# Package 'WeightedCluster'

July 6, 2020

**Version** 1.4-1

**Date** 2020-07-05

**Title** Clustering of Weighted Data

**Author** Matthias Studer [aut, cre]

**Maintainer** Matthias Studer <matthias.studer@unige.ch>

**Depends** R (>= 3.0.0), TraMineR (>= 2.0-6), cluster

**Imports** utils, RColorBrewer

**Suggests** RUnit, knitr, isotone, vegan, lattice

**SystemRequirements** C++11

**VignetteBuilder** knitr

**Description** Clusters state sequences and weighted data. It provides an optimized weighted PAM algorithm as well as functions for aggregating replicated cases, computing cluster quality measures for a range of clustering solutions and plotting clusters of state sequences.

**License** GPL (>= 2)

**URL** <http://mephisto.unige.ch/weightedcluster>

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2020-07-06 13:10:02 UTC

## R topics documented:

---

as.clustrange                      *Build a clustrange object to compare different clustering solutions.*

---

### Description

Build a clustrange object to compare different clustering solutions.

### Usage

```
as.clustrange(object, diss, weights=NULL, R=1,  samplesize=NULL, ...)
## S3 method for class 'twins'
as.clustrange(object, diss, weights=NULL, R=1, samplesize=NULL,
ncluster=20, ...)
## S3 method for class 'hclust'
as.clustrange(object, diss, weights=NULL, R=1, samplesize=NULL,
ncluster=20, ...)
## S3 method for class 'dtclust'
as.clustrange(object, diss, weights=NULL, R=1, samplesize=NULL,
ncluster=20, labels = TRUE, ...)
## S3 method for class 'clustrange'
plot(x, stat="noCH", legendpos="bottomright",
    norm="none", withlegend=TRUE, lwd=1, col=NULL, ylab="Indicators",
xlab="N clusters", conf.int=0.9, ci.method="none", ci.alpha=.3, line="t0", ...)
```

### Arguments

| | |
|---|---|
| object | The object to convert such as a data.frame. |
| diss | A dissimilarity matrix or a dist object (see [dist](#)). |
| weights | Optional numerical vector containing weights. |
| R | Optional number of bootstrap that can be used to build confidence intervals. |
| samplesize | Size of bootstrap sample. Default to sum of weights. |
| ncluster | Integer. Maximum number of cluster. The range will include all clustering solution starting from two to ncluster. |
| labels | Logical. If TRUE, rules to assign an object to a sequence is used to label the cluster (instead of a number). |
| x | A clustrange object to be plotted. |
| stat | Character. The list of statistics to plot or "noCH" to plot all statistics except "CH" and "CHsq" or "all" for all statistics. See [wcClusterQuality](#) for a list of possible values. It is also possible to use "RHC" to plot the quality measure 1-HC. Unlike HC, RHC should be maximized as all other quality measures. |
| legendpos | Character. legend position, see [legend](#). |

| norm | Character. Normalization method of the statistics can be one of "none" (no normalization), "range" (given as (value -min)/(max-min), "zscore" (adjusted by mean and standard deviation) or "zscoremed" (adjusted by median and median of the difference to the median). |
|------|------|
| withlegend | Logical. If FALSE, the legend is not plotted. |
| lwd | Numeric. Line width, see [par]. |
| col | A vector of line colors, see [par]. If NULL, a default set of color is used. |
| xlab | x axis label. |
| ylab | y axis label. |
| conf.int | Confidence to build the confidence interval (default: 0.9). |
| ci.method | Method used to build the confidence interval (only if bootstrap has been used, see R above). One of "none" (do not plot confidence interval), "norm" (based on normal approximation), "perc" (based on percentile).) |
| ci.alpha | alpha color value used to plot the interval. |
| line | Which value should be plotted by the line? One of "t0" (value for actual sample), "mean" (average over all bootstraps), "median"(median over all bootstraps). |
| ... | Additionnal parameters passed to/from methods. |

## Details

as.clustrange convert objects to clustrange objects. clustrange objects contains a list of clustering solution with associated statistics and can be used to find the optimal clustering solution.

