Package 'DasyMapR'

July 17, 2016

Type Package

Title Geographical ETRS-LAEA Grid Based Dasymetric Mapping

Version 0.0.0.9000

Author Evangelos Tsaklanos

Maintainer <etsakla@gmail.com>

Description The process of dasymetric mapping is the transformation of data from a set of arbitrary source zones to a dasymetric map via the overlay of the source zones with an ancillary data set.This package contains a number of fucntions that assist preparing vector and raster source and ancillary datasets for intelligent dasymetric mapping, performs the dasymetric calculations, and then generates a floating point output raster of revised density. TO BE NOTICED : It uses the ETRS89-LAEA geographic grid to do the job. Depends: R (>= 3.2.3)

License GPLv3

LazyData true

Depends rgdal,
rgeos,
methods,
sp,
doParallel,
foreach,
raster,
eurostat,
deldir

URL https://github.com/etsakl/DasyMapR.git

RoxygenNote 5.0.1

Collate 'Class-ETRS.R'
'Class-EtrsSurface.R'
'Class-EtrsGrid.R'
'Class-EtrsCells.R'
'Class-EtrsAncillarySurface.R'
'Class-EtrsSourceSurface.R'

'Class-DasymetricSurface.R'
'Class-EtrsPoint.R'
'CreateEtrsVectorGrid.R'
'DasyMapR-GenUt.R'
'DasyMapR-data-documentation.R'
'DasyMapR-package.R'

Suggests knitr,
rmarkdown,
plyr,
classInt,
RColorBrewer,
leaflet

VignetteBuilder knitr

R topics documented:

`DasyMapR-package` *DasyMapR*

Description

The process of dasymetric mapping is the transformation of data from a set of arbitrary source zones to a dasymetric map via the overlay of the source zones with an ancillary data set.This package contains a number of fucntions that assist preparing vector and raster source and ancillary datasets for intelligent dasymetric mapping, performs the dasymetric calculations, and then generates a floating point output raster of revised density. TO BE NOTICED : It uses the ETRS89-LAEA geographic grid to do the job.

`ActuallVal2Density` *Converts actuall value of a Surface to Value Density (in km^2 by default)*

Description

Converts actuall value of a Surface to Value Density (in km^2 by default)

Usage

```
ActuallVal2Density(input.surface = "SpatialPolygonsDataFrame",
  surface.value.col = "numeric", area.unit = 1e+06)
```

Arguments

`input.surface`	A SpatialPolygonsDataFrame object
`surface.value.col`	Index of VALUE column
`area.uint`	usually to km^2

Value

the original SpatialPolygonsDataFrame object but VALUE is now Density

Examples

```
# test  ActuallVal2Density ------------------------------------------------------
# Make a spatial polygon
S<-readWKT("POLYGON((0 0,2000 0,2000 2000,0 2000,0 0))",id = "S",p4s = CRS("+init=epsg:3035"))
df<-cbind(AREA=gArea(S),VALUE=1000)
row.names(df)<-sapply(slot(S,"polygons"),function(x) slot(x,"ID"))
df<-as.data.frame(df)
S<-SpatialPolygonsDataFrame(S,data = df,match.ID = TRUE)
# Covert actuall value to density
S_D<-ActuallVal2Density(input.surface = S,surface.value.col = 2,area.unit = 1e+06)
# plot
X11(width=12,height = 12)
split.screen(figs = c(2,1))
screen(1)
plot(S)
title("the input surface")
text(coordinates(S)[,1],coordinates(S)[,2],paste("Actuall Value =",S[["VALUE"]]))
screen(2)
plot(S_D)
title("the input surface changed the attr VALUE")
text(coordinates(S_D)[,1],coordinates(S_D)[,2],paste("Density =", S_D[["VALUE"]]))
```

candidates_addresses *Candidates addresses of HOU*

Description

Non personal data of HOU candidate students

Usage

```
candidates_addresses
```

Format

An object of class `data.frame` with 794875 rows and 6 columns.

Note

It is subset of dataset and any use for purposes other than educational is prohbited by the owner (HOU)

Author(s)

HOU

CheckEtrsResolution,numeric-method *Checks if the given resolution is INSPIRED compatible*

Description

The grid is defined as a hierarchical one in metric coordinates to the power of 10.The resolution of the grid is 1m, 10m, 100m, 1000m, 10,000m, 100,000m. Althought EEA recommend grid size of metric resolution in standard size 100 m, 1 km, 10 km and 100 km. Alternatively use 25 m or 250 m for analysis purposes, where standard 100 m or 1 km grid size is not appropriate.

Usage

```
## S4 method for signature 'numeric'
CheckEtrsResolution(cell.size)
```

Arguments

`cell.size`	numeric one of the resolutions of the grid that are 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km

Value

TRUE if everything is ok if it is not returns 100km cell-size based grid

Examples

```
CheckEtrsResolution(cell.size=1000)
CheckEtrsResolution(cell.size=125)
```

`CheckEtrsValidity,Spatial-method`

Checks the ETRS validity of a Spatial Object

Description

Checks the ETRS validity of a Spatial Object `CheckEtrsValidity()` actually checks if the given object is Spatial and also if it is projected in ETRS-LAEA

Usage

```
## S4 method for signature 'Spatial'
CheckEtrsValidity(object)
```

Arguments

`ETRS`	: Accepts as an argument a ETRS object

Value

TRUE if everything is ok

Examples

```
xc <- round(runif(10, min = 4321000, max = 4323000), 2)
yc <- round(runif(10, min = 3210000, max = 3212000), 2)
xy <- cbind(xc, yc)
# a SpatialPoints object
xy.sp <- SpatialPoints(xy, proj4string = CRS("+init=epsg:2100"))
# CheckEtrsValidity(xy.sp)
xy.sp@proj4string<-CRS("+init=epsg:3035")
CheckEtrsValidity(xy.sp)
```

CLC2000.ARGOLIDA.RES *Corine Land Cover 2000 seamless vector data*

Description

Version 17 (12/2013) - Corine land cover 2000 is the year 2000 update of the first CLC database which was finalised in the early 1990s as part of the European Commission programme to COoRdinate INformation on the Environment (Corine)

Usage

```
CLC2000.ARGOLIDA.RES
```

Format

Spatial Polygons Data Frames

Note

EEA standard re-use policy: unless otherwise indicated, re-use of content on the EEA website for commercial or non-commercial purposes is permitted free of charge, provided that the source is acknowledged (http://www.eea.europa.eu/legal/copyright). Copyright holder: European Environment Agency (EEA).

Author(s)

European Environment Agency (EEA)

Source

http://www.eea.europa.eu/data-and-maps/data/corine-land-cover-2000-clc2000-seamless-vector-databas
tab-interactive-maps-produced

CreateEtrsVectorGrid *CreateEtrsVectorGrid*

Description

Implements a geographical grid system as vector data (polygons) based on ETRS89-LAEA

Usage

```
CreateEtrsVectorGrid(x, cell.size, save.flag = FALSE)
```

Arguments

x	a Spatial object or an object with a bbox(object) matrix my Deafult method
cell.size	a numeric one of the resolutions of the grid that is 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km.
save.flag	if you want to save the results to a file in the disk turn it to TRUE

Value

An object of SpatialPolygonsDataFrame containing an attribute coding system follows the recommendations from the European Environmental Agency [EEA 2008]

Examples

```
xc <- round(runif(10, min = 4321000, max = 4323000), 2)
yc <- round(runif(10, min = 3210000, max = 3212000), 2)
xy <- cbind(xc, yc)
xy.sp <- SpatialPoints(xy, proj4string = CRS("+init=epsg:3035"))
df<-as.data.frame(rep(1,length(xy.sp)))
colnames(df)<-"VALUE"
xy.sp<-SpatialPointsDataFrame(xy.sp,df)
cell.size <- 1000
aGrid <- CreateEtrsVectorGrid(x = xy.sp, cell.size = cell.size)
# ... aggregate some data into it
agg<-aggregate(xy.sp,aGrid,sum)
X11(width = 10,10)
split.screen(figs=c(1,2))
screen(1)
plot(aGrid)
plot(xy.sp,add=TRUE,pch=21,bg=rgb(1,0,0,.1),cex=xy.sp[["VALUE"]])
screen(2)
plot(aGrid)
plot(gCentroid(agg,byid=TRUE),add=TRUE,pch=21,bg=rgb(1,0,0,.1),cex=agg[["VALUE"]])
```

---

dasymapPlot,EtrsSurface,numeric-method

a utility for plotting EtrsSurface combining attributes

Description

produce a 5 class choroplethe map for en ETRS surface

(lealet version)

Usage

```
## S4 method for signature 'EtrsSurface,numeric'
dasymapPlot(aEtrsSurface, col.value)

## S4 method for signature 'EtrsSurface,numeric'
dasymapPlot(aEtrsSurface, col.value)
```

Arguments

aEtrsSurface	EtrsSurface.
col.value	numeric.
sppdf	EtrsSurface
column	of the class value

Value

a plot

a leaflet plot

ETRS-class *Virtual Parent Class for ETRS objects*

Description

The ETRS class contains (extends) the Spatial Class. As a result of that has the slots of the parent class : @bbox and proj4string.

