# Package ‘ppclust’

July 23, 2025

**Type** Package

**Title** Probabilistic and Possibilistic Cluster Analysis

**Version** 1.1.0.1

**Date** 2020-02-08

**Encoding** UTF-8

**Author** Zeynel Cebeci [aut, cre],
Figen Yildiz [aut],
Alper Tuna Kavlak [aut],
Cagatay Cebeci [aut],
Hasan Onder [aut]

**Maintainer** Zeynel Cebeci <zcebeci@cukurova.edu.tr>

**Description** Partitioning clustering divides the objects in a data set into non-overlapping subsets or clusters by using the prototype-based probabilistic and possibilistic clustering algorithms. This package covers a set of the functions for Fuzzy C-Means (Bezdek, 1974) <doi:10.1080/01969727308546047>, Possibilistic C-Means (Krishnapuram & Keller, 1993) <doi:10.1109/91.227387>, Possibilistic Fuzzy C-Means (Pal et al, 2005) <doi:10.1109/TFUZZ.2004.840099>, Possibilistic Clustering Algorithm (Yang et al, 2006) <doi:10.1016/j.patcog.2005.07.005>, Possibilistic C-Means with Repulsion (Wachs et al, 2006) <doi:10.1007/3-540-31662-0_6> and the other variants of hard and soft clustering algorithms. The cluster prototypes and membership matrices required by these partitioning algorithms are initialized with different initialization techniques that are available in the package 'inaparc'. As the distance metrics, not only the Euclidean distance but also a set of the commonly used distance metrics are available to use with some of the algorithms in the package.

**Depends** R (>= 3.3.0)

**License** GPL (>= 2)

**LazyData** true

**Imports** graphics, grDevices, inaparc, MASS, stats, utils, methods

**Suggests** cluster, factoextra, fclust, knitr, rmarkdown, vegclust

**VignetteBuilder** knitr, rmarkdown

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2023-12-13 17:27:57 UTC

# Contents

---

`ppclust-package` *Probabilistic and Possibilistic Cluster Analysis*

---

## Description

Partitioning clustering simply divides the objects in a data set into non-overlapping subsets or clusters by using the prototype-based probabilistic and possibilistic clustering algorithms. The package covers various functions for K-Means (MacQueen, 1967), Fuzzy C-Means (Bezdek, 1974), Possibilitic C-Means (Krishnapuram & Keller, 1993;1996), Possibilistic and Fuzzy C-Means (Pal et al, 2005), Possibilistic Clustering Algorithm (Yang et al, 2006), Possibilistic C-Means with Repulsion (Wachs et al, 2006), Unsupervised Possibilistic Fuzzy C-Means (Wu et al, 2010) and the other variant algorithms which produce hard, fuzzy and possibilistic partitions of numeric data sets. The cluster prototypes and membership matrices required by the partitioning algorithms can be initialized with many initialization techniques that are available in the package '**inaparc**'. As the distance metrics, not only the Euclidean distance but also a set of the commonly used distance metrics are available to use with some of the algorithms in the package.

## Details

The goal of prototype-based algorithms as the most widely-used group of partitioning clustering algorithms is to partition a data set of *n* objects with *p* features into *k*, a pre-defined number of clusters which are the non-hierchical subsets of data. On the clustering context, a prototype is a data item that represents or characterizes a cluster. Usually, a prototype can be regarded as the most central point in a data subspace (Tan et al. 2006).

Among the partitioning-based algorithms, the hard clustering algorithms, i.e. K-means, assume that each data point belongs to one cluster; however, in practice clusters may overlap and data points may belong to more than one cluster. In this case the membership degrees of a data point to clusters should be a value between zero and one. This idea has been modelled with the fuzzy clustering algorithms. The Fuzzy C-means proposed by Bezdek (FCM) (Bezdek, 1981), is the well-known partitioning based fuzzy algorithm. It assigns a fuzzy membership degree to each data point based on its distances to the cluster centers. If a data point is closer to a cluster center, its membership to this cluster be higher than its memberships to the other clusters.

As an extension of the basic FCM algorithm, Gustafson and Kessel (GK) clustering algorithm employs an adaptive distance norm in order to detect clusters with different geometrical shapes (Babuska, 2001; Correa et al, 2011). Gath and Geva (1989) revealed that the fuzzy maximum likelihood estimates (FMLE) clustering algorithm can be used to detect clusters of varying shapes, sizes and densities.

In the related literature it has been revealed that FCM is sensitive to noise and outliers in the data sets. Krishnapuram and Keller proposed and later improved the Possibilistic C-means (PCM) algorithm in order to avoid the effect of outliers or noises on the clustering performance (Krishnapuram & Keller, 1993;1996). Althouh PCM solves the problems due to noise and outliers by relaxing the probabilistic constraint of FCM, it has the disadvantage to generate coincident clusters with poor initializations. In order to overcome this problem, several variants of FCM and PCM have been proposed. Fuzzy Possibilistic C-means (FPCM) (Pal et al, 1997) is one of the mixed algorithms for simultaneously constructing memberships and typicalities. However FPCM has some problems because of the row sum constraint with the typicality values that produces unrealistic typicality values for large data sets. By adding a repulsion term forcing clusters far away from each other, an extension of PCM with repulsion has been introduced by Timm et al (2004). Pal et al (2005) proposed the Possibilistic Fuzzy C-means (PFCM) to overcome the noise sensitivity defect of FCM, to eliminate the coincident clusters problem of PCM and to eliminate the row sum constraints of FPCM.

Possibilistic Clustering Algorithm (PCA) proposed by Yang and Wu (2006) was in the front lines of another direction of algorithms improving FCM and PCM. The authors argued that the resulting membership of their proposed algorithm becomes an exponential function, so that it is robust to noise and outliers. Recently, Wu et al (2010) introduced the Unsupervised Possibilistic Fuzzy Clustering (UPFC) algorithm. UPFC is an extension of PCA by combining FCM and PCA algorithms in order to overcome the noise sensitivity problem of FCM and the coincident clusters problem of PCM. When compared to previous algorithms, UPFC seems promising algorithm for clustering in fuzzy and possibilistic environments since it needs not a probabilistic initialization.

## Basic Notations

Given a data set $\mathbf{X}$ describing $n$ data objects, the probabilistic and possibilistic clustering algorithms partition data into $k$, a predefined number of clusters through the minimization of their related objective functions with some probabilistic or possibilistic constraints.

$\mathbf{X} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\} \subseteq \Re^p$ is the data set for $n$ objects in the $p$-dimensional data space $\Re$, where:

- $n$ is the number of objects in the data set, $1 \leq n \leq \infty$
- $p$ is the number of features or variables which describes the data objects,
- $\vec{x}_i$ is $p$-length data vector for the i*th* data object.

On the clustering context, clusters are mostly represented by their prototypes. The prototypes are generally the centers of clusters which can be either centroids or medoids. The probabilistic and possibilistic partitioning clustering algorithms start with initialization of a cluster prototype matrix $\mathbf{V}$, and updates it through the iteration steps until it is stabilized.

$\mathbf{V} = \{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_k\} \subseteq \Re^n$ is the protoype matrix of the clusters, where:

- $k$ is the number of clusters, $1 \leq k \leq n$
- $\vec{v}_j$ is the $p$-length prototype vector for the j*th* cluster

The clustering algorithms compute the membership degrees of data objects by using some distance metrics for calculation of their proximities to the cluster centers.

$d(\vec{x}_i, \vec{v}_j)$ is the distance measure between the data object $\vec{x}_i$ and cluster prototype $\vec{v}_j$. In general, the squared Euclidean distance metric are used in most of the applications:

$d_{sq.euclidean}(\vec{x}_i, \vec{v}_j) = d^2(\vec{x}_i, \vec{v}_j) = \| \vec{x}_i - \vec{v}_j \|^2 = \ (\vec{x}_i - \vec{v}_j)^T \cdot (\vec{x}_i - \vec{v}_j)$

The clustering algorithms usually are run with the standard and squared Euclidean distance norms, which induce hyperspherical clusters. Therefore they are able find the clusters with the same shape and orientation because the norm inducing matrix is an identity matrix $\mathbf{A} = \mathbf{I}$. On the other hand, the distance metrics can be employed with a $n \times n$ diagonal norm inducing matrix $\mathbf{A} = \mathbf{I}\ 1/\sigma_j^2$ which modifies the distances depending on the direction in which the distance is measured (Timm et al, 2004; Balasko et al 2005). In this case, the squared Euclidean distance with the norm matrix $\mathbf{A}$ becomes:

$d_{sq.euclidean}(\vec{x}_i, \vec{v}_j) = d_A^2(\vec{x}_i, \vec{v}_j) = \| \vec{x}_i - \vec{v}_j \|_A^2 = \ (\vec{x}_i - \vec{v}_j)^T \mathbf{A} (\vec{x}_i - \vec{v}_j)$

$\mathbf{A}$ can be also formed as the inverse of the $n \times n$ covariance matrix $\mathbf{F}$:

$\mathbf{A} = \mathbf{F}^{-1}$, where:

$\mathbf{F} = \frac{1}{n} \sum\limits_{i=1}^{n} (\vec{x}_i - \bar{x})^T (\vec{x}_i - \bar{x})$.

Where $\bar{x}$ stands for the sample mean of the data. When the distance is induced with the norm matrix $\mathbf{A}$ the Mahalanobis norm on $\Re^n$ can be written as follows:

$d_{mahalanobis}(\vec{x}_i, \vec{v}_j)_A = \ (\vec{x}_i - \vec{v}_j)^T \mathbf{A}_j (\vec{x}_i - \vec{v}_j)$

The membership degrees are the measures specifying the amount of belongingness of the data objects to the different clusters. A data point nearer to the center of a cluster has a higher degree of membership to this cluster.

$\mathbf{U} = [u_{ij}]$ is the matrix for an hard, fuzzy or possibilistic partition of $\mathbf{X}$, where:

- $u_{ij} = u_i(\vec{x}_j)$ is the membership degree of $\vec{x}_i$ to the j*th* cluster

## Author(s)

Zeynel Cebeci, Figen Yildiz, A. Tuna Kavlak, Cagatay Cebeci & Hasan Onder

## References

Babuska, R. (2001). Fuzzy and neural control. DISC Course Lecture Notes. Delft University of Technology. Delft, the Netherlands. <https://tr.scribd.com/document/209211977/Fuzzy-and-Neural-Control>.

Balasko, B., Abonyi, J. & Feil, B. (2005). Fuzzy clustering and data analysis toolbox. Department of Process Eng., Univ. of Veszprem, Veszprem.

Bezdek, J.C. (1981). *Pattern recognition with fuzzy objective function algorithms*. Plenum, NY. <ISBN:0306406713>

Cebeci, Z. & Yildiz, F. (2015). Bulanik C-Ortalamalar algoritmasýnýn farklý kume buyuklukleri icin hesaplama performansi ve kumeleme gecerliliginin karsilastirilmasi, In Proc.pf *9. Ulusal Zootekni Bilim Kongresi*, Sep. 2015, Konya. pp. 227-239. doi:10.13140/RG.2.1.2909.9288

Cebeci, Z., Kavlak, A.T. & Yildiz, F. (2017). Validation of fuzzy and possibilistic clustering results, in Proc. of *2017 Int. Artificial Intelligence & Data Processing Symposium*, IEEE. pp. 1-7. doi:10.1109/IDAP.2017.8090183

Cebeci, Z. (2018). Initialization of membership degree matrix for fast convergence of Fuzzy C-Means clustering, in Proc. of *2018 Int.Conf.on Artificial Intelligence & Data Processing*, pp. 1-5. IEEE. doi:10.1109/IDAP.2018.8620920

Cebeci, Z. & Cebeci, C. (2018) kpeaks: an R package for quick selection of k for cluster analysis, in Proc. of *2018 Int. Conf. on Artificial Intelligence & Data Processing*, pp. 1-7, IEEE. doi:10.1109/IDAP.2018.8620896

Cebeci, Z. (2019). Comparison of internal validity indices for fuzzy clustering. *Journal of Agricultural Informatics*, 10(2):1-14. doi:10.17700/jai.2019.10.2.537

Correa, C., Valero, C., Barreiro, P., Diago, M. P., & Tardáguila, J. (2011). A comparison of fuzzy clustering algorithms applied to feature extraction on vineyard. In *Proc. of the 14th Conf. of the Spanish Assoc. for Artificial Intelligence*. <`http://oa.upm.es/9246/`>.

Gath, I. & Geva, A.B. (1989). Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11 (7): 773-781. <doi:10.1109/34.192473>

Gustafson, D. E. & Kessel, W. C. (1979). Fuzzy clustering with a fuzzy covariance matrix. In *Proc. of IEEE Conf. on Decision and Control including the 17th Symposium on Adaptive Processes*, San Diego. pp. 761-766. <doi:10.1109/CDC.1978.268028>

Pal, N.R., Pal, K., & Bezdek, J.C. (1997). A mixed c-means clustering model. In *Proc. of the 6th IEEE Int. Conf. on Fuzzy Systems*, 1, pp. 11-21. <doi:10.1109/FUZZY.1997.616338>

Pal, N.R., Pal, S.K., Keller,J.M. & Bezdek, J.C. (2005). A possibilistic fuzzy c-means clustering algorithm. *IEEE Transactions on Fuzzy Systems*, 13 (4): 517-530. <doi: 10.1109/TFUZZ.2004.840099>

Tan, P. N., Steinbach, M., & Kumar, V. (2006). Cluster analysis: Basic concepts and algorithms. In *Introduction to Data Mining*. Pearson Addison Wesley. <`http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf`>

Timm, H., Borgelt, C., Doring, C. & Kruse, R. (2004). An extension to possibilistic fuzzy cluster analysis. *Fuzzy Sets and Systems*, 147 (1): 3-16. <doi:10.1016/j.fss.2003.11.009>

Yang, M. S. & Wu, K. L. (2006). Unsupervised possibilistic clustering. *Pattern Recognition*, 39(1): 5-21. <doi:10.1016/j.patcog.2005.07.005>

Wu, X., Wu, B., Sun, J. & Fu, H. (2010). Unsupervised possibilistic fuzzy clustering. *J. of Information & Computational Sci.*, 7 (5): 1075-1080.

### See Also

as.ppclust, comp.omega, fcm, fcm2, fpcm, fpppcm, gg, gk, gkpfcm, hcm, is.ppclust pca, ppclust2 pcm, pcmr, pfcm, summary.ppclust, upfc

---

## as.ppclust *Convert object to 'ppclust' class*

---

### Description

Converts an object of the classes fanny.object, summary.fclust, kmeans or vegclust to 'ppclust' class.

### Usage

```
as.ppclust(objx, ...)
```

### Arguments

| | |
|---|---|
| objx | an object to be converted to an instance of ppclust class. |
| ... | additional arguments. |

### Value

an object of ppclust class.

### Author(s)

Zeynel Cebeci

### See Also

is.ppclust, ppclust2, summary.ppclust

### Examples

```
data(iris)
# Create an fclust object
ofc <- fclust::FKM(X=iris[,1:4], k=3)

# Test the class of object 'ofc'
class(ofc)

# Convert 'ofc' to ppclust object
opc <- as.ppclust(ofc)

# Test the class of 'opc' object
class(opc)
```

## comp.omega — *Compute the possibilistic penalty argument for PCM*

### Description

Computes the vector $\vec{\Omega}$, the possibilistic penalty argument for Possibilistic C-Means clustering analysis.

### Usage

```
comp.omega(d, u, m=2, pco=NULL, K=1)
```

### Arguments

| | |
|---|---|
| `d` | a numeric matrix containing the distances of objects to the centers of clusters. |
| `u` | a numeric matrix containing the fuzzy or possibilistic membership degrees of the data objects. |
| `pco` | an object of 'ppclust' class that contains the results from a clustering algorithm. If it is supplied, there is no need to input the arguments `d`, `u` and `m` separately because they are parsed from the object `pco` itself. |
| `m` | a number for the fuzzy exponent. The default is 2. |
| `K` | a number greater than 0 to be used as the weight of penalty term. The default is 1. |

### Details

The vector $\vec{\Omega}$ is called possibilistic penalty term that controls the variance of clusters (Gosztolya & Szilagyi, 2015). In general, it is computed by using the fuzzy intra cluster distances from a previous run of FCM. This vector of mobilization scale parameters related with the spread of clusters contains the distance values at where membership degrees becomes 0.5 for each cluster (Timm et al, 2004; Wachs et al, 2006; Alamelumangai, 2014).

$$\vec{\Omega} = K \frac{\sum\limits_{i=1}^{n} u_{ij}^m \; d^2(\vec{x}_i, \vec{v}_j)}{\sum\limits_{i=1}^{n} u_{ij}^m} \; ; \; 1 \leq j \leq k$$

Where: $K$ is a coefficent, $K \in (0, \infty)$. It is generally chosen to be 1.