If object is a data.frame or a matrix, each column should be a clustering solution to be evaluated.

If object is an hclust or twins objects (i.e. hierarchical clustering output, see [hclust], [diana] or [agnes]), the function compute all clustering solution ranging from two to ncluster and compute the associated statistics.

## Value

An object of class clustrange with the following elements:

clustering: A data.frame of all clustering solutions.

stats: A matrix containing the clustering statistics of each cluster solution.

## See Also

See also [wcKMedRange], [wcClusterQuality].

## Examples

```
data(mvad)
## Aggregating state sequence
aggMvad <- wcAggregateCases(mvad[, 17:86], weights=mvad$weight)

## Creating state sequence object
mvad.seq <- seqdef(mvad[aggMvad$aggIndex, 17:86], weights=aggMvad$aggWeights)
```

```
## COmpute distance using Hamming distance
diss <- seqdist(mvad.seq, method="HAM")

## Ward clustering
wardCluster <- hclust(as.dist(diss), method="ward", members=aggMvad$aggWeights)

## Computing clustrange from Ward clustering
wardRange <- as.clustrange(wardCluster, diss=diss,
weights=aggMvad$aggWeights, ncluster=15)

## Plot all statistics (standardized)
plot(wardRange, stat="all", norm="zscoremed", lwd=3)

## Plot HC, RHC and ASW
plot(wardRange, stat=c("HC", "RHC", "ASWw"), norm="zscore", lwd=3)
```

---

as.seqtree                          *Convert a hierarchical clustering object to a seqtree object.*

---

### Description

Convert a hierarchical clustering object to a seqtree object which can then be displayed using
[seqtreedisplay](#).

### Usage

```
as.seqtree(object, seqdata, diss, weighted=TRUE, ...)
## S3 method for class 'twins'
as.seqtree(object, seqdata, diss, weighted=TRUE, ncluster, ...)
## S3 method for class 'hclust'
as.seqtree(object, seqdata, diss, weighted=TRUE, ncluster, ...)
```

### Arguments

| | |
|---|---|
| object | An object to be converted to a [seqtree](#). |
| seqdata | State sequence object. |
| diss | A dissimilarity matrix or a dist object (see [dist](#)) |
| weighted | Logical. If `TRUE`, `weights` of the `seqdata` object are taken to build the tree. |
| ncluster | Maximum number of cluster. The tree will be builded until this number of cluster. |
| ... | Additionnal parameters passed to/from methods. |

## Details

By default as.seqtree try to convert the object to a data.frame assuming that it contains a list of nested clustering solutions. Be aware that seqtree and as.seqtree only support binary splits.

If object is an hclust or twins objects (i.e. hierarchical clustering output, see [hclust](), [diana]() or [agnes]()), the function returns a seqtree object reproducing the agglomerative schedulde.

## Value

A [seqtree]() object.

## Examples

```
data(mvad)
## Aggregating state sequence
aggMvad <- wcAggregateCases(mvad[, 17:86], weights=mvad$weight)

## Creating state sequence object
mvad.seq <- seqdef(mvad[aggMvad$aggIndex, 17:86], weights=aggMvad$aggWeights)

## COmpute distance using Hamming distance
diss <- seqdist(mvad.seq, method="HAM")

## Ward clustering
wardCluster <- hclust(as.dist(diss), method="ward", members=aggMvad$weight)

st <- as.seqtree(wardCluster, seqdata=mvad.seq, diss=diss, weighted=TRUE, ncluster=10)

print(st)

## You typically want to run (You need to install GraphViz before)
## seqtreedisplay(st, type="d", border=NA)
```

---

fuzzyseqplot                 *Plot sequences according to a fuzzy clustering.*

---

## Description

This funciton propose a graphical representation of a fuzzy clustering results where sequences are weighted according to their cluster membership strength.

