Package 'COveR'

January 20, 2025

Title Clustering with Overlaps

Version 1.1.0

Description Provide functions for overlaps clustering, fuzzy clustering and interval-valued data manipulation. The package implement the following algorithms:
OKM (Overlapping Kmeans) from Cleuziou, G. (2007) <doi:10.1109/icpr.2008.4761079> ;
NEOKM (Non-exhaustive overlapping Kmeans) from Whang, J. J., Dhillon, I. S., and Gleich, D. F. (2015) <doi:10.1137/1.9781611974010.105> ;
Fuzzy Cmeans from Bezdek, J. C. (1981) <doi:10.1007/978-1-4757-0450-1> ;
Fuzzy I-Cmeans from de A.T. De Carvalho, F. (2005) <doi:10.1016/j.patrec.2006.08.014>.

Depends R (>= 3.2.3)

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Suggests testthat

SystemRequirements Gnu Scientific Library version >= 1.12

NeedsCompilation yes

Author Guillaume Cleuziou [aut],
Nicolas Hiot [cre] (<https://orcid.org/0000-0003-4318-4906>)

Maintainer Nicolas Hiot <nicolas.hiot@univ-orleans.fr>

Repository CRAN

Date/Publication 2024-11-13 13:00:22 UTC

Contents

`as.array.interval` *Converts an interval object to an array representation.*

Description

Converts an interval object to an array representation.

Usage

```
## S3 method for class 'interval'
as.array(x, ...)
```

Arguments

x	An interval object to be converted.
...	Additional arguments to be passed to as.array().

Value

An array representation of the interval.

Examples

```
as.array(inter_city)
```

---

as.data.frame.interval

Converts an interval object to a data frame representation.

Description

Converts an interval object to a data frame representation.

Usage

```
## S3 method for class 'interval'
as.data.frame(x, ...)
```

Arguments

x	An interval object to be converted.
...	dditional arguments to be passed to as.data.frame().

Value

A data frame representation of the interval.

Examples

```
as.data.frame(inter_city)
```

as.interval *A generic function to convert various R objects into interval objects.*

Description

A generic function to convert various R objects into interval objects.

Usage

```
as.interval(x)
```

Arguments

x	An R object to be converted to an interval.

Value

An interval object constructed from the R object or NULL if the type is not supported.

as.interval.array *Converts an array to an interval object.*

Description

The array must have three dimensions, with the second dimension of size 2, representing the minimum and maximum values.

Usage

```
## S3 method for class 'array'
as.interval(x)
```

Arguments

x	An array to be converted to an interval object.

Value

An interval object constructed from the array if it meets the requirements, otherwise attempts to convert it to a matrix first.

Examples

```
as.interval(array(1:12, dim = c(2, 2, 3)))
```

as.interval.default *Provides a default method for converting unsupported data types to interval.*

Description

Provides a default method for converting unsupported data types to interval.

Usage

```
## Default S3 method:
as.interval(x)
```

Arguments

x	An object that does not have a supported conversion method.

Value

'NULL', indicating no conversion is possible.

as.interval.interval *Identity Conversion for Interval*

Description

Identity Conversion for Interval

Usage

```
## S3 method for class 'interval'
as.interval(x)
```

Arguments

x	An interval object.

Value

The input interval object, without modification.

Examples

```
as.interval(inter_city)
```

as.interval.matrix *Converts a matrix to an interval object.*

Description

The number of columns in the matrix must be even, representing pairs of minimum and maximum values.

Usage

```
## S3 method for class 'matrix'
as.interval(x)
```

Arguments

x	A matrix where each pair of columns represents the minimum and maximum bounds of intervals.

Value

An interval object constructed from the matrix.

Examples

```
as.interval(matrix(1:12, 3, 4))
```

as.interval.numeric *Converts a numeric vector to an interval object.*

Description

The length of the numeric vector must be even, representing pairs of minimum and maximum values.

Usage

```
## S3 method for class 'numeric'
as.interval(x)
```

Arguments

x	A numeric vector where each consecutive pair of values represents an interval.

Value

An interval object constructed from the numeric vector.

Examples

```
as.interval(1:6)
```

---

as.matrix.interval *Converts an interval object to a matrix representation.*

Description

Each interval is expanded into its minimum and maximum bounds.

