Package 'geocmeans'

September 12, 2023

Type Package

Title Implementing Methods for Spatial Fuzzy Unsupervised Classification

Version 0.3.4

Maintainer Jeremy Gelb <jeremy.gelb@ucs.inrs.ca>

Imports ggplot2 (>= 3.2.1), tmap (>= 3.3-1), spdep (>= 1.1.2), reldist (>= 1.6.6), dplyr (>= 0.8.3), fclust (>= 2.1.1), fmsb (>= 0.7.0), future.apply (>= 1.4.0), progressr (>= 0.4.0), reshape2 (>= 1.4.4), stats (>= 3.5), grDevices (>= 3.5), shiny (>= 1.6.0), sf (>= 1.0-6), leaflet (>= 2.1.1), plotly (>= 4.9.3), Rdpack (>= 2.1.1), matrixStats (>= 0.58.0), methods (>= 3.5), terra (>= 1.6-47), Rcpp (>= 1.0.6)

Depends R (>= 3.5)

Suggests knitr (>= 1.28), rmarkdown (>= 2.1), markdown (>= 1.1), future (>= 1.16.0), ppclust (>= 1.1.0), ClustGeo (>= 2.0), car (>= 3.0-7), rgl (>= 0.100), ggpubr (>= 0.2.5), RColorBrewer (>= 1.1-2), kableExtra (>= 1.1.0), viridis (>= 0.5.1), testthat (>= 3.0.0), bslib (>= 0.2.5), shinyWidgets (>= 0.6), shinyhelper (>= 0.3.2), waiter (>= 0.2.2), classInt(>= 0.4-3), covr

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

VignetteBuilder knitr

Description Provides functions to apply spatial fuzzy unsupervised classification, visualize and interpret results. This method is well suited when the user wants to analyze data with a fuzzy clustering algorithm and to account for the spatial dimension of the dataset. In addition, indexes for estimating the spatial consistency and classification quality are proposed.
The methods were originally proposed in the field of brain imagery (seed Cai and al. 2007 <doi:10.1016/j.patcog.2006.07.011> and Zaho and al. 2013 <doi:10.1016/j.dsp.2012.09.016>) cently applied in geography (see Gelb and Apparicio <doi:10.4000/cybergeo.36414>).

URL https://github.com/JeremyGelb/geocmeans

BugReports https://github.com/JeremyGelb/geocmeans/issues

RdMacros Rdpack

LinkingTo Rcpp, RcppArmadillo

SystemRequirements C++17

Language en-CA

NeedsCompilation yes

Author Jeremy Gelb [aut, cre] (<https://orcid.org/0000-0002-7114-2714>),
Philippe Apparicio [ctb] (<https://orcid.org/0000-0001-6466-9342>)

Repository CRAN

Date/Publication 2023-09-12 03:10:02 UTC

R topics documented:

adjustSpatialWeights . . . 3
Arcachon . . . 4
barPlots . . . 5
boot_group_validation . . . 5
boot_group_validation.mc . . . 7
calcCalinskiHarabasz . . . 9
calcDaviesBouldin . . . 10
calcELSA . . . 11
calcexplainedInertia . . . 12
calcFukuyamaSugeno . . . 13
calcFuzzyELSA . . . 14
calcGD43 . . . 15
calcGD53 . . . 16
calcNegentropyI . . . 17
calcqualityIndexes . . . 18
calcSilhouetteIdx . . . 19
calcUncertaintyIndex . . . 20
calc_local_moran_raster . . . 21
calc_moran_raster . . . 21
cat_to_belongings . . . 22
circular_window . . . 22
CMeans . . . 23
GCMeans . . . 25
groups_matching . . . 27
is.FCMres . . . 28
LyonIris . . . 29
mapClusters . . . 30
plot.FCMres . . . 31
predict.FCMres . . . 32
predict_membership . . . 33
print.FCMres . . . 34
select_parameters . . . 35

select_parameters.mc . . . 38
SFCMeans . . . 41
SGFCMeans . . . 44
spatialDiag . . . 47
spConsistency . . . 48
spiderPlots . . . 50
sp_clust_explorer . . . 51
standardizer . . . 52
summarizeClusters . . . 52
summary.FCMres . . . 53
uncertaintyMap . . . 54
undecidedUnits . . . 55
violinPlots . . . 56

Index **57**

`adjustSpatialWeights` *Semantic adjusted spatial weights*

Description

Function to adjust the spatial weights so that they represent semantic distances between neighbours

Usage

```
adjustSpatialWeights(data, listw, style, mindist = 1e-11)
```

Arguments

`data`	A dataframe with numeric columns
`listw`	A nb object from spdep
`style`	A letter indicating the weighting scheme (see spdep doc)
`mindist`	A minimum value for distance between two observations. If two neighbours have exactly the same values, then the euclidean distance between them is 0, leading to an infinite spatial weight. In that case, the minimum distance is used instead of 0.

Value

A listw object (spdep like)

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
Wqueen2 <- adjustSpatialWeights(dataset,queen,style="C")
```

Arcachon *SpatRaster of the bay of Arcachon*

Description

A Landsat 8 image of the bay of Arcachon (France), with a resolution of 30mx30m and 6 bands: blue, green, red, near infrared, shortwave infrared 1 and shortwave infrared 2. The dataset is saved as a Large RasterBrick with the package raster and has the following crs: EPSG:32630. It is provided as a tiff file.

Usage

```
load_arcachon()
```

Format

A spaRast with 6 bands

blue wavelength: 0.45-0.51

green wavelength: 0.53-0.59

red wavelength: 0.64-0.67

near infrared wavelength: 0.85-0.88

shortwave infrared wavelength: 1.57-1.65

shortwave infrared wavelength: 2.11-2.29

Source

https://earthexplorer.usgs.gov/

Examples

```
# loading directly from file
Arcachon <- terra::rast(system.file("extdata/Littoral4_2154.tif", package = "geocmeans"))
names(Arcachon) <- c("blue", "green", "red", "infrared", "SWIR1", "SWIR2")

# loading with the provided function
Arcachon <- load_arcachon()
```

barPlots *Bar plots*

Description

Return bar plots to compare groups

Usage

```
barPlots(data, belongmatrix, ncol = 3, what = "mean")
```

Arguments

data	A dataframe with numeric columns
belongmatrix	A membership matrix
ncol	An integer indicating the number of columns for the bar plot
what	Can be "mean" (default) or "median"

Value

a barplot created with ggplot2

Examples

```
## Not run:
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
barPlots(dataset, result$Belongings)

## End(Not run)
```

boot_group_validation *Check the robustness of a classification by Bootstrap*

Description

Check that the obtained groups are stable by bootstrap

Usage

```
boot_group_validation(
  object,
  nsim = 1000,
  maxiter = 1000,
  tol = 0.01,
  init = "random",
  verbose = TRUE,
  seed = NULL
)
```

Arguments

object	A FCMres object, typically obtained from functions CMeans, GCMeans, SFCMeans, SGFCMeans
nsim	The number of replications to do for the bootstrap evaluation
maxiter	An integer for the maximum number of iterations
tol	The tolerance criterion used in the evaluateMatrices function for convergence assessment
init	A string indicating how the initial centres must be selected. "random" indicates that random observations are used as centres "kpp" use a distance-based method resulting in more dispersed centres at the beginning. Both of them are heuristic.
verbose	A boolean to specify if the progress bar should be displayed.
seed	An integer used for random number generation. It ensures that the starting centres will be the same if the same value is selected.

Details

Considering that the classification produced by a FCM like algorithm depends on its initial state, it is important to check if the groups obtained are stable. This function uses a bootstrap method to do so. During a selected number of iterations (at least 1000), a sample of size n (with replacement) is drawn from the original dataset. For each sample, the same classification algorithm is applied and the results are compared with the reference results. For each original group, the most similar group is identified by calculating the Jaccard similarity index between the columns of the two membership matrices. This index is comprised between 0 (exact difference) and 1 (perfect similarity) and a value is calculated for each group at each iteration. One can investigate the values obtained to determine if the groups are stable. Values under 0.5 are a concern and indicate that the group is dissolving. Values between 0.6 and 0.75 indicate a pattern in the data, but a significant uncertainty. Values above 0.8 indicate strong groups. The values of the centres obtained at each iteration are also returned, it is important to ensure that they approximately follow a normal distribution (or are at least unimodal).

Value

A list of two values: group_consistency: a dataframe indicating the consistency across simulations each cluster ; group_centres: a list with a dataframe for each cluster. The values in the dataframes are the centres of the clusters at each simulation.

Examples

```
## Not run:
data(LyonIris)

#selecting the columns for the analysis
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14",
                   "Pct_65","Pct_Img","TxChom1564","Pct_brevet","NivVieMed")

#rescaling the columns
Data <- sf::st_drop_geometry(LyonIris[AnalysisFields])
for (Col in names(Data)){
  Data[[Col]] <- as.numeric(scale(Data[[Col]]))
}

Cmean <- CMeans(Data,4,1.5,500,standardize = FALSE, seed = 456,
    tol = 0.00001, verbose = FALSE)

validation <- boot_group_validation(Cmean, nsim = 1000, maxiter = 1000,
    tol = 0.01, init = "random")

