Zoning simulated data

B. Charnomordic

2017-09-26

Contents

Step 1: generate map 1
Step 2 - generate zoning from map for a given probability vector 3
Step 3 - Generate tree of possible corrections for small zones 7

Session informations **10**

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

This vignette illustrates the zoning with corrections procedure on simulated data.

Step 1: generate map

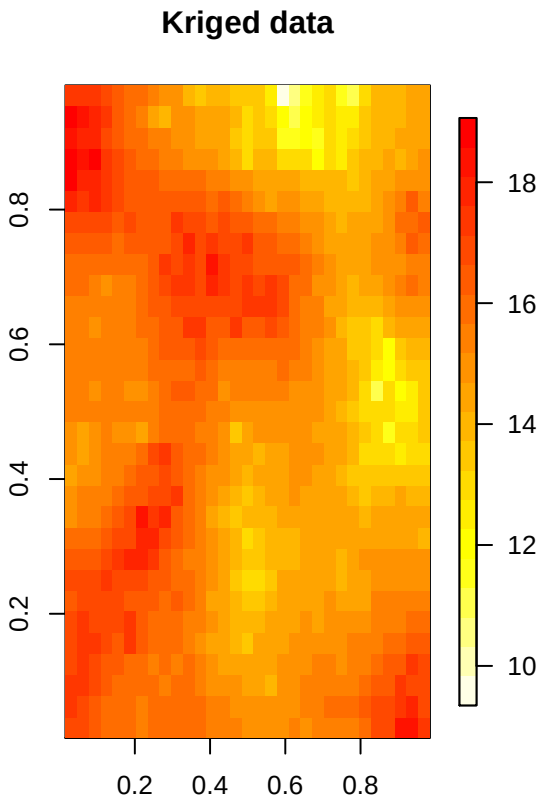
A map object is simulated with a Gaussian field and a variogram model. 450 points are randomly allocated on a square field of size 1.Then 1936 points are kriged on a regular grid using inverse distance weighted interpolation. A Delaunay tesselation yields point neighborhood in the sense of Voronoi.

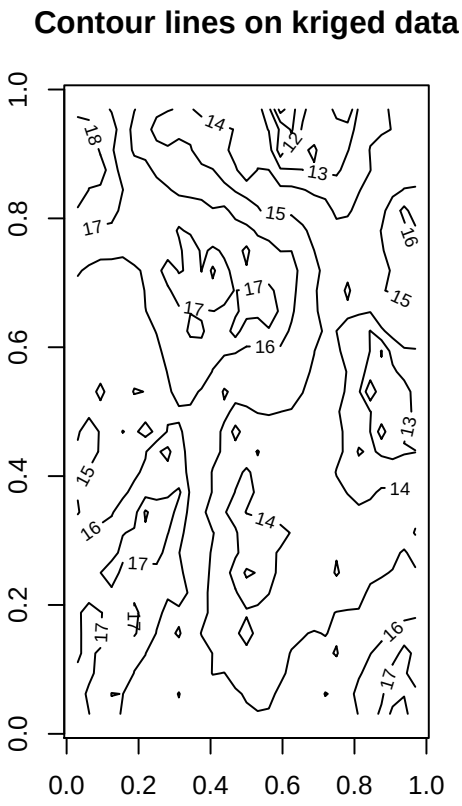
```
seed=80
map=genMap(DataObj=NULL,seed=seed,disp=FALSE,krig=2,Vmean=15)
```

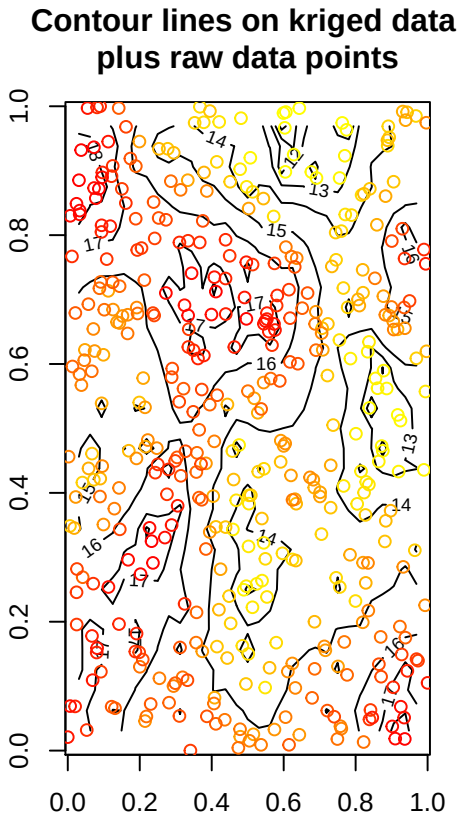
```
## [1] "DataObj=NULL, generating DataObj-seed= 80"
## [inverse distance weighted interpolation]
```

Display 2D map with three different views: first one=kriged data, second one=contour lines, third one=raw data.

```
plotMap(map)
```







The map object contains all components required for zoning generation, evaluation and visualization. The data-related

components include the kriged data, grid resolution, size in each dimension and map boundary. Kriged data are available as a SpatialPointsDataFrame object, as well as a matrix object directly usable by image functions. rawData are stored as well in the map object, for traceability purposes. The neighborhood-related components include the list of neighbor point indices for each kriged data point, as well as the areas of Voronoi polygons associated to all points. Finally the variogram-related components include the VGM model used to simulate the field, as well as the equivalent RandomFields model. “‘ Check the mean and standard deviation of generated data.

