Geozoning structures

B. Charnomordic

2018-02-06

Contents

```
library(geozoning)
library(sp)
library(fields)
library(gstat)
```

This vignette describes the main structures used by the geozoning package: map, zoning object and zoning geometry, how to create and use them. We illustrate as well the criterion behaviour on 2 different zonings.

Structure 1: map

A map structure contains the data and parameters necessary to perform a zoning. The structure is identical wether the data are simulated or not. The map structure contains raw data for traceability: rawData component (class SpatialPointsDataFrame) and kriged data in 2 useful forms: krigData component (class SpatialPointsDataFrame) used for zoning calculations and krigGrid component (class matrix), used for image plot displays. The resolution grid is given in the step component. Kriged data are normalized so that x-coordinates are between 0 and 1. y-coordinates are normalized with the same ratio used for x-coordinates. The ratio is recorded in the ratio component. Kriging is either done with inverse distance interpolation, or with a variogram model fitted to data, which is recorded in 2 components: VGMmodel (variogramModel) and modelGen (class RMmodel). The boundary component is a list with x and y components, normalized with the same ratio used for data, corresponding to the map boundary. The map structure also contains the list of neighbours of each kriged data point: krigN. For each kriged data point, the corresponding list element contains the sorted indexes of all its neighbours (sharing an edge in the kriged point Voronoi tesselation). The list of areas of kriged Voronoi polygons is given in the krigSurfVoronoi component. It is used to assign weights in calculations.

The map object can be created by 2 functions: *genMap* or *randKmap*.

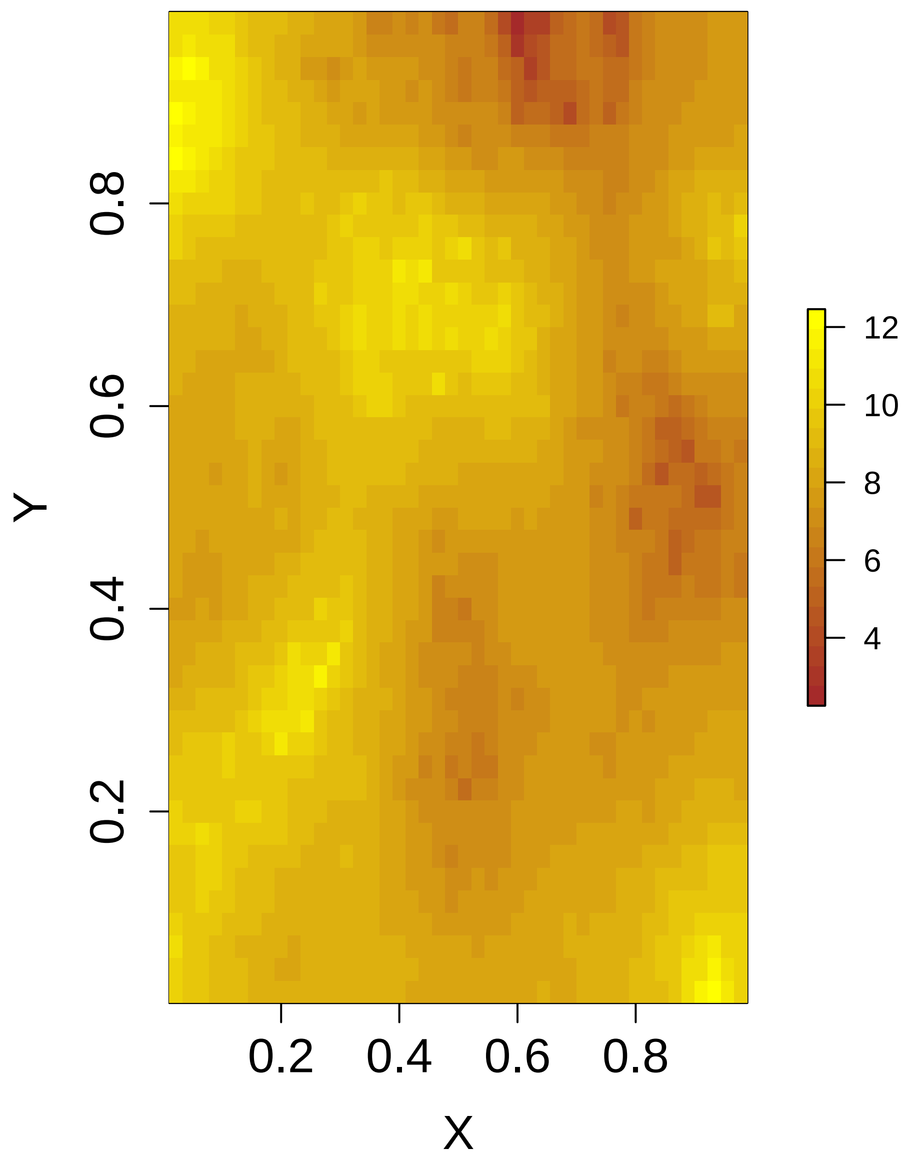
```r
seed=80
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,Vmean=15,krig=2,typeMod="Gau")
```

```
## [1] "DataObj=NULL, generating DataObj-seed= 80"
## [inverse distance weighted interpolation]
```

```r
# or with the randKmap function:
map=randKmap(DataObj=NULL,nPointsK=500,Vmean=10,krig=1)
```