Usage

```
## S3 method for class 'interval'
as.matrix(x, ...)
```

Arguments

x	An interval object to be converted.
...	Additional arguments to be passed to as.vector().

Value

A matrix representation of the interval, with two columns for each interval's minimum and maximum values.

Examples

```
as.matrix(inter_city)
```

---

as.vector.interval *Converts an interval object to its vector representation.*

Description

Converts an interval object to its vector representation.

Usage

```
## S3 method for class 'interval'
as.vector(x, ...)
```

Arguments

x	An interval object to be converted.
...	Additional arguments to be passed to as.vector().

Value

A numeric vector where each consecutive pair of values represents an interval.

Examples

```
as.vector(inter_city)
```

`cluster_color` *Generate Colors for Clustering Visualization*

Description

Generates a color for each data point based on its clustering assignment, facilitating visual distinction of clusters in plots.

Usage

```
cluster_color(x)
```

Arguments

`x`	A clustering vector or a matrix. If a vector is provided, it represents the cluster assignments for each data point. If a matrix is provided, each row should represent a data point's membership across multiple clusters.

Value

A character vector of colors (in hexadecimal format) corresponding to the clustering assignments, suitable for use in plotting functions.

Examples

```
plot(iris[, 1:2], col = cluster_color(neokm(iris, 2, 0.2, 0.05)$cluster))
```

`degree2logical` *Transforms a matrix of membership degrees into a logical matrix based on a specified threshold.*

Description

Transforms a matrix of membership degrees into a logical matrix based on a specified threshold.

Usage

```
degree2logical(x, t = min(apply(x, 1, max)))
```

Arguments

x	A matrix of membership degrees.
t	Threshold value for converting the degrees to logical values. By default, it uses the minimum of the maximum values in each row.

Value

A logical matrix where each element is 'TRUE' if it meets or exceeds the threshold, and 'FALSE' otherwise.

Examples

```
degrees <- matrix(runif(9), nrow = 3)
degree2logical(degrees, t = 0.5)
```

---

fuzzy_icmeans *Performs fuzzy c-means clustering on interval data, allowing for soft clustering of data points into multiple clusters.*

Description

Performs fuzzy c-means clustering on interval data, allowing for soft clustering of data points into multiple clusters.

Usage

```
fuzzy_icmeans(
  x,
  centers,
  m = 2,
  nstart = 2,
  distance = "euclid",
  trace = FALSE,
  iter.max = 40
)
```

Arguments

x	A 3D interval array representing the data to be clustered.
centers	Either the number of clusters or a set of pre-initialized cluster centers. If a number is provided, it specifies how many clusters to create.
m	A number greater than 1 that controls the degree of fuzziness in the clustering process (default is 2).
nstart	Number of times to run the clustering algorithm with different starting values to find the best solution (default is 2).

distance	A string specifying the distance metric to use, either 'euclid' for Euclidean distance or 'hausdorff' for Hausdorff distance (default is 'euclid').
trace	Logical, if 'TRUE', tracing information on the progress of the algorithm is displayed (default is 'FALSE').
iter.max	Maximum number of iterations allowed for the clustering algorithm (default is 40).

Value

A list of clustering results, including: - 'cluster': The membership matrix indicating the degree of belonging of each data point to each cluster. - 'centers': The final cluster centers. - 'totss': Total sum of squares. - 'withinss': Within-cluster sum of squares by cluster. - 'tot.withinss': Total within-cluster sum of squares. - 'betweenss': Between-cluster sum of squares. - 'size': Sizes of each cluster. - 'iter': Number of iterations run by the algorithm. - 'overlaps': The average overlap among clusters.

Examples

```
fuzzy_icmeans(iaggregate(iris, col = 5), 2)
fuzzy_icmeans(iaggregate(iris, col = 5), iaggregate(iris, col = 5))
```

---

iaggregate *Aggregates data into a 3D interval array based on a specified column.*

Description

Aggregates data into a 3D interval array based on a specified column.

Usage

```
iaggregate(data, col = 1)
```

Arguments

data	The data frame to aggregate.
col	The index of the column to aggregate by.

Value

A structured interval object representing the aggregated data.

Examples

```
iaggregate(iris, col = 5)
iaggregate(rock, col = 4)
iaggregate(cars, col = 1)
```

ibind — *Combines multiple interval objects into a single interval object.*

Description

Combines multiple interval objects into a single interval object.