## End(Not run)
```

---

boot_group_validation.mc

Check that the obtained groups are stable by bootstrap (multicore)

Description

Check that the obtained groups are stable by bootstrap with multicore support

Usage

```
boot_group_validation.mc(
  object,
  nsim = 1000,
  maxiter = 1000,
  tol = 0.01,
  init = "random",
  verbose = TRUE,
  seed = NULL
)
```

Arguments

object	A FCMres object, typically obtained from functions CMeans, GCMeans, SFCMeans, SGFCMeans
nsim	The number of replications to do for the bootstrap evaluation
maxiter	An integer for the maximum number of iterations

tol	The tolerance criterion used in the evaluateMatrices function for convergence assessment
init	A string indicating how the initial centres must be selected. "random" indicates that random observations are used as centres. "kpp" use a distance based method resulting in more dispersed centres at the beginning. Both of them are heuristic.
verbose	A boolean to specify if the progress bar should be displayed.
seed	An integer to control randomness, default is NULL

Details

For more details, see the documentation of the function boot_group_validation

Value

A list of two values: group_consistency: a dataframe indicating the consistency across simulations each cluster ; group_centres: a list with a dataframe for each cluster. The values in the dataframes are the centres of the clusters at each simulation.

Examples

```
## Not run:
data(LyonIris)

#selecting the columns for the analysis
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14",
                   "Pct_65","Pct_Img","TxChom1564","Pct_brevet","NivVieMed")

#rescaling the columns
Data <- sf::st_drop_geometry(LyonIris[AnalysisFields])
for (Col in names(Data)){
  Data[[Col]] <- as.numeric(scale(Data[[Col]]))
}

Cmean <- CMeans(Data,4,1.5,500,standardize = FALSE, seed = 456,
    tol = 0.00001, verbose = FALSE)

future::plan(future::multisession(workers=2))

validation <- boot_group_validation.mc(Cmean, nsim = 1000, maxiter = 1000,
    tol = 0.01, init = "random")
## make sure any open connections are closed afterward
if (!inherits(future::plan(), "sequential")) future::plan(future::sequential)

## End(Not run)
```

calcCalinskiHarabasz *Calinski-Harabasz index*

Description

Calculate the Calinski-Harabasz index of clustering quality.

Usage

```
calcCalinskiHarabasz(data, belongmatrix, centers)
```

Arguments

data	The original dataframe used for the clustering (n*p)
belongmatrix	A membership matrix (n*k)
centers	The centres of the clusters

Details

The Calinski-Harabasz index (Da Silva et al. 2020) is the ratio between the clusters separation (between groups sum of squares) and the clusters cohesion (within groups sum of squares). A greater value indicates either more separated clusters or more cohesive clusters.

Value

A float: the Calinski-Harabasz index

References

Da Silva LEB, Melton NM, Wunsch DC (2020). "Incremental cluster validity indices for online learning of hard partitions: Extensions and comparative study." *IEEE Access*, **8**, 22025–22047.

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
calcCalinskiHarabasz(result$Data, result$Belongings, result$Centers)
```

`calcDaviesBouldin` *Davies-Bouldin index*

Description

Calculate the Davies-Bouldin index of clustering quality.

Usage

```
calcDaviesBouldin(data, belongmatrix, centers)
```

Arguments

`data`	The original dataframe used for the clustering (n*p)
`belongmatrix`	A membership matrix (n*k)
`centers`	The centres of the clusters

Details

The Davies-Bouldin index (Da Silva et al. 2020) can be seen as the ratio of the within cluster dispersion and the between cluster separation. A lower value indicates a higher cluster compacity or a higher cluster separation. The formula is:

$$DB = \frac{1}{k}\sum_{i=1}^{k} R_i$$

with:

$$R_i = \max_{i \neq j}\left(\frac{S_i + S_j}{M_{i,j}}\right)$$

$$S_l = \left[\frac{1}{n_l}\sum_{l=1}^{n} \|\boldsymbol{x_l} - \boldsymbol{c_i}\| * u_i\right]^{\frac{1}{2}}$$

$$M_{i,j} = \sum \|\boldsymbol{c}_i - \boldsymbol{c}_j\|$$

So, the value of the index is an average of R_i values. For each cluster, they represent its worst comparison with all the other clusters, calculated as the ratio between the compactness of the two clusters and the separation of the two clusters.

Value

A float: the Davies-Bouldin index

References

Da Silva LEB, Melton NM, Wunsch DC (2020). "Incremental cluster validity indices for online learning of hard partitions: Extensions and comparative study." *IEEE Access*, **8**, 22025–22047.

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
calcDaviesBouldin(result$Data, result$Belongings, result$Centers)
```

---

calcELSA *calculate ELSA statistic for a hard partition*

Description

Calculate ELSA statistic for a hard partition. This local indicator of spatial autocorrelation can be used to determine where observations belong to different clusters.

Usage

```
calcELSA(object, nblistw = NULL, window = NULL, matdist = NULL)
```

Arguments

object	A FCMres object, typically obtained from functions CMeans, GCMeans, SFCMeans, SGFCMeans. Can also be a vector of categories. This vector must be filled with integers starting from 1. -1 can be used to indicate missing categories.
nblistw	A list.w object describing the neighbours typically produced by the spdep package. Required if data is a dataframe, see the parameter window if you use a list of rasters as input.
window	A binary (0,1) matrix representing the neighbours spatial weights when working with rasters. The matrix must have odd dimensions.
matdist	A matrix representing the dissimilarity between the clusters. The matrix must be squared and the diagonal must be filled with zeros.

Details

The ELSA index (Naimi et al. 2019) can be used to measure local autocorrelation for a categorical variable. It varies between 0 and 1, 0 indicating a perfect positive spatial autocorrelation and 1 a perfect heterogeneity. It is based on the Shanon entropy index, and uses a measure of difference between categories. Thus it can reflect that proximity of two similar categories is still a form of positive autocorelation. The authors suggest to calculate the mean of the index at several lag distance to create an entrogram which quantifies global spatial structure and can be represented as a variogram-like graph.

Value

A depending of the input, a vector of ELSA values or a raster with the ELSA values.

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
elsa_valus <- calcELSA(result)
```

---

calcexplainedInertia *Explained inertia index*

Description

Calculate the explained inertia by a classification

Usage

```
calcexplainedInertia(data, belongmatrix)
```

Arguments

data	The original dataframe used for the classification (n*p)
belongmatrix	A membership matrix (n*k)

Value

A float: the percentage of the total inertia explained

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
calcexplainedInertia(result$Data,result$Belongings)
```

`calcFukuyamaSugeno` *Fukuyama and Sugeno index*

Description

Calculate Fukuyama and Sugeno index of clustering quality

Usage

```
calcFukuyamaSugeno(data, belongmatrix, centers, m)
```

Arguments

`data`	The original dataframe used for the clustering (n*p)
`belongmatrix`	A membership matrix (n*k)
`centers`	The centres of the clusters
`m`	The fuzziness parameter

Details

The Fukuyama and Sugeno index (Fukuyama 1989) is the difference between the compacity of clusters and the separation of clusters. A smaller value indicates a better clustering. The formula is:

$$S(c) = \sum_{k=1}^{n} \sum_{i=1}^{c} (U_{ik})^m \left(\|x_k - v_i\|^2 - \|v_i - \bar{x}\|^2 \right) 2$$

with *n* the number of observations, *k* the number of clusters and $\bar{x}$ the mean of the dataset.

Value

A float: the Fukuyama and Sugeno index

References

Fukuyama Y (1989). "A new method of choosing the number of clusters for the fuzzy c-mean method." In *Proc. 5th Fuzzy Syst. Symp., 1989*, 247–250.

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
calcFukuyamaSugeno(result$Data,result$Belongings, result$Centers, 1.5)
```

`calcFuzzyELSA` *calculate ELSA statistic for a fuzzy partition*

Description

Calculate ELSA statistic for a fuzzy partition. This local indicator of spatial autocorrelation can be used to identify areas where close observations tend to belong to different clusters.

Usage

```
calcFuzzyELSA(object, nblistw = NULL, window = NULL, matdist = NULL)
```

Arguments

`object`	A FCMres object, typically obtained from functions CMeans, GCMeans, SFCMeans, SGFCMeans. Can also be a membership matrix. Each row of this matrix must sum up to 1. Can also be a list of rasters, in which case each raster must represent the membership values for one cluster and the sum of all the rasters must be a raster filled with ones.
`nblistw`	A list.w object describing the neighbours typically produced by the spdep package. Required if data is a dataframe, see the parameter window if you use a list of rasters as input.
`window`	A binary (0,1) matrix representing the neighbours spatial weights when working with rasters. The matrix must have odd dimensions.
`matdist`	A matrix representing the dissimilarity between the clusters. The matrix must be squared and the diagonal must be filled with zeros.

Details

The fuzzy ELSA index is a generalization of the ELSA index (Naimi et al. 2019). It can be used to measure local autocorrelation for a membership matrix. It varies between 0 and 1, 0 indicating a perfect positive spatial autocorrelation and 1 a perfect heterogeneity. It is based on the Shannon entropy index, and uses a measure of dissimilarity between categories.

