Package 'inaparc'

July 22, 2025

Type Package

Title Initialization Algorithms for Partitioning Cluster Analysis

Version 1.2.0

Date 2022-06-15

Author Zeynel Cebeci [aut, cre],
Cagatay Cebeci [aut]

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

Description Partitioning clustering algorithms divide data sets into k subsets or partitions so-called clusters. They require some initialization procedures for starting the algorithms. Initialization of cluster prototypes is one of such kind of procedures for most of the partitioning algorithms. Cluster prototypes are the centers of clusters, i.e. centroids or medoids, representing the clusters in a data set. In order to initialize cluster prototypes, the package 'inaparc' contains a set of the functions that are the implementations of several linear time-complexity and loglinear time-complexity methods in addition to some novel techniques. Initialization of fuzzy membership degrees matrices is another important task for starting the probabilistic and possibilistic partitioning algorithms. In order to initialize membership degrees matrices required by these algorithms, a number of functions based on some traditional and novel initialization techniques are also available in the package 'inaparc'.

Depends R (>= 3.3.0)

License GPL (>= 2)

Imports kpeaks, lhs, stats, methods

NeedsCompilation no

Repository CRAN

Date/Publication 2022-06-16 13:50:02 UTC

Contents

`inaparc-package` *Initialization Algorithms for Partitioning Cluster Analysis*

Description

Partitioning clustering algorithms divide data sets into k subsets or partitions which are so-called clusters. They require some initialization procedures for starting to partition the data sets. Initialization of cluster prototypes is one of such kind of procedures for most of the partitioning algorithms. Cluster prototypes are the data elements, i.e. centroids or medoids, representing the clusters in a data set. In order to initialize cluster prototypes, the package '**inaparc**' contains a set of the functions that are the implementations of widely-used algorithms in addition to some novel techniques. Initialization of fuzzy membership degrees matrices is another important task for starting the probabilistic and possibilistic partitioning algorithms. In order to initialize membership degrees matrices required by these algorithms, the package '**inaparc**' contains a number of functions for most of the data independent and dependent initialization techniques (Borgelt, 2005) which are categorized as the linear time-complexity and loglinear time complexity-initialization methods in Celebi et al (2013).

Details

Clustering is one of the most widely used exploratory statistical analysis in data mining. Its goal is to explore the groups of objects that are similar to each other within the group but different from the objects in other groups. According to a common taxonomy, the existing clustering algorithms are classified in two groups: Hierarchical and Non-hierarchical (or flat) algorithms (Rokah & Maimon, 2005). As a dominant subfamily of non-hierarchical algorithms, the partitioning clustering algorithms divide data objects into a pre-defined number of clusters, which are the non-overlapping subsets of data. Although the choice of an appropriate algorithm for any clustering task depends on many criteria or purposes. When data size and dimensions are the concerned criteria, the non-hierarchical algorithms may be more practical way of clustering the large size and high dimensional data sets because they quickly process the large data sets when compared to the hierarchical clustering algorithms.

As the most crowded group of the partitioning clustering tools, the prototype-based algorithms partition data objects into clusters in which each data object is more similar to its prototype than the prototypes of other clusters. On clustering context, a prototype is a typical data item that represents or characterizes a cluster (Tan et al. 2006). Usually, it can be regarded as the most central data point in a data subspace so-called cluster. The prototype of a cluster is so often a centroid, i.e., the mean of all the objects in a cluster. On the other hand, centroids can not be computed for non-numeric data, i.e., on nominal or ordinal data. In such case, medoids can be used as the prototypes of clusters (Tan et al, 2006).

Initialization or seeding is a process for selecting the starting values of cluster prototypes matrix which serves the initial representatives of clusters. It is an important task in partitioning cluster analysis because it is known that the final clustering result is to be highly sensitive to the initial prototypes of the clusters (Khan, 2012). When the prototypes are chosen to be equal or close to the actual centers of clusters in a data set, the partitioning converges quickly and yields quality results. Contrarily, poor initializations of prototype matrix may result with no-good quality of final partitions.

In fuzzy and possibilistic clustering, an object is a member of all clusters in varying degrees of membership instead of being a member of only one cluster. A membership degrees matrix is required by the fuzzy clustering algorithms, i.e., Fuzzy C-means (FCM) (Bezdek, 1981). Initialization of membership degrees for starting FCM and its various variants must satisfy the following constraints:

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

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

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

Membership degrees matrices are usually initialized with the techniques based on random number generating as the function `imembrand` does. In addition to these common techiques, a novel technique using the information from synthetically produced classes over a selected feature is provided in the package '**inaparc**'. The novel technique which is implemented in `figen` may contribute to the fast convergence of the clustering algorithms when compared to the random sampling based techniques. The package also serves the functions for building hard or crisp membership degrees which can be used for testing purposes.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Bezdek J.C. (1981). Pattern recognition with fuzzy objective function algorithms. Plenum, NY, 256 p. <ISBN:0306406713>

Borgelt, C., (2005). *Prototype-based classification and clustering*. Habilitationsschrift zur Erlangung der Venia legendi fuer Informatik, vorgelegt der Fakultaet fuer Informatik der Otto-von-Guericke-Universitaet Magdeburg, Magdeburg, 22 June 2005. url:https://borgelt.net/habil/pbcc.pdf

Cebeci, Z. (2018). Initialization of Membership Degree Matrix for Fast Convergence of Fuzzy C-Means Clustering", In Proc. of *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, IEEE, Sep. 2018, pp. 1-5., doi:10.1109/IDAP.2018.8620920

Cebeci, Z., Sahin, M. & Cebeci, C. (2018). Data dependent techniques for initialization of cluster prototypes in partitioning cluster analysis. In Proc. of *4th International Conference on Engineering and Natural Science*, Kiev, Ukraine, May 2018. pp. 12-22.

Rokah, L. & Maimon, O. (2005). Clustering methods. In *Data Mining and Knowledge Discovery Handbook (ed. O. Maimon)*, Springer US. pp. 321-352. doi:10.1.1.149.9326

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

Khan, F. (2012). An initial seed selection algorithm for k-means clustering of georeferenced data to improve replicability of cluster assignments for mapping application. *Applied Soft Computing*, 12 (11) : 3698-3700. doi:10.1016/j.asoc.2012.07.021

Celebi, M.E., Kingravi, H.A. & Vela, P.A. (2013). A comparative study of efficient initialization methods for the K-means clustering algorithm, *Expert Systems with Applications*, 40 (1): 200-210. arXiv:https://arxiv.org/pdf/1209.1960.pdf

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, imembones, imembrand, figen, inofrep, inscsf, insdev, is.inaparc, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

---

aldaoud *Initialization of cluster prototypes using Al-Daoud's algorithm*

Description

Initializes the cluster prototypes matrix using the variance-based algorithm proposed by Al-Daoud (Al-Daoud, 2005).

Usage

```
aldaoud(x, k)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer specifying the number of clusters.

Details

At first, the algorithm finds the feature having the greatest variance and sorts the data set on this feature in any order. Then it divides the data set into n/k-length k segments. The medians of the segments are assigned as the protoypes of clusters. Al-Daoud's algorithm is likely to be effective only for data sets in which the variability is mostly on one dimension because it considers only one feature with the highest variance (Celebi et al, 2013).

Value

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

v	a numeric matrix containing the initial cluster prototypes.
sfidx	an integer for the column index of the feature with the highest variance.
ctype	a string for the type of used centroid to determine the cluster prototypes. It is 'med' with this function.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Al-Daoud, M.B. (2005). A new algorithm for cluster initialization, in *Proc. of 2nd World Enformatika Conf.*, pp.74-76.

Celebi, M.E., Kingravi, H.A. & Vela, P.A. (2013). A comparative study of efficient initialization methods for the K-means clustering algorithm, *Expert Systems with Applications*, 40 (1): 200-210. arXiv:https://arxiv.org/pdf/1209.1960.pdf

See Also

ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
res <- aldaoud(iris[,1:4], k=5)
v <- res$v
print(v)
```

`ballhall` *Initialization of cluster prototypes using Ball & Hall's algorithm*

Description

Initializes the prototypes of clusters by using the cluster seeding algorithm which has been proposed by Ball & Hall (1967).