Display map with kriged data

```r
dispZ(mapTest$step,matVal=mapTest$krigGrid)
```



```
## NULL
```

Check the mean and standard deviation of generated data.

```
meanvarSimu(map)
```

```
##    raw mean kriged mean      raw sd   kriged sd
##   15.327609   15.244374    1.969228    1.310783
```

Structure 2: zoning object with different criterion behaviour

This section illustrates the criterion behaviour on 2 different zonings in 3 steps, using the *loopQ3* and *correctionTree* functions. First an exploraty loop is run to rank the zonings for a size 3 quantile vector (4-label map). Then the correction procedure is run independently for the best and the worst zonings found in that exploratory loop. The criterion values are printed and explained.

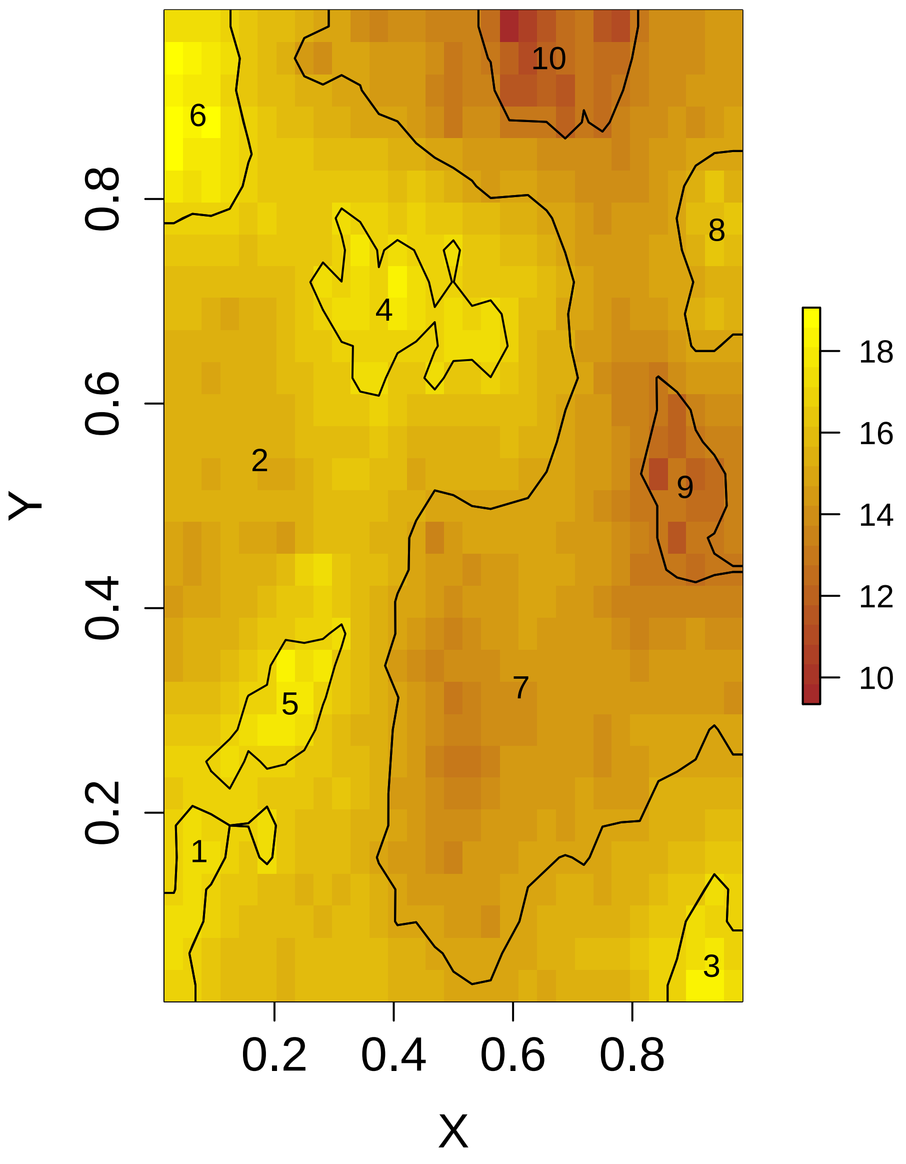
Exploratory loop for a size 3 quantile vector (4-label map)

```
ro=loopQ3(map,step=0.1,disp=0,QUIET=T)
```

