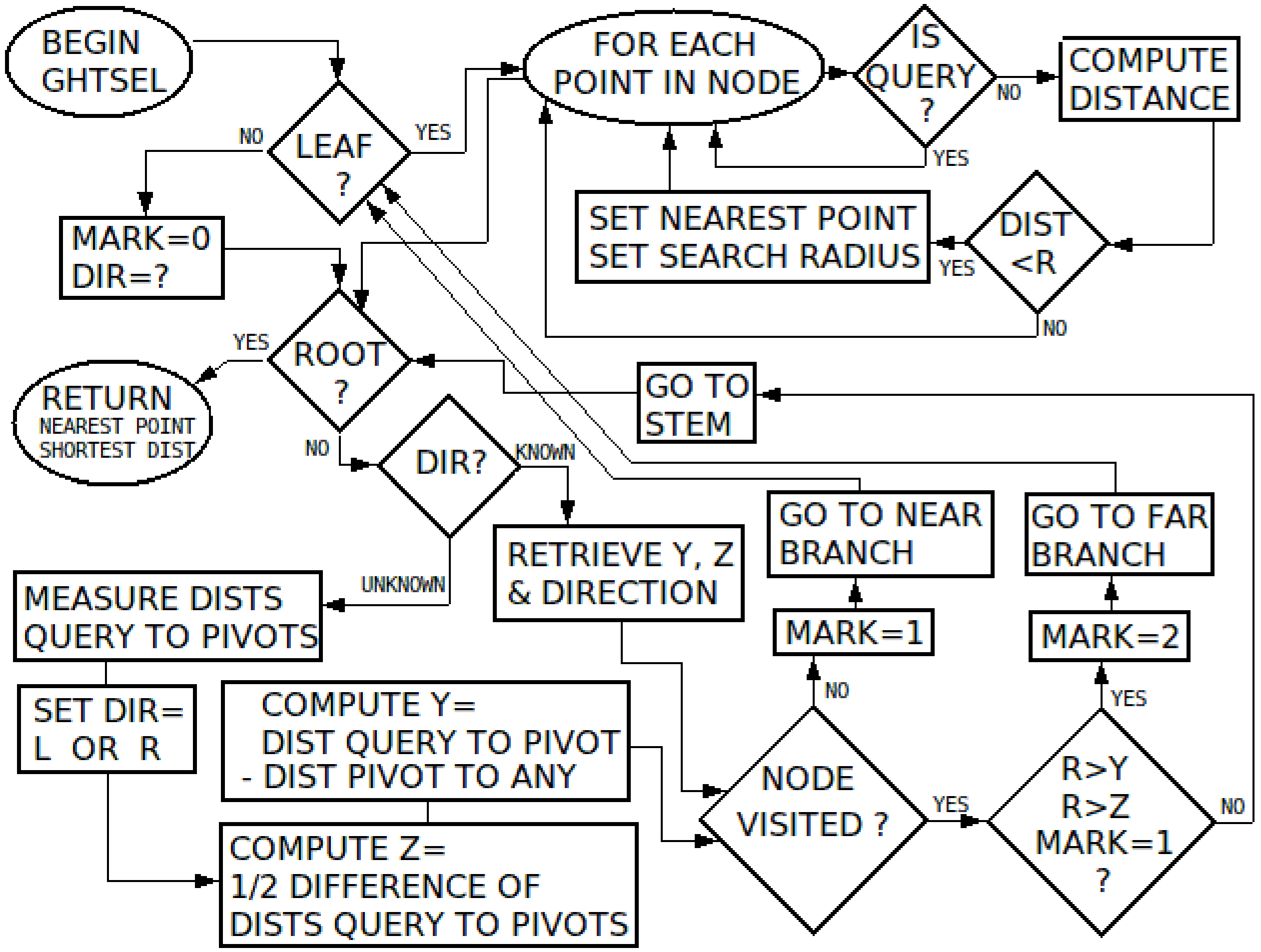
then no point in the far partition can be within the search radius. Therefore, if

$$R < |PQ| - |PF|$$

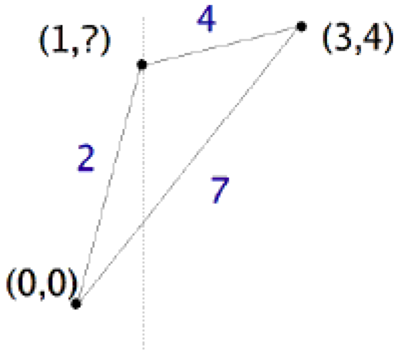
the far branch does not need to be explored.