Usage

```
ballhall(x, k, tv)
```

Arguments

`x`	a numeric vector, data frame or matrix.
`k`	an integer specifying the number of clusters.
`tv`	a number to be used as T, a threshold distance value. It is directly input by the user. Also it is possible to compute T with the following options of `tv` argument:

- T is the mean of differences between the consecutive pairs of objects with the option '`cd1`'.
- T is the minimum of differences between the consecutive pairs of objects with the option '`cd2`'.
- T is the mean of Euclidean distances between the consecutive pairs of objects divided into k with the option '`md`'. This is the default if `tv` is not supplied by the user.
- T is the range of maximum and minimum of Euclidean distances between the consecutive pairs of objects divided into k with the option '`mm`'.

Details

In the Ball and Hall's algorithm (Ball & Hall, 1967), the center of gravity of data is assigned as the prototype of first cluster. It then passes the data objects in arbitrary order and takes an object as the next prototype if it is T units far from the previously selected prototypes. The purpose of using T, the distance threshold, is to make the cluster protoypes at least T units away from each other. Ball & Hall's method may be sensitive to the order of data, and moreover, deciding for an appropriate value of T is is also difficult (Celebi et al, 2013). As the solutions to this problem, the function `ballhall` in this package computes a T value using some distance measures, if it is not specified by the user (for details, see the section 'Arguments' above.)

Value

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

v	a numeric matrix containing the initial cluster prototypes.
ctype	a string for the type of used centroid. It is 'obj' with this function because the created cluster prototypes matrix contains the selected objects.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Ball, G.H. & Hall, D.J. (1967). A clustering technique for summarizing multivariate data, *Systems Res. & Behavioral Sci.*, 12 (2): 153-155.

Celebi, M.E., Kingravi, H.A. & Vela, P.A. (2013). A comparative study of efficient initialization methods for the K-means clustering algorithm, *Expert Systems with Applications*, 40 (1): 200-210. arXiv:https://arxiv.org/pdf/1209.1960.pdf

See Also

aldaoud, crsamp, firstk, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp,

Examples

```
data(iris)
# Run with a user described threshold value
v1 <- ballhall(x=iris[,1:4], k=5, tv=0.6)$v
print(v1)

# Run with the internally computed default threshold value
v2 <- ballhall(x=iris[,1:4], k=5)$v
print(v2)
```

---

crsamp *Initialization of cluster prototypes using the centers of random samples*

Description

Initializes the cluster prototypes matrix using the centers of *r* data objects. The options for centers are mean and median of the sampled objects in addition to the objects nearest to the mean of the sampled objects.

Usage

```
crsamp(x, k, r, ctype)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer specifying the number of clusters.
r	an integer for the number of objects to be sampled from the data set. If missing, the default value is 2.
ctype	a string for the type of centroids to be computed. The options are 'avg' for average, 'med' for median or 'obj' for the object nearest to the average. The default is 'obj'.

Details

Instead of sampling only one random object as the function rsamp does, the function arsamp randomly samples *r* data objects, and then computes the average and median of these sampled objects. The nearest data object to the mean of sampled objects is also found. If ctype is 'avg' the mean of the sampled *r* objects is assigned as the prototype of first cluster. When ctype is 'med' the median of the sampled *r* objects is assigned as the prototype of first cluster. If the ctype is 'obj', the nearest object to the mean of sampled *r* objects is assigned as the the prototype of first cluster. The same process is repeated for all of the remaining clusters. The logic behind this novel technique is to avoid to select the outliers in the data set which may occur with random sampling for only one object.

Value

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

v	a numeric matrix containing the initial cluster prototypes.
ctype	a string for the type of used centroid to build the cluster prototypes matrix.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

See Also

aldaoud, ballhall, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
# Prototypes are the objects nearest to the mean of
# five randomly sampled objects for each cluster
res <- crsamp(iris[,1:4], k=5, r=5, ctype="obj")
v <- res$v
print(v)
```

`figen` *Initialization of membership degrees over class range of a selected feature*

Description

Initializes the membership degrees matrix by using the class range of the coefficient of variation of a selected feature in the data set being processed.

Usage

```
figen(x, k, mtype, sfidx)
```

Arguments

`x`	an data.frame or matrix for the data set.
`k`	an integer for the number of clusters.
`mtype`	a character representing the type of membership degrees to be generated. The default type is 'f' for generating fuzzy membership matrix. Use 'h' for creating an hard (crisp) membership matrix.
`sfidx`	an integer for the column index of a selected feature. The default is the column index of a feature whose coefficient of variation is the maximum among all features in the data set.

Details

The function `figen` generates a numeric matrix containing the fuzzy initial membership degrees.

Value

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

`u`	a numeric matrix containing the initial membership degrees.
`sfidx`	an integer for the column index of the selected feature, which used for random sampling.
`call`	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Cebeci, Z. (2018), "Initialization of Membership Degree Matrix for Fast Convergence of Fuzzy C-Means Clustering", In Proc. of *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, IEEE, Sep. 2018, pp. 1-5., doi: 10.1109/IDAP.2018.8620920

See Also

imembrand, imembones

Examples

```
data(iris)

# Generate a fuzzy membership matrix using the 1st feature
u <- figen(iris[,1:4], k=5, sfidx=1)$u
head(u)
tail(u)

# Generate a fuzzy membership matrix using the internally determined feature
res <- figen(iris[,1:4], k=5)
u <- res$u
head(u)
tail(u)
```

---

firstk *Initialization of cluster prototypes using the first k objects*

Description

Initializes the cluster prototypes matrix using the first k objects at the top of data set.

Usage

```
firstk(x, k)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer specifying the number of clusters.

Details

The technique so-called the *first method of MacQueen* (MacQueen, 1967) that simply selects the first k objects as the initial centroids. It is sensitive to the order of data (Celebi et al, 2013). If the data set is already sorted in any order it may result with no good initial prototypes because the data objects are close to each other in a sorted data set. Therefore, shuffling of the data set as a pre-processing step may improve the quality with this initialization technique.

Value

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

`v`	a numeric matrix containing the initial cluster prototypes.
`ctype`	a string representing the type of used centroid to build prototype matrix. Its value is 'obj' with this function because it returns the selected objects.
`call`	a string containing the matched function call that generates the object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

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. url:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.308.8619&rep=rep1&type=pdf

Celebi, M.E., Kingravi, H.A. & Vela, P.A. (2013). A comparative study of efficient initialization methods for the K-means clustering algorithm, *Expert Systems with Applications*, 40 (1): 200-210. arXiv:https://arxiv.org/pdf/1209.1960.pdf

See Also

`aldaoud`, `ballhall`, `crsamp`, `forgy`, `hartiganwong`, `inofrep`, `inscsf`, `insdev`, `kkz`, `kmpp`, `ksegments`, `ksteps`, `lastk`, `lhsmaximin`, `lhsrandom`, `maximin`, `mscseek`, `rsamp`, `rsegment`, `scseek`, `scseek2`, `spaeth`, `ssamp`, `topbottom`, `uniquek`, `ursamp`

Examples

```
data(iris)
res <- firstk(x=iris[,1:4], k=5)
v <- res$v
print(v)
```

forgy *Initialization of cluster prototypes using Forgy's algorithm*

Description

Initializes the cluster prototypes using the centers that are calculated with Forgy's algorithm (Forgy, 1965), which is the earliest algorithm for seeding the clusters in the standard K-means clustering.

Usage

```
forgy(x, k)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer specifying the number of clusters.