### Value

| | |
|---|---|
| `omega` | a numeric vector containing the mobilization scale values for each cluster. |

### Author(s)

Zeynel Cebeci

### References

Alamelumangai N. (2014). Computer aided segmentation of mammary carcinoma on ultrasound images using soft computing techniques. PhD Thesis, Anna Univ., IN. <http://hdl.handle.net/10603/50590>

Wachs, J., Shapira, O., & Stern, H. (2006). A method to enhance the ‘Possibilistic C-Means with Repulsion’ algorithm based on cluster validity index. In *Applied Soft Computing Technologies: The Challenge of Complexity*, pp. 77-87. Springer, Berlin, Heidelberg. <doi: 10.1007/3-540-31662-0_6>

Gosztolya, G. & Szilagyi, L. (2015). Application of fuzzy and possibilistic c-means clustering models in blind speaker clustering. *Acta Polytechnica Hungarica*, 12(7):41-56. <http://publicatio.bibl.u-szeged.hu/6151/1/2015-acta-polytechnica.pdf>

### See Also

pcm, pcmr, fpcm, gkpfcm, pfcm

### Examples

```
data(iris)
x <- iris[,-5]

# Run FCM
res.fcm <- fcm(x=x, centers=3)

# Compute the mobilization scale values using the results from FCM
vomg1 <- comp.omega(pco=res.fcm)
vomg1

# Compute the mobilization scale values using the distances and memberships from FCM
vomg2 <- comp.omega(res.fcm$d, res.fcm$u, m=3)
vomg2

# Compute the mobilization scale values with the K value of 10
vomg3 <- comp.omega(res.fcm$d, res.fcm$u, m=2, K=10)
vomg3
```

## crisp *Crisp the fuzzy membership degrees*

### Description

Crisps the fuzzy and possibilistic membership degrees from the fuzzy or possibilistic clustering algorithms.

### Usage

```
crisp(u, method, tv)
```

### Arguments

`u` a numeric matrix containing the fuzzy or possibilistic membership degrees of the data objects.

`method` a string for selection of the crisping method. The default is '`max`' that assigns the data object to the cluster in which the object has maximum membership. The alternative is '`threshold`' that assigns the objects to a cluster if its maximum membership degree is greater than `tv`, a threshold value.

`tv` a number for the threshold membership degree. The default is 0.5 with the `method` is '`threshold`' if it is not speficied by the user.

### Details

The function `crisp` produces the crisp or hard membership degrees of the objects in order to place them into only one cluster.

### Value

`cluster` a numeric vector containing the indexes (labels) of clusters for the maximum membership of the objects.

### Author(s)

Zeynel Cebeci

### Examples

```
data(iris)
x <- iris[,1:4]

# Run FCM
res.fcm <- fcm(x, centers=3)

# Crisp the fuzzy memberships degrees and plot the crisp memberships
cllabels <- crisp(res.fcm$u)
plot(x, col=cllabels)
```

---

## `ekm` *K-Means Clustering Using Different Seeding Techniques*

---

### Description

The function `ekm` partitions a numeric data set by using the K-means clustering algorithm. It is a wrapper function of the standard `kmeans` function with more initialization (seeding) techniques and output values obtained in the multiple starts of the algorithm.

## Usage

```
ekm(x, centers, dmetric="euclidean", alginitv="hartiganwong",
    nstart=1, iter.max=1000, stand=FALSE, numseed)
```

## Arguments

`x` a numeric vector, data frame or matrix.

`centers` an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers.

`dmetric` a string for the distance calculation method. The default is ‘`euclidean`’ for the Euclidean distances.

`alginitv` a string for the initialization of cluster prototypes matrix. The default is ‘`hartiganwong`’ for Hartigan-Wong seeding method. See `get.algorithms` for the alternative options.

`nstart` an integer for the number of starts for clustering. The default is 1.

`iter.max` an integer for the maximum number of iterations allowed. The default is 1000.

`stand` a logical flag to standardize data. Its default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized.

`numseed` a seeding number to set the seed of R’s random number generator.

## Details

K-Means (KM) clustering algorithm partitions a data set of $n$ objects into $k$, a pre-defined number of clusters. It is an iterative relocation algorithm, and in each iteration step the objects are assigned to the nearest cluster by using the Euclidean distances. The objective of KM is to minimize total intra-cluster variance (or the sum of squared errors):

$$J_{KM}(\mathbf{X};\mathbf{V}) = \sum\limits_{i=1}^{n} d^2(\vec{x}_i, \vec{v}_j)$$

In the above equation for $J_{KM}$:

$d^2(\vec{x}_i, \vec{v}_j)$ is the distance measure between the object $\vec{x}_j$ and cluster prototype $\vec{v}_i$.The Euclidean distance metric is usually employed with the implementations of K-means.

See `kmeans` and `ppclust-package` for more details about the terms of the objective function $J_{KM}$.

The update equation of the cluster prototypes:

$$\vec{v}_j = \frac{1}{n_j} \sum\limits_{i=1}^{n_j} x_{ij} \;;\; 1 \leq j \leq k$$

## Value

an object of class ‘ppclust’, which is a list consists of the following items:

`x` a numeric matrix containing the processed data set.

`v` a numeric matrix containing the final cluster prototypes (centers of clusters).

`u` a numeric matrix containing the crisp membership degrees of the data objects.

`d` a numeric matrix containing the distances of objects to the final cluster prototypes.

`k` an integer for the number of clusters.

`cluster` a numeric vector containing the cluster labels found by defuzzying the fuzzy membership degrees of the objects.

`csize` a numeric vector containing the number of objects in the clusters.

`iter` an integer vector for the number of iterations in each start of the algorithm.

`best.start` an integer for the index of start with the minimum objective functional.

`func.val` a numeric vector for the objective function values in each start of the algorithm.

`comp.time` a numeric vector for the execution time in each start of the algorithm.

`stand` a logical value, `TRUE` shows that `x` data set contains the standardized values of raw data.

`wss` a number for the within-cluster sum of squares for each cluster.

`bwss` a number for the between-cluster sum of squares.

`tss` a number for the total within-cluster sum of squares.

`twss` a number for the total sum of squares.

`algorithm` a string for the name of partitioning algorithm. It is ‘KM’ with this function.

`call` a string for the matched function call generating this ‘ppclust’ object.

## Author(s)

Zeynel Cebeci, Figen Yildiz & Hasan Onder

## References

MacQueen, J.B. (1967). Some methods for classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symp. on Mathematical Statistics and Probability*, Berkeley, University of California Press, 1: 281-297. <`http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.308.8619&rep=rep1&type=pdf`>

## See Also

`kmeans`, `fcm`, `fcm2`, `fpcm`, `fpppcm`, `gg`, `gk`, `gkpfcm`, `hcm`, `pca`, `pcm`, `pcmr`, `pfcm`, `upfc`

## Examples

```
# Load dataset iris
data(iris)
x <- iris[,-5]

# Run EKM for 3 clusters
res.ekm <- ekm(x, centers=3)

# Print and plot the clustering result
print(res.ekm$cluster)
plot(x, col=res.ekm$cluster, pch=16)
```

# fcm *Fuzzy C-Means Clustering*

## Description

Partitions a numeric data set by using the Fuzzy C-Means (FCM) clustering algorithm (Bezdek, 1974;1981).

## Usage

```
fcm(x, centers, memberships, m=2, dmetric="sqeuclidean", pw = 2,
    alginitv="kmpp", alginitu="imembrand",
    nstart=1, iter.max=1000, con.val=1e-09,
    fixcent=FALSE, fixmemb=FALSE, stand=FALSE, numseed)
```

## Arguments

`x` a numeric vector, data frame or matrix.

`centers` an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers.

`memberships` a numeric matrix containing the initial membership degrees. If missing, it is internally generated.

`m` a number greater than 1 to be used as the fuzziness exponent or fuzzifier. The default is 2.

`dmetric` a string for the distance metric. The default is ‘`sqeuclidean`’ for the squared Euclidean distances. See `get.dmetrics` for the alternative options.

`pw` a number for the power of Minkowski distance calculation. The default is 2 if the `dmetric` is ‘`minkowski`’.

`alginitv` a string for the initialization of cluster prototypes matrix. The default is ‘`kmpp`’ for K-means++ initialization method (Arthur & Vassilvitskii, 2007). For the list of alternative options see `get.algorithms`.

`alginitu` a string for the initialization of memberships degrees matrix. The default is ‘`imembrand`’ for random sampling of initial membership degrees.

`nstart` an integer for the number of starts for clustering. The default is 1.

`iter.max` an integer for the maximum number of iterations allowed. The default is 1000.

`con.val` a number for the convergence value between the iterations. The default is 1e-09.

`fixcent` a logical flag to make the initial cluster centers not changed along the different starts of the algorithm. The default is `FALSE`. If it is `TRUE`, the initial centers are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`fixmemb` a logical flag to make the initial membership degrees not changed along the different starts of the algorithm. The default is `FALSE`. If it is `TRUE`, the initial memberships are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`stand` a logical flag to standardize data. Its default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized.

`numseed` an optional seeding number to set the seed of R's random number generator.

### Details

Fuzzy C-Means (FCM) clustering algorithm was firstly studied by Dunn (1973) and generalized by Bezdek in 1974 (Bezdek, 1981). Unlike K-means algorithm, each data object is not the member of only one cluster but is the member of all clusters with varying degrees of memberhip between 0 and 1. It is an iterative clustering algorithm that partitions the data set into a predefined *k* partitions by minimizing the weighted within group sum of squared errors. The objective function of FCM is:

$$J_{FCM}(\mathbf{X};\mathbf{V},\mathbf{U}) = \sum\limits_{i=1}^{n} u_{ij}^m d^2(\vec{x}_i, \vec{v}_j)$$

In the objective function, $m$ is the fuzzifier to specify the amount of 'fuzziness' of the clustering result; $1 \leq m \leq \infty$. It is usually chosen as 2. The higher values of $m$ result with the more fuzzy clusters while the lower values give harder clusters. If it is 1, FCM becomes an hard algorithm and produces the same results with K-means.

FCM must satisfy the following constraints:

$$u_{ij} = [0,1] \;\; ; \; 1 \leq i \leq n \; , 1 \leq j \leq k$$

$$0 \leq \sum\limits_{i=1}^{n} u_{ij} \leq n \;\; ; \; 1 \leq j \leq k$$

$$\sum\limits_{j=1}^{k} u_{ij} = 1 \;\; ; \; 1 \leq i \leq n$$

The objective function of FCM is minimized by using the following update equations:

$$u_{ij} = \left[ \sum\limits_{j=1}^{k} \left( \frac{d^2(\vec{x}_i, \vec{v}_j)}{d^2(\vec{x}_i, \vec{v}_l)} \right)^{1/(m-1)} \right]^{-1} \;\; ; 1 \leq i \leq n, \; 1 \leq l \leq k$$

$$\vec{v}_j = \frac{\sum\limits_{i=1}^{n} u_{ij}^m \vec{x}_i}{\sum\limits_{i=1}^{n} u_{ij}^m} \;\; ; 1 \leq j \leq k$$

### Value

an object of class 'ppclust', which is a list consists of the following items:

`x` a numeric matrix containing the processed data set.

`v` a numeric matrix containing the final cluster prototypes (centers of clusters).

`u` a numeric matrix containing the fuzzy memberships degrees of the data objects.

`d` a numeric matrix containing the distances of objects to the final cluster prototypes.

`k` an integer for the number of clusters.

`m` a number for the fuzzifier.

`cluster` a numeric vector containing the cluster labels found by defuzzying the fuzzy membership degrees of the objects.

`csize` a numeric vector containing the number of objects in the clusters.

`iter` an integer vector for the number of iterations in each start of the algorithm.

`best.start` an integer for the index of start that produced the minimum objective functional.

`func.val` a numeric vector for the objective function values in each start of the algorithm.

`comp.time` a numeric vector for the execution time in each start of the algorithm.

`stand` a logical value, `TRUE` shows that data set `x` contains the standardized values of raw data.

`wss` a number for the within-cluster sum of squares for each cluster.

`bwss` a number for the between-cluster sum of squares.

`tss` a number for the total within-cluster sum of squares.

`twss` a number for the total sum of squares.

`algorithm` a string for the name of partitioning algorithm. It is 'FCM' with this function.

`call` a string for the matched function call generating this 'ppclust' object.

## Author(s)

Zeynel Cebeci, Figen Yildiz & Alper Tuna Kavlak

## References

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding, in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027-1035. <http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>

Dunn, J.C. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *J. Cybernetics*, 3(3):32-57. <doi:10.1080/01969727308546046>

Bezdek, J.C. (1974). Cluster validity with fuzzy sets. *J. Cybernetics*, 3: 58-73. <doi:10.1080/01969727308546047>

Bezdek J.C. (1981). *Pattern recognition with fuzzy objective function algorithms*. Plenum, NY. <ISBN:0306406713>

## See Also

`ekm`, `fcm2`, `fpcm`, `fpppcm`, `gg`, `gk`, `gkpfcm`, `hcm`, `pca`, `pcm`, `pcmr`, `pfcm`, `upfc`

## Examples

```
# Load dataset iris
data(iris)
x <- iris[,-5]

# Initialize the prototype matrix using K-means++ algorithm
v <- inaparc::kmpp(x, k=3)$v

# Initialize the memberships degrees matrix
u <- inaparc::imembrand(nrow(x), k=3)$u

# Run FCM with the initial prototypes and memberships
```

```
fcm.res <- fcm(x, centers=v, memberships=u, m=2)

# Show the fuzzy membership degrees for the top 5 objects
head(fcm.res$u, 5)
```

## fcm2 *Type-2 Fuzzy C-Means Clustering*

### Description

Partitions a numeric data set by using the Type-2 Fuzzy C-Means (FCM2) clustering algorithm (Rhee & Hwang, 2001). It has been reported that it is effective for spherical clusters in the data sets, but it fails when the data sets contain non-spherical and complex structures (Gosain & Dahiya, 2016).

### Usage

```
fcm2(x, centers, memberships, m=2, dmetric="sqeuclidean", pw = 2,
    alginitv="kmpp", alginitu="imembrand",
    nstart=1, iter.max=1000, con.val=1e-09,
    fixcent=FALSE, fixmemb=FALSE, stand=FALSE, numseed)
```

### Arguments

`x` a numeric vector, data frame or matrix.

`centers` an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers.

`memberships` a numeric matrix containing the initial membership degrees. If missing, it is internally generated.

`m` a number greater than 1 to be used as the fuzziness exponent or fuzzifier. The default is 2.

`dmetric` a string for the distance metric. The default is ‘`sqeuclidean`’ for the squared Euclidean distances. See `get.dmetrics` for the alternative options.

`pw` a number for the power of Minkowski distance calculation. The default is 2 if the `dmetric` is ‘`minkowski`’.

`alginitv` a string for the initialization of cluster prototypes matrix. The default is ‘`kmpp`’ for K-means++ initialization method (Arthur & Vassilvitskii, 2007). For the list of alternative options, see `get.algorithms`.

`alginitu` a string for the initialization of memberships degrees matrix. The default is ‘`imembrand`’ for random sampling of initial membership degrees.

`nstart` an integer for the number of starts for clustering. The default is 1.

`iter.max` an integer for the maximum number of iterations allowed. The default is 1000.

`con.val` a number for the convergence value between the iterations. The default is 1e-09.

`fixcent` a logical flag to make the initial cluster centers not changed along the different starts of the algorithm. The default is `FALSE`. If it is `TRUE`, the initial centers are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`fixmemb` a logical flag to make the initial membership degrees not changed along the different starts of the algorithm. The default is `FALSE`. If it is `TRUE`, the initial memberships are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`stand` a logical flag to standardize data. Its default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized.

`numseed` an optional seeding number to set the seed of R's random number generator.

## Details

In the Type-2 Fuzzy C-Means (T2FCM) clustering algorithm proposed by (Rhee & Hwang, 2001), the idea is that all data points should not have the same contribution in computing the cluster prototypes, instead the data points with higher membership value should contribute much more in the cluster prototypes (Gosain & Dahiya, 2016). Based on this idea, a type-2 membership value is computed by using the following equation:

$$a_{ij} = u_{ij}^m - \frac{1-u_{ij}^m}{2}$$

In the above equation, $u_{ij}$ and $a_{ij}$ respectively stand for the Type-1 (standard) and Type-2 membership values. $a_{ij}$ is only used to update the cluster prototypes.

The objective function of FCM2 and other terms are the same with those of FCM except the following update equation for the cluster prototypes:

$$\vec{v}_j = \frac{\sum\limits_{i=1}^n a_{ij}^m \vec{x}_i}{\sum\limits_{i=1}^n a_{ij}^m} \;\; ; 1 \leq j \leq k$$

## Value

an object of class 'ppclust', which is a list consists of the following items:

`x` a numeric matrix containing the processed data set.

