Package 'btb'

June 3, 2020

Encoding UTF-8

Type Package

Title Beyond the Border - Kernel Density Estimation for Urban Geography

Description The kernelSmoothing() function allows you to square and smooth geolocated data. It calculates a classical kernel smoothing (conservative) or a geographically weighted median. There are four major call modes of the function.
The first call mode is kernelSmoothing(obs, epsg, cellsize, bandwith) for a classical kernel smoothing and automatic grid.
The second call mode is kernelSmoothing(obs, epsg, cellsize, bandwith, quantiles) for a geographically weighted median and automatic grid.
The third call mode is kernelSmoothing(obs, epsg, cellsize, bandwith, centroids) for a classical kernel smoothing and user grid.
The fourth call mode is kernelSmoothing(obs, epsg, cellsize, bandwith, quantiles, centroids) for a geographically weighted median and user grid.
Geographically weighted summary statistics : a framework for localised exploratory data analysis, C.Brunsdon & al., in Computers, Environment and Urban Systems C.Brunsdon & al. (2002) <doi:10.1016/S0198-9715(01)00009-6>,
Statistical Analysis of Spatial and Spatio-Temporal Point Patterns, Third Edition, Diggle, pp. 83-86, (2003) <doi:10.1080/13658816.2014.937718>.

Version 0.1.30.3

Depends R (>= 3.3.0)

Date 2020-06-03

License GPL (>= 2)

Imports methods, Rcpp (>= 0.11.3), sp, sf, RcppParallel

Suggests cartography

LinkingTo Rcpp, RcppParallel, BH (>= 1.60.0-1), RcppArmadillo

NeedsCompilation yes

RoxygenNote 5.0.1

LazyData true

Repository CRAN

Date/Publication 2020-06-03 14:20:06 UTC

Author Arlindo Dos Santos [cre],
Francois Semecurbe [drt, aut],
Auriane Renaud [ctb],
Cynthia Faivre [ctb],
Thierry Cornely [ctb],
Farida Marouchi [ctb],
Farida Marouchi [ctb]

Maintainer Arlindo Dos Santos <Arlindo.Dos-Santos@insee.fr>

R topics documented:

constituerGrappes 2
constituerMatriceEffectifs 3
dfPrix_SP95_2016 4
dfRestaurantParis 5
dfToGrid 5
kernelSmoothing 6
pixel_france 10
reunion 11

Index **12**

`constituerGrappes` *Quadtree variant function (Variante de la fonction Quadtree)*

Description

Quadtree variant function.

(Variante de la fonction Quadtree.)

Usage

```
constituerGrappes(iNbObsMin, mEffectifs)
```

Arguments

`iNbObsMin`	An `integer` representing the minimum number of elements in each cluster (Un `data.frame` représentant le nombre minimum d'éléments dans chaque grappe.)
`mEffectifs`	staffing matrix for each cell (`matrix`). (matrice des effectifs pour chaque case (`matrix`). .)

Value

Returns a `matrix` with the cluster number for each cell .

(Retourne une `matrix` avec le numéro de grappe pour chaque cellule.)

Author(s)

Psar Analyse Urbaine Insee - Arlindo Dos Santos and Francois Semecurbe

Examples

```
dfObservations <- data.frame(x = c(15, 35, 15, 25, 35, 55, 45, 45, 55, 65, 70, 75, 85, 90,
                                   65, 75, 85, 65, 70, 75, 85, 90, 65, 70, 75)
                           ,  y = c(10, 10, 30, 30, 35, 35, 45, 55, 55, 65, 65, 65, 65, 65,
                                   70, 70, 70, 75, 75, 75, 75, 75, 85, 85, 85)
)

cellSize <- 20L

# calcul de l'indice des observations
# on prend le rectangle englobant
# et on positionne le debut de la numérotation sur la première observation
dfObservations$col <- as.integer(floor((dfObservations$x) / cellSize)
                                 - floor(min(dfObservations$x / cellSize)) + 1)
dfObservations$row <- as.integer(floor((dfObservations$y) / cellSize)
                                 - floor(min(dfObservations$y / cellSize)) + 1)

mEffectifs <- constituerMatriceEffectifs(dfObservations$row - 1, dfObservations$col - 1)

#### matrice des grappes
mGrappes <- constituerGrappes(1, mEffectifs)
```

---

constituerMatriceEffectifs

Function constituting a membership matrix (Fonction constituant une matrice des effectifs)

Description

Function constituting a membership matrix.