Details

In this algorithm, each object in the data set is randomly assigned to one of k clusters, and then the mean of the objects assigned to the clusters are used as the initial cluster prototypes. The algorithm lacks of theoretical basis, and the clusters generated randomly may have no internal homogeneity (Celebi et al, 2013).

Value

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

v	a numeric matrix containing the initial cluster prototypes.
ctype	a string representing the type of centroid, which used to build prototype matrix. Its value is 'avg' with this function because the cluster prototypes are the averages of sampled objects for each cluster.
call	a string containing the matched function call that generates the object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Forgy, E.W. (1965). Cluster analysis of multivariate data: Efficiency vs interpretability of classification, *Biometrics*, 21 (3) : 768-769.

Celebi, M.E., Kingravi, H.A. & Vela, P.A. (2013). A comparative study of efficient initialization methods for the K-means clustering algorithm, *Expert Systems with Applications*, 40 (1): 200-210. arXiv:https://arxiv.org/pdf/1209.1960.pdf

See Also

aldaoud, ballhall, crsamp, firstk, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
res <- forgy(iris[,1:4], k=5)
v <- res$v
print(v)
```

`get.algorithms` *Get the names of algorithms in 'inaparc'*

Description

Gets the names of initialization algorithms which are available in the package '**inaparc**'.

Usage

```
get.algorithms(atype="prototype")
```

Arguments

`atype`	an string for the type of algorithms. The default value is '`prototype`' for the names of algorithms for initialization of cluster prototypes. Use '`membership`' for the names of algorithms for initialization of hard and fuzzy membership degrees.

Value

a vector containing the names of algorithms.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

See Also

inaparc-package

Examples

```
get.algorithms(atype="prototype")
get.algorithms(atype="membership")
```

`hartiganwong` *Initialization of cluster prototypes using Hartigan-Wong's algorithm*

Description

Initializes the cluster prototypes matrix using the Hartigan-Wong's algorithm (Hartigan & Wong, 1979).

Usage

```
hartiganwong(x, k)
```

Arguments

`x`	a numeric vector, data frame or matrix.
`k`	an integer specifying the number of clusters.

Details

Firstly, the algorithm computes the center of gravity of data and the distances of data objects to this center. Then, it sorts the data set in any order of the computed distances. The prototypes of *k* clusters are determined by using the formula $(1 + (i - 1)n/k))$, where *i* and *n* stand for the index of a cluster and the number of data rows, respectively. This algorithm leads to increase in the computational cost due to complexity of sorting, which is $O(n\ log(n))$ (Celebi et al, 2013).

Value

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

`v`	a numeric matrix containing the initial cluster prototypes.
`ctype`	a string for the type of used centroid to determine the cluster prototypes. It is 'obj' with this function because the generated prototype matrix contains the selected objects.
`call`	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Hartigan, J.A. & Wong, W.A., (1979). Algorithm AS 136: A K-means clustering algorithm, *J of the Royal Statistical Society*, C 28 (1): 100-108.

Celebi, M.E., Kingravi, H.A. & Vela, P.A. (2013). A comparative study of efficient initialization methods for the K-means clustering algorithm, *Expert Systems with Applications*, 40 (1): 200-210. arXiv:https://arxiv.org/pdf/1209.1960.pdf

See Also

aldaoud, ballhall, crsamp, firstk, forgy, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
res <- hartiganwong(iris[,1:4], k=5)
v <- res$v
print(v)
```

---

imembones *Initialization of a crisp membership matrix using a selected cluster*

Description

Initializes a crisp membership degrees matrix which is used to start a partitional clustering algorithm.

Usage

```
imembones(n, k, mtype, numseed)
```

Arguments

n	an integer for the number of objects in the data set.
k	an integer for the number of clusters.
mtype	a string representing the type of crisp initialization for a selected cluster. The default is 'hrc'. The alternatives are 'hfc' in which all objects are assumed as the member of the first cluster, and 'hlc' in which all objects are assumed as the member of the last cluster.
numseed	a number to be used for the seed of RNG.

Details

The function imembones generates a numeric membership degrees matrix containing the crisp initial values for a selected cluster.

Value

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

u	a numeric matrix containing the crisp initial membership degrees of the objects to k clusters.
sfidx	an integer for the column index of the selected feature, which used for random sampling.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

See Also

imembrand, figen

Examples

```
# Generate membership degrees matrix whose last column contains crisp
# membership degrees
u <- imembones(n=10, k=5, mtype="hlc")$u
head(u)
tail(u)

# Generate membership degrees matrix using a seed number
u <- imembones(n=10, k=5, mtype="hrc", numseed=123)$u
head(u)
tail(u)
```

---

imembrand *Initialization of membership matrix using simple random sampling*

Description

Initializes the membership degrees matrix which is used to start a fuzzy and possibilistic partitioning clustering algorithm.

Usage

```
imembrand(n, k, mtype, numseed)
```

Arguments

n	an integer for the number of objects in the data set.
k	an integer for the number of clusters.
mtype	a string for any of three random initialization methods. The default method is 'f1' for fuzzy memberships. The options are 'f2' and 'f3' for fuzzy memberships and 'h' for hard (crisp) memberships.
numseed	a number to be used for the seed of RNG.

Details

The function imembrand generates a numeric matrix containing the initial membership degrees by using simple random sampling technique.

Value

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

u	a numeric matrix containing the crisp initial membership degrees of n objects to k clusters.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

See Also

figen, imembones

Examples

```
data(iris)
n <- dim(iris)[1]

# Generate a fuzzy membership degrees matrix using default values
u <- imembrand(n=n, k=5)$u
head(u)
tail(u)

# Generate a fuzzy membership degrees matrix using the method 3
u <- imembrand(n=n, k=5, mtype="f3", numseed=123)$u
head(u)
tail(u)

# Generate a crisp membership degrees matrix
u <- imembrand(n=n, k=5, mtype="h")$u
head(u)
tail(u)
```

---

inofrep *Initialization of cluster prototypes using Inofrep algorithm*

Description

Initializes cluster prototypes using Inofrep which is a novel prototypes initialization algorithm using the peaks of frequency polygon of a selected feature.

Usage

```
inofrep(x, k, sfidx, sfpm, binrule, nbins, tcmethod, tc)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer for the number of clusters.
sfidx	an integer specifying the column index of a selected feature which is used for determination of protoypes. If missing, it is internally determined by comparing the peak counts of all features in the data set, and the feature having maximum number of peaks is used as the selected feature.
sfpm	a numeric two-column matrix containing the middle values and frequencies of the peaks of the selected feature, respectively.
binrule	a string containing the name of binning rule to generate the classes of frequency polygons of features in the data set. If missing, 'sqr' rule is used as the default, and square root of the row number of data matrix is assigned as the number of classes to generate frequency polygons.
nbins	an integer for the number of classes of frequency polygons of features in the data set. It should be given if the binning rule 'usr' is selected as the threshold computing method. If missing, it is internally assigned by the binning rule given as the input.
tcmethod	a string representing the threshold value computing method which is used to remove small peaks and empty classes. If missing, the defult method is 'min' which assigns the threshold value to the minimum frequency of the classes in a frequency polygon.
tc	a numeric threshold value for removing the small peaks and empty classes. If missing, it is assigned internally by the used threshold computing method if it is described or 1 if it is not described.

Details

Inofrep, *initialization on the frequency polygon* of a selected feature is a data dependent semi-deterministic initialization algorithm to improve the computational efficiency in prototype-based hard and fuzzy clustering. In the descriptive statistics, frequency polygons serve the structural information about the data. Since a cluster is a dense region of objects that is surrounded by a region of low density (Tan et al, 2006), the peaks of a frequency polygon occur in the center of dense regions of data (Aitnouri et al, 1999). Based on this assumption, the algorithm Inofrep uses that the peak values in frequency polygons as the estimates of central tendency locations or the centres of different dense regions, namely the clusters in the data set. Thus, the peak values can be used as the prototypes of clusters.

Value

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

v	a numeric matrix containing the initial cluster prototypes.
sfidx	an integer for the column index of the selected feature, which used for determination of cluster prototypes.
ctype	a string for the type of centroid, which used for assigning the cluster prototypes.
call	a string containing the matched function call that generates this 'inaparc' object.

Note

In order to supply the peak matrices directly, the functions findpolypeaks and rmshoulders of the package '**kpeaks**' can be used.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Aitnouri E.M., Wang, S., Ziou, D., Vaillancourt, J. & Gagnon, L. (1999). An algorithm for determination of the number of modes for pdf estimation of multi-modal histograms, in *Proc. of Vision Interface '99*, Trois-Rivieres, Canada, May 1999, p. 368-374.

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

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
# set 2nd feature as the selected feature
sfidx <- 2

# generate frequency polygon for the selected feature with user-defined class number
hvals <- kpeaks::genpolygon(iris[,sfidx], binrule="usr", nbins=20)

# Call findpolypeaks for calculating the peaks matrix for the peaks of frequency polygon
resfpp <- kpeaks::findpolypeaks(hvals$mids, hvals$freqs, tcmethod="min")
sfpm <- resfpp$pm

# Call inofrep with the peaks matrix calculated in previous step
v <- inofrep(x=iris[,1:4], k=5, sfidx=sfidx, sfpm=sfpm)$v
print(v)
```