Value

either a vector or a raster with the ELSA values.

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
elsa_valus <- calcFuzzyELSA(result)
```

`calcGD43` *Generalized Dunn's index (43)*

Description

Calculate the Generalized Dunn's index (v43) of clustering quality.

Usage

```
calcGD43(data, belongmatrix, centers)
```

Arguments

`data`	The original dataframe used for the clustering (n*p)
`belongmatrix`	A membership matrix (n*k)
`centers`	The centres of the clusters

Details

The Generalized Dunn's index (Da Silva et al. 2020) is a ratio of the worst pair-wise separation of clusters and the worst compactness of clusters. A higher value indicates a better clustering. The formula is:

$$GD_{rs} = \frac{\min_{i \neq j} \left[\delta_r\left(\omega_i, \omega_j\right)\right]}{\max_k \left[\Delta_s\left(\omega_k\right)\right]}$$

The numerator is a measure of the minimal separation between all the clusters *i* and *j* given by the formula:

$$\delta_r\left(\omega_i, \omega_j\right) = \|\boldsymbol{c}_i - \boldsymbol{c}_j\|$$

which is basically the Euclidean distance between the centres of clusters c_i and c_j

The denominator is a measure of the maximal dispersion of all clusters, given by the formula:

$$\frac{2 * \sum_{l=1}^{n} \|\boldsymbol{x}_l - \boldsymbol{c_i}\|^{\frac{1}{2}}}{\sum u_i}$$

Value

A float: the Generalized Dunn's index (43)

References

Da Silva LEB, Melton NM, Wunsch DC (2020). "Incremental cluster validity indices for online learning of hard partitions: Extensions and comparative study." *IEEE Access*, **8**, 22025–22047.

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
calcGD43(result$Data, result$Belongings, result$Centers)
```

`calcGD53` *Generalized Dunn's index (53)*

Description

Calculate the Generalized Dunn's index (v53) of clustering quality.

Usage

```
calcGD53(data, belongmatrix, centers)
```

Arguments

`data`	The original dataframe used for the clustering (n*p)
`belongmatrix`	A membership matrix (n*k)
`centers`	The centres of the clusters

Details

The Generalized Dunn's index (Da Silva et al. 2020) is a ratio of the worst pair-wise separation of clusters and the worst compactness of clusters. A higher value indicates a better clustering. The formula is:

$$GD_{rs} = \frac{\min_{i \neq j} \left[\delta_r\left(\omega_i, \omega_j\right)\right]}{\max_k \left[\Delta_s\left(\omega_k\right)\right]}$$

The numerator is a measure of the minimal separation between all the clusters *i* and *j* given by the formula:

$$\delta_r\left(\omega_i, \omega_j\right) = \frac{\sum_{l=1}^{n} \left\|\boldsymbol{x_l} - \boldsymbol{c_i}\right\|^{\frac{1}{2}} .u_{il} + \sum_{l=1}^{n} \left\|\boldsymbol{x_l} - \boldsymbol{c_j}\right\|^{\frac{1}{2}} .u_{jl}}{\sum u_i + \sum u_j}$$

where *u* is the membership matrix and u_i is the column of *u* describing the membership of the *n* observations to cluster *i*. c_i is the center of the cluster *i*.

The denominator is a measure of the maximal dispersion of all clusters, given by the formula:

$$\frac{2 * \sum_{l=1}^{n} \left\|\boldsymbol{x}_l - \boldsymbol{c_i}\right\|^{\frac{1}{2}}}{\sum u_i}$$

Value

A float: the Generalized Dunn's index (53)

References

Da Silva LEB, Melton NM, Wunsch DC (2020). "Incremental cluster validity indices for online learning of hard partitions: Extensions and comparative study." *IEEE Access*, **8**, 22025–22047.

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
calcGD53(result$Data, result$Belongings, result$Centers)
```

---

calcNegentropyI *Negentropy Increment index*

Description

Calculate the Negentropy Increment index of clustering quality.

Usage

```
calcNegentropyI(data, belongmatrix, centers)
```

Arguments

data	The original dataframe used for the clustering (n*p)
belongmatrix	A membership matrix (n*k)
centers	The centres of the clusters

Details

The Negentropy Increment index (Da Silva et al. 2020) is based on the assumption that a normally shaped cluster is more desirable. It uses the difference between the average negentropy of all the clusters in the partition, and that of the whole partition. A smaller value indicates a better partition. The formula is:

$$NI = \frac{1}{2}\sum_{j=1}^{k} p_i \ln|\mathbf{\Sigma}_j| - \frac{1}{2}\ln|\mathbf{\Sigma}_{data}| - \sum_{j=1}^{k} p_j \ln p_j$$

with a cluster, |.| the determinant of a matrix,

- *j* a cluster
- |.| the determinant of a matrix
- $|\mathbf{\Sigma}_j|$ the covariance matrix of the dataset weighted by the membership values to cluster *j*
- $|\mathbf{\Sigma}_{data}|$ the covariance matrix of the dataset
- p_j the sum of the membership values to cluster *j* divided by the number of observations.

Value

A float: the Negentropy Increment index

References

Da Silva LEB, Melton NM, Wunsch DC (2020). "Incremental cluster validity indices for online learning of hard partitions: Extensions and comparative study." *IEEE Access*, **8**, 22025–22047.

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
calcNegentropyI(result$Data, result$Belongings, result$Centers)
```

---

calcqualityIndexes *Quality indexes*

Description

calculate several clustering quality indexes (some of them come from fclust package)

Usage

```
calcqualityIndexes(
  data,
  belongmatrix,
  m,
  indices = c("Silhouette.index", "Partition.entropy", "Partition.coeff",
    "XieBeni.index", "FukuyamaSugeno.index", "Explained.inertia")
)
```

Arguments

data	The original dataframe used for the classification (n*p)
belongmatrix	A membership matrix (n*k)
m	The fuzziness parameter used for the classification
indices	A character vector with the names of the indices to calculate, default is : c("Silhouette.index", "Partition.entropy", "Partition.coeff", "XieBeni.index", "FukuyamaSugeno.index", "Explained.inertia"). Other available indices are : "DaviesBoulin.index", "CalinskiHarabasz.index", "GD43.index", "GD53.index" and "Negentropy.index"

Value

A named list with with the values of the required indices

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
calcqualityIndexes(result$Data,result$Belongings, m=1.5)
```

---

calcSilhouetteIdx *Fuzzy Silhouette index*

Description

Calculate the Silhouette index of clustering quality.

Usage

```
calcSilhouetteIdx(data, belongings)
```

Arguments

data	The original dataframe used for the clustering (n*p)
belongings	A membership matrix (n*k)

Details

The index is calculated with the function SIL.F from the package fclust. When the dataset is too large, an approach by subsampling is used to avoid crash.

Value

A float, the fuzzy Silhouette index

`calcUncertaintyIndex` *Diversity index*

Description

Calculate the diversity (or entropy) index.

Usage

```
calcUncertaintyIndex(belongmatrix)
```

Arguments

`belongmatrix`	A membership matrix

Details

The diversity (or entropy) index (Theil 1972) is calculated for each observation an varies between 0 and 1. When the value is close to 0, the observation belong to only one cluster (as in hard clustering). When the value is close to 1, the observation is undecided and tends to belong to each cluster. Values above 0.9 should be investigated. The formula is:

$$H2_i = \frac{-\sum[u_{ij}\ln(u_{ij})]}{\ln(k)}$$

with *i* and observation, *j* a cluster, *k* the number of clusters and *u* the membership matrix.

It is a simplified formula because the sum of each row of a membership matrix is 1.

Value

A vector with the values of the diversity (entropy) index

References

Theil H (1972). *Statistical decomposition analysis; with applications in the social and administrative sciences*. North-Holland.

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
calcUncertaintyIndex(result$Belongings)
```

calc_local_moran_raster *Local Moran I for raster*

Description

Calculate the Local Moran I for a numeric raster

Usage

```
calc_local_moran_raster(rast, window)
```

Arguments

rast	A SpatRaster or a matrix
window	The window defining the neighbour weights

Value

A SpatRaster or a matrix depending on the input with the local Moran I values

Examples

```
Arcachon <- terra::rast(system.file("extdata/Littoral4_2154.tif", package = "geocmeans"))
names(Arcachon) <- c("blue", "green", "red", "infrared", "SWIR1", "SWIR2")
rast <- Arcachon[[1]]
w <- matrix(1, nrow = 3, ncol = 3)
calc_local_moran_raster(rast, w)
```

calc_moran_raster *Global Moran I for raster*

Description

Calculate the global Moran I for a numeric raster

Usage

```
calc_moran_raster(rast, window)
```

Arguments

rast	A SpatRaster or a matrix
window	The window defining the neighbour weights

Value

A float: the global Moran I

Examples

```
Arcachon <- terra::rast(system.file("extdata/Littoral4_2154.tif", package = "geocmeans"))
names(Arcachon) <- c("blue", "green", "red", "infrared", "SWIR1", "SWIR2")
rast <- Arcachon[[1]]
w <- matrix(1, nrow = 3, ncol = 3)
calc_moran_raster(rast, w)
```

---

cat_to_belongings *Convert categories to membership matrix*

Description

Function to convert a character vector to a membership matrix (binary matrix). The columns of the matrix are ordered with the order function.