(Fonction constituant une matrice des effectifs.)

Usage

```
constituerMatriceEffectifs(vLigneObservation, vColonneObservation)
```

Arguments

`vLigneObservation`
: A `vector` containing the line number of each observation
(Un `vector` contenant le numéro de ligne de chaque observation.)

`vColonneObservation`
: A `vector` containing the column number of each observation
(Un `vector` contenant le numéro de colonne de chaque observation.)

Value

Returns a `matrix` with the number of observations for each cell.

(Retourne une `matrix` avec le nombre d'observations pour chaque cellule.)

Author(s)

Psar Analyse Urbaine Insee - Arlindo Dos Santos and Francois Semecurbe

Examples

```
dfObservations <- data.frame(x = c(15, 35, 15, 25, 35, 55, 45, 45, 55, 65, 70, 75, 85, 90,
                                   65, 75, 85, 65, 70, 75, 85, 90, 65, 70, 75)
                           ,  y = c(10, 10, 30, 30, 35, 35, 45, 55, 55, 65, 65, 65, 65, 65,
                                   70, 70, 70, 75, 75, 75, 75, 75, 85, 85, 85)
)

  cellSize <- 20L

# calcul de l'indice des observations
# on prend le rectangle englobant
# et on positionne le debut de la numérotation sur la première observation
dfObservations$col <- as.integer(floor((dfObservations$x) / cellSize)
                                 - floor(min(dfObservations$x / cellSize)) + 1)
dfObservations$row <- as.integer(floor((dfObservations$y) / cellSize)
                                 - floor(min(dfObservations$y / cellSize)) + 1)

mEffectifs <- constituerMatriceEffectifs(dfObservations$row - 1, dfObservations$col - 1)
```

---

dfPrix_SP95_2016 *Unleaded 95 price in France in 2016 (Prix du sans plomb 95 en france en 2016)*

Description

Unleaded 95 price in France in 2016. source : https://www.prix-carburants.gouv.fr/rubrique/opendata/

(Données carroyées à 200 mètres. source : https://www.prix-carburants.gouv.fr/rubrique/opendata/)

Value

x	longitude
y	latitude
SP95	Unleaded price - prix du sans plomb

Examples

```
data(dfPrix_SP95_2016)
```

dfRestaurantParis *Parisian restaurants (Restaurants parisiens)*

Description

Parisian restaurants data frame. source : https://opendata.paris.fr

(data frame contenant les restaurants parisiens. source : https://opendata.paris.fr)

Value

x	longitude
y	latitude
french	French restaurant
asian	asian restaurant

Examples

```
data(dfRestaurantParis)
```

dfToGrid *Grid function (Fonction de carroyage)*

Description

Function to compute a grid (regular or not) from a data.frame.

(Fonction permettant de générer une grille (régulière ou non) à partir d'un data.frame.)

Usage

```
dfToGrid(df, sEPSG, iCellSize = NULL)
```

Arguments

df	A data.frame with the centroids coordinates of the squares to draw. To generate an irregular grid, a third column wiht each cell size must be provided. (x, y, iCellSize) (Un data.frame comportant les coordonnées des carrés à dessiner. Pour obtenir une grille irrégulière, il faut fournir une troisième colonne indiquant la taille de chaque carreau. (x, y, iCellSize).)
sEPSG	EPSG code of projection (character). For example, the RGF93 / Lambert-93 projection has "2154" code. (code EPSG de la projection (character). Par exemple, la projection RGF93 / Lambert-93 a pour code "2154".)
iCellSize	Cell size of the grid. If this argument is provided, the grid is regular. (Taille des carreaux de la grille. Si cet argument est fourni, la grille est régulière.)