---

inscsf *Initialization cluster prototypes using Inscsf algorithm*

Description

Initializes cluster prototypes with `Inscsf` which is a novel prototype initialization algorithm using a selected central tendency measure of a selected feature. For reducing the computational complexity and increasing the accuracy in initialization, the algorithm works on only one feature which can be selected according to its importance in clustering. Furthermore, with a selection mechanism using the distribution of data the algorithm also automatically decides what type of center measure should be used.

Usage

```
inscsf(x, k, sfidx, ctype)
```

Arguments

`x`	a numeric vector, data frame or matrix.
`k`	an integer specifying the number of clusters.
`sfidx`	an integer specifying the column index of the selected feature. If missing, it is internally determined by comparing the number of unique values for all the features in the data set. The feature having the maximum number of unique values is used as the selected feature.
`ctype`	a string for the type of the selected center. The options are 'avg' for average, 'med' for median or 'mod' for mode. The default value is 'avg'.

Details

The `inscsf` is based on a technique so-called "*initialization using a selected center of a selected feature*". It resembles Ball and Hall's method (Ball and Hall, 1967) for assignment of the first cluster prototype but it differs by the use of two different interval values (R_1 and R_2) instead of using only one fixed threshold value (T) for determining the prototypes of remaining clusters. The technique `inscsf` does not require to sort the data set. R_1 is an interval which is calculated by dividing the distance between the center and maximum of the selected feature (x_f) by half of the number of clusters minus 1.

$$R_1 = \frac{max(x_f) - center(x_f)}{(c-1)/2}$$

Similarly, R_2 is an interval which is calculated by dividing the distance between the maximum and center of the selected feature by half of the number of clusters minus 1.

$$R_2 = \frac{center(x_f) - min(x_f)}{(k-1)/2}$$

These two intervals become equal to each other if the selected feature is normally distributed, and thus, cluster prototypes are located in equidistant positions from each other in the p-dimensional space of n data objects.

Depending on the distribution of selected feature, the mean, median or mode of the selected feature can be used to determine the prototype of first cluster. If the type of center measure is not input by the user, it is internally determined according to the distribution of data. Then, the nearest data

instance to the center of the selected feature is searched on the selected feature column, and assigned as the prototype of first cluster.

$$v_1 = x_i; i = row\ index\ of\ the\ nearest\ data\ object\ to\ center(x_f))$$

The prototype of an even-numbered cluster is determined by adding the value of R_1 times the cluster index minus 1 to the first cluster prototype.

$$v_j = (x_{(i+(j-1)}\ R_1)$$

On the other hand, R_2 is used to calculate the prototypes for the odd-numbered clusters.

$$v_j = (x_{(i+(j-1)}\ R_2)$$

Value

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

`v`	a numeric matrix containing the initial cluster prototypes.
`sfidx`	an integer for the column index of the selected feature.
`ctype`	a string for the type of centroid. It is 'obj' with this function because the prototypes matrix contain contains the selected objects.
`call`	a string containing the matched function call that generates this 'inaparc' object.

Note

The selected feature can be determined in several ways. The feature with highest number of peaks among the others can be also utilized as the selected feature with this function. For determination of it, the function `findpolypeaks` of the package '**kpeaks**' can be used.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Ball, G.H. & Hall, D.J. (1967). A clustering technique for summarizing multivariate data, *Systems Res. & Behavioral Sci.*, 12 (2): 153-155.

Cebeci, Z., Sahin, M. & Cebeci, C. (2018). Data dependent techniques for initialization of cluster prototypes in partitioning cluster analysis. In Proc. of *4th International Conference on Engineering and Natural Science*, Kiev, Ukraine, May 2018. pp. 12-22.

See Also

`aldaoud`, `ballhall`, `crsamp`, `firstk`, `forgy`, `hartiganwong`, `inofrep`, `insdev`, `kkz`, `kmpp`, `ksegments`, `ksteps`, `lastk`, `lhsmaximin`, `lhsrandom`, `maximin`, `mscseek`, `rsamp`, `rsegment`, `scseek`, `scseek2`, `spaeth`, `ssamp`, `topbottom`, `uniquek`, `ursamp`

Examples

```
data(iris)
# Use the 4th feature as the selected feature
v1 <- inscsf(x=iris[,1:4], k=5, sfidx=4)$v
print(v1)

# Use the internally selected feature
v2 <- inscsf(x=iris[,1:4], k=5)$v
print(v2)
```

`insdev` *Initialization of cluster prototypes using Insdev algorithm*

Description

`Insdev` is a novel algorithm that initializes the cluster prototypes by using the standard deviation of a selected feature. The selected feature is the most important feature in regard of variation. For this purpose the coefficients of variation of the features are compared, and then the feature with highest coefficient of variation is selected for further processes.

Usage

```
insdev(x, k, sfidx)
```

Arguments

`x`	a numeric vector, data frame or matrix.
`k`	an integer specifying the number of clusters.
`sfidx`	an integer specifying the column index of the selected feature. Here, in this function we use the feature with high variability as the selected feature because it dominates the clustering results (Khan, 2912). If missing, so it is internally determined by comparing the coefficents of variation for all the features in the data set. The feature having the maximum coefficient of variation is used as the selected feature.

Details

At first the algorithm computes the mean of the selected feature ($\bar{x_s}$) and then seeks the object whose distance is minimum to $\bar{x_s}$ as the prototype of first cluster. The prototypes of remaining clusters are determined by using a stepping range (*R*), computed from the standard deviation of selected feature with the formula $R = 1/2\sigma_{x_s}/k$. The prototype of second cluster is the object whose distance is minimum to $\bar{x_s} + (i-1)\ R$, where *i* is the cluster index. The prototype of third cluster is the object whose distance is minimum to $\bar{x_s} - i\ R$ in the opposite direction to previous prototype. The prototypes remaining clusters are cyclically determined in similar way.

Since it produces the same prototypes in each run of it, `insdev` is a deterministic algorithm. Therefore, this characteristic of the algorithm provides replicability in initialization procedure.

