Package ‘geocmeans’

April 21, 2021

Type Package

Title Implementing Methods for Spatial Fuzzy Unsupervised Classification

Version 0.1.1

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

Imports ggplot2 (>= 3.2.1), spdep (>= 1.1.2), reldist (>= 1.6.6),
dplyr (>= 0.8.3), fclust (>= 2.1.1), fmsb (>= 0.7.0), broom (>= 0.5.2), future.apply (>= 1.4.0), progressr (>= 0.4.0), reshape2 (>= 1.4.4), sp (>= 1.4-4), stats (>= 3.5)

Depends R (>= 3.5)

Suggests knitr (>= 1.28), rmarkdown (>= 2.1), markdown (>= 1.1),
maptools (>= 0.9-5), rgeos (>= 0.5-2), 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), sf(>= 0.9-8)

License GPL-2

Encoding UTF-8

LazyData true

RooxygenNote 7.1.1

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. Indexes for estimating the spatial consistency and classification quality are proposed in addition. The methods were originally proposed in the field of brain imaging (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>) and recently 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

NeedsCompilation no
adjustSpatialWeights

**Description**

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

**Usage**

```r
adjustSpatialWeights(data, listw, style)
```

**Arguments**

- `data` A dataframe with numeric columns
- `listw` A nb object from spdep
- `style` A letter indicating the weighting scheme (see spdep doc)
barPlots

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 <- LyonIris@data[AnalysisFields]
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
Wqueen2 <- adjustSpatialWeights(dataset,queen,style="C")

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

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

*Explained inertia index*

**Description**

Calculate the explained inertia by a classification

**Usage**

```r
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**

```r
data(LyonIris)
AnalysisFields <- c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- LyonIris@data[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**

```r
calcFukuyamaSugeno(data, belongmatrix, centers, m)
```
**calcqualityIndexes**

Arguments

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

Value

A float : the Fukuyama and Sugeno index

Examples

```r
data(LyonIris)
dataset <- LyonIris@data[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)
```

**Description**

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

**Usage**

calcqualityIndexes(data, belongmatrix, m)

Arguments

- `data` : The original dataframe used for the classification (n*p)
- `belongmatrix` : A membership matrix (n*k)
- `m` : The fuzzyness parameter used for the classification

Value

A named list with

- Silhouette.index: the silhouette index (fclust::SIL.F)
- Partition.entropy: the partition entropy index (fclust::PE)
- Partition.coeff: the partition entropy coefficient (fclust::PC)
- Modified.partition.coeff: the modified partition entropy coefficient (fclust::MPC)
• XieBeni.index: the Xie and Beni index (fclust::XB)
• FukuyamaSugeno.index: the Fukuyama and Sugeno index (geocmeans::calcFukuyamaSugeno)
• Explained.inertia: the percentage of total inertia explained by the solution

Examples

data(LyonIris)
dataset <- LyonIris@data[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)

---

cat_to_belongings Convert categories to membership matrix

Description

Function to Convert categories to membership matrix (binary matrix)

Usage

cat_to_belongings(categories)
catToBelongings(categories)

Arguments

categories A vector with the categories of each observation

Value

A binary matrix
CMeans

Description

The classical c-mean algorithm

Usage

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

Arguments

data: A dataframe with only numerical variable
k: An integer describing the number of cluster to find
m: A float for the fuzziness degree
maxiter: A float for the maximum number of iteration
tol: The tolerance criterion used in the evaluateMatrices function for convergence assessment
standardize: A boolean to specify if the variables must be centered and reduced (default = True)
verbose: A boolean to specify if the messages should be displayed
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.
seed: An integer used for random number generation. It ensures that the start centers will be the same if the same integer is selected.

Value

A named list with:

- 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)
Examples

```r
data(LyonIris)
AnalysisFields <- c("Lden","NO2","PM25","VegHautPrt","Pct_0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- LyonIris@data[AnalysisFields]
result <- CMeans(dataset,k = 5, m = 1.5, standardize = TRUE)
```

---

**Description**

The generalized c-mean algorithm

**Usage**

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

**Arguments**

- **data**: A dataframe with only numerical variable
- **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**: A float for the maximum number of iteration
- **tol**: The tolerance criterion used in the evaluateMatrices function for convergence assessment
- **standardize**: A boolean to specify if the variables must be centered and reduced (default = True)
- **verbose**: A boolean to specify if the messages should be displayed
- **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.
- **seed**: An integer used for random number generation. It ensures that the start centers will be the same if the same integer is selected.
Value