`v` a numeric matrix containing the final cluster prototypes (centers of clusters).

`u` a numeric matrix containing the fuzzy memberships degrees of the data objects.

`d` a numeric matrix containing the distances of objects to the final cluster prototypes.

`k` an integer for the number of clusters.

`m` a number for the fuzzifier.

`cluster` a numeric vector containing the cluster labels found by defuzzying the fuzzy membership degrees of the objects.

`csize` a numeric vector containing the number of objects in the clusters.

`iter` an integer vector for the number of iterations in each start of the algorithm.

`best.start` an integer for the index of start that produced the minimum objective functional.

`func.val` a numeric vector for the objective function values in each start of the algorithm.

`comp.time` a numeric vector for the execution time in each start of the algorithm.

`stand` a logical value, `TRUE` shows that data set `x` contains the standardized values of raw data.

`wss` a number for the within-cluster sum of squares for each cluster.

`bwss` a number for the between-cluster sum of squares.

`tss` a number for the total within-cluster sum of squares.

`twss` a number for the total sum of squares.

`algorithm` a string for the name of partitioning algorithm. It is ‘FCM’ with this function.

`call` a string for the matched function call generating this ‘ppclust’ object.

## Author(s)

Zeynel Cebeci & Alper Tuna Kavlak

## References

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding, in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 1027-1035. <`http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf`>

Rhee, F.C.H. & Hwang, C. (2001). A type-2 fuzzy c-means clustering algorithm. In <*IEEE 9th IFSA World Congress and 20th NAFIPS Conf. 2001.*, 4:1926-1929. <doi:10.1109/NAFIPS.2001.944361>

Gosain, A. & Dahiya, S. (2016). Performance analysis of various fuzzy clustering algorithms: A review. *Procedia Comp. Sci.*, 79:100-111. <doi:10.1016/j.procs.2016.03.014>

## See Also

`ekm`, `fcm`, `fpcm`, `fpppcm`, `gg`, `gk`, `gkpfcm`, `hcm`, `pca`, `pcm`, `pcmr`, `pfcm`, `upfc`

## Examples

```
# Load dataset X12
data(x12)

# Initialize the prototype matrix using K-means++ algorithm
v <- inaparc::kmpp(x12, k=2)$v
# Initialize the membership degrees matrix
u <- inaparc::imembrand(nrow(x12), k=2)$u

# Run FCM2 with the initial prototypes and memberships
fcm2.res <- fcm2(x12, centers=v, memberships=u, m=2)

# Show the fuzzy membership degrees for the top 5 objects
head(fcm2.res$u, 5)
```

## fpcm *Fuzzy Possibilistic C-Means Clustering*

### Description

Partitions a numeric data set by using the Fuzzy and Possibilistic C-Means (FPCM) clustering algorithm (Pal et al, 1997).

### Usage

```
fpcm(x, centers, memberships, m=2, eta=2,
     dmetric=”sqeuclidean”, pw=2, alginitv=”kmpp”, alginitu=”imembrand”,
     nstart=1, iter.max=1000, con.val=1e-09,
     fixcent=FALSE, fixmemb=FALSE, stand=FALSE, numseed)
```

### Arguments

`x` a numeric vector, data frame or matrix.

`centers` an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers.

`memberships` a numeric matrix containing the initial membership degrees. If missing, it is internally generated.

`m` a number greater than 1 to be used as the fuzziness exponent or fuzzifier. The default is 2.

`eta` a number greater than 1 to be used as the typicality exponent. The default is 3.

`dmetric` a string for the distance metric. The default is ‘`sqeuclidean`’ for the squared Euclidean distances. See `get.dmetrics` for the alternative options.

`pw` a number for the power of Minkowski distance calculation. The default is 2 if the `dmetric` is ‘`minkowski`’.

`alginitv` a string for the initialization of cluster prototypes matrix. The default is ‘`kmpp`’ for K-means++ initialization method (Arthur & Vassilvitskii, 2007). For the list of alternative options see `get.algorithms`.

`alginitu` a string for the initialization of memberships degrees matrix. The default is ‘`imembrand`’ for random sampling of initial membership degrees.

`nstart` an integer for the number of starts for clustering. The default is 1.

`iter.max` an integer for the maximum number of iterations allowed. The default is 1000.

`con.val` a number for the convergence value between the iterations. The default is 1e-09.

`fixcent` a logical flag to make the initial cluster centers not changed along the different starts of the algorithm. The default is `FALSE`. If it is `TRUE`, the initial centers are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`fixmemb` a logical flag to make the initial membership degrees not changed along the different starts of the algorithm. The default is `FALSE`. If it is `TRUE`, the initial memberships are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`stand` a logical flag to standardize data. Its default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized.

`numseed` a seeding number to set the seed of R's random number generator.

## Details

Fuzzy and Possibilistic C Means (FPCM) algorithm which has been proposed by Pal et al (1997) indended to combine the characteristics of FCM and PCM, and hence, was also so-called *Mixed C-Means* (MCM) algorithm.

The objective function of FPCM is:

$$J_{FPCM}(\mathbf{X}; \mathbf{V}, \mathbf{U}, \mathbf{T}) = \sum\limits_{i=1}^n (u_{ij}^m + t_{ij}^\eta) \; d^2(\vec{x}_i, \vec{v}_j)$$

In the above equation:

$\mathbf{X} = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\} \subseteq \Re^p$ is the data set for $n$ objects in the *p*-dimensional data space $\Re$,

$\mathbf{V} = \{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_k\} \subseteq \Re^n$ is the protoype matrix of the clusters,

$\mathbf{U} = \{u_{ij}\}$ is the matrix for a fuzzy partition of $\mathbf{X}$,

$\mathbf{T} = \{t_{ij}\}$ is the matrix for a possibilistic partition of $\mathbf{X}$,

$d^2(\vec{x}_i, \vec{v}_j)$ is the squared Euclidean distance between the object $\vec{x}_j$ and cluster prototype $\vec{v}_i$.

$d^2(\vec{x}_i, \vec{v}_j) = ||\vec{x}_i - \vec{v}_j||^2 = (\vec{x}_i - \vec{v}_j)^T (\vec{x}_i - \vec{v}_j)$

$m$ is the fuzzifier to specify the amount of fuzziness for the clustering; $1 \leq m \leq \infty$. It is usually chosen as 2.

$\eta$ is the typicality exponent to specify the amount of typicality for the clustering; $1 \leq \eta \leq \infty$. It is usually chosen as 2.

FPCM must satisfy the following constraints:

$$\sum\limits_{j=1}^k u_{ij} = 1 \;\; ; \;\; 1 \leq i \leq n$$

$$\sum\limits_{i=1}^n t_{ij} = 1 \;\; ; \;\; 1 \leq j \leq k$$

The objective function of FPCM is minimized by using the following update equations:

$$u_{ij} = \left[ \sum\limits_{j=1}^k \left( \frac{d^2(\vec{x}_i, \vec{v}_j)}{d^2(\vec{x}_i, \vec{v}_l)} \right)^{1/(m-1)} \right]^{-1} \;\; ; 1 \leq i \leq n, \; 1 \leq l \leq k$$

$$t_{ij} = \left[ \sum\limits_{l=1}^n \left( \frac{d^2(\vec{x}_i, \vec{v}_j)}{d^2(\vec{x}_i, \vec{v}_l)} \right)^{1/(\eta-1)} \right]^{-1} \;\; ; 1 \leq i \leq n, \; 1 \leq j \leq k$$

$$\vec{v}_j = \frac{\sum\limits_{i=1}^n (u_{ij}^m + t_{ij}^\eta) \vec{x}_i}{\sum\limits_{i=1}^n (u_{ij}^m + t_{ij}^\eta)} \;\; ; 1 \leq j \leq k$$

## Value

an object of class ‘ppclust’, which is a list consists of the following items:

x a numeric matrix containing the processed data set.

v a numeric matrix containing the final cluster prototypes (centers of clusters).

u a numeric matrix containing the fuzzy memberships degrees of the data objects.

d a numeric matrix containing the distances of objects to the final cluster prototypes.

k an integer for the number of clusters.

m a number for the fuzzifier.

eta a number for the typicality exponent.

cluster a numeric vector containing the cluster labels found by defuzzying the fuzzy membership degrees of the objects.

csize a numeric vector containing the number of objects in the clusters.

iter an integer vector for the number of iterations in each start of the algorithm.

best.start an integer for the index of start that produced the minimum objective functional.

func.val a numeric vector for the objective function values in each start of the algorithm.

comp.time a numeric vector for the execution time in each start of the algorithm.

stand a logical value, TRUE shows that data set x contains the standardized values of raw data.

wss a number for the within-cluster sum of squares for each cluster.

bwss a number for the between-cluster sum of squares.

tss a number for the total within-cluster sum of squares.

twss a number for the total sum of squares.

algorithm a string for the name of partitioning algorithm. It is ‘FCM’ with this function.

call a string for the matched function call generating this ‘ppclust’ object.

## Author(s)

Zeynel Cebeci, Alper Tuna Kavlak & Figen Yildiz

## References

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding, in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 1027-1035. <http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>

Pal, N.R., Pal, K., & Bezdek, J.C. (1997). A mixed c-means clustering model. In *Proc. of the 6th IEEE Int. Conf. on Fuzzy Systems*, 1, pp. 11-21. <doi:10.1109/FUZZY.1997.616338>

## See Also

ekm, fcm, fcm2, fpppcm, gg, gk, gkpfcm, hcm, pca, pcm, pcmr, pfcm, upfc

## Examples

```
# Load dataset iris
data(iris)
x <- iris[,-5]

# Initialize the prototype matrix using K-means++
v <- inaparc::kmpp(x, k=3)$v

# Initialize the memberships degrees matrix
u <- inaparc::imembrand(nrow(x), k=3)$u

# Run FPCM with the initial prototypes and memberships
fpcm.res <- fpcm(x, centers=v, memberships=u, m=2, eta=2)

# Show the fuzzy membership degrees for the top 5 objects
head(fpcm.res$u, 5)

# Show the possibilistic membership degrees for the top 5 objects
head(fpcm.res$t, 5)
```

# fpppcm *Fuzzy Possibilistic Product Partition C-Means Clustering*

## Description

Partitions a numeric data set by using the Fuzzy Possibilistic Product Partition C-Means (FPPPCM) clustering algorithm which has been proposed by Szilagyi & Szilagyi (2014).

## Usage

```
fpppcm(x, centers, memberships, m=2, eta=2, K=1, omega,
    dmetric="sqeuclidean", pw=2, alginitv="kmpp", alginitu="imembrand",
    nstart=1, iter.max=1000, con.val=1e-09,
    fixcent=FALSE, fixmemb=FALSE, stand=FALSE, numseed)
```

## Arguments

`x` a numeric vector, data frame or matrix.

`centers` an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers.

`memberships` a numeric matrix containing the initial membership degrees. If missing, it is internally generated.

`m` a number greater than 1 to be used as the fuzziness exponent. The default is 2.

`eta` a number greater than 1 to be used as the typicality exponent. The default is 2.

`K` a number greater than 0 to be used as the weight of penalty term. The default is 1.

`omega` a numeric vector of reference distances. If missing, it is internally generated.

`dmetric` a string for the distance metric. The default is '`sqeuclidean`' for the squared Euclidean distances. See `get.dmetrics` for the alternative options.

`pw` a number for the power of Minkowski distance calculation. The default is 2 if the `dmetric` is '`minkowski`'.

`alginitv` a string for the initialization of cluster prototypes matrix. The default is '`kmpp`' for K-means++ initialization method (Arthur & Vassilvitskii, 2007). For the list of alternative options see `get.algorithms`.

`alginitu` a string for the initialization of memberships degrees matrix. The default is '`imembrand`' for random sampling of initial membership degrees.

`nstart` an integer for the number of starts for clustering. The default is 1.

`iter.max` an integer for the maximum number of iterations allowed. The default is 1000.

`con.val` a number for the convergence value between the iterations. The default is 1e-09.

`fixcent` a logical flag to fix the initial cluster centers. The default is `FALSE`. If it is `TRUE`, the initial centers are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`fixmemb` a logical flag to fix the initial membership degrees. The default is `FALSE`. If it is `TRUE`, the initial memberships are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`stand` a logical flag to standardize data. Its default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized.

`numseed` a seeding number to set the seed of R's random number generator.

## Details

Fuzzy Possibilistic Product Partition C-Means (FPPPCM) clustering algorithm aimed to eliminate the effect of outliers in the other fuzzy and possibilistic clustering algorithms. The algorithm includes a probabilistic and a possibilistic term via multiplicative way instead of additive combination (Gosztolya & Szilagyi, 2015). The objective function of the algorithm as follows:

$$J_{FPPPCM}(\mathbf{X};\mathbf{V},\mathbf{U},\mathbf{T}) = \sum\limits_{j=1}^k \sum\limits_{i=1}^n u_{ij}^m \left[t_{ij}^\eta \; d^2(\vec{x}_i,\vec{v}_j) + \Omega_j(1-t_{ij})^\eta\right]$$

The fuzzy membership degrees in the probabilistic part of the objective function $J_{FPPPCM}$ is updated as follows:

$$u_{ij} = \frac{\left[t_{ij}^\eta \; d^2(\vec{x}_i,\vec{v}_j) + \Omega_j(1-t_{ij})^\eta\right]^{-1/(m-1)}}{\left[\sum\limits_{l=1}^k t_{il}^\eta \; d^2(\vec{x}_i,\vec{v}_l) + \Omega_l(1-t_{il})^\eta\right]^{-1/(m-1)}} \;;\; 1 \leq i \leq n,\; 1 \leq j \leq k$$

The typicality degrees in the possibilistic part of the objective function $J_{FPPPCM}$ is calculated as follows:

$$t_{ij} = \left[1 + \left(\frac{d^2(\vec{x}_i,\vec{v}_j)}{\Omega_j}\right)^{1/(\eta-1)}\right]^{-1} \;;\; 1 \leq i \leq n,\; 1 \leq j \leq k$$

$m$ is the fuzzifier to specify the amount of fuzziness for the clustering; $1 \leq m \leq \infty$. It is usually chosen as 2.

$\eta$ is the typicality exponent to specify the amount of typicality for the clustering; $1 \leq \eta \leq \infty$. It is usually chosen as 2.

$\Omega$ is the possibilistic penalty term to control the variance of the clusters.

The update equation for cluster prototypes:

$$\vec{v}_j = \frac{\sum\limits_{i=1}^{n} u_{ij}^m \; t_{ij}^\eta \; \vec{x}_i}{\sum\limits_{i=1}^{n} u_{ij}^m \; t_{ij}^\eta} \; ; \; 1 \leq j \leq k$$

## Value

an object of class 'ppclust', which is a list consists of the following items:

`v` a numeric matrix containing the final cluster prototypes.

`t` a numeric matrix containing the typicality degrees of the data objects.

`d` a numeric matrix containing the distances of objects to the final cluster prototypes.

`x` a numeric matrix containing the processed data set.

`cluster` a numeric vector containing the cluster labels found by defuzzifying the typicality degrees of the objects.

`csize` a numeric vector for the number of objects in the clusters.

`k` an integer for the number of clusters.

`m` a number for the used fuzziness exponent.

`eta` a number for the used typicality exponent.

`omega` a numeric vector of reference distances.

`iter` an integer vector for the number of iterations in each start of the algorithm.

`best.start` an integer for the index of start that produced the minimum objective functional.

`func.val` a numeric vector for the objective function values in each start of the algorithm.

`comp.time` a numeric vector for the execution time in each start of the algorithm.

`stand` a logical value, `TRUE` shows that `x` data set contains the standardized values of raw data.

`wss` a number for the within-cluster sum of squares for each cluster.

`bwss` a number for the between-cluster sum of squares.

`tss` a number for the total within-cluster sum of squares.

`twss` a number for the total sum of squares.

`algorithm` a string for the name of partitioning algorithm. It is 'PCM' with this function.

`call` a string for the matched function call generating this 'ppclust' object.

## Author(s)

Zeynel Cebeci, Alper Tuna Kavlak & Figen Yildiz

### References

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding, in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 1027-1035. <http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>

Szilagyi, L. & Szilagyi, S. M. (2014). Generalization rules for the suppressed fuzzy c-means clustering algorithm. *Neurocomputing*, 139:298-309. <doi:10.1016/j.neucom.2014.02.027>

Gosztolya, G. & Szilagyi, L. (2015). Application of fuzzy and possibilistic c-means clustering models in blind speaker clustering. *Acta Polytechnica Hungarica*, 12(7):41-56. <http://publicatio.bibl.u-szeged.hu/6151/1/2015-acta-polytechnica.pdf>

### See Also

ekm, fcm, fcm2, fpcm, gg, gk, gkpfcm, hcm, pca, pcm, pcmr, upfc

### Examples

```
# Load dataset X16
data(x16)
x <- x16[,-3]
# Initialize the prototype matrix using K-means++
v <- inaparc::kmpp(x, k=2)$v
# Initialize the memberships degrees matrix
u <- inaparc::imembrand(nrow(x), k=2)$u

# Run FPPPCM
res.fpppcm <- fpppcm(x, centers=v, memberships=u, m=2, eta=2)

# Display typicality degrees
res.fpppcm$t

# Run FPPPCM for eta=3
res.fpppcm <- fpppcm(x, centers=v, memberships=u, m=2, eta=3)

# Display typicality degrees
res.fpppcm$t
```

---

## get.dmetrics *List the names of distance metrics*

---

### Description

Displays the distance metric options for calculation of the distances between the data objects and the cluster centers.