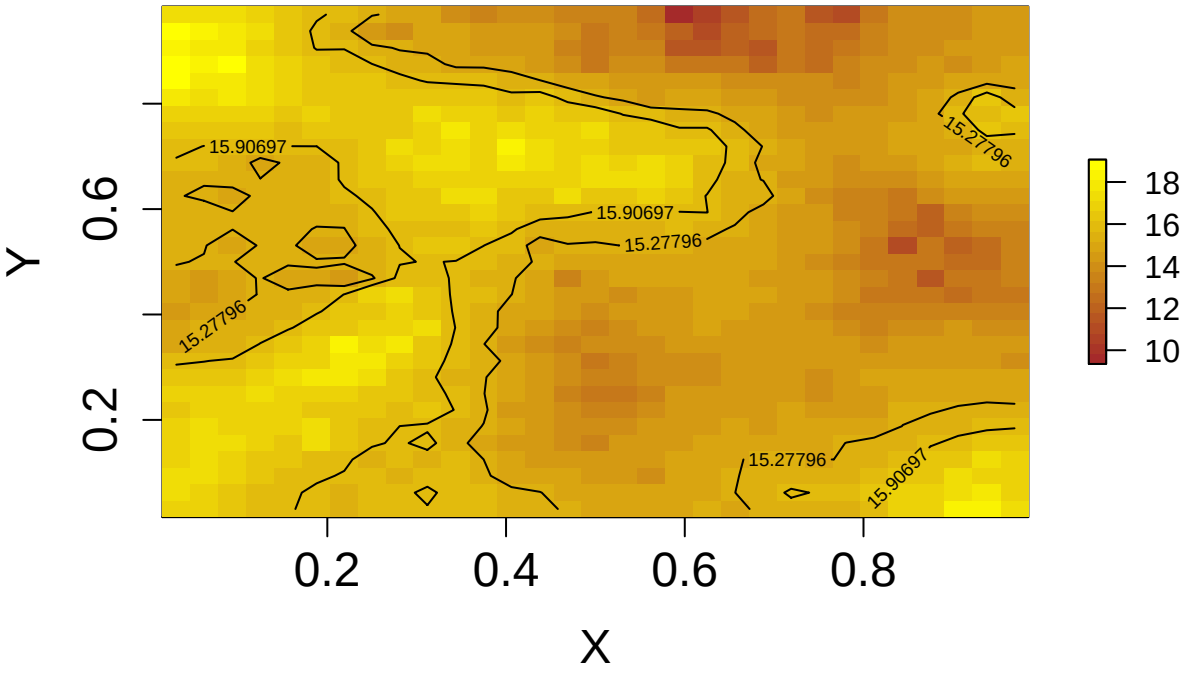
```
  meanvarSimu(map)
```

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

Step 2 - generate zoning from map for a given probability vector

Given a probability vector, a vector of values is obtained using the quantile function. Display map image with contour levels corresponding to qq values

```
qq=quantile(map$krigGrid,na.rm=TRUE,prob=c(0.5,0.7))
dispZ(map$step,map$krigGrid,valQ=qq)
```