Usage

```
ibind(..., class = FALSE)
```

Arguments

...	Interval objects to bind together.
class	Logical value indicating whether to assign a new class label to each interval object when binding. If 'TRUE', each set of intervals will have a distinct class label.

Value

A new interval object containing the combined intervals from the input objects.

Examples

```
ibind(iaggregate(iris, 5), iaggregate(iris, 5))
ibind(iaggregate(iris, 5), iaggregate(iris, 5), iaggregate(iris, 5),
class = TRUE)
```

igenerate — *Creates intervals from Normal Distribution using specified mean and standard deviation values for both the center and half-size of the intervals.*

Description

Creates intervals from Normal Distribution using specified mean and standard deviation values for both the center and half-size of the intervals.

Usage

```
igenerate(n, ...)
```

Arguments

n	Number of intervals to generate.
...	Vectors representing parameters for generating intervals: each vector should contain four values ('center mean', 'center sd', 'half-size mean', 'half-size sd').

Value

An interval object containing the generated intervals.

Examples

```
igenerate(1, c(0, 1, 2, 1))
igenerate(1, c(0, 1, 2, 1), c(100, 1, 2, 1))
```

`ikmeans` *Performs k-means clustering on interval data, allowing for partitioning of data points into distinct clusters.*

Description

Performs k-means clustering on interval data, allowing for partitioning of data points into distinct clusters.

Usage

```
ikmeans(
  x,
  centers,
  nstart = 10,
  distance = "euclid",
  trace = FALSE,
  iter.max = 20
)
```

Arguments

`x`	A 3D interval array representing the data to be clustered.
`centers`	Either the number of clusters to create or a set of pre-initialized cluster centers. If a number is provided, it specifies how many clusters to create.
`nstart`	The number of times to run the k-means algorithm with different starting values in order to find the best solution (default is 10).
`distance`	A string specifying the distance metric to use: 'euclid' for Euclidean distance or 'hausdorff' for Hausdorff distance (default is 'euclid').
`trace`	Logical value indicating whether to show progress of the algorithm (default is 'FALSE').
`iter.max`	Maximum number of iterations allowed for the k-means algorithm (default is 20).

Value

A list of clustering results, including: - ‘cluster‘: A vector indicating the cluster assignment of each data point. - ‘centers‘: The final cluster centers. - ‘totss‘: Total sum of squares. - ‘withinss‘: Within-cluster sum of squares by cluster. - ‘tot.withinss‘: Total within-cluster sum of squares. - ‘betweenss‘: Between-cluster sum of squares. - ‘size‘: The number of points in each cluster. - ‘iter‘: Number of iterations the algorithm executed.

Examples

```
ikmeans(iaggregate(iris, col = 5), 2)
ikmeans(iaggregate(iris, col = 5), iaggregate(iris, col = 5))
```

---

ineokm *Performs clustering on interval data using the Neo-KM algorithm, which allows for overlapping and non-exhaustive cluster membership.*

Description

Performs clustering on interval data using the Neo-KM algorithm, which allows for overlapping and non-exhaustive cluster membership.

Usage

```
ineokm(
  x,
  centers,
  alpha = 0.3,
  beta = 0.05,
  nstart = 10,
  trace = FALSE,
  iter.max = 20
)
```

Arguments

x	A 3D interval array representing the data to be clustered.
centers	Either the number of clusters to create or a set of pre-initialized cluster centers. If a number is provided, it specifies how many clusters to create.
alpha	A numeric value that controls the degree of overlap between clusters (default is 0.3).
beta	A numeric value that controls the non-exhaustiveness of clusters (default is 0.05).
nstart	The number of times to run the Neo-KM algorithm with different starting values in order to find the best solution (default is 10).
trace	Logical value indicating whether to show the progress of the algorithm (default is ‘FALSE‘).

`iter.max` Maximum number of iterations allowed for the Neo-KM algorithm (default is 20).

Value

A list of clustering results, including: - 'cluster': A vector indicating the cluster assignment of each data point. - 'centers': The final cluster centers. - 'totss': Total sum of squares. - 'withinss': Within-cluster sum of squares by cluster. - 'tot.withinss': Total within-cluster sum of squares. - 'betweenss': Between-cluster sum of squares. - 'size': The number of points in each cluster. - 'iter': Number of iterations the algorithm executed.