Value

Returns an object of class `sf` and `data.frame`.

(Retourne un objet de classe `sf` et `data.frame`.)

Author(s)

Psar Analyse Urbaine Insee - Thierry Cornely, Laure Genebes, Arlindo Dos Santos, Cynthia Faivre, Auriane Renaud and Francois Semecurbe

Examples

```
library(sf)
# example 1 - regular grid
df <- data.frame(x = c(100, 100, 300, 300, 500), y = c(100, 300, 100, 300, 100))
dfResult <- dfToGrid(df = df, sEPSG = "2154", iCellSize = 200)
# write_sf(obj = dfResult, dsn = "regularGrid.shp", delete_layer = TRUE)

# example 2 - irregular grid
df <- data.frame(x = c(50, 50, 150, 150, 300)
                , y = c(50, 150, 50, 150, 100)
                , iCellSize = c(50, 50, 50, 50, 100))
dfResult <- dfToGrid(df = df, sEPSG = "2154")
# write_sf(obj = dfResult, dsn = "irregularGrid.shp", delete_layer = TRUE)
```

---

kernelSmoothing *Smoothing function (Fonction de lissage)*

Description

Smoothing function with a bisquare kernel or median.

(Fonction de lissage à partir d'un noyau bisquare ou de la médiane.)

Usage

```
# Call mode 1: bisquare kernel smoothing - automatic grid
kernelSmoothing(  dfObservations
                , sEPSG
                , iCellSize
                , iBandwidth
                , vQuantiles = NULL
                , dfCentroids = NULL
                , fUpdateProgress = NULL
                , iNeighbor = NULL
                , iNbObsMin = 250
)

# Call mode 2: median smoothing - automatic grid
```

```
kernelSmoothing(  dfObservations
                , sEPSG
                , iCellSize
                , iBandwidth
                , vQuantiles
                , dfCentroids = NULL
                , fUpdateProgress = NULL
                , iNeighbor = NULL
                , iNbObsMin = 250
)

# Call mode 3: bisquare kernel smoothing - user grid
kernelSmoothing(  dfObservations
                , sEPSG
                , iCellSize
                , iBandwidth
                , vQuantiles = NULL
                , dfCentroids
                , fUpdateProgress = NULL
                , iNeighbor = NULL
                , iNbObsMin = 250
)

# Call mode 4: median smoothing - user grid
kernelSmoothing(  dfObservations
                , sEPSG
                , iCellSize
                , iBandwidth
                , vQuantiles
                , dfCentroids
                , fUpdateProgress = NULL
                , iNeighbor = NULL
                , iNbObsMin = 250
)
```

Arguments

`dfObservations`	A `data.frame` with cartesian geographical coordinates and variables to smooth. (x, y, var1, var2, ...) (Un `data.frame` comportant les coordonnées géographiques cartésiennes (x,y), ainsi que les variables que l'on souhaite lisser. (x, y, var1, var2, ...)
`sEPSG`	EPSG code of projection (`character`). For example, the RGF93 / Lambert-93 projection has "2154" code. (code EPSG de la projection (`character`). Par exemple, la projection RGF93 / Lambert-93 a pour code "2154".)
`iCellSize`	Cell size of the grid (`integer`). The unit of measurement is free. It must be the same as the unit of `iBandwidth` variable.

(Taille des carreaux (integer). Le choix de l'unité de mesure est laissé libre à l'utilisateur. Elle doit seulement être la même que celle de la variable iBandwidth.)

iBandwidth
: Radius of the Kernel Density Estimator (integer). This bandwidth acts as a smoothing parameter, controlling the balance between bias and variance. A large bandwidth leads to a very smooth (i.e. high-bias) density distribution. A small bandwidth leads to an unsmooth (i.e. high-variance) density distribution. The unit of measurement is free. It must be the same as the unit of iCellSize variable.