Generalized hyperplane trees are not the subject of this article, but they deserve to be better known. Here are flowcharts for the INSERT and SELECT operations. Since an object may be a pivot, the DELETE operation cannot be performed.


BEGIN GHTINS AT ROOT NODE
LEAF ?
NO
YES
GO TO LEFT BRANCH
GO TO RIGHT BRANCH
SET FARTHEST TO DISTANCE
SET FARTHEST TO DISTANCE
YES
YES
NO
DIST > FARTHEST ?
DIST > FARTHEST ?
NO
NEARER RIGHT OR LEFT
LEFT
RIGHT
INSERT POINT
YES
SPACE AVAILABLE ?
RETURN
NO
CHOOSE PIVOTS
MAKE 2 NEW LEAF NODES
SORT POINTS INTO NEW LEAF NODES
SET CURRENT NODE AS STEM


BEGIN GHTSEL
FOR EACH POINT IN NODE
IS QUERY ?
NO
COMPUTE DISTANCE
YES
NO
LEAF ?
YES
MARK=0 DIR=?
SET NEAREST POINT SET SEARCH RADIUS
YES
DIST <R
NO
YES
ROOT ?
RETURN NEAREST POINT SHORTEST DIST
GO TO STEM
NO
DIR?
KNOWN
GO TO NEAR BRANCH
GO TO FAR BRANCH
UNKNOWN
RETRIEVE Y, Z & DIRECTION
MEASURE DISTS QUERY TO PIVOTS
MARK=1
MARK=2
NO
YES
SET DIR= L OR R
COMPUTE Y= DIST QUERY TO PIVOT - DIST PIVOT TO ANY
NODE VISITED ?
YES
R>Y R>Z MARK=1 ?
NO
COMPUTE Z= 1/2 DIFFERENCE OF DISTS QUERY TO PIVOTS

Missing data is Non-Metric



Proof by counterexample: Consider the Manhattan dissimilarity function for missing data:

$$d(X,Y)=(\sum Wtotal\ /\ \sum Wavail)\ \sum|Xi-Yi|$$

Consider the points O = (0,0), P = (3,4), and Q = (1,?), and let the weights be equal. The dissimilarities are:
d(O,P) = (2/2) (|3-0| + |4-0|) = 7
d(O,Q) = (2/1) (|1-0| + ∅) = 2
d(P,Q) = (2/1) (|3-1| + ∅) = 4
2 + 4 < 7 and the triangle inequality is violated.

Decrease and Increase of Sum of Squared Distance: [Hartigan, 1979] Removing a point X from a cluster C of N points reduces the sum of squared distances by

$$d^2(X,C)\ N/(N-1)$$

Placing a point in a cluster increases the sum of squared distances by

$$d^2(X,C)\ N\ /\ (N+1)$$

Proof for removing a point: Let X' denote the point transferred. Let C' denote the new position of the cluster center. Let X_i, 1 ≤ i ≤ N be the initial points in the cluster. Then the change in squared distance U is:

$$\Delta U_{del}=\sum(X_i-C')^2-\sum(X_i-C)^2-(X'-C')^2$$

Expand:

$$-2(\sum X_i)(C'-C)+\sum(C'^2-C^2)-(X'-C')^2$$

Doing some algebra shows that:

$$\sum X_i = NC$$
$$C'-C = (-X'-C)\ /\ (N-1)$$
$$C'^2-C^2 = (X'+C-2NC)(X'-C)/(N-1)^2$$
$$X'-C' = (X'-C)\ N\ /\ (N-1)$$

Substitute:

$$\Delta U_{del}=-2NC-(X'-C)/(N-1)+N(X'+C-2NC)(X'-C)/(N-1)^2-(N/(N-1))^2(X'-C)^2$$

Simplify to:

$$\Delta U_{del}=-(X'-C)^2\,N\ /\ (N-1)$$

Derivation of the formula for inserting a point is left to the interested reader.