Usage

```
cat_to_belongings(categories)

catToBelongings(categories)
```

Arguments

categories	A vector with the categories of each observation

Value

A binary matrix

---

circular_window *Circular window*

Description

Create a matrix that can be used as a window when working with rasters. It uses a radius to set to 0 the weights of pixels that are farther than this distance. This is helpful to create circular focals.

Usage

```
circular_window(radius, res)
```

Arguments

radius	The size in metres of the radius of the circular focal
res	The width in metres of a pixel. It is assumed that pixels are squares.

Details

The original function comes from here: https://scrogster.wordpress.com/2012/10/05/applying-a-circular-moving-window-filter-to-raster-data-in-r/ but we reworked it to make it faster and to ensure that the result is a matrix with odd dimensions.

Value

A binary weight matrix

Examples

```
# wide of 100 metres for pixels of 2 metres
window <- circular_window(100, 2)
# row standardisation
window_row_std <- window / sum(window)
```

CMeans *C-means*

Description

The classical c-mean algorithm

Usage

```
CMeans(
  data,
  k,
  m,
  maxiter = 500,
  tol = 0.01,
  standardize = TRUE,
  robust = FALSE,
  noise_cluster = FALSE,
  delta = NULL,
  verbose = TRUE,
  init = "random",
  seed = NULL
)
```

Arguments

data	A dataframe with only numerical variables. Can also be a list of rasters (produced by the package raster). In that case, each raster is considered as a variable and each pixel is an observation. Pixels with NA values are not used during the classification.
k	An integer describing the number of cluster to find
m	A float for the fuzziness degree
maxiter	An integer for the maximum number of iterations
tol	The tolerance criterion used in the evaluateMatrices function for convergence assessment
standardize	A boolean to specify if the variables must be centred and reduced (default = True)
robust	A boolean indicating if the "robust" version of the algorithm must be used (see details)
noise_cluster	A boolean indicatong if a noise cluster must be added to the solution (see details)
delta	A float giving the distance of the noise cluster to each observation
verbose	A boolean to specify if the progress should be printed
init	A string indicating how the initial centres must be selected. "random" indicates that random observations are used as centres. "kpp" use a distance-based method resulting in more dispersed centres at the beginning. Both of them are heuristic.
seed	An integer used for random number generation. It ensures that the starting centres will be the same if the same value is selected.

Value

An S3 object of class FCMres with the following slots

- Centers: a dataframe describing the final centers of the groups
- Belongings: the final membership matrix
- Groups: a vector with the names of the most likely group for each observation
- Data: the dataset used to perform the clustering (might be standardized)
- isRaster: TRUE if rasters were used as input data, FALSE otherwise
- k: the number of groups
- m: the fuzyness degree
- alpha: the spatial weighting parameter (if SFCM or SGFCM)
- beta: beta parameter for generalized version of FCM (GFCM or SGFCM)
- algo: the name of the algorithm used
- rasters: a list of rasters with membership values and the most likely group (if rasters were used)
- missing: a boolean vector indicating raster cell with data (TRUE) and with NA (FALSE) (if rasters were used)

- maxiter: the maximum number of iterations used
- tol: the convergence criterio
- lag_method: the lag function used (if SFCM or SGFCM)
- nblistw: the neighbours list used (if vector data were used for SFCM or SGFCM)
- window: the window used (if raster data were used for SFCM or SGFCM)

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
result <- CMeans(dataset,k = 5, m = 1.5, standardize = TRUE)
```

GCMeans *Generalized C-means*

Description

The generalized c-mean algorithm

Usage

```
GCMeans(
  data,
  k,
  m,
  beta,
  maxiter = 500,
  tol = 0.01,
  standardize = TRUE,
  robust = FALSE,
  noise_cluster = FALSE,
  delta = NULL,
  verbose = TRUE,
  init = "random",
  seed = NULL
)
```

Arguments

data	A dataframe with only numerical variables. Can also be a list of rasters (produced by the package raster). In that case, each raster is considered as a variable and each pixel is an observation. Pixels with NA values are not used during the classification.
k	An integer describing the number of cluster to find

m	A float for the fuzziness degree
beta	A float for the beta parameter (control speed convergence and classification crispness)
maxiter	An integer for the maximum number of iterations
tol	The tolerance criterion used in the evaluateMatrices function for convergence assessment
standardize	A boolean to specify if the variables must be centred and reduced (default = True)
robust	A boolean indicating if the "robust" version of the algorithm must be used (see details)
noise_cluster	A boolean indicatong if a noise cluster must be added to the solution (see details)
delta	A float giving the distance of the noise cluster to each observation
verbose	A boolean to specify if the progress should be printed
init	A string indicating how the initial centres must be selected. "random" indicates that random observations are used as centres. "kpp" use a distance-based method resulting in more dispersed centres at the beginning. Both of them are heuristic.
seed	An integer used for random number generation. It ensures that the starting centres will be the same if the same value is selected.

Value

An S3 object of class FCMres with the following slots

- Centers: a dataframe describing the final centers of the groups
- Belongings: the final membership matrix
- Groups: a vector with the names of the most likely group for each observation
- Data: the dataset used to perform the clustering (might be standardized)
- isRaster: TRUE if rasters were used as input data, FALSE otherwise
- k: the number of groups
- m: the fuzyness degree
- alpha: the spatial weighting parameter (if SFCM or SGFCM)
- beta: beta parameter for generalized version of FCM (GFCM or SGFCM)
- algo: the name of the algorithm used
- rasters: a list of rasters with membership values and the most likely group (if rasters were used)
- missing: a boolean vector indicating raster cell with data (TRUE) and with NA (FALSE) (if rasters were used)
- maxiter: the maximum number of iterations used
- tol: the convergence criterio
- lag_method: the lag function used (if SFCM or SGFCM)
- nblistw: the neighbours list used (if vector data were used for SFCM or SGFCM)
- window: the window used (if raster data were used for SFCM or SGFCM)

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
result <- GCMeans(dataset,k = 5, m = 1.5, beta = 0.5, standardize = TRUE)
```

---

groups_matching *Match the groups obtained from two classifications*

Description

Match the groups obtained from two classifications based on the Jaccard index calculated on the membership matrices.

Usage

```
groups_matching(object.x, object.y)
```

Arguments

object.x	A FCMres object, or a simple membership matrix. It is used as the reference for the ordering of the groups
object.y	A FCMres object, or a simple membership matrix. The order of its groups will be updated to match with the groups of object.x

Details

We can not expect to obtain the groups in the same order in each run of a classification algorithm. This function can be used match the clusters of a first classification with the most similar clusters in a second classification. Thus it might be easier to compare the results of two algorithms or two runs of the same algorithm.

Value

The FCMres object or the membership matrix provided for the parameter object.y with the order of the groups updated.

Examples

```
data(LyonIris)

#selecting the columns for the analysis
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14",
                   "Pct_65","Pct_Img","TxChom1564","Pct_brevet","NivVieMed")

#rescaling the columns
Data <- sf::st_drop_geometry(LyonIris[AnalysisFields])
```

```
for (Col in names(Data)){
  Data[[Col]] <- as.numeric(scale(Data[[Col]]))
}

Cmean <- CMeans(Data,4,1.5,500,standardize = FALSE, seed = 456, tol = 0.00001, verbose = FALSE)
Cmean2 <- CMeans(Data,4,1.5,500,standardize = FALSE, seed = 789, tol = 0.00001, verbose = FALSE)
ordered_Cmean2 <- groups_matching(Cmean,Cmean2)
```

`is.FCMres` *is method for FCMres*

Description

Check if an object can be considered as a FCMres object

Usage

```
## S3 method for class 'FCMres'
is(object, class2 = "FCMres")
```

Arguments

`object`	A FCMres object, typically obtained from functions CMeans, GCMeans, SFCMeans, SGFCMeans
`class2`	Character string giving the names of the classe to test (usually "FCMres")

Value

A boolean, TRUE if x can be considered as a FCMres object, FALSE otherwise group

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
is(result, "FCMres")
```

`LyonIris` *social and environmental indicators for the Iris of the metropolitan region of Lyon (France)*

Description

A dataset containing social and environmental data for the Iris of Lyon (France)

Usage

```
LyonIris
```

Format

A SpatialPolygonsDataFrame with 506 rows and 32 variables:

OBJECTID a simple OID (integer)

INSEE_COM the code of each commune (factor)

CODE_IRIS the code of each unit area : iris (factor)

Lden the annual daily mean noise exposure values in dB (numeric)

NO2 the annual mean of NO2 concentration in ug/m3 (numeric)

PM25 the annual mean of PM25 concentration in ug/m3 (numeric)

PM10 the annual mean of PM25 concentration in ug/m3 (numeric)

Pct0_14 the percentage of people that are 0 to 14 year old (numeric)

Pct_65 the percentage of people older than 64 (numeric)

Pct_Img the percentage immigrants (numeric)

TxChom1564 the unemployment rate (numeric)

Pct_brevet the percentage of people that obtained the college diploma (numeric)

NivVieMed the median standard of living in euros (numeric)

VegHautPrt the percentage of the iris surface covered by trees (numeric)

X the X coordinate of the center of the Iris (numeric)

Y the Y coordinate of the center of the Iris (numeric) ...

Source

https://data.grandlyon.com/portail/fr/accueil

mapClusters *Mapping the clusters*

Description

Build some maps to visualize the results of the clustering

Usage