ro is a matrix sorted by reverse order of crit. It has 8 columns and 56 rows. 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. Each row corresponds to the best zoning obtained for the corresponding quantile at the end of the correction procedure. To save time and memory, details are not saved.

Run *correctionTree* function for best zoning and save results

```
bqProb=ro[1,5:7]
criti=correctionTree(bqProb,map,SAVE=TRUE)
res=searchNODcrit1(bqProb,criti)
b=res$ind[[1]][1]
K=criti$zk[[2]][[b]]
bZ=K$zonePolygone
dispZ(map$step,map$krigGrid,zonePolygone=bZ)
```



```
## NULL
```

```
# distance matrix has high values, criterion is the smallest one (6.417)
# distance between zones 5 and 7
bmd=criti$mdist[[2]][[b]]
bcrit=criti$criterion[[2]][[b]]
bcrit
```

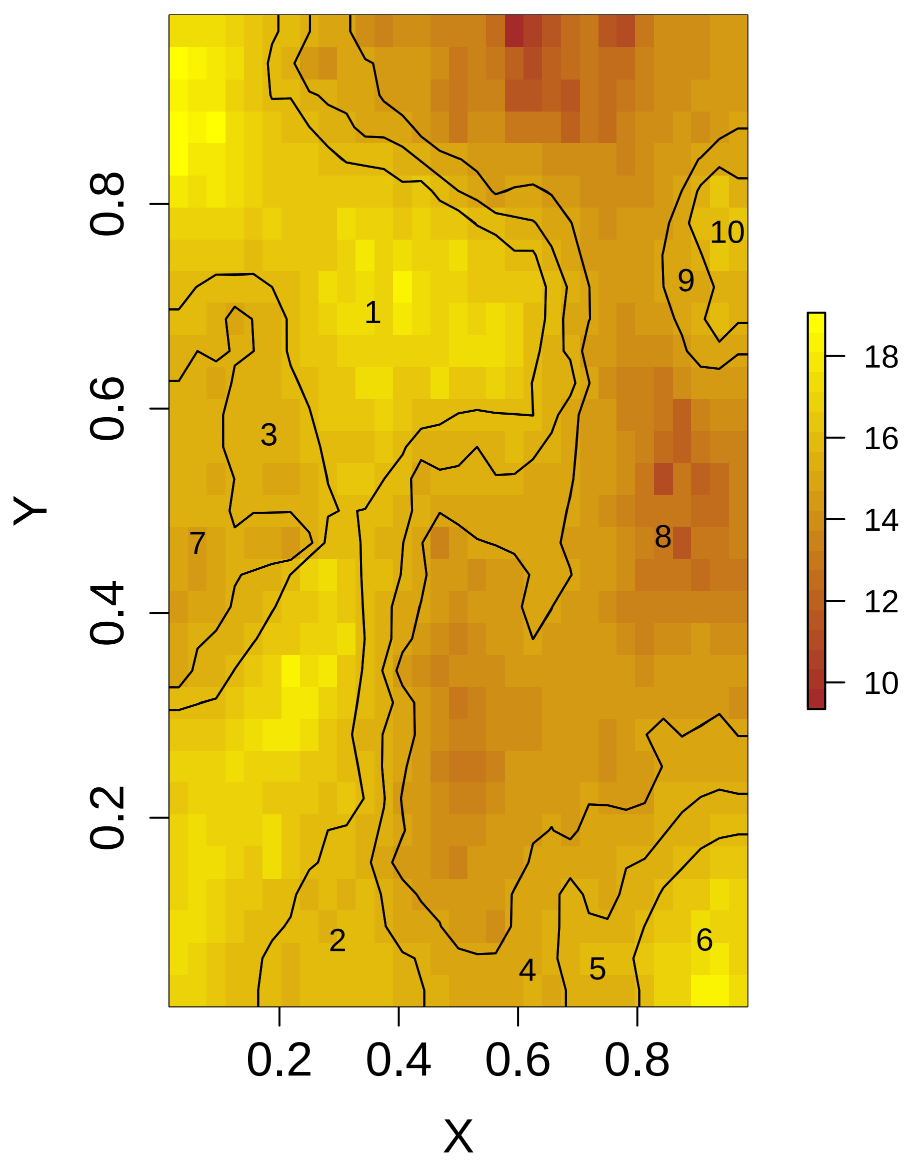
```
## [1] 5.148598
```

```
bmd
```

```
##           [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## [1,] 1.000000 6.272111 0.000000 0.000000 0.000000 0.000000 0.000000
## [2,] 6.272111 1.000000 6.391126 6.044603 6.542246 6.664077 5.550692
## [3,] 0.000000 6.391126 1.000000 0.000000 0.000000 0.000000 0.000000
##   [4,] 0.000000 6.044603 0.000000 1.000000 0.000000 0.000000 0.000000
##   [5,] 0.000000 6.542246 0.000000 0.000000 1.000000 0.000000 0.000000
##   [6,] 0.000000 6.664077 0.000000 0.000000 0.000000 1.000000 0.000000
##   [7,] 0.000000 5.550692 0.000000 0.000000 0.000000 0.000000 1.000000
##   [8,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 5.148598
##   [9,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 6.968832
##  [10,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 6.075828
##            [,8]     [,9]    [,10]
##   [1,] 0.000000 0.000000 0.000000
##   [2,] 0.000000 0.000000 0.000000
##   [3,] 0.000000 0.000000 0.000000
##   [4,] 0.000000 0.000000 0.000000
##   [5,] 0.000000 0.000000 0.000000
##   [6,] 0.000000 0.000000 0.000000
##   [7,] 5.148598 6.968832 6.075828
##   [8,] 1.000000 0.000000 0.000000
##   [9,] 0.000000 1.000000 0.000000
##  [10,] 0.000000 0.000000 1.000000
```