Value

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

v	a numeric matrix containing the initial cluster prototypes.
sfidx	an integer for the column index of the selected feature, used in the calculations.
ctype	a string representing the type of centroid, which used to build prototype matrix. Its value is 'obj' with this function because the cluster prototypes are the objects sampled from the data set.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Khan, F. (2012). An initial seed selection algorithm for k-means clustering of georeferenced data to improve replicability of cluster assignments for mapping application. *Applied Soft Computing*, 12 (11) : 3698-3700. doi:10.1016/j.asoc.2012.07.021

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
res <- insdev(x=iris[,1:4], k=5)
v <- res$v
print(v)
```

---

is.inaparc *Checking the object class for 'inaparc'*

Description

Checks whether the given object is an instance of the inaparc class.

Usage

```
is.inaparc(x)
```

Arguments

x	an object to check.

Value

TRUE if x is a valid inaparc object and FALSE for the other type of object classes.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

See Also

crsamp, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsrandom, lhsmaximin, mscseek, rsamp, rsegment, scseek, scseek2, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
res <- firstk(x=iris[,1:4], k=5)
is.inaparc(res)

x <- c(1,5,8)
is.inaparc(x)
```

---

kkz *Initialization of cluster prototypes using KKZ algorithm*

Description

Initializes the cluster prototypes matrix using 'KKZ' algorithm proposed by Katsavounidis et al (1994).

Usage

```
kkz(x, k)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer specifying the number of clusters.

Details

The function kkz is an implementation of the cluster seeding algorithm which has been proposed by Katsavounidis et al (1994). As the first cluster prototype, the algorithm so-called 'KKZ' selects one data object on the edges of data. It is the object having the greatest squared Euclidean norm in the function kkz. The second cluster prototype is the farthest object from the previously selected object. After assignment of the prototypes of first two clusters, the distances of all of the remaining objects to them are computed. The object which is the farthest from its nearest prototype is assigned as the third prototype. The above process is repeated for selecting the prototypes of remaining clusters in the same way. The algorithm 'KKZ' is considered to be sensitive to the outliers in the data set.

Value

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

v	a numeric matrix containing the initial cluster prototypes.
ctype	a string representing the type of centroid, which used to build prototype matrix. Its value is 'obj' with this function because the cluster prototypes are the selected objects.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Katsavounidis, I., Kuo, C. & Zhang, Z. (1994). A new initialization technique for generalized Lloyd iteration. *IEEE Signal Processing Letters*, 1 (10): 144-146. url:https://www.semanticscholar.org/paper/A-new-initialization-technique-for-generalized-Katsavounidis-Kuo/0103d3599757c77f6f3cbe3c
p2df

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
res <- kkz(x=iris[,1:4], k=5)
v <- res$v
print(v)
```

---

kmpp *Initialization of cluster prototypes using K-means++ algorithm*

Description

Initializes the cluster prototypes matrix by using K-means++ algorithm which has been proposed by Arthur and Vassilvitskii (2007).

Usage

```
kmpp(x, k)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer specifying the number of clusters.

Details

K-means++ (Arthur & Vassilvitskii, 2007) is usually reported as an efficient approximation algorithm in overcoming the poor clustering problem with the standard K-means algorithm. K-means++ is an algorithm that merges MacQueen's second method with the 'Maximin' method to initialize the cluster prototypes (Ji et al, 2015). K-means++ initializes the cluster centroids by finding the data objects that are farther away from each other in a probabilistic manner. In K-means++, the first cluster protoype (center) is randomly assigned. The prototypes of remaining clusters are determined with a probability of $md(x')^2 / \sum_{k=1}^{n} md(x_k)^2$, where $md(x)$ is the minimum distance between a data object and the previously computed prototypes.

The function kmpp is an implementation of the initialization algorithm of K-means++ that is based on the code'k-meansp2.R', authored by M. Sugiyama. It needs less execution time due to its vectorized distance computations.

Value

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

v	a numeric matrix containing the initial cluster prototypes.
ctype	a string representing the type of centroid, which used to build prototype matrix. Its value is 'obj' with this function because the cluster prototypes are the objects selected by the algorithm.
call	a string containing the matched function call that generates this sQuoteinaparc 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. url:http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf

M. Sugiyama, 'mahito-sugiyama/k-meansp2.R'. url:https://gist.github.com/mahito-sugiyama/ef54a3b17fff4629f106

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
res <- kmpp(x=iris[,1:4], k=5)
v <- res$v
print(v)
```

---

ksegments *Initialization of cluster prototypes using the centers of k segments*

Description

Initializes the cluster prototypes matrix using the centers of k segments (subsets) of the data set.

Usage

```
ksegments(x, k, ctype)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer specifying the number of clusters.
ctype	a string for the type of centroid. The options are 'avg' for average and 'med' for median of the objects in the segments. The default is 'avg'.

Details

The first segment consists of the first n/k objects. The second segment consists of n/k objects starting from the n/k+1-*th* object. The process is repeated for k segments. The centers of k segments are assigned as the cluster prototypes.

Value

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

v	a numeric matrix containing the initial cluster prototypes.
ctype	a string representing the type of centroid. Its value is 'avg' for average or 'med' for median of the objects in the segments.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

See Also

crsamp, firstk, inofrep, inscsf, insdev, kkz, kmpp, ksteps, lastk, lhsrandom, lhsmaximin, mscseek, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)

# Generate the prototypes matrix using the means of segments
res <- ksegments(x=iris[,1:4], k=5, ctype="avg")
v <- res$v
print(v)
```

---

ksteps *Initialization of cluster prototypes using the centers of k blocks*

Description

Initializes the cluster prototypes matrix using the centers of objects in k blocks that are generated with a kind of systematic sampling method as described in the section 'Details'.

Usage

```
ksteps(x, k, ctype)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer for the number of clusters.
ctype	a string for the type of centroid. The options are 'avg' for average and 'med' for median of the objects in the blocks. The default is 'avg'.

Details

The algorithm ksteps is similar to ksegments but it differs for the selection of the members of the segments or blocks. The objects whose row indexes are 1, 1+k, 1+2k, ... are assigned to the first segment, and then the objects whose row indexes are 2, 2+k, 2+2k, ... to the second block. In this way, k blocks of the objects are formed. The centers of these k blocks are assigned as the cluster prototypes.

Value

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

v	a numeric matrix containing the initial cluster prototypes.
ctype	a string representing the type of centroid, which used to build prototype matrix.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
res <- ksteps(x=iris[,1:4], k=5)
v <- res$v
print(v)
```

`lastk` *Initialization of cluster prototypes using the last k objects*

Description

Initializes the cluster prototypes matrix using the last k objects at the bottom of data set.

Usage

```
lastk(x, k)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer specifying the number of clusters.

Details

The function `lastk` simply uses the last k objects as the protoypes of clusters. If the data is already sorted in any order it may result with no good initial prototypes because the objects be close to each other in a sorted matrix. Therefore, shuffling of the data set as a pre-processing step may improve the quality with this prototyping technique.

Value

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

v	a numeric matrix containing the initial cluster prototypes.
ctype	a string representing the type of centroid, which used to build prototype matrix. Its value is 'obj' with this function because the cluster prototype matrix contains the objects.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
res <- lastk(x=iris[,1:4], k=5)
v <- res$v
print(v)
```

`lhsmaximin` *Initialization of cluster prototypes using Maximin LHS*

Description

Initializes the cluster prototypes matrix using the Maximin version of Latin Hypercube Sampling (LHS). A square grid containing possible sample points is a Latin Square (LS) if there is only one sample in each row and each column. LHS is a generalized version of LS, which has been developed to generate a distribution of collections of parameter values from a multidimensional distribution. LHS generates more efficient estimates of desired parameters than simple Monte Carlo sampling (Carnell, 2016).

