Package ‘geozoning’

September 11, 2017

Title a Zoning Method for Spatial Data

Version 1.0.0

Author Brigitte Charnomordic [aut, cre],
Hazaël Jones[aut],
Patrice Loisel [aut]

Maintainer Hazaël Jones <hazael.jones@supagro.fr>

Description A set of zoning method and criteria for spatial data.

Depends R (>= 3.3.2)

License file LICENSE

Encoding UTF-8

LazyData true

Imports gstat, RandomFields, sp, maptools, deldir, fields, raster,
graphics, stats, utils,grDevices, rgeos

Suggests rmarkdown, knitr

RoxygenNote 6.0.1

VignetteBuilder knitr

R topics documented:

---

addContour *addContour*

Description

addContour

Usage

```
addContour(map, val, col = "blue", super = TRUE)
```

Arguments

map	object returned by function genMap
val	quantile value vector
col	color parameter
super	if TRUE add to existing plot lines coresponding to contour, if FALSE plot boundary and add lines

Details

add contour lines to plot

Value

void

Examples

```
data(mapTest)
addContour(mapTest,c(5,7),super=FALSE)
# not run
```

buffToValid *buffToValid*

Description

buffToValid

Usage

```
buffToValid(zone)
```

Arguments

zone	a SpatialPolygon

Details

function that check if a zone has a valid geometry, if not , makes zone valid by using gBuffer(width = 0,...)

Value

a new valid zone

cal.max.width.Zone *cal.max.width.Zone*

Description

cal.max.width.Zone

Usage

```
cal.max.width.Zone(z, step = 0.001, widthMax = 0.05, boundary,
  erosion = TRUE)
```

Arguments

z — spatial polygon

step — the difference between 2 values of parameter width in the function gBuffer

widthMax — the maximum value of the parameter width in gBuffer

boundary — union of all zones of the corrected map (result of correctBoundaryMap())

erosion — logical, if TRUE, compute the maximum value of width in case erosion->dilatation, otherwise in case dilatation->erosion

Details

function that return the maximal value of the parameter "width" in function gBuffer in order not to make zone disappear or not to split a zone into 2 differents zones

Value

maximum value of parameter width in the function smoothingZone

Examples

```
seed=1
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2)
criti = correctionTree(qProb = c(0.4,0.6), map = map)
Z = criti$zk[[2]][[1]]$zonePolygone
lab = criti$zk[[2]][[1]]$lab
# zones' correction
res = correctBoundaryMap(Zi = Z, map = map)
Z = res$Z
# map boundary after correction
boundary = Z[[1]]
for(i in 2:length(Z)){
  boundary = rgeos::gUnion(boundary, Z[[i]])
}
# plot map
plotM(map = map, Z = Z, lab = lab, byLab = FALSE)
widthMax = cal.max.width.Zone(z = Z[[3]], step = 0.001,
          widthMax = 0.05, boundary = boundary, erosion = TRUE)
zone = zone.extended(z = Z[[3]], boundary = boundary)
erosion1 = rgeos::gBuffer(zone ,width = - (widthMax + 0.002) ,joinStyle="ROUND",capStyle = "ROUND")
erosion2 = rgeos::gBuffer(zone ,width = - (widthMax - 0.002) ,joinStyle="ROUND",capStyle = "ROUND")
rgeos::plot(erosion1)
rgeos::plot(erosion2)
```

calcCritNarrow *detection of narrow zones (ratio area/perimeter^2)*

Description

detection of narrow zones (ratio area/perimeter^2)

Usage

```
calcCritNarrow(zonePolygone)
```

Arguments

zonePolygone	zoning

Details

computes for each zone of a zoning the ratio area/squared perimeter

Value

a numerical value

Examples

```
data(resZTest)
calcCritNarrow(resZTest$zonePolygone)
# not run
```

calcDCrit *calcDCrit*

Description

calcDCrit

Usage

```
calcDCrit(Z, map, optiCrit = 2, pErr = 0.9, simplitol = 0.001)
```

Arguments

Z	zoning geometry (list pf SpatialPolygons)
map	object returned by function genMap
optiCrit	criterion choice
pErr	equality tolerance for distance calculations, default 0.9
simplitol	tolerance for spatial polygons geometry simplification, default 0.001

Details

computes distances and criterion value for zoning Z

Value

a list with components

resD list with uncorrected and corrected distance matrix

resCrit list with criterion and cost values

Examples

```
data(mapTest)
data(resZTest)
Z=resZTest$zonePolygone
Z1=zoneFusion4(Z,6,2)
calcDCrit(Z1,mapTest)
# not run
```

calCrit *calCrit*

Description

calCrit

Usage

```
calCrit(matDistanceCorr, zoneNModif, optiCrit = 2)
```

Arguments

matDistanceCorr
: corrected distance matrix between zones, result of call to calDistance

zoneNModif
: modified zone neighborhood matrix (FALSE values on diagonal), result of call to calNei

optiCrit
: criterion to be optimized. Possible values are * 1 for min(mean(dij^2/(dii^2+dij^2))) * 2 for min(2*min(dij/(dii+djj))) * 3 for min(2*min(dij/(dii+djj))) * 4 for min(min(dij^2/sqrt(dii^2*djj^2)) * 5 for min(median(dij^2/sqrt(dii^2*djj^2))) * 7 for mean(2*mean(dij/(dii+djj)))

Details

wrapper function that redirects to the proper criterion calculation function according to optiCrit arg value

Value

the criterion value as a real positive number indicating the zoning quality.

Examples

```
# compute criterion on test zoning included in package
# load test map with simulated data
data(mapTest)
# load zoning results from test file
data(resZTest)
K=resZTest
resD = calDistance(typedist=1,mapTest$krigData,K$listZonePoint,
       K$zoneN,mapTest$krigSurfVoronoi,K$meanZone,pErr=0.9)
crit = calCrit(resD$matDistanceCorr,K$zoneNModif,2)
print(crit)
```

calCrit1 *calCrit1*

Description

calCrit1

Usage

```
calCrit1(matDistance, zoneNModif)
```

Arguments

`matDistance` zone distance matrix resulting from a call to calDistance

`zoneNModif` matrix of zone neigbors with FALSE on the diagonal

Details

computes a quality criterion equal to min(mean(dij^2/(dii^2+dij^2)))

Value

a numerical value

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
resD = calDistance(typedist=1,mapTest$krigData,K$listZonePoint,K$zoneN,
       mapTest$krigSurfVoronoi,K$meanZone,pErr=0.9)
calCrit1(resD$matDistanceCorr,K$zoneNModif)
# not run
```

calCrit2 *calCrit2*

Description

calCrit2

Usage

```
calCrit2(matDistance, zoneNModif)
```

Arguments

matDistance	zone distance matrix resulting from a call to calDistance
zoneNModif	matrix of zone neigbors with FALSE on the diagonal

Details

computes a quality criterion equal to min(2*min(dij/(dii+djj)))

Value

a numerical value equal to min(mean(dij^2/(dii^2+dij^2)))

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
resD = calDistance(typedist=1,mapTest$krigData,K$listZonePoint,K$zoneN,
        mapTest$krigSurfVoronoi,K$meanZone,pErr=0.9)
calCrit2(resD$matDistanceCorr,K$zoneNModif)
# not run
```

calCrit2bis *calCrit2bis*

Description

calCrit2bis

Usage

```
calCrit2bis(matDistance, zoneNModif)
```

Arguments

matDistance zone distance matrix resulting from a call to calDistance

zoneNModif matrix of zone neigbors with FALSE on the diagonal

Details

computes a quality criterion equal to min(min(dij/(dii^2+dij^2)))

Value

a numerical value

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
resD = calDistance(typedist=1,mapTest$krigData,K$listZonePoint,K$zoneN,
       mapTest$krigSurfVoronoi,K$meanZone,pErr=0.9)
calCrit2(resD$matDistanceCorr,K$zoneNModif)
# not run
```

calCrit3 *calCrit3*

Description

calCrit3

Usage

```
calCrit3(matDistance, zoneNModif)
```

Arguments

matDistance zone distance matrix resulting from a call to calDistance

zoneNModif matrix of zone neigbors with FALSE on the diagonal

Details

computes a quality criterion equal to min(mean(dij^2/sqrt(dii^2*dij^2)))

Value

a numerical value

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
resD = calDistance(typedist=1,mapTest$krigData,K$listZonePoint,K$zoneN,
      mapTest$krigSurfVoronoi,K$meanZone,pErr=0.9)
calCrit3(resD$matDistanceCorr,K$zoneNModif)
# not run
```

---

calCrit4 *calCrit4*

Description

calCrit4

Usage

```
calCrit4(matDistance, zoneNModif)
```

Arguments

matDistance	zone distance matrix resulting from a call to calDistance
zoneNModif	matrix of zone neigbors with FALSE on the diagonal

Details

computes a quality criterion equal to min(min(dij^2/sqrt(dii^2*djj^2)))

Value

a numerical value

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
resD = calDistance(typedist=1,mapTest$krigData,K$listZonePoint,K$zoneN,
      mapTest$krigSurfVoronoi,K$meanZone,pErr=0.9)
calCrit4(resD$matDistanceCorr,K$zoneNModif)
# not run
```

calCrit5 *calCrit5*

Description

calCrit5

Usage

```
calCrit5(matDistance, zoneNModif)
```

Arguments

matDistance	zone distance matrix resulting from a call to calDistance
zoneNModif	matrix of zone neigbors with FALSE on the diagonal

Details

computes a quality criterion equal to min(median(dij/sqrt(dii*dij)))

Value

a numerical value

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
resD = calDistance(typedist=1,mapTest$krigData,K$listZonePoint,K$zoneN,
       mapTest$krigSurfVoronoi,K$meanZone,pErr=0.9)
calCrit5(resD$matDistanceCorr,K$zoneNModif)
# not run
```

calCrit7 *calCrit7*

Description

calCrit7

Usage

```
calCrit7(matDistance, zoneNModif)
```

Arguments

matDistance	zone distance matrix resulting from a call to calDistance
zoneNModif	matrix of zone neigbors with FALSE on the diagonal

Details

computes a quality criterion equal to mean(2*mean(dij/(dii+djj)))

Value

a numerical value

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
resD = calDistance(typedist=1,mapTest$krigData,K$listZonePoint,K$zoneN,
        mapTest$krigSurfVoronoi,K$meanZone,pErr=0.9)
calCrit7(resD$matDistanceCorr,K$zoneNModif)
# not run
```

---

calCritMinMean *calCritMinMean*

Description

calCritMinMean

Usage

```
calCritMinMean(matDistance, zoneNModif)
```

Arguments

matDistance	zone distance matrix resulting from a call to calDistance
zoneNModif	matrix of zone neigbors with FALSE on the diagonal

Details

computes a quality criterion equal to min(mean(dij^2/sqrt(dii^2*djj^2)))

Value

a numerical value

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
resD = calDistance(typedist=1,mapTest$krigData,K$listZonePoint,K$zoneN,
        mapTest$krigSurfVoronoi,K$meanZone,pErr=0.9)
calCritMinMean(resD$matDistanceCorr,K$zoneNModif)
# not run
```

---

calDistance *calDistance*

Description

calDistance

Usage

```
calDistance(typedist = 1, tabVal = NULL, listZonePoint = NULL,
  zoneN = NULL, surfVoronoi = NULL, meanZone = NULL, pErr = 0.9)
```

Arguments

typedist	default value is 1, other values not implemented yet.
tabVal	SpatialPointsDataFrame, contains data points to be used for zoning (spatial coordinates plus attribute values) result of call to genMap
listZonePoint	list of indices of data points within zones, result of call to calNei
zoneN	zone neighborhood matrix (TRUE values on diagonal), result of call to calNei
surfVoronoi	vector of Voronoi polygon surfaces corresponding to all data points,result of call to genMap
meanZone	vector of average attribute values for all zones
pErr	error percentage for correcting distances

Details

calculates matrix of heterogeneities between neighbour zones. max(sigmai2[i],(fxmean*pErr/100)^2) + max(sigmai2[j],(fymean*pErr/100)^2) + (fxmean-fymean)^2

Value

a list with components

matDistance matrix of real values, corresponding to heterogeneities between neighbour zones. All other values are set to 0.

matDistanceCorr corrected distance matrix using pErr

cost sum or errors obtained by replacing all data values within a zone by the zone mean value

Examples

```
# load test map with simulated data
data(mapTest)
# load zoning results from test file
data(resZTest)
K=resZTest
resD = calDistance(typedist=1,mapTest$krigData,K$listZonePoint,K$zoneN,
      mapTest$krigSurfVoronoi,K$meanZone,pErr=0.9)
```

---

calFrame *calFrame*

Description

calFrame

Usage

```
calFrame(iZ, Z, zoneNModif, distIsoZ = 0.075)
```

Arguments

iZ	index of zone for which the envelope is searched
Z	zoning
zoneNModif	modified zone neighborhood matrix (FALSE values on diagonal
distIsoZ	threshold distance above which a zone is considered as isolated

Details

description, a paragraph

Value

a apatial polygon corresponding to the frame within which grown zone must be contained

Examples

```
data(resZTest)
Z=resZTest$zonePolygone
zN=resZTest$zoneNModif
f=calFrame(6,Z,zN)
plotZ(Z)
rgeos::plot(f,add=TRUE,col="red")
```

calGearyGlo *calGearyGlo*

Description

calGearyGlo

Usage

```
calGearyGlo(matN, vectMean, meanTot, vectSurface)
```

Arguments

matN	xxxx
vectMean	xxxx
meanTot	xxxx
vectSurface	xxxx

Details

computes global Geary criterion

Value

a ?

Examples

```
# not run
```

calGearyLoc *local Geary criteria*

Description

local Geary criteria

Usage

```
calGearyLoc(matN, vectMean, meanTot, vectSurface)
```

Arguments

matN	neighborhood (zone or point) matrix
vectMean	vector of mean zone values
meanTot	global mean
vectSurface	vector of zone areas

Details

computes local Geary indices

Value

a vector of local Geary criteria

Examples

```
K=resZTest
zoneA=sapply(K$zonePolygone,rgeos::gArea)
calGearyLoc(K$zoneNModif,K$meanZone,K$meanTot,zoneA)
# not run
```

---

calMoranBLocal *compute local Moran indices (per zone)*

Description

compute local Moran indices (per zone)

Usage

```
calMoranBLocal(NZone, matDistanceMoranB, vectSurface)
```

Arguments

NZone	xxxx
matDistanceMoranB	xxxx
vectSurface	xxxx

Details

description, a paragraph

Value

a ?

Examples

```
# not run
```

calMoranBTot *computes Moran criterion on whole zoning*

Description

computes Moran criterion on whole zoning

Usage

```
calMoranBTot(NZone, matDistanceMoranB, vectSurface)
```

Arguments

NZone	xxxx
matDistanceMoranB	xxxx
vectSurface	xxxx

Details

computes Moran criterion on zoning

Value

a ?

Examples

```
# not run
```

calMoranGlo *computes specific Moran criterion*

Description

computes specific Moran criterion

Usage

```
calMoranGlo(matNZone, vectMean, meanTot, vectSurface)
```

Arguments

matNZone	xxxx
vectMean	xxxx
meanTot	xxxx
vectSurface	xxxx

Details

description, a paragraph

Value

a ?

Examples

```
# not run
```

calMoranLoc *calMoranLoc*

Description

calMoranLoc

Usage

```
calMoranLoc(matN, vectMean, meanTot, vectSurface)
```

Arguments

matN	xxxx
vectMean	xxxx
meanTot	xxxx
vectSurface	xxxx

Details

description, a paragraph

Value

a ?