### Usage

```
get.dmetrics(dmt="all")
```

## Arguments

dmt — a string for the type of distance metrics. The default is 'all' for the list of all of the distance metrics which are available in the package. The other options are 'l1', 'l2', 'lp', 'sl2' and 'sc' for L1, L2, Lp, squared L2 and squared Chord, respectively.

## Value

dmlist — a two-column matrix containing the distance metrics and their descriptions.

## Author(s)

Zeynel Cebeci

## References

Podani, J. (2000). *Introduction to the Exploration of Multivariate Biological Data*. Backhuys Publishers, Leiden, The Netherlands. 407 pages. <ISBN 90-5782-067-6>

McCune, B., & Grace, J. B. (2002). *Analysis of ecological communities*. Gleneden Beach, Oregon: MjM Software Design. <ISBN:0972129006>

## See Also

ppclust-package

## Examples

```
# Get all distance metrics
dmlist <- get.dmetrics(dmt="all")
dmlist

# Get only L2 type distance metrics
dmlist <- get.dmetrics(dmt="l2")
dmlist
```

---

# gg *Gath-Geva Clustering Algorithm*

---

## Description

Partitions a numeric data set by using the Gath-Geva (GG) clustering algorithm (Gath & Geva, 1989). The function gg is based on the code in fuzzy.GG function of the package **advclust** by Bagus and Pramana (2016) with some more extended initialization methods and distance metrics.

## Usage

```
gg(x, centers, memberships, m=2, ggversion="simple",
   dmetric="sqeuclidean", pw = 2, alginitv="kmpp",
   alginitu="imembrand", nstart=1, iter.max=1e03, con.val=1e-09,
   fixcent=FALSE, fixmemb=FALSE, stand=FALSE, numseed)
```

## Arguments

| | |
|---|---|
| `x` | a numeric vector, data frame or matrix. |
| `centers` | an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers. |
| `memberships` | a numeric matrix containing the initial membership degrees. If missing, it is internally generated. |
| `ggversion` | a string for the version of Gath-Geva algorithm. The default is '`simple`' for the simplified version (Hoepner et al (1999). Use '`original`' for the original Gath-Geva algorithm (Gath & Geva, 1989). |
| `m` | a number greater than 1 to be used as the fuzziness exponent or fuzzifier. The default is 2. |
| `dmetric` | a string for the distance metric. The default is '`sqeuclidean`' for the squared Euclidean distances. See `get.dmetrics` for the alternative options. |
| `pw` | a number for the power of Minkowski distance calculation. The default is 2 if the `dmetric` is '`minkowski`'. |
| `alginitv` | a string for the initialization of cluster prototypes matrix. The default is '`kmpp`' for K-means++ initialization method (Arthur & Vassilvitskii, 2007). For the list of alternative options see `get.algorithms`. |
| `alginitu` | a string for the initialization of memberships degrees matrix. The default is '`imembrand`' for random sampling of initial membership degrees. |
| `nstart` | an integer for the number of starts for clustering. The default is 1. |
| `iter.max` | an integer for the maximum number of iterations allowed. The default is 1000. |
| `con.val` | a number for the convergence value between the iterations. The default is 1e-09. |
| `fixcent` | a logical flag to make the initial cluster centers not changed along the different starts of the algorithm. The default is `FALSE`. If it is `TRUE`, the initial centers are not changed in the successive starts of the algorithm when the `nstart` is greater than 1. |
| `fixmemb` | a logical flag to make the initial membership degrees not changed along the different starts of the algorithm. The default is `FALSE`. If it is `TRUE`, the initial memberships are not changed in the successive starts of the algorithm when the `nstart` is greater than 1. |
| `stand` | a logical flag to standardize data. Its default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized. |
| `numseed` | a seeding number to set the seed of R's random number generator. |

## Details

Gath and Geva (1989) proposed that the fuzzy maximum likelihood estimates (FMLE) clustering algorithm can be used to detect clusters of varying shapes, sizes and densities. Instead of using Euclidean distance as in FCM, Gath-Geva (GG) algorithm uses a distance norm based on fuzzy maximum likelihood estimates as described by Bezdek & Dunn (1975). These are the Gauss distances, and hence, GG is also so-called Gaussian Mixture Decomposition.

The objective function of GG is:

$$J_{GG}(\mathbf{X};\mathbf{V},\mathbf{A},\mathbf{U}) = \sum_{i=1}^{n}\sum_{j=1}^{k} u_{ij}^{m} d_{A_j}(\vec{x}_i,\vec{v}_j)$$

In the above equation, $d_{A_j}(\vec{x}_i,\vec{v}_j)$ is the Gauss distance between the object $\vec{x}_j$ and cluster prototype $\vec{v}_i$.

$$d_{A_j}(\vec{x}_i,\vec{v}_j) = \frac{(2\pi)^{\frac{n}{2}}\sqrt{\det \mathbf{A}_j}}{\alpha_j}\; exp\big(\tfrac{1}{2}(\vec{x}_i-\vec{v}_j)^T\mathbf{A}_j^{-1}(\vec{x}_i-\vec{v}_j)\big)$$

$$\mathbf{A}_j = \frac{\sum_{i=1}^{n} u_{ij}^{m}(\vec{x}_i-\vec{v}_j)^T(\vec{x}_i-\vec{v}_j)}{\sum_{i=1}^{n} u_{ij}^{m}} \;;\; 1 \leq j \leq k$$

The argument $\alpha_j$ is the prior probability for belonging $\vec{x}_i$ to the cluster $j$:

$$\alpha_j = \frac{\sum_{i=1}^{n} u_{ij}^{m}}{n}$$

The argument $\alpha_j$ is used to evaluate the size of a cluster since bigger clusters attract more elements.

$m$ is the fuzzifier to specify the amount of fuzziness for the clustering. Although it is 1 in the original FMLE algorithm (Bezdek & Dunn, 1975) a higher value of it ($1 \leq m \leq \infty$) can be used to make the partition more fuzzy compensating the exponential term of the distance norm (Balasko et al, 2005).

The objective function of GG is minimized by using the following update equations:

$$u_{ij} = \left[\sum_{j=1}^{k}\left(\frac{d_{A_j}(\vec{x}_i,\vec{v}_j)}{d_{A_l}(\vec{x}_i,\vec{v}_l)}\right)^{1/(m-1)}\right]^{-1} \;;1 \leq i \leq n,\; 1 \leq l \leq k$$

$$\vec{v}_j = \frac{\sum_{i=1}^{n} u_{ij}^{m}\vec{x}_i}{\sum_{i=1}^{n} u_{ij}^{m}} \;;1 \leq j \leq k$$

## Value

an object of class 'ppclust', which is a list consists of the following items:

x a numeric matrix containing the processed data set.

v a numeric matrix containing the final cluster prototypes (centers of clusters).

u a numeric matrix containing the fuzzy memberships degrees of the data objects.

d a numeric matrix containing the distances of objects to the final cluster prototypes.

k an integer for the number of clusters.

m a number for the fuzzifier.

`cluster` a numeric vector containing the cluster labels found by defuzzying the fuzzy membership degrees of the objects.

`csize` a numeric vector containing the number of objects in the clusters.

`iter` an integer vector for the number of iterations in each start of the algorithm.

`best.start` an integer for the index of start that produced the minimum objective functional.

`func.val` a numeric vector for the objective function values in each start of the algorithm.

`comp.time` a numeric vector for the execution time in each start of the algorithm.

`stand` a logical value, `TRUE` shows that data set `x` contains the standardized values of raw data.

`wss` a number for the within-cluster sum of squares for each cluster.

`bwss` a number for the between-cluster sum of squares.

`tss` a number for the total within-cluster sum of squares.

`twss` a number for the total sum of squares.

`algorithm` a string for the name of partitioning algorithm. It is 'FCM' with this function.

`call` a string for the matched function call generating this 'ppclust' object.

## Note

Due to the exponential distance norm, this algorithm needs a good initialization since it converges to a near local optimum. So, usually another clustering algorithm, i.e. FCM, is used to initialize the partition matrix $\mathbf{U}$ (Balasko et al, 2005).

## Author(s)

Zeynel Cebeci

## References

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding, in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 1027-1035. <http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>

Bezdek, J.C. & Dunn J.C. (1975). Optimal fuzzy partitions: A heuristic for estimating the parameters in a mixture of normal dustrubutions. *IEEE Transactions on Computers*, C-24(8):835-838. <doi: 10.1109/T-C.1975.224317>

Gath, I. & Geva, A.B. (1989). Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11 (7): 773-781. <doi:10.1109/34.192473>

Hoeppner, F., Klawonn, F., Kruse, R. & Runkler, T. (1999). *Fuzzy cluster analysis*. New York: John Wiley and Sons. <ISBN:0471988642>

Balasko, B., Abonyi, J. & Feil, B. (2005). Fuzzy clustering and data analysis toolbox. Department of Process Eng., Univ. of Veszprem, Veszprem.

Bagus, A. F. & Pramana, S. (2016). advclust: Object Oriented Advanced Clustering. R package version 0.4. https://CRAN.R-project.org/package=advclust

### See Also

ekm, fcm, fcm2, fpcm, fpppcm, gk, gkpfcm, hcm, pca, pcm, pcmr, pfcm, upfc

### Examples

```
## Not run:
# Load dataset iris
data(iris)
x <- iris[,-5]

# Initialize the prototype matrix using Inofrep algorithm
v <- inaparc::inofrep(x, k=3)$v
# Initialize the memberships degrees matrix
u <- inaparc::imembrand(nrow(x), k=3)$u

# Run Gath & Geva with the initial prototypes and memberships
gg.res <- gg(x, centers=v, memberships=u, m=2)

# Show the fuzzy memberships degrees for the top 5 objects
head(gg.res$u, 5)

## End(Not run)
```

## gk *Gustafson-Kessel Clustering*

### Description

Partitions a numeric data set by using the Gustafson-Kessel (GK) clustering algorithm (Gustafson & Kessel, 1979). Unlike FCM using the Euclidean distance, GK uses cluster specific Mahalanobis distance.

### Usage

```
gk(x, centers, memberships, m=2,
   dmetric="sqeuclidean", pw = 2, alginitv="kmpp",
   alginitu="imembrand", nstart=1, iter.max=1e03, con.val=1e-09,
   fixcent=FALSE, fixmemb=FALSE, stand=FALSE, numseed)
```

### Arguments

| | |
|---|---|
| x | a numeric vector, data frame or matrix. |
| centers | an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers. |
| memberships | a numeric matrix containing the initial membership degrees. If missing, it is internally generated. |
| m | a number greater than 1 to be used as the fuzziness exponent or fuzzifier. The default is 2. |

`dmetric` a string for the distance metric. The default is '`sqeuclidean`' for the squared Euclidean distances. See `get.dmetrics` for the alternative options.

`pw` a number for the power of Minkowski distance calculation. The default is 2 if the `dmetric` is '`minkowski`'.

`alginitv` a string for the initialization of cluster prototypes matrix. The default is '`kmpp`' for K-means++ initialization method (Arthur & Vassilvitskii, 2007). For the list of alternative options see `get.algorithms`.

`alginitu` a string for the initialization of memberships degrees matrix. The default is '`imembrand`' for random sampling of initial membership degrees.

`nstart` an integer for the number of starts for clustering. The default is 1.

`iter.max` an integer for the maximum number of iterations allowed. The default is 1000.

`con.val` a number for the convergence value between the iterations. The default is 1e-09.

`fixcent` a logical flag to make the initial cluster centers not changed along the different starts of the algorithm. The default is `FALSE`. If it is `TRUE`, the initial centers are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`fixmemb` a logical flag to make the initial membership degrees not changed along the different starts of the algorithm. The default is `FALSE`. If it is `TRUE`, the initial memberships are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`stand` a logical flag to standardize data. Its default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized.

`numseed` a seeding number to set the seed of R's random number generator.

## Details

As an extension of basic FCM algorithm, Gustafson and Kessel (GK) clustering algorithm employs an adaptive distance norm in order to detect clusters with different geometrical shapes (Babuska, 2001; Correa et al, 2011).

The objective function of GK is:

$$J_{GK}(\mathbf{X};\mathbf{V},\mathbf{A},\mathbf{U}) = \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{k} u_{ij}^m d_{A_j}(\vec{x}_i,\vec{v}_j)$$

In the above equation $d_{A_j}(\vec{x}_i,\vec{v}_j)$ is the Mahalanobis distance between the data object $\vec{x}_j$ and cluster prototype $\vec{v}_i$.

$$d_{A_j}(\vec{x}_i,\vec{v}_j) = (\vec{x}_i - \vec{v}_j)^T \mathbf{A}_j (\vec{x}_i - \vec{v}_j)$$

As it is understood from the above equation, each cluster has its own norm-inducing matrix in GK, so that the norm matrix $\mathbf{A}$ is a $k$-length tuples of the cluster-specific norm-inducing matrices:

$$\mathbf{A} = \{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_k\}.$$

The objective function of GK cannot be directly minimized with respect to $\mathbf{A}_j$ since it is linear in $\mathbf{A}_j$. For obtaining a feasible solution, $\mathbf{A}_j$ must be constrained in some way. One of the usual ways of accomplishing this is to constrain the determinant of $\mathbf{A}_j$ (Babuska, 2001):

$$\mid \mathbf{A}_j \mid = \rho_j \ , \ \rho_j > 0 \ , \ 1 \leq j \leq k$$

Allowing the matrix $\mathbf{A}_j$ to vary with its determinant fixed corresponds to optimizing the shape of cluster while its volume remains constant. By using the Lagrange-multiplier method, the norm-inducing matrix for the cluster $j$ is defined as follows (Babuska, 2001):

$\mathbf{A}_j = [\rho_i \; det(\mathbf{F}_j)]^{-1/n} {\mathbf{F}_j}^{-1}$,

where:

$n$ is the number of data objects,

$\rho_j$ represents the volume of cluster $j$,

$\mathbf{F}_j$ is the fuzzy covariance matrix for the cluster $j$, and is calculated as follows:

$$\mathbf{F}_j = \frac{\sum\limits_{i=1}^{n} u_{ij}^m \; d^2(\vec{x}_i,\vec{v}_j)}{\sum\limits_{i=1}^{n} u_{ij}^m}$$

$m$ is the fuzzifier to specify the amount of fuzziness for the clustering; $1 \leq m \leq \infty$. It is usually chosen as 2.

GK must satisfy the following constraints:

$$\sum\limits_{j=1}^{k} u_{ij} = 1 \;\; ; \; 1 \leq i \leq n$$

$$\sum\limits_{i=1}^{n} u_{ij} > 0 \;\; ; \; 1 \leq j \leq k$$

The objective function of GK is minimized by using the following update equations:

$$u_{ij} = \Bigg[ \sum\limits_{j=1}^{k} \Big( \frac{d_{A_j}(\vec{x}_i,\vec{v}_j)}{d_{A_l}(\vec{x}_i,\vec{v}_l)} \Big)^{1/(m-1)} \Bigg]^{-1} \;\; ; 1 \leq i \leq n, \; 1 \leq l \leq k$$

$$\vec{v}_j = \frac{\sum\limits_{i=1}^{n} u_{ij}^m \vec{x}_i}{\sum\limits_{i=1}^{n} u_{ij}^m} \;\; ; 1 \leq j \leq k$$

## Value

an object of class 'ppclust', which is a list consists of the following items:

x a numeric matrix containing the processed data set.

v a numeric matrix containing the final cluster prototypes (centers of clusters).

u a numeric matrix containing the fuzzy memberships degrees of the data objects.

d a numeric matrix containing the distances of objects to the final cluster prototypes.

k an integer for the number of clusters.

m a number for the fuzzifier.

cluster a numeric vector containing the cluster labels found by defuzzying the fuzzy membership degrees of the objects.

csize a numeric vector containing the number of objects in the clusters.

iter an integer vector for the number of iterations in each start of the algorithm.

best.start an integer for the index of start that produced the minimum objective functional.

`func.val` a numeric vector for the objective function values in each start of the algorithm.

`comp.time` a numeric vector for the execution time in each start of the algorithm.

`stand` a logical value, `TRUE` shows that data set `x` contains the standardized values of raw data.

`wss` a number for the within-cluster sum of squares for each cluster.

