Spatial Fuzzy CMean with R

Jeremy Gelb

02/04/2021

Contents

Introduction

This document is a short introduction to the package **geocmeans**. It implements a fuzzy classification method bringing spatial information and neighbouring in its calculation

In their article, Cai, Chen, and Zhang (2007) described the method, originally applied in the analysis of brain imagery. The generalized version of the spatial fuzzy c-means is presented by Zhao, Jiao, and Liu (2013).

Gelb and Apparicio (2021) applied the method to socio-residential and environmental data comparing the results with other unsupervised classification algorithms (in French).

There are actually numerous packages and functions to perform unsupervised classification in R (*hclust*, *kmeans*, *cmeans*, *factoextra*, etc.). However, these methods are not always well-suited to analyze spatial data. Indeed, they do not account for spatial information such as proximity or contiguity between observations. This may lead to solutions for which close observations end up in different groups event though they are very similar.

To our knowledge, the package **ClustGeo** is the only package proposing an unsupervised classification method which directly consider spatial proximity between observations. The proposed approach is appealing because the user can select a parameter (*alpha*) that controls the weight of the spatial distance matrix (calculated between observations with their locations) versus the semantic distance matrix (calculated between observations with their variables).

However, this method belongs to the category of "hard-clustering" algorithms. Each observation ends up in one cluster/group. The main draw-back here is the difficulty to identify observations that are undecided, at the frontier of two clusters/groups. The soft or fuzzy clustering algorithms provide more information because they calculate the "probability" of each observation to belong to each group.

The algorithms SFCM (spatial fuzzy c-means) and SGFCM (spatial generalized fuzzy c-means) propose to combine the best of both worlds.

The package **geocmeans** is an implementation in R of these methods (originally developed for analysis of brain imagery). It comes with a set of functions to facilitate the analysis of the final membership matrices :

- calculating many quality indices (coming mainly from the package *fclust*)
- mapping the results

- giving summary statistics for each cluster/group.

Loading the packages and the data

geocmeans comes with a toy dataset *LyonIris*, combining many demographic and environmental variables aggregated at the scale of the Iris (aggregated units for statistical information) in Lyon (France).

Before starting the analysis, the data must be standardized because most of the calculus is based on Euclidean distance (we plan to also include Manhattan distance in a future release).

```
#charging packages and data
library(geocmeans)
library(ggplot2)
library(ggpubr)
library(dplyr)
library(viridis)
library(spdep)

data(LyonIris)

#selecting the columns for the analysis
AnalysisFields <-c("Lden","NO2","PM25","VegHautPrt","Pct0_14",
                   "Pct_65","Pct_Img","TxChom1564","Pct_brevet","NivVieMed")

#rescaling the columns
Data <- LyonIris@data[AnalysisFields]
for (Col in names(Data)){
  Data[[Col]] <- scale(Data[[Col]])
}

#preparing some elements for further mapping
LyonIris$OID <- as.character(1:nrow(LyonIris))
FortiData <- broom::tidy(LyonIris,region="OID")
```

Classical Kmeans

To explore the dataset and choose the right number of cluster/groups (k) we propose to start with a classical kmeans.

```
#finding the best k by using the r2 of the classification
#trying for k from 2 to 10
R2s <- sapply(2:10,function(k){
  Clust <- kmeans(Data,centers=k,iter.max = 150)
  R2 <- Clust$betweenss / Clust$totss
  return(R2)
})

Df <- data.frame(K=2:10,
                 R2 = R2s)

ggplot(Df)+
  geom_line(aes(x=K,y=R2s))+
  geom_point(aes(x=K,y=R2s),color="red")+
  xlab("Number of groups")+
  ylab("R2 of classification")
```

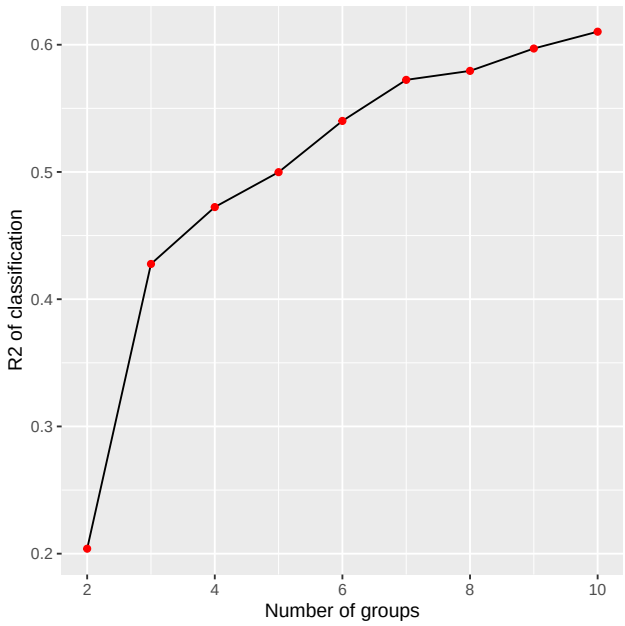


Figure 1: Impact of the number of groups on the explained variance

By plotting the R-squared of the kmeans classification for each k between 2 and 10, we can see a first elbow at k=3. But this small number of groups leads to a classification explaining only 43% of the original data variance. We decide to keep k=4 to have one more group in the analysis.