```
mapClusters(geodata = NULL, object, undecided = NULL)
```

Arguments

geodata	An object of class features collection from sf / ordered like the original data used for the clustering. Can be Null if object is a FCMres and has been created with rasters.
object	A FCMres object, typically obtained from functions CMeans, GCMeans, SFCMeans, SGFCMeans. Can also be a simple membership matrix.
undecided	A float between 0 and 1 giving the minimum value that an observation must get in the membership matrix to not be considered as uncertain (default = NULL)

Value

A named list with :

- ProbaMaps : a list of tmap maps showing for each group the probability of the observations to belong to that group
- ClusterMap : a tmap map showing the most likely group for observation

Examples

```
## Not run:
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
MyMaps <- mapClusters(LyonIris, result$Belongings)

## End(Not run)
```

plot.FCMres *Plot method for FCMres object*

Description

Method to plot the results of a FCM.res object

Usage

```
## S3 method for class 'FCMres'
plot(x, type = "spider", ...)
```

Arguments

x	A FCMres object, typically obtained from functions CMeans, GCMeans, SFCMeans, SGFCMeans. Can also be a simple membership matrix.
type	A string indicating the type of plot to show. Can be one of "bar", "violin", or "spider". Default is spider.
...	not used

Details

This S3 method is a simple dispatcher for the functions barPlots, violinPlots and spiderPlots. To be able to use all their specific parameters, one can use them directly.

Value

a ggplot2 object, a list, or NULL, depending on the type of plot requested

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")

# rescaling all the variables used in the analysis
for (field in AnalysisFields) {
    LyonIris[[field]] <- scale(LyonIris[[field]])
}

# doing the initial clustering
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SGFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, beta = 0.5, standardize = FALSE)

plot(result, type = "spider")
```

predict.FCMres *Predict method for FCMres object*

Description

Function to predict the membership matrix of a new set of observations

Usage

```
## S3 method for class 'FCMres'
predict(
  object,
  new_data,
  nblistw = NULL,
  window = NULL,
  standardize = TRUE,
  ...
)
```

Arguments

object	A FCMres object, typically obtained from functions CMeans, GCMeans, SFCMeans, SGFCMeans. Can also be a simple membership matrix.
new_data	A DataFrame with the new observations
nblistw	A list.w object describing the neighbours typically produced by the spdep package. Required if data is a dataframe, see the parameter window if you use a list of rasters as input.
window	If data is a list of rasters, then a window must be specified instead of a list.w object. It will be used to calculate a focal function on each raster. The window must be a square numeric matrix with odd dimensions (such 3x3). The values in the matrix indicate the weight to give to each pixel and the centre of the matrix is the centre of the focal function.
standardize	A boolean to specify if the variable must be centred and reduced (default = True)
...	not used

Value

A numeric matrix with the membership values for each new observation

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")

# rescaling all the variables used in the analysis
```

```
for (field in AnalysisFields) {
    LyonIris[[field]] <- scale(LyonIris[[field]])
}

# doing the initial clustering
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SGFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, beta = 0.5, standardize = FALSE)

# using a subset of the original dataframe as "new data"
new_data <- LyonIris[c(1, 27, 36, 44, 73),]
new_dataset <- sf::st_drop_geometry(new_data[AnalysisFields])
new_nb <- spdep::poly2nb(new_data,queen=TRUE)
new_Wqueen <- spdep::nb2listw(new_nb,style="W")

# doing the prediction
predictions <- predict(result, new_dataset, new_Wqueen, standardize = FALSE)
```

`predict_membership` *Predict matrix membership for new observations*

Description

Function to predict the membership matrix of a new set of observations

Usage

```
predict_membership(
  object,
  new_data,
  nblistw = NULL,
  window = NULL,
  standardize = TRUE,
  ...
)
```

Arguments

`object`	A FCMres object, typically obtained from functions CMeans, GCMeans, SFCMeans, SGFCMeans. Can also be a simple membership matrix.
`new_data`	A DataFrame with the new observations or a list of rasters if object$isRaster is TRUE
`nblistw`	A list.w object describing the neighbours typically produced by the spdep package. Required if data is a dataframe, see the parameter window if you use a list of rasters as input.

window	If data is a list of rasters, then a window must be specified instead of a list.w object. It will be used to calculate a focal function on each raster. The window must be a square numeric matrix with odd dimensions (such 3x3). The values in the matrix indicate the weight to give to each pixel and the centre of the matrix is the centre of the focal function.
standardize	A boolean to specify if the variable must be centered and reduced (default = True)
...	not used

Value

A numeric matrix with the membership values for each new observation. If rasters were used, return a list of rasters with the membership values.

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")

# rescaling all the variables used in the analysis
for (field in AnalysisFields) {
    LyonIris[[field]] <- scale(LyonIris[[field]])
}

# doing the initial clustering
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SGFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, beta = 0.5, standardize = FALSE)

# using a subset of the original dataframe as "new data"
new_data <- LyonIris[c(1, 27, 36, 44, 73),]
new_dataset <- sf::st_drop_geometry(new_data[AnalysisFields])
new_nb <- spdep::poly2nb(new_data,queen=TRUE)
new_Wqueen <- spdep::nb2listw(new_nb,style="W")

# doing the prediction
predictions <- predict_membership(result, new_dataset, new_Wqueen, standardize = FALSE)
```

---

print.FCMres *print method for FCMres*

Description

print a FCMres object

Usage

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

Arguments

x	A FCMres object, typically obtained from functions CMeans, GCMeans, SFCMeans, SGFCMeans
...	not used

Value

A boolean, TRUE if x can be considered as a FCMres object, FALSE otherwise group

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
result <- CMeans(dataset, k = 5, m = 1.5, standardize = TRUE)
print(result, "FCMres")
```

---

select_parameters *Select parameters for a clustering algorithm*

Description

Function to select the parameters for a clustering algorithm.

Usage

```
select_parameters(
  algo,
  data,
  k,
  m,
  alpha = NA,
  beta = NA,
  nblistw = NULL,
  lag_method = "mean",
  window = NULL,
  spconsist = TRUE,
  classidx = TRUE,
  nrep = 30,
  indices = NULL,
  standardize = TRUE,
```

```
  robust = FALSE,
  noise_cluster = FALSE,
  delta = NA,
  maxiter = 500,
  tol = 0.01,
  seed = NULL,
  init = "random",
  verbose = TRUE
)

selectParameters(
  algo,
  data,
  k,
  m,
  alpha = NA,
  beta = NA,
  nblistw = NULL,
  lag_method = "mean",
  window = NULL,
  spconsist = TRUE,
  classidx = TRUE,
  nrep = 30,
  indices = NULL,
  standardize = TRUE,
  robust = FALSE,
  noise_cluster = FALSE,
  delta = NA,
  maxiter = 500,
  tol = 0.01,
  seed = NULL,
  init = "random",
  verbose = TRUE
)
```

Arguments

algo	A string indicating which method to use (FCM, GFCM, SFCM, SGFCM)
data	A dataframe with numeric columns or a list of rasters.
k	A sequence of values for k to test (>=2)
m	A sequence of values for m to test
alpha	A sequence of values for alpha to test (NULL if not required)
beta	A sequence of values for beta to test (NULL if not required)
nblistw	A list of list.w objects describing the neighbours typically produced by the spdep package (NULL if not required)
lag_method	A string indicating if a classical lag must be used ("mean") or if a weighted median must be used ("median"). Both can be tested by specifying a vector :

	c("mean","median"). When working with rasters, the string must be parsable to a function like mean, min, max, sum, etc. and will be applied to all the pixels values in the window designated by the parameter window and weighted according to the values of this matrix.
window	A list of windows to use to calculate neighbouring values if rasters are used.
spconsist	A boolean indicating if the spatial consistency must be calculated
classidx	A boolean indicating if the quality of classification indices must be calculated
nrep	An integer indicating the number of permutation to do to simulate the random distribution of the spatial inconsistency. Only used if spconsist is TRUE.
indices	A character vector with the names of the indices to calculate, to evaluate clustering quality. default is :c("Silhouette.index", "Partition.entropy", "Partition.coeff", "XieBeni.index", "FukuyamaSugeno.index", "Explained.inertia"). Other available indices are : "DaviesBoulin.index", "CalinskiHarabasz.index", "GD43.index", "GD53.index" and "Negentropy.index".
standardize	A boolean to specify if the variable must be centered and reduce (default = True)
robust	A boolean indicating if the "robust" version of the algorithm must be used (see details)
noise_cluster	A boolean indicatong if a noise cluster must be added to the solution (see details)
delta	A float giving the distance of the noise cluster to each observation
maxiter	An integer for the maximum number of iteration
tol	The tolerance criterion used in the evaluateMatrices function for convergence assessment
seed	An integer used for random number generation. It ensures that the start centers will be the same if the same integer is selected.
init	A string indicating how the initial centers must be selected. "random" indicates that random observations are used as centers. "kpp" use a distance based method resulting in more dispersed centers at the beginning. Both of them are heuristic.
verbose	A boolean indicating if a progressbar should be displayed

Value

A dataframe with indicators assessing the quality of classifications

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
#set spconsist to TRUE to calculate the spatial consistency indicator
#FALSE here to reduce the time during package check
values <- select_parameters(algo = "SFCM", dataset, k = 5, m = seq(2,3,0.1),
```

```
    alpha = seq(0,2,0.1), nblistw = Wqueen, spconsist=FALSE)


data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
#set spconsist to TRUE to calculate the spatial consistency indicator
#FALSE here to reduce the time during package check
values <- selectParameters(algo = "SFCM", dataset, k = 5, m = seq(2,3,0.1),
    alpha = seq(0,2,0.1), nblistw = Wqueen, spconsist=FALSE)
```

`select_parameters.mc` *Select parameters for clustering algorithm (multicore)*

Description

Function to select the parameters for a clustering algorithm. This version of the function allows to use a plan defined with the package future to reduce calculation time.