Usage

```
lhsmaximin(x, k, ncp)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer specifying the number of clusters.
ncp	an integer determining the number of candidate points used in the search by maximin LHS algorithm.

Details

LHS aims at initial cluster centers whose coordinates are well spread out in the individual dimensions (Borgelt, 2005). It is the generalization of Latin Square for an arbitrary number of dimensions (features). When sampling a function of p features, the range of each feature is divided into k equally probable intervals. k samples are then drawn such that a Latin Hypercube is created.

The current version of the function `lhsmaximin` in this package uses the results from the `maximinLHS` function from the '**lhs**' library created by Carnell (2016). Once the uniform samples are created by the `maximinLHS`, they are transformed to normal distribution samples by using the quantile functions. But all the features in the data set may not be normally distributed, instead they may fit to different distributions. In such cases, the transformation for any feature should be specisific to its distribution. Determination of the distribution types of features is planned in the future versions of the function '`lhsmaximin`'.

Value

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

v	a numeric matrix containing the initial cluster prototypes.
ctype	a string for the type of used centroid to determine the cluster prototypes. It is 'obj' with this function.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Borgelt, C., (2005). *Prototype-based classification and clustering*. Habilitationsschrift zur Erlangung der Venia legendi fuer Informatik, vorgelegt der Fakultaet fuer Informatik der Otto-von-Guericke-Universitaet Magdeburg, Magdeburg, 22 June 2005. url:https://borgelt.net/habil/pbcc.pdf

Carnell, R., (2016). lhs: Latin Hypercube Samples. R package version 0.14. https://CRAN.R-project.org/package=lhs

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
res <- lhsmaximin(iris[,1:4], k=5)
v <- res$v
print(v)
```

---

lhsrandom *Initialization of cluster prototypes using random LHS*

Description

Initializes the cluster prototypes matrix using the random version of Latin Hypercube Sampling (LHS). A square grid containing possible sample points is a Latin Square (LS) if there is only one sample in each row and each column. LHS is a generalized version of LS, which has been developed to generate a distribution of collections of parameter values from a multidimensional distribution. LHS generates more efficient estimates of desired parameters than simple Monte Carlo sampling (Carnell, 2016).

Usage

```
lhsrandom(x, k)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer for specifying the number of clusters.

Details

LHS aims at initial cluster centers whose coordinates are well spread out in the individual dimensions (Borgelt, 2005). LHS is the generalization of Latin Square for an arbitrary number of dimensions (features). When sampling a function of p features, the range of each feature is divided into k equally probable intervals. k samples are then drawn such that a Latin Hypercube is created.

The current version of the function `lhsrandom` in this package uses the results from the `randomLHS` function from the R package '**lhs**' (Carnell, 2016), which contains several variants of LHS. Once the uniform samples are created by the `randomLHS`, they are transformed to normal distributed samples by using the quantile functions. But all the features in the data set may not be normally distributed, instead they may have the different type of distributions. In such cases, the transformation of any feature should be specific to its distribution. Determination of the distribution types of features is planned in the future versions of the function '`lhsrandom`'.

Value

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

v	a numeric matrix containing the initial cluster prototypes.
ctype	a string for the type of used centroid to determine the cluster prototypes. It is 'obj' with this function.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Borgelt, C., (2005). *Prototype-based classification and clustering*. Habilitationsschrift zur Erlangung der Venia legendi fuer Informatik, vorgelegt der Fakultaet fuer Informatik der Otto-von-Guericke-Universitaet Magdeburg, Magdeburg, 22 June 2005. url:https://borgelt.net/habil/pbcc.pdf

Carnell, R., (2016). lhs: Latin Hypercube Samples. R package version 0.14. https://CRAN.R-project.org/package=lhs

See Also

`aldaoud`, `ballhall`, `crsamp`, `firstk`, `forgy`, `hartiganwong`, `inofrep`, `inscsf`, `insdev`, `kkz`, `kmpp`, `ksegments`, `ksteps`, `lastk`, `lhsmaximin`, `maximin`, `mscseek`, `rsamp`, `rsegment`, `scseek`, `scseek2`, `spaeth`, `ssamp`, `topbottom`, `uniquek`, `ursamp`

Examples

```
data(iris)
res <- lhsrandom(iris[,1:4], k=5)
v <- res$v
print(v)
```

`maximin` *Initialization of cluster prototypes using Maximin algorithm*

Description

Initializes the cluster prototypes matrix by using the Maximin algorithm.

Usage

```
maximin(x, k)
```

Arguments

`x`	a numeric vector, data frame or matrix.
`k`	an integer for the number of clusters.

Details

The main idea of the *Maximin* algorithm is to isolate the cluster prototypes that are farthest apart (Philpot, 2001). The algorithm randomly samples one data object from the data set and assigns it as the first cluster prototype. The prototype of second cluster is determined as the data object which is farthest from the first prototype. Then, the remaining part of data set is scanned for the data objects whose distances are minimum to the previously selected prototypes. The object having the maximum of minimum distances is assigned the prototype of third cluster. The same procedure is repeated for determining the prototypes of other clusters (Spaeth, 1997; Gonzales, 1985; Duda et al, 2000, Celebi et al, 2013).

The algorithm generally works well with circular shaped clusters whose radius are smaller than the separation between clusters. However, it is very sensitive to the order of object in data sets. Also it is computationally expensive because each time once a new cluster prototype is selected, the distances must be computed for every object from every cluster prototype (Philpot, 2001). In order to contribute to the solutions of this problem, the current implementation of `maximin` includes a simple control that if an object has the minimum distance of zero, the seeking procedure is no more continued to compute the distances for the remaining objects. This control may speed the algorithm up with the `maximin` function in this package.

Value

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

`v`	a numeric matrix of the initial cluster prototypes.

ctype	a string representing the type of centroid, which used to build prototype matrix. Its value is 'obj' with this function because the cluster prototype matrix contains the objects.
call	a string containing the matched function call that generates the object 'inaparc'.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Spaeth, H. (1977). Computational experiences with the exchange method: Applied to four commonly used partitioning cluster analysis criteria, *European Journal of Operational Research* 1 (1): 23-31. doi:10.1016/S03772217(77)810059

Gonzalez, T. (1985), Clustering to minimize the maximum intercluster distance, *Theoretical Computer Science* 38 (2-3): 293-306. url:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.366.8183&rep=rep1&type=pdf

Duda, R.O., Hart, P.E. & Stork, D.G. (2000). *Pattern Classification*, Wiley-Interscience. <ISBN:978-0-471-05669-0>

Celebi, M.E., Kingravi, H.A. & Vela, P.A. (2013). A comparative study of efficient initialization methods for the K-means clustering algorithm, *Expert Systems with Applications*, 40 (1): 200-210. arXiv:https://arxiv.org/pdf/1209.1960.pdf

Philpot, W. (2001). Topic 8: Clustering/Unsupervised Classification in *Lecture Notes, CEE 615: Digital Image Processing - Jan 2001, Cornell Univ.*, url:https://www-users.cse.umn.edu/~kumar/dmbook/ch8.pdf

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, mscseek, rsamp, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
res <-maximin(x=iris[,1:4], k=5)
v <- res$v
print(v)
```

---

mscseek *Initialization of cluster prototypes using the modified SCS algorithm*

Description

Initializes the cluster prototypes matrix using a modified version of the Simple Cluster Seeking (SCS) algorithm proposed by Tou & Gonzales(1974). While SCS uses a fixed threshold distance value T for selecting all candidates of clusters, the modified SCS recomputes T with the average Euclidean distances between the previously determined prototypes. This adjustment makes possible to select more cluster prototypes when compared to SCS.