The Sum-Of-Squares Dissimilarity is Non-Metric

Proof by counterexample: Suppose that there is a data point X, and two identical centers C_1 and C_2 Suppose that X is an element of C_1 and not of C_2. Then

$d(X,C_1) > d(X,C_2)$

but $d(C_1,C_2) = 0$
$d(X,C_1) > d(X,C_2) + d(C_1,C_2)$
and the triangle inequality is violated.

Regular Simplex Coordinates

0.000	1.000	0.500	0.500	0.500
0.000	0.000	0.866	0.289	0.289
0.000	0.000	0.000	0.816	0.204
0.000	0.000	0.000	0.000	0.791

Regular Simplex Coordinates in L1 Geometry

0.000	1.000	0.500	0.500	0.500
0.000	0.000	0.500	0.000	0.000
0.000	0.000	0.000	0.500	0.000
0.000	0.000	0.000	0.000	0.500

References

1. Robert L. Thorndike, "Who Belongs in the Family?" *Psychometrika*, v.18 n.4, December 1953
2. Geoffrey H. Ball and David J. Hall, *Isodata, a Novel Method of Data Analysis and Pattern Classification* Stanford Research Institute Technical Report, Project 5533, April 1965. at DTIC
3. A. J. B. Anderson, "Similarity Measure for Mixed Attribute Types", *Nature*, v.232 pp.416-417, 6 Aug 1971.
4. John C. Gower, "A General Coefficient of Similarity and Some of Its Properties" *Biometrics*, v.28 n.4 pp.857-871, Dec. 1971. at JSTOR
5. John A. Hartigan and Manchek A. Wong, "Algorithm AS 136: A K-Means Clustering Algorithm" *Applied Statistics* v.28 n.1 pp.100-108, 1979. at JSTOR
6. R.S. Michalski and R.L. Chilausky, "Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis", International Journal of Policy Analysis and Information Systems, Vol. 4, No. 2, 1980.
7. I. Kalantari and G. McDonald, "A data structure and an algorithm for the nearest point problem" IEEE Transactions on Software Engineering, v.9 p.5, 1983
8. Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, Volume XL: Congressional Quarterly Inc. Washington, D.C., 1985.

9. Glenn W. Milligan and Martha C. Cooper, "An Examination of Procedures for Determining the Number of Clusters in a Data Set" *Psychometrika* v.50 n.2 pp.159-179, June 1985.
10. M. Lichman (Admin.), UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science, 1987-present
11. Ross Quinlan, "Simplifying decision trees" *Int J Man-Machine Studies* v.27, pp. 221-234, December 1987
12. Mary McLeish & Matt Cecile, "Horse Colic Data Set", Department of Computer Science, University of Guelph, Guelph, Ontario, Canada N1G 2W1, [c.1989]
13. Leonard Kaufman and Peter J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley, 1990, reissued 2005.
14. M. Forina, et al, PARVUS - An Extendable Package for Data Exploration, Classification and Correlation. Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy.[c.1991]
15. Jeffrey K. Uhlmann, "Satisfying General Proximity/Similarity Queries with Metric Trees" *Information Processing Letters*, v.40 pp.175-179, November 1991
16. Thomas Martinetz and Klaus Schulten, 'A "Neural-Gas" Network Learns Topologies' in *Artificial Neural Networks*, Ed. T. Kohonen, K. Makisara, O. Simula, and J. Kangas. Holland: Elsevier, 1991. at researchgate
17. J.W. Miller, W.A. Woodward, H.L. Gray & G.D. McCartor "A Hypothesis-Testing Approach to Discriminant Analysis with Mixed Categorical and Continuous Variables When Data Are Missing" Phillips Laboratory, [U.S.] Air Force Materiel Command, July 1994. at DTIC
18. Michael Falk, "On MAD and Comedians" *Ann. Inst. Statist. Math.* v.49 n.4 pp.615-644, 1997.
19. Zhexue Huang, "Clustering Large Data Sets With Mixed Numeric and Categorical Values" Canberra: CSIRO Mathematical and Information Sciences, 1997.
20. Maria Laura Carranza, Enrico Feoli & Paola Ganis, "Analysis of vegetation structural diversity by Burnaby's similarity index," *Plant Ecology*, v.138 pp.77-87, 1998. at researchgate:
21. Geoff McLachlan & David Peel "The EMMIX Algorithm for the Fitting of Normal and t-Components" Journal of Statistical Software, v.4 i.2, 12 July 1999. online
22. Edgar Chávez, Gonzallo Navarro, and José Luis Marroquín, "Searching in Metric Spaces" Association for Computing Machinery, 1999
23. Dan Pelleg and Andrew Moore, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters." Draft report, 2000. at citeseerx
24. János Podani, *Introduction to the Exploration of Multivariate Biological Data* Leiden: Backhuys Publishers, 2000. online:
25. Samuel Kotz, Tomasz J. Kozubowski, and Krzysztof Podgórski, *The Laplace Distribution and Generalizations: A Revisit with New Applications.* Internet draft copy, 24 January 2001
26. Marina Meilă, "Comparing Clusterings" University of Washington, technical report, 2002. at citeseerx
27. Charles Elkan, "Using the Triangle Inequality to Accelerate k-Means" Proceedings of the Twentieth International Conference on Machine Learning, 2003. online
28. Francis R. Bach and Michael I. Jordan, "Learning Spectral Clustering" University of California, draft report, 2003. at École Normale Supérieure