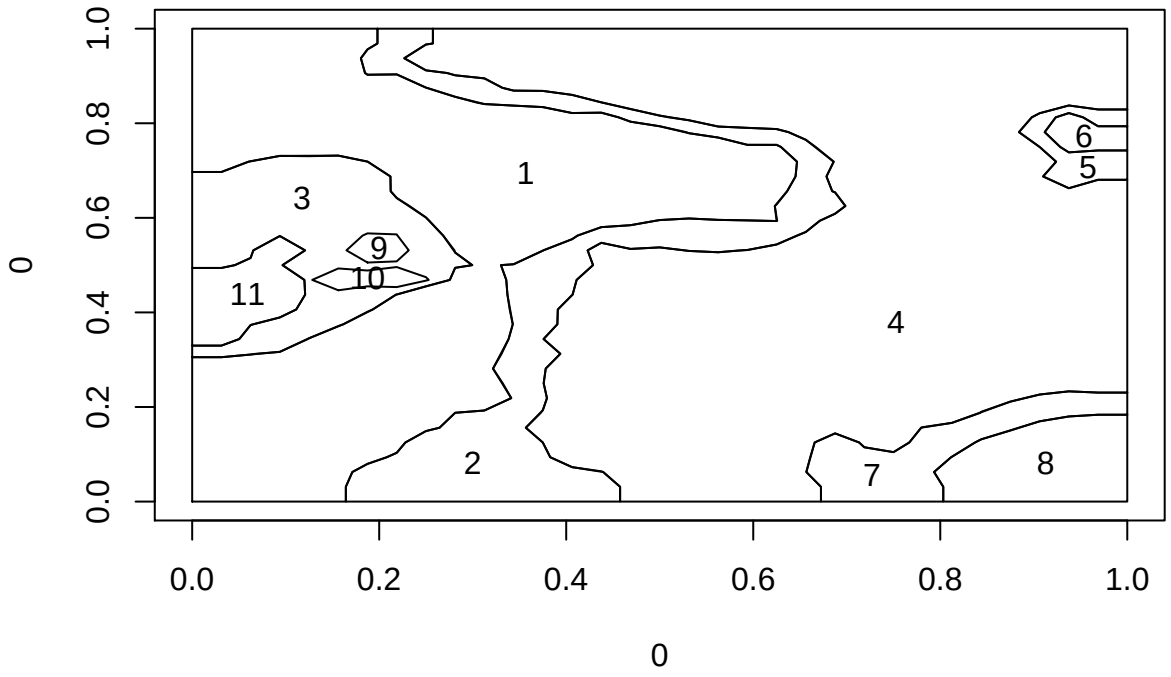

```
## NULL
```

A zoning is done on the kriged data, by computing the contour lines corresponding to the vector of values given by the probability vector, trimming them to the map boundary and defining zones corresponding to the closed contour lines. A zoning is a list of SpatialPolygons objects. It does not contain data, only polygon geometry.

```
  ZK=initialZoning(c(0.5,0.7),map) # names(ZK)  "resCrit"  "resDist" "resZ" "cL" "qProb"
```

Plot zoning (12 zones in this case)

```
  K=ZK$resZ
  Z=K$zonePolygone
  plotZ(Z)
```




```
## NULL
```

Or more detailed plot - compare it to previous plot of contour lines We see that contour lines are now extended to the map boundary in order to close zones and that contour lines impossible to close or yielding zones with 0 or 1 point are removed.

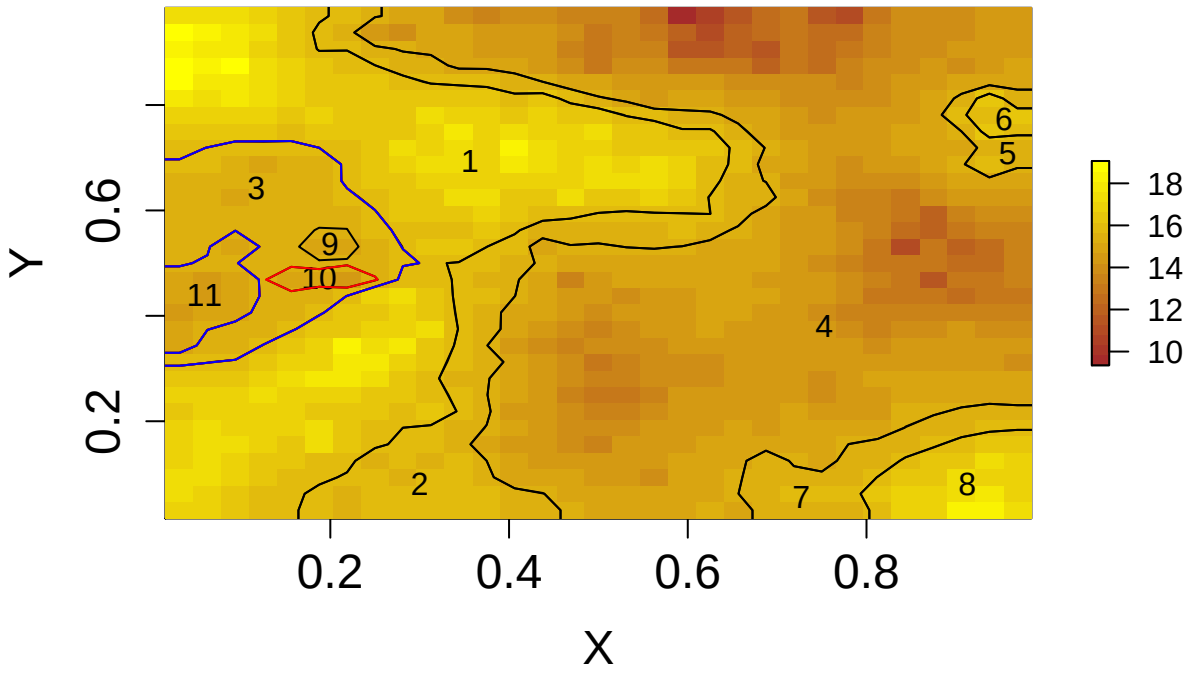
```
    dispZ(map$step,map$krigGrid,zonePolygone=Z)
```

```
## NULL
```

```
   # Outline boundaries of zone 3
   dispZ(map$step,map$krigGrid,zonePolygone=Z,iZ=3)
```

```
## NULL
```

```
   # Outline all polygons of zone 3- It contains 4 polygons
   linesSp(Z[[3]],col="blue") # first one in blue
   linesSp(Z[[3]],k=2,col="red") # second one in red, and so on
```




```
   # A zone can have one or several holes, and each hole is an independent zone.zone 3 has 3 holes (zon
   holeSp(Z[[3]])
```