`bwss` a number for the between-cluster sum of squares.

`tss` a number for the total within-cluster sum of squares.

`twss` a number for the total sum of squares.

`algorithm` a string for the name of partitioning algorithm. It is ‘FCM’ with this function.

`call` a string for the matched function call generating this ‘ppclust’ object.

## Author(s)

Zeynel Cebeci & Cagatay Cebeci

## References

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding, in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027-1035. `http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf`

Gustafson, D. E. & Kessel, W. C. (1979). Fuzzy clustering with a fuzzy covariance matrix. In *Proc. of IEEE Conf. on Decision and Control including the 17th Symposium on Adaptive Processes*, San Diego. pp. 761-766. doi:10.1109/CDC.1978.268028

Babuska, R. (2001). Fuzzy and neural control. DISC Course Lecture Notes. Delft University of Technology. Delft, the Netherlands. https://tr.scribd.com/document/209211977/Fuzzy-and-Neural-Control.

Correa, C., Valero, C., Barreiro, P., Diago, M. P., & Tardáguila, J. (2011). A comparison of fuzzy clustering algorithms applied to feature extraction on vineyard. In *Proc. of the 14th Conf. of the Spanish Assoc. for Artificial Intelligence*. `http://oa.upm.es/9246/`

## See Also

`ekm`, `fcm`, `fcm2`, `fpcm`, `fpppcm`, `gg`, `gkpfcm`, `hcm`, `pca`, `pcm`, `pcmr`, `pfcm`, `upfc`

## Examples

```
## Not run:
# Load dataset iris
data(iris)
x <- iris[,-5]

# Initialize the prototype matrix using K-means++
v <- inaparc::kmpp(x, k=3)$v

# Initialize the membership degrees matrix
u <- inaparc::imembrand(nrow(x), k=3)$u
```

```
# Run FCM with the initial prototypes and memberships
gk.res <- gk(x, centers=v, memberships=u, m=2)

# Show the fuzzy membership degrees for the top 5 objects
head(gk.res$u, 5)

## End(Not run)
```

## gkpfcm *Gustafson-Kessel Clustering Using PFCM*

### Description

Partitions a numeric data set by using the Gustafson-Kessel (GK) algorithm within the PFCM (Possibilistic Fuzzy C-Means) clustering algorithm (Ojeda-Magaina et al, 2006).

### Usage

```
gkpfcm(x, centers, memberships, m=2, eta=2, K=1, omega, a, b,
    dmetric="sqeuclidean", pw = 2, alginitv="kmpp",
    alginitu="imembrand", nstart=1, iter.max=1000, con.val=1e-09,
    fixcent=FALSE, fixmemb=FALSE, stand=FALSE, numseed)
```

### Arguments

| | |
|---|---|
| x | a numeric vector, data frame or matrix. |
| centers | an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers. |
| memberships | a numeric matrix containing the initial membership degrees. If missing, it is internally generated. |
| m | a number greater than 1 to be used as the fuzziness exponent or fuzzifier. The default is 2. |
| eta | a number greater than 1 to be used as the typicality exponent. The default is 2. |
| a | a number for the relative importance of the fuzzy part of the objective function. The default is 1. |
| b | a number for the relative importance of the possibilistic part of the objective function. The default is 1. |
| K | a number greater than 0 to be used as the weight of penalty term. The default is 1. |
| omega | a numeric vector of reference distances. If missing, it is internally generated. |
| dmetric | a string for the distance metric. The default is 'sqeuclidean' for the squared Euclidean distances. See get.dmetrics for the alternative options. |
| pw | a number for the power of Minkowski distance calculation. The default is 2 if the dmetric is 'minkowski'. |

`alginitv` a string for the initialization of cluster prototypes matrix. The default is '`kmpp`' for K-means++ initialization method (Arthur & Vassilvitskii, 2007). For the list of alternative options see `get.algorithms`.

`alginitu` a string for the initialization of memberships degrees matrix. The default is '`imembrand`' for random sampling of initial membership degrees.

`nstart` an integer for the number of starts for clustering. The default is 1.

`iter.max` an integer for the maximum number of iterations allowed. The default is 1000.

`con.val` a number for the convergence value between the iterations. The default is 1e-09.

`fixcent` a logical flag to make the initial cluster centers not changed along the different starts of the algorithm. The default is `FALSE`. If it is `TRUE`, the initial centers are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`fixmemb` a logical flag to make the initial membership degrees not changed along the different starts of the algorithm. The default is `FALSE`. If it is `TRUE`, the initial memberships are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`stand` a logical flag to standardize data. Its default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized.

`numseed` a seeding number to set the seed of R's random number generator.

## Details

Gustafson-Kessel clustering within Possibilistic Fuzzy C-Means (GKPFCM) algorithm is an improvement for PFCM algorithm that consists of the modification of the distance metric for $d_{ij_A}$. The original PFCM uses the Euclidean distance whereas GKPFCM uses the Mahalanobis distance with GK algorithm. Babuska et al (2002) have proposed an improvement for calculating the covariance matrix $\mathbf{F}_j$ as follows:

$$\mathbf{F}_j := (1-\gamma)\mathbf{F}_j + \gamma(\mathbf{F}_0)^{1/n}\mathbf{I}$$

In the above equation, $\mathbf{F}_j$ involves a weighted identity matrix. The eigenvalues $\lambda_{ij}$ and the eigenvectors $\Phi_{ij}$ of the resulting matrix are calculated, and the maximum eigenvalue $\lambda_{i,max} = max_j/\lambda_{ij}$ is determined. With the obtained results, $\lambda_{i,max} = \lambda_{ij}/\beta, \forall j$, which satisfies $\lambda_{i,max}/\lambda_{ij} \geq \beta$ is calculated. Finally, $\mathbf{F}_j$ is recomputed with the following equation:

$$\mathbf{F}_j = [\Phi_{j,1},\ldots,\Phi_{j,n}]diag(\lambda_{j,1},\ldots,\lambda_{j,n})[\Phi_{j,1},\ldots,\Phi_{j,n}]^{-1} \;\; \forall j$$

The objective function of GKPFCM is:

$$J_{GKPFCM}(\mathbf{X};\mathbf{V},\mathbf{A},\mathbf{U}) = \sum_{i=1}^{n}\sum_{j=1}^{k} u_{ij}^m d_{A_j}(\vec{x}_i,\vec{v}_j)$$

$m$ is the fuzzifier to specify the amount of fuzziness for the clustering; $1 \leq m \leq \infty$. It is usually chosen as 2.

$\eta$ is the typicality exponent to specify the amount of typicality for the clustering; $1 \leq \eta \leq \infty$. It is usually chosen as 2.

The objective function $J_{GKPFCM}$ is minimized by using the following update equations:

$$u_{ij} = \left[\sum_{j=1}^{k}\left(\frac{d_{A_j}(\vec{x}_i,\vec{v}_j)}{d_{A_j}(\vec{x}_i,\vec{v}_l)}\right)^{2/(m-1)}\right]^{-1} \quad ; 1 \leq i \leq n \; , \; 1 \leq l \leq k$$

$$t_{ij} = \left[ \sum_{j=1}^{k} \left( \frac{d_{A_j}(\vec{x}_i, \vec{v}_j))}{d_{A_j}(\vec{x}_i, \vec{v}_l))} \right)^{2/(\eta-1)} \right]^{-1} \;;\; 1 \leq i \leq n \;;\; 1 \leq l \leq k$$

$$\vec{v}_j = \frac{\sum\limits_{i=1}^{n} (u_{ij}^m + t_{ij}^\eta)\vec{x}_i}{\sum\limits_{i=1}^{n} (u_{ij}^m + t_{ij}^\eta)} \;;1 \leq j \leq k$$

## Value

an object of class ‘ppclust’, which is a list consists of the following items:

`x` a numeric matrix containing the processed data set.

`v` a numeric matrix containing the final cluster prototypes (centers of clusters).

`u` a numeric matrix containing the fuzzy memberships degrees of the data objects.

`d` a numeric matrix containing the distances of objects to the final cluster prototypes.

`k` an integer for the number of clusters.

`m` a number for the fuzzifier.

`eta` a number greater than 1 to be used as the typicality exponent.

`a` a number for the relative importance of the fuzzy part of the objective function.

`b` a number for the relative importance of the possibilistic part of the objective function.

`omega` a numeric vector of reference distances.

`cluster` a numeric vector containing the cluster labels found by defuzzying the fuzzy membership degrees of the objects.

`csize` a numeric vector containing the number of objects in the clusters.

`iter` an integer vector for the number of iterations in each start of the algorithm.

`best.start` an integer for the index of start that produced the minimum objective functional.

`func.val` a numeric vector for the objective function values in each start of the algorithm.

`comp.time` a numeric vector for the execution time in each start of the algorithm.

`stand` a logical value, `TRUE` shows that data set `x` contains the standardized values of raw data.

`wss` a number for the within-cluster sum of squares for each cluster.

`bwss` a number for the between-cluster sum of squares.

`tss` a number for the total within-cluster sum of squares.

`twss` a number for the total sum of squares.

`algorithm` a string for the name of partitioning algorithm. It is ‘FCM’ with this function.

`call` a string for the matched function call generating this ‘ppclust’ object.

## Author(s)

Zeynel Cebeci & Cagatay Cebeci

## References

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding, in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 1027-1035. <http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>

Gustafson, D. E. & Kessel, W. C. (1979). Fuzzy clustering with a fuzzy covariance matrix. In *Proc. of IEEE Conf. on Decision and Control including the 17th Symposium on Adaptive Processes*, San Diego. pp. 761-766. <doi:10.1109/CDC.1978.268028>

Babuska, R., van der Veen, P. J. & Kaymak, U. (2002). Improved covariance estimation for Gustafson-Kessel clustering. In *Proc. of Int. Conf. on Fuzzy Systems*, Hawaii, 2002, pp. 1081-1085. <https://tr.scribd.com/document/209211977/Fuzzy-and-Neural-Control>.

Ojeda-Magaina, B., Ruelas, R., Corona-Nakamura, M. A. & Andina, D. (2006). An improvement to the possibilistic fuzzy c-means clustering algorithm. In *Proc. of IEEE World Automation Congress, 2006 (WAC'06)*. pp. 1-8. <doi:10.1109/WAC.2006.376056>

## See Also

ekm, fcm, fcm2, fpcm, fpppcm, gg, gk, hcm, pca, pcm, pcmr, pfcm, upfc

## Examples

```
## Not run:
# Load dataset iris
data(iris)
x <- iris[,-5]

# Initialize the prototype matrix using K-means++
v <- inaparc::kmpp(x, k=3)$v

# Initialize the memberships degrees matrix
u <- inaparc::imembrand(nrow(x), k=3)$u

# Run FCM with the initial prototypes and memberships
gkpfcm.res <- gkpfcm(x, centers=v, memberships=u, m=2)

# Show the fuzzy membership degrees for the top 5 objects
head(gkpfcm.res$u, 5)

## End(Not run)
```

# hcm *Hard C-Means Clustering*

## Description

Partitions a numeric data set by using Hard C-Means (HCM) clustering algorithm (or K-Means) which has been proposed by MacQueen(1967). The function hcm is an extension of the basic kmeans with more input arguments and output values in order to make the clustering results comparable with those of other fuzzy and possibilistic algorithms. For instance, not only the Euclidean distance metric but also a number of distance metrics such as the squared Euclidean distance, the squared Chord distance etc. can be employed with the function `hcm`.

## Usage

```
hcm(x, centers, dmetric="euclidean", pw=2, alginitv="kmpp",
   nstart=1, iter.max=1000, con.val=1e-9, stand=FALSE, numseed)
```

## Arguments

`x` a numeric vector, data frame or matrix.

`centers` an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers.

`dmetric` a string for the distance metric. The default is '`euclidean`' for the squared Euclidean distances. See `get.dmetrics` for the alternative options.

`pw` a number for the power of Minkowski distance calculation. The default is 2 if the `dmetric` is '`minkowski`'.

`alginitv` a string for the initialization of cluster prototypes matrix. The default is '`kmpp`' for K-means++ initialization method (Arthur & Vassilvitskii, 2007). For the list of alternative options see `get.algorithms`.

`nstart` an integer for the number of starts for clustering. The default is 1.

`iter.max` an integer for the maximum number of iterations allowed. The default is 1000.

`con.val` a number for the convergence value between the iterations. The default is 1e-09.

`stand` a logical flag to standardize data. Its default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized.

`numseed` a seeding number to set the seed of R's random number generator.

## Details

Hard C-Means (HCM) clustering algorithm (or K-means) partitions a data set into $k$ groups, so-called clusters. The objective function of HCM is:

$$J_{HCM}(\mathbf{X};\mathbf{V}) = \sum\limits_{i=1}^n d^2(\vec{x}_i,\vec{v}_j)$$

See `ppclust-package` for the details about the terms in the above equation of $J_{HCM}$.

The update equation for membership degrees is:

$$u_{ij} = \left\{ \begin{array}{rl} 1 & if\; d^2(\vec{x}_i,\vec{v}_j) = min_{1\leq l\leq k} \; (d^2(\vec{x}_i,\vec{v}_l)) \\ 0 & otherwise \end{array} \right.$$

The update equation for cluster prototypes is:

$$\vec{v}_j = \frac{\sum\limits_{i=1}^n u_{ij}\vec{x}_i}{\sum\limits_{i=1}^n u_{ij}} \;\; ; 1 \leq j \leq k$$

## Value

an object of class 'ppclust', which is a list consists of the following items:

`x` a numeric matrix containing the processed data set.

`v` a numeric matrix containing the final cluster prototypes (centers of clusters).

`u` a numeric matrix containing the hard membership degrees of the data objects.

`d` a numeric matrix containing the distances of objects to the final cluster prototypes.

`k` an integer for the number of clusters.

`cluster` a numeric vector containing the cluster labels of the data objects.

`csize` a numeric vector containing the number of objects in the clusters.

`best.start` an integer for the index of start with the minimum objective functional.

`iter` an integer vector for the number of iterations in each start of the algorithm.

`func.val` a numeric vector for the objective function values of each start of the algorithm.

`comp.time` a numeric vector for the execution time of each start of the algorithm.

`wss` a numeric vector containing the within-cluster sum of squares for each cluster.

`bwss` a number for the between-cluster sum of squares.

`tss` a number for the total within-cluster sum of squares.

`twss` a number for the total sum of squares.

`stand` a logical value, `TRUE` shows that `x` data set contains the standardized values of raw data.

`algorithm` a string for the name of partitioning algorithm. It is 'HCM' with this function.

`call` a string for the matched function call generating this 'ppclust' object.

## Author(s)

Zeynel Cebeci & Figen Yildiz

## References

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding, in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 1027-1035. <http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>

MacQueen, J.B. (1967). Some methods for classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symp. on Mathematical Statistics and Probability*, Berkeley, Univ. of California Press, 1: 281-297. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.308.8619&rep=rep1&type=pdf>

## See Also

`kmeans`, `ekm`, `fcm`, `fcm2`, `fpcm`, `fpppcm`, `gg`, `gk`, `gkpfcm`, `pca`, `pcm`, `pcmr`, `pfcm`, `upfc`

## Examples

```
## Not run:
# Load dataset iris
data(iris)
x <- iris[,-5]

# Initialize the prototype matrix using K-means++
v <- inaparc::kmpp(x, k=3)$v

# Run HCM with the initial prototypes
res.hcm <- hcm(x, centers=v)

# Print, summarize and plot the clustering result
res.hcm$cluster
summary(res.hcm$cluster)
plot(x, col=res.hcm$cluster, pch=16)

## End(Not run)
```

---

# `is.ppclust` *Check the class of object for 'ppclust'*

---

## Description

Checks the class of given object whether it is an instance of the `ppclust` class or not.

## Usage

```
is.ppclust(objx)
```

## Arguments

| | |
|---|---|
| `objx` | an object to be checked for its class. |

## Value

`TRUE` if `objx` is a valid `ppclust` object and `FALSE` for the other types of object classes.

## Author(s)

Zeynel Cebeci

## See Also

`as.ppclust`, `ppclust2`, `summary.ppclust`

## Examples

```
data(iris)

# Run FCM for 3 clusters
res.fcm <- fcm(x=iris[,1:4], centers=2)

# Test for a ppclust object returned by the fcm function
is.ppclust(res.fcm)

# Test for a matrix object
x <- matrix(nrow=3, ncol=2, c(1,4,2,5,7,8))
is.ppclust(x)
```

---

# mfpcm *Modified Fuzzy Possibilistic C-Means Clustering*

---

## Description

Partitions a numeric data set by using the Modified Fuzzy and Possibilistic C-Means (MFPCM) clustering algorithm (Saad & Alimi, 2009).