Usage

```
mscseek(x, k, tv)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer for the number of clusters.
tv	a number to be used as the threshold distance which is directly input by the user. Also it is possible to compute T, a threshold distance value with the following options of tv argument: • T is the mean of differences between the consecutive pairs of objects with the option 'cd1'. • T is the minimum of differences between the consecutive pairs of objects with the option 'cd2'. • T is the mean of Euclidean distances between the consecutive pairs of objects divided into k with the option 'md'. This is the default if tv is not supplied by the user. • T is the range of maximum and minimum of Euclidean distances between the consecutive pairs of objects divided into k with the option 'mm'.

Details

This is a modification of the Simple Cluster Seeking (SCS) algorithm (Tou & Gonzalez, 1974). The algorithm selects the first object in the data set as the prototype of the first cluster. Then, next object whose distance to the first prototype is greater than a threshold distance value is searched and assigned as the second cluster prototype. Instead of using a fixed the T, threshold distance value as SCS does, the modified SCS recomputes the T by the average Euclidean distances between the previously determined prototypes of clusters. The next object whose distance to the previously selected object is greater than the adjusted T is searched and assigned as the third cluster prototype. The selection process is repeated for the remaining clusters in similar way. The method is sensitive to the order of the data, it may not yield good initializations with the ordered data.

Value

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

v	a numeric matrix of the initial cluster prototypes.
ctype	a string representing the type of centroid, which used to build prototype matrix. Its value is 'obj' with this function because the cluster prototype matrix contains the objects.

call	a string containing the matched function call that generates the object 'inaparc'.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Tou, J.T. & Gonzalez, R.C. (1974). *Pattern Recognition Principles*. Addison-Wesley, Reading, MA. <ISBN:9780201075861>

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, rsamp, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
# Run with the threshold value of 0.1
res <- mscseek(x=iris[,1:4], k=5, tv=0.1)
v1 <- res$v
print(v1)

# Run with the internally computed default threshold value
res <- mscseek(x=iris[,1:4], k=5)
v2 <- res$v
print(v2)
```

---

rsamp *Initialization of cluster prototypes using simple random sampling*

Description

Initializes the cluster prototypes matrix using the randomly selected k objects from the data set.

Usage

```
rsamp(x, k)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer for the number of clusters.

Details

The function rsamp generates a protoype matrix using the k objects which are randomly sampled from the data set without replacement. Simple random sampling (SRS), also so-called the *second method of MacQueen* in the clustering context, assumes that cluster areas have a high density; in consequence, the good candidates of the cluster prototypes can be sampled from these dense regions of data with a higher chance (Celebi et al, 2013). SRS is probably the most common approach to initialize prototype matrices. So, it can be seen a *de facto standard* because it has been widely applied with the basic K-means algorithm for the years. Since SRS has no rule to avoid to select the outliers or the objects close to each other, it may result with no good initializations. Before initialization of SRS, multivariate outliers removal on the data set as a data pre-processing step may be helpful to avoid for selection of the outliers, but increases the computational cost.

Value

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

v	a numeric matrix containing the initial cluster prototypes.
ctype	a string representing the type of centroid, which used to build prototype matrix. Its value is 'obj' with this function because it samples the objects only.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

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

Celebi, M.E., Kingravi, H.A. & Vela, P.A. (2013). A comparative study of efficient initialization methods for the K-means clustering algorithm, *Expert Systems with Applications*, 40 (1): 200-210. arXiv:https://arxiv.org/pdf/1209.1960.pdf

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
res <- rsamp(x=iris[,1:4], k=5)
v <- res$v
print(v)
```

`rsegment` *Initialization of cluster prototypes using a randomly selected segment*

Description

Initializes the cluster prototypes matrix using using a k-length segment of data set consists of consecutive objects that starts with a randomly sampled data object.

Usage

```
rsegment(x, k)
```

Arguments

`x`	a numeric vector, data frame or matrix.
`k`	an integer for the number of clusters.

Details

The function `rsegment` randomly samples one data object as the prototype of first cluster, and then it assigns the next *k-1* data objects as the prototype of remaining clusters.

Value

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

`v`	a numeric matrix containing the initial cluster prototypes.
`ctype`	a string representing the type of centroid, which used to build prototype matrix. Its value is 'obj' with this function because the cluster prototype matrix contains the sampled objects.
`call`	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

See Also

`aldaoud`, `ballhall`, `crsamp`, `firstk`, `forgy`, `hartiganwong`, `inofrep`, `inscsf`, `insdev`, `kkz`, `kmpp`, `ksegments`, `ksteps`, `lastk`, `lhsmaximin`, `lhsrandom`, `maximin`, `mscseek`, `rsamp`, `scseek`, `scseek2`, `spaeth`, `ssamp`, `topbottom`, `uniquek`, `ursamp`

Examples

```
data(iris)
res <- rsegment(x=iris[,1:4], k=5)
v <- res$v
print(v)
```

scseek *Initialization of cluster prototypes using SCS algorithm*

Description

Initializes the cluster prototypes matrix with the Simple Cluster Seeking (SCS) algorithm (Tou & Gonzales, 1974).

Usage

```
scseek(x, k, tv)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer for the number of clusters.
tv	a number to be used as the threshold distance which is directly input by the user. Also it is possible to compute T, a threshold distance value with the following options of tv argument:

- T is the mean of differences between the consecutive pairs of objects with the option 'cd1'.
- T is the minimum of differences between the consecutive pairs of objects with the option 'cd2'.
- T is the mean of Euclidean distances between the consecutive pairs of objects divided into k with the option 'md'. This is the default if tv is not supplied by the user.
- T is the range of maximum and minimum of Euclidean distances between the consecutive pairs of objects divided into k with the option 'mm'.

Details

The algorithm Simple Cluster Seeking (SCS) (Tou & Gonzales, 1974) is similar to Ball and Hall's algorithm (Ball & Hall, 1967) with an exception for selection of the first object (Celebi et al, 2013). In SCS, the first object in the data set is selected as the prototype of the first cluster. Then, the next object whose distance to the first prototype is greater than T, a threshold distance value is seeked and assigned as the second cluster prototype, if found. Afterwards, the next object whose distance to already determined prototypes is greater than T is searched and assigned as the third cluster prototype. The selection process is repeated for determining the prototypes of remaining clusters in similar way.

Because SCS is sensitive to the order of the data (Celebi et al, 2013), it may not yield good initializations with the sorted data. On the other hand, the distance between the cluster prototypes can be controlled T, which is an arbitrary number specified by the user. But the problem is that how the user decides on this threshold value. As a solution to this problem in the function scseek, some internally computed distance measures can be used. (See the section'Arguments' above for the available options.)

Value

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

v	a numeric matrix of the initial cluster prototypes.
ctype	a string representing the type of centroid, which used to build prototype matrix. Its value is 'obj' with this function because the cluster prototype matrix contains the objects.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Ball, G.H. & Hall, D.J. (1967). A clustering technique for summarizing multivariate data, *Systems Res. & Behavioral Sci.*, 12 (2): 153-155.

Tou, J.T. & Gonzalez,R.C. (1974). *Pattern Recognition Principles*. Addison-Wesley, Reading, MA. <ISBN:9780201075861>

Celebi, M.E., Kingravi, H.A. & Vela, P.A. (2013). A comparative study of efficient initialization methods for the K-means clustering algorithm, *Expert Systems with Applications*, 40 (1): 200-210. arXiv:https://arxiv.org/pdf/1209.1960.pdf

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek2, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
# Run with the threshold value of 0.5
res <- scseek(x=iris[,1:4], k=5, tv=0.5)
v1 <- res$v
print(v1)

# Run with the internally computed default threshold value
res <- scseek(x=iris[,1:4], k=5)
v2 <- res$v
print(v2)
```

scseek2 *Initialization of cluster prototypes using SCS algorithm over a selected feature*

Description

Initializes the cluster prototypes matrix with the Simple Cluster Seeking (SCS) algorithm (Tou & Gonzales, 1974) over a selected feature.