Examples

```
# not run
```

calNei *calNei*

Description

calNei

Usage

```
calNei(Z, spdata, surfVoronoi, ptN, simplitol = 0.001, remove = TRUE,
  correct = FALSE, nmin = 2)
```

Arguments

Z	zoning geometry (list pf SpatialPolygons)
spdata	SpatialPointsDataFrame containing the data pts and values
surfVoronoi	Surfaces of the Voronoi polygons corresponding to data pts
ptN	indices of data pts neighbours
simplitol	tolerance for spatial polygons geometry simplification
remove	if TRUE remove zones with less than nmin data points
correct	if TRUE correct zone neighborhood
nmin	number of points below wich a zone is removed from the zoning

Details

calculates neighborhood for zoning geometry Z (list of SpatialPolygons)

Value

a list with components

zoneN matrix of zone neigbors

zoneNModif modified matrix with FALSE on the diagonal

listZonePoint indices of pts within each zone

meanTot zoning mean data value

meanZone vector of zone data mean values

listSurf vector of zone areas

critSurf vector of filiform zone characteristics

zonePolygone list of zones, each zone is a SpatialPolygons

Examples

```
data(mapTest)
ptN=mapTest$krigN
spdata=mapTest$krigData
surfVoronoi=mapTest$surfVoronoi
data(resZTest)
Z=resZTest$zonePolygone
K=calNei(Z,spdata,surfVoronoi,ptN)
names(K)
plotZ(K$zonePolygone)
K=calNei(Z,spdata,surfVoronoi,ptN,nmin=20) #keep only zones with a minimum of 20 data points
plotZ(K$zonePolygone)
```

---

calRMmodel *transform VGM model into model usable by RandomFields*

Description

transform VGM model into model usable by RandomFields

Usage

```
calRMmodel(vgmodel)
```

Arguments

vgmodel	model provided by a call to vgm

Value

model suitable for RandomFields simulation

Examples

```
modv=gstat::vgm(model="Gau",range=100,psill=10,mean=7)
RMmodel=calRMmodel(modv)
```

calStep *compute step for non square grid*

Description

compute step for non square grid

Usage

```
calStep(nPointsK, xsize, ysize)
```

Arguments

nPointsK	numeric value giving the number of points after kriging
xsize	numeric value giving the data range on the x axis
ysize	numeric value giving the data range on the y axis

Value

a numerical step value

Examples

```
calStep(1000,1,1)
# not run
```

calZoneN *calZoneN*

Description

calZoneN

Usage

```
calZoneN(ptN, zoneN, listZonePoint)
```

Arguments

ptN	pt neighborhood Logical matrix
zoneN	empty zone neighborhood Logical matrix
listZonePoint	list of indices of data points within zones

Details

calculate zone neighborhood

Value

a list with component zoneN holding filled zone neighborhood Logical matrix

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
ptN=mapTest$krigN
nZ=length(K$zonePolygone)
zoneN=matrix(logical(nZ*nZ),nZ,nZ)
listZonePoint=K$listZonePoint
calZoneN(ptN,zoneN,listZonePoint)
# not run
```

---

checkContour *checkContour*

Description

checkContour

Usage

```
checkContour(contourSp, step, refPoint, minSizeNG = 0.001)
```

Arguments

contourSp	SpatialPolygons corresponding to closed contour line
step	grid resolution
refPoint	referene point
minSizeNG	zone area threshold under which a zone is not admissible

Details

check admissibility for contour line: surface >minSizeNG and refPoint close enough

Value

Null if contour is not admissible or a list with components

contourSp SpatialPolygons corresponding to admissible contour

polyBuffSpatialPolygons corresponding to gBuffer around admissible contour

Examples

```
data(mapTest)
cL=contourAuto(list(),mapTest$step,mapTest$xsize,mapTest$ysize,
  mapTest$krigGrid,c(5,7),mapTest$boundary)
pG=polyToSp2(sp::Polygon(mapTest$boundary)) #SpatialPolygons corresponding to map boundary
rgeos::plot(pG)
sp8 = contourToSpp(cL[[8]],0.1)$sp
refPoint = rgeos::gCentroid(sp8)
resp=checkContour(sp8,mapTest$step,refPoint)
rgeos::plot(resp$contourSp,col="red",add=TRUE)
# not run
```

---

cleanSp *cleanSp*

Description

cleanSp

Usage

```
cleanSp(sp, tol = 1e-05)
```

Arguments

sp	SpatialPolygons
tol	minimum area for removal

Details

removes from sp polygons that are too small (artefacts of gDifference)

Value

a SpatialPolygons

Examples

```
# not run
```

contourArea *contourArea*

Description

contourArea

Usage

```
contourArea(co)
```

Arguments

co	contour line

Details

area corresponding to closed contour line

Value

the area within the contour line

Examples

```
data(mapTest)
cL=list()
cL=contourAuto(cL,mapTest$step,mapTest$xsize,mapTest$ysize,mapTest$krigGrid,c(5,7),mapTest$boundary)
contourArea(cL[[8]])
# not run
```

contourAuto *contourAuto*

Description

contourAuto

Usage

```
contourAuto(cL, step, xsize, ysize, matVal, vRef, boundary, GridData = FALSE)
```

Arguments

cL	empty or existing list of contour lines
step	grid step as returned by calStep
xsize	size of map along x-axis
ysize	size of map along y-axis
matVal	dataframe with data values organized into a grid
vRef	quantile vector
boundary	list, contains x and y dy on a regular grid
GridData	logical value indicating if data are already on a regular grid

Details

builds contout Lines qith the quantile vector given in argument and closes them with the map border

Value

a list of contour lines

Examples

```
data(mapTest)
cL=list()
cL=contourAuto(cL,mapTest$step,mapTest$xsize,mapTest$ysize,mapTest$krigGrid,c(5,7),mapTest$boundary)
plot(mapTest$boundary,type="l",col="red")
linesC(cL)
# not run
```

---

contourBetween *contourBetween*

Description

contourBetween

Usage

```
contourBetween(map, krigGrid, q1, q2, nbContourBetween = 5)
```

Arguments

map	: object map defined in package geozoning
krigGrid	: object that can
q1, q2	: 2 quantiles that defined zone
nbContourBetween	: the number of discretisation between q1 and q2

Details

: For the given krigGrid, this funtion returns the contourLines of the map following the 2 quantiles that defined at the beginning.

Value

listContours : List of Spatial Lines and the value of quantile that represent the contours generated

Examples

```
seed=2
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2)
ZK=initialZoning(qProb=c(0.55,0.85),map)
Z=ZK$resZ$zonePolygone # list of zones
lab = ZK$resZ$lab # label of zones
plotM(map = map,Z = Z,lab = lab, byLab = FALSE)
# zone 6 is a transition zone that has commun boundary with the map
numZ = 6
Estimation = Transition_Zone_Near_Boundary(map = map, Z = Z, numZ = numZ)
result = new_krigGrid_for_visualisation(map = map, Z = Z, numZ = numZ, solution = Estimation)
new_krigGrid = result$new_krigGrid
new_data = result$new_data
quant1 = quantile(map$krigData@data$var1.pred,probs = 0.55)
quant2 = quantile(map$krigData@data$var1.pred,probs = 0.85)
# plot modified isocontours
plotM(map = map,Z = Z,lab = lab, byLab = TRUE)
listContours = contourBetween(map = map, krigGrid = new_krigGrid, q1 = quant1, q2 = quant2)
for (i in 1:length(listContours)){
  sp::plot(listContours[[i]]$contour,add=TRUE,col = "red")
}
```

---

contourToSpp *contourToSpp*

Description

contourToSpp

Usage

```
contourToSpp(co, step)
```

Arguments

co	contour line (list with contour level and x,y coordinates
step	grid resolution

Details

transform contour line into SpatialPolygons

Value

a list with components

sp SpatialPolygons corresponding to contour line

contour SpatialLines corresponding to contour line

polyBuff SpatialPolygons corresponding to buffer around contour line

surface SpatialPolygons area

Examples

```
data(mapTest)
cL=list()
cL=contourAuto(cL,mapTest$step,mapTest$xsize,mapTest$ysize,mapTest$krigGrid,c(5,7),mapTest$boundary)
contourToSpp(cL[[8]],0.1)
# not run
```

correctBoundaryMap *correctBoundaryMap*

Description

correctBoundaryMap

Usage

```
correctBoundaryMap(Zi, map)
```

Arguments

Zi	list of initiales zones
map	object returned by function genMap

Details

function for post treatment of zoning that fixes the problem linked to the border between two neighbour zones and between zones and the map boundary

Value

new list of zones with correct boundary ang the parameter "width" used for correction

Examples

```
seed=1
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2)
criti = correctionTree(qProb = c(0.5), map = map)
Z = criti$zk[[1]][[1]]$zonePolygone
lab = criti$zk[[1]][[1]]$lab
plotM(map = map, Z = Z, lab = lab, byLab = FALSE)
class(rgeos::gIntersection(Z[[1]],Z[[2]])) [1]
class(rgeos::gIntersection(Z[[1]],Z[[5]])) [1]
class(rgeos::gIntersection(Z[[2]],Z[[3]])) [1]
class(rgeos::gIntersection(Z[[2]],Z[[4]])) [1]
res = correctBoundaryMap(Zi = Z, map = map)
Z = res$Z
class(rgeos::gIntersection(Z[[1]],Z[[2]])) [1]
class(rgeos::gIntersection(Z[[1]],Z[[5]])) [1]
class(rgeos::gIntersection(Z[[2]],Z[[3]])) [1]
class(rgeos::gIntersection(Z[[2]],Z[[4]])) [1]
plotM(map = map, Z = Z, lab = lab, byLab = FALSE)
```

---

correctionTree *correctionTree*

Description

correctionTree

Usage

```
correctionTree(qProb, map, pErr = 0.9, optiCrit = 2, minSize = 0.012,
  minSizeNG = 0.001, distIsoZ = 0.075, simplitol = 0.001, LEQ = 5,
  MAXP = 0.1, LASTPASS = TRUE, disp = 0, SAVE = TRUE, ONE = FALSE,
  ALL = FALSE)
```

Arguments

qProb	probability vector used to generate quantile values
map	object returned by function genMap
pErr	equality tolerance for distance calculations
optiCrit	criterion choice
minSize	zone area threshold under which a zone is too small to be manageable
minSizeNG	zone area threshold under which a zone will be removed
distIsoZ	threshold distance to next zone, above which a zone is considered to be isolated
simplitol	tolerance for spatial polygons geometry simplification
LEQ	length of quantile sequence used to grow isolated zone
MAXP	quantile sequence maximum shift quantile sequence maximum shift

LASTPASS if TRUE, remove zones that are still too small at the last level of the correction tree

disp 0: no info, 1: some info, 2: detailed info

SAVE logical value, if TRUE function returns last level zonings, if FALSE function only returns best last level results

ONE logical value, if TRUE function returns only criterion value

ALL logical value, if TRUE function returns zonings at all levels

Details

description, a paragraph

Value

a list with components

bestcrit best criterion value at last level (in all cases)

critList criterion values at last level (in all cases if ONE=FALSE)

costList cost values at last level (in all cases if ONE=FALSE)

costLList cost per label values at last level (in all cases if ONE=FALSE)

nzList vector of number of zones at last level (in all cases if ONE=FALSE)

qProb vector of probabilities values used for quantiles (in all cases if ONE=FALSE)

zk list of zoning objects (such as returned by calNei function), first element corresponds to initial zoning, each other element is a list with each (last if ALL=FALSE) level zoning objects (only if SAVE=TRUE)

mdist list of initial distance matrix and all (last if ALL=FALSE) level distance matrices (only if SAVE=TRUE)

criterion list of initial criterion and all (last if ALL=FALSE) level criteria (only if SAVE=TRUE)

cost list of initial cost and all (last if ALL=FALSE) level costs (only if SAVE=TRUE)

costL list of initial cost per label and all (last if ALL=FALSE) level costs per label (only if SAVE=TRUE)

nz list of initial number of zones and all (last if ALL=FALSE) level number of zones (only if SAVE=TRUE)

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.7),mapTest,SAVE=TRUE)
plotZ(criti$zk[[1]][[1]]$zonePolygone)
plotZ(criti$zk[[2]][[1]]$zonePolygone) # zones 7 and 8 were handled
```

correctN *correctN*

Description

correctN

Usage

```
correctN(Z, zoneN, dN = 0.001)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
zoneN	zone neighborhood Logical matrix
dN	maximum distance beyond which 2 zones cannot be considered as neighbors

Details

description, a paragraph

Value

a new zone neighborhood Logical matrix

Examples

```
data(resZTest)
Z=resZTest$zonePolygone
H=correctN(Z,resZTest$zoneN,1e-8)
# not run
```

costLab *costLab*

Description

costLab

Usage

```
costLab(K, map)
```

Arguments

K	zoning object, as returned by the calNei function
map	object returned by genMap function

Details

description, a paragraph

Value

the sum of per label costs

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.7),mapTest,SAVE=TRUE)
K=criti$zk[[1]][[1]] # initial zoning
costLab(K,mapTest) #identical to criti$costL[[1]][[1]]
# not run
```

Cost_By_Laplace *Cost_By_Laplace*

Description

Cost_By_Laplace

Usage

```
Cost_By_Laplace(map, Z, numZ, Estimation)
```

Arguments

map	object returned by function genMap or genMapR
Z	: an example of zoning (a list of zones)
numZ	: number of the zone in which the cost will be computed
Estimation	: value of linear interpolation by solving Laplace's equation

Details

: function that returns the criterion COST by approximating the valeur in a point of the grid by the linear interpolation (approximate solution of Laplace's equation. For more details see help of function Transition_Zone_Near_Boundary, Transition_Zone_Far_Boundary or Extreme_Zone)

Value

cost computed by replacing values in zone by linear interpolation

Examples

```
seed=2
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2)
ZK=initialZoning(qProb=c(0.55,0.85),map)
Z=ZK$resZ$zonePolygone # list of zones
lab = ZK$resZ$lab # label of zones
plotM(map = map,Z = Z,lab = lab, byLab = FALSE)
# zone 6 is a transition zone that has commun boundary with the map
numZ = 6
Estimation = Transition_Zone_Near_Boundary(map = map, Z = Z, numZ = numZ)
# compute the cost
cL = Cost_By_Laplace(map = map, Z = Z, numZ = numZ, Estimation = Estimation)
cM = Cost_By_Mean(map = map, Z = Z, numZ = numZ)
print(cL$cost_Laplace)
print(cM$cost_Mean)
# zone 6 is a zone with gradient
```

Cost_By_Mean *Cost_By_Mean*

Description

Cost_By_Mean

Usage

```
Cost_By_Mean(map, Z, numZ)
```

Arguments

map	object returned by function genMap or genMapR
Z	: an example of zoning (a list of zones)
numZ	: number of the zone in which the cost will be computed

Details

: function that returns the criterion COST by approximating the valeur in a point of the grid by the mean value of the zone.