  (Rayon de lissage de l'estimation d'intensité par noyau (integer). Cette bande-passante se comporte comme un paramètre de lissage, controlant l'équilibre entre biais et variance. Un rayon élevé conduit à une densité tres lissée, avec un biais élevé. Un petit rayon génère une densité peu lissée avec une forte variance. Le choix de l'unité de mesure est laissé libre à l'utilisateur. Elle doit seulement être la même que celle de la variable iCellSize.

vQuantiles
: Percentile vector to calculate. For example c(0.1, 0.25, 0.5) will calculate the first decile, the first quartile and the median.

  (Vecteur des quantiles à calculer. Par exemple c(0.1, 0.25, 0.5) retournera le premier décile, le premier quartile et la mediane.)

dfCentroids
: A data.frame with two columns (x, y) containing coordinates of the user's centroids. The coordinates must be in the same projection than (dfObservations).

  (Un data.frame avec deux colonnes (x, y) contenant les coordonnées des centroides de l'utilisateur. Les coordonnées doivent être dans le même système de coordonnées que (dfObservations).)

fUpdateProgress
: A function to see compute progress.

  (Une function pour voir la progression du calcul.)

iNeighbor
: Technical parameter, leave empty. (integer)

  (Paramètre technique pour calculer l'étendue des points d'estimations, à ne pas remplir. (integer))

iNbObsMin
: Minimum size of constituted grappes for median smoothing. (integer)

  (Taille minimale des grappes constituées pour le lissage "médian" (géographiquement pondéré). (integer))

Details

Returns an object inheriting from the data.frame class. (Retourne un objet qui se comporte comme un data.frame, par heritage.)

- Smoothing covers a set of methods to extract pertinent and structuring information from noisy data. In the field of spatial analysis, and most widely in quantitative geography, smoothing is used to modelise density variations of a population distribution in geographical space. Kernel smoothing methods are widely used. In this method, for each location x, we count the number of events of a process within a distance h of x, and weighted by the square reciprocal of the radius h. We apply a edge-correction to deal with edge-effects. So the method is conservative..

- Le lissage recouvre un ensemble de méthodes pour extraire d'une source de données bruitées une information pertinente et structurante. Dans le champ de l'analyse spatiale et plus largement de la géographie quantitative, le lissage est principalement utilisé pour modéliser les variations de densités d'une distribution de population dans l'espace géographique. On utilise principalement des méthodes de lissage par noyau. Il s'agit ici, pour chaque point x, de comptabliser le nombre d' "évènements" d'un processus à une distance h de ce point, tout en ponderant ce nombre par l'inverse de la distance h au carré. On applique une correction à la ponderation afin de traiter les effets de bord. Cette méthode est conservative.

Author(s)

Psar Analyse Urbaine Insee - Thierry Cornely, Laure Genebes, Arlindo Dos Santos, Cynthia Faivre, Auriane Renaud and Francois Semecurbe

References

- "Geographically weighted summary statistics : a framework for localised exploratory data analysis", C.Brunsdon & al., in Computers, Environment and Urban Systems 2002
- Statistical Analysis of Spatial and Spatio-Temporal Point Patterns, Third Edition, Diggle, 2003, pp. 83-86