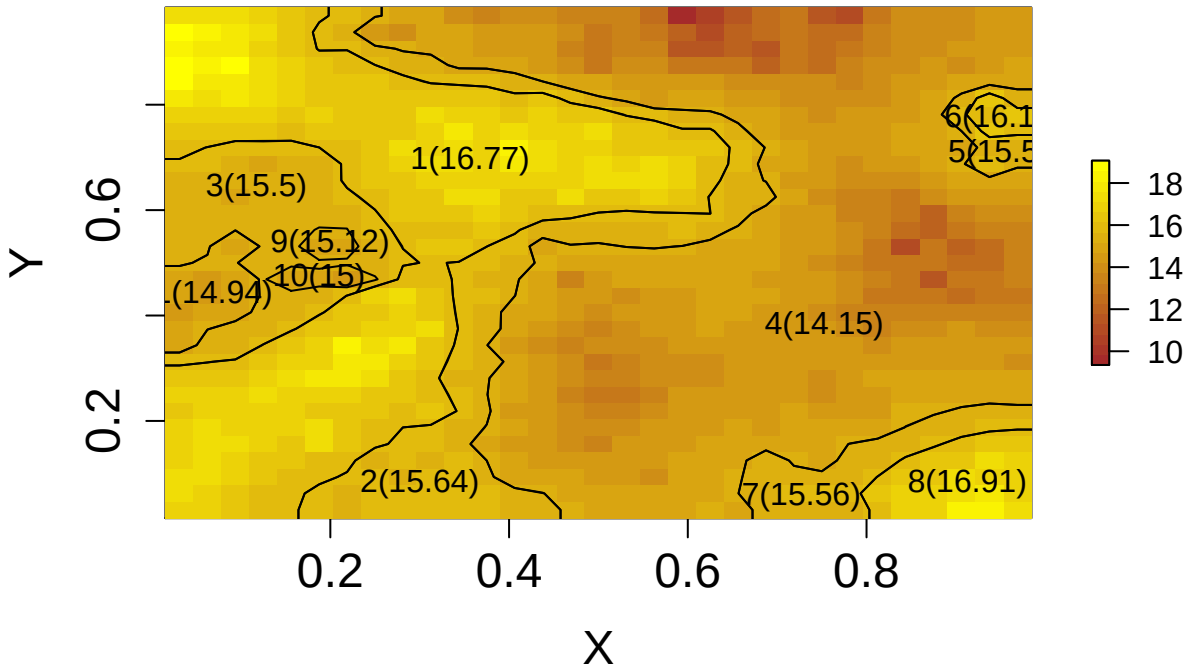
```
## [1] 2
```

A more detailed plot: each zone is labelled with its number and its mean value

```
dispZ(map$step,map$krigGrid,zonePolygone=Z,K=K,boundary=map$boundary,nbLvl=0,id=FALSE,mu=2)
```

```
## NULL
```

```
  title(paste(" q=[",toString(round(qq,2)),"]   crit=",round(ZK$resCrit,2),sep="")) # add quantile valu
```




```
  # print zone labels
  printLabZ(list(K))
```

```
## [1] "2q zone labels= c(3, 2, 2, 1, 2, 3, 2, 3, 1, 1, 1)"
```

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

```
  # print zone areas
  printZsurf(K$zonePolygone)
```

```
## [1] "iZ= 1 area= 0.26918"
## [1] "iZ= 2 area= 0.09655"
## [1] "iZ= 3 area= 0.06488"
## [1] "iZ= 4 area= 0.4698"
## [1] "iZ=5 area=0.00967 < minSize(0.012)"
## [1] "iZ=6 area=0.00506 < minSize(0.012)"
## [1] "iZ= 7 area= 0.02899"
## [1] "iZ= 8 area= 0.03089"
## [1] "iZ=9 area=0.00296 < minSize(0.012)"
## [1] "iZ=10 area=0.00386 < minSize(0.012)"
## [1] "iZ= 11 area= 0.01815"
```