Value

the cost value as described

Examples

```
seed=2
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2)
ZK=initialZoning(qProb=c(0.55,0.85),map)
Z=ZK$resZ$zonePolygone # list of zones
lab = ZK$resZ$lab # label of zones
plotM(map = map,Z = Z,lab = lab, byLab = FALSE)
# zone 6 is a transition zone that has commun boundary with the map
numZ = 6
Estimation = Transition_Zone_Near_Boundary(map = map, Z = Z, numZ = numZ)
# compute the cost
cL = Cost_By_Laplace(map = map, Z = Z, numZ = numZ, Estimation = Estimation)
cM = Cost_By_Mean(map = map, Z = Z, numZ = numZ)
print(cL$cost_Laplace)
print(cM$cost_Mean)
# zone 6 is a zone with gradient
```

---

crComment *crComment*

Description

crComment

Usage

```
crComment(Z)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)

Details

create comment corresponding to holes in a zoning

Value

a zoning

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
Z1=crComment(Z)
# not run
```

createHoles *createHoles*

Description

createHoles

Usage

```
createHoles(Z)
```

Arguments

Z	list of zones, each zone is a SpatialPolygons

Details

description, a paragraph

Value

a list of zones where holes are distinct SpatialPolygons

Examples

```
# not run
```

datanorm *normalize data coordinates and border*

Description

normalize data coordinates and border

Usage

```
datanorm(data, bd)
```

Arguments

data	data frame with x and y components
bd	boundary (list with x and y components)

Details

normalize boundary between 0 and 1 and data coordinates accordingly

Value

a list with components

dataN normalized data

boundaryN normalized boundary

xmin minimum vaue of x within boundary

xmax maximum vaue of x within boundary

ymin minimum vaue of y within boundary

ymax maximum vaue of y within boundary

Examples

```
x=runif(100, min=0, max=1)
y=runif(100, min=0.2, max=1.7)
range(x) # not [0,1]
tabData=data.frame(x=x,y=y)
bd=list(x=c(0,0,1,1,0), y=c(0.2,1.7,1.7,0.2,0.2))
res=datanorm(tabData,bd)
apply(res$dataN,2,range)#
# not run
```

---

datanormX *normalize data coords with same ratio (for non square field)*

Description

normalize data coords with same ratio (for non square field)

Usage

```
datanormX(data, bd)
```

Arguments

data	frame with x and y components
bd	list with x and y components

Details

normalize x between 0 and 1, y and boundary with same ratio

Value

a list with components

dataN normalized data

boundaryN normalized boundary

ratio normalizing ratio

xmin minimum value of x within boundary

xmax maximum value of x within boundary

ymin minimum value of y within boundary

ymax maximum value of y within boundary

Examples

```
x=runif(100, min=0, max=1)
y=runif(100, min=0.2, max=1.7)
range(x) # not [0,1]
tabData=data.frame(x=x,y=y)
bd=list(x=c(0,0,1,1,0), y=c(0.2,1.7,1.7,0.2,0.2))
res=datanormX(tabData,bd)
apply(res$dataN,2,range)# x range is now [0,1], not y range
res$ratio # normalization ratio
# not run
```

datanormXY *normalize data coords*

Description

normalize data coords

Usage

```
datanormXY(data)
```

Arguments

data	frame with x and y components

Details

normalize data coordinates between 0 and 1 with different ratios for x and y

Value

a normalized data frame

Examples

```
nPoints=500
x=runif(nPoints, min=0, max=1)
y=runif(nPoints, min=0, max=1)
range(x) # not [0,1]
tabData=data.frame(x=x,y=y)
tabData=datanormXY(tabData) # x,y ranges are now [0,1]
# not run
```

dataReg *A data frame with simulated data on a regular grid*

Description

A data frame with simulated data on a regular grid

Usage

```
dataReg
```

Format

a data frame containing a regular grid with 1936 rows and 3 variables

x x coordinate

y y coordinate

z numeric variable - simulated

detectSmallZones *detectSmallZones*

Description

detectSmallZones

Usage

```
detectSmallZones(zonePolygone, minSize)
```

Arguments

zonePolygone	list of zones, each zone is a SpatialPolygons
minSize	zone area threshold under which a zone is too small to be manageable

Details

detect zones with area < minSize

Value

a vector of small zones indices

Examples

```
data(mapTest)
ZK=initialZoning(qProb=c(0.4,0.7),mapTest)
Z=ZK$resZ$zonePolygone
minSize=0.012
iSmall=detectSmallZones(Z,minSize) # 2 small zones
# not run
```

detZoneClose *detZoneClose*

Description

detZoneClose

Usage

```
detZoneClose(iZ, Z, zoneN, distIsoZ = 0.075)
```

Arguments

iZ	zone number
Z	zoning geometry (list of SpatialPolygons)
zoneN	modified zone neighborhood Logical matrix (FALSE values on diagonal)
distIsoZ	threshold distance above which a zone is considered as isolated

Details

determines zones that are close to current zone, but not neighbors (common border). Therefore embedded or englobing zones are excluded.

Value

a list with components

InterZoneSpace TRUE if zone is isolated, FALSE otherwise

zoneClose indices of zones close to zone iZ, empty if zone is isolated

Examples

```
data(resZTest)
Z=resZTest$zonePolygone
zoneN=resZTest$zoneNModif
plotZ(Z)
detZoneClose(4,Z,zoneN) # zone 4 is close to zone 3
detZoneClose(6,Z,zoneN) # zone 6 is isolated (no zone at a distance smaller than 0.075).
# not run
```

---

detZoneEng *detZoneEng*

Description

detZoneEng

Usage

```
detZoneEng(iZ, Z, zoneN)
```

Arguments

iZ	index of zone for which englobing zone is searched
Z	zoning
zoneN	modified zone neighborhood matrix (FALSE values on diagonal)

Details

description, a paragraph

Value

an integer value (0 if no englobing zone was found, englobing zone index otherwise)

Examples

```
# load zoning results from test file
data(resZTest)
Z=resZTest$zonePolygone
zoneN=resZTest$zoneNModif
detZoneEng(3,Z,zoneN) # zone 2 englobes zone 3
detZoneEng(2,Z,zoneN) # no englobing zone for zone 2
```

DIJ *DIJ*

Description

DIJ

Usage

```
DIJ(i, j, sigmai2, meanZone, pErr)
```

Arguments

i	zone index
j	neighbor zone index
sigmai2	vector of zone variances
meanZone	list of zone mean values
pErr	tolerance for distance correction

Details

description, a paragraph

Value

a list with components d and dCorr

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
nz=length(K$zonePolygone)
si2=rep(NA,nz)
for (kk in 1:nz){
si2[kk]=Sigmai2(kk,K$listZonePoint,mapTest$krigData,
       mapTest$krigSurfVoronoi,K$meanZone)$sigmai2
}
d12=DIJ(1,2,si2,K$meanzone,0.9)
# not run
```

dispZ *dispZ*

Description

dispZ

Usage

```
dispZ(step, matVal, nbLvl = 0, zonePolygone = NULL, K = NULL,
  colBreaks = 0, texMain = "", boundary = NULL, id = FALSE,
  valQ = NULL, palCol = colorRampPalette(c("brown", "yellow")),
  noXY = FALSE, iZ = 0, mu = 1, cex = 1, ptz = NULL)
```

Arguments

step	grid resolution
matVal	data frame of values
nbLvl	number of contour lines to generate
zonePolygone	zoning geometry (list of SpatialPolygons)
K	zoning object, as returned by the calNei function
colBreaks	if vector of length 1 number of color breaks, or else color breaks themselves
texMain	main title
boundary	map boundary (list with x and y values)
id	logical value (if TRUE display zone ids on plot)
valQ	quantile values to use for contour lines
palCol	color palette
noXY	if TRUE do not draw axes
iZ	index of zone to outline in red
mu	mu=1-only display zone number or id, mu=2-also display mean zone value
cex	text size
ptz	zone id location, if NULL automatically find the best locations

Details

plots a color image representation of values and zones

Value

an empty value

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
Z=K$zonePolygone
dispZ(mapTest$step,mapTest$krigGrid)
# not run
```

---

dispZmap *dispZmap*

Description

dispZmap

Usage

```
dispZmap(map, Z = NULL, qProb = NULL, valbp = NULL, scale = NULL,
  lev = 20, palCol = colorRampPalette(c("brown", "yellow")),
  legend.width = 1, parG = NULL, ptz = NULL)
```

Arguments

map	map object returned by genMap function
Z	zoning geometry (list of SpatialPolygons)
qProb	quantile associated probability vector
valbp	values used for boxplots
scale	field scale
lev	number of color levels
palCol	color palette
legend.width	relative width of legend
parG	graphics parameters (result of call to par)
ptz	zone id location, if NULL automatically find the best locations

Details

plots a color representation of values and zones

Value

an empty value

Examples

```
seed=80
data(mapTest)
ZK=initialZoning(c(0.5,0.7),mapTest)
K=ZK$resZ
Z=K$zonePolygone
#order zone ids by attribute mean value
ord=order(K$meanZone)
  Z=orderZ(Z,ord)
  plotZ(Z,id=TRUE)
# not run
```

distanceNormalisationSqrt *distanceNormalisationSqrt*

Description

distanceNormalisationSqrt

Usage

```
distanceNormalisationSqrt(matDistance)
```

Arguments

matDistance	distance matrix as returned by a call to calDistance

Details

normalize all MIJ terms of the distance matrix by dividing it by square root of diagonal terms MII*MJJ

Value

a normalized distance matrix

Examples

```
# load test map with simulated data
data(mapTest)
# load zoning results from test file
data(resZTest)
K=resZTest
resD = calDistance(typedist=1,mapTest$krigData,K$listZonePoint,K$zoneN,
       mapTest$krigSurfVoronoi,K$meanZone,pErr=0.9)
distanceNormalisationSqrt(resD$matDistanceCorr)
# not run
```

distanceNormalisationSum *distanceNormalisationSum*

Description

distanceNormalisationSum

Usage

```
distanceNormalisationSum(matDistance)
```

Arguments

matDistance	distance matrix as returned by a call to calDistance

Details

normalize all MIJ terms of the distance matrix by dividing it by sum of squared diagonal terms sum(MII^2+MJJ^2)

Value

a normalized distance matrix

Examples

```
# load test map with simulated data
data(mapTest)
# load zoning results from test file
data(resZTest)
K=resZTest
resD = calDistance(typedist=1,mapTest$krigData,K$listZonePoint,K$zoneN,
      mapTest$krigSurfVoronoi,K$meanZone,pErr=0.9)
distanceNormalisationSqrt(resD$matDistanceCorr)
# not run
```

extensionLine *extensionLine*

Description

extensionLine

Usage

```
extensionLine(contourL = NULL, step = NULL, bdSP, superLines)
```

Arguments

contourL	contour line
step	grid step as returned by calStep
bdSP	list, contains x and y coordinates of map boundaries
superLines	object returned by superLines(bdSP)

Details

closes contour lines by extending them to their interesection with the map border

Value

a list

Examples

```
data(mapTest)
step=mapTest$step
xsize=mapTest$xsize
ysize=mapTest$ysize
cL=contourLines(seq(step, xsize-step, by=step),seq(step, ysize-step, by=step),
            mapTest$krigGrid, levels = c(5,7))
plot(mapTest$boundary,type="l",col="red")
lines(cL[[1]])#contour line is not closed
lines(extensionLine(cL[[1]],step,sp::SpatialPoints(mapTest$boundary),
    superLines(mapTest$boundary)),col="red") #contour line is closed
# not run
```

---

extractionPoly *extractionPoly*

Description

extractionPoly

Usage

```
extractionPoly(polyTot)
```

Arguments

polyTot	SpatialPolygons

Details

extract all elements from SpatialPolygons, holes and full polygons are handled equally

Value

a list of SpatialPolygons

Examples

```
data(mapTest)
ZK=initialZoning(qProb=c(0.2,0.4,0.7),mapTest)
Z=ZK$resZ$zonePolygone
extractionPoly(Z[[5]]) # returns 2 SpatialPolygons
# not run
```

Extreme_Zone *Extreme_Zone*

Description

Extreme_Zone

Usage

```
Extreme_Zone(map, Z, numZ, label.is.min = TRUE)
```

Arguments

map	object returned by function genMap or genMapR
Z	list of zones.
numZ	number of the zone whose values will be approximated.
label.is.min	boolean value that is TRUE if the label of the zone is minimum and FALSE if the label is maximum

Details

funtion that approximates the value in a extreme zone (zone with label maximum or minimum, zones which have only one neighbour) by the solution of the Laplace's equation. The iso contours plotted on the approximate data will take the form of concentric circles as we supposed the extreme value of the zone is at the zone center (furthest point from the zone boundary.)

Value

approximated values of the values in zone (numZ).

Examples

```
seed=6
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2)
ZK=initialZoning(qProb=c(0.8),map)
Z=ZK$resZ$zonePolygone # list of zones
lab = ZK$resZ$lab # label of zones
plotM(map = map,Z = Z,lab = lab, byLab = FALSE)
# zone 2 is a zone with maximum label
numZ = 2
Estimation = Extreme_Zone(map = map, Z = Z, numZ = numZ, label.is.min = FALSE)
# compute the cost
cL = Cost_By_Laplace(map = map, Z = Z, numZ = numZ, Estimation = Estimation)
cM = Cost_By_Mean(map = map, Z = Z, numZ = numZ)
print(cL$cost_Laplace)
print(cM$cost_Mean)
# zone 2 is homogeneous
```

figCrit *figCrit*

Description

figCrit

Usage

```
figCrit(seed = 89, gr = 1, m1 = NULL, m2 = NULL, m3 = NULL,
  m4 = NULL, NEW = FALSE, ONE = FALSE, title = NULL)
```

Arguments

seed	xxxx
gr	xxxx
m1	xxxx
m2	xxxx
m3	xxxx
m4	xxxx
NEW	xxxx
ONE	xxxx
title	xxxx

Details

description, a paragraph

Value

a plot

Examples

```
# not run
```

---

figCritN *figCritN*

Description

figCritN

Usage

```
figCritN(m1 = NULL, m2 = NULL, m3 = NULL, m4 = NULL, m5 = NULL,
  NEW = FALSE, ONE = FALSE, title = "Gaussian field simulation",
  pdf = NULL)
```

Arguments

m1	dataset with loopQ1 results
m2	dataset with loopQ2 results
m3	dataset with loopQ3 results
m4	dataset with loopQ4 results
m5	dataset with loopQ5 results
NEW	new plot
ONE	single plot
title	plot title
pdf	pdf file name

Details

reads loopQ1-5 results, filters results by keeping th best criteria) and plots them together with corresponding costs.

Value

a vector of probabilities corresponding to best results

Examples

```
# not run
```

findCinZ *find contour for a given quantile value, within an envelope and englobing current zone*

Description

find contour for a given quantile value, within an envelope and englobing current zone

Usage

```
findCinZ(iC, Z, K, map, vRef, envel)
```

Arguments

iC	zone number
Z	zoning geometry (list of SpatialPolygons)
K	zoning object (such as returned by calNei function)
map	object returned by genMap function
vRef	quantile value
envel	SpatialPolygons within which the contour must be contained

Details

withing a zoning, find contour for a given vRef quantile value, contour contains current zone and is included in envel (spatial Polygon)

Value

a list with components

area area of SpatialPolygons corresponding to contour

contourSp SpatialPolygons corresponding to contour

Examples

```
data(mapTest)
qProb=c(0.3,0.5)
criti = correctionTree(qProb,mapTest)
best = criti$zk[[2]][[1]]
Z=best$zonePolygone
plotZ(Z)
iC=4
envel=calFrame(iC,Z,best$zoneNModif)
sp::plot(envel,col="blue",add=TRUE)
vRef=quantile(mapTest$krigGrid,0.6)
resp=findCinZ(iC,Z,best,mapTest,vRef,envel)
sp::plot(resp$contourSp,col="red",add=TRUE)
# not run
```

findN *findN*

Description

findN

Usage

```
findN(K, listN, iZ, minSize = 0.012)
```

Arguments

K	zoning object, as returned by the calNei function
listN	list of neighbor zones
iZ	index of current zone in zoning
minSize	minimum admissible zone size

Details

Find the neighbor zone into which to merge the current zone. It must be a neighbor in the sense of Voronoi polygons. In case of ties, choose the smallest zone for merging into

Value

the index of the zone into which to merge the current zone

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
Ns=getNs(K$zoneNModif,4) # neighbors of zone 4
listN = grep( TRUE , Ns) # zones 2 and 5
findN(K,listN,4) # zone 4 will be merged into zone 5
# not run
```

findNptInZone *findNptInZone*

Description

findNptInZone

Usage

```
findNptInZone(K, i1, i2, map)
```

Arguments

K	zoning object, as returned by the calNei function
i1	first zone
i2	second zone, where to search for neighbors of points in first zone
map	object returned by function genMap

Details

find, in a given zone, neighbor points of points belonging to another zone

Value

a two-column matrix, the first column contains indices of pts in first zone which have at least one neighbor in second zone, the second column contains the neighbor indices.

Examples

```
data(resZTest)
K=resZTest
# not run
```

findZCenter *findZCenter*

Description

findZCenter

Usage

```
findZCenter(Z, num = NULL)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
num	zone number

Details

find point within zone for pretty labelling

Value

a SpatialPoints

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.7),mapTest,SAVE=TRUE)
Z=criti$zk[[2]][[1]]$zonePolygone
findZCenter(Z)
# not run
```

findZCenterpt *findZCenterpt*

Description

findZCenterpt

Usage

```
findZCenterpt(data, K, num = NULL)
```

Arguments

data	SpatialPointsDataFrame
K	zoning object, as returned by the calNei function
num	zone number or NULL for all zones

Details

find point within zone for pretty labelling

Value

a matrix of x and y coordinates for chosen points with as many rows as zones

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.7),mapTest,SAVE=TRUE)
K=criti$zk[[2]][[1]]
data=mapTest$krigData
findZCenterpt(data,K)
# not run
```

genData *generate data*

Description

generate data

Usage

```
genData(DataObj = NULL, seed = 0, nPoints = 450, typeMod = "Gau",
  Vpsill = 5, Vrange = 0.2, Vmean = 8, Vnugget = 0, Vanis = 1,
  boundary = list(x = c(0, 0, 1, 1, 0), y = c(0, 1, 1, 0, 0)),
  manualBoundary = FALSE)
```

Arguments

DataObj	=NULL: simulated data with given seed or a data frame with real data
seed	numeric value used to generate simulated data
nPoints	number of generated raw data points
typeMod	type of variogram model (see vgm)
Vpsill	partial sill in variogram
Vrange	variogram range
Vmean	average data value
Vnugget	nugget in variogram
Vanis	anisotropy in variogram
boundary	list, contains x and y boundaries
manualBoundary	logical, if TRUE a manual boundary is drawn.