Value

One cannot create an ETRS instance inasmuch as ETRS is VIRTUAL class

Slots

`etrs.cell.codes.columns` character: a character vector c("CELLCODE","EASTOFORIGIN","NORTHORIGIN") The Enviromental European Agency suggests to be provided, except "CELLCODE", for computations and analysis the lower left corner coordinate in meters, formatted as two separate integers attribute values named "EOFORIGIN" and "NOFORIGIN"

`VIRTUAL.` Actually only to declare the ETRS class abstract

`etrsAncillarySurface,SpatialPolygonsDataFrame,character,numeric,numeric,logical-method`
etrsAncillarySurface

Description

the default method that creates an EtrsAncillary Surface Object

Usage

```
  ## S4 method for signature
## 'SpatialPolygonsDataFrame,character,numeric,numeric,logical'
etrsAncillarySurface(input.surface,
  over.method.type, surface.value.col, cell.size, binary)
```

Arguments

`input.surface`	A SpatialPolygonsDataFrame object that represents a ancillary dataset to be used to redistribute VALUE. The output from this tool can be used in as an input to `EtrsDasymetricSurface()` method Land-use or land-cover are the most frequently used ancillary datasets, but any dataset that has classes of relatively homogenous density relativy with the VALUE could be used here.
`over.method.type`	PropCal Proportional calculation: the cell takes a calculated value depending on the values of the units falling inside and their share within the cell. This method seems very appropriate for countable variables
`surface.value.col`	Index number of colum that keeps the relative density of a cell with land-cover some type
`cell.size:`	one of the resolutions of the grid that is 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km

Value

an EtrsAncillarySurface Object

Examples

```
# test EtrsAncillarySurface ------------------------------------------------------
x <- round(runif(1, min = 4320200, max = 4322000), 1)
y <- round(runif(1, min = 3210000, max = 3212000), 1)
xy <- cbind(x, y)
size = round(runif(1,min = 500,max = 1000),0)
pl1 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)
```

```
  ))),round(runif(1,min = 1,max = 100),0))
x <- x + size

pl2 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)
  ))),round(runif(1,min = 1,max = 100),0))
ancS <-SpatialPolygons(list(pl1,pl2),proj4string = CRS("+init=epsg:3035"))

df <-data.frame(RelDens=c(0.15,0.85),bin=c(1,1),row.names = sapply(slot(ancS, "polygons"), function(x)slot(x, "
ancS<-SpatialPolygonsDataFrame(ancS,data = df)
ancS.grided<-etrsAncillarySurface(input.surface = ancS,over.method.type = "PropCal",surface.value.col = 1,cell.s
plot(ancS.grided,border=4,lwd=2)
plot(ancS,add=TRUE,col=rgb(0,.25,.3,.1))
text(coordinates(ancS)[,1],coordinates(ancS)[,2],paste("reDens=",ancS[[1]]))
text(coordinates(ancS.grided)[,1],coordinates(ancS.grided)[,2],paste("CELLVALUE=",ancS.grided[[4]]))
```

`EtrsAncillarySurface-class`
The EtrsAncillarySurface class

Description

The EtrsAncillarySurface class holds the basic information for ancillary surface represanation in ETRS-LAEA grid form. Land-use or land-cover are the most frequently used ancillary datasets, but any dataset that has classes of relatively density could be used here.

Value

An EtrsAncillarySurface Object

Slots

`SpatialPolygonsDataFrame.` the input surface plus new data column WCELLWEIGHT

`over.method.type` character. PropCal only. An ancillary Surface must have a CELLVALUE

`cell.size` numeric. Indexing the column of data frame that contains the value of #'interest

`CELLVALUE` character. The size of the cell (the new map unit)

Examples

```
# test EtrsAncillarySurface ------------------------------------------------------
x <- round(runif(1, min = 4320200, max = 4322000), 1)
y <- round(runif(1, min = 3210000, max = 3212000), 1)
xy <- cbind(x, y)
size = round(runif(1,min = 500,max = 1000),0)
pl1 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)
```

```
  ))),round(runif(1,min = 1,max = 100),0))
x <- x + size

pl2 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)
  ))),round(runif(1,min = 1,max = 100),0))
ancS <-SpatialPolygons(list(pl1,pl2),proj4string = CRS("+init=epsg:3035"))

df <-data.frame(RelDens=c(0.15,0.85),bin=c(1,1),row.names = sapply(slot(ancS, "polygons"), function(x)slot(x, "
ancS<-SpatialPolygonsDataFrame(ancS,data = df)
ancS.grided<-etrsAncillarySurface(input.surface = ancS,over.method.type = "PropCal",surface.value.col = 1,cell.s
plot(ancS.grided,border=4,lwd=2)
plot(ancS,add=TRUE,col=rgb(0,.25,.3,.1))
text(coordinates(ancS)[,1],coordinates(ancS)[,2],paste("reDens=",ancS[[1]]))
text(coordinates(ancS.grided)[,1],coordinates(ancS.grided)[,2],paste("CELLVALUE=",ancS.grided[[4]]))
```

`etrsAncillarySurface.default` *etrsAncillarySurface*

Description

the default method that creates an EtrsAncillary Surface Object

Usage

```
etrsAncillarySurface.default(input.surface, over.method.type, surface.value.col,
  cell.size, binary)
```

Arguments

`input.surface`	A SpatialPolygonsDataFrame object that represents a ancillary dataset to be used to redistribute VALUE. The output from this tool can be used in as an input to `EtrsDasymetricSurface()` method Land-use or land-cover are the most frequently used ancillary datasets, but any dataset that has classes of relatively homogenous density relativy with the VALUE could be used here.
`over.method.type`	"PropCal" in proportional calculation the cell takes a calculated value depending on the values of the units falling inside and their share within the cell. This method seems very appropriate for countable variables
`surface.value.col`	the number of colum that keeps the relative density of a cell with land-cover some type
`cell.size:`	one of the resolutions of the grid that is 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km

Value

an EtrsAncillarySurface Object

Examples

```
#' # test EtrsAncillarySurface ----------------------------------------------------
x <- round(runif(1, min = 4320200, max = 4322000), 1)
y <- round(runif(1, min = 3210000, max = 3212000), 1)
xy <- cbind(x, y)
size = round(runif(1,min = 500,max = 1000),0)
pl1 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)
  ))),round(runif(1,min = 1,max = 100),0))
x <- x + size

pl2 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)
  ))),round(runif(1,min = 1,max = 100),0))
ancS <-SpatialPolygons(list(pl1,pl2),proj4string = CRS("+init=epsg:3035"))

df <-data.frame(RelDens=c(0.15,0.85),bin=c(1,1),row.names = sapply(slot(ancS, "polygons"), function(x)slot(x, "
ancS<-SpatialPolygonsDataFrame(ancS,data = df)
ancS.grided<-etrsAncillarySurface(input.surface = ancS,over.method.type = "PropCal",surface.value.col = 1,cell.s
plot(ancS.grided,border=4,lwd=2)
plot(ancS,add=TRUE,col=rgb(0,.25,.3,.1))
text(coordinates(ancS)[,1],coordinates(ancS)[,2],paste("reDens=",ancS[[1]]))
text(coordinates(ancS.grided)[,1],coordinates(ancS.grided)[,2],paste("CELLVALUE=",ancS.grided[[4]]))
```

`etrsCellCenter` *The polygon in which the center of the cell yields the attribute to assign to the cell.*

Description

The polygon in which the center of the cell yields the attribute to assign to the cell.

Usage

```
etrsCellCenter(the.etrs.grid = "EtrsGrid")
```

Arguments

`the.etrs.grid`	an EtrsGrid

Value

Returns centroids of an etrsgrid

EtrsCellCodes,matrix,numeric,missing-method
A cell code identifier for ETRS

Description

Define the cell size prefix

Define the cell size prefix

Usage

```
## S4 method for signature 'matrix,numeric,missing'
EtrsCellCodes(eastings.northings, cell.size)

## S4 method for signature 'matrix,numeric,logical'
EtrsCellCodes(eastings.northings, cell.size,
  NE)
```

Arguments

eastings.northings	matrix.
cell.size	numeric.
eastings.northings	matrix.
cell.size	numeric.