A named list with:

- 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)

Examples

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

geocmeans

geocmeans: A package implementing methods for spatially constrained c-means algorithm

Description

The geocmeans package implements a modified c-means algorithm more suited to work with spatial data (characterized by spatial autocorrelation). The spatial information is introduced with a spatial weight matrix \(W(n \times n)\) where \(wij\) indicate the strength of the spatial relationship between the observations \(i\) and \(j\). It is recommended to use a matrix standardized by row (so that the sum of each row is 1). More specifically, the spatial c-means combine the euclidean distance of each observation in the data matrix \(X\) to each center with the euclidean distance of the lagged version of \(X\) by \(W\) (\(WX\)). A parameter \(alpha\) controls for the weight of the lagged matrix. If \(alpha = 0\), then the spatial c-means is equal to a classical c-means. If \(alpha = 1\), then the weights given to \(X\) and \(WX\) are equals. If \(alpha = 2\), then the weight of \(WX\) is twice the one of \(X\) and so on. An index to measure the spatial consistency of a classification is proposed in this package.

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
mapClusters

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/accueil

---

mapClusters | Mapping the clusters

Description

Build some maps to visualize the results of the clustering

Usage

mapClusters(geodata, belongmatrix, undecided = NULL)

Arguments

- **geodata** A object of class spatialpolygonsdataframe / spatiallinesdataframe or spatialpointsdataframe ordered like the original data used for the clustering
- **belongmatrix** The membership matrix obtained at the end of the algorithm
- **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)
select_parameters

Value

A named list with:

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

Examples

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

---

**select_parameters**  
*Select parameters for a clustering algorithm*

Description

Function to select the parameters for a clustering algorithm.

Usage

```r
select_parameters(
  algo,
  data,
  k,
  m,
  alpha = NA,
  beta = NA,
  nblistw = NULL,
  lag_method = "mean",
  spconsist = TRUE,
  classidx = TRUE,
  standardize = TRUE,
  maxiter = 500,
  tol = 0.01,
  seed = NULL,
  verbose = TRUE
)
```

selectParameters(  
  algo,
```
select_parameters

data, 
k, 
m, 
alpha = NA, 
beta = NA, 
nblistw = NULL, 
lag_method = "mean", 
spconsist = TRUE, 
classidx = TRUE, 
standardize = TRUE, 
maxiter = 500, 
tol = 0.01, 
seed = NULL, 
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")
spconsist A boolean indicating if the spatial consistency must be calculated
classidx A boolean indicating if the quality of classification indices must be calculated
standardize A boolean to specify if the variable must be centered and reduce (default = True)
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.
verbose A boolean indicating if a progressbar should be displayed

Value

A dataframe with indicators assessing the quality of classifications
Examples

```r
data(LyonIris)
dataset <- LyonIris@data[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)
```