## Usage

```
fuzzyseqplot(seqdata, group = NULL, membership.threashold = 0, type = "i",
members.weighted = TRUE, memb.exp = 1, ...)
```

## Arguments

| | |
|---|---|
| seqdata | State sequence object created with the [seqdef](#) function. |
| group | A fuzzy partition of the data, either as a membership matrix or as a `fanny` object. |
| membership.threashold | |
| | Numeric. Minimum membership strength to be included in plots. |
| type | the type of the plot. Available types are `"d"` for state distribution plots (chronograms), `"f"` for sequence frequency plots, `"i"` for selected sequence index plots, `"I"` for whole set index plots, `"ms"` for plotting the sequence of modal states, `"mt"` for mean times plots, `"pc"` for parallel coordinate plots and `"r"` for representative sequence plots. |
| members.weighted | |
| | Logical. Should the sequences be weighted by their membership strength in each group before being plotted? |
| memb.exp | Optional. Fuzzyness parameter used in the `fanny` algorithm. |
| ... | arguments to be passed to [seqplot](#). |

## Details

The dataset is augmented by repeating the sequence $s_i$ of individual $i$ $k$ times (i.e., once per cluster). We therefore have $k$ sequences for individual $i$, denoted as $s_{i1}...s_{ik}$. These sequences are therefore weighted according to their membership degree $u_{i1}...u_{ik}$. Hence, even if the same sequence were repeated $k$ times, its total weight sum to 1. An additional categorical covariate is created in this augmented dataset that specifies the cluster (ranging from 1 to $k$) of the associated membership degree. This weighting strategy allows us to use any tools available for weighted sequence data (see [seqplot](#)).

For index plots, we additionally suggest ordering the sequences according to membership degree by setting `sortv="membership"` (see example). The most typical sequence lies at the top of the subfigures, with a high membership degree; meanwhile, the bottom shows less-characteristic patterns. Restricting to sequences with the highest membership degree can be achieved with the `membership.treashold` argument.

## References

Studer, M. (2018). Divisive property-based and fuzzy clustering for sequence analysis. In G. Ritschard and M. Studer (Eds.), *Sequence Analysis and Related Approaches: Innovative Methods and Applications*, Life Course Research and Social Policies.

## See Also

See also [fanny](#) for fuzzy clustering.

## Examples

```
data(mvad)
mvad.seq <- seqdef(mvad[1:100, 17:86])

## COmpute distance using Hamming distance
```

```
diss <- seqdist(mvad.seq, method="HAM")
library(cluster)
fclust <- fanny(diss, k=2, diss=TRUE)

fuzzyseqplot(mvad.seq, group=fclust, type="d")
fuzzyseqplot(mvad.seq, group=fclust, type="I", sortv="membership")
fuzzyseqplot(mvad.seq, group=fclust, type="f")
```

---

| seqclustname | *Automatic labeling of cluster using sequence medoids* |
|---|---|

---

### Description

This function automatically name the cluster using the sequence medoid of each cluster.

### Usage

```
seqclustname(seqdata, group, diss, weighted = TRUE, perc = FALSE)
```

### Arguments

| | |
|---|---|
| seqdata | State sequence object (see [seqdef](#)). |
| group | A vector of clustering membership. |
| diss | a dissimilarity matrix or a `dist` object. |
| weighted | Logical. If TRUE, `weights` of the `seqdata` object are taken to find the medoids. |
| perc | Logical. If TRUE, the percentage of sequences in each cluster is added to the label of each group. |

### Value

A factor of clustering membership. The labels are defined using sequences medoids and optionnaly percentage of case in each cluster.