Examples

```
ineokm(iaggregate(iris, col = 5), 3)
ineokm(iaggregate(iris, col = 5), iaggregate(iris, col = 5), 1, 2)
```

---

inter_city *Temperature by month and humidity in european city.*

Description

Temperature by month and humidity in european city.

Usage

```
inter_city
```

Format

A interval structure with 68 rows and 13 variables divided in min and max with 17 class:

temp.jan

temp.fev

temp.mars

temp.avr

temp.mai

temp.juin

temp.juil

temp.aout

temp.sep

temp.oct

temp.nov

temp.dec

humid

Class :

Allemagne
Angleterre
Autriche
Belgique
Bulgarie
Croatie
Danemark
Espagne
France
Italie
Pays-Bas
Pologne
Portugal
Roumanie
Russie
Turquie
Ukraine

`inter_emotions` *Emotions in music aggregate on BPM to interval multi label data.*

Description

Emotions in music aggregate on BPM to interval multi label data.

Usage

```
inter_emotions
```

Format

A interval structure with 59 rows and 71 variables divided in min and max with 6 class:

Mean_Acc1298_Mean_Mem40_Centroid
Mean_Acc1298_Mean_Mem40_Rolloff
Mean_Acc1298_Mean_Mem40_Flux
Mean_Acc1298_Mean_Mem40_MFCC_0 ...
Mean_Acc1298_Mean_Mem40_MFCC_12
Mean_Acc1298_Std_Mem40_Centroid

Mean_Acc1298_Std_Mem40_Rolloff

Mean_Acc1298_Std_Mem40_Flux

Mean_Acc1298_Std_Mem40_MFCC_0 ...

Mean_Acc1298_Std_Mem40_MFCC_12

Std_Acc1298_Mean_Mem40_Centroid

Std_Acc1298_Mean_Mem40_Rolloff

Std_Acc1298_Mean_Mem40_Flux

Std_Acc1298_Mean_Mem40_MFCC_0 ...

Std_Acc1298_Mean_Mem40_MFCC_12

Std_Acc1298_Std_Mem40_Centroid

Std_Acc1298_Std_Mem40_Rolloff

Std_Acc1298_Std_Mem40_Flux

Std_Acc1298_Std_Mem40_MFCC_0 ...

Std_Acc1298_Std_Mem40_MFCC_12

BH_LowPeakAmp

BH_LowPeakBPM

BH_HighPeakAmp

BH_HighLowRatio

BHSUM1

BHSUM2

BHSUM3

Class :

amazed.suprised

happy.pleased

relaxing.calm

quiet.still

sad.lonely

angry.aggresive

Source

https://mulan.sourceforge.net/datasets-mlc.html

`inter_wine` *Results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars, aggregate on sulfur dioxide to interval simple label data.*

Description

Results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars, aggregate on sulfur dioxide to interval simple label data.

Usage

```
inter_wine
```

Format

A interval structure with 132 rows and 10 variables divided in min and max with 7 class:

fixed.acidity

volatile.acidity

citric.acid

residual.sugar

chlorides

total.sulfur.dioxide

density

pH

sulphates

alcohol

Class :

Class3

Class4

Class5

Class6

Class7

Class8

Class9

Source

https://archive.ics.uci.edu/dataset/186/wine+quality

iokm *Clusters interval data using the OKM (Overlapping K-means) algorithm.*

Description

Clusters interval data using the OKM (Overlapping K-means) algorithm.

Usage

```
iokm(
  x,
  centers,
  nstart = 10,
  distance = "euclid",
  algorithm = "std",
  update = "mean",
  trace = FALSE,
  iter.max = 20,
  secure = FALSE
)
```

Arguments

x	A 3D interval array representing the data to be clustered.
centers	Either the number of clusters to create or a set of pre-initialized cluster centers. If a number is provided, it indicates how many clusters to create.
nstart	The number of times to run the OKM algorithm with different starting values to find the best result (default is 10).
distance	A string specifying the distance metric to use: 'euclid' for Euclidean distance or 'hausdorff' for Hausdorff distance (default is 'euclid').
algorithm	A string specifying the algorithm type to use: 'std' for the standard algorithm or 'matrix' for matrix-based algorithm (default is 'std').
update	A string specifying the update method for cluster centers. Either: 'mean', 'sum', 'join' or 'meet' (default is 'mean').
trace	Logical value indicating whether to show progress of the algorithm (default is 'FALSE').
iter.max	Maximum number of iterations allowed for the OKM algorithm (default is 20).
secure	Logical value indicating whether to ensure that the minimum is less than or equal to the maximum in intervals (default is 'FALSE').