Value

etrs cell code vector

etrs cell code vector

Examples

```
bb <- bbox(cbind(c(4321000, 4323000), c(3210000,3212000)))
cell.size = 1000
# Calculate the limits of the grid
bb.outer.limits <-  cbind((floor(bb[, 1] / cell.size)) * cell.size, (ceiling(bb[, 2] / cell.size)) * cell.size)
# Calculates eastings
eastings <-seq(from = bb.outer.limits[1, 1], to = bb.outer.limits[1, 2], by = cell.size)
# Calculates northings
northings <-seq(from = bb.outer.limits[2, 1], to = bb.outer.limits[2, 2], by = cell.size)
eastings.northings <- as.matrix(expand.grid(eastings, northings))
low.corner.cell.etrs.points<-etrsPoints(eastings.northings = eastings.northings,cell.size =1000)
EtrsCellCodes<-EtrsCellCodes(eastings.northings,cell.size=1000)

bb <- bbox(cbind(c(4321000, 4323000), c(3210000,3212000)))
```

```
cell.size = 1000
# Calculate the limits of the grid
bb.outer.limits <-  cbind((floor(bb[, 1] / cell.size)) * cell.size, (ceiling(bb[, 2] / cell.size)) * cell.size)
# Calculates eastings
eastings <-seq(from = bb.outer.limits[1, 1], to = bb.outer.limits[1, 2], by = cell.size)
# Calculates northings
northings <-seq(from = bb.outer.limits[2, 1], to = bb.outer.limits[2, 2], by = cell.size)
eastings.northings <- as.matrix(expand.grid(eastings, northings))
low.corner.cell.etrs.points<-etrsPoints(eastings.northings = eastings.northings,cell.size =1000)
EtrsCellCodes<-EtrsCellCodes(eastings.northings,cell.size=1000,NE=TRUE)
```

---

etrsCells,data.frame,numeric-method

The Contsructor of the EtrsCells object

Description

The Contsructor of the EtrsCells object

Usage

```
## S4 method for signature 'data.frame,numeric'
etrsCells(etrs.table.codes = "data.frame",
  cell.size = "numeric")
```

Arguments

etrs.table.codes	a data frame CELLCODE,EASTOFORIGIN,NORTHOFORIGIN
cell.size	a numeric 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km

Value

an EtrsCells object

Examples

```
bb <- bbox(cbind(c(4321000, 4323000), c(3210000,3212000)))
cell.size = 1000
# Calculate the limits of the grid
bb.outer.limits <-
 cbind((floor(bb[, 1] / cell.size)) * cell.size, (ceiling(bb[, 2] / cell.size)) * cell.size)
# Calculates eastings
eastings <-
 seq(from = bb.outer.limits[1, 1], to = bb.outer.limits[1, 2], by = cell.size)
# Calculates northings
northings <-
 seq(from = bb.outer.limits[2, 1], to = bb.outer.limits[2, 2], by = cell.size)
```

```
eastings.northings <- as.matrix(expand.grid(eastings, northings))
low.corner.cell.etrs.points<-etrsPoints(eastings.northings = eastings.northings,cell.size =1000)
df <-EtrsTableCodes(eastings.northings,1000)
sp.df <- etrsCells(df,1000)
summary(sp.df)
```

EtrsCells-class *The etrsCells class*

Description

Produce a SpatialPolygons object of EtrsGridCells given a data.frame of CELLCODE NORTHINGS EASTINGS

Slots

SpatialPolygons Object

Examples

```
bb <- bbox(cbind(c(4321000, 4323000), c(3210000,3212000)))
cell.size = 1000
# Calculate the limits of the grid
bb.outer.limits <-
 cbind((floor(bb[, 1] / cell.size)) * cell.size, (ceiling(bb[, 2] / cell.size)) * cell.size)
# Calculates eastings
eastings <-
 seq(from = bb.outer.limits[1, 1], to = bb.outer.limits[1, 2], by = cell.size)
# Calculates northings
northings <-
 seq(from = bb.outer.limits[2, 1], to = bb.outer.limits[2, 2], by = cell.size)
eastings.northings <- as.matrix(expand.grid(eastings, northings))
low.corner.cell.etrs.points<-etrsPoints(eastings.northings = eastings.northings,cell.size =1000)
df <-EtrsTableCodes(eastings.northings,1000)
sp.cells <- etrsCells(df,1000)
summary(sp.cells)
```

---

etrsCells.default *The Contsructor of the EtrsCells object*

Description

The Contsructor of the EtrsCells object

Usage

```
etrsCells.default(etrs.table.codes = "data.frame", cell.size = "numeric")
```

Arguments

etrs.table.codes	a data frame CELLCODE,EASTOFORIGIN,NORTHOFORIGIN
cell.size	numeric : one of the resolutions of the grid that is 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km

Value

an EtrsCells object

Examples

```
bb <- bbox(cbind(c(4321000, 4323000), c(3210000,3212000)))
cell.size = 1000
# Calculate the limits of the grid
bb.outer.limits <-
 cbind((floor(bb[, 1] / cell.size)) * cell.size, (ceiling(bb[, 2] / cell.size)) * cell.size)
# Calculates eastings
eastings <-
 seq(from = bb.outer.limits[1, 1], to = bb.outer.limits[1, 2], by = cell.size)
# Calculates northings
northings <-
 seq(from = bb.outer.limits[2, 1], to = bb.outer.limits[2, 2], by = cell.size)
eastings.northings <- as.matrix(expand.grid(eastings, northings))
low.corner.cell.etrs.points<-etrsPoints(eastings.northings = eastings.northings,cell.size =1000)
df <-EtrsTableCodes(eastings.northings,1000)
sp.cells <- etrsCells(df,1000)
summary(sp.cells)
```

EtrsCheckCodeColumns,ETRS-method

checks if colum names data are etrs table.codes

Description

Is the slot etrs.table.cell.code identical to data colum names?

Usage

```
## S4 method for signature 'ETRS'
EtrsCheckCodeColumns(object)
```

Arguments

Etrs	an etrs object

etrsDasymetric2Ancillary,EtrsDasymetricSurface-method

A method to convert a DasyMetric surface to Ancillary

Description

A method to convert a DasyMetric surface to Ancillary

Usage

```
## S4 method for signature 'EtrsDasymetricSurface'
etrsDasymetric2Ancillary(dasymetric.surface)
```

Arguments

`dasymetric.surface`
: EtrsDasymetricSurface.

Value

EtrsAncillarySurface

etrsDasymetric2Raster,EtrsDasymetricSurface-method

A simple method to produce a floating point raster

Usage

```
## S4 method for signature 'EtrsDasymetricSurface'
etrsDasymetric2Raster(dasymetric.surface)
```

Arguments

`dasymetric.surface`
: EtrsDasymetricSurface.

`dasymetric.surface`
: EtrsDasymetricSurface.

Value

a Raster object and a geo tif file in working directory

a Raster object and a geo tif file in working directory

Examples

```
data(DASY_GDP.rda)
par(mar = c(0.1, 0.1, 0.1, 0.1))
DASY_GPD_RASTER <- etrsDasymetric2Raster(dasymetric.surface = DASY_GPD)
rw.colors <- grey.colors
image(DASY_GPD_RASTER, col = rw.colors(5))
A simple method to produce a rfloating point aster floating point
```

---

etrsDasymetric2Source,EtrsDasymericSurface-method *simple utillity converts EtrsDasymetric 2 EtrsSourceSurface*

Description

simple utillity converts EtrsDasymetric 2 EtrsSourceSurface

Usage

```
## S4 method for signature 'EtrsDasymericSurface'
etrsDasymetric2Source(d)
```

Arguments

d	EtrsDasymericSurface.

Value

EtrsSourceSurface

EtrsDasymetricSurface *The DasymetricSurface class*

Description

The DasyemetricSurface class holds the basic information for dasymetric surface represanation in ETRS-LAEA grid form

P_mu = (R_A x P_A) x N / E P_cell = (R_A x P_A /P_A) x (N / A_T) / E = (R_A x N / A_T) / E Where, P_cell is the population of a cell, R_A is the relative density of a cell with land-cover type A, P_A is the proportion of cells of land-cover type A in the enumeration unit. N is the actual population of enumeration unit (i.e., census block B group) E is the expected population of enumeration unit calculated using the relative densities. A_T is the total number of cells in the enumeration unit. P_A / P_A cancels P_A out of the equation, i.e., not used in the cell- based method.