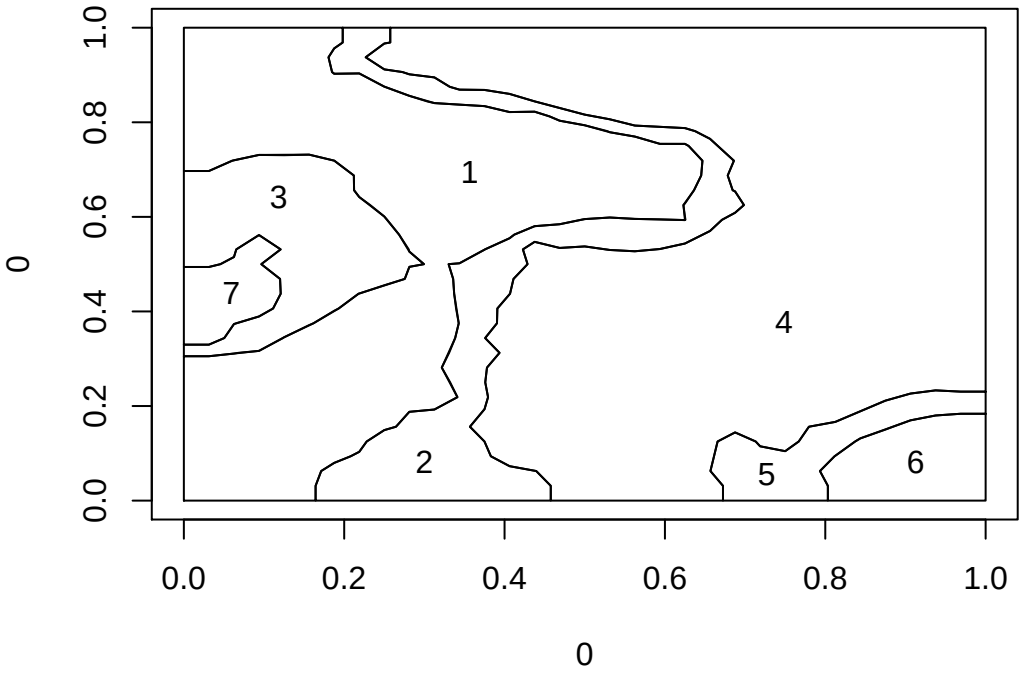
```
## [1]  9 10  6  5
```

```
  # print zone ids
  printZid(Z)
```

```
## [1] "ii= 1  ID= 1"
## [1] "ii= 2  ID= 2"
## [1] "ii= 3  ID= 3"
## [1] "ii= 4  ID= 4"
## [1] "ii= 5  ID= 5"
```

```
## [1] "ii= 6  ID= 6"
## [1] "ii= 7  ID= 7"
## [1] "ii= 8  ID= 8"
## [1] "ii= 9  ID= 9"
## [1] "ii= 10  ID= 10"
## [1] "ii= 11  ID= 11"
```

```
  # remove zones with less than 5 data points
  K=calNei(Z,map$krigData,map$krigSurfVoronoi,map$krigN,nmin=10)
  plotZ(K$zonePolygone)
```




```
## NULL
```

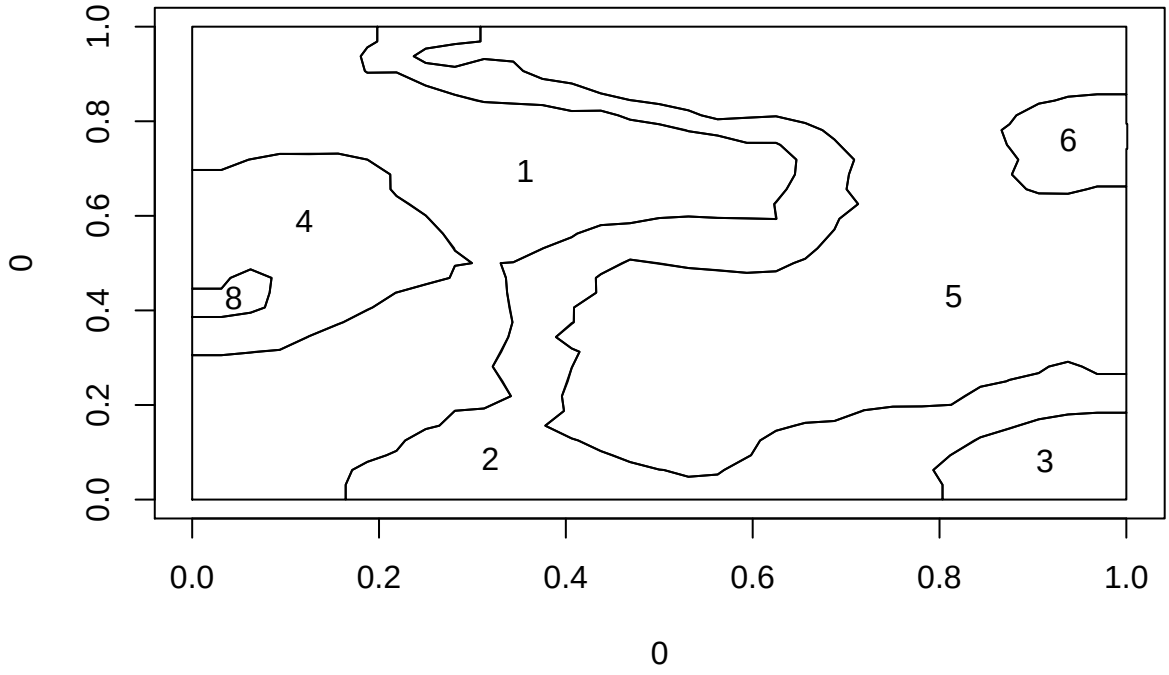
Step 3 - Generate tree of possible corrections for small zones

2 operations are done for each small zone : 1-remove zone, i.e. merge into englobing zone, 2- grow zone. Growing is done in 2 different ways depending on zone proximity to other ones. If zone is isolated (distance to other zones controlled by distIsoZ parameter), it grows bigger but remains isolated from others. Zone growing in that case is performed by finding the contour line close to the current zone contour, that mximizes the zoning quality criterion. A small value of distIsoZ ensures that zones 7 and 8 have enough space to grow. If zone is non isolated, it is joined to the closest zone with the same label.

```
criti<-correctionTree(c(0.4,0.7),map,SAVE=TRUE,ALL=TRUE,LASTPASS=FALSE,distIsoZ=0.001)#save all branche
zk=criti$zk
```