Usage

```
scseek2(x, k, sfidx, tv)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer for the number of clusters.
sfidx	an integer specifying the column index of the selected feature for random sampling. If missing, it is internally determined by comparing the coefficients of variation of all features in the data set. The feature having the maximum coefficent of variation is used as the selected feature.
tv	a number to be used as the threshold distance which is directly input by the user. Also it is possible to compute T, a threshold distance value with the following options of tv argument: • T is the mean of differences between the consecutive pairs of objects with the option 'cd1'. • T is the minimum of differences between the consecutive pairs of objects with the option 'cd2'. • T is the mean of Euclidean distances between the consecutive pairs of objects divided into k with the option 'md'. This is the default if tv is not supplied by the user. • T is the range of maximum and minimum of Euclidean distances between the consecutive pairs of objects divided into k with the option 'mm'.

Details

The scseek2 is a novel variant of the function scseek based on the Simple Cluster Seeking (SCS) algorithm (Tou & Gonzales, 1974). It differs from SCS that the distances and threshold value are computed over a selected feature having the maximum coefficient of variation, instead of using all the features.

Value

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

v	a numeric matrix of the initial cluster prototypes.

sfidx	an integer for the column index of the selected feature, which used for random sampling.
ctype	a string representing the type of centroid, which used to build prototype matrix. Its value is 'obj' with this function because the cluster prototype matrix contains the sampled objects.
call	a string containing the matched function call that generates this 'proclus' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Tou, J.T. & Gonzalez,R.C. (1974). *Pattern Recognition Principles*. Addison-Wesley, Reading, MA. <ISBN:9780201075861>

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, spaeth, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
# Run over 4th feature with the threshold value of 0.5
res <- scseek2(x=iris[,1:4], k=5, sfidx=4, tv=0.5)
v1 <- res$v
print(v1)

# Run with the internally computed default threshold value
res <- scseek2(x=iris[,1:4], k=5)
v2 <- res$v
print(v2)
```

---

spaeth *Initialization of cluster prototypes using Spaeth's algorithm*

Description

Initializes the cluster prototypes using the centroids that are calculated with Spaeth's algorithm (Spaeth, 1977), which is similar to Forgy's algorithm.

Usage

```
spaeth(x, k)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer specifying the number of clusters.

Details

In this algorithm, each object in the data set is assigned to one of k clusters in cyclical fashion. The j-*th* $(j\epsilon 1, 2, \cdots, n)$ object is assigned to the $(j-1(modk)+1)(j-1(mod\,k)+1)$-*th* cluster. In contrast to Forgy's method, this method is sensitive to order of data (Celebi et al, 2013).

Value

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

v	a numeric matrix containing the initial cluster prototypes.
ctype	a string representing the type of centroid, which used to build prototype matrix. Its value is 'avg' with this function because the cluster prototypes are the averages of sampled objects for each feature.
call	a string containing the matched function call that generates the 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

References

Spaeth, H. (1977). Computational experiences with the exchange method: Applied to four commonly used partitioning cluster analysis criteria, *European J of Operational Rsch.*, 1(1):23-31. doi:10.1016/S03772217(77)810059

Celebi, M.E., Kingravi, H.A. & Vela, P.A. (2013). A comparative study of efficient initialization methods for the K-means clustering algorithm, *Expert Systems with Applications*, 40 (1): 200-210. arXiv:https://arxiv.org/pdf/1209.1960.pdf

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, ssamp, topbottom, uniquek, ursamp

Examples

```
data(iris)
res <- spaeth(iris[,1:4], k=5)
v <- res$v
print(v)
```

`ssamp` *Initialization of cluster prototypes using systematic random sampling*

Description

Initializes the cluster prototypes matrix using the systemically sampled data objects.

Usage

```
ssamp(x, k)
```

Arguments

`x`	a numeric vector, data frame or matrix.
`k`	an integer for the number of clusters.

Details

The function `ssamp` generates a prototype matrix using the sytematic random sampling technique. Since the data objects are enough away from each other with this technique it may provide better initializations than the simple random sampling. The first object is randomly sampled from the top `n/k` objects of data set and assigned as the prototype of first cluster. The prototypes of remaining clusters are the objects whose row indexes are $v_1 + i\ (n/k)$, where *v1* and *i* are the index of first selected object and index of cluster, respectively.

Value

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

`v`	a numeric matrix containing the initial cluster prototypes.
`ctype`	a string representing the type of centroid, which used to build prototype matrix. Its value is 'obj' with this function because the cluster prototype matrix contains the sampled objects.
`call`	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

See Also

`aldaoud`, `ballhall`, `crsamp`, `firstk`, `forgy`, `hartiganwong`, `inofrep`, `inscsf`, `insdev`, `kkz`, `kmpp`, `ksegments`, `ksteps`, `lastk`, `lhsmaximin`, `lhsrandom`, `maximin`, `mscseek`, `rsamp`, `rsegment`, `scseek`, `scseek2`, `spaeth`, `topbottom`, `uniquek`, `ursamp`

Examples

```
data(iris)
res <- ssamp(x=iris[,1:4], k=5)
v <- res$v
print(v)
```

`topbottom` *Initialization of cluster prototypes using the top and bottom objects*

Description

Initializes the cluster prototypes matrix using the alternately selected k objects from the top and bottom of the data set.

Usage

```
topbottom(x, k)
```

Arguments

`x`	a numeric vector, data frame or matrix.
`k`	an integer for the number of clusters.

Details

The function combines the `firstk` and `lastk` techniques. It takes the first object of the data set as the prototype of first cluster, and then the last object as the prototype of second cluster. This rotating assigment process continues until the prototypes of k clusters are assigned.

Value

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

`v`	a numeric matrix containing the initial cluster prototypes.
`ctype`	a string representing the type of centroid, which used to build prototype matrix. Its value is 'obj' with this function because the cluster prototype matrix contains the objects.
`call`	a string containing the matched function call that generates this 'inaparc' object.

Note

If the sorted data set is used, the function `topbottom` may yield better initializations when compared to the functions `firstk` and `lastk`.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, ssamp, spaeth, uniquek, ursamp

Examples

```
data(iris)
res <- topbottom(x=iris[,1:4], k=5)
v <- res$v
print(v)
```

---

uniquek *Initialization of cluster prototypes over the unique values*

Description

Initializes the cluster prototypes matrix using the randomly sampled data objects over the unique values of a selected feature.

Usage

```
uniquek(x, k, sfidx)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer specifying the number of clusters.
sfidx	an integer specifying the column index of the selected feature for random sampling. If missing, it is internally determined by comparing the number of unique values for all the features in the data set. The feature having the maximum number of unique values is used as the selected feature.

Details

The set of unique values of the selected feature is determined, and then k objects were randomly sampled from this set.

Value

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

v	a numeric matrix containing the initial cluster prototypes.
ctype	a string representing the type of centroid, which used to build prototype matrix. Its value is 'obj' with this function because the cluster prototype matrix contains the sampled objects.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, spaeth, ssamp, topbottom, ursamp

Examples

```
data(iris)
# Run with the internally selected feature
res <- uniquek(x=iris[,1:4], k=5)
v <- res$v
print(v)

# Run with the 1st feature
res <- uniquek(x=iris[,1:4], k=5, sfidx=1)
v <- res$v
print(v)
```

---

ursamp *Initialization of cluster prototypes using random sampling on each future*

Description

Initializes the cluster prototypes matrix by using random uniform sampling for each of p features in the data set, independently.

Usage

```
ursamp(x, k)
```

Arguments

x	a numeric vector, data frame or matrix.
k	an integer for the number of clusters.

Details

The ursamp generates the prototypes by binding randomly sampled values for each of p features, independently. In this novel approach proposed by the authors of the package, an object is randomly sampled from data set and the value of first feature is assigned as the value of first feature of the first prototype. Then next object is sampled and the value of second feature of the sampled object is assigned as the value of second feature of the first prototype. The sampling process is repeated for the other features in similar way. Afterwards the same sampling procedure is repeated for determining the prototypes of remaining clusters.

Value

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

v	a numeric matrix containing the initial cluster prototypes.
ctype	a string representing the type of centroids in the prototype matrix. Its value is 'obj' with this function because it returns objects.
call	a string containing the matched function call that generates this 'inaparc' object.

Author(s)

Zeynel Cebeci, Cagatay Cebeci

See Also

aldaoud, ballhall, crsamp, firstk, forgy, hartiganwong, inofrep, inscsf, insdev, kkz, kmpp, ksegments, ksteps, lastk, lhsmaximin, lhsrandom, maximin, mscseek, rsamp, rsegment, scseek, scseek2, ssamp, spaeth, topbottom, uniquek

Examples

```
data(iris)
res <- ursamp(x=iris[,1:4], k=5)
v <- res$v
print(v)
```

Index