Usage

```
EtrsDasymetricSurface(input.surface.grided, ancillary.grided, ...)

EtrsDasymetricSurface(input.surface.grided, ancillary.grided, ...)
```

Arguments

`ancillary.grided`

Value

An DasymetricSurface Object

EtrsDasymetricSurface

Slots

`SpatialPolygonsDataFrame.` the input surface plus new data columns

`over.method.type` character. MaxArea for categorical data PropCal for numeric #'values

`cell.size` numeric. Indexing the column of data frame that contains the value of #'interest

`CELLVALUE` character. The size of the cell (the new map unit)

Examples

```
# test Dasymetric  ------------------------------------------------------
# Make a spatial polygon
S<-readWKT("POLYGON((0 0,2000 0,2000 2000,0 2000,0 0))",id = "S",p4s = CRS("+init=epsg:3035"))
df<-cbind(AREA=gArea(S),VALUE=1000)
row.names(df)<-sapply(slot(S,"polygons"),function(x) slot(x,"ID"))
df<-as.data.frame(df)
S<-SpatialPolygonsDataFrame(S,data = df,match.ID = TRUE)
# Covert actuall value to density
S<-ActuallVal2Density(input.surface = S,surface.value.col = 2,area.unit = 1e+06)
# plot
X11(width=12,height = 12)
split.screen(figs = c(3,2))
screen(1)
plot(S)
title("the input surface")
text(coordinates(S)[,1],coordinates(S)[,2],S[["VALUE"]])
text(coordinates(S)[,1],coordinates(S)[,2],S[["VALUE"]])
#grided
S.grided <- etrsSourceSurface(input.surface = S,over.method.type = "PropCal",surface.value.col = 2,cell.size = 5
screen(2)
plot(S.grided)
title("... Projected to ETRS-LAEA 500 m")
text(coordinates(S.grided)[,1],coordinates(S.grided)[,2],S.grided[["CELLVALUE"]])

#the ancillary surface
A<-readWKT("POLYGON((0 0,1000 0,1000 500,0 500,0 0))",id="A" ,p4s = CRS("+init=epsg:3035" ))
```

```
df<-cbind(AREA=gArea(A),RelDens=1)
row.names(df)<-sapply(slot(A,"polygons"),function(x) slot(x,"ID"))
df<-as.data.frame(df)
A<-SpatialPolygonsDataFrame(A,data = df,match.ID = TRUE)
screen(3)
plot(S,border="lightgrey")
plot(A,add=T)
title("the ancillary surface")
text(coordinates(A)[,1],coordinates(A)[,2],A[["RelDens"]])
#grided
A.grided <- etrsAncillarySurface(input.surface = A,over.method.type = "PropCal",surface.value.col = 2,cell.size
screen(4)
plot(S.grided,border="lightgrey")
plot(A.grided,add=T)
title("... Projected to ETRS-LAEA 500 m")
text(coordinates(A.grided)[,1],coordinates(A.grided)[,2],A.grided[["WCELLWEIGHT"]])
#EtrsDasymetric
screen(6)
D.grided<-EtrsDasymetricSurface(input.surface.grided = S.grided,ancillary.grided = A.grided,actuall.value = TRUE
plot(S.grided, border = "lightgrey")
plot(D.grided,add=T)
title("Finally the Dasymetric Surface")
text(coordinates(D.grided)[,1],coordinates(D.grided)[,2],D.grided[["DASYCELL"]])
```

EtrsDasymetricSurface,EtrsSourceSurface,EtrsAncillarySurface-method
EtrsDasymetricSurface method

Description

EtrsDasymetricSurface method

Usage

```
  ## S4 method for signature 'EtrsSourceSurface,EtrsAncillarySurface'
EtrsDasymetricSurface(input.surface.grided,
  ancillary.grided, ...)
```

Arguments

input.surface.grided	EtrsSourceSurface.
ancillary.grided	EtrsAncillarySurface.
actuall.value	logical.

`etrsGrid,SpatialPolygonsDataFrame,numeric-method`
the etrsGrid default method

Description

the etrsGrid default method

Usage

```
  ## S4 method for signature 'SpatialPolygonsDataFrame,numeric'
etrsGrid(obj = "SpatialPolygonsDataFrame",
  cell.size = "numeric")
```

Arguments

`obj`	SpatialPolygonsDataFrame.
`cell.size`	numeric. @param cell.size a numeric one of the resolutions of the grid that is 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km.

Value

EtrsGrid class object

Examples

```
#' x <- round(runif(1, min = 4320200, max = 4322000), 1)
y <- round(runif(1, min = 3210000, max = 3212000), 1)
xy <- cbind(x, y)
size = round(runif(1,min = 1500,max = 2000),0)
pl1 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)
  ))),round(runif(1,min = 1,max = 100),0))
x <- x + size
pl2 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + 2 * size,y + 2 * size,y)
  ))),round(runif(1,min = 1,max = 100),0))
x<-x-size
y<-y+size
pl3 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y +  size,y +  size,y)
  ))),round(runif(1,min = 1,max = 100),0))
sps <-
  SpatialPolygons(list(pl1,pl2,pl3),proj4string = CRS("+init=epsg:3035"))
```

```
df <-
  data.frame(val=c("R5","R40","R80"),row.names = sapply(slot(sps, "polygons"), function(x)
    slot(x, "ID")))
sps<-SpatialPolygonsDataFrame(sps,data = df)
the.etrs.grid<-etrsGrid(sps,1000)
slotNames(the.etrs.grid)
the.etrs.grid@the.grid.name
head(the.etrs.grid@data,3)
plot(the.etrs.grid)
title(paste("the etrs grid",the.etrs.grid@the.grid.name))
```

`EtrsGrid-class` *Create the EtrsGrid Class*

Description

Represents A Geographical grid on ERTS89-LAEA Set the name of the class

Arguments

`cell.size`	a numeric one of the resolutions of the grid that is 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km.

Value

An object of the class EtrsGrid

Slots

`SpatialPolygonsDataFrame.` An object that the Grid net should cover completely grid

`the.grid.name` character. The grid is designated as Grid_ETRS89-LAEA. For identification of an individual resolution level the cell size in metres is appended to the name. EXAMPLE LAEA_100K. The grid at a resolution level of 100km is designated as Grid_ETRS89-

Examples

```
x <- round(runif(1, min = 4320200, max = 4322000), 1)
y <- round(runif(1, min = 3210000, max = 3212000), 1)
xy <- cbind(x, y)
size = round(runif(1,min = 1500,max = 2000),0)
pl1 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)
  ))),round(runif(1,min = 1,max = 100),0))
x <- x + size
pl2 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + 2 * size,y + 2 * size,y)
```

```
  ))),round(runif(1,min = 1,max = 100),0))
x<-x-size
y<-y+size
pl3 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y +  size,y +  size,y)
  ))),round(runif(1,min = 1,max = 100),0))
sps <-
  SpatialPolygons(list(pl1,pl2,pl3),proj4string = CRS("+init=epsg:3035"))
df <-
 data.frame(val=c("R5","R40","R80"),row.names = sapply(slot(sps, "polygons"), function(x)
    slot(x, "ID")))
sps<-SpatialPolygonsDataFrame(sps,data = df)
the.etrs.grid<-etrsGrid(sps,1000)
slotNames(the.etrs.grid)
the.etrs.grid@the.grid.name
head(the.etrs.grid@data,3)
plot(the.etrs.grid)
title(paste("the etrs grid",the.etrs.grid@the.grid.name))
```

`etrsGrid.default` *the etrsGrid default method*

Description

the etrsGrid default method

Usage

```
etrsGrid.default(obj = "SpatialPolygonsDataFrame", cell.size = "numeric")
```

Arguments

`obj`	SpatialPolygonsDataFrame. An object that the Grid net should cover completely grid
`cell.size`	ell.size a numeric one of the resolutions of the grid that is 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km.

Value

An object of the class EtrsGrid

Examples

```
x <- round(runif(1, min = 4320200, max = 4322000), 1)
y <- round(runif(1, min = 3210000, max = 3212000), 1)
xy <- cbind(x, y)
size = round(runif(1,min = 1500,max = 2000),0)
pl1 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)
  ))),round(runif(1,min = 1,max = 100),0))
x <- x + size
pl2 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + 2 * size,y + 2 * size,y)
  ))),round(runif(1,min = 1,max = 100),0))
x<-x-size
y<-y+size
pl3 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y +  size,y +  size,y)
  ))),round(runif(1,min = 1,max = 100),0))
sps <-
  SpatialPolygons(list(pl1,pl2,pl3),proj4string = CRS("+init=epsg:3035"))
df <-
  data.frame(val=c("R5","R40","R80"),row.names = sapply(slot(sps, "polygons"), function(x)
    slot(x, "ID")))
sps<-SpatialPolygonsDataFrame(sps,data = df)
the.etrs.grid<-etrsGrid(sps,1000)
slotNames(the.etrs.grid)
the.etrs.grid@the.grid.name
head(the.etrs.grid@data,3)
plot(the.etrs.grid)
title(paste("the etrs grid",the.etrs.grid@the.grid.name))
```