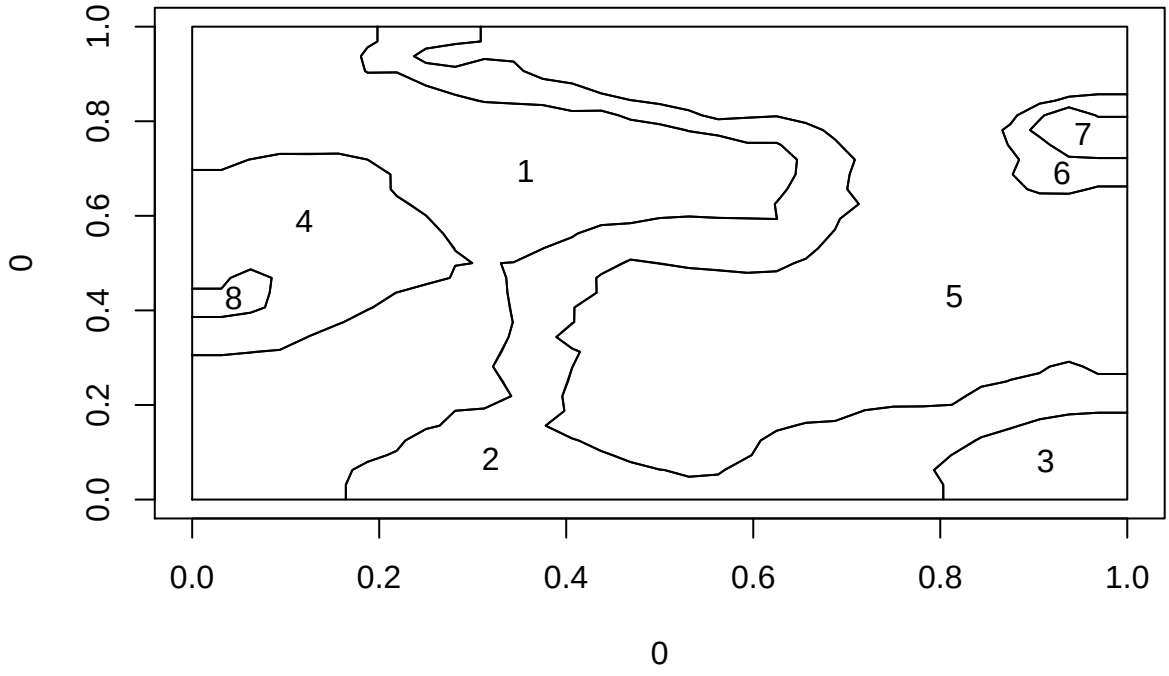
In that case we have 2 small zones (#7 and #8), hence 3 levels (level 1 is initial zoning, level 2 has 2 branches, level 3 has 4 branches. For each correction step-first branch=zone removal, second branch=zone junctioh. The procedure starts with the smallest zone, here zone #7.

```
  Z21=zk[[2]][[1]]$zonePolygone
  Z22=zk[[2]][[2]]$zonePolygone
  plotZ(Z21,id=TRUE) # result of removal of zone #7
```