Value

A list of clustering results, including: - ‘cluster‘: Matrix indicating the cluster assignment for each data point. - ‘centers‘: The final cluster centers. - ‘totss‘: Total sum of squares. - ‘withinss‘: Within-cluster sum of squares by elements. - ‘tot.withinss‘: Total within-cluster sum of squares. - ‘betweenss‘: Between-cluster sum of squares. - ‘size‘: The number of points in each cluster. - ‘iter‘: The number of iterations the algorithm executed. - ‘overlaps‘: The average overlap across clusters.

Examples

```
iokm(iaggregate(iris, col = 5), 2)
iokm(iaggregate(iris, col = 5), iaggregate(iris, col = 5))
```

---

is.interval *Determines if an object is a strictly valid interval object.*

Description

Determines if an object is a strictly valid interval object.

Usage

```
is.interval(x)
```

Arguments

x	An R object to be tested.

Value

A logical value indicating whether the object is a valid interval.

Examples

```
is.interval(inter_city)
is.interval(1:4)
```

`measure` *Plots the overlap of membership degrees in a matrix as a function of a threshold.*

Description

Plots the overlap of membership degrees in a matrix as a function of a threshold.

Usage

```
measure(x, min = 0, max = 1, step = 0.1)
```

Arguments

`x`	A matrix of membership degrees.
`min`	Minimum threshold value for the plot (default is 0).
`max`	Maximum threshold value for the plot (default is 1).
`step`	Step size for the threshold values (default is 0.1).

Value

No return value, it plot the overlap as a function of the threshold.

Examples

```
membership_matrix <- matrix(runif(20), nrow = 5)
measure(membership_matrix, min = 0, max = 1, step = 0.2)
```

`neokm` *Clusters data using the NEOKM (Non-Exhaustive Overlapping K-means) algorithm.*

Description

Clusters data using the NEOKM (Non-Exhaustive Overlapping K-means) algorithm.

Usage

```
neokm(
  x,
  centers,
  alpha = 0.3,
  beta = 0.05,
  nstart = 10,
  trace = FALSE,
  iter.max = 20
)
```

Arguments

x	A numeric matrix or data frame containing the data to be clustered.
centers	Either the number of clusters to create or a set of pre-initialized cluster centers. If a number is provided, it indicates how many clusters to create.
alpha	A numeric value representing the degree of overlap allowed between clusters (default is 0.3).
beta	A numeric value representing non-exhaustiveness, which affects the cluster formation (default is 0.05).
nstart	The number of times to run the NEOKM algorithm with different starting values to find the best result (default is 10).
trace	Logical value indicating whether to show progress of the algorithm (default is 'FALSE').
iter.max	Maximum number of iterations allowed for the NEOKM algorithm (default is 20).

Value

A list of clustering results, including: - 'cluster': Matrix indicating the cluster assignment for each data point. - 'centers': The final cluster centers. - 'totss': Total sum of squares. - 'withinss': Within-cluster sum of squares by elements. - 'tot.withinss': Total within-cluster sum of squares. - 'betweenss': Between-cluster sum of squares. - 'size': The number of points in each cluster. - 'iter': The number of iterations the algorithm executed. - 'overlaps': The average overlap across clusters.

Examples

```
neokm(iris[, -5], 3)
neokm(iris[, -5], iris[, -5], 1, 2)
```

---

okm *Clusters data using the OKM (Overlapping K-Means) clustering algorithm.*

Description

Clusters data using the OKM (Overlapping K-Means) clustering algorithm.

Usage

```
okm(x, centers, iter.max = 10, nstart = 1, trace = FALSE, method = "euclid")
```

Arguments

x	A numeric data matrix or data frame containing the data to be clustered.
centers	Either a positive integer indicating the number of clusters to create or a matrix of pre-initialized cluster centers.
iter.max	Maximum number of iterations allowed for the clustering algorithm (default is 10).
nstart	Number of random initializations to find the best result (default is 1).
trace	Logical value indicating whether to display the progress of the algorithm (default is ‘FALSE‘).
method	A string specifying the distance metric to use; options are ’euclid’ (Euclidean distance) or ’manhattan’ (Manhattan distance) (default is "euclid").