`etrsGrid2Spdf` *Converts an etrsgrid obj to standard SpatiaPolygonsDataFrame*

Description

It is usefull for rGeos that doesn't support coercion. ALso to be used for coercion

Usage

```
etrsGrid2Spdf(the.etrs.grid = "etrsGrid")
```

Arguments

`the.etrs.grid`

Value

SpatialPolygonsDataFrame object

`etrsMaxArea` *Computes a single figure by each reference grid cell using Maximum Area as integration methods*

Description

The Maximum area criteria: the cell takes the value of the unit which covers most of the cell area. It should be a good option for uncountable variables

Usage

```
etrsMaxArea(the.etrs.grid = "EtrsGrid", the.surface = "EtrsSurface")
```

Arguments

`the.etrs.grid`	An object of the class EtrsGrid
`the.surface`	an object of class EtrsSurfaface

Value

An EtrsSurface

Examples

```
x <- round(runif(1, min = 4321000, max = 4322000), 1)
y <- round(runif(1, min = 3211000, max = 3212000), 1)
xy <- cbind(x, y)
size = round(runif(1,min = 1500,max = 1500),0)
pl1 <-Polygons(list(Polygon(cbind(c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)  ))),round(runif(1,min =
x <- x + size
pl2 <- Polygons(list(Polygon(cbind(c(x,x + size,x + size,x,x),c(y,y,y + 2 * size,y + 2 * size,y)))),round(runif(1
x<-x-size
y<-y+size
pl3 <-Polygons(list(Polygon(cbind(c(x,x + size,x + size,x,x),c(y,y,y +  size,y +  size,y)  ))),round(runif(1,min
sps <- SpatialPolygons(list(pl1,pl2,pl3),proj4string = CRS("+init=epsg:2100"))
df <-data.frame(val=c("R5","R40","R80"),row.names = sapply(slot(sps, "polygons"), function(x) slot(x, "ID")))
Source.Surface <-SpatialPolygonsDataFrame(sps,data = df)

# Uses the default etrsSurface method
Source.Surface.MaxArea <- etrsSurface(input.surface = Source.Surface, over.method.type = "MaxArea", cell.size =
Source.Surface.MaxArea <-
etrsSurface(input.surface = Source.Surface, over.method.type = "MaxArea", cell.size = 1000)
Source.Surface<-EtrsTransform(Source.Surface)
plot(Source.Surface)
plot(Source.Surface.MaxArea,lty = 3,lwd = 1.2,border = 3,add=TRUE)
x.y.s.s <- coordinates(EtrsTransform(Source.Surface))
```

```
x.y.max <- coordinates(Source.Surface.MaxArea)
text(x.y.s.s[,1],x.y.s.s[,2],Source.Surface@data$val,col = 4,cex = 1.5)
text(x.y.s.s[,1],x.y.s.s[,2] - 100,paste("Feature=",rownames(Source.Surface@data)),col = 4,cex = 1.2)
text(x.y.max[,1],x.y.max[,2],Source.Surface.MaxArea@data$FEATURE,col=3)
title("The 3 regions in Etrs Grid using Max Area Intergration")
```

`etrsPoint2Grid` *Produce a etrs aggregating point data*

Description

Produce a etrs aggregating point data

Usage

```
etrsPoint2Grid(obj = "SpatialPointsDataFrame", point.value.col = "numeric",
  cell.size = "numeric", mean.flag = FALSE)
```

Arguments

`obj`	a SpatialPointsDataFrame object
`point.value.col`	The value that is going to be aggregated
`cell.size`	the resolution of the grid
`mean.flag`	False for `sum()` the value TRUE `mean()` the value

Value

SpatialPointsDataFrame

Examples

```
xc <- round(runif(10, min = 4321000, max = 4323000), 2)
yc <- round(runif(10, min = 3210000, max = 3212000), 2)
xy <- cbind(xc, yc)
xy.sp <- SpatialPoints(xy, proj4string = CRS("+init=epsg:3035"))
df<-as.data.frame(rep(1,length(xy.sp)))
colnames(df)<-"VALUE"
xy.sp<-SpatialPointsDataFrame(xy.sp,df)
aGrid <- etrsPoint2Grid(obj = xy.sp, point.value.col=1,cell.size = 1000,mean.flag =FALSE)
X11(width = 10,10)
split.screen(figs=c(1,2))
screen(1)
plot(aGrid)
title("Some Points ...")
plot(xy.sp,add=TRUE,pch=21,bg=rgb(1,0,0,.1),cex=xy.sp[["VALUE"]])
```

```
screen(2)
 plot(aGrid)
plot(gCentroid(etrsGrid2Spdf(aGrid),byid=TRUE),add=TRUE,pch=21,bg=rgb(1,0,0,.1),cex=aGrid[["VALUE"]])
title(paste("aggregated in the",aGrid@the.grid.name))
```

`etrsPoints,matrix,numeric-method`
`"etrsPoints"` *is the* `"EtrsPoints"` *object constructor*

Description

`"etrsPoints"` is the `"EtrsPoints"` object constructor

Usage

```
## S4 method for signature 'matrix,numeric'
etrsPoints(eastings.northings = "matrix",
  cell.size = "numeric")
```

Arguments

`eastings.northings`	: a matrix of coordinates
`cell.size`	: a numeric object represnts the cell size of the grid

Value

An EtrsPoints object

Examples

```
# Test for EtrsPoint
bb <- bbox(cbind(c(4321000, 4323000), c(3210000,3212000)))
cell.size = 1000
# Calculate the limits of the grid
bb.outer.limits <- cbind((floor(bb[, 1]/cell.size))*cell.size,(ceiling(bb[,2]/cell.size))*cell.size)
eastings <- seq(from = bb.outer.limits[1, 1], to = bb.outer.limits[1, 2], by = cell.size)
#Calculates northings
northings <- seq(from = bb.outer.limits[2, 1], to = bb.outer.limits[2, 2], by = cell.size)
eastings.northings <- as.matrix(expand.grid(eastings, northings))
low.corner.cell.etrs.points<-etrsPoints(eastings.northings = eastings.northings,cell.size=1000)
plot(low.corner.cell.etrs.points)
text(eastings.northings[,1]+200,eastings.northings[,2],low.corner.cell.etrs.points[["NORTHOFORIGIN"]])
text(eastings.northings[,1],eastings.northings[,2]+200,low.corner.cell.etrs.points[["EASTOFORIGIN"]],srt=90)
title("Lower Cell Corner Etrs Points using ETRS-LAEA")
```

EtrsPoints-class *EtrsPoints class extends* "ETRS" *parrent class*

Description

Represents a set of lower corner grid points of the geographical grid ETRS89-LAEA

Slots

SpatialPointsDataFrame : an object of SPatialPointsDataFrame

etrs.cell.codes.columns : a character object that keeps the colnames of the dataframe

Examples

```
# Test for EtrsPoint
bb <- bbox(cbind(c(4321000, 4323000), c(3210000,3212000)))
cell.size = 1000
# Calculate the limits of the grid
bb.outer.limits <-
cbind((floor(bb[, 1] / cell.size)) * cell.size, (ceiling(bb[, 2] / cell.size))*cell.size)
eastings <- seq(from = bb.outer.limits[1, 1], to = bb.outer.limits[1, 2], by = cell.size)
#Calculates northings
northings <- seq(from = bb.outer.limits[2, 1], to = bb.outer.limits[2, 2], by = cell.size)
eastings.northings <- as.matrix(expand.grid(eastings, northings))
low.corner.cell.etrs.points<-etrsPoints(eastings.northings = eastings.northings,cell.size=10000)
plot(low.corner.cell.etrs.points)
text(eastings.northings[,1]+200,eastings.northings[,2],low.corner.cell.etrs.points[["NORTHOFORIGIN"]])
text(eastings.northings[,1],eastings.northings[,2]+200,low.corner.cell.etrs.points[["EASTOFORIGIN"]],srt=90)
title("Lower Cell Corner Etrs Points using ETRS-LAEA")
```

etrsPoints.default "etrsPoints" *is the* "EtrsPoints" *object constructor*

Description

"etrsPoints" is the "EtrsPoints" object constructor

Usage

```
etrsPoints.default(eastings.northings = "matrix", cell.size = "numeric")
```