Details

description, a paragraph

Value

a list

tabData data frame of generated or real data with x,y,z values. x is standardized between 0 and 1, y is standardized with the same ratio used for x

boundary standardized boundary

VGMmodel VGM variogram model

modelGen RM transformed variogram model

ratio ratio used to normalize x data

Examples

```
resGene=genData(NULL,10,450,"Gau",5,0.2,8,0,list(x=c(0,0,1,1,0),y=c(0,1,1,0,0)),FALSE)
plot(resGene$tabData)

# not run
```

---

genEmptyGrid *generate grid from raw data*

Description

generate grid from raw data

Usage

```
genEmptyGrid(step, xsize, ysize)
```

Arguments

step	numeric step for grid
xsize	numeric value giving the data range on the x axis
ysize	numeric value giving the data range on the y axis

Value

a list that contains x and y kriged positions based on original ones,#' plus nx and ny (number of x and y positions).

Examples

```
genEmptyGrid(calStep(1000,1,1),1,1)
# not run
```

genMap *wrapper for randKmap, generate 2D map*

Description

wrapper for randKmap, generate 2D map

Usage

```
genMap(DataObj = NULL, seed = 80, krig = 2, Vpsill = 5, Vrange = 0.2,
  Vnugget = 0.2, Vmean = 8, nPointsK = 1000, boundary = list(x = c(0, 0,
  1, 1, 0), y = c(0, 1, 1, 0, 0)), disp = 0, FULL = FALSE)
```

Arguments

DataObj	=NULL: simulated data with seed or = a data frame with real data
seed	numeric,
krig	numeric, 1: kriging with vgm model, 2: inverse distance kriging
Vpsill	numeric parameter of the variogram model,
Vrange	numeric parameter of the variogram model,
Vnugget	numeric parameter of the variogram model,
Vmean	numeric parameter of the variogram model,
nPointsK	number of generated points after kriging
boundary	list, contains x and y coordinates of map boundaries
disp	numeric,
FULL	logical, if TRUE the returned list is complete

Details

wrapper for randKmap, generate 2D map with 1000 kriged data points, Gaussian field

Value

a map object as a list with components

tabAlea raw data, SpatialPointsDataFrame

surfaceVoronoi Voronoi polygon surfaces

krigTabAlea kriged data, SpatialPointsDataFrame

fitVarioAlea variogram

DataObj DataObj

ratio ratio used to normalize x data

Examples

```
m=genMap(seed=1,krig=2,disp=1) #generates a map and plots data
mean(m$krigGrid) # mean of generated kriged data
# not run
```

---

genQseq *genQseq*

Description

genQseq

Usage

```
genQseq(qProb, K, map, i1, i2, LEQ = 5, MAXP = 0.1, disp = 0)
```

Arguments

qProb	probability vector used to generate quantile values
K	zoning object, as returned by the calNei function
map	object returned by function genMap
i1	current zone index
i2	englobing zone index
LEQ	length of quantile sequence
MAXP	maximum shift from center for quantile sequence
disp	0: no info, 1: some info

Details

description, a paragraph

Value

a plot

Examples

```
qProb=c(0.4,0.7)
ZK=initialZoning(qProb,mapTest)
K=ZK$resZ
print(K$lab)
genQseq(qProb,K,mapTest,1,2) # from label 3 to label 2
# not run
```

getClosePt *getClosePt*

Description

getClosePt

Usage

```
getClosePt(Z, iC, iZC, disp = FALSE)
```

Arguments

Z	zoning (list of SpatialPolygons)
iC	current zone indes
iZC	close zone index
disp	information level (FALSE-no info)

Details

description, a paragraph

Value

a SpatialPoints of length 1

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
getClosePt(Z,1,3)
plotZ(Z)
points( getClosePt(Z,1,3),col="blue",pch=20)
# not run
```

getClosestZone *getClosestZone*

Description

getClosestZone

Usage

```
getClosestZone(iZ, Z, zoneN)
```

Arguments

iZ	current zone number
Z	current zone
zoneN	zone neighborhood Logical matrix

Details

get closest non neighbor zone (i.e. excluding neighbor zones and englobing zone)

Value

the closest zone index

Examples

```
data(resZTest)
Z=resZTest$zonePolygone
getClosestZone(4,Z,resZTest$zoneNModif)
# not run
```

getCoords *getCoords*

Description

getCoords

Usage

```
getCoords(sp, k = 1)
```

Arguments

sp	SpatialPolygons
k	polygon number

Details

description, a paragraph

Value

some coordinates

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
getCoords(Z[[1]])
# not run
```

getId *getId*

Description

getId

Usage

```
getId(Z, iZ)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
iZ	zone number

Details

get zone identifier in a zoning

Value

a character vector giving the zone identifier

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.5),mapTest,SAVE=TRUE)
Z=criti$zk[[2]][[1]]$zonePolygon
getId(Z,6)
# not run
```

getIds *getIds*

Description

getIds

Usage

```
getIds(Z, nums = NULL)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
nums	zone numbers

Details

get zone identifiers in a zoning

Value

a character vector giving the zone identifiers

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.5),mapTest,SAVE=TRUE)
Z=criti$zk[[2]][[1]]$zonePolygon
getIds(Z)
# not run
```

getNq *getNq*

Description

getNq

Usage

```
getNq(critList)
```

Arguments

critList	component critList of result from correctionTree

Details

determine size of quantile in result from correctionTree

Value

a vector with the size of quantile vectors for each zoning corresponding to critList

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.7),mapTest,SAVE=FALSE)
getNq(criti$critList)
# not run
```

getNs *getNs*

Description

getNs

Usage

```
getNs(zoneN, iZ)
```

Arguments

zoneN	zone neighborhood Logical matrix
iZ	index of current zone in zoning

Details

get zone numbers of neighbors of a given zone

Value

a Logical vector of current zone neighbors

Examples

```
data(mapTest)
K=resZTest
Ns=getNs(K$zoneNModif,5) # find neighbors of zone 5
```

---

getNumZone *getNumZone*

Description

getNumZone

Usage

```
getNumZone(ptsp, Z)
```

Arguments

ptsp	SpatialPointsDataFrame
Z	zoning geometry (list of SpatialPolygons)

Details

get zone numbers to which each point in a SpatialPointsDataFrame belongs

Value

the zone number

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
Z=K$zonePolygone
getNumZone(mapTest$krigData,Z)
# not run
```

getPoly *getPoly*

Description

getPoly

Usage

```
getPoly(Z, iZ, k)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
iZ	current zone index
k	polygon number within current zone

Details

get the kth polygon of the current zone in zoning Z

Value

a polygon (object of class Polygon)

Examples

```
ZK=initialZoning(qProb=c(0.4,0.2,0.7),mapTest)
Z=ZK$resZ$zonePolygone
P1=getPoly(Z,5,1)
P2=getPoly(Z,5,2) # second polygon is a hole
plot(P1@coords,type="l")
lines(P2@coords,type="l",col="blue")
# not run
```

getPolySp *getPolySp*

Description

getPolySp

Usage

```
getPolySp(sp, k = 1)
```

Arguments

sp	SpatialPolygons object
k	polygon number

Details

get the kth polygon of the current SpatialPolygons

Value

a polygon (object of class Polygon)

Examples

```
ZK=initialZoning(qProb=c(0.4,0.2,0.7),mapTest)
Z=ZK$resZ$zonePolygone
sp=Z[[5]]
P1=getPolySp(sp,1)
P2=getPolySp(sp,2) # second polygon is a hole
plot(P1@coords,type="l")
lines(P2@coords,type="l",col="blue")
# not run
```

getSurf *getSurf*

Description

getSurf

Usage

```
getSurf(Z, iZ)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
iZ	zone number

Details

description, a paragraph

Value

zone area

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
getSurf(Z,1)
# not run
```

---

getZoneId *getZoneId*

Description

getZoneId

Usage

```
getZoneId(zone)
```

Arguments

zone	SpatialPolygons

Details

get the zone unique identifier

Value

the zone identifier (a character vector of length 1)

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.5),mapTest,SAVE=TRUE)
Z=criti$zk[[2]][[1]]$zonePolygon
getZoneId(Z[[4]])
# not run
```

getZonePts *getZonePts*

Description

getZonePts

Usage

```
getZonePts(ptsp, zone)
```

Arguments

ptsp	SpatialPointsDataFrame with data values
zone	SpatialPolygons defining a zone

Details

get all data points within a zone

Value

a list with components

pts SpatialPointsDataFrame with the data points within the zone

mask Logical vector of the within zone data points indices

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
data(mapTest)
ptsp=mapTest$krigData
res=getZonePts(ptsp,Z[[5]])
plotZ(Z)
points(res$pts,col="blue",pch=20)
# not run
```

getZsize *getZsize*

Description

getZsize

Usage

```
getZsize(Z)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)

Details

compute maximum x and y values of zoning Z

Value

a vector with x and y maximum values

Examples

```
data(resZTest)
Z=resZTest$zonePolygone
getZsize(Z)
# not run
```

gridXY *generate empty grid*

Description

generate empty grid

Usage

```
gridXY(mat)
```

Arguments

mat	matrix with x and y coordinates in the first two columns, data in third column

Details

generate rectangular empty grid corresponding to x and y values in matrix

Value

a grid

Examples

```
data(dataReg)
gridXY(dataReg)
# not run
```

---

holeSp *holeSp*

Description

holeSp

Usage

```
holeSp(sp)
```

Arguments

sp	SpatialPolygons

Details

number of holes in SpatialPolygons

Value

the number of holes within sp

Examples

```
ZK=initialZoning(qProb=c(0.4,0.2,0.7),mapTest)
Z=ZK$resZ$zonePolygone
holeSp(Z[[5]]) #zone 5 has 1 hole
# not run
```

Identify *Identify*

Description

Identify

Usage

```
Identify(id, Z)
```

Arguments

id	zone identifier (character vector)
Z	zoning geometry (list of SpatialPolygons)

Details

get the number of a zone with a given identifier in a zoning this is necessary because correction procedures may remove zones from initial#' zoning. Therefore zone numbers change, but identifiers are conserved.

Value

the zone number

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.5),mapTest,SAVE=TRUE)
Z=criti$zk[[2]][[1]]$zonePolygon
Identify(6,Z)
# not run
```

initialZoning *initialZoning*

Description

initialZoning

Usage

```
initialZoning(qProb, map, pErr = 0.9, simplitol = 0.001, optiCrit = 2,
  disp = 0, GridData = F)
```

Arguments

qProb	probability vector used to generate quantile values
map	object returned by function genMap or genMapR
pErr	equality tolerance for distance calculations
simplitol	tolerance for spatial polygons geometry simplification
optiCrit	criterion choice
disp	0: no info, 1: some info, 2: detailed info
GridData	logical value indicating if data are already on a regular grid (no kriging in that case)

Details

description, a paragraph

Value

a list with components

resCrit criterion value

resDist list with components matDistance, matDistanceCorr and cost, such as returned by a call to calDistance

resZ list with components zoneN, zoneNModif, listZonePoint, meanTot, meanZone,listSurf, critSurf, zonePolygone, such as the object returned by calNei

Examples

```
data(mapTest)
ZK=initialZoning(qProb=c(0.4,0.7),mapTest)
plotZ(ZK$resZ$zonePolygone)
# not run
```

---

interCB *interCB*

Description

interCB

Usage

```
interCB(co, step, bd = list(x = c(0, 0, 1, 1, 0), y = c(0, 1, 1, 0, 0)),
  envel, disp = 0)
```

Arguments

co	contour line
step	map grid resolution
bd	map boundary
envel	envelope
disp	info level (0-no info, 1- add lines to plot)

Details

generates SpatialPolygons object corresponding to intersection of contour with boundary, must be within SpatialPolygons given in envel argument

Value

a SpatialPolygons

Examples

```
data(mapTest)
pG=polyToSp2(sp::Polygon(mapTest$boundary)) #SpatialPolygons corresponding to map boundary
cL=contourAuto(list(),mapTest$step,mapTest$xsize,mapTest$ysize,
   mapTest$krigGrid,c(5,7),mapTest$boundary)
ps = interCB(cL[[8]],mapTest$step,mapTest$boundary,pG)#envelope is the whole map
sp::plot(pG)
sp::plot(ps,col="red",add=TRUE)
# not run
```

interZoneC *interZoneC*

Description

interZoneC

Usage

```
interZoneC(Z, iC, iZC, closePt)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
iC	zone number
iZC	other zone number
closePt	SpatialPoints object in other zone used as circle center

Details

finds two intersection points of a circle with a zone. The circle radius is chosen so that it will intersect both zones given as arguments.

Value

a list with components

spi Two SpatialPoints to be used for the junction of the two zones

ord Order in which to use the points

Examples

```
data(mapTest)
qProb=c(0.2,0.5)
ZK = initialZoning(qProb, mapTest)
K=ZK$resZ
Z=K$zonePolygone
plotZ(K$zonePolygone) # zoning
closePt = getClosePt(Z,6,8)
points(closePt,col="red")
res  = interZoneC(Z,6,8,closePt)
points(res$spi,col="red")
# not run
```

---

labZone *labZone*

Description

labZone

Usage

```
labZone(K, qProb, dataF)
```

Arguments

K	zoning object, as returned by the calNei function
qProb	probability vector used to generate quantile values for Z
dataF	data used to generate labels and zoning

Details

assigns a class label (integer) to a zone depending on the zone mean value and on the quantile values (as in PA paper). Default label is 1, corresponding to me#' an value smaller or equal to first quantile. For p ordered quantile values, if mean #' value is greater than quantile k and smaller or equal to quantile k+1, zone label is#' k+1. if mean value is greater than quantile p, zone lable is p+1.