Value

A list containing the clustering results, including: - ‘cluster‘: Matrix indicating the cluster assignments for each data point. - ‘centers‘: The final cluster centers. - ‘tot.withinss‘: Total within-cluster sum of squares. - ‘overlaps‘: The measure of overlap among clusters.

Examples

```
okm(iris[, -5], 3)
```

---

plot.interval *Generates a visual representation of interval data as rectangles on a plot.*

Description

Generates a visual representation of interval data as rectangles on a plot.

Usage

```
## S3 method for class 'interval'
plot(x, ...)
```

Arguments

x	An interval object to be plotted.
...	Additional graphical parameters such as ‘col‘ and ‘add‘.

Value

No return value, it plot the interval.

Examples

```
plot(iaggregate(iris, 5))
plot(iaggregate(iris, 5), col = 4)
plot(iaggregate(iris, 5), add = TRUE)
```

`print.icmeans` *Displays the results of fuzzy icmeans clustering in a readable format.*

Description

Displays the results of fuzzy icmeans clustering in a readable format.

Usage

```
## S3 method for class 'icmeans'
print(x, ...)
```

Arguments

`x`	An ‘icmeans‘ object resulting from the ‘fuzzy_icmeans‘ function.
`...`	Additional arguments passed to print().

Value

No return value, it prints the clustering results to the console.

`print.ikmeans` *Displays the results of ikmeans clustering in a readable format.*

Description

Displays the results of ikmeans clustering in a readable format.

Usage

```
## S3 method for class 'ikmeans'
print(x, ...)
```

Arguments

`x`	An ‘ikmeans‘ object resulting from the ‘ikmeans‘ function.
`...`	Additional arguments passed to print().

Value

No return value, it prints the clustering results to the console.

print.ineokm *Displays the results of Neo-KM clustering in a user-friendly format.*

Description

Displays the results of Neo-KM clustering in a user-friendly format.

Usage

```
## S3 method for class 'ineokm'
print(x, ...)
```

Arguments

x	An ‘ineokm‘ object resulting from the ‘ineokm‘ function.
...	Additional arguments passed to print().

Value

No return value, it prints the clustering results to the console.

print.interval *Custom print method for displaying interval objects in a readable format.*

Description

Custom print method for displaying interval objects in a readable format.

Usage

```
## S3 method for class 'interval'
print(x, ...)
```

Arguments

x	An interval object to be printed.
...	Additional arguments passed to the underlying print() function.

Value

No return value, it prints the interval to the console.

Examples

```
print(inter_city)
```

print.iokm *Displays the results of IOKM clustering in a user-friendly format.*

Description

Displays the results of IOKM clustering in a user-friendly format.

Usage

```
## S3 method for class 'iokm'
print(x, ...)
```

Arguments

x	An ‘iokm‘ object resulting from the ‘iokm‘ function.
...	Additional arguments passed to print().

Value

No return value, it prints the clustering results to the console.

print.neokm *Displays the results of NEOKM clustering in a user-friendly format.*

Description

Displays the results of NEOKM clustering in a user-friendly format.

Usage

```
## S3 method for class 'neokm'
print(x, ...)
```

Arguments

x	A ‘neokm‘ object resulting from the ‘neokm‘ function.
...	Additional arguments passed to print().

Value

No return value, it prints the clustering results to the console.

print.okm *Displays the results of OKM clustering in a readable format.*

Description

Displays the results of OKM clustering in a readable format.

Usage

```
## S3 method for class 'okm'
print(x, ...)
```

Arguments

x	An OKM object resulting from the ‘okm‘ function.
...	Additional arguments passed to print().

Value

No return value, it prints the clustering results to the console.

print.r1okm *Displays the results of R1-OKM clustering in a readable format.*

Description

Displays the results of R1-OKM clustering in a readable format.

Usage

```
## S3 method for class 'r1okm'
print(x, ...)
```

Arguments

x	An R1-OKM object resulting from the ‘r1okm‘ function.
...	Additional arguments passed to print().