### Examples

```
data(mvad)
## Aggregating state sequence
aggMvad <- wcAggregateCases(mvad[, 17:86], weights=mvad$weight)

## Creating state sequence object
mvad.seq <- seqdef(mvad[aggMvad$aggIndex, 17:86], weights=aggMvad$aggWeights)
## Computing Hamming distance between sequence
diss <- seqdist(mvad.seq, method="HAM")

## KMedoids using PAMonce method (clustering only)
clust5 <- wcKMedoids(diss, k=5, weights=aggMvad$aggWeights)

clust5.labels <- seqclustname(mvad.seq, clust5$clustering, diss=diss, perc=TRUE)
seqdplot(mvad.seq, group=clust5.labels)
```

---

**seqpropclust**                      *Monothetic clustering of state sequences*

---

### Description

Monothetic divisive clustering of the data using object properties. For state sequences object different set of properties are automoatically extracted.

### Usage

```
seqpropclust(seqdata, diss, properties = c("state", "duration", "spell.age",
"spell.dur", "transition", "pattern", "AFtransition", "AFpattern",
"Complexity"), other.prop = NULL, prop.only = FALSE, pmin.support = 0.05,
max.k = -1, with.missing = TRUE, R = 1, weight.permutation = "diss",
min.size = 0.01, max.depth = 5, maxcluster = NULL, ...)

wcPropertyClustering(diss, properties, maxcluster = NULL, ...)
dtcut(st, k, labels = TRUE)
```

### Arguments

| | |
|---|---|
| seqdata | State sequence object (see [seqdef](#)). |
| diss | a dissimilarity matrix or a dist object. |
| properties | Character or data.frame. In seqpropclust, it can be a list of properties to be extracted from seqdata. It can also be a data.frame specifying the properties to use for the clustering. |
| other.prop | data.frame. Additional properties to be considered to cluster the sequences. |
| prop.only | Logical. If TRUE, the function returns a data.frame containing the extracted properties (without clustering the data). |
| pmin.support | Numeric. Minimum support (as a proportion of sequences). See [seqefsub](#). |
| max.k | Numeric. The maximum number of events allowed in a subsequence. See [seqefsub](#). |
| with.missing | Logical. If TRUE, property of missing spell are also extracted. |
| R | Number of permutations used to assess the significance of the split. See [disstree](#). |
| weight.permutation | |
| | Weight permutation method: "diss" (attach weights to the dissimilarity matrix), "replicate" (replicate cases using weights), "rounded-replicate" (replicate case using rounded weights), "random-sampling" (random assignment of covariate profiles to the objects using distributions defined by the weights.). See [disstree](#). |
| min.size | Minimum number of cases in a node, will be treated as a proportion if less than 1. See [disstree](#). |
| max.depth | Maximum depth of the tree. See [disstree](#). |

| maxcluster | Maximum number of cluster to consider. |
| st | A divise clustering tree as produced by `seqpropclust` |
| k | The number of groups to extract. |
| labels | Logical. If `TRUE`, rules to assign an object to a sequence is used to label the cluster (instead of a number). |
| ... | Arguments passed to/from other methods. |

## Details

The method implement the DIVCLUS-T algorithm.

## Value

Return a `seqpropclust` object, which is (in fact) a `distree` object. See [`disstree`](#).

## References

Studer, M. (2018). Divisive property-based and fuzzy clustering for sequence analysis. In G. Ritschard and M. Studer (Eds.), *Sequence Analysis and Related Approaches: Innovative Methods and Applications*, Life Course Research and Social Policies. Springer.

Piccarreta R, Billari FC (2007). Clustering work and family trajectories by using a divisive algorithm. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(4), 1061-1078.

Chavent M, Lechevallier Y, Briant O (2007). DIVCLUS-T: A monothetic divisive hierarchical clustering method. *Computational Statistics & Data Analysis*, 52(2), 687-701.

## See Also

[`as.clustrange`](#), [`seqtreedisplay`](#), [`disstree`](#).