Let us map the obtained groups.

```
KMeanClust <-  kmeans(Data,centers=4,iter.max = 150)
LyonIris$Cluster <-paste("cluster",KMeanClust$cluster,sep="_")

#mapping the groups
DFmapping <- merge(FortiData,LyonIris,by.x="id",by.y="OID")

ggplot(data=DFmapping)+
  geom_polygon(aes(x=long,y=lat,group=group,fill=Cluster),color=rgb(0,0,0,0))+
  coord_fixed(ratio = 1)+
  scale_fill_manual(name="Cluster",values = c("cluster_1"="palegreen3",
                                "cluster_2"="firebrick",
                                "cluster_3"="lightyellow2",
                                "cluster_4"="steelblue"))+
  theme( axis.title = ggplot2::element_blank(),
    axis.text = ggplot2::element_blank(),
    axis.ticks = ggplot2::element_blank()
    )
```

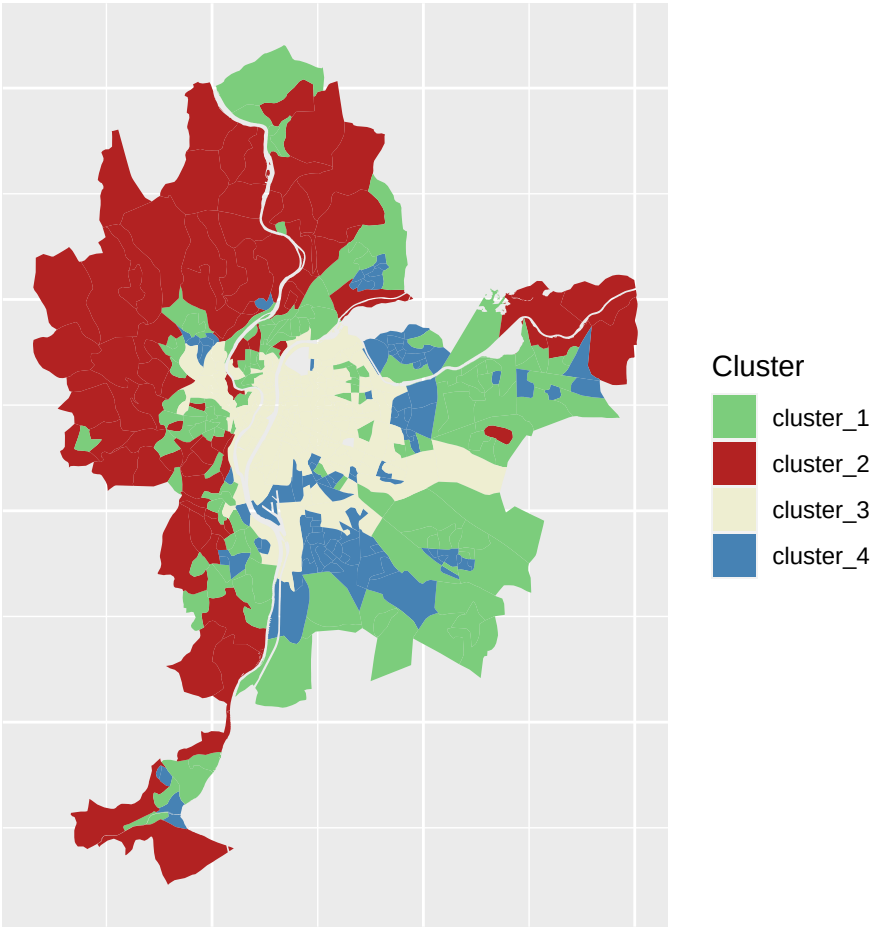


Figure 2: Clusters identified by classical k-means

We can clearly distinguish 4 strong spatial structures, but with some mixing between the clusters.

We could now compare this solution with a classical c-means algorithm.

Classical c-means and generalized c-means

The classical c-means is a simple method to perform fuzzy unsupervised classification. The package **geocmeans** proposes the function *CMeans*. We set the fuzziness degree (m) to 1.5.

We do not present the algorithm here, but only the two main formulas.

The first one is used to update the values of the membership matrix at each iteration u_ik

$$u_{ik} = \frac{(||x_k - v_i||^2)^{(-1/(m-1))}}{\sum_{j=1}^{c}(||x_k - v_j||^2)^{(-1/(m-1))}}$$

And the second one to update the centers of the clusters

$$v_i = \frac{\sum_{k=1}^{N} u_{ik}^m(x_k)}{\sum_{k=1}^{N} u_{ik}^m}$$

With :

- x_k the values of the observation k
- v_i the values of the center of the cluster i
- c the number of clusters
- m the fuzziness index

```
Cmean <- CMeans(Data,4,1.5,500,standardize = FALSE, seed = 456, tol = 0.00001, verbose = FALSE)
```

We can now use the function *calcqualityIndexes* which combines many indices from the package **fclust** to analyze the quality of the classification. We will use these values later for the purpose of comparisons among the different algorithms.

```
calcqualityIndexes(Data, Cmean$Belongings, m = 1.5)
```

```
## $Silhouette.index
## [1] 0.3628511
##
## $Partition.entropy
## [1] 0.7320123
##
## $Partition.coeff
## [1] 0.6070551
##
## $Modified.partition.coeff
## [1] 0.4760735
##
## $XieBeni.index
## [1] 1.352272
##
## $FukuyamaSugeno.index
## [1] 910