Value

No return value, it prints the clustering results to the console.

print.r2okm *Displays the results of R2-OKM clustering in a readable format.*

Description

Displays the results of R2-OKM clustering in a readable format.

Usage

```
## S3 method for class 'r2okm'
print(x, ...)
```

Arguments

x	An R2-OKM object resulting from the ‘r2okm‘ function.
...	Additional arguments passed to print().

Value

No return value, it prints the clustering results to the console.

r1okm *Cluster data using the R1-OKM algorithm.*

Description

Cluster data using the R1-OKM algorithm.

Usage

```
r1okm(x, centers, alpha = 0, nstart = 10, trace = FALSE, iter.max = 20)
```

Arguments

x	A numeric data matrix or data frame containing the data to be clustered.
centers	Either a positive integer indicating the number of clusters to create or a matrix of initial cluster centers.
alpha	A numeric parameter controlling the clustering behavior, influencing the degree of overlap between clusters (default is 0).
nstart	Number of random initializations to find the best clustering result (default is 10).
trace	Logical value indicating whether to display progress information during execution (default is ‘FALSE‘).
iter.max	Maximum number of iterations allowed for the clustering algorithm (default is 20).

Value

A list containing the clustering results, including: - ‘cluster‘: Matrix indicating the cluster assignments for each data point. - ‘centers‘: The final cluster centers. - ‘totss‘: Total sum of squares. - ‘withinss‘: Within-cluster sum of squares for each cluster. - ‘tot.withinss‘: Total within-cluster sum of squares. - ‘betweenss‘: Between-cluster sum of squares. - ‘size‘: Number of data points in each cluster. - ‘iter‘: Number of iterations performed. - ‘overlaps‘: Average number of clusters that each point overlaps with.

Examples

```
r1okm(iris[, -5], 3)
r1okm(iris[, -5], 3, alpha = -0.5)
r1okm(iris[, -5], iris[, -5], alpha = 1)
```

`r2okm` *Cluster data using the R2-OKM algorithm.*

Description

Cluster data using the R2-OKM algorithm.

Usage

```
r2okm(x, centers, lambda = 0, nstart = 10, trace = FALSE, iter.max = 20)
```

Arguments

`x`	A numeric data matrix or data frame containing the data to be clustered.
`centers`	Either a positive integer specifying the number of clusters to create or a matrix of initial cluster centers.
`lambda`	A numeric parameter that controls the clustering behavior, influencing the shape and separation of clusters (default is 0).
`nstart`	Number of random initializations to find the best clustering result (default is 10).
`trace`	Logical value indicating whether to display progress information during execution (default is ‘FALSE‘).
`iter.max`	Maximum number of iterations allowed for the clustering algorithm (default is 20).

Value

A list containing the clustering results, which includes: - ‘cluster‘: Matrix indicating the cluster assignments for each data point. - ‘centers‘: The final cluster centers. - ‘totss‘: Total sum of squares. - ‘withinss‘: Within-cluster sum of squares for each cluster. - ‘tot.withinss‘: Total within-cluster sum of squares. - ‘betweenss‘: Between-cluster sum of squares. - ‘size‘: Number of data points in each cluster. - ‘iter‘: Number of iterations performed. - ‘overlaps‘: Average number of clusters that each point overlaps with.

Examples

```
r2okm(iris[, -5], 3)
r2okm(iris[, -5], 3, lambda = 0.3)
r2okm(iris[, -5], iris[, -5], lambda = 1)
```

---

read.interval *Reads a CSV file and converts the data into a 3D interval array.*

Description

Reads a CSV file and converts the data into a 3D interval array.

Usage

```
read.interval(..., row.names = FALSE, class = NULL)
```

Arguments

...	Additional arguments passed to read.csv().
row.names	Logical indicating if the first column contains row names.
class	The column index of class labels (set to 'NULL' if not present).

Value

A structured interval object representing the data from the CSV file.

---

write.interval *Writes an interval object to a CSV file.*

Description

Writes an interval object to a CSV file.

Usage

```
write.interval(x, ..., class = FALSE)
```

Arguments

x	An interval object to be saved.
...	Additional arguments passed to write.csv().
class	Logical indicating whether to add the class column in the CSV.

Value

No return value, it saves the interval to the given CSV file.

Index