## Usage

```
mfpcm(x, centers, memberships, m=2, eta=2,
      dmetric="sqeuclidean", pw=2, alginitv="kmpp", alginitu="imembrand",
      nstart=1, iter.max=1000, con.val=1e-09,
      fixcent=FALSE, fixmemb=FALSE, stand=FALSE, numseed)
```

## Arguments

| | |
|---|---|
| `x` | a numeric vector, data frame or matrix. |
| `centers` | an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers. |
| `memberships` | a numeric matrix containing the initial membership degrees. If missing, it is internally generated. |
| `m` | a number greater than 1 to be used as the fuzziness exponent or fuzzifier. The default is 2. |
| `eta` | a number greater than 1 to be used as the typicality exponent. The default is 3. |
| `dmetric` | a string for the distance metric. The default is ‘`sqeuclidean`’ for the squared Euclidean distances. See `get.dmetrics` for the alternative options. |
| `pw` | a number for the power of Minkowski distance calculation. The default is 2 if the `dmetric` is ‘`minkowski`’. |
| `alginitv` | a string for the initialization of cluster prototypes matrix. The default is ‘`kmpp`’ for K-means++ initialization method (Arthur & Vassilvitskii, 2007). For the list of alternative options see `get.algorithms`. |

`alginitu` a string for the initialization of memberships degrees matrix. The default is ‘`imembrand`’ for random sampling of initial membership degrees.

`nstart` an integer for the number of starts for clustering. The default is 1.

`iter.max` an integer for the maximum number of iterations allowed. The default is 1000.

`con.val` a number for the convergence value between the iterations. The default is 1e-09.

`fixcent` a logical flag to make the initial cluster centers not changed along the different starts of the algorithm. The default is `FALSE`. If it is `TRUE`, the initial centers are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`fixmemb` a logical flag to make the initial membership degrees not changed along the different starts of the algorithm. The default is `FALSE`. If it is `TRUE`, the initial memberships are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`stand` a logical flag to standardize data. Its default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized.

`numseed` a seeding number to set the seed of R’s random number generator.

## Details

Modified Fuzzy and Possibilistic C Means (MFPCM) algorithm was proposed by Pal et al (1997) intented to incorporate a weight parameter to the objective function of FPCM as follows:

$$J_{MFPCM}(\mathbf{X}; \mathbf{V}, \mathbf{U}, \mathbf{T}) = \sum\limits_{i=1}^n u_{ij}^m w_{ij}^m \; d^{2m}(\vec{x}_i, \vec{v}_j) + t_{ij}^\eta w_{ij}^\eta \; d^{2\eta}(\vec{x}_i, \vec{v}_j)$$

In the above ojective function, every data object is considered to has its own weight in relation to every cluster. Therefore it is expected that the weight permits to have a better classification especially in the case of noise data (Saad & Alimi, 2009). The weight is calculated with the following equation:

$$w_{ij} = exp \Bigg[ - \frac{d^2(\vec{x}_i, \vec{v}_j)}{\sum\limits_{i=1}^n d^2(\vec{x}_i, \bar{v}) \frac{k}{n}} \Bigg]$$

The objective function of MFPCM is minimized by using the following update equations:

$$u_{ij} = \Bigg[ \sum\limits_{j=1}^k \Big( \frac{d^2(\vec{x}_i, \vec{v}_j)}{d^2(\vec{x}_i, \vec{v}_l)} \Big)^{2m/(m-1)} \Bigg]^{-1} \;\; ; 1 \leq i \leq n, \; 1 \leq l \leq k$$

$$t_{ij} = \Bigg[ \sum\limits_{l=1}^n \Big( \frac{d^2(\vec{x}_i, \vec{v}_j)}{d^2(\vec{x}_i, \vec{v}_l)} \Big)^{2\eta/(\eta-1)} \Bigg]^{-1} \;\; ; 1 \leq i \leq n, \; 1 \leq j \leq k$$

$$\vec{v}_j = \frac{\sum\limits_{i=1}^n (u_{ij}^m w_{ij}^m + t_{ij}^\eta w_{ij}^\eta) \vec{x}_i}{\sum\limits_{i=1}^n (u_{ij}^m w_{ij}^m + t_{ij}^\eta) w_{ij}^\eta} \;\; ; 1 \leq j \leq k$$

## Value

an object of class ‘ppclust’, which is a list consists of the following items:

`x` a numeric matrix containing the processed data set.

`v` a numeric matrix containing the final cluster prototypes (centers of clusters).

`u` a numeric matrix containing the fuzzy memberships degrees of the data objects.

`d` a numeric matrix containing the distances of objects to the final cluster prototypes.

`k` an integer for the number of clusters.

`m` a number for the fuzzifier.

`eta` a number for the typicality exponent.

`cluster` a numeric vector containing the cluster labels found by defuzzying the fuzzy membership degrees of the objects.

`csize` a numeric vector containing the number of objects in the clusters.

`iter` an integer vector for the number of iterations in each start of the algorithm.

`best.start` an integer for the index of start that produced the minimum objective functional.

`func.val` a numeric vector for the objective function values in each start of the algorithm.

`comp.time` a numeric vector for the execution time in each start of the algorithm.

`stand` a logical value, `TRUE` shows that data set `x` contains the standardized values of raw data.

`wss` a number for the within-cluster sum of squares for each cluster.

`bwss` a number for the between-cluster sum of squares.

`tss` a number for the total within-cluster sum of squares.

`twss` a number for the total sum of squares.

`algorithm` a string for the name of partitioning algorithm. It is ‘FCM’ with this function.

`call` a string for the matched function call generating this ‘ppclust’ object.

## Author(s)

Zeynel Cebeci, Alper Tuna Kavlak & Figen Yildiz

## References

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding, in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 1027-1035. <`http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf`>

Saad, M. F. & Alimi, A. M. (2009). Modified fuzzy possibilistic c-means. In *Proc. of the Int. Multiconference of Engineers and Computer Scientists*, 1: 18-20. <ISBN:978-988-17012-2-0>

## See Also

`ekm`, `fcm`, `fcm2`, `fpppcm`, `gg`, `gk`, `gkpfcm`, `hcm`, `pca`, `pcm`, `pcmr`, `pfcm`, `upfc`

## Examples

```
# Load dataset iris
data(iris)
x <- iris[,-5]

# Initialize the prototype matrix using K-means++
v <- inaparc::kmpp(x, k=3)$v

# Initialize the memberships degrees matrix
u <- inaparc::imembrand(nrow(x), k=3)$u

# Run FCM with the initial prototypes and memberships
mfpcm.res <- mfpcm(x, centers=v, memberships=u, m=2, eta=2)

# Show the fuzzy membership degrees for the top 5 objects
head(mfpcm.res$u, 5)

# Show the possibilistic membership degrees for the top 5 objects
head(mfpcm.res$t, 5)
```

# pca *Possibilistic Clustering Algorithm*

## Description

Partitions a numeric data set by using the Possibilistic Clustering Algorithm (PCA) which has been proposed by Yang & Wu (2006).

## Usage

```
pca(x, centers, memberships, m=2, eta=2,
   dmetric="sqeuclidean", pw=2, alginitv="kmpp", alginitu="imembrand",
   nstart=1, iter.max=1000, con.val=1e-09,
   fixcent=FALSE, fixmemb=FALSE, stand=FALSE, numseed)
```

## Arguments

`x` a numeric vector, data frame or matrix.

`centers` an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers.

`memberships` a numeric matrix containing the initial membership degrees. If missing, it is internally generated.

`m` a number greater than 1 to be used as the fuzziness exponent. The default is 2.

`eta` a number greater than 1 to be used as the typicality exponent. The default is 2.

`dmetric` a string for the distance metric. The default is ‘`sqeuclidean`’ for the squared Euclidean distances. See `get.dmetrics` for the alternative options.

`pw` a number for the power of Minkowski distance calculation. The default is 2 if the `dmetric` is ‘`minkowski`’.

`alginitv` a string for the initialization of cluster prototypes matrix. The default is ‘`kmpp`’ for K-means++ initialization method (Arthur & Vassilvitskii, 2007). For the list of alternative options see `get.algorithms`.

`alginitu` a string for the initialization of memberships degrees matrix. The default is ‘`imembrand`’ for random sampling of initial membership degrees.

`nstart` an integer for the number of starts for clustering. The default is 1.

`iter.max` an integer for the maximum number of iterations allowed. The default is 1000.

`con.val` a number for the convergence value between the iterations. The default is 1e-09.

`fixcent` a logical flag to fix the initial cluster centers. The default is `FALSE`. If it is `TRUE`, the initial centers are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`fixmemb` a logical flag to fix the initial membership degrees. The default is `FALSE`. If it is `TRUE`, the initial memberships are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`stand` a logical flag to standardize data. Its default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized.

`numseed` a seeding number to set the seed of R’s random number generator.

## Details

Unlike the Possibilistic C-Means (PCM) algorithm requiring the results of a previous run of Fuzzy C-Means (FCM) clustering in order to calculate the parameter $\Omega$, Possibilistic Clustering Algorithm (PCA) is based on the FCM objective function, the partition coefficient (PC) and partition entropy (PE) validity indexes. So that PCA directly computes the typicality values and needs not run FCM beforehand to compute this parameter. The resulting membership becomes the exponential function, hence, it is reported that it is robust to noise and outliers (Yang & Wu, 2006). However, Wu et al (2010) reported that PCA is very sensitive to initializations and sometimes generates coincident clusters.

The objective function of PCA is:

$$J_{PCA}(\mathbf{X};\mathbf{V},\mathbf{T}) = \sum\limits_{j=1}^k \sum\limits_{i=1}^n t_{ij}^m \; d^2(\vec{x}_i,\vec{v}_j) + \frac{\beta}{m^2\sqrt{k}} \sum\limits_{j=1}^k \sum\limits_{i=1}^n (t_{ij}^m \; log \; t_{ij}^m - t_{ij}^m)$$

Where:

$$t_{ij} = exp\Big(- \frac{m\sqrt{k} \; d^2(\vec{x}_i,\vec{v}_j)}{\beta}\Big) \;\; ; 1 \leq i \leq n, \; 1 \leq j \leq k$$

The update equation for cluster prototypes:

$$\vec{v}_j = \frac{\sum\limits_{i=1}^n t_{ij}^m \; \vec{x}_i}{\sum\limits_{i=1}^n t_{ij}^m} \;\; ; 1 \leq j \leq k$$

Where:

$$\beta = \frac{\sum\limits_{i=1}^n \; d^2(\vec{x}_i,\overline{x})}{n} \text{ with } \overline{x} = \frac{\sum\limits_{i=1}^n \vec{x}_i}{n}$$

## Value

an object of class 'ppclust', which is a list consists of the following items:

`v` a numeric matrix containing the final cluster prototypes.

`u` a numeric matrix containing the fuzzy membership degrees of the data objects.

`t` a numeric matrix containing the typicality degrees of the data objects.

`d` a numeric matrix containing the distances of objects to the final cluster prototypes.

`x` a numeric matrix containing the processed data set.

`cluster` a numeric vector containing the cluster labels found by defuzzifying the typicality degrees of the objects.

`csize` a numeric vector containing the number of objects in the clusters.

`k` an integer for the number of clusters.

`m` a number for the fuziness exponent.

`eta` a number for the typicality exponent.

`omega` a numeric vector of reference distances.

`iter` an integer vector for the number of iterations in each start of the algorithm.

`best.start` an integer for the index of start that produced the minimum objective functional.

`func.val` a numeric vector for the objective function values in each start of the algorithm.

`comp.time` a numeric vector for the execution time in each start of the algorithm.

`stand` a logical value, `TRUE` shows that data set `x` contains the standardized values of raw data.

`wss` a number for the within-cluster sum of squares for each cluster.

`bwss` a number for the between-cluster sum of squares.

`tss` a number for the total within-cluster sum of squares.

`twss` a number for the total sum of squares.

`algorithm` a string for the name of partitioning algorithm. It is 'PCM' with this function.

`call` a string for the matched function call generating this 'ppclust' object.

## Author(s)

Zeynel Cebeci, Alper Tuna Kavlak

## References

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding, in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 1027-1035. <`http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf`>

Yang, M. S. & Wu, K. L. (2006). Unsupervised possibilistic clustering. *Pattern Recognition*, 39(1): 5-21. <doi:10.1016/j.patcog.2005.07.005>

### See Also

ekm, fcm, fcm2, fpcm, fpppcm, gg, gk, gkpfcm, hcm, pcm, pcmr, pfcm, upfc

### Examples

```
# Load dataset X16
data(x16)
x <- x16[,-3]

# Initialize the prototype matrix using K-means++
v <- inaparc::kmpp(x, k=2)$v
# Initialize the membership degrees matrix
u <- inaparc::imembrand(nrow(x), k=2)$u

# Run PCA
pca.res <- pca(x, centers=v, memberships=u, m=2, eta=2)

# Display the fuzzy membership degrees
print(round(pca.res$u,2))

# Display the typicality degrees
print(round(pca.res$t,2))
```

## pcm *Possibilistic C-Means Clustering*

### Description

Partitions a numeric multidimensional data set by using the Possibilistic C-Means (PCM) clustering algorithm which has been proposed by Krishnapuram & Keller (1993, 1996).

### Usage

```
pcm(x, centers, memberships, eta=2, K=1, omega, oftype = 1,
    dmetric="sqeuclidean", pw=2, fcmrun=TRUE,
    alginitv="kmpp", alginitu="imembrand",
    nstart=1, iter.max=1000, con.val=1e-09,
    fixcent=FALSE, fixmemb=FALSE, stand=FALSE, numseed)
```

### Arguments

`x` a numeric vector, data frame or matrix.

`centers` an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers.

`memberships` a numeric matrix containing the initial membership degrees. If missing, it is internally generated.

`eta` a number greater than 1 to be used as the typicality exponent. The default is 2.

`K` a number greater than 0 to be used as the weight of penalty term. The default is 1.

`omega` a numeric vector of reference distances. If missing, it is internally generated.

`oftype` an integer for the type of objective function. The default option is '`1`' for the PCM objective function in Krishnapuram & Keller (1993). Use '`2`' for PCM objective function in Krishnapuram & Keller (1996).

`dmetric` a string for the distance metric. The default is '`sqeuclidean`' for the squared Euclidean distances. See `get.dmetrics` for the alternative options.

`pw` a number for the power of Minkowski distance calculation. The default is '`2`' if the `dmetric` is '`minkowski`'.

`alginitv` a string for the initialization of cluster prototypes matrix. The default is '`kmpp`' for K-means++ initialization method (Arthur & Vassilvitskii, 2007). For the list of alternative options see `get.algorithms`.

`alginitu` a string for the initialization of memberships degrees matrix. The default is '`imembrand`' for random sampling of initial membership degrees.

`fcmrun` a logical value for using the results from a previous FCM run. The default is '`TRUE`' for starting the algorithm with the initial centers and memberships from FCM run. Set the value to '`FALSE`' to cancel to use the results of FCM run.

`nstart` an integer for the number of starts for clustering. The default is 1.

`iter.max` an integer for the maximum number of iterations allowed. The default is 1000.

`con.val` a number for the convergence value between the iterations. The default is 1e-09.

`fixcent` a logical value to fix the initial cluster centers. The default is `FALSE`. If it is `TRUE`, the initial centers are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`fixmemb` a logical value to fix the initial membership degrees. The default is `FALSE`. If it is `TRUE`, the initial memberships are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`stand` a logical value to standardize data. The default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized.

`numseed` a seeding number to set the seed of R's random number generator.

## Details

Possibilistic C-Means (PCM) clustering algorithm was proposed by Krishnapuram and Keller (1993) to overcome the noise problem of FCM (Nasraoui, 1999). The algorithm differs from the other clustering algorithms in that the resulting partition of the data can be interpreted as a possibilistic partition, and the membership values can be interpreted as degrees of possibility of the points belonging to the classes, i.e., the compatibilities of the points with the class prototypes (Krishnapuram & Keller, 1993). PCM relaxes the probabilistic constraint on the sum of the memberships of an object to all clusters in FCM. However, PCM must run on the fuzzy clustering results of FCM in order to calculate the parameter $\Omega$. Although it solves the noise sensitivity problem of FCM, the performance of PCM depends heavily on the initialization and often deteriorates due to the coincident clustering problem (Filippone et al, 2007).

The objective function of PCM is:

$$J_{PCM}(\mathbf{X};\mathbf{V},\mathbf{T}) = \sum\limits_{i=1}^n t_{ij}^\eta \; d^2(\vec{x}_i,\vec{v}_j) + \sum\limits_{j=1}^k \Omega_j \sum\limits_{i=1}^n (1-t_{ij})^\eta$$

In the objective function above, the first term leads to a minimization of the weighted distances while the second term suppresses the trivial solution (Timm et al, 2004).