Arguments

eastings.northings
: a matrix of coordinates

cell.size : a numeric object represnts the cell size of the grid

Value

An EtrsPoints object

Examples

```
# Test for EtrsPoint
bb <- bbox(cbind(c(4321000, 4323000), c(3210000,3212000)))
cell.size = 1000
# Calculate the limits of the grid
bb.outer.limits <- cbind((floor(bb[, 1]/cell.size))*cell.size,(ceiling(bb[,2]/cell.size))*cell.size)
eastings <- seq(from = bb.outer.limits[1, 1], to = bb.outer.limits[1, 2], by = cell.size)
#Calculates northings
northings <- seq(from = bb.outer.limits[2, 1], to = bb.outer.limits[2, 2], by = cell.size)
eastings.northings <- as.matrix(expand.grid(eastings, northings))
low.corner.cell.etrs.points<-etrsPoints(eastings.northings = eastings.northings,cell.size=1000)
plot(low.corner.cell.etrs.points)
text(eastings.northings[,1]+200,eastings.northings[,2],low.corner.cell.etrs.points[["NORTHOFORIGIN"]])
text(eastings.northings[,1],eastings.northings[,2]+200,low.corner.cell.etrs.points[["EASTOFORIGIN"]],srt=90)
title("Lower Cell Corner Etrs Points using ETRS-LAEA")
```

---

etrsPropValue — *Computes a single figure by each reference grid cell using Proportional calculation as integration methods*

Description

Proportional calculation: the cell takes a calculated value depending on the values of the units falling inside and their share within the cell. This method seems very appropriate for countable variables. Cell value = <ce><a3> (V i * Share i) V i = Value of unit i, Share i = Share of unit i within the cell

Usage

```
etrsPropValue(the.etrs.grid = "EtrsGrid",
  the.surface = "SpatialPoygonsDataFrame", surface.value.col = "numeric")
```

Arguments

the.etrs.grid	An object of the class EtrsGrid
the.surface	An object of class EtrsSurfaface

Value

An EtrsSurface

Examples

```
x <- round(runif(1, min = 4321000, max = 4322000), 1)
y <- round(runif(1, min = 3211000, max = 3212000), 1)
xy <- cbind(x, y)
size = round(runif(1,min = 1500,max = 1500),0)
pl1 <-Polygons(list(Polygon(cbind(c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)  ))),round(runif(1,min =
x <- x + size
pl2 <- Polygons(list(Polygon(cbind(c(x,x + size,x + size,x,x),c(y,y,y + 2 * size,y + 2 * size,y)))),round(runif(1
x<-x-size
y<-y+size
pl3 <-Polygons(list(Polygon(cbind(c(x,x + size,x + size,x,x),c(y,y,y +  size,y +  size,y)  ))),round(runif(1,min
sps <- SpatialPolygons(list(pl1,pl2,pl3),proj4string = CRS("+init=epsg:2100"))
df <-data.frame(val=c("R5","R40","R80"),row.names = sapply(slot(sps, "polygons"), function(x) slot(x, "ID")),VAL
Source.Surface <-SpatialPolygonsDataFrame(sps,data = df)
Source.Surface.propcal<-etrsSurface(input.surface = Source.Surface,over.method.type = "PropCal",surface.value.c
plot(Source.Surface.propcal,lty=3,lwd=1.2,border=3)
plot(EtrsTransform(Source.Surface),add=TRUE)
x.y.s.s <-coordinates(EtrsTransform(Source.Surface))
x.y.propcal <-coordinates(Source.Surface.propcal)
text(x.y.s.s[,1],x.y.s.s[,2],paste("ID=",Source.Surface@data$val),col=4,cex = 1.5)
text(x.y.s.s[,1],x.y.s.s[,2]-100,paste("Feature=",rownames(Source.Surface@data)),col=4,cex = 1.2)
text(x.y.s.s[,1],x.y.s.s[,2]-200,paste("VALUE =",Source.Surface@data$VALUE),col=4,cex = 1.5)
text(x.y.propcal[,1],x.y.propcal[,2]+100,Source.Surface.propcal@data$CELLVALUE,col=3)
```

`etrsPropWeightedValue` *Proportional and weighted calculation*

Description

the cell takes also a proportionally calculated value, but this value is weighted for each cell, according to an external variable (e.g. population). This method can be applied to improve the territorial distribution of a socioeconomic indicator. Cell value = W c <ce><a3> (V i * Share i) V i = Value of unit i Share i = Share of unit i within the cell, W c = weight assigned to cell

Usage

```
etrsPropWeightedValue(input.surface.grided = "EtrsSourceSurface",
  ancillary.grided = "EtrsAncillarySurface")
```

Arguments

`ancillary.grided`	an EtrsAncillarySurface
`etrs.grided.source.sur`	EtrsSourceSurface

Value

a EtrsDasymetricSurface

`etrsReverseCellCode,data.frame,numeric-method`
Reversing etrs cell codew from resoloution easting northings to res northing eastings

---

Description

It 's possible to find out, out there data not comabatible to Inspire. so reversin is a option. Such an example the GEOASTAT_grid_EU_POP .... files the method also set the data frame row names as etrs cell codes. As a result of it it can be used to as a spatial etrs grid

Usage

```
## S4 method for signature 'data.frame,numeric'
etrsReverseCellCode(df = "data.frame",
  cell.code.col = "numeric")
```

Arguments

`df`	. A data frame that has a Etrs reversed ceel code column
`cell.code.col`	. Define the column of the reversed cellcode

Value

a data frame with CELLCODE and ID = CELLCODE

`etrsSourceSurface,SpatialPolygonsDataFrame,ANY,ANY,numeric-method`
The constructor for an EtrsSourceSurface object

Description

Creates an EtrsSourceSurface class that holds the basic information for Source surface represanation in ETRS-LAEA grid form

Usage

```
  ## S4 method for signature 'SpatialPolygonsDataFrame,ANY,ANY,numeric'
etrsSourceSurface(input.surface,
  over.method.type, surface.value.col, cell.size)
```

Arguments

input.surface	SpatialPolygonsDataFrame.
over.method.type	ANY.
surface.value.col	ANY.
cell.size	numeric : cell.size numeric : one of the resolutions of the grid that is 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km

Value

an EtrsSourceSurface object

Examples

```
  x <- round(runif(1, min = 4320200, max = 4322000), 1)
  y <- round(runif(1, min = 3210000, max = 3212000), 1)
  xy <- cbind(x, y)
  size = round(runif(1,min = 1500,max = 2000),0)
  pl1 <-
    Polygons(list(Polygon(cbind(
      c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)
    ))),round(runif(1,min = 1,max = 100),0))
  x <- x + size

  pl2 <-
    Polygons(list(Polygon(cbind(
      c(x,x + size,x + size,x,x),c(y,y,y + 2 * size,y + 2 * size,y)
    ))),round(runif(1,min = 1,max = 100),0))
  sps <-
    SpatialPolygons(list(pl1,pl2),proj4string = CRS("+init=epsg:3035"))

  df <-
    data.frame(AREA = sapply(slot(sps, "polygons"), function(x)
     slot(x, "area")),VALUE=c(10,20),row.names = sapply(slot(sps, "polygons"), function(x)
      slot(x, "ID")))
  sps<-SpatialPolygonsDataFrame(sps,data = df)
sps.source<-etrsSourceSurface(input.surface = sps,over.method.type = "PropCal",surface.value.col = 2,cell.size =

plot(sps,border=2,lwd=3,col=rgb(.4,sps@data$VALUE/100,0,.25))
text(coordinates(sps)[,1],coordinates(sps)[,2],paste("VALUE=",sps@data$VALUE,sep=" "),cex=1)
text(coordinates(sps)[,1],coordinates(sps)[,2]+100,paste("ID=",row.names(sps@data),sep=" "),cex=1.2)
plot(sps.source,add=TRUE,lty=3,border=4)
text(coordinates(sps.source)[,1],coordinates(sps.source)[,2],sps.source@data$CELLVALUE,col=4)
```

EtrsSourceSurface-class
The EtrsSourceSurface class

Description

The EtrsSourceSurface class holds the basic information for Source surface represanation in ETRS-LAEA grid form

Value

An EtrsSourceSurface Object

Slots

`SpatialPolygonsDataFrame.` the input surface plus new data columns

`over.method.type` character. MaxArea for categorical data PropCal for numeric #'values

`cell.size` numeric. Indexing the column of data frame that contains the value of #'interest

`CELLVALUE` character. The size of the cell (the new map unit)