Examples

```
## Not run:
######### example 1 #########
data(dfPrix_SP95_2016)
dfPrix_SP95_2016$nbObs <- 1L
dfSmoothed <- kernelSmoothing(dfObservations = dfPrix_SP95_2016
                              , sEPSG = "2154"
                              , iCellSize = 5000L
                              , iBandwidth = 30000L)
dfSmoothed$prix95 <- dfSmoothed$SP95 / dfSmoothed$nbObs * 100

library(cartography)
choroLayer(dfSmoothed
           , var = "prix95"
           , nclass = 5
           , method = "fisher-jenks"
           , border = NA
           , legend.title.txt = "prix du SP95 en centimes")

######### example 2 #########
library(sp)
library(cartography)

data(reunion)

# Smoothing all variables for Reunion (Lissage de toutes les variables pour la Reunion)

# Call mode 1: classic smoothing - automatic grid
reunionSmoothed <- kernelSmoothing( dfObservations = reunion
```

```
                                    , sEPSG = "32740"
                                    , iCellSize = 200L
                                    , iBandwidth = 400L)

# preview (Apercu)
choroLayer(reunionSmoothed, var = "houhold", nclass = 5, method = "fisher-jenks", border = NA)

# Call mode 2: median smoothing - automatic grid
reunionSmoothed <- kernelSmoothing( dfObservations = reunion
                                    , sEPSG = "32740"
                                    , iCellSize = 200L
                                    , iBandwidth = 400L
                                    , vQuantiles = c(0.1, 0.5, 0.9))
# preview (Apercu)
choroLayer(reunionSmoothed, var = "houhold_05", nclass = 5, method = "fisher-jenks", border = NA)


# Call mode 3: classic smoothing - user grid
dfCentroidsUser <- merge( x = seq(from =  314400L, to =  378800L, by = 200L)
                        , y = seq(from = 7634000L, to = 7691200L, by = 200L))
reunionSmoothed <- kernelSmoothing( dfObservations = reunion
                                    , sEPSG = "32740"
                                    , iCellSize = 200L
                                    , iBandwidth = 400L
                                    , dfCentroids = dfCentroidsUser)

# preview (Apercu)
reunionSmoothed <- reunionSmoothed[reunionSmoothed$houhold > 0, ]
choroLayer(reunionSmoothed, var = "houhold", nclass = 5, method = "fisher-jenks", border = NA)

# Call mode 4: median smoothing - user grid
reunionSmoothed <- kernelSmoothing( dfObservations = reunion
                                    , sEPSG = "32740"
                                    , iCellSize = 200L
                                    , iBandwidth = 400L
                                    , vQuantiles = c(0.1, 0.5, 0.9)
                                    , dfCentroids = dfCentroidsUser)

# preview (Apercu)
reunionSmoothed <- reunionSmoothed[reunionSmoothed$nbObs > 0, ]
choroLayer(reunionSmoothed, var = "houhold_05", nclass = 5, method = "fisher-jenks", border = NA)


## End(Not run)
```

`pixel_france` *Whole France grid with 1km square tiles (Grille France entiere avec des carreaux d'1km de cote))*

Description

Whole France grid with 1km square tiles. Lambert 93 projection

(Grille France entière avec des carreaux d'1km de côté. Projection Lambert 93)

Value

x	longitude
y	latitude

Examples

```
data(pixel_france_1km_2154)
```

---

reunion *Households of Reunion (Menages de La Reunion)*

Description

Gridded database with a grid cell resolution of 200 meters. Source : Insee, Localized data : Fiscal Revenue - 31/12/2010 and Housing Tax - 01/01/2011.

(Données carroyées à 200 mètres. Source : Insee, Revenus Fiscaux Localisés (RFL) au 31 decembre 2010 et Taxe d'habitation (TH) au 1er janvier 2011)

Value

x	longitude
y	latitude
houhold	number of households - nombre de ménages
phouhold	number of poor households - nombre de ménages pauvres

Examples

```
data(reunion)
# Smoothing of the variable houhold (households) and phouhold (poor households) for Reunion
# Lissage de la variable houhold(menages) et phouhold(menages pauvres) pour la Reunion
reunionSmooth <- kernelSmoothing(reunion, "32740", 200, 800)

# Calculating the poverty rate (Calcul du taux de pauvrete)
reunionSmooth$ratio <- reunionSmooth$phouhold / reunionSmooth$houhold * 100

#library(rgdal)
# Export of the basemap in shapefile format (Export du fond de carte au format shapefile)
#writeOGR(grid, "reunion.shp", "reunion", driver = "ESRI Shapefile")
```

Index

constituerGrappes, 2
constituerMatriceEffectifs, 3

dfPrix_SP95_2016, 4
dfRestaurantParis, 5
dfToGrid, 5

kernelSmoothing, 6

pixel_france, 10

reunion, 11