## Examples

```
data(mvad)
mvad.seq <- seqdef(mvad[1:100, 17:86])

## COmpute distance using Hamming distance
diss <- seqdist(mvad.seq, method="HAM")

pclust <- seqpropclust(mvad.seq , diss=diss, maxcluster=5, properties=c("state", "duration"))

## Run it to visualize the results
##seqtreedisplay(pclust, type="d", border=NA, showdepth=TRUE)

pclustqual <- as.clustrange(pclust, diss=diss, ncluster=5)
```

wcAggregateCases            *Aggregate identical cases.*

---

**Description**

Function to aggregate identical cases.

**Usage**

```
wcAggregateCases(x, weights = NULL, ...)
## S3 method for class 'data.frame'
wcAggregateCases(x, weights=NULL, ...)
## S3 method for class 'matrix'
wcAggregateCases(x, weights=NULL, ...)
## S3 method for class 'wcAggregateCases'
print(x,  ...)
```

**Arguments**

| | |
|---|---|
| x | The object to aggregate. |
| weights | Numeric. An optional case weights vector. |
| ... | Optional additionnal arguments. |

**Value**

A wcAggregateCases object with the following components:

**aggIndex**  Index of the unique cases in the original object data.

**aggWeights**  Aggregated case weights

**disaggIndex**  Index of the original object data in the unique cases.

**disaggWeights**  Original weights used.

**Examples**

```
data(mvad)
## Taking only the father unemployment and
## success at the end of compulsory schooling.
myData <- mvad[ , c("funemp", "gcse5eq")]
## Computing aggregated cases informations
ac <- wcAggregateCases(myData, weights=mvad$weight)
print(ac)
## Retrieving unique cases in the original data set
uniqueData <- myData[ac$aggIndex, ]
## Table from original data
table.orig <- xtabs(mvad$weight~funemp+gcse5eq, data=myData)

## Table from aggregated data
```

```
table.agg <- xtabs(ac$aggWeights~funemp+gcse5eq, data=uniqueData)

## Both table are equal, no information is lost
## (only the call command is different)
all(table.orig == table.agg)
```

---

wcClusterQuality            *Cluster quality statistics*

---

### Description

Compute several quality statistics of a given clustering solution.

### Usage

```
wcClusterQuality(diss, clustering, weights = NULL)
```

### Arguments

| | |
|---|---|
| diss | A dissimilarity matrix or a dist object (see [dist](#)) |
| clustering | Factor. A vector of clustering membership. |
| weights | optional numerical vector containing weights. |

### Details

Compute several quality statistics of a given clustering solution. See value for details.

### Value

A list with two elements stats and ASW:

stats with the following statistics:

**PBC** Point Biserial Correlation. Correlation between the given distance matrice and a distance which equal to zero for individuals in the same cluster and one otherwise.

**HG** Hubert's Gamma. Same as previous but using Kendall's Gamma coefficient.

**HGSD** Hubert's Gamma (Somers'D). Same as previous but using Somers' D coefficient.

**ASW** Average Silhouette width (observation).

**ASWw** Average Silhouette width (weighted).

**CH** Calinski-Harabasz index (Pseudo F statistics computed from distances).

**R2** Share of the discrepancy explained by the clustering solution.

**CHsq** Calinski-Harabasz index (Pseudo F statistics computed from *squared* distances).

**R2sq** Share of the discrepancy explained by the clustering solution (computed using *squared* distances).

**HC** Hubert's C coefficient.

ASW: The Average Silhouette Width of each cluster, one column for each ASW measure.

**Examples**

```
data(mvad)
## Aggregating state sequence
aggMvad <- wcAggregateCases(mvad[, 17:86], weights=mvad$weight)

## Creating state sequence object
mvad.seq <- seqdef(mvad[aggMvad$aggIndex, 17:86], weights=aggMvad$aggWeights)
## Computing Hamming distance between sequence
diss <- seqdist(mvad.seq, method="HAM")

## KMedoids using PAMonce method (clustering only)
clust5 <- wcKMedoids(diss, k=5, weights=aggMvad$aggWeights, cluster.only=TRUE)

## Compute the silhouette of each observation
qual <- wcClusterQuality(diss, clust5, weights=aggMvad$aggWeights)

print(qual)
```

---

wcCmpCluster                    *Automatic comparison of clustering methods.*

---

**Description**

Automatically compute different clustering solutions and associated quality measures to help iden-
tifying the best one.