Run *correctionTree* function for worst zoning and save results

```
wqProb=ro[56,5:7]
criti=correctionTree(wqProb,map,SAVE=TRUE)
res=searchNODcrit1(wqProb,criti)
w=res$ind[[1]][1]
K=criti$zk[[2]][[w]]
wZ=K$zonePolygone
dispZ(map$step,map$krigGrid,zonePolygone=wZ)
```



```
## NULL
```

```
# distance matrix has some low values, criterion is the smallest one (3.747)
# distance between zones 4 and 8
wmd=criti$mdist[[2]][[w]]
wcrit=criti$criterion[[2]][[w]]
wcrit
```

```
## [1] 2.433347
```

```
wmd
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## [1,] 1.000000 3.902011 4.484055 8.301718 0.000000 0.000000 6.957725
## [2,] 3.902011 1.000000 0.000000 7.684557 0.000000 0.000000 0.000000
## [3,] 4.484055 0.000000 1.000000 0.000000 0.000000 0.000000 2.433347
##  [4,] 8.301718 7.684557 0.000000 1.000000 5.838337 0.000000 0.000000
##  [5,] 0.000000 0.000000 0.000000 5.838337 1.000000 4.745025 0.000000
##  [6,] 0.000000 0.000000 0.000000 0.000000 4.745025 1.000000 0.000000
##  [7,] 6.957725 0.000000 2.433347 0.000000 0.000000 0.000000 1.000000
##  [8,] 0.000000 5.716722 0.000000 2.910667 0.000000 0.000000 0.000000
##  [9,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
## [10,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
##           [,8]     [,9]    [,10]
##  [1,] 0.000000 0.000000 