```r
data(LyonIris)
dataset <- LyonIris@data[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**

```r
select_parameters.mc(
  algo,
  data,
  k,
  m,
  alpha = NA,
  beta = NA,
  nblistw = NULL,
  lag_method = "mean",
  spconsist = TRUE,
  classidx = TRUE,
  standardize = TRUE,
)```

maxiter = 500,
tol = 0.01,
seed = NULL,
chunk_size = 100,
verbose = FALSE
)

selectParameters.mc(
  algo,
data,
k,
m,
alpha = NA,
beta = NA,
nblistw = NULL,
lag_method = "mean",
spconsist = TRUE,
classidx = TRUE,
standardize = TRUE,
maxiter = 500,
tol = 0.01,
seed = NULL,
chunk_size = 100,
verbose = FALSE
)

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")
spconsist
  A boolean indicating if the spatial consistency must be calculated
classidx
  A boolean indicating if the quality of classification indices must be calculated
standardize
  A boolean to specify if the variable must be centered and reduce (default = True)
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.

chunk_size  The size of a chunk used for multiprocessing. Default is 100.

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 <- LyonIris@data[AnalysisFields]
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
future::plan(future::multiprocess(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)

data(LyonIris)
AnalysisFields <- c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- LyonIris@data[AnalysisFields]
queen <- spdep::poly2nb(LyonIris,queen=TRUE)
Wqueen <- spdep::nb2listw(queen,style="W")
future::plan(future::multiprocess(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

Description

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

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

Arguments

data A dataframe with only numerical variable
nblistw A list.w object describing the neighbours typically produced by the spdep package
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")
maxiter An integer for the maximum number of iteration
tol The tolerance criterion used in the evaluateMatrices function for convergence assessment
standardize A boolean to specify if the variable must be centered and reduced (default = True)
verbose A boolean to specify if the progress bar should be displayed
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.
seed An integer used for random number generation. It ensures that the start centers will be the same if the same integer is selected.

Details

The implementation is based on the following article: doi: 10.1016/j.patcog.2006.07.011.
the matrix of belonging ($u$) is calculated as follow

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

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

- $v_i$ the center of the group $vi$
- $x_k$ the data point $k$
- $x_k_{\text{bar}}$ the spatially lagged data point $k$

Value

A named list with

- Centers: a dataframe describing the final centers of the groups
- Belongings: the final bmembership 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)

Examples

data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img","TxChom1564","Pct_brevet","NivVieMed")
dataset <- LyonIris@data[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(  
data,  
nblistw,  
k,  
m,  
alpha,  
beta,  
lag_method = "mean",  
maxiter = 500,  
tol = 0.01,  
standardize = TRUE,  
verbose = TRUE,  
init = "random",  
seed = NULL  )

Arguments

data
A dataframe with only numerical variable

nblistw
A list.w object describing the neighbours typically produced by the spdep package

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")

maxiter
An integer for the maximum number of iteration

tol
The tolerance criterion used in the evaluateMatrices function for convergence assessment

standardize
A boolean to specify if the variable must be centered and reduced (default = True)

verbose
A boolean to specify if the progress bar should be displayed

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.

seed
An integer used for random number generation. It ensures that the start centers will be the same if the same integer is selected.
Details

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

The matrix of belonging ($u$) is calculated as follow:

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

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

- $v_i$ the center of the group $v_i$
- $x_k$ the data point $k$
- $x_k_{\text{bar}}$ the spatially lagged data point $k$

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

Value

A named list with

- 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)

Examples

data(LyonIris)
AnalysisFields <- c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img","TxChom1564","Pct_brevet","NivVieMed")
dataset <- LyonIris@data[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 belonging matrix, Join count test (spdep::joincount.multi) for the most likely groups of each datapoint, Spatial consistency index (see function spConsistency)

Usage

spatialDiag(belongmatrix, nblistw, undecided = NULL, nrep = 50)

Arguments

- **belongmatrix**: A membership matrix
- **nblistw**: A list.w object describing the neighbours (spdep package)
- **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)
- **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 fo 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 <- LyonIris@data[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$Belongings, Wqueen, undecided=0.45, nrep=30)
spConsistency

Description

Calculate a spatial consistency index

Usage

spConsistency(belongmatrix, nblistw, nrep = 999)

Arguments

belongmatrix  A membership matrix
nblistw  A list.w object describing the neighbours (spdep package) observation must get in the membership matrix to not be considered as uncertain (default = NULL)
nrep  An integer indicating the number of permutation to do to simulate

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

\[
isp = \sum_i \sum_j \sum_k (u_{ik} - u_{jk})^2 \ast W_{ij}
\]

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
Examples

data(LyonIris)
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img","TxChom1564","Pct_brevet","NivVieMed")
dataset <- LyonIris@data[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)
spConsistency(result$Belongings, Wqueen, nrep=50)

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 <- LyonIris@data[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)
spConsistency(result$Belongings, Wqueen, nrep=50)

spiderPlots
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 <- LyonIris@data[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)
### undecidedUnits

**Undecided observations**

**Description**

Identify the observation for with the classification is uncertain

**Usage**

```r
undecidedUnits(belongmatrix, tol = 0.1)
```

**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

**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**

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

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

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

data(LyonIris)
AnalysisFields <- c("Lden","NO2","PM25","VegHautPrt","Pct0_14","Pct_65","Pct_Img",
"TxChom1564","Pct_brevet","NivVieMed")
dataset <- LyonIris@data[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)
# Index

* datasets
  LyonIris, 9

adjustSpatialWeights, 2

barPlots, 3

calcexplainedInertia, 4
calcFukuyamaSugeno, 4
calcqualityIndexes, 5
cat_to_belongings, 6
catToBelongings (cat_to_belongings), 6
CMeans, 7

GCMeans, 8
geocmeans, 9

LyonIris, 9

mapClusters, 10

select_parameters, 11
select_parameters.mc, 13
selectParameters (select_parameters), 11
selectParameters.mc
  (select_parameters.mc), 13
SFCMeans, 15
SGFCMeans, 17
spatialDiag, 20
spConsistency, 21
spiderPlots, 22
summarizeClusters, 23

undecidedUnits, 24

violinPlots, 24