Usage

```
select_parameters.mc(
  algo,
  data,
  k,
  m,
  alpha = NA,
  beta = NA,
  nblistw = NULL,
  lag_method = "mean",
  window = NULL,
  spconsist = TRUE,
  classidx = TRUE,
  nrep = 30,
  indices = NULL,
  standardize = TRUE,
  robust = FALSE,
  noise_cluster = FALSE,
  delta = NA,
  maxiter = 500,
  tol = 0.01,
  chunk_size = 5,
  seed = NULL,
  init = "random",
```

```
  verbose = TRUE
)

selectParameters.mc(
  algo,
  data,
  k,
  m,
  alpha = NA,
  beta = NA,
  nblistw = NULL,
  lag_method = "mean",
  window = NULL,
  spconsist = TRUE,
  classidx = TRUE,
  nrep = 30,
  indices = NULL,
  standardize = TRUE,
  robust = FALSE,
  noise_cluster = FALSE,
  delta = NA,
  maxiter = 500,
  tol = 0.01,
  chunk_size = 5,
  seed = NULL,
  init = "random",
  verbose = TRUE
)
```

Arguments

algo	A string indicating which method to use (FCM, GFCM, SFCM, SGFCM)
data	A dataframe with numeric columns
k	A sequence of values for k to test (>=2)
m	A sequence of values for m to test
alpha	A sequence of values for alpha to test (NULL if not required)
beta	A sequence of values for beta to test (NULL if not required)
nblistw	A list of list.w objects describing the neighbours typically produced by the spdep package (NULL if not required)
lag_method	A string indicating if a classical lag must be used ("mean") or if a weighted median must be used ("median"). Both can be tested by specifying a vector : c("mean","median"). When working with rasters, the string must be parsable to a function like mean, min, max, sum, etc. and will be applied to all the pixels values in the window designated by the parameter window and weighted according to the values of this matrix.
window	A list of windows to use to calculate neighbouring values if rasters are used.

spconsist	A boolean indicating if the spatial consistency must be calculated
classidx	A boolean indicating if the quality of classification indices must be calculated
nrep	An integer indicating the number of permutation to do to simulate the random distribution of the spatial inconsistency. Only used if spconsist is TRUE.
indices	A character vector with the names of the indices to calculate, to evaluate clustering quality. default is :c("Silhouette.index", "Partition.entropy", "Partition.coeff", "XieBeni.index", "FukuyamaSugeno.index", "Explained.inertia"). Other available indices are : "DaviesBoulin.index", "CalinskiHarabasz.index", "GD43.index", "GD53.index" and "Negentropy.index".
standardize	A boolean to specify if the variable must be centered and reduce (default = True)
robust	A boolean indicating if the "robust" version of the algorithm must be used (see details)
noise_cluster	A boolean indicatong if a noise cluster must be added to the solution (see details)
delta	A float giving the distance of the noise cluster to each observation
maxiter	An integer for the maximum number of iteration
tol	The tolerance criterion used in the evaluateMatrices function for convergence assessment
chunk_size	The size of a chunk used for multiprocessing. Default is 100.
seed	An integer used for random number generation. It ensures that the start centers will be the same if the same integer is selected.
init	A string indicating how the initial centers must be selected. "random" indicates that random observations are used as centers. "kpp" use a distance based method resulting in more dispersed centers at the beginning. Both of them are heuristic.
verbose	A boolean indicating if a progressbar should be displayed

Value

A dataframe with indicators assessing the quality of classifications

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
future::plan(future::multisession(workers=2))
#set spconsist to TRUE to calculate the spatial consistency indicator
#FALSE here to reduce the time during package check
values <- select_parameters.mc("SFCM", dataset, k = 5, m = seq(1,2.5,0.1),
    alpha = seq(0,2,0.1), nblistw = Wqueen, spconsist=FALSE)
## make sure any open connections are closed afterward
if (!inherits(future::plan(), "sequential")) future::plan(future::sequential)
```

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
future::plan(future::multisession(workers=2))
#set spconsist to TRUE to calculate the spatial consistency indicator
#FALSE here to reduce the time during package check
values <- select_parameters.mc("SFCM", dataset, k = 5, m = seq(1,2.5,0.1),
    alpha = seq(0,2,0.1), nblistw = Wqueen, spconsist=FALSE)
```

SFCMeans *SFCMeans*

Description

spatial version of the c-mean algorithm (SFCMeans, FCM_S1)

Usage

```
SFCMeans(
  data,
  nblistw = NULL,
  k,
  m,
  alpha,
  lag_method = "mean",
  window = NULL,
  noise_cluster = FALSE,
  delta = NULL,
  maxiter = 500,
  tol = 0.01,
  standardize = TRUE,
  robust = FALSE,
  verbose = TRUE,
  init = "random",
  seed = NULL
)
```

Arguments

data	A dataframe with only numerical variables. Can also be a list of rasters (produced by the package raster). In that case, each raster is considered as a variable and each pixel is an observation. Pixels with NA values are not used during the classification.

nblistw	A list.w object describing the neighbours typically produced by the spdep package. Required if data is a dataframe, see the parameter window if you use a list of rasters as input.
k	An integer describing the number of cluster to find
m	A float for the fuzziness degree
alpha	A float representing the weight of the space in the analysis (0 is a typical fuzzy-c-mean algorithm, 1 is balanced between the two dimensions, 2 is twice the weight for space)
lag_method	A string indicating if a classical lag must be used ("mean") or if a weighted median must be used ("median"). When working with rasters, a function can be given (or a string which will be parsed). It will be applied to all the pixels values in the matrix designated by the parameter window and weighted according to the values of this matrix. Typically, to obtain an average of the pixels in a 3x3 matrix one could use the function sum (or "sum") and set the window as: window <- matrix(1/9,nrow = 3, ncol = 3). There is one special case when working with rasters: one can specify "nl" (standing for non-local) which calculated a lagged version of the input rasters, using the inverse of the euclidean distance as spatial weights (see the section Advanced examples in the vignette introduction for more details).
window	If data is a list of rasters, then a window must be specified instead of a list.w object. It will be used to calculate a focal function on each raster. The window must be a square numeric matrix with odd dimensions (such 3x3). The values in the matrix indicate the weight to give to each pixel and the centre of the matrix is the centre of the focal function.
noise_cluster	A boolean indicatong if a noise cluster must be added to the solution (see details)
delta	A float giving the distance of the noise cluster to each observation
maxiter	An integer for the maximum number of iterations
tol	The tolerance criterion used in the evaluateMatrices function for convergence assessment
standardize	A boolean to specify if the variables must be centred and reduced (default = True)
robust	A boolean indicating if the "robust" version of the algorithm must be used (see details)
verbose	A boolean to specify if the progress should be printed
init	A string indicating how the initial centres must be selected. "random" indicates that random observations are used as centres. "kpp" use a distance-based method resulting in more dispersed centres at the beginning. Both of them are heuristic.
seed	An integer used for random number generation. It ensures that the starting centres will be the same if the same value is selected.