**Usage**

```
wcCmpCluster(diss, weights = NULL, maxcluster, method = "all", pam.combine = TRUE)
## S3 method for class 'clustrangefamily'
print(x, max.rank=1, ...)
## S3 method for class 'clustrangefamily'
summary(object, max.rank=1, ...)
## S3 method for class 'clustrangefamily'
plot(x, group="stat", method="all", pam.combine=FALSE,
     stat="noCH", norm="none", withlegend=TRUE, lwd=1, col=NULL, legend.prop=NA,
rows=NA, cols=NA, main=NULL, xlab="", ylab="", ...)
```

**Arguments**

| | |
|---|---|
| diss | A dissimilarity matrix or a dist object (see [dist](#)). |
| weights | Optional numerical vector containing weights. |
| maxcluster | Integer. Maximum number of cluster. The range will include all clustering solution starting from two to ncluster. |

| method | A vector of hierarchical clustering methods to compute or "all" for all methods. Possible values include "ward", "single", "complete", "average", "mcquitty", "median", "centroid" (using [hclust](#)), "pam" (using [wcKMedRange](#)), "diana" (only for unweighted datasets using [diana](#)), "beta.flexible" (only for unweighted datasets using [agnes](#)) |
|---|---|
| pam.combine | Logical. Should we try all combinations of hierarchical and PAM clustering? |
| x | A clustrangefamily object to plot or print |
| object | A clustrangefamily object to summarize |
| max.rank | Integer. The different number of solution to print/summarize |
| group | One of "stat" or "method". If "stat", plots are grouped by statistics, otherwise by clustering methods. |
| stat | Character. The list of statistics to plot or "noCH" to plot all statistics except "CH" and "CHsq" or "all" for all statistics. See [wcClusterQuality](#) for a list of possible values. It is also possible to use "RHC" to plot the quality measure 1-HC. Unlike HC, RHC should be maximized as all other quality measures. |
| norm | Character. Normalization method of the statistics can be one of "none" (no normalization), "range" (given as (value -min)/(max-min), "zscore" (adjusted by mean and standard deviation) or "zscoremed" (adjusted by median and median of the difference to the median). |
| withlegend | Logical. If FALSE, the legend is not plotted. |
| lwd | Numeric. Line width, see [par](#). |
| col | A vector of line colors, see [par](#). If NULL, a default set of color is used. |
| legend.prop | When withlegend=TRUE, sets the proportion of the graphic area used for plotting the legend. Default value is set according to the place (bottom or right of the graphic area) where the legend is plotted. Values from 0 to 1. |
| rows,cols | optional arguments to arrange plots. |
| xlab | x axis label. |
| ylab | y axis label. |
| main | main title of the plot. |
| ... | Additionnal parameters passed to [lines](#). |

### Value

An object of class clustrangefamily with the following elements:

**Method name:** the results of [as.clustrange](#) objects under each method name (see argument method for a list of possible values)

**allstats:** A matrix containing the clustering statistics for each cluster solution and method.

**param:** The parameters set when the function was called.

### See Also

See Also [as.clustrange](#)

## Examples

```
data(mvad)

#Creating state sequence object
mvad.seq <- seqdef(mvad[, 17:86])

# COmpute distance using Hamming distance
diss <- seqdist(mvad.seq, method="HAM")

#Ward clustering
allClust <- wcCmpCluster(diss, maxcluster=15, method=c("average", "pam", "beta.flexible"),
                         pam.combine=FALSE)

summary(allClust, max.rank=3)

##Plot PBC, RHC and ASW
plot(allClust, stat=c("PBC", "RHC", "ASW"), norm="zscore", lwd=2)


##Plot PBC, RHC and ASW grouped by cluster method
plot(allClust, group="method", stat=c("PBC", "RHC", "ASW"), norm="zscore", lwd=2)
```

---

wcKMedoids                    *K-Medoids or PAM clustering of weighted data.*

---

## Description

K-Medoids or PAM clustering of weighted data.