Value

a zoning object with labelled zones in lab component

Examples

```
data(mapTest)
dataF=mapTest$krigGrid
data(resZTest)
K=resZTest
p = K$qProb
labZone(K,p,dataF)
# not run
```

labZone0 *labZone0*

Description

labZone0

Usage

```
labZone0(K, qProb, dataF)
```

Arguments

K	zoning object, as returned by the calNei function
qProb	probability vector used to generate quantile values for Z
dataF	data used to generate labels and zoning

Details

assigns a class label (integer) to a zone depending on the zone mean value and on the quantile values. Default label is 1, corresponding to mean value samller #' or equal to first quantile. For k ordered quantile values, if mean value is greater #' than quantile k plus 10

Value

a zoning object with labelled zones in lab component

Examples

```
data(mapTest)
dataF=mapTest$krigGrid
data(resZTest)
K=resZTest
p = K$qProb
labZone(K,p,dataF)
# not run
```

lastPass *lastPass*

Description

lastPass

Usage

```
lastPass(map, qProb, listOfZ, crit, cost, costL, nz, mdist, pErr = 0.9,
  optiCrit = 2, minSize = 0.012, simplitol = 0.001, disp = 0)
```

Arguments

map	object returned by function genMap or genMapR
qProb	probability vector used to generate quantile values
listOfZ	list of zoning objects (such as returned by calNei function)
crit	criterion value list
cost	cost value list
costL	cost per lable value list
nz	number of zones list
mdist	distance matrix list
pErr	equality tolerance for distance calculations
optiCrit	criterion choice
minSize	zone area threshold under which a zone is too small to be manageable
simplitol	tolerance for spatial polygons geometry simplification
disp	0: no info, 1: detailed info

Details

description, a paragraph

Value

a list with components

listZ list of zoning objects (such as returned by calNei function)

crit criterion value list

cost cost value list

costL cost per label value list

nz number of zones list

mdist distance matrix list

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.7),mapTest,LASTPASS=FALSE)
Z=criti$zk[[1]][[1]]$zonePolygone #initial zoning
printZsurf(Z) # 8 zones with 2 small zones (7 and 8)
newRes=lastPass(mapTest,c(0.4,0.7),criti$zk[1],criti$criterion[1],
criti$cost[1],criti$costL[1],criti$nz[1],criti$mdist[1])
newZ=newRes$listOfZ[[1]][[1]]$zonePolygone
printZsurf(newZ) # 6 zones, 2 small zones were removed
# not run
```

linesC *linesC*

Description

linesC

Usage

```
linesC(listContour, col = "blue")
```

Arguments

listContour	list of contour lines
col	line color

Details

add contour Lines to plot

Value

an empty value

Examples

```
data(mapTest)
cL=list()
cL=contourAuto(cL,mapTest$step,mapTest$xsize,mapTest$ysize,mapTest$krigGrid,c(5,7),mapTest$boundary)
plot(mapTest$boundary)
linesC(cL,col="black")
# not run
```

linesSp *linesSp*

Description

linesSp

Usage

```
linesSp(sp, k = 1, lty = 1, col = "red", lwd = 1)
```

Arguments

sp	SpatialPolygons object
k	polygon number
lty	line type
col	color
lwd	line width

Details

description, a paragraph

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
plotZ(Z)
linesSp(Z[[4]])
# not run
```

lineToSp *lineToSp*

Description

lineToSp

Usage

```
lineToSp(lin)
```

Arguments

lin	list with x and y line coordinates

Details

transform closed line into SpatialPolygons

Value

a SpatialPolygons

Examples

```
data(mapTest)
cL=list()
cL=contourAuto(cL,mapTest$step,mapTest$xsize,mapTest$ysize,mapTest$krigGrid,c(5,7),mapTest$boundary)
lin=data.frame(x=cL[[8]]$x,y=cL[[8]]$y)
sp=lineToSp(lin)
# not run
```

listContourArea *listContourArea*

Description

listContourArea

Usage

```
listContourArea(cL)
```

Arguments

cL	list of contour lines

Details

area of all contour lines in list

Value

a list of areas

Examples

```
data(mapTest)
cL=list()
cL=contourAuto(cL,mapTest$step,mapTest$xsize,mapTest$ysize,mapTest$krigGrid,c(5,7),mapTest$boundary)
listContourArea(cL)
# not run
```

list_Zone_2_Neighbours *list_Zone_2_Neighbours*

Description

list_Zone_2_Neighbours

Usage

```
list_Zone_2_Neighbours(Z, lab)
```

Arguments

Z	list of Zones
lab	vector labels of zones

Details

Returns the numbers of zones that have exactly 2 neighbours with different labels. These zone are susceptible to be transitions zones

Value

a vector containing zone numbers

Examples

```
seed=6
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2)
ZK=initialZoning(qProb=c(0.67,0.8),map)
Z=ZK$resZ$zonePolygone # list of zones
lab = ZK$resZ$lab # label of zones
plotM(map = map,Z = Z,lab = lab, byLab = FALSE)
# zone 4 and 6 are transition zones and have exactly 2 neighbours with different labels.
list_Zone_2_Neighbours(Z = Z, lab = lab)
```

loopQ1 *loopQ1*

Description

loopQ1

Usage

```
loopQ1(map, disp = 1, step = 0.075, minSize = 0.012, minSizeNG = 0.001,
  QUIET = FALSE)
```

Arguments

map	object returned by function genMa
disp	0: no info, 1: some info, 2: detailed info
step	loop increment
minSize	zone area threshold under which a zone is too small to be manageable
minSizeNG	zone area threshold under which a zone will be removed
QUIET	run in silence-no display

Details

exploratory loop on probability values associated to quantiles. Performs map zonings for each value of the 1 quantile loop (yielding a 42-label zoning).

Value

a matrix with 6 columns and as many rows as loop elements. Columns contain the following values calculated for each quantile vector: criterion, cost, cost per label, number of zones, quantile associated probability values and number of non degenerated quantiles.

Examples

```
# not run
```

loopQ2 *loopQ2*

Description

loopQ2

Usage

```
loopQ2(map, disp = 1, step = 0.075, minSize = 0.012, minSizeNG = 0.001,
  QUIET = FALSE)
```

Arguments

map	object returned by function genMa
disp	0: no info, 1: some info, 2: detailed info
step	loop increment
minSize	zone area threshold under which a zone is too small to be manageable
minSizeNG	zone area threshold under which a zone will be removed
QUIET	run in silence-no display

Details

exploratory loop on probability values associated to quantiles. Performs map zonings for each value of the 1 quantile loop (yielding a 42-label zoning).

Value

a matrix with 7 columns and as many rows as loop elements. Columns contain the following values calculated for each quantile vector: criterion, cost, cost per label, number of zones, quantile associated probability values and number of non degenerated quantiles.

Examples

```
# not run
```

loopQ3 *loopQ3*

Description

loopQ3

Usage

```
loopQ3(map, disp = 1, step = 0.075, minSize = 0.012, minSizeNG = 0.001,
  QUIET = F)
```

Arguments

map	object returned by function genMa
disp	0: no info, 1: some info, 2: detailed info
step	loop increment
minSize	zone area threshold under which a zone is too small to be manageable
minSizeNG	zone area threshold under which a zone will be removed
QUIET	run in silence-no display

Details

exploratory loop on probability values associated to quantiles. Performs map zonings for each value of the 3 quantile loop (yielding a 4-label zoning).

Value

a matrix with 8 columns and as many rows as loop elements. Columns contain the following values calculated for each quantile vector: criterion, cost, cost per label, number of zones, quantile associated probability values and number of non degenerated quantiles.

Examples

```
seed=10
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=1)
# not run
loopQ3(map,step=0.1,disp=0,QUIET=TRUE)
```

---

loopQ4 *loopQ4*

Description

loopQ4

Usage

```
loopQ4(map, disp = 1, step = 0.075, minSize = 0.012, minSizeNG = 0.001,
  QUIET = F)
```

Arguments

map	object returned by function genMa
disp	0: no info, 1: some info, 2: detailed info
step	loop increment
minSize	zone area threshold under which a zone is too small to be manageable
minSizeNG	zone area threshold under which a zone will be removed
QUIET	run in silence-no display

Details

exploratory loop on probability values associated to quantiles. Performs map zonings for each value of the 1 quantile loop (yielding a 42-label zoning).

Value

a matrix with 9 columns and as many rows as loop elements. Columns contain the following values calculated for each quantile vector: criterion, cost, cost per label, number of zones, quantile associated probability values and number of non degenerated quantiles.

Examples

```
# not run
```

---

loopQ5 *loopQ5*

Description

loopQ5

Usage

```
loopQ5(map, disp = 1, step = 0.075, minSize = 0.012, minSizeNG = 0.001,
  QUIET = F)
```

Arguments

map	object returned by function genMa
disp	0: no info, 1: some info, 2: detailed info
step	loop increment
minSize	zone area threshold under which a zone is too small to be manageable
minSizeNG	zone area threshold under which a zone will be removed
QUIET	run in silence-no display

Details

exploratory loop on probability values associated to quantiles. Performs map zonings for each value of the 1 quantile loop (yielding a 42-label zoning).

Value

a matrix with 9 columns and as many rows as loop elements. Columns contain the following values calculated for each quantile vector: criterion, cost, cost per label, number of zones, quantile associated probability values and number of non degenerated quantiles.

Examples

```
# not run
```

mapTest *a map*

Description

A map object for zoning, result from the genMap function

Usage

```
mapTest
```

Format

a list of SpatialPolygons

maxDistSP *maxDistSP*

Description

maxDistSP

Usage

```
maxDistSP(sp)
```

Arguments

sp	SpatialPolygons

Details

maximum distance within kth polygon of current zone

Value

the maximum distance within sp

Examples

```
ZK=initialZoning(qProb=c(0.4,0.2,0.7),mapTest)
Z=ZK$resZ$zonePolygone
maxDistSP(Z[[5]])
# not run
```

maxDistZone *maxDistZone*

Description

maxDistZone

Usage

```
maxDistZone(Z, iZ, k)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
iZ	current zone index
k	polygon number within current zone

Details

maximum distance within kth polygon of current zone

Value

the maximum distance within kth polygon of the current zone

Examples

```
ZK=initialZoning(qProb=c(0.4,0.2,0.7),mapTest)
Z=ZK$resZ$zonePolygone
maxDistZone(Z,5,1)
# not run
```

maxId *maxId*

Description

maxId

Usage

```
maxId(Z)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)

Details

get highest number corresponding to a zone identifier in a zoning

Value

a number

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.5),mapTest,SAVE=TRUE)
Z=criti$zk[[2]][[1]]$zonePolygon
maxId(Z)
# not run
```

meanL *meanL*

Description

meanL

Usage

```
meanL(zlab, listZonePoint, tabVal, surfVoronoi)
```

Arguments

zlab	list with zone numbers for each zone label
listZonePoint	list of indices of data points within zones, result of call to calNei
tabVal	SpatialPointsDataFrame containing data values
surfVoronoi	Surfaces of the Voronoi polygons corresponding to data pts

Details

compute overall mean of all zones for each label

Value

a list with components

mL vector of weighted (with Voronoi surfaces) per label average values

SL vector of per label Voronoi surfaces

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.7),mapTest,SAVE=TRUE)
K=criti$zk[[2]][[1]]
uni=unique(K$lab)
zlab=sapply(uni,function(x){(1:length(K$lab))[K$lab==x]})
resL=meanL(zlab,K$listZonePoint,mapTest$krigData,mapTest$krigSurfVoronoi)
# not run
```

---

meansdSimu *meansdSimu*

Description

meansdSimu

Usage

```
meansdSimu(vseed = NULL, krig = 2)
```

Arguments

vseed	list of simulation seeds
krig	type of kriging (1-variogram model-based, 2-inverse distance-based)

Details

computes mean and standard deviation of a set of map simulations

Value

a matrix with as many rows as simulations, and 4 columns, the first two columns give mean and standard deviation of generated raw data, the last two columns give mean and standard deviation of kriged data

Examples

```
meansdSimu(c(1,2))
# not run
```

meanvarSimu *meanVarsimu*

Description

meanVarsimu

Usage

```
meanvarSimu(map)
```

Arguments

map	object generated by genMap

Details

computes mean and standard deviation of a set of map simulations

Value

a vector with 4 elements, the first two give mean and standard deviation of generated raw data, the last two give mean and standard deviation of kriged data

Examples

```
meansdSimu(c(1,2))
# not run
```

MeanVarWPts *MeanVarWPts*

Description

MeanVarWPts

Usage

```
MeanVarWPts(map, zone, w = NULL)
```

Arguments

map	object returned by function genMap
zone	SpatialPolygons defining a zone
w	weighting vector (default NULL)

Details

computes (weighted) mean and variance of zone data

Value

a list with components

mean (weighted) mean of the within zone attribute value

var (weighted) variance of the within zone attribute value

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
data(mapTest)
MeanVarWPts(mapTest,Z[[1]])
# Weights are areas of the Voronoi polygons corresponding to data points
MeanVarWPts(mapTest,Z[[1]],mapTest$krigSurfVoronoi) #slightly different result
# not run
```

modlm *modlm*

Description

modlm

Usage

```
modlm(ptsp, Z)
```

Arguments

ptsp	SpatialPointsDataFrame with data values
Z	zoning (list of SpatialPolygons)

Details

description, a paragraph

Value

the result of a call to lm (anova model with zone number as factor)

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
data(mapTest)
ptsp=mapTest$krigData
modlm(ptsp,Z)
# not run
```

---

moveHoles *moveHoles*

Description

moveHoles

Usage

```
moveHoles(zoneMain, zoneSuppr)
```

Arguments

zoneMain	SpatialPolygons
zoneSuppr	SpatialPolygons inside main

Details

creates SpatialPolygons excluding holes

Value

a new SpatialPolygons object

Examples

```
# not run
```

new_krigGrid_for_visualisation *new_krigGrid_for_visualisation*

Description

new_krigGrid_for_visualisation

Usage

```
new_krigGrid_for_visualisation(map, Z, numZ, solution)
```

Arguments

map	object returned by function genMap or genMapR
Z	list of zones.
numZ	number of the zone whose values will be approximated.
solution	the result of function "Transition_Zone_Near_Boundary" or "Transition_Zone_Far_Boundary" or "Extreme_Zone"

Details

Elementary function that create a new krigGrid by replacing the real values by the approximation of the function "Transition_Zone_Near_Boundary","Transition_Zone_Far_Boundary" or "Extreme_Zone" in order to have a look at the new iso contour

Value

new krigGrid and new data importFrom sp plot

Examples

```
seed=2
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2)
ZK=initialZoning(qProb=c(0.55,0.85),map)
Z=ZK$resZ$zonePolygone # list of zones
lab = ZK$resZ$lab # label of zones
plotM(map = map,Z = Z,lab = lab, byLab = FALSE)
# zone 6 is a transition zone that has commun boundary with the map
numZ = 6
Estimation = Transition_Zone_Near_Boundary(map = map, Z = Z, numZ = numZ)
result = new_krigGrid_for_visualisation(map = map, Z = Z, numZ = numZ, solution = Estimation)
new_krigGrid = result$new_krigGrid
new_data = result$new_data
quant1 = quantile(map$krigData@data$var1.pred,probs = 0.55)
quant2 = quantile(map$krigData@data$var1.pred,probs = 0.85)
# plot initial isocontours
plotM(map = map,Z = Z,lab = lab, byLab = TRUE)
```

```
listContours = contourBetween(map = map, krigGrid = map$krigGrid, q1 = quant1, q2 = quant2)
for (i in 1:length(listContours)){
  sp::plot(listContours[[i]]$contour,add=TRUE,col = "red")
}
# plot modified isocontours
plotM(map = map,Z = Z,lab = lab, byLab = TRUE)
listContours = contourBetween(map = map, krigGrid = new_krigGrid, q1 = quant1, q2 = quant2)
for (i in 1:length(listContours)){
  sp::plot(listContours[[i]]$contour,add=TRUE,col = "red")
}
```

normDistMat *normDistMat*

Description

normDistMat

Usage

```
normDistMat(matDistanceCorr, optiCrit)
```

Arguments

matDistanceCorr	corrected distance matrix using pErr, result of calDistance
optiCrit	criterion choice

Details

normalize distance matrix so that diagonal is equal to 1

Value

a normalized distance matrix

Examples

```
# load test map with simulated data
data(mapTest)
# load zoning results from test file
data(resZTest)
K=resZTest
resD = calDistance(typedist=1,mapTest$krigData,K$listZonePoint,K$zoneN,
        mapTest$krigSurfVoronoi,K$meanZone,pErr=0.9)
normDistMat(resD$matDistanceCorr,2)
# not run
```

normSize *normSize*

Description

normSize

Usage

```
normSize(boundaryN, minSize, minSizeNG)
```

Arguments

boundaryN	normalized map boundary
minSize	minimum size threshold
minSizeNG	no grow size threshold

Details

normalize thresholds for small zone detection and no grow zone, considering mapo boundary

Value

a list with components

minSize normalized minimum size threshold

minSizeNG normalized no grow size threshold

Examples

```
data(mapTest)
resT=normSize(mapTest$boundary,0.012,0.001)#normalize thresholds relatively to map boundary area
# not run
```

normZcoords *normZcoords*

Description

normZcoords

Usage

```
normZcoords(Z, boundary)
```

Arguments

Z	list of SpatialPolygons
boundary	list with components x and y, used to normalize polygons in zoning

Details

description, a paragraph

Value

a list with components

Zn list of normalized SpatialPolygons

boundaryn normalized boundary

Examples

```
shape1 = geozoning::shape1
p = shape1@polygons
P=sp::SpatialPolygons(p) #SpatialPolygons
Z1=list()
for (kk in 1:length(P)){Z1[[kk]]=P[kk]} # transform into list of SpatialPolygons
bd=list(x=c(7723131,7723132,7723294,7723295,7723131),y=c(3576432,3576814,3576809,3576436,3576432))
# not run
```

---

nPolySp *nPolySp*

Description

nPolySp

Usage

```
nPolySp(sp)
```

Arguments

sp	SpatialPolygons

Details

number of polygons in SpatialPolygons

not run

Value

the number of polygons within the current zone

Examples

```
ZK=initialZoning(qProb=c(0.4,0.2,0.7),mapTest)
Z=ZK$resZ$zonePolygone
print(paste(nPolySp(Z[[2]]),"polygons"))
```

---

nPolyZone *nPolyZone*

Description

nPolyZone

Usage

```
nPolyZone(Z, iC)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
iC	current zone number within Z

Details

number of polygons in current zone

Value

the number of polygons within the current zone

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
iC=1;print(paste(nPolyZone(Z,iC),"polygons in zone",iC))
# not run
```

optiGrow *optiGrow*

Description

optiGrow

Usage

```
optiGrow(K, iC, qProb, refPoint, map, optiCrit = 2, minSize = 0.012,
  minSizeNG = 0.001, distIsoZ = 0.075, LEQ = 5, MAXP = 0.1,
  simplitol = 1e-12, disp = 0)
```

Arguments

K	zoning object (such as returned by calNei function)
iC	index of zone to grow
qProb	probability vector used to generate quantile values
refPoint	xxxx
map	object returned by genMap function
optiCrit	criterion choice
minSize	zone area threshold under which a zone is too small to be manageable
minSizeNG	zone area threshold under which a zone will be removed
distIsoZ	threshold distance to next zone, above which a zone is considered to be isolated
LEQ	length of quantile sequence used to grow zone
MAXP	quantile sequence maximum shift
simplitol	tolerance for spatial polygons geometry simplification
disp	0: no info, 1: detailed info

Details

grow an isolated zone by finding a bigger contour line

Value

a list with components

crit criterion value of the new zoning

area area of the grown zone

Zopti new zoning geometry (list of SpatialPolygons)

qM quantile corresponding to new zone

Examples

```
data(mapTest)
qProb=c(0.3,0.5)
criti = correctionTree(qProb,mapTest)
best = criti$zk[[2]][[1]]
Z=best$zonePolygone
plotZ(Z)
refPoint = rgeos::gCentroid(Z[[4]])
sp::plot(refPoint,add=TRUE,col="blue",pch=21)
zg=optiGrow(best,4,qProb,refPoint,mapTest) #grow zone 4
id=as.numeric(getZoneId(Z[[4]]))
linesSp(zg$Zopti[[id]],col="blue") # new zoning with grown zone 4
# not run
```

optiRG *optiRG*

Description

optiRG

Usage

```
optiRG(K, map, iC, iZC, simplitol = 0.001, disp = 0)
```

Arguments

K	zoning object (such as returned by calNei function)
map	object returned by function genMap or genMapR
iC	first zone
iZC	second zone
simplitol	tolerance for spatial polygons geometry simplification
disp	0: no info, 1: detailed info

Details

join two zones close to each other

Value

a zoning object

Examples

```
data(mapTest)
qProb=c(0.2,0.5)
ZK = initialZoning(qProb, mapTest)
K=ZK$resZ
Z=K$zonePolygone
plotZ(K$zonePolygone) # zoning
kmi=optiRG(K,mapTest,6,7,disp=1)
#zones 6 and 7 are joined into new zone 6
sp::plot(kmi$zonePolygone[[6]],col="red",add=TRUE)
# not run
```

orderZ *orderZ*

Description

orderZ

Usage

```
orderZ(Z, ord)
```

Arguments

Z	zoning geometry
ord	sorting order

Details

sorts zones according to ord vector

Value

a zoning geometry (list of SpatialPolygons)

Examples

```
map=genMap(DataObj=NULL,seed=40,disp=FALSE,krig=1,Vnugget=1.2)
qProb=c(0.275,0.8)
criti=correctionTree(qProb,map,LASTPASS=FALSE)
res=searchNODcrit1(qProb,criti)
b=res$ind[[1]][1]
K=criti$zk[[2]][[b]]
Z=K$zonePolygone
plotZ(Z)
ord=valZ(map,K)$ord
Z=orderZ(Z,ord)
plotZ(Z)
# not run
```

plotListC *plotListC*

Description

plotListC

Usage

```
plotListC(cL, col = "red")
```

Arguments

cL	list of contour lines
col	color to use

Details

add contour lines to a plot

Value

a plot

Examples

```
data(mapTest)
cL=list()
cL=contourAuto(cL,mapTest$step,mapTest$xsize,mapTest$ysize,mapTest$krigGrid,c(5,7),mapTest$boundary)
plot(mapTest$boundary,type="l")
plotListC(cL)
# not run
```

plotM *plotM*

Description

plotM

Usage

```
plotM(map, Z = NULL, lab = NULL, byLab = TRUE, quantile = NULL,
  crit = NULL, cost = NULL, bestCrit = NULL, bestCost = NULL,
  newCost = NULL, line = 0, cex = 2)
```

Arguments

map object returned by function genMap or genMapR.

Z list of zones, each zone is a SpatialPolygons.

lab label of each zones.

byLab boolean, if TRUE display the label of each zone, else display the zone number.

quantile probability vector used to generate "Z". This will be displayed in the title of the plot.

crit criterion value corresponding to "Z. This will be displayed in the title of the plot.

cost cost value corresponding to "Z". This will be displayed in the title of the plot.

bestCrit best criterion value. This will be displayed in the title of the plot.

bestCost best cost value. This will be displayed in the title of the plot.

newCost new cost value. This will be displayed in the title of the plot.

line position of the title. if negative, the title goes down, otherwise, goes up.

cex text size

Details

plot the map in color with zones and details.

Value

an empty value

Examples

```
seed=2
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2)
ZK=initialZoning(qProb=c(0.55,0.85),map)
Z=ZK$resZ$zonePolygone # list of zones
lab = ZK$resZ$lab # label of zones
plotM(map = map,Z = Z,lab = lab, byLab = FALSE)
```

---

plotMap *plot a map*

Description

plot a map

Usage

```
plotMap(map)
```

Arguments

map	a map object, such as returned by genMap

Details

plot 3 different graphics of a map object

Value

a plot

Examples

```
m=genMap(seed=1,krig=2,disp=0)
plotMap(m)
# not run
```

---

plotSp *plotSp*

Description

plotSp

Usage

```
plotSp(sp, k = 1, xlim, ylim)
```

Arguments

sp	SpatialPolygons object
k	polygon number
xlim	x range
ylim	y range

Details

description, a paragraph

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
plotSp(Z[[1]],xlim=c(0,1),ylim=c(0,1))
# not run
```

plotVario *plotVario*

Description

plotVario

Usage

```
plotVario(map, ylim = NULL)
```

Arguments

map	object returned by function genMap
ylim	range of y axis

Details

plot empirical variogram for model and data in map (raw data plus kriged data)

Value

a plot

Examples

```
data(mapTest)
plotVario(mapTest)
# not run
```

plotZ *plotZ*

Description

plotZ

Usage

```
plotZ(Z, map = NULL, id = FALSE, noXY = FALSE,
  palCol = colorRampPalette(topo.colors(20)))
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
map	map object returned by genMap function
id	logical value, if TRUE display zone ids, if FALSE display zone numbers
noXY	logical value, if TRUE do not display x and y axes
palCol	argument of colorRampPalette

Details

wrapper function for dispZ

Value

an empty value

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
Z=K$zonePolygone
plotZ(Z,mapTest)
# not run
```

pointsSp *pointsSp*

Description

pointsSp

Usage

```
pointsSp(sp, k = 1, col = "red")
```

Arguments

sp	SpatialPolygons object
k	polygon number
col	color

Details

description, a paragraph

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
plotZ(Z)
pointsSp(Z[[1]])
# not run
```

Points_Near_Boundary *Points_Near_Boundary*

Description

Points_Near_Boundary

Usage

```
Points_Near_Boundary(map)
```

Arguments

map	object returned by function genMap or genMapR

Details

function that returns a list of points in a zone that are near boundary of the map

Examples

```
map = mapTest
Points_Near_Boundary(map = map)
```

polyToSp *polyToSp*

Description

polyToSp

Usage

```
polyToSp(Z, iZ, k)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
iZ	zone number
k	polygon number

Details

transforms kth polygon of zone into SpatialPolygons

Value

a SpatialPolygons

Examples

```
ZK=initialZoning(qProb=c(0.4,0.2,0.7),mapTest)
Z=ZK$resZ$zonePolygone
sph=polyToSp(Z,5,2)
plotZ(Z)
sp::plot(sph,type="l",col="blue",add=TRUE)
# not run
```

polyToSp2 *polyToSp2*

Description

polyToSp2

Usage

```
polyToSp2(p)
```

Arguments

p	polygon

Details

transforms polygon into SpatialPolygons

Value

a SpatialPolygons

Examples

```
ZK=initialZoning(qProb=c(0.4,0.2,0.7),mapTest)
Z=ZK$resZ$zonePolygone
sp=Z[[5]]
P1=getPolySp(sp,1)
sph=polyToSp2(P1)
plotZ(Z)
sp::plot(sph,col="blue",lwd=2,add=TRUE)
# not run
```

printInterZ *printInterZ*

Description

printInterZ

Usage

```
printInterZ(Z, sp)
```

Arguments

Z	list of zones, each zone is a SpatialPolygons
sp	SpatialPolygons object

Details

checks intersection of sp and each element of Z

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
Z=K$zonePolygone
plotZ(Z,mapTest)
printInterZ(Z,Z[[1]])
# not run
```

printLabZ *printLabZ*

Description

printLabZ

Usage

```
printLabZ(Klist)
```

Arguments

Klist	list of zoning objects, typically result of a call to correctionTree

Details

print zoning labels for a list of zoning objects

Value

a list of zoning objects

Examples

```
data(mapTest)
qProb=c(0.1,0.2,0.4);criti=correctionTree(qProb,mapTest) # 2 zonings at last level
printLabZ(criti$zk[[2]])
# not run
```

printZid *printZid*

Description

printZid

Usage

```
printZid(Z)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)

Details

print zone identifiers in a zoning

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
printZid(Z)
# not run
```

printZsurf *printZsurf*

Description

printZsurf

Usage

```
printZsurf(Z, minSize = 0.012)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
minSize	minimum size threshold

Details

print zone surfaces

Value

a vector of small zone indices

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
printZsurf(Z,0.03)
# not run
```

ptInZone *ptInZone*

Description

ptInZone

Usage

```
ptInZone(zone, pts, numpt)
```

Arguments

zone	SpatialPolygons
pts	data points
numpt	data point indices

Details

description, a paragraph

Value

1 if point is within zone, 0 if not

Examples

```
data(mapTest)
data(resZTest)
Z=resZTest$zonePolygone
ptInZone(Z[[1]],mapTest$krigData,c(5,500))
# not run
```

ptNei *returns list of point neigbors for each point*

Description

returns list of point neigbors for each point

Usage

```
ptNei(neighBool)
```

Arguments

neighBool numeric, boolean neighborhood matrix for pts

Value

a list of pt neigbors for each pt

Examples

```
data(mapTest) # simulated data
grid=genEmptyGrid(calStep(2000,1,1),1,1)
nbP= grid$nx*grid$ny
neighBool=matrix(logical(nbP^2),nbP,nbP)
resVoronoi=voronoiPolygons(mapTest$krigData,c(0,1,0,1),neighBool)
neighBool=resVoronoi$neighBool
listeNpt=ptNei(neighBool)
# not run
```

---

ptsInSp *ptInSp*

Description

ptInSp

Usage

```
ptsInSp(sp, pts, hole = FALSE)
```

Arguments

sp SpatialPolygons

pts data points

hole if TRUE also consider points in holes

Details

finds data points in sp

Value

a data frame with data points within sp

Examples

```
data(mapTest)
data(resZTest)
Z=resZTest$zonePolygone
ptsInSp(Z[[5]],mapTest$krigData) # 5 data points within zone 5

# not run
```

---

r2 *r2*

Description

r2

Usage

```
r2(reslm)
```

Arguments

reslm	result of a call to lm

Details

adjusted R2

Value

the adjusted r-square of the lm model

Examples

```
# not run
```

---

randKmap *randKmap: Generate data for zoning or prepare real data*

Description

randKmap: Generate data for zoning or prepare real data

Usage

```
randKmap(DataObj, seed = NULL, nPoints = 450, nPointsK = 2000,
  nSimuCond = 0, typeMod = "Gau", Vpsill = 5, Vrange = 0.2, Vmean = 8,
  Vnugget = 0, boundary = list(x = c(0, 0, 1, 1, 0), y = c(0, 1, 1, 0, 0)),
  manualBoundary = FALSE, krig = 2, disp = 0, FULL = FALSE)
```

Arguments

DataObj	=NULL: simulated data with seed or = a data frame with real data
seed	numeric, seed
nPoints	numeric, number of points, default 450
nPointsK	numeric, default 2000
nSimuCond	numeric
typeMod	character, model type
Vpsill	numeric, default 5
Vrange	numeric, default 0.2
Vmean	numeric, default 8
Vnugget	numeric, default 0
boundary	list contains x and y
manualBoundary	logical, default FALSE
krig	numeric
disp	numeric
FULL	logical, if TRUE the returned list is complete

Details

generates a map object from simulated data or real data

Value

a list

rawData simulated or real raw data within the boundary

step grid step

krigData kriged data

krigGrid kriged data in form of grid

krigN kriged neighbours of each data point

krigSurfVoronoi areas of Voronoi polygons in the tesselation of kriged data

modelGen random fields model

VGMmodel vgm model

boundary (x,y) list of boundary points

ratio ratio used to normalize x data

Examples

```
# not run
map = randKmap(NULL,nPointsK=500,Vmean=15,krig=2)
mean(map$krigGrid) # mean of generated kriged data
plotMap(map)
```

randKmapGrid *randKmapGrid*

Description

randKmapGrid

Usage

```
randKmapGrid(DataObj, nSimuCond = 0, boundary = list(x = c(0, 0, 1, 1, 0), y
  = c(0, 1, 1, 0, 0)), manualBoundary = FALSE, disp = 0, FULL = FALSE)
```

Arguments

DataObj	=NULL: simulated data with seed or = a data frame with real data
nSimuCond	numeric
boundary	list contains x and y
manualBoundary	logical, default FALSE
disp	numeric
FULL	logical, if TRUE the returned list is complete

Details

Prepare real data for zoning, data are already on a regular grid, hence no kriging is done.

Value

a list

rawData simulated or real raw data within the boundary

step grid step

krigData kriged data

krigGrid kriged data in form of grid

krigN kriged neighbours of each data point

krigSurfVoronoi areas of Voronoi polygons in the tesselation of kriged data

modelGen random fields model

VGMmodel vgm model

boundary (x,y) list of boundary points

Examples

```
data(dataReg) #regular data on a square grid between 0 and 1
map = randKmapGrid(dataReg)
plotMap(map)
# not run
```

readS *readS returns coords, ranges for x and y of a shapefile*

Description

readS returns coords, ranges for x and y of a shapefile

Usage

```
readS(file, dir)
```

Arguments

file	file name
dir	directory

Details

reads a polygon shp file in a directory and extracts coordinates and x and y ranges.

Value

a list with components SpatialPolygonsDataFrame, ranges for x and y

Examples

```
#z=readS("Field_8_zones.shp",dir="../data/")
#plot(z$sp)
# not run
```

remove1FromZ *remove1FromZ*

Description

remove1FromZ

Usage

```
remove1FromZ(Z, iC, zoneN, simplitol = 0.001, disp = 0)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
iC	current zone index
zoneN	zone neighborhood Logical matrix
simplitol	tolerance for spatial polygons geometry simplification
disp	0: no info, 1: some info

Details

description, a paragraph

Value

a new zoning where current zone has been removed

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
plotZ(Z)
plotZ(remove1FromZ(Z,2,K$zoneN))
# not run
```

removeFromZ *removeFromZ*

Description

removeFromZ

Usage

```
removeFromZ(Z, zoneN, ptN, listZonePoint, spdata, simplitol = 0.001, n = 1)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
zoneN	zone neighborhood Logical matrix
ptN	indices of data pts neighbours
listZonePoint	list of indices of data points within zones, result of call to calNei
spdata	spatial data
simplitol	tolerance for spatial polygons geometry simplification
n	minimal number of points below which a zone is removed from zoning

Details

description, a paragraph

Value

a list with components

Z new zoning geometry (list of SpatialPolygons)where zones with less than n points were removed

zoneN new zone neighborhood Logical matrix

listZonePoint new list of indices of data points within zones

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
plotZ(Z)
# remove from Z all zones with less than 10 data points
Z2=removeFromZ(Z,K$zoneN,K$krigN,K$listZonePoint,mapTest$krigData,n=10)
printZid(Z2$Z)
# not run
```

resZTest *A list of initialZoning results*

Description

A list of initialZoning results

Usage

```
resZTest
```

Format

a list with all results from initialZoning function: criterion value, list with components matDistance, matDistanceCorr and cost, such as returned by a call to calDistance, a list with components zoneN, zoneNModif, listZonePoint, meanTot, meanZone,listSurf, critSurf, zonePolygone, such as the object returned by calNei.

saveZK *saveZK function called by correctionTree*

Description

saveZK function called by correctionTree

Usage

```
saveZK(map, K1, Z2, qProb, listOfZ, indCur, crit, cost, costL, nz, mdist,
  pErr = 0.9, optiCrit = 2, simplitol = 0.001)
```

Arguments

map	object returned by function genMap or genMapR
K1	previous zoning
Z2	current zoning geometry (list of SpatialPolygons)
qProb	probability vector used to generate quantile values
listOfZ	list of zoning objects
indCur	index of new list element
crit	list of criteria
cost	list of costs
costL	list of per label costs
nz	list of number of zones
mdist	list of distance matrices
pErr	equality tolerance for distance calculations
optiCrit	criterion choice
simplitol	tolerance for spatial polygons geometry simplification

Details

Given a map object, a list of zonings, a current and a previous zoning, adds the current zoning to the list of zonings if it has at least 2 zones,after recalculating zone neighborhood and transferring zone labels.

Value

a list with components

listOfZ updated list of zoning objects, first element corresponds to initial zoning, each other element is a list with each (last if ALL=FALSE) level zoning objects

mdist list of initial distance matrix and all (last if ALL=FALSE) level distance matrices

crit list of initial criterion and all (last if ALL=FALSE) level criteria

cost list of initial cost and all (last if ALL=FALSE) level costs

costL list of initial cost per label and all (last if ALL=FALSE) level costs per label

nz list of initial number of zones and all (last if ALL=FALSE) level number of zones

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.7),mapTest,LASTPASS=FALSE,SAVE=TRUE)
K1=criti$zk[[1]][[1]]#initial zoning
Z1=K1$zonePolygone
printZsurf(Z1) # 8 zones with 2 small zones (7 and 8)
Z2 = remove1FromZ(Z1,7,K1$zoneN)
printZsurf(Z2) #7 zones
indCur=2
newRes=saveZK(mapTest,K1,Z2,c(0.4,0.7),criti$zk,indCur,
       criti$criterion,criti$cost,criti$costL,criti$nz,criti$mdist)
newZ=newRes$listOfZ[[2]][[1]]$zonePolygone
printZsurf(newZ) #6 zones
# not run
```

searchNODcrit *searchNODcrit*

Description

searchNODcrit

Usage

```
searchNODcrit(qProb, le, zk, criterion, cost, costL, nz)
```

Arguments

qProb	probability vector used to generate quantile values
le	level index
zk	list of zonings
criterion	list of criteria
cost	list of costs
costL	list of per label costs
nz	list of numbers of zones

Details

description, a paragraph

Value

a list with components

ind index of last level zoning that has the higher criterion value

critList criterion value corresponding to best last level zoning

costlist cost value corresponding to best last level zoning

costLlist cost per label value corresponding to best last level zoning

nzList number of zones of best last level zoning

nq lenght of quantile vector

Examples

```
qProb=c(0.1,0.2);criti=correctionTree(qProb,mapTest)
res=searchNODcrit1(qProb,criti)
# not run
```

---

searchNODcrit1 *searchNODcrit1*

Description

searchNODcrit1

Usage

```
searchNODcrit1(qProb, crit)
```

Arguments

qProb	probability vector used to generate quantile values
crit	result of call to correctionTree (with SAVE=TRUE)

Details

description, a paragraph

Value

a list with components

ind index of last level zoning that has the higher criterion value

critList criterion value corresponding to best last level zoning

costlist cost value corresponding to best last level zoning

costLlist cost per label value corresponding to best last level zoning

nzList number of zones of best last level zoning

Examples

```
data(mapTest)
qProb=c(0.1,0.2,0.4);criti=correctionTree(qProb,mapTest) # 2 zonings at last level
res=searchNODcrit1(qProb,criti)# best one is frist element of last level
# not run
```

separationPoly *separationPoly*

Description

separationPoly

Usage

```
separationPoly(polyTot)
```

Arguments

polyTot	a SpatialPolygons object

Details

separates holes and non holes

Value

a SpatialPolygons with holes in separate polygons

Examples

```
data(mapTest)
cL=list()
cL=contourAuto(cL,mapTest$step,mapTest$xsize,mapTest$ysize,mapTest$krigGrid,c(5,7),mapTest$boundary)
plot(mapTest$boundary,type="l",col="red")
graphics::lines(cL[[8]])
pG=polyToSp2(sp::Polygon(mapTest$boundary)) # transform boundary into SpatialPolygons objects
cLSp=maptools::ContourLines2SLDF(list(cL[[8]])) # transform contour line into SpatialLines objects
polyBuff=rgeos::gBuffer(cLSp,width=0.00001) # extend geometry
polyDiff=rgeos::gDifference(pG,polyBuff)
recupPoly=separationPoly(polyDiff)
Z1=list(recupPoly[[1]],recupPoly[[2]])
plotZ(Z1)
# not run
```

setId *setId*

Description

setId

Usage

```
setId(Z, iZ, id)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
iZ	zone number
id	zone identifier to assign

Details

assign zone identifier in a zoning

Value

a zoning geometry

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.5),mapTest,SAVE=TRUE)
Z=criti$zk[[2]][[1]]$zonePolygon
Z1=setId(Z,4,"4")
# not run
```

setIds *setIds*

Description

setIds

Usage

```
setIds(Z)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)

Details

set all zone identifiers in a zoning by assigning zone number to each identifier.

Value

a zoning geometry

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.5),mapTest,SAVE=TRUE)
Z=criti$zk[[2]][[1]]$zonePolygon
Z1=setIds(Z)
# not run
```

---

shape1 *An external zoning read from a shape file*

Description

An external zoning read from a shape file

Usage

```
shape1
```

Format

a SpatialPolygons object:

Sigmai2 *Sigmai2*

Description

Sigmai2

Usage

```
Sigmai2(index, listZonePoint, tabVal, surfaceVoronoi, meanZone)
```

Arguments

index	zone number
listZonePoint	list of pts within each zone
tabVal	SpatalPointsDataFrame holding data
surfaceVoronoi	vector of Voronoi surfaces associated to data values
meanZone	Zone mean values

Details

computes mean (weighted) variance and Voronoi area for zone

Value

a list with components sigmai2 and SI

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
Sigmai2(5,K$listZonePoint,mapTest$krigData,mapTest$krigSurfVoronoi,K$meanZone)
# not run
```

SigmaL2 *SigmaL2*

Description

SigmaL2

Usage

```
SigmaL2(zlab, listZonePoint, tabVal, surfVoronoi)
```

Arguments

zlab	list with zone numbers for each zone label
listZonePoint	list of indices of data points within zones, result of call to calNei
tabVal	SpatialPointsDataFrame containing data values
surfVoronoi	Surfaces of the Voronoi polygons corresponding to data pts

Details

compute overall mean and variance of all zones for each label plus sum of them for all labels

Value

a list with components

cL weighted (with Voronoi surfaces) average of per label variances

SigmaL2 vector of per label variances

SL vector of per label Voronoi surfaces

mL vector of weighted (with Voronoi surfaces) per label average values

voroLab vector of per label data

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.7),mapTest,SAVE=TRUE)
K=criti$zk[[2]][[1]]
uni=unique(K$lab)
zlab=sapply(uni,function(x){(1:length(K$lab))[K$lab==x]})
sig=SigmaL2(zlab,K$listZonePoint,mapTest$krigData,mapTest$krigSurfVoronoi)
# not run
```

---

smoothingMap *smoothingMap*

Description

smoothingMap

Usage

```
smoothingMap(Z, width = 0.01, map, disp = FALSE)
```

Arguments

Z	list of zone
width	smoothing parameter
map	object returned by function genMap or genMapR
disp	logical, if TRUE display the successful step of the program, otherwise do not display

Details

function that smooths all zones of map

Value

a new list of smoothed zones.

smoothingZone *smoothingZone*

Description

smoothingZone

Usage

```
smoothingZone(z, width, boundary, disp = TRUE)
```

Arguments

z zone to be modified (SpatialPolygon)

width smoothing parameter in gBuffer if dilatation is followed by erosion

boundary union of all zones of the corrected map (result of correctBoundaryMap())

disp logical, if TRUE, display the value of "widthExt" in case of dilatation->erosion, otherwise display "widthInt" in case of erosion->dilatation

Details

function that returns a new smoothed zones. Attention: this function is just a tool for a better visualisation of the map, if it doesn't work properly, please choose another value of the width parameter.

Value

a zone (SpatialPolygon)

Examples

```
seed=1
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2)
criti = correctionTree(qProb = c(0.5), map = map)
Z = criti$zk[[1]][[1]]$zonePolygone
lab = criti$zk[[1]][[1]]$lab
# zones' correction
res = correctBoundaryMap(Zi = Z, map = map)
Z = res$Z
# map boundary after correction
boundary = Z[[1]]
for(i in 2:length(Z)){
  boundary = rgeos::gUnion(boundary, Z[[i]])
}
# plot map
plotM(map = map, Z = Z, lab = lab, byLab = FALSE)
# smoothing
zone = Z[[2]]
```

```
newZone = smoothingZone(z = zone, width = 0.05, boundary = boundary)
sp::plot(zone)
sp::plot(newZone)
```

---

sortCrit *sortCrit called by correctionTree*

Description

sortCrit called by correctionTree

Usage

```
sortCrit(qProb, crit, cost, costL, nz, mdist, listOfZ, map, disp = 0,
  SAVE = FALSE)
```

Arguments

qProb	probability vector used to generate quantile values
crit	list of criteria
cost	list of costs
costL	list of per label costs
nz	list of number of zones
mdist	list of distance matrices
listOfZ	list of zoning objects
map	object returned by function genMap or genMapR
disp	0: no info, 1: plot best corrected zoning
SAVE	logical value, if TRUE function returns more elements

Details

sort last level criteria from list of zonings, return criteria and list of zonings if SAVE=TRUE, otherwise only return last level criteria

Value

a list with components

bestcrit best criterion value at last level

critList criterion values at last level

costList cost values at last level

costLList cost per label values at last level

nzList vector of number of zones at last level

qProb vector of probabilities values used for quantiles

- **zk** (SAVE=TRUE) list of zoning objects (such as returned by calNei function), first element corresponds to initial zoning, each other element is a list with each (last if ALL=FALSE) level zoning objects
- **mdist** (SAVE=TRUE) list of initial distance matrix and all (last if ALL=FALSE) level distance matrices
- **crit** (SAVE=TRUE) list of initial criterion and all (last if ALL=FALSE) level criteria
- **cost** (SAVE=TRUE) list of initial cost and all (last if ALL=FALSE) level costs
- **costL** (SAVE=TRUE) list of initial cost per label and all (last if ALL=FALSE) level costs per label
- **nz** (SAVE=TRUE) list of initial number of zones and all (last if ALL=FALSE) level number of zones

Examples

```
data(mapTest)
qProb=c(0.4,0.7)
criti=correctionTree(qProb,mapTest)
sortCrit(qProb,criti$criterion,criti$cost,criti$costL,criti$nz,criti$mdist,criti$zk,mapTest)
# not run
```

---

spnorm *spnorm*

Description

spnorm

Usage

```
spnorm(sp, boundary)
```

Arguments

sp	object of class Polygons
boundary	list with x and y components, used to normalize sp

Details

normalize Polygon according to border limits

Value

a list with components

- **pn** normalized Polygon
- **boundaryn** normalized boundary

Examples

```
z=geozoning::shape1
bb=list(x=z@bbox[1,],y=z@bbox[2,])
p=z@polygons
p1=p[[1]]
P1=p1@Polygons[[1]]
NP1=spnorm(P1,bb)$pn
Nbb=spnorm(P1,bb)$boundaryn
plot(NP1@coords,xlim=Nbb$x,ylim=Nbb$y)
# not run
```

---

spToSL *spToSL*

Description

spToSL

Usage

```
spToSL(sp)
```

Arguments

sp	SpatialPolygons

Details

tranform SpatialPolygons into SpatialLines

Value

a SpatialLines

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
spToSL(Z[[5]])
# not run
```

superLines *superLines*

Description

superLines

Usage

```
superLines(boundary)
```

Arguments

boundary	list, contains x and y coordinates of map boundaries

Details

converts boundary (list of x and y pts) into Spatial Lines

Value

a SpatialLines object

Examples

```
data(mapTest)
superL=superLines(mapTest$boundary)
sp::plot(superL)
# not run
```

testInterSpe *testInterSpe*

Description

testInterSpe

Usage

```
testInterSpe(Z, i1, i2)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
i1	first zone number
i2	second zone number

Details

checks if 2 zones in a zoning share somme common part (using gOverlaps)

Value

a Logical value, TRUE if there is an intersection, FALSE if not.

Examples

```
data(mapTest)
qProb=c(0.2,0.5)
ZK = initialZoning(qProb, mapTest)
K=ZK$resZ
Z=K$zonePolygone
plotZ(Z)
Z58=rgeos::gConvexHull(rgeos::gUnion(Z[[8]],Z[[5]]))
Z[[length(Z)+1]]=Z58 # add new zone to zoning
plotZ(Z)
testInterSpe(Z,6,length(Z))
# not run
```

testInterSpeZ *testInterSpeZ*

Description

testInterSpeZ

Usage

```
testInterSpeZ(Z)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)

Details

checks, within a zoning, if any zone intersects with any other zone not within it and not englobing it

Value

a Logical value, TRUE if there is any intersection, FALSE if not

Examples

```
qProb=c(0.2,0.5)
ZK = initialZoning(qProb, mapTest)
K=ZK$resZ
Z=K$zonePolygone
plotZ(Z)
Z58=rgeos::gConvexHull(rgeos::gUnion(Z[[8]],Z[[5]]))
Z[[length(Z)+1]]=Z58 # add new zone to zoning
plotZ(Z)
testInterSpeZ(Z)
# not run
```

---

testInterSpeZ1 *testInterSpeZ1*

Description

testInterSpeZ1

Usage

```
testInterSpeZ1(Z, iZ)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
iZ	zone number

Details

checks, within a zoning, if a given zone intersects with any other zone not within it

Value

a Logical value, TRUE if there is an intersection, FALSE if not.

Examples

```
data(mapTest)
qProb=c(0.2,0.5)
ZK = initialZoning(qProb, mapTest)
K=ZK$resZ
Z=K$zonePolygone
plotZ(Z)
Z58=rgeos::gConvexHull(rgeos::gUnion(Z[[8]],Z[[5]]))
Z[[length(Z)+1]]=Z58 # add new zone to zoning
plotZ(Z)
testInterSpe(Z,6,length(Z))
# not run
```

touch.border *touch.border*

Description

touch.border

Usage

```
touch.border(z, boundary)
```

Arguments

z	a zone (SpatialPolygon)
boundary	union of all zones of the corrected map (result of correctBoundaryMap())

Details

verify if a zone has a commun boundary with the map

Value

logical, TRUE if zone has a commun boundary with the map, FALSE otherwise

Examples

```
seed=1
map = genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2)
criti = correctionTree(qProb = c(0.5), map = map)
Z = criti$zk[[1]][[1]]$zonePolygone
lab = criti$zk[[1]][[1]]$lab
# zone correction
res = correctBoundaryMap(Zi = Z, map = map)
Z = res$Z
# map boundary after correction
boundary = Z[[1]]
for(i in 2:length(Z)){
  boundary = rgeos::gUnion(boundary, Z[[i]])
}
# plot map
plotM(map = map, Z = Z, lab = lab, byLab = FALSE)
# verification
for(i in 1:length(Z)){
  print(touch.border(z = Z[[i]], boundary = boundary))
}
```

Transition_Zone_Far_Boundary
Transition_Zone_Far_Boundary

Description

Transition_Zone_Far_Boundary

Usage

```
Transition_Zone_Far_Boundary(map, Z, numZ)
```

Arguments

map	object returned by function genMap or genMapR
Z	list of zones.
numZ	number of the zone whose values will be approximated.

Details

funtion that approximates the value in a transition zone (which doesn't have commun boundary with the map) by the solution of the Laplace's equation. The numerical resolution of the Laplace's equation will be based on the discretisation of the data on the grid (map$krigGrid).

Value

approximated values of the zone (numZ) given as parameter.

Examples

```
seed=9
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2)
ZK=initialZoning(qProb=c(0.65,0.8),map)
Z=ZK$resZ$zonePolygone # list of zones
lab = ZK$resZ$lab # label of zones
plotM(map = map,Z = Z,lab = lab, byLab = FALSE)
# zone 7 is a transition zone that is far from map boundary
numZ = 7
Estimation = Transition_Zone_Far_Boundary(map = map, Z = Z, numZ = numZ)
# compute the cost
cL = Cost_By_Laplace(map = map, Z = Z, numZ = numZ, Estimation = Estimation)
cM = Cost_By_Mean(map = map, Z = Z, numZ = numZ)
print(cL$cost_Laplace)
print(cM$cost_Mean)
# zone 7 is a zone with gradient
```

Transition_Zone_Near_Boundary

Transition_Zone_Near_Boundary

Description

Transition_Zone_Near_Boundary

Usage

```
Transition_Zone_Near_Boundary(map, Z, numZ)
```

Arguments

map	object returned by function genMap or genMapR
Z	list of zones.
numZ	number of the zone whose values will be approximated.

Details

funtion that approximates the value in a transition zone (which has commun boundary with the map) by the solution of the Laplace's equation. The numerical resolution of the Laplace's equation will be based on the discretisation of the data on the grid (map$krigGrid). The domaine of study is a transition zone which have a commun border with the map.

Value

approximated values of the zone (numZ) given as parameter.

Examples

```
seed=2
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2)
ZK=initialZoning(qProb=c(0.55,0.85),map)
Z=ZK$resZ$zonePolygone # list of zones
lab = ZK$resZ$lab # label of zones
plotM(map = map,Z = Z,lab = lab, byLab = FALSE)
# zone 6 is a transition zone that has commun boundary with the map
numZ = 6
Estimation = Transition_Zone_Near_Boundary(map = map, Z = Z, numZ = numZ)
# compute the cost
cL = Cost_By_Laplace(map = map, Z = Z, numZ = numZ, Estimation = Estimation)
cM = Cost_By_Mean(map = map, Z = Z, numZ = numZ)
print(cL$cost_Laplace)
print(cM$cost_Mean)
# zone 6 is a zone with gradient
```

trLabZone *trLabZone*

Description

trLabZone

Usage

```
trLabZone(K1, K2, map, qProb, disp = 0)
```

Arguments

K1	zoning object (such as returned by calNei function)
K2	zoning object (such as returned by calNei function)
map	object returned by genMap function
qProb	probability vector used to generate quantile values
disp	0: no info, 1: detailed info

Details

transfer zone labels from K1 to K2

Value

a zoning object

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
Ns=getNs(K$zoneNModif,5) # find neighbors of zone 5
zf=zoneFusion3(K,5,Ns,mapTest,disp=0) # merge zone 5 with englobing one
K2=calNei(zf,mapTest$krigData,mapTest$krigSurfVoronoi,mapTest$krigN)
K2=trLabZone(K,K2,mapTest,K$qProb)
# not run
```

updateZK *updateZK called by lastPass*

Description

updateZK called by lastPass

Usage

```
updateZK(map, qProb, le, kk, listOfZ, crit, cost, costL, nz, mdist, K1, Z2,
  pErr = 0.9, optiCrit = 2, simplitol = 0.001)
```

Arguments

map	object returned by function genMap or genMapR
qProb	probability vector used to generate quantile values
le	index of current level in list
kk	index of current zoning in level list
listOfZ	list of zoning objects
crit	list of criteria
cost	list of costs
costL	list of per label costs
nz	list of number of zones
mdist	list of distance matrices
K1	zoning to be replaced
Z2	xxxx
pErr	equality tolerance for distance calculations
optiCrit	xxxx
simplitol	xxxx

Details

Given a map object, a list of zonings, a current and a previous zoning, replaces a zoning in the list of zonings

Value

a ?

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.7),mapTest,LASTPASS=FALSE)
K1=criti$zk[[1]][[1]]#initial zoning
Z1=K1$zonePolygone
printZsurf(Z1) # 8 zones with 2 small zones (7 and 8)
Z2 = remove1FromZ(Z1,7,K1$zoneN)
printZsurf(Z2) #7 zones
newRes=updateZK(mapTest,c(0.4,0.7),1,1,criti$zk[1],criti$criterion[1],
       criti$cost[1],criti$costL[1],criti$nz[1],criti$mdist[1],K1,Z2)
newZ=newRes$listOfZ[[1]][[1]]$zonePolygone
printZsurf(newZ) #7 zones
# not run
```

valZ *valZ*

Description

valZ

Usage

```
valZ(map, K)
```

Arguments

map	map object returned by genMap function
K	zoning object (such as returned by calNei function)

Details

sorts zones according to attribute mean value

Value

a list with components

val list with vector of data values for each zone, zones are sorted by increasing mean values

ord order of zones sorted by increasing mean values

Examples

```
data(mapTest)
criti=correctionTree(c(0.4,0.7),mapTest,SAVE=TRUE)
K=criti$zk[[2]][[1]]
valZ(mapTest,K)
# not run
```

voronoiPolygons *voronoiPolygons*

Description

voronoiPolygons

Usage

```
voronoiPolygons(spdata, gridLim = c(0, 1, 0, 1), neighBool,
  PTJUNCTION = FALSE, FULL = FALSE)
```

Arguments

spdata	SpatialPointsDataFrame
gridLim	list of boundary coordinates
neighBool	empty point neighborhood Logical matrix
PTJUNCTION	logical value, if FALSE (default): pts are not neighbors if their Voronoi polygons only have a vertex in common
FULL	logical value, if FALSE (default): do not return Vornoi polygons

Details

determines the Voronoi neighborhood of data points

Value

a list with components

surfVoronoi Voronoi polygons areas

neighBool Voronoi point neighborhood Logical matrix if FULL=TRUE (warning: uses a lot of memory space), also:

voronoi Voronoi polygons

See Also

http://www.carsonfarmer.com/2009/09/voronoi-polygons-with-r/

Examples

```
data(mapTest)
rx=range(mapTest$krigData$x)
ry=range(mapTest$krigData$y)
nx=nrow(mapTest$krigGrid)
ny=ncol(mapTest$krigGrid)
nB=matrix(logical((nx*ny)^2),nx*ny,nx*ny) # big matrix
vP=voronoiPolygons(mapTest$krigData,c(rx,ry),nB)
```

```
length(vP$surfVoronoi) #as many as kriged data points
# not run
```

wMean *wMean*

Description

wMean

Usage

```
wMean(type, listZonePoint, surfVoronoi, data)
```

Arguments

type	1-squared mean, 2-mean
listZonePoint	list of data points belonging to zone
surfVoronoi	areas of Voronoi polygon corresponding to data points
data	SpatialPointsDataFrame

Details

computes weighted mean or squared mean of zone data

Value

a vector of mean zone values

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
wMean(1,K$listZonePoint,mapTest$krigSurfVoronoi,mapTest$krigData)
# not run
```

yield *A data frame with real data used for zoning*

Description

A data frame with real data used for zoning

Usage

```
yield
```

Format

a data frame with 6415 rows and 3 variables:

x x coordinate

y y coordinate

Yield numeric variable - phenotype

zone.extended *zone.extended*

Description

zone.extended

Usage

```
zone.extended(z, boundary)
```

Arguments

z	a zone of the map
boundary	union of all zones of the corrected map (result of correctBoundaryMap())

Details

for a zone that has commun border with the map, it will be extended at the side of commun border. We search the commun border which is a spatiaLines. This spatialLines is composed of several Lines containing only 2 points. For each Lines, we project the 2 points to the convexHull of the "relaxation" of the map's boundary. We have then 4 points (2 come from a Line, 2 come from the projection). with 4 points, we will have a SpatialPolygone which is the extension part of the Line.

Examples

```
seed=1
map = genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2)
criti = correctionTree(qProb = c(0.5), map = map)
Z = criti$zk[[1]][[1]]$zonePolygone
lab = criti$zk[[1]][[1]]$lab
# zones' correction
res = correctBoundaryMap(Zi = Z, map = map)
Z = res$Z
# map boundary after correction
boundary = Z[[1]]
for(i in 2:length(Z)){
  boundary = rgeos::gUnion(boundary, Z[[i]])
}
# plot map
plotM(map = map, Z = Z, lab = lab, byLab = FALSE)
# extend zone
z = zone.extended(z = Z[[1]], boundary = boundary)
sp::plot(z)
sp::plot(Z[[1]],add=TRUE)
```

---

zoneAssign *zoneAssign*

Description

zoneAssign

Usage

```
zoneAssign(tab, Z)
```

Arguments

tab	data frame with data values
Z	zoning object

Details

assigns points to zones

Value

a list of data points within each zone

Examples

```
data(mapTest)
ZK=initialZoning(qProb=c(0.4,0.7),mapTest)
Z=ZK$resZ$zonePolygone
listZpts=zoneAssign(mapTest$krigData,Z)
#identical to ZK$resZ$listZonePoint
listZptsRaw=zoneAssign(mapTest$rawData,Z)
plotZ(Z)
points(mapTest$rawData[listZptsRaw[[1]],],col="blue") # add raw data for zone 1
# not run
```

---

zoneFusion2 *zoneFusion2 basic function for merging 2 zones*

Description

zoneFusion2 basic function for merging 2 zones

Usage

```
zoneFusion2(zoneMain, zoneSuppr, simplitol = 0.001)
```

Arguments

zoneMain	zone to merge into
zoneSuppr	zone to remove by merging it into main zone
simplitol	tolerance for spatial polygons geometry simplification

Details

merge 2 zones, called by zoneFusion3 and zoneFusion4

Value

a zone

Examples

```
data(resZTest)
Z=resZTest$zonePolygone
plotZ(Z)
sp::plot(zoneFusion2(Z[[6]],Z[[2]]),add=TRUE,col="blue")
# not run
```

zoneFusion3 *zoneFusion3*

Description

zoneFusion3

Usage

```
zoneFusion3(K, iC, Ns, map, minSize = 0.01, simplitol = 0.001, disp = 0)
```

Arguments

K	zoning object, as returned by the calNei function
iC	index of current zone in zoning
Ns	zone neighborhood Boolean matrix
map	object returned by function genMap or genMapR
minSize	minimum admissible zone size
simplitol	tolerance for spatial polygons geometry simplification
disp	information level (0-no info, 1-print info, 2-plot)

Details

merge current zone #iC with neighbor zone in zoning. If there are several neighbor zones, the selected one is the zone whose area is greater than the admissible size threshold that has the closest average value to the current one.

Value

a zone obtained by merging current zone with neighbor zone

Examples

```
data(mapTest)
data(resZTest)
K=resZTest
Ns=getNs(K$zoneNModif,5) # find neighbors of zone 5
zoneFusion3(K,5,Ns,mapTest,disp=2) # merge and plot result of merging
```

zoneFusion4 *zoneFusion4*

Description

zoneFusion4

Usage

```
zoneFusion4(Z, iSmall, iBig, simplitol = 0.001, disp = 0)
```

Arguments

Z	zoning geometry (list of SpatialPolygons)
iSmall	index of zone to remove by merging it into other zone
iBig	index of zone to merge into
simplitol	tolerance for spatial polygons geometry simplification
disp	0: no info, 1: some info

Details

merge 2 zones from given zoning

Value

a new zoning geometry

Examples

```
data(resZTest)
K=resZTest
Z=K$zonePolygone
zoneFusion4(Z,5,4,disp=2)
# not run
```

zoneGeneration *zoneGeneration*

Description

zoneGeneration

Usage

```
zoneGeneration(map, qProb = c(0.25, 0.75), GridData = FALSE)
```

Arguments

map	object returned by function genMap or genMapR
qProb	probability vector used to generate quantile values
GridData	logical value indicating if data are already on a regular grid (no kriging in that case)

Details

Generates zones from map data using quantile values associated to given probabilities

Value

a list of zones, each zone is a SpatialPolygons

Examples

```
data(mapTest)
Z=zoneGeneration(mapTest)
# not run
```

zoneGrow *zoneGrow*

Description

zoneGrow

Usage

```
zoneGrow(K, map, iC, optiCrit = 2, minSize = 0.012, minSizeNG = 0.001,
  distIsoZ = 0.075, LEQ = 5, MAXP = 0.1, simplitol = 0.001, disp = 0)
```

Arguments

K	zoning object, such as returned by the calNei function
map	object returned by function genMap
iC	index of current zone
optiCrit	criterion choice
minSize	admissible zone area threshold
minSizeNG	zone area threshold under which a zone will be removed
distIsoZ	threshold distance to next zone, above which a zone is considered to be isolated
LEQ	length of quantile sequence used to grow isolated zone
MAXP	quantile sequence maximum shift from center
simplitol	tolerance for spatial polygons geometry simplification
disp	information level (0-no info, 1-print info)

Details

either grow isolated zone or group 2 zones together if isolated zone, run optimization procedure to find the new quantile if zone very small (area < minSizeNG) do not grow it

Value

a zone obtained by growing current zone

Examples

```
data(mapTest)
qProb=c(0.2,0.5)
ZK = initialZoning(qProb, mapTest)
K=ZK$resZ
Z=K$zonePolygone
plotZ(K$zonePolygone) # plot zoning
kmi=zoneGrow(K,mapTest,6) # grow zone 6 by grouping it with its closest neighbor with same label
linesSp(kmi[[7]])
qProb=c(0.3,0.5)
criti = correctionTree(qProb,mapTest)
best = criti$zk[[2]][[1]]
Z=best$zonePolygone
plotZ(Z)
refPoint = rgeos::gCentroid(Z[[4]])
sp::plot(refPoint,add=TRUE,col="blue",pch=21)
zg=zoneGrow(best,mapTest,4) #grow isolated zone 4 by searching for other quantile
plotZ(zg)
# not run
```

---

zoneModifnonIso *zoneModifnonIso*

Description

zoneModifnonIso

Usage

```
zoneModifnonIso(K, qProb, map, zoneClose, iC, simplitol = 0.001, disp = 0)
```

Arguments

K	zoning object (such as returned by calNei function)
qProb	probability vector used to generate quantile values
map	object returned by function genMap or genMapR
zoneClose	indices of close zones
iC	current zone index
simplitol	tolerance for spatial polygons geometry simplification
disp	0: no info, 1: detailed info

Details

modify non isolated zone (depends on distIsoZ parameter) so that it is joined to the closest neighbour zone with the same label.

Value

a zoning object

Examples

```
data(mapTest)
qProb=c(0.2,0.5)
ZK = initialZoning(qProb, mapTest)
K=ZK$resZ
Z=K$zonePolygone
plotZ(Z)
resP=detZoneClose(6,Z,K$zoneNModif) # zone 6 is close to zone 5 and zone 7
zoneClose = resP$zoneClose
kmi = zoneModifnonIso(K,qProb,mapTest,zoneClose,6,disp=1)
plotZ(kmi$zonePolygone) # zones 6 and 7 are joined into new zone 6
# now it is the turn of zone 5
Z=kmi$zonePolygone
resP=detZoneClose(5,Z,kmi$zoneNModif) # zone 5 is close to zone 7 and zone 6
kmi2 = zoneModifnonIso(kmi,qProb,mapTest,resP$zoneClose,5,disp=1)
plotZ(kmi2$zonePolygone) # zones 5 and 6 are joined into new zone 5
# not run
```

---

zoneQ *zoneQ*

Description

zoneQ

Usage

```
zoneQ(contourSp, iC, iE, Z, K, simplitol = 0.001)
```

Arguments

contourSp	contour line transformed into SpatialPolygons
iC	zone to grow
iE	englobing zone
Z	zoning geometry (list of SpatialPolygons)
K	zoning object (such as returned by calNei function)
simplitol	tolerance for spatial polygons geometry simplification

Details

called by optiGrow,replaces the current zone by a bigger one

Value

a zoning geometry updated with the grown zone(list of SpatialPolygons)

Examples

```
data(mapTest)
qProb=c(0.3,0.5)
criti = correctionTree(qProb,mapTest)
K = criti$zk[[2]][[1]]
Z=K$zonePolygone
plotZ(Z)
iC=4
iE=detZoneEng(iC,Z,K$zoneNModif)
envel=calFrame(iC,Z,K$zoneNModif)
sp::plot(envel,add=TRUE,col="blue")
Qseq = genQseq(qProb,K,mapTest,iC,iE)
resi = findCinZ(iC,Z,K,mapTest,Qseq[5],envel)
Zopti=zoneQ(resi$contourSp,iC,iE,Z,K)
plotZ(Zopti)
# not run
```

Index