Krishnapuram and Keller later proposed an alternative objective function for PCM (Krishnapuram & Keller, 1996):

$$J_{PCM_2}(\mathbf{X};\mathbf{V},\mathbf{T}) = \sum\limits_{i=1}^n t_{ij}^\eta \; d^2(\vec{x}_i,\vec{v}_j) + \sum\limits_{j=1}^k \Omega_j \sum\limits_{i=1}^n (t_{ij}^\eta \; \log t_{ij}^\eta - t_{ij}^\eta)$$

Where:

$$\vec{\Omega} = K \sum\limits_{i=1}^n u_{ij}^m \; d^2(\vec{x}_i,\vec{v}_j) / \sum\limits_{i=1}^n u_{ij}^m$$

Since the membership calculation in PCM is possibilistic, the membership degrees can be treated as typicality values which measure how typical a data object is for a given cluster, regardless of all other clusters. The update equation of typicality degrees remains identical to those of FCM, and is derived from the objective function of PCM is:

$$t_{ij} = \left[1 + \left(\frac{d^2(\vec{x}_i,\vec{v}_j)}{\Omega_j}\right)^{(1/(m-1)}\right]^{-1} \;\; ; 1 \le i \le n,\; 1 \le j \le k$$

The update equation for cluster prototypes is the same with those of FCM:

$$\vec{v}_j = \frac{\sum\limits_{i=1}^n t_{ij}^m \vec{x}_i}{\sum\limits_{i=1}^n t_{ij}^m} \;\; ; 1 \le j \le k$$

## Value

an object of class ‘ppclust’, which is a list consists of the following items:

`v` a numeric matrix containing the final cluster prototypes.

`t` a numeric matrix containing the typicality degrees of the data objects.

`d` a numeric matrix containing the distances of objects to the final cluster prototypes.

`x` a numeric matrix containing the processed data set.

`cluster` a numeric vector containing the cluster labels found by defuzzifying the typicality degrees of the objects.

`csize` a numeric vector containing the number of objects in the clusters.

`k` an integer for the number of clusters.

`eta` a number for the typicality exponent.

`omega` a numeric vector of reference distances.

`iter` an integer vector for the number of iterations in each start of the algorithm.

`best.start` an integer for the index of start that produced the minimum objective functional.

`func.val` a numeric vector for the objective function values in each start of the algorithm.

`comp.time` a numeric vector for the execution time in each start of the algorithm.

`stand` a logical value, `TRUE` shows that data set `x` contains the standardized values of raw data.

`wss` a number for the within-cluster sum of squares for each cluster.

`bwss` a number for the between-cluster sum of squares.

`tss` a number for the total within-cluster sum of squares.

`twss` a number for the total sum of squares.

`algorithm` a string for the name of partitioning algorithm. It is 'PCM' with this function.

`call` a string for the matched function call generating this 'ppclust' object.

## Note

PCM usually leads to acceptable results, although it suffers from stability problems if it is not initialized with the corresponding probabilistic algorithm. Therefore, it needs fuzzy partitioning results of a probabilistic algorithm, i.e. FCM, in order to compute some input arguments such as the cluster prototypes and mobilization scale parameter (Timm et al, 2004).

## Author(s)

Zeynel Cebeci, Alper Tuna Kavlak

## References

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding, in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 1027-1035. <http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>

Krishnapuram, R. & Keller, J.M. (1993). A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98-110. <doi:10.1109/91.227387>

Krishnapuram, R. & Keller, J.M. (1996). The possibilistic c-means algorithm: insights and recommendations. *IEEE Transactions on Fuzzy Systems*, 4(3):385-393. <doi:10.1109/91.531779>

Timm, H., Borgelt, C., Doring, C. & Kruse, R. (2004). An extension to possibilistic fuzzy cluster analysis. *Fuzzy Sets and Systems*, 147 (1): 3-16. <doi:10.1016/j.fss.2003.11.009>

Filippone, M., Masulli, F., & Rovetta, S. (2007). Possibilistic clustering in feature space. In *Int. Workshop on Fuzzy Logic and Applications*, pp. 219-226. Springer, Berlin, Heidelberg. <doi:10.1007/978-3-540-73400-0_27>

## See Also

`ekm`, `fcm`, `fcm2`, `fpcm`, `fpppcm`, `gg`, `gk`, `gkpfcm`, `hcm`, `pca`, `pcmr`, `pfcm`, `upfc`

## Examples

```
# Load the dataset X12
data(x12)

# Initialize the prototype matrix using K-means++
v <- inaparc::kmpp(x12, k=2)$v
# Initialize the memberships degrees matrix
```

```
u <- inaparc::imembrand(nrow(x12), k=2)$u

# Run FCM with the initial prototypes and memberships
fcm.res <- fcm(x12, centers=v, memberships=u, m=2)

# Run PCM with the prototypes and memberships from FCM run
pcm.res <- pcm(x12, centers=fcm.res$v, memberships=fcm.res$u, eta=2)

# Display the typicality degrees
print(pcm.res$t)

# Plot the crisp memberships using maximum typicality degrees
plotcluster(pcm.res, mt="t", trans=TRUE )

# Plot the crisp memberships using the typicality degrees > 0.5
plotcluster(pcm.res, mt="t", cm="threshold", tv=0.5)
```

## pcmr *Possibilistic C-Means Clustering with Repulsion*

### Description

Partitions a numeric data set by using the Possibilistic C-Means with Repulsion (PCMR) clustering algorithm which has been proposed by Wachs et al (2006).

### Usage

```
pcmr(x, centers, memberships, eta=2, K=1, omega, gamma=15,
    dmetric="sqeuclidean", pw=2, alginitv="kmpp", alginitu="imembrand",
    nstart=1, iter.max=1000, con.val=1e-09,
    fixcent=FALSE, fixmemb=FALSE, stand=FALSE, numseed)
```

### Arguments

`x` a numeric vector, data frame or matrix.

`centers` an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers.

`memberships` a numeric matrix containing the initial membership degrees. If missing, it is internally generated.

`eta` a number greater than 1 to be used as the typicality exponent. The default is 2.

`K` a number greater than 0 to be used as the weight of penalty term. The default is 1.

`omega` a numeric vector of reference distances. If missing, it is internally generated.

`gamma` a number for normalization. Gamma value can be in the range of 0.1 and 200, but generally 10 is used. In Shapira & Wachs(2004) gamma = 15 gave the best accuracy for PCMR.

`dmetric` a string for the distance metric. The default is ‘`sqeuclidean`’ for the squared Euclidean distances. See `get.dmetrics` for the alternative options.

`pw` a number for the power of Minkowski distance calculation. The default is 2 if the `dmetric` is ‘`minkowski`’.

`alginitv` a string for the initialization of cluster prototypes matrix. The default is ‘`kmpp`’ for K-means++ initialization method (Arthur & Vassilvitskii, 2007). For the list of alternative options see `get.algorithms`.

`alginitu` a string for the initialization of memberships degrees matrix. The default is ‘`imembrand`’ for random sampling of initial membership degrees.

`nstart` an integer for the number of starts for clustering. The default is 1.

`iter.max` an integer for the maximum number of iterations allowed. The default is 1000.

`con.val` a number for the convergence value between the iterations. The default is 1e-09.

`fixcent` a logical flag to fix the initial cluster centers. The default is `FALSE`. If it is `TRUE`, the initial centers are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`fixmemb` a logical flag to fix the initial membership degrees. The default is `FALSE`. If it is `TRUE`, the initial memberships are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`stand` a logical flag to standardize data. Its default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized.

`numseed` a seeding number to set the seed of R’s random number generator.

## Details

Possibilistic C-Means with Repulsion (PCMR) aims to minimize the intracluster distances while maximizing the intercluster distances without using implicitly the constraints of FCM, but by adding a cluster repulsion term to the objective function of PCM (Wachs et al, 2006).

$$J_{PCMR}(\mathbf{X};\mathbf{V},\mathbf{T}) = \sum\limits_{i=1}^n t_{ij}^\eta \; d^2(\vec{x}_i,\vec{v}_j) + \sum\limits_{j=1}^k \Omega_j \sum\limits_{i=1}^n (1-t_{ij})^\eta + \gamma \sum\limits_{j=1}^k \sum\limits_{l=1,l\neq j}^k (1/d^2(\vec{v}_j,\vec{v}_l))$$

Where $\gamma$ is a weighting factor, and $t_{ij}$ satisfies:

$$t_{ij} \in [0,1], \forall j$$

The repulsion term is relevant if the clusters are close enough. When the distance increases it becomes smaller until it is compensated by the attraction of the clusters. On the other hand, if the clusters are sufficiently spread out, and the intercluster distance decreases (due to the first two terms), the attraction of the cluster can be compensated only by the repulsion term.

The update equation for the cluster prototypes:

$$\vec{v}_j = \frac{\sum\limits_{i=1}^n t_{ij}\vec{x}_i - \gamma \sum\limits_{j=1}^k v_j \; (1/d^2(\vec{v}_j,\vec{v}_l))}{\sum\limits_{i=1}^n t_{ij} - \gamma \sum\limits_{j=1}^k v_j \; (1/d^2(\vec{v}_j,\vec{v}_l))} \; ; \; 1 \leq l \leq k$$

## Value

an object of class 'ppclust', which is a list consists of the following items:

`v` a numeric matrix containing the final cluster prototypes.

`t` a numeric matrix containing the typicality degrees of the data objects.

`d` a numeric matrix containing the distances of objects to the final cluster prototypes.

`x` a numeric matrix containing the processed data set.

`cluster` a numeric vector containing the cluster labels found by defuzzifying the typicality degrees of the objects.

`csize` a numeric vector containing the number of objects in the clusters.

`k` an integer for the number of clusters.

`eta` a number for the typicality exponent.

`omega` a numeric vector of reference distances.

`gamma` a number for normalization.

`iter` an integer vector for the number of iterations in each start of the algorithm.

`best.start` an integer for the index of start that produced the minimum objective functional.

`func.val` a numeric vector for the objective function values in each start of the algorithm.

`comp.time` a numeric vector for the execution time in each start of the algorithm.

`stand` a logical value, `TRUE` shows that `x` data set contains the standardized values of raw data.

`wss` a number for the within-cluster sum of squares for each cluster.

`bwss` a number for the between-cluster sum of squares.

`tss` a number for the total within-cluster sum of squares.

`twss` a number for the total sum of squares.

`algorithm` a string for the name of partitioning algorithm. It is 'PCM' with this function.

`call` a string for the matched function call generating this 'ppclust' object.

## Author(s)

Zeynel Cebeci, A. Tuna Kavlak, Figen Yildiz

## References

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding, in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 1027-1035. <http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>

Wachs, J., Shapira, O. & Stern, H. (2006). A method to enhance the 'Possibilistic C-Means with Repulsion' algorithm based on cluster validity index. In *Applied Soft Computing Technologies: The Challenge of Complexity*, pp. 77-87. Springer, Berlin, Heidelberg. <doi:10.1007/3-540-31662-0_6>

## See Also

ekm, fcm, fcm2, fpcm, fpppcm, gg, gk, gkpfcm, hcm, pca, pcm, pfcm, upfc

## Examples

```
# Load data set X12
data(x12)

# Initialize the prototype matrix using K-means++
v <- inaparc::kmpp(x12, k=2)$v
# Initialize the memberships degrees matrix
u <- inaparc::imembrand(nrow(x12), k=2)$u

# Run FCM with the initial prototypes and memberships
fcm.res <- fcm(x12, centers=v, memberships=u, m=2)

# Run PCMR with the prototypes and memberships from FCM run
pcmr.res <- pcmr(x12, centers=fcm.res$v, memberships=fcm.res$u, eta=2)

# Show the typicality degrees for the top 5 objects
head(pcmr.res$t, 5)

# Plot the crisp memberships using maximum typicality degrees
plotcluster(pcmr.res, mt="t", cm="max")

# Plot the crisp memberships using the typicality degrees > 0.5
plotcluster(pcmr.res, mt="t", cm="threshold", tv=0.5)
```

---

# pfcm *Possibilistic Fuzzy C-Means Clustering Algorithm*

---

## Description

Partitions a numeric data set by using the Possibilistic Fuzzy C-Means (PFCM) clustering algorithm proposed by Pal et al (2005).

## Usage

```
pfcm(x, centers, memberships, m=2, eta=2, K=1, omega, a, b,
    dmetric="sqeuclidean", pw=2, alginitv="kmpp", alginitu="imembrand",
    nstart=1, iter.max=1000, con.val=1e-09,
    fixcent=FALSE, fixmemb=FALSE, stand=FALSE, numseed)
```

## Arguments

| | |
|---|---|
| x | a numeric vector, data frame or matrix. |
| centers | an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers. |

`memberships` a numeric matrix containing the initial membership degrees. If missing, it is internally generated.

`m` a number greater than 1 to be used as the fuzziness exponent. The default is 2.

`eta` a number greater than 1 to be used as the typicality exponent. The default is 2.

`a` a number for the relative importance of the fuzzy part of the objective function. The default is 1.

`b` a number for the relative importance of the possibilistic part of the objective function. The default is 1.

`K` a number greater than 0 to be used as the weight of penalty term. The default is 1.

`omega` a numeric vector of reference distances. If missing, it is internally generated.

`dmetric` a string for the distance metric. The default is ‘`sqeuclidean`’ for the squared Euclidean distances. See `get.dmetrics` for the alternative options.

`pw` a number for the power of Minkowski distance calculation. The default is 2 if the `dmetric` is ‘`minkowski`’.

`alginitv` a string for the initialization of cluster prototypes matrix. The default is ‘`kmpp`’ for K-means++ initialization method (Arthur & Vassilvitskii, 2007). For the list of alternative options see `get.algorithms`.

`alginitu` a string for the initialization of memberships degrees matrix. The default is ‘`imembrand`’ for random sampling of initial membership degrees.

`nstart` an integer for the number of starts for clustering. The default is 1.

`iter.max` an integer for the maximum number of iterations allowed. The default is 1000.

`con.val` a number for the convergence value between the iterations. The default is 1e-09.

`fixcent` a logical flag to fix the initial cluster centers. The default is `FALSE`. If it is `TRUE`, the initial centers are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`fixmemb` a logical flag to fix the initial membership degrees. The default is `FALSE`. If it is `TRUE`, the initial memberships are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`stand` a logical flag to standardize data. Its default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized.

`numseed` a seeding number to set the seed of R’s random number generator.

## Details

In FPCM, the constraint corresponding to the sum of all the typicality values of all data objects to a cluster must be equal to one causes problems; particularly for a big data set. In order to avoid this problem Pal et al (2005) proposed Possibilistic Fuzzy C-Means (PFCM) clustering algorithm with following objective function:

$$J_{PFCM}(\mathbf{X};\mathbf{V},\mathbf{U},\mathbf{T}) = \sum_{j=1}^{k} \sum_{i=1}^{n} (a \; u_{ij}^{m} + b \; t_{ij}^{\eta}) \; d^2(\vec{x}_i, \vec{v}_j) + \sum_{j=1}^{k} \Omega_j \sum_{i=1}^{n} (1 - t_{ij})^{\eta}$$

The fuzzy membership degrees in the probabilistic part of the objective function $J_{PFCM}$ is calculated in the same way as in FCM, as follows:

$$u_{ij} = \left[ \sum_{j=1}^{k} \left( \frac{d^2(\vec{x}_i, \vec{v}_j)}{d^2(\vec{x}_i, \vec{v}_l)} \right)^{1/(m-1)} \right]^{-1} \;\; ; \; 1 \leq i \leq n, \; 1 \leq l \leq k$$

The typicality degrees in the possibilistic part of the objective function $J_{PFCM}$ is calculated as follows:

$$t_{ij} = \left[ 1 + \left( \frac{b \; d^2(\vec{x}_i, \vec{v}_j)}{\Omega_j} \right)^{1/(\eta-1)} \right]^{-1} \;\; ; \; 1 \leq i \leq n, \; 1 \leq j \leq k$$

The constraints with PFCM are:

$$0 \leq u_{ij}, t_{ij} \leq 1$$

$$0 \leq \sum_{i=1}^{n} u_{ij} \leq n \;\; ; \; 1 \leq j \leq k$$

$$0 \leq \sum_{j=1}^{k} t_{ij} \leq k \;\; ; \; 1 \leq i \leq n$$

$$\sum_{j=1}^{k} u_{ij} = 1 \;\; ; \; 1 \leq i \leq n$$

$a$ and $b$ are the coefficients to define the relative importance of fuzzy membership and typicality degrees for weighting the probabilistic and possibilistic terms of the objective function, $a > 0; \; b > 0$.

$m$ is the fuzzifier to specify the amount of fuzziness for the clustering; $1 \leq m \leq \infty$. It is usually chosen as 2.

$\eta$ is the typicality exponent to specify the amount of typicality for the clustering; $1 \leq \eta \leq \infty$. It is usually chosen as 2.

$\Omega$ is the possibilistic penalty terms for controlling the variance of the clusters.