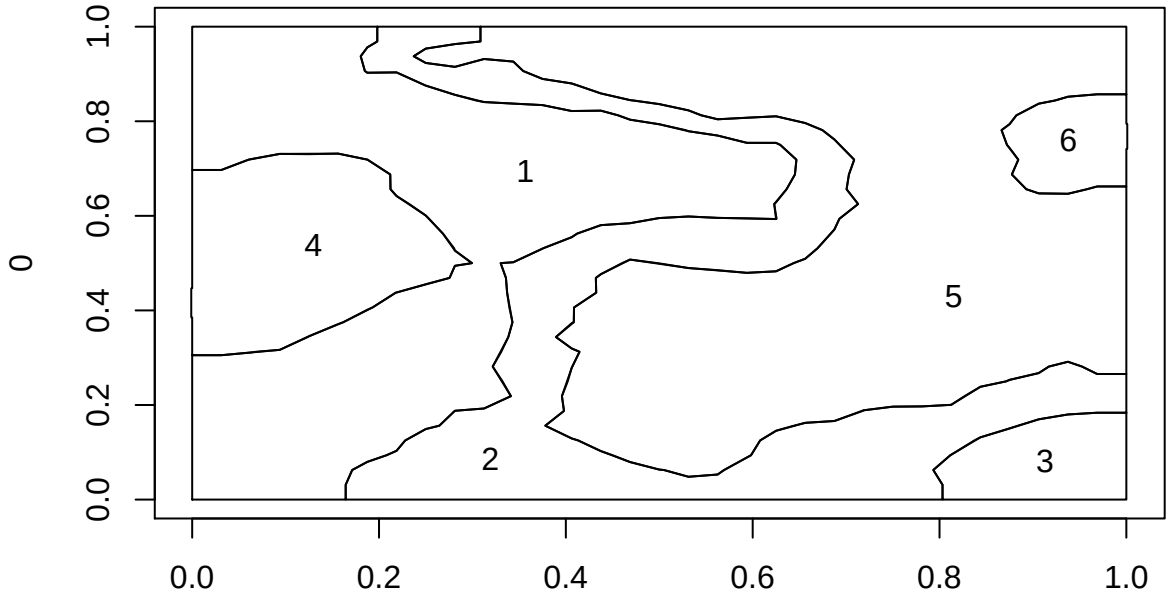

```
## NULL
```

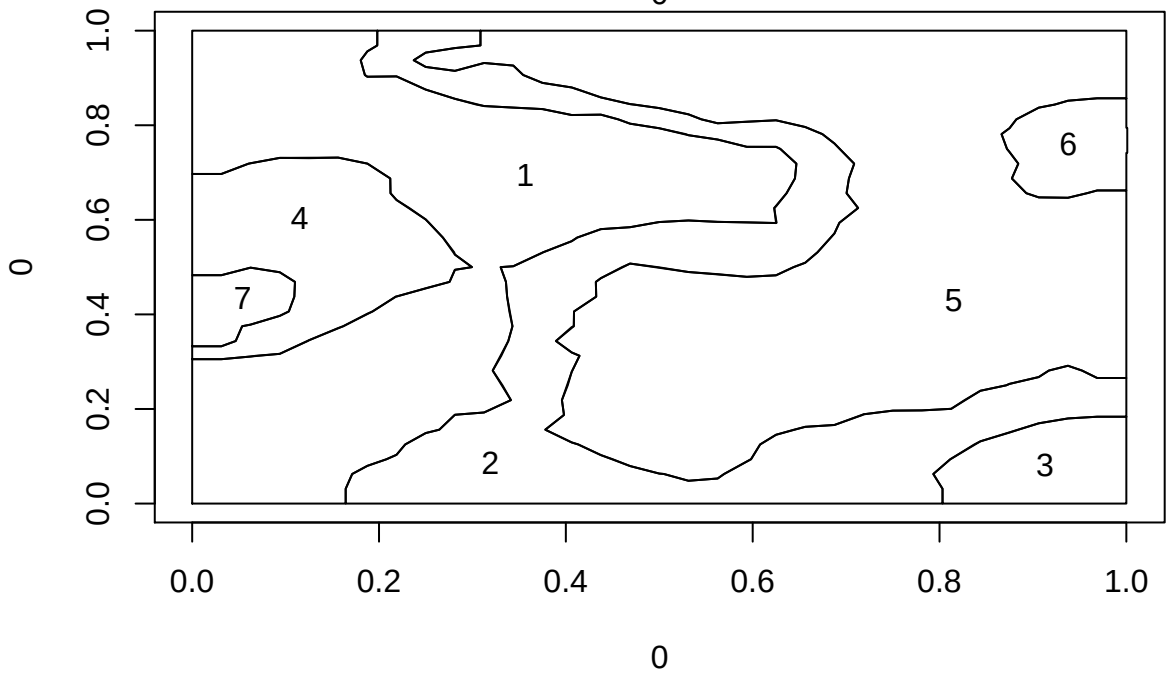
```
plotZ(Z22,id=TRUE) # result of growing of zone #7
```