Details

The implementation is based on the following article : doi:10.1016/j.patcog.2006.07.011.

the membership matrix (u) is calculated as follow

$$u_{ik} = \frac{(||x_k - v_i||^2 + \alpha||\bar{x_k} - v_i||^2)^{(-1/(m-1))}}{\sum_{j=1}^{c}(||x_k - v_j||^2 + \alpha||\bar{x_k} - v_j||^2)^{(-1/(m-1))}}$$

the centers of the groups are updated with the following formula

$$v_i = \frac{\sum_{k=1}^{N} u_{ik}^m (x_k + \alpha\bar{x_k})}{(1+\alpha)\sum_{k=1}^{N} u_{ik}^m}$$

with

- vi the center of the group vi
- xk the data point k
- xk_bar the spatially lagged data point k

Value

An S3 object of class FCMres with the following slots

- Centers: a dataframe describing the final centers of the groups
- Belongings: the final membership matrix
- Groups: a vector with the names of the most likely group for each observation
- Data: the dataset used to perform the clustering (might be standardized)
- isRaster: TRUE if rasters were used as input data, FALSE otherwise
- k: the number of groups
- m: the fuzyness degree
- alpha: the spatial weighting parameter (if SFCM or SGFCM)
- beta: beta parameter for generalized version of FCM (GFCM or SGFCM)
- algo: the name of the algorithm used
- rasters: a list of rasters with membership values and the most likely group (if rasters were used)
- missing: a boolean vector indicating raster cell with data (TRUE) and with NA (FALSE) (if rasters were used)
- maxiter: the maximum number of iterations used
- tol: the convergence criterio
- lag_method: the lag function used (if SFCM or SGFCM)
- nblistw: the neighbours list used (if vector data were used for SFCM or SGFCM)
- window: the window used (if raster data were used for SFCM or SGFCM)

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
```

SGFCMeans *SGFCMeans*

Description

spatial version of the generalized c-mean algorithm (SGFCMeans)

Usage

```
SGFCMeans(
  data,
  nblistw = NULL,
  k,
  m,
  alpha,
  beta,
  lag_method = "mean",
  window = NULL,
  maxiter = 500,
  tol = 0.01,
  standardize = TRUE,
  robust = FALSE,
  noise_cluster = FALSE,
  delta = NULL,
  verbose = TRUE,
  init = "random",
  seed = NULL
)
```

Arguments

data	A dataframe with only numerical variables. Can also be a list of rasters (produced by the package raster). In that case, each raster is considered as a variable and each pixel is an observation. Pixels with NA values are not used during the classification.
nblistw	A list.w object describing the neighbours typically produced by the spdep package. Required if data is a dataframe, see the parameter window if you use a list of rasters as input.

k	An integer describing the number of cluster to find
m	A float for the fuzziness degree
alpha	A float representing the weight of the space in the analysis (0 is a typical fuzzy-c-mean algorithm, 1 is balanced between the two dimensions, 2 is twice the weight for space)
beta	A float for the beta parameter (control speed convergence and classification crispness)
lag_method	A string indicating if a classical lag must be used ("mean") or if a weighted median must be used ("median"). When working with rasters, a function can be given (or a string which will be parsed). It will be applied to all the pixels values in the matrix designated by the parameter window and weighted according to the values of this matrix. Typically, to obtain an average of the pixels in a 3x3 matrix one could use the function sum (or "sum") and set the window as: window <- matrix(1/9,nrow = 3, ncol = 3). There is one special case when working with rasters: one can specify "nl" (standing for non-local) which calculated a lagged version of the input rasters, using the inverse of the euclidean distance as spatial weights (see the section Advanced examples in the vignette introduction for more details).
window	If data is a list of rasters, then a window must be specified instead of a list.w object. It will be used to calculate a focal function on each raster. The window must be a square numeric matrix with odd dimensions (such 3x3). The values in the matrix indicate the weight to give to each pixel and the centre of the matrix is the centre of the focal function.
maxiter	An integer for the maximum number of iterations
tol	The tolerance criterion used in the evaluateMatrices function for convergence assessment
standardize	A boolean to specify if the variables must be centred and reduced (default = True)
robust	A boolean indicating if the "robust" version of the algorithm must be used (see details)
noise_cluster	A boolean indicatong if a noise cluster must be added to the solution (see details)
delta	A float giving the distance of the noise cluster to each observation
verbose	A boolean to specify if the progress should be printed
init	A string indicating how the initial centres must be selected. "random" indicates that random observations are used as centres. "kpp" use a distance-based method resulting in more dispersed centres at the beginning. Both of them are heuristic.
seed	An integer used for random number generation. It ensures that the starting centres will be the same if the same value is selected.

Details

The implementation is based on the following article : doi:10.1016/j.dsp.2012.09.016.

the membership matrix (u) is calculated as follow

$$u_{ik} = \frac{(||x_k - v_i||^2 - b_k + \alpha||\bar{x_k} - v_i||^2)^{(-1/(m-1))}}{\sum_{j=1}^{c}(||x_k - v_j||^2 - b_k + \alpha||\bar{x_k} - v_j||^2)^{(-1/(m-1))}}$$

the centers of the groups are updated with the following formula

$$v_i = \frac{\sum_{k=1}^{N} u_{ik}^m (x_k + \alpha\bar{x_k})}{(1+\alpha)\sum_{k=1}^{N} u_{ik}^m}$$

with

- vi the center of the group vi
- xk the data point k
- xk_bar the spatially lagged data point k

$$b_k = \beta \times min(||x_k - v||)$$

Value

An S3 object of class FCMres with the following slots

- Centers: a dataframe describing the final centers of the groups
- Belongings: the final membership matrix
- Groups: a vector with the names of the most likely group for each observation
- Data: the dataset used to perform the clustering (might be standardized)
- isRaster: TRUE if rasters were used as input data, FALSE otherwise
- k: the number of groups
- m: the fuzyness degree
- alpha: the spatial weighting parameter (if SFCM or SGFCM)
- beta: beta parameter for generalized version of FCM (GFCM or SGFCM)
- algo: the name of the algorithm used
- rasters: a list of rasters with membership values and the most likely group (if rasters were used)
- missing: a boolean vector indicating raster cell with data (TRUE) and with NA (FALSE) (if rasters were used)
- maxiter: the maximum number of iterations used
- tol: the convergence criterio
- lag_method: the lag function used (if SFCM or SGFCM)
- nblistw: the neighbours list used (if vector data were used for SFCM or SGFCM)
- window: the window used (if raster data were used for SFCM or SGFCM)

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SGFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, beta = 0.5, standardize = TRUE)
```

---

spatialDiag *Spatial diagnostic*

Description

Utility function to facilitate the spatial diagnostic of a classification

Calculate the following indicators: Moran I index (spdep::moranI) for each column of the membership matrix, Join count test (spdep::joincount.multi) for the most likely groups of each datapoint, Spatial consistency index (see function spConsistency) and the Elsa statistic (see function calcElsa). Note that if the FCMres object given was constructed with rasters, the joincount statistic is not calculated and no p-values are provided for the Moran I indices.

Usage

```
spatialDiag(
  object,
  nblistw = NULL,
  window = NULL,
  undecided = NULL,
  matdist = NULL,
  nrep = 50
)
```

Arguments

object	A FCMres object, typically obtained from functions CMeans, GCMeans, SFCMeans, SGFCMeans. Can also be a simple membership matrix.
nblistw	A list.w object describing the neighbours typically produced by the spdep package. Required if data is a dataframe, see the parameter window if you use a list of rasters as input. Can also be NULL if object is a FCMres object.
window	If rasters were used for the classification, the window must be specified instead of a list.w object. Can also be NULL if object is a FCMres object.
undecided	A float giving the threslhod to detect undecided observations. An observation is undecided if its maximum membership value is bellow this float. If null, no observations are undecided.
matdist	A matrix representing the dissimilarity between the clusters. The matrix must be squared and the diagonal must be filled with zeros.

nrep An integer indicating the number of permutation to do to simulate the random distribution of the spatial inconsistency

Value

A named list with :

- MoranValues : the moran I values for each column of the membership matrix (spdep::MoranI)
- JoinCounts : the result of the join count test calculated with the most likely group for each datapoint (spdep::joincount.multi)
- SpConsist : the mean value of the spatial consistency index (the lower, the better, see ?spConsistency for details)

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
spatialDiag(result, undecided=0.45, nrep=30)
```

---

spConsistency *Spatial consistency index*

Description

Calculate a spatial consistency index

Usage

```
spConsistency(
  object,
  nblistw = NULL,
  window = NULL,
  nrep = 999,
  adj = FALSE,
  mindist = 1e-11
)
```

Arguments

object A FCMres object, typically obtained from functions CMeans, GCMeans, SFCMeans, SGFCMeans. Can also be a simple membership matrix.

nblistw	A list.w object describing the neighbours typically produced by the spdep package. Required if data is a dataframe, see the parameter window if you use a list of rasters as input. Can also be NULL if object is a FCMres object.
window	if rasters were used for the classification, the window must be specified instead of a list.w object. Can also be NULL if object is a FCMres object.
nrep	An integer indicating the number of permutation to do to simulate spatial randomness. Note that if rasters are used, each permutation can be very long.
adj	A boolean indicating if the adjusted version of the indicator must be calculated when working with rasters (globally standardized). When working with vectors, see the function adjustSpatialWeights to modify the list.w object.
mindist	When adj is true, a minimum value for distance between two observations. If two neighbours have exactly the same values, then the euclidean distance between them is 0, leading to an infinite spatial weight. In that case, the minimum distance is used instead of 0.

Details

This index is experimental, it aims to measure how much a clustering solution is spatially consistent. A classification is spatially inconsistent if neighbouring observation do not belong to the same group. See detail for a description of its calculation

The total spatial inconsistency (*Scr*) is calculated as follow

$$isp = \sum_i \sum_j \sum_k (u_{ik} - u_{jk})^2 * W_{ij}$$

With U the membership matrix, i an observation, k the neighbours of i and W the spatial weight matrix This represents the total spatial inconsistency of the solution (true inconsistency) We propose to compare this total with simulated values obtained by permutations (simulated inconsistency). The values obtained by permutation are an approximation of the spatial inconsistency obtained in a random context Ratios between the true inconsistency and simulated inconsistencies are calculated A value of 0 depict a situation where all observations are identical to their neighbours A value of 1 depict a situation where all observations are as much different as their neighbours that what randomness can produce A classification solution able to reduce this index has a better spatial consistency

Value

A named list with

- Mean : the mean of the spatial consistency index
- prt05 : the 5th percentile of the spatial consistency index
- prt95 : the 95th percentile of the spatial consistency index
- samples : all the value of the spatial consistency index
- sum_diff : the total sum of squarred difference between observations and their neighbours

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
# NOTE : more replications are needed for proper inference
spConsistency(result$Belongings, nblistw = Wqueen, nrep=25)
```