Examples

```
# test Source Surface -------------------------------------------------------

  x <- round(runif(1, min = 4320200, max = 4322000), 1)
  y <- round(runif(1, min = 3210000, max = 3212000), 1)
  xy <- cbind(x, y)
  size = round(runif(1,min = 1500,max = 2000),0)
  pl1 <-
    Polygons(list(Polygon(cbind(
      c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)
    ))),round(runif(1,min = 1,max = 100),0))
  x <- x + size

  pl2 <-
    Polygons(list(Polygon(cbind(
      c(x,x + size,x + size,x,x),c(y,y,y + 2 * size,y + 2 * size,y)
    ))),round(runif(1,min = 1,max = 100),0))
  sps <-
    SpatialPolygons(list(pl1,pl2),proj4string = CRS("+init=epsg:3035"))

  df <-
    data.frame(AREA = sapply(slot(sps, "polygons"), function(x)
     slot(x, "area")),VALUE=c(10,20),row.names = sapply(slot(sps, "polygons"), function(x)
       slot(x, "ID")))
  sps<-SpatialPolygonsDataFrame(sps,data = df)
sps.source<-etrsSourceSurface(input.surface = sps,over.method.type = "PropCal",surface.value.col = 2,cell.size =
```

```
plot(sps,border=2,lwd=3,col=rgb(.4,sps@data$VALUE/100,0,.25))
text(coordinates(sps)[,1],coordinates(sps)[,2],paste("VALUE=",sps@data$VALUE,sep=" "),cex=1)
text(coordinates(sps)[,1],coordinates(sps)[,2]+100,paste("ID=",row.names(sps@data),sep=" "),cex=1.2)
plot(sps.source,add=TRUE,lty=3,border=4)
text(coordinates(sps.source)[,1],coordinates(sps.source)[,2],sps.source@data$CELLVALUE,col=4)
```

---

etrsSurface,SpatialPolygonsDataFrame,ANY,missing,numeric-method

Etrs (source or Ancillary) surface creation

Description

Depending on each type of indicator or variable to be integrated within the reference grid, a different type of integration should be decided and tested. Besides the method finally chosen to integrate, it is important to highlight that indicator figures given by area unit, e.g. by square kilometre, should be converted considering that each cell has a total area of 1 km 2.

Usage

```
  ## S4 method for signature 'SpatialPolygonsDataFrame,ANY,missing,numeric'
etrsSurface(input.surface,
  over.method.type, surface.value.col, cell.size, ...)
```

Arguments

`over.method.type`	"PropCal" in proportional calculation the cell takes a calculated value depending on the values of the units falling inside and their share within the cell. This method seems very appropriate for countable variables
`surface.value.col`	the number of colum that keeps the relative density of a cell with land-cover some type
`cell.size:`	one of the resolutions of the grid that is 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km

Value

EtrsSurface

See Also

etrsSurface.default, etrsMaxArea, etrsPropCal

`etrsSurface,SpatialPolygonsDataFrame,character,numeric,numeric-method`
Etrs (source or Ancillary) surface creation

Description

Depending on each type of indicator or variable to be integrated within the reference grid, a different type of integration should be decided and tested. Besides the method finally chosen to integrate, it is important to highlight that indicator figures given by area unit, e.g. by square kilometre, should be converted considering that each cell has a total area of 1 km 2.

Usage

```
  ## S4 method for signature 'SpatialPolygonsDataFrame,character,numeric,numeric'
etrsSurface(input.surface = "SpatialPolygonsDataFrame",
  over.method.type = "character", surface.value.col = "numeric",
  cell.size = "numeric", ...)
```

Arguments

`over.method.type`	"PropCal" in proportional calculation the cell takes a calculated value depending on the values of the units falling inside and their share within the cell. This method seems very appropriate for countable variables
`surface.value.col`	the number of colum that keeps the relative density of a cell with land-cover some type
`cell.size:`	one of the resolutions of the grid that is 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km

Value

an EtrsSurface

See Also

etrsSurface.default, etrsMaxArea, etrsPropCal

etrsSurface,SpatialPolygonsDataFrame,missing,missing,numeric-method *Etrs (source or Ancillary) surface creation*

Description

Depending on each type of indicator or variable to be integrated within the reference grid, a different type of integration should be decided and tested. Besides the method finally chosen to integrate, it is important to highlight that indicator figures given by area unit, e.g. by square kilometre, should be converted considering that each cell has a total area of 1 km 2.

Usage

```
  ## S4 method for signature 'SpatialPolygonsDataFrame,missing,missing,numeric'
etrsSurface(input.surface,
  over.method.type, surface.value.col, cell.size, ...)
```

Arguments

over.method.type	"PropCal" in proportional calculation the cell takes a calculated value depending on the values of the units falling inside and their share within the cell. This method seems very appropriate for countable variables
surface.value.col	the number of colum that keeps the relative density of a cell with land-cover some type
cell.size:	one of the resolutions of the grid that is 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km

See Also

etrsSurface.default, etrsMaxArea, etrsPropCal

EtrsSurface-class *The EtrsSurface class*

Description

The etrs surface class holds the basic information for surface represanation in ETRS-LAEA grid from

Value

An EtrsSurface Object

Slots

`SpatialPolygonsDataFrame.` the input surface plus new data columns

`over.method.type` character. MaxArea for categorical data PropCal for numeric #'values

`cell.size` numeric. Indexing the column of data frame that contains the value of #'interest

`CELLVALUE` character. The size of the cell (the new map unit)

See Also

etrsSurface.default, etrsMaxArea, etrsPropCal

Examples

```
testpropcal <- function() {
x <- round(runif(1, min = 4320200, max = 4322000), 1)
y <- round(runif(1, min = 3210000, max = 3212000), 1)
xy <- cbind(x, y)
size = round(runif(1,min = 1500,max = 2000),0)
pl1 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)
  ))),round(runif(1,min = 1,max = 100),0))
x <- x + size
pl2 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + 2 * size,y + 2 * size,y)
  ))),round(runif(1,min = 1,max = 100),0))
x<-x-size
y<-y+size
pl3 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y +  size,y +  size,y)
  ))),round(runif(1,min = 1,max = 100),0))
sps <-
  SpatialPolygons(list(pl1,pl2,pl3),proj4string = CRS("+init=epsg:2100"))
df <-
 data.frame(val=c("R5","R40","R80"),row.names = sapply(slot(sps, "polygons"), function(x)
    slot(x, "ID")))
sps<-SpatialPolygonsDataFrame(sps,data = df)
}
Source.Surface <- testpropcal()

# Uses the default etrsSurface method
Source.Surface.MaxArea <-
etrsSurface(
  input.surface = Source.Surface, over.method.type = "MaxArea", cell.size = 1000
)
summary(Source.Surface.MaxArea)
```

`etrsSurface.default` *Etrs (source or Ancillary) surface creation*

Description

Depending on each type of indicator or variable to be integrated within the reference grid, a different type of integration should be decided and tested. Besides the method finally chosen to integrate, it is important to highlight that indicator figures given by area unit, e.g. by square kilometre, should be converted considering that each cell has a total area of 1 km 2.

Usage

```
etrsSurface.default(input.surface = "SpatialPolygonsDataFrame",
  over.method.type = "character", surface.value.col = "numeric",
  cell.size = "numeric", ...)
```

Arguments

`over.method.type`
: "MaxArea" The Maximum area criteria: the cell takes the value of the unit which covers most of the cell area. It should be a good option for uncountable variables "PropCal" in proportional calculation the cell takes a calculated value depending on the values of the units falling inside and their share within the cell. This method seems very appropriate for countable variables

`surface.value.col`
: the number of colum that keeps the relative density of a cell with land-cover some type

`...`

`cell.size:`
: one of the resolutions of the grid that is 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km

Value

an EtrsSurface

See Also

etrsMaxArea, etrsPropValue

Examples

```
testpropcal <- function() {
x <- round(runif(1, min = 4320200, max = 4322000), 1)
y <- round(runif(1, min = 3210000, max = 3212000), 1)
xy <- cbind(x, y)
size = round(runif(1,min = 1500,max = 2000),0)
```

```
pl1 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + size,y + size,y)
  ))),round(runif(1,min = 1,max = 100),0))
x <- x + size
pl2 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y + 2 * size,y + 2 * size,y)
  ))),round(runif(1,min = 1,max = 100),0))
x<-x-size
y<-y+size
pl3 <-
  Polygons(list(Polygon(cbind(
    c(x,x + size,x + size,x,x),c(y,y,y +  size,y +  size,y)
  ))),round(runif(1,min = 1,max = 100),0))
sps <-
  SpatialPolygons(list(pl1,pl2,pl3),proj4string = CRS("+init=epsg:2100"))
df <-
 data.frame(val=c("R5","R40","R80"),row.names = sapply(slot(sps, "polygons"), function(x)
   slot(x, "ID")))
sps<-SpatialPolygonsDataFrame(sps,data = df)
}
Source.Surface <- testpropcal()

# Uses the default etrsSurface method
Source.Surface.MaxArea <-
etrsSurface(input.surface = Source.Surface, over.method.type = "MaxArea", cell.size = 1000)
summary(Source.Surface.MaxArea)
```

`etrsSurface2Spdf` *Simple tool to convert a etrsSurface Object to Spatial*

Description

Simple tool to convert a etrsSurface Object to Spatial

Usage

```
etrsSurface2Spdf(the.etrs.surface = "EtrsSurface")
```

Arguments

`the.etrs.surface`

Value

SpatialPolygonsDataFrame

etrsSurfacePar,SpatialPolygonsDataFrame,ANY,numeric,numeric-method *etrsSurfacePar*

Description

etrsSurfacePar

Usage

```
  ## S4 method for signature 'SpatialPolygonsDataFrame,ANY,numeric,numeric'
etrsSurfacePar(input.surface,
  over.method.type, surface.value.col, cell.size)
```

Arguments

input.surface	SpatialPolygonsDataFrame.
over.method.type	ANY.
surface.value.col	numeric.
cell.size	numeric.