## Usage

```
wcKMedoids(diss, k, weights=NULL, npass = 1, initialclust=NULL,
method="PAMonce", cluster.only = FALSE, debuglevel=0)
```

## Arguments

| | |
|---|---|
| diss | A dissimilarity matrix or a dist object (see [dist](#)). |
| k | Integer. The number of cluster. |
| weights | Numeric. Optional numerical vector containing case weights. |
| npass | Integer. Number of random start solution to test. |
| initialclust | An integer vector, a factor, an "hclust" or a "twins" object. Can be either the index of the initial medoids (length should equal to k) or a vector specifying an initial clustering solution (length should then be equal to the number of observation.). If initialclust is an "hclust" or a "twins" object, then the initial clustering solution is taken from the hierarchical clustering in k groups. |
| method | Character. One of "KMedoids", "PAM" or "PAMonce" (default). See details. |
| cluster.only | Logical. If FALSE, the quality of the retained solution is computed. |
| debuglevel | Integer. If greater than zero, print some debugging messages. |

**Details**

K-Medoids algorithms aim at finding the best partition of the data in a k predefined number of groups. Based on a dissimilarity matrix, those algorithms seeks to minimize the (weighted) sum of distance to the medoid of each group. The medoid is defined as the observation that minimize the sum of distance to the other observations of this group. The function wcKMedoids support three differents algorithms specified using the method argument:

**"KMedoids"** Start with a random solution and then iteratively adapt the medoids using an algorithm similar to kmeans. Part of the code is inspired (but completely rewritten) by the C clustering library (see de Hoon et al. 2010). If you use this solution, you should set npass>1 to try several solution.

**"PAM"** See pam in the cluster library. This code is based on the one available in the cluster library (Maechler et al. 2011). The advantage over the previous method is that it try to minimize a global criteria instead of a local one.

**"PAMonce"** Same as previous but with two optimizations. First, the optimization presented by Reynolds et al. 2006. Second, only evaluate possible swap if the dissimilarity is greater than zero. This algorithm is used by default.

wcKMedoids works differently according to the diss argument. It may be faster using a matrix but require more memory (since all distances are stored twice). All combination between method and diss argument are possible, except for the "PAM" algorithm were only distance matrix may be used (use the "PAMonce" algorithm instead).

**Value**

An integer vector with the index of the medoids associated with each observation.

**References**

Maechler, M., P. Rousseeuw, A. Struyf, M. Hubert and K. Hornik (2011). cluster: Cluster Analysis Basics and Extensions. R package version 1.14.1 — For new features, see the 'Changelog' file (in the package source).

Hoon, M. d.; Imoto, S. & Miyano, S. (2010). The C Clustering Library. Manual

**See Also**

pam in the cluster library, wcClusterQuality, wcKMedRange.

**Examples**

```
data(mvad)
## Aggregating state sequence
aggMvad <- wcAggregateCases(mvad[, 17:86], weights=mvad$weight)

## Creating state sequence object
mvad.seq <- seqdef(mvad[aggMvad$aggIndex, 17:86], weights=aggMvad$aggWeights)
## Computing Hamming distance between sequence
diss <- seqdist(mvad.seq, method="HAM")
```

```
## K-Medoids
clust5 <- wcKMedoids(diss, k=5, weights=aggMvad$aggWeights)

## clust5$clustering contains index number of each medoids
## Those medoids are
unique(clust5$clustering)

## Print the medoids sequences
print(mvad.seq[unique(clust5$clustering), ], informat="SPS")

## Some info about the clustering
print(clust5)

## Plot sequences according to clustering solution.
seqdplot(mvad.seq, group=clust5$clustering)
```

---

wcKMedRange                 *Compute* wcKMedoids *clustering for different number of clusters.*

---

### Description

Compute wcKMedoids clustering for different number of clusters.