---

spiderPlots *Spider chart*

Description

Display spider charts to quickly compare values between groups

Usage

```
spiderPlots(data, belongmatrix, chartcolors = NULL)
```

Arguments

data	A dataframe with numeric columns
belongmatrix	A membership matrix
chartcolors	A vector of color names used for the spider plot

Details

For each group, the weighted mean of each variable in data is calculated based on the probability of belonging to this group of each observation. On the chart the exterior ring represents the maximum value obtained for all the groups and the interior ring the minimum. The groups are located between these two limits in a linear way.

Value

NULL, the plots are displayed directly by the function (see fmsb::radarchart)

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
spiderPlots(dataset,result$Belongings)
```

`sp_clust_explorer` *Classification result explorer*

Description

Start a local Shiny App to explore the results of a classification

Usage

```
sp_clust_explorer(
  object = NULL,
  spatial = NULL,
  membership = NULL,
  dataset = NULL,
  port = 8100,
  ...
)
```

Arguments

`object`	A FCMres object, typically obtained from functions CMeans, GCMeans, SFCMeans, SGFCMeans
`spatial`	A feature collection (sf) used to map the observations. Only needed if object was not created from rasters.
`membership`	A matrix or a dataframe representing the membership values obtained for each observation. If NULL, then the matrix is extracted from object.
`dataset`	A dataframe or matrix representing the data used for the classification. If NULL, then the matrix is extracted from object.
`port`	An integer of length 4 indicating the port on which to start the Shiny app. Default is 8100
`...`	Other parameters passed to the function runApp

Examples

```
## Not run:
data(LyonIris)

#selecting the columns for the analysis
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14",
                   "Pct_65","Pct_Img","TxChom1564","Pct_brevet","NivVieMed")

#rescaling the columns
Data <- sf::st_drop_geometry(LyonIris[AnalysisFields])
for (Col in names(Data)){
  Data[[Col]] <- as.numeric(scale(Data[[Col]]))
}
```

```
Cmean <- CMeans(Data,4,1.5,500,standardize = FALSE, seed = 456, tol = 0.00001, verbose = FALSE)

sp_clust_explorer(Cmean, LyonIris)

## End(Not run)
```

---

standardizer *Standardizing helper*

Description

Create functions to standardize and unstandardize data

Usage

```
standardizer(x)
```

Arguments

x	a numeric vector or a data.frame with only numeric columns. Non numeric columns are dropped.

Value

If x was a vector, the function returns a list containing two functions : scale and unscale. The first one is an equivalent of the classical function scale(x, center = TRUE, scale = TRUE). The second can be used to reverse the scaling and get back original units. If x was a data.frame, the same pair of functions is returned inside of a list for each numeric column.

Examples

```
data(LyonIris)
LyonScales <- standardizer(sf::st_drop_geometry(LyonIris))
```

---

summarizeClusters *Descriptive statistics by group*

Description

Calculate some descriptive statistics of each group

Usage

```
summarizeClusters(data, belongmatrix, weighted = TRUE, dec = 3, silent = TRUE)
```

Arguments

data	The original dataframe used for the classification
belongmatrix	A membership matrix
weighted	A boolean indicating if the summary statistics must use the membership matrix columns as weights (TRUE) or simply assign each observation to its most likely cluster and compute the statistics on each subset (FALSE)
dec	An integer indicating the number of digits to keep when rounding (default is 3)
silent	A boolean indicating if the results must be printed or silently returned

Value

A list of length k (the number of group). Each element of the list is a dataframe with summary statistics for the variables of data for each group

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
summarizeClusters(dataset, result$Belongings)
```

---

summary.FCMres *Summary method for FCMres*

Description

Calculate some descriptive statistics of each group of a FCMres object

Usage

```
## S3 method for class 'FCMres'
summary(object, data = NULL, weighted = TRUE, dec = 3, silent = TRUE, ...)
```

Arguments

object	A FCMres object, typically obtained from functions CMeans, GCMeans, SFCMeans, SGFCMeans
data	A dataframe to use for the summary statistics instead of obj$data
weighted	A boolean indicating if the summary statistics must use the membership matrix columns as weights (TRUE) or simply assign each observation to its most likely cluster and compute the statistics on each subset (FALSE)
dec	An integer indicating the number of digits to keep when rounding (default is 3)
silent	A boolean indicating if the results must be printed or silently returned
...	Not used

Value

A list of length k (the number of group). Each element of the list is a dataframe with summary statistics for the variables of data for each group

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
summary(result)
```

---

uncertaintyMap *Uncertainty map*

Description

Return a map to visualize membership matrix

Usage

```
uncertaintyMap(
  geodata,
  belongmatrix,
  njit = 150,
  radius = NULL,
  colors = NULL,
  pt_size = 0.05
)
```

Arguments

geodata	An object of class feature collection from sf ordered like the original data used for the clustering.
belongmatrix	A membership matrix
njit	The number of points to map on each feature.
radius	When mapping points, the radius indicates how far random points will be plotted around the original features.
colors	A vector of colors to use for the groups.
pt_size	A float giving the size of the random points on the final map (default is 0.05)

Details

This function maps the membership matrix by plotting random points in polygons, along lines or around points representing the original observations. Each cluster is associated with a color and each random point has a probability to be of that color equal to the membership value of the feature it belongs itself. Thus, it is possible to visualize regions with uncertainty and to identify the strongest clusters.

Value

a map created with tmap

Examples

```
## Not run:
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
  "TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
uncertaintyMap(LyonIris, result$Belongings)

## End(Not run)
```

`undecidedUnits` *Undecided observations*

Description

Identify the observation for which the classification is uncertain

Usage

```
undecidedUnits(belongmatrix, tol = 0.1, out = "character")
```

Arguments

`belongmatrix`	The membership matrix obtained at the end of the algorithm
`tol`	A float indicating the minimum required level of membership to be not considered as undecided
`out`	The format of the output vector. Default is "character". If "numeric", then the undecided units are set to -1.

Value

A vector indicating the most likely group for each observation or "Undecided" if the maximum probability for the observation does not reach the value of the tol parameter

Examples

```
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometry(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
undecidedUnits(result$Belongings, tol = 0.45)
```

`violinPlots` *Violin plots*

Description

Return violin plots to compare the distribution of each variable for each group.

Usage

```
violinPlots(data, groups)
```

Arguments

`data`	A dataframe with numeric columns
`groups`	A vector indicating the group of each observation

Value

A list of plots created with ggplot2

Examples

```
## Not run:
data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- sf::st_drop_geometrie(LyonIris[AnalysisFields])
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
result <- SFCMeans(dataset, Wqueen,k = 5, m = 1.5, alpha = 1.5, standardize = TRUE)
violinPlots(dataset, result$Groups)

## End(Not run)
```

Index

∗ **datasets**
    LyonIris, 29

adjustSpatialWeights, 3
Arcachon, 4

barPlots, 5
boot_group_validation, 5
boot_group_validation.mc, 7

calc_local_moran_raster, 21
calc_moran_raster, 21
calcCalinskiHarabasz, 9
calcDaviesBouldin, 10
calcELSA, 11
calcexplainedInertia, 12
calcFukuyamaSugeno, 13
calcFuzzyELSA, 14
calcGD43, 15
calcGD53, 16
calcNegentropyI, 17
calcqualityIndexes, 18
calcSilhouetteIdx, 19
calcUncertaintyIndex, 20
cat_to_belongings, 22
catToBelongings *(*cat_to_belongings*)*, 22
circular_window, 22
CMeans, 23

GCMeans, 25
groups_matching, 27

is.FCMres, 28

load_arcachon *(*Arcachon*)*, 4
LyonIris, 29

mapClusters, 30

plot.FCMres, 31
predict.FCMres, 32
predict_membership, 33
print.FCMres, 34

select_parameters, 35
select_parameters.mc, 38
selectParameters *(*select_parameters*)*, 35
selectParameters.mc
        *(*select_parameters.mc*)*, 38
SFCMeans, 41
SGFCMeans, 44
sp_clust_explorer, 51
spatialDiag, 47
spConsistency, 48
spiderPlots, 50
standardizer, 52
summarizeClusters, 52
summary.FCMres, 53

uncertaintyMap, 54
undecidedUnits, 55

violinPlots, 56