0.000000
##  [2,] 5.716722 0.000000 0.000000
##  [3,] 0.000000 0.000000 0.000000
##  [4,] 2.910667 0.000000 0.000000
##  [5,] 0.000000 0.000000 0.000000
##  [6,] 0.000000 0.000000 0.000000
##  [7,] 0.000000 0.000000 0.000000
##  [8,] 1.000000 2.821798 0.000000
##  [9,] 2.821798 1.000000 6.773802
## [10,] 0.000000 6.773802 1.000000
```

Structure 2: zoning geometry

We use the **mapTest** internal object and the *initialZoning* function for the following examples:

```
data(mapTest)
ZK=initialZoning(qProb=c(0.4,0.7),mapTest)
```

A zoning geometry Z contains the polygons corresponding to the boundaries of zones within a zoning. For size limiting considerations, it contains no other data (see zoning structure that contains data and zone geometry). Z is a list of SpatialPolygons. Some zones may have holes, in that case there is always an additional SpatialPolygons corresponding to the hole.

Z can be created or updated by many functions: *initialZoning*, *correctionTree*, *zoneFusion2*, *optiGrow*, etc.

```
Z=ZK$resZ$zonePolygone
class(Z)
```

```
## [1] "list"
```

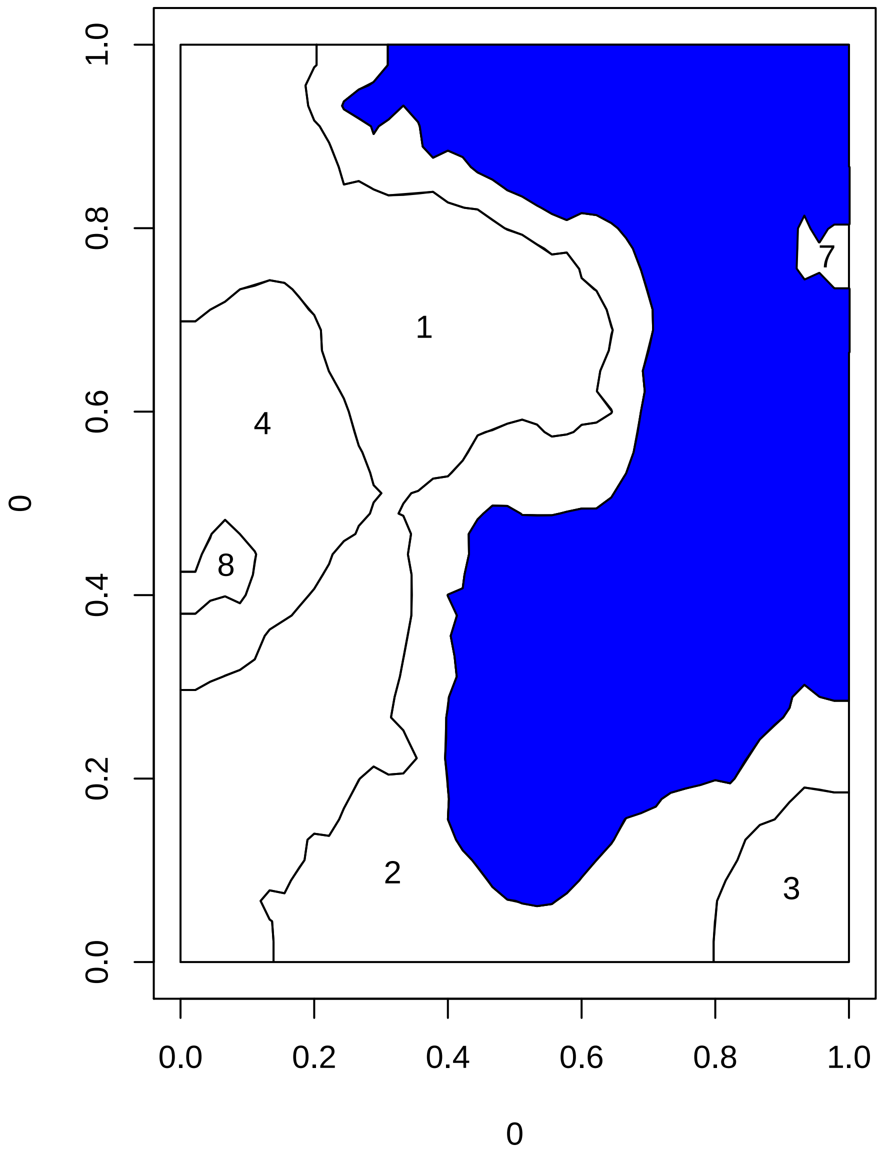
```
plotZ(Z)
```

```
## NULL
```

```
Zf=zoneFusion2(Z[[5]],Z[[6]])
class(Zf)
```

```
## [1] "SpatialPolygons"
## attr(,"package")
## [1] "sp"
```

```r
sp::plot(Zf,add=TRUE,col="blue")
```