Value

an EtrsSurface

etrsSurfacePar,SpatialPolygonsDataFrame,character,missing,numeric-method *Etrs (source or Ancillary) surface creation (Parrarel)*

Description

Depending on each type of indicator or variable to be integrated within the reference grid, a different type of integration should be decided and tested. Besides the method finally chosen to integrate, it is important to highlight that indicator figures given by area unit, e.g. by square kilometre, should be converted considering that each cell has a total area of 1 km 2. plush Parallel computation Single Instruction, Multiple Data (SIMD) Parallel functions within an R script starts on single processor runs looped elements on multiple(total system cpus - 1) 'slave' processors returns results of all iterations to the original instance

Usage

```
  ## S4 method for signature 'SpatialPolygonsDataFrame,character,missing,numeric'
etrsSurfacePar(input.surface,
  over.method.type, surface.value.col, cell.size)
```

Arguments

over.method.type	"MaxArea" The Maximum area criteria: the cell takes the value of the unit which covers most of the cell area. It should be a good option for uncountable variables "PropCal" in proportional calculation the cell takes a calculated value depending on the values of the units falling inside and their share within the cell. This method seems very appropriate for countable variables
surface.value.col	the number of colum that keeps the relative density of a cell with land-cover some type
cell.size:	one of the resolutions of the grid that is 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km
...	

Value

an EtrsSurface

EtrsTableCodes,matrix,numeric-method

A method that produces a data frame of cell codes and lower corner easting northings

---

Description

A method that produces a data frame of cell codes and lower corner easting northings

Usage

```
## S4 method for signature 'matrix,numeric'
EtrsTableCodes(eastings.northings, cell.size)
```

Arguments

eastings.northings	a matrix populated with lower corner grid coordinates
cell.size:	one of the resolutions of the grid that is 0.5m, 1m, 2.5m, 5m, 10m, 25m, 50m, 100m, 250m, 500m, 1Km, 2.5Km, 5Km, 10Km, 25Km, 50Km, 100Km

Value

dataframe

Examples

```
bb <- bbox(cbind(c(4321000, 4323000), c(3210000,3212000)))
cell.size = 1000
# Calculate the limits of the grid
bb.outer.limits <-
 cbind((floor(bb[, 1] / cell.size)) * cell.size, (ceiling(bb[, 2] / cell.size)) * cell.size)
# Calculates eastings
eastings <-
 seq(from = bb.outer.limits[1, 1], to = bb.outer.limits[1, 2], by = cell.size)
# Calculates northings
northings <-
 seq(from = bb.outer.limits[2, 1], to = bb.outer.limits[2, 2], by = cell.size)
eastings.northings <- as.matrix(expand.grid(eastings, northings))
low.corner.cell.etrs.points<-etrsPoints(eastings.northings = eastings.northings,cell.size =1000)
df<-EtrsTableCodes(eastings.northings,1000)
df
```

EtrsTransform *Transforms from the current CRS to ETRS-LAEA*

Description

Transforms from the current CRS to ETRS-LAEA

Usage

```
EtrsTransform(obj)
```

Arguments

obj

Value

an EtrsSurface in ETRS CRS

joinMaxAreaSurfaceDataFrames *Simple tool to join etrsSurface with DataFrame*

Description

Simple tool to join etrsSurface with DataFrame

Usage

```
joinMaxAreaSurfaceDataFrames(the.surface = "SpatialPolygonsDataFrame",
  the.EtrsSurface = "EtrsSurface")
```

Arguments

the.surface	An input Surface
the.EtrsSurface	an EtrsSurface object

Value

an EtrsSurface

nama_10r_3gdp *Gross domestic product (GDP) at current market prices by NUTS 3 regions*

Description

Downloaded from Eurostat via eurostat package. Just a sample elected from the original dataset for NUTS65 and year 2001

Usage

```
nama_10r_3gdp
```

Format

An object of class `data.frame` with 10980 rows and 4 columns.

Note

<c2><a9> European Union, 1995-2016 Reuse is authorised, provided the source is acknowledged. The reuse policy of the European Commission is implemented by a Decision of 12 December 2011

Author(s)

Eurostat

NUTS3_OCMG *NUTS3_OMG : Boundaries of Greece administartive Units of level NUTS3*

Description

the data set was provided by Hellinic Organiztion of Cadastral and Mapping as the official boundaries for perfectures (NUTS3) greece. The data is under the creative commons licences 3.0 and is .shp file. The db file also contains some population data.

Usage

```
NUTS3_OCMG
```

Format

A Spatial polygons data frame

Note

Share <e2><80><94> copy and redistribute the material in any medium or format Adapt <e2><80><94> remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. CC 3.0

Source

http://geodata.gov.gr/en/dataset/6deb6a12-1a54-41b4-b53b-6b36068b8348/resource/3e571f7f-42a4-4b49-8db0-311695d72fa3/download/nomoiokxe.zip

NUTSV9_LEAC *NUTSV9_LEAC: Administrative land accounting units in Lambert Equal Area*

Description

GISCO administrative boundaries (NUTS) v9 generalised using the 1 km reference grid for the Land cover accounts project (LEAC) Administrative land accounting units are used to allocate land cover changes and to relate socio-economic processes to land cover dynamics. The GISCO database (Eurostat) provides a medium-scale layer for regional administrative boundaries (NUTS) covering the entire EU territory. The hierarchy of administrative land accounting units allows the analysis of the data at various scales, from NUTS 3 (province level) to NUTS 0 (country level). A mixed level, combining NUTS 3 and NUTS 2 was defined in order to homogenize the size of the units.

Usage

```
NUTSV9_LEAC
```

Format

A Spatial polygons data frame

Note

Rights: EEA standard re-use policy, unless otherwise indicated, re-use of content on the EEA website for commercial or non-commercial purposes is permitted free of charge, provided that the source is acknowledged (http://www.eea.europa.eu/legal/copyright). Copyright holder: European Environment Agency (EEA).

Author(s)

EEA

Source

http://epp.eurostat.ec.europa.eu/portal/page?_pageid=2254,62148876,2254_62153824&_dad=portal&_schema=PORTAL

NUTS_2013_01M_EL *Administrative or Statistical unit NUTS2013*

Description

At the beginning of the 1970s, Eurostat set up the NUTS classification as a single, coherent system for dividing up the EUs territory in order to produce regional statistics for the Community. For around thirty years, implementation and updating of the NUTS classification was managed under a series of "gentlemen s agreements" between the Member States and Eurostat.However, sometimes national interests require changing the regional breakdown of a country. When this happens the country concerned informs the European Commission about the changes.

@format Spatial Polygons Data Frames @source http://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units/nuts#nuts13 @author Eurostat @notes Copyright notice EN: <c2><a9> EuroGeographics for the administrative boundaries

Usage

```
NUTS_2013_01M_EL
```

Format

An object of class `SpatialPolygonsDataFrame` with 70 rows and 4 columns.

pntsattr2surface,SpatialPolygonsDataFrame,SpatialPointsDataFrame-method

pntsattr2surface is an ancillary utility for point surface combining attributes

Description

If the polygon contains 1 point then takes its attributes. if the polygon contains more than 2 points voronoi method of package deldir it's used to subdevide the region. If no point is contained in the region thn NA is provided as values

Usage

```
  ## S4 method for signature 'SpatialPolygonsDataFrame,SpatialPointsDataFrame'
pntsattr2surface(sppdf,
  sppntdf)
```

Arguments

sppdf	A sptialpolygonsdatframeobject
sppntdf	A spatialpointsdataframe object

Value

A spattial polygons data frame objec

raster2Ancillary,raster,numeric,numeric-method

create a EtrsAncillary Surface from a Raster oBject

Description

create a EtrsAncillary Surface from a Raster oBject

Usage

```
## S4 method for signature 'raster,numeric,numeric'
raster2Ancillary(aRaster, cell.size,
  attr_divisor = 1)
```

Arguments

aRaster	raster.
cell.size	numeric
attr_divisor	numeric.

Value

EtrsAncillarySurface

`wgsTransform,Spatial-method`
Transforms to wgs crs

Description

usefull for Google,Osm e.t.c.

Usage

```
## S4 method for signature 'Spatial'
wgsTransform(obj)
```

Arguments

`EtrsSurface`

Value

EtrsSurface

Index