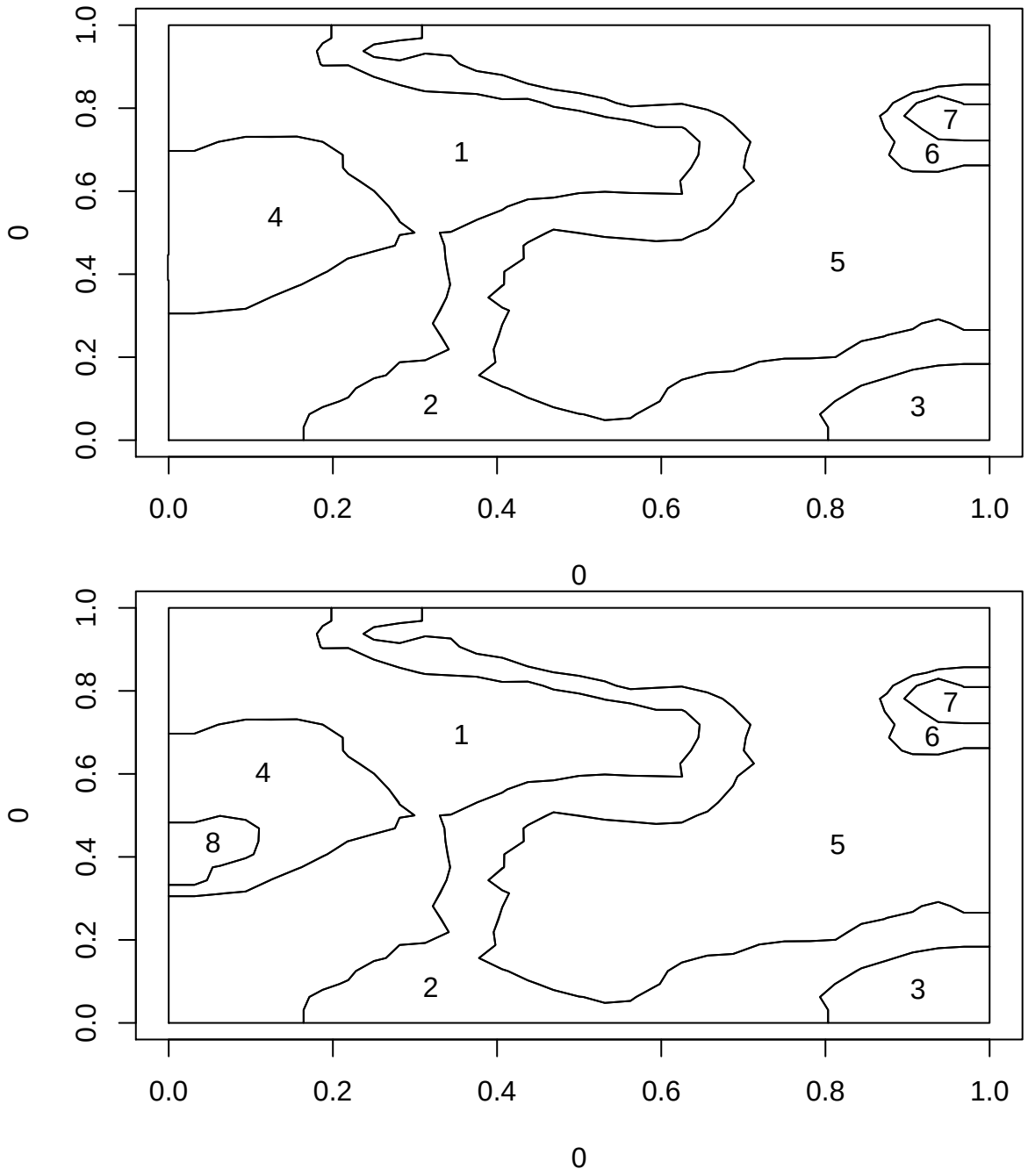



```
## NULL
```

```
# then the correction procedure is done for zone #8 in these two zonings
for (ii in 1:length(zk[[3]]))
{
plotZ(zk[[3]][[ii]]$zonePolygone) # result of removal of zone #8 in Z21
}
```


0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6


0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
7




```
#successively:  removal of zone#8  in Z21, growing of zone#8  in Z21
# removal of zone#8  in Z22, growing of zone#8  in Z22
criti<-correctionTree(c(0.4,0.7),map,SAVE=TRUE,ALL=TRUE,LASTPASS=TRUE,distIsoZ=0.001)#other try with
# other run with ALL=FALSE saves memory by keeping only the first and the last levels
criti<-correctionTree(c(0.4,0.7),map,SAVE=TRUE,ALL=FALSE,LASTPASS=FALSE,distIsoZ=0.001)
```

Session informations

```
## R version 3.4.0 (2017-04-21)
## Platform: x86_64-pc-linux-gnu (64-bit)
```

```
## Running under: Debian GNU/Linux 8 (jessie)
##
## Matrix products: default
## BLAS: /usr/lib/libblas/libblas.so.3.0
## LAPACK: /usr/lib/lapack/liblapack.so.3.0
##
## locale:
##  [1] LC_CTYPE=fr_FR.utf8       LC_NUMERIC=C
##  [3] LC_TIME=fr_FR.utf8        LC_COLLATE=C
##  [5] LC_MONETARY=fr_FR.utf8    LC_MESSAGES=fr_FR.utf8
##  [7] LC_PAPER=fr_FR.utf8       LC_NAME=C
##  [9] LC_ADDRESS=C              LC_TELEPHONE=C
## [11] LC_MEASUREMENT=fr_FR.utf8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] gstat_1.1-5     fields_8.15     maps_3.1.1      spam_1.4-0
## [5] sp_1.2-4        ggplot2_2.2.1   rgeos_0.3-23    geozoning_1.0.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.11             compiler_3.4.0
##  [3] plyr_1.8.4               tools_3.4.0
##  [5] xts_0.9-7                digest_0.6.12
##  [7] evaluate_0.10.1          tibble_1.3.1
##  [9] gtable_0.2.0             lattice_0.20-35
## [11] rlang_0.1.1              yaml_2.1.14
## [13] stringr_1.2.0            knitr_1.17
## [15] raster_2.5-8             RandomFieldsUtils_0.3.25
## [17] rprojroot_1.2            spacetime_1.2-0
## [19] foreign_0.8-68           rmarkdown_1.6
## [21] deldir_0.1-14            magrittr_1.5
## [23] backports_1.1.0          scales_0.4.1
## [25] htmltools_0.3.6          intervals_0.15.1
## [27] RandomFields_3.1.50      maptools_0.9-2
## [29] colorspace_1.3-2         labeling_0.3
## [31] stringi_1.1.5            lazyeval_0.2.0
## [33] munsell_0.4.3            FNN_1.1
## [35] zoo_1.8-0
```