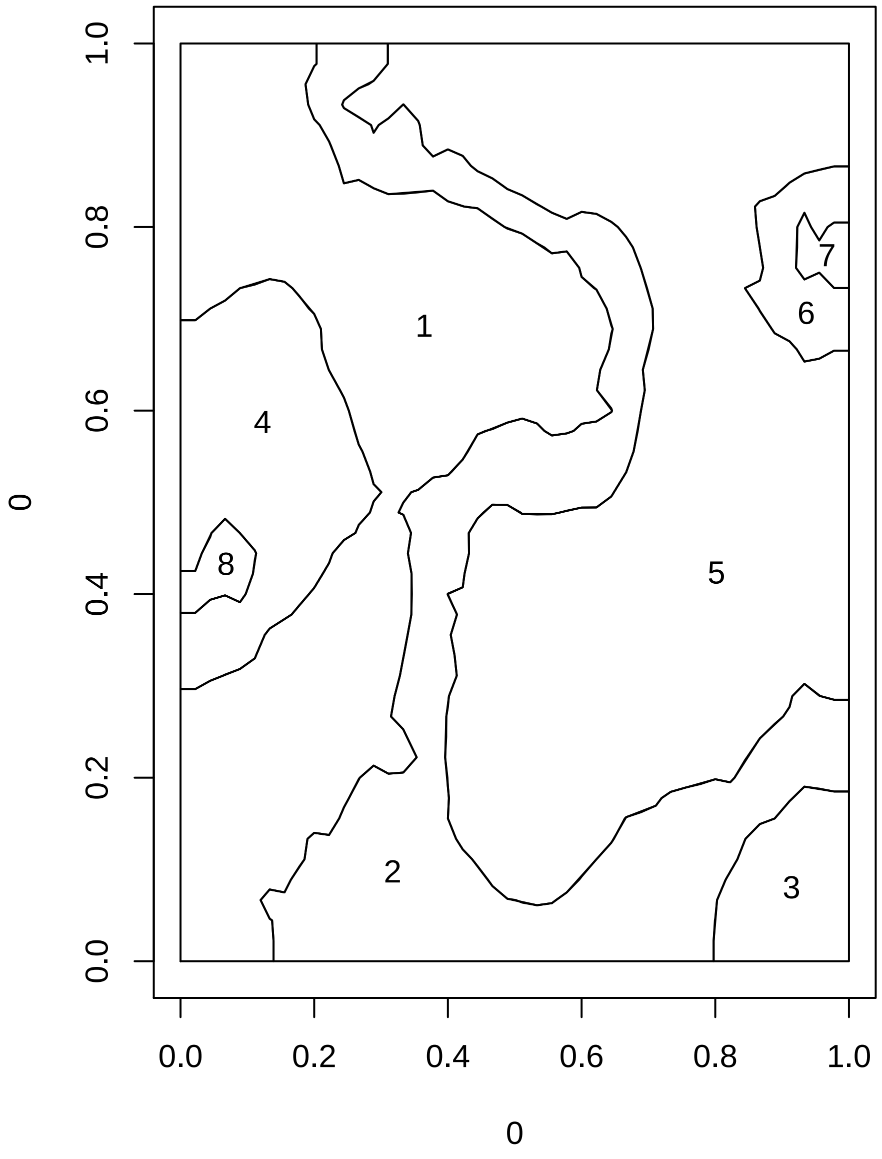
Structure 3: zoning structure

A zoning structure K contains all elements resulting from a zoning: zoning geometry (zonePolygone component - list of SpatialPolygons), zone neighborhood (zoneN or zoneNmodif with no self-neighborhood), data point zone assignment, zone areas and mean values. It is created by the calNei function, that is called by many functions, for instance from within initialZoning or correctedTree.

```r
K=ZK$resZ
names(K)
```

```
## [1] "zoneN"          "zoneNModif"     "listZonePoint" "meanTot"
## [5] "critSurf"       "meanZone"       "listSurf"      "zonePolygone"
## [9] "lab"            "qProb"
```

```r
Z=K$zonePolygone
plotZ(Z)
```



```
## NULL
```

The *labZone* function may be called to assign zone labels depending on the zone mean value as follows. Default label is 1, corresponding to a mean 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 label is p+1.

```r
p = K$qProb
K=labZone(K,p,mapTest$krigGrid)
print(K$lab)
```