The update equation for cluster prototypes:

$$\vec{v}_j = \frac{\sum_{i=1}^{n} (a \; u_{ij}^m + b \; t_{ij}^\eta) \; \vec{x}_i}{\sum_{i=1}^{n} (a \; u_{ij}^m + b \; t_{ij}^\eta)} \;\; ; \; 1 \leq j \leq k$$

## Value

an object of class 'ppclust', which is a list consists of the following items:

`v` a numeric matrix containing the final cluster prototypes.

`t` a numeric matrix containing the typicality degrees of the data objects.

`d` a numeric matrix containing the distances of objects to the final cluster prototypes.

`x` a numeric matrix containing the processed data set.

`cluster` a numeric vector containing the cluster labels found by defuzzifying the typicality degrees of the objects.

`csize` a numeric vector for the number of objects in the clusters.

`k` an integer for the number of clusters.

`m` a number for the used fuzziness exponent.

`eta` a number for the used typicality exponent.

`a` a number for the fuzzy part of the objective function.

`b` a number for the possibilistic part of the objective function.

`omega` a numeric vector of reference distances.

`iter` an integer vector for the number of iterations in each start of the algorithm.

`best.start` an integer for the index of start that produced the minimum objective functional.

`func.val` a numeric vector for the objective function values in each start of the algorithm.

`comp.time` a numeric vector for the execution time in each start of the algorithm.

`stand` a logical value, `TRUE` shows that `x` data set contains the standardized values of raw data.

`wss` a number for the within-cluster sum of squares for each cluster.

`bwss` a number for the between-cluster sum of squares.

`tss` a number for the total within-cluster sum of squares.

`twss` a number for the total sum of squares.

`algorithm` a string for the name of partitioning algorithm. It is ‘PCM’ with this function.

`call` a string for the matched function call generating this ‘ppclust’ object.

## Author(s)

Zeynel Cebeci, Alper Tuna Kavlak & Figen Yildiz

## References

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding, in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 1027-1035. <`http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf`>

Pal, N. R., Pal, K. & Bezdek, J. C. (2005). A possibilistic fuzzy c-means clustering algorithm. *IEEE Trans. Fuzzy Systems*, 13 (4): 517-530. <doi:10.1109/TFUZZ.2004.840099>

## See Also

`ekm`, `fcm`, `fcm2`, `fpcm`, `fpppcm`, `gg`, `gk`, `gkpfcm`, `hcm`, `pca`, `pcm`, `pcmr`, `upfc`

## Examples

```
# Load the dataset X12
data(x12)

# Set the initial centers of clusters
v0 <- matrix(nrow=2, ncol=2, c(-3.34, 1.67, 1.67, 0.00), byrow=FALSE)

# Run FCM with the initial centers in v0
res.fcm <- fcm(x12, centers=v0, m=2)

# Run PFCM with the final centers and memberhips from FCM
res.pfcm <- pfcm(x12, centers=res.fcm$v, memberships=res.fcm$u, m=2, eta=2)
```

```
# Show the typicality and fuzzy membership degrees from PFCM
res.pfcm$t
res.pfcm$u
```

## `plotcluster` *Plot Clustering Results*

### Description

Plots clustering results from a cluster analysis with ‘ppclust’.

### Usage

```
plotcluster(objx, mt, cm, tv, cp=1, pt=19, trans=FALSE)
```

### Arguments

`objx` an object of ‘ppclust’ class.

`mt` a character to specify the membership type. The default is ‘`u`’ for fuzzy membership degrees. The alternative option is ‘`t`’ for typicality degrees. The default is ‘`t`’ for the algorithms which produce both types of membership matrices.

`cm` a character to specify the crisping method. The default is ‘`max`’ for using maximum degree of memberships for each data obejct. The alternative option is ‘`threshold`’ for the membership degrees exceeding a user-specified threshold value with `tv`.

`tv` a number specifying the threshold membership degree when the value of `cm` is selected as ‘`threshold`’. The value of `tv` should be between 0 and 1. The default is ‘`max`’ for using maximum degree of memberships for each data obejct. The alternative option is ‘`threshold`’ for the 0.5.

`cp` an integer for the index of available color palettes. The default is 1. The options are 2, 3, 4 and 5 for different color themes.

`pt` an integer specifying the ASCII code of a point character to be used in plotting. The default is 19 for solid circles. Use ‘`#`’ for displaying the cluster with their cluster labels.

`trans` a logical value for the type of plots. The default is ‘`FALSE`’ for solid point colors. The alternative option is ‘`TRUE`’ for transparent point colors.

### Author(s)

Zeynel Cebeci

### Examples

```
# Run FCM for 3 clusters on the data set Iris
res.fcm <- fcm(x=iris[,-5], centers=3)

par(ask=TRUE)
# Plot the clustering results with solid colors
plotcluster(res.fcm, cp=1)

# Plot the same clustering results with transparent colors
plotcluster(res.fcm, cp=1, trans=TRUE)

# Plot the same clustering results for the memberships > 0.75
plotcluster(res.fcm, cp=1, cm="threshold", tv=0.75, trans=TRUE)
par(ask=FALSE)
```

## ppclust2 *Convert ‘ppclust’ objects to the other types of cluster objects*

### Description

Converts an object of ‘ppclust’ class to the other types of cluster objects.

### Usage

```
ppclust2(objx, otype, ...)
```

### Arguments

| | |
|---|---|
| `objx` | an object of `ppclust` class. |
| `otype` | target object class type for conversion. |
| `...` | additional arguments. |

### Value

an object of `fanny.object`, `summary.fclust`, `kmeans` or `vegclust` class.

### Author(s)

Zeynel Cebeci

### See Also

`as.ppclust`, `is.ppclust`, `summary.ppclust`

## Examples

```
data(iris)
# Create a object of ppclust
opc <- fcm(x=iris[,1:4], centers=3)

# Check the class of opc object
is.ppclust(opc)

# Convert ppclust object 'opc' to the fanny object
ofc <- ppclust2(opc, otype="fanny")

# Check the class of 'ofc' for ppclust
is.ppclust(ofc)

# Check the class of 'ofc'
class(ofc)

# Convert ppclust object 'opc' to fclust object
ofc <- ppclust2(opc, otype="fclust")

# Check the class of 'ofc'
class(ofc)
```

---

# summary.ppclust *Summarize the clustering results*

---

## Description

Summarizes the clustering results for an object which is an instance of ‘ppclust’ class.

## Usage

```
## S3 method for class 'ppclust'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | an object of `ppclust` class to be summarized. |
| `...` | additional arguments for S3 method summary. |

## Value

summary of the clustering results from the object of `ppclust` class.

## Author(s)

Zeynel Cebeci

### See Also

`as.ppclust`, `is.ppclust`, `ppclust2`

### Examples

```
data(iris)
# Run FCM for three clusters
res.fcm <- fcm(x=iris[,1:4], centers=3)

# Summarize the result
summary(res.fcm)
```

## upfc *Unsupervised Possibilistic Fuzzy C-Means Clustering Algorithm*

### Description

Partitions a numeric data set by using the Unsupervised Possibilistic Fuzzy C-Means clustering algorithm which has been proposed by Wu et al (2010).

### Usage

```
upfc(x, centers, memberships, m=2, eta=2, a, b,
    dmetric="sqeuclidean", pw=2, alginitv="kmpp", alginitu="imembrand",
    nstart=1, iter.max=1000, con.val=1e-09,
    fixcent=FALSE, fixmemb=FALSE, stand=FALSE, numseed)
```

### Arguments

`x` a numeric vector, data frame or matrix.

`centers` an integer specifying the number of clusters or a numeric matrix containing the initial cluster centers.

`memberships` a numeric matrix containing the initial membership degrees. If missing, it is internally generated.

`m` a number greater than 1 to be used as the fuzziness exponent. The default is 2.

`eta` a number greater than 1 to be used as the typicality exponent. The default is 2.

`a` a number for the relative importance of the fuzzy part of the objective function. The default is 1.

`b` a number for the relative importance of the possibilistic part of the objective function. The default is 1.

`dmetric` a string for the distance metric. The default is ‘`sqeuclidean`’ for the squared Euclidean distances. See `get.dmetrics` for the alternative options.

`pw` a number for the power of Minkowski distance calculation. The default is 2 if the `dmetric` is ‘`minkowski`’.

`alginitv` a string for the initialization of cluster prototypes matrix. The default is ‘`kmpp`’ for K-means++ initialization method (Arthur & Vassilvitskii, 2007). For the list of alternative options see `get.algorithms`.

`alginitu` a string for the initialization of memberships degrees matrix. The default is ‘`imembrand`’ for random sampling of initial membership degrees.

`nstart` an integer for the number of starts for clustering. The default is 1.

`iter.max` an integer for the maximum number of iterations allowed. The default is 1000.

`con.val` a number for the convergence value between the iterations. The default is 1e-09.

`fixcent` a logical flag to fix the initial cluster centers. The default is `FALSE`. If it is `TRUE`, the initial centers are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`fixmemb` a logical flag to fix the initial membership degrees. The default is `FALSE`. If it is `TRUE`, the initial memberships are not changed in the successive starts of the algorithm when the `nstart` is greater than 1.

`stand` a logical flag to standardize data. Its default value is `FALSE`. If its value is `TRUE`, the data matrix `x` is standardized.

`numseed` a seeding number to set the seed of R’s random number generator.

## Details

Unsupervised Possibilistic Fuzzy C-Means (UPFC) is an extension of Possibilistic Clustering Algorithm (PCA) by Yang & Wu (2006). Wu et al (2010) reported that PCA is very sensitive to initializations and sometimes generates coincident clusters, and proposed the algorithm UPFC to overcome this problem with inspiration by Pal et al’s PFCM algorithm (Pal et al, 2005). The algorithm UPFC produces both possibilistic and probabilistic memberships simultaneously, and overcomes the noise sensitivity problem of FCM and the coincident clusters problem of PCA.

The objective function of UPFC is:

$$J_{UPFC}(\mathbf{X}; \mathbf{V}, \mathbf{U}, \mathbf{T}) = \sum\limits_{j=1}^k \sum\limits_{i=1}^n (a\; u_{ij}^m + b\; t_{ij}^\eta \; d^2(\vec{x}_i, \vec{v}_j) + \frac{\beta}{n^2\sqrt{k}} \; \sum\limits_{j=1}^k \sum\limits_{i=1}^n (t_{ij}^\eta \; log \; t_{ij}^\eta - t_{ij}^\eta)$$

Where:

$$u_{ij} = \Bigg[\sum\limits_{j=1}^k \Big(\frac{d^2(\vec{x}_i, \vec{v}_j)}{d^2(\vec{x}_i, \vec{v}_l)}\Big)^{1/(m-1)} \Bigg]^{-1} \;\; ; 1 \leq i \leq n,\; 1 \leq l \leq k$$

$$t_{ij} = exp\Big(- \frac{b \; n \; \sqrt{k} \; d^2(\vec{x}_i, \vec{v}_j)}{\beta}\Big) \;\; ; 1 \leq i \leq n,\; 1 \leq j \leq k$$

Where:

$$\beta = \frac{\sum\limits_{i=1}^n \; d^2(\vec{x}_i, \overline{x})}{n} \text{ with } \overline{x} = \frac{\sum\limits_{i=1}^n \vec{x}_i}{n}$$

The constraints with UPFC are:

$$0 \leq \sum\limits_{i=1}^n u_{ij} \leq n \;\; ; \; 1 \leq j \leq k$$

$$\sum\limits_{j=1}^k u_{ij} = 1 \;\; ; \; 1 \leq i \leq n$$

$a$ and $b$ are the coefficients to define the relative importance of fuzzy membership and typicality degrees in the objective function, $a > 0; \ b > 0$.

The update equation for cluster prototypes:

$$\vec{v}_j = \frac{\sum\limits_{i=1}^{n} (a \ u_{ij}^m + b \ t_{ij}^\eta) \ \vec{x}_i}{\sum\limits_{i=1}^{n} (a \ u_{ij}^m + b \ t_{ij}^\eta)} \ \ ; 1 \leq j \leq k$$

## Value

an object of class ‘ppclust’, which is a list consists of the following items:

`v` a numeric matrix containing the final cluster prototypes.

`t` a numeric matrix containing the typicality degrees of the data objects.

`d` a numeric matrix containing the distances of objects to the final cluster prototypes.

`x` a numeric matrix containing the processed data set.

`cluster` a numeric vector containing the cluster labels found by defuzzifying the typicality degrees of the objects.

`csize` a numeric vector containing the number of objects in the clusters.

`k` an integer for the number of clusters.

`m` a number for the used fuzziness exponent.

`eta` a number for the used typicality exponent.

`a` a number for the fuzzy part of the objective function.

`b` a number for the possibilistic part of the objective function.

`beta` a numeric vector of normalization...

`iter` an integer vector for the number of iterations in each start of the algorithm.

`best.start` an integer for the index of start that produced the minimum objective functional.

`func.val` a numeric vector for the objective function values in each start of the algorithm.

`comp.time` a numeric vector for the execution time in each start of the algorithm.

`stand` a logical value, `TRUE` shows that `x` data set contains the standardized values of raw data.

`wss` a number for the within-cluster sum of squares for each cluster.

`bwss` a number for the between-cluster sum of squares.

`tss` a number for the total within-cluster sum of squares.

`twss` a number for the total sum of squares.

`algorithm` a string for the name of partitioning algorithm. It is ‘PCM’ with this function.

`call` a string for the matched function call generating this ‘ppclust’ object.

## Author(s)

Zeynel Cebeci, Figen Yildiz & Alper Tuna Kavlak

### References

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding, in *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 1027-1035. <http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>

Pal, N. R., Pal, K. & Bezdek, J. C. (2005). A possibilistic fuzzy c-means clustering algorithm. *IEEE Trans. Fuzzy Systems*, 13 (4): 517-530. <doi:10.1109/TFUZZ.2004.840099>

Yang, M. S. & Wu, K. L. (2006). Unsupervised possibilistic clustering. *Pattern Recognition*, 39(1): 5-21. <doi:10.1016/j.patcog.2005.07.005>

Wu, X., Wu, B., Sun, J. & Fu, H. (2010). Unsupervised possibilistic fuzzy clustering. *J. of Information & Computational Sci.*, 7 (5): 1075-1080. <https://www.researchgate.net/publication/267691137_Unsupervised_Possibilistic_Fuzzy_Clustering>

### See Also

`ekm`, `fcm`, `fcm2`, `fpcm`, `fpppcm`, `gg`, `gk`, `gkpfcm`, `hcm`, `pca`, `pcm`, `pcmr`, `pfcm`

### Examples

```
# Load dataset X16
data(x16)
x <- x16[,-3]

# Initialize the prototype matrix using K-means++
v <- inaparc::kmpp(x, k=2)$v
# Initialize the memberships degrees matrix
u <- inaparc::imembrand(nrow(x), k=2)$u

# Run UPFC
res.upfc <- upfc(x, centers=v, memberships=u, eta=2)

# Display the fuzzy membership degrees
print(round(res.upfc$u, 2))

# Display the typicality degrees
print(round(res.upfc$t, 2))
```

## x12 *Synthetic data set of two variables*

### Description

A synthetic data set described by Pal et al (2005). It consists of two continous variables forming two well-separated clusters in addition to two noise.

### Usage

```
data(x12)
```

### Format

A data frame with 12 rows and 2 numeric variables:

**p1** a numeric variable ranging from -5.0 to 5.0

**p2** a numeric variable ranging from -1.67 to 10.0

### Note

The data set `x12` is recommended to test the performances of the possibilistic and probabilistic clustering algorithms.

### References

Pal, N. R., Pal, K. & Bezdek, J. C. (2005). A possibilistic fuzzy c-means clustering algorithm. *IEEE Trans. Fuzzy Systems*, 13 (4): 517-530. <doi:10.1109/TFUZZ.2004.840099>

### Examples

```
data(x12)
# Descriptive statistics of the data set
summary(x12)
# Plot the data set
pairs(x12, pch=19, cex=2)
```

## x16 *Synthetic data set of two variables forming two clusters*

### Description

A synthetic data set described by Yang & Wu (2006). It consists of two continous variables forming two well-separated clusters in addition to two noise points.

### Usage

```
data(x16)
```

### Format

A data frame with 16 rows and 2 numeric variables:

**p1** a numeric variable ranging from 50 to 150

**p2** a numeric variable ranging from 145 to 200

**cl** a numeric variable ranging from 1 to 4

### Note

The data set `x16` is recommended to test the performances of the possibilistic and noise clustering algorithms.

## References

Yang, M. S. & Wu, K. L. (2006). Unsupervised possibilistic clustering. *Pattern Recognition*, 39(1): 5-21. <doi:10.1016/j.patcog.2005.07.005>

## Examples

```
data(x16)
x <- x16[,-3]
# descriptive statistics of the variables
summary(x)
# scatter plots for the variable pairs
pairs(x, col=x16$cl, pch=20, cex=2)
```

# Index