### Usage

```
wcKMedRange(diss, kvals, weights=NULL, R=1,  samplesize=NULL, ...)
```

### Arguments

| | |
|---|---|
| diss | A dissimilarity matrix or a dist object (see dist). |
| kvals | A numeric vector containing the number of cluster to compute. |
| weights | Numeric. Optional numerical vector containing case weights. |
| R | Optional number of bootstrap that can be used to build confidence intervals. |
| samplesize | Size of bootstrap sample. Default to sum of weights. |
| ... | Additionnal parameters passed to wcKMedoids. |

### Details

Compute a clustrange object using the wcKMedoids method. clustrange objects contains a list of clustering solution with associated statistics and can be used to find the optimal clustering solution.

See as.clustrange for more details.

### See Also

See as.clustrange.

## Examples

```
data(mvad)
## Aggregating state sequence
aggMvad <- wcAggregateCases(mvad[, 17:86], weights=mvad$weight)

## Creating state sequence object
mvad.seq <- seqdef(mvad[aggMvad$aggIndex, 17:86], weights=aggMvad$aggWeights)

## Compute distance using Hamming distance
diss <- seqdist(mvad.seq, method="HAM")

## Pam clustering
pamRange <- wcKMedRange(diss, 2:15)

## Plot all statistics (standardized)
plot(pamRange, stat="all", norm="zscoremed", lwd=3)

## Plotting sequences in 3 groups
seqdplot(mvad.seq, group=pamRange$clustering$cluster3)
```

---

wcSilhouetteObs          *Compute the silhouette of each object using weighted data.*

---

### Description

Compute the silhouette of each object using weighted data.

### Usage

```
wcSilhouetteObs(diss, clustering, weights = NULL, measure="ASW")
```

### Arguments

| | |
|---|---|
| diss | A dissimilarity matrix or a dist object (see [dist](#)) |
| clustering | Factor. A vector of clustering membership. |
| weights | optional numerical vector containing weights. |
| measure | "ASW" or "ASWw", the measure of the silhouette. See the WeigthedCluster vignettes. |

### Details

See the [silhouette](#) function in the [cluster](#) package for a detailed explanation of the silhouette.

### Value

A numeric vector containing the silhouette of each observation.

**References**

Maechler, M., P. Rousseeuw, A. Struyf, M. Hubert and K. Hornik (2011). cluster: Cluster Analysis Basics and Extensions. R package version 1.14.1 — For new features, see the 'Changelog' file (in the package source).

**See Also**

See also [silhouette](#).

**Examples**

```
data(mvad)
## Aggregating state sequence
aggMvad <- wcAggregateCases(mvad[, 17:86], weights=mvad$weight)

## Creating state sequence object
mvad.seq <- seqdef(mvad[aggMvad$aggIndex, 17:86], weights=aggMvad$aggWeights)
## Computing Hamming distance between sequence
diss <- seqdist(mvad.seq, method="HAM")

## KMedoids using PAMonce method (clustering only)
clust5 <- wcKMedoids(diss, k=5, weights=aggMvad$aggWeights, cluster.only=TRUE)

## Compute the silhouette of each observation
sil <- wcSilhouetteObs(diss, clust5, weights=aggMvad$aggWeights, measure="ASWw")

## If you want to compute the average silhouette width,
## you should take weights into account
weighted.mean(sil, w=aggMvad$aggWeights)

## Plotting sequences ordred by silhouette width,
## best classified are draw on the top.
seqIplot(mvad.seq, group=clust5, sortv=sil)
```

# Index