29. Julian Laub, *Non-Metric Pairwise Proximity Data* Dipl. Ing. Phys. Thesis, Technical University of Berlin, 10 December 2004. at TU Berlin
30. D T Pham, S S Dimov, and C D Nguyen, "Selection of K in K-means Clustering" *J. of Mechanical Engineering Science*, 2005.
31. Shai Ben-David, Ulrike von Luxburg, and Dávid Pál, "A Sober Look at Clustering Stability", [c.2005] at citeseerx
32. Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy, "The Effectiveness of Lloyd-Type Methods for the k-means Problem", *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006. at University of Waterloo
33. Brendan McCane & Michael Albert, "Distance Functions for Categorical and Mixed Variables", Department of Computer Science, University of Otago, Dunedin, New Zealand, 14 Dec 2007. online Be warned that "symbolic covariance" proposed in this article is invalid.
34. P. Cortez and A. Morais, "A Data Mining Approach to Predict Forest Fires using Meteorological Data." In J. Neves, M. F. Santos and J. Machado Eds., New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence, December 2007, Guimaraes, Portugal, pp. 512-523. APPIA, ISBN-13 978-989-95618-0-9. available:
35. David Arthur and Sergei Vassilvitskii, "k-means++: the advantages of careful seeding", *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007. at Stanford
36. Sung-Hyuk Cha, "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions" *International Journal of Mathematical Models and Methods in Applied Sciences* v.1 i.4, 2007.
37. Shyam Boriah, Varun Chandola, and Vipin Kumar, "Similarity Measures for Categorical Data: A Comparative Evaluation" Proceedings of the SIAM International Conference on Data Mining, Atlanta, Georgia, USA, 24-26 April 2008. at researchgate
38. Hans-Hermann Bock, "Origins and extensions of the k-means algorithm in cluster analysis" Electronic Journ@l for History of Probability and Statistics v.4 n.2 December 2008 online
39. Michiel de Hoon, Seiya Imoto, and Satoru Miyano. Users' manual for: *The C Clustering Library for cDNA Microarray Data*, University of Tokyo, Institute of Medical Science, Human Genome Center, 27 December 2010. online
40. Michel Deza and Elena Deza, *Distances in PR, Audio and Image* Séminaire Brillouin, IRCAM, 31 May 2012. online
41. Brian Kulis and Michael I. Jordan, "Revisiting K-means: New Algorithms via Bayesian Nonparametrics" *Proceedings of the 29th International Conference on Machine Learning*, 2012. at researchgate
42. Malika Charrad, Nadia Ghazzali, Veronique Boiteau, and Adam Niknafs, "NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set" available from CRAN *Journal of Statistical Software* v.61 i.6 pp.1-36, 2014.
43. Michael Bowles, *Machine Learning in Python: Essential Techniques for Predictive Analysis* Wiley, 2015.