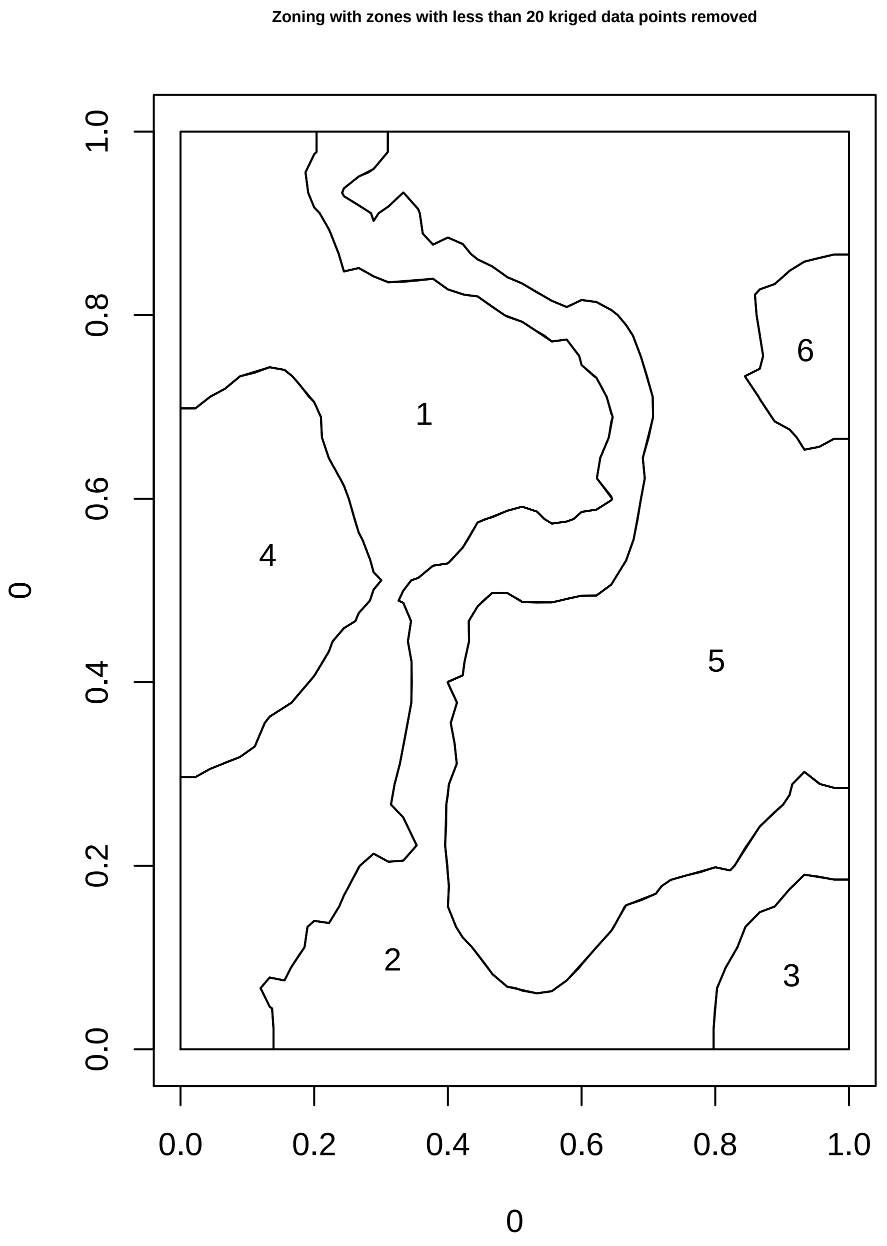
```
## [1] 3 2 3 2 1 2 3 1
```

To keep only zones with a minimum of 20 data points, remake zoning structure.

```r
K=calNei(Z,mapTest$krigData,mapTest$krigSurfVoronoi,mapTest$krigN,nmin=20)
Z=K$zonePolygone
plotZ(Z)
```

```
## NULL
```

```r
title("Zoning with zones with less than 20 kriged data points removed",cex.main = 0.5)
```



Session informations

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 14.04.5 LTS
##
## locale:
##  [1] LC_CTYPE=fr_FR.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=fr_FR.UTF-8        LC_COLLATE=fr_FR.UTF-8
##  [5] LC_MONETARY=fr_FR.UTF-8    LC_MESSAGES=fr_FR.UTF-8
##  [7] LC_PAPER=fr_FR.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=fr_FR.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] ggplot2_2.2.1            rgeos_0.3-26
##  [3] raster_2.6-7             fields_9.0
##  [5] maps_3.2.0               spam_2.1-1
##  [7] dotCall64_0.9-04         deldir_0.1-14
##  [9] maptools_0.9-2           RandomFields_3.1.50
## [11] RandomFieldsUtils_0.3.25 sp_1.2-5
## [13] gstat_1.1-5              geozoning_1.0.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.14     pillar_1.0.1     plyr_1.8.4       xts_0.10-0
##  [5] tools_3.3.2      digest_0.6.12    evaluate_0.10.1  memoise_1.1.0
##  [9] tibble_1.4.1     gtable_0.2.0     lattice_0.20-34  rlang_0.1.6
## [13] rstudioapi_0.7   commonmark_1.4   yaml_2.1.14      knitr_1.17
## [17] withr_2.1.0      stringr_1.2.0    roxygen2_6.0.1   xml2_1.1.1
## [21] devtools_1.13.4  desc_1.1.1       rprojroot_1.2    spacetime_1.2-1
## [25] R6_2.2.2         rmarkdown_1.8    foreign_0.8-67   magrittr_1.5
## [29] htmltools_0.3.6  backports_1.1.1  scales_0.5.0     intervals_0.15.1
## [33] assertthat_0.2.0 colorspace_1.3-2 stringi_1.1.6    lazyeval_0.2.1
## [37] munsell_0.4.3    crayon_1.3.4     FNN_1.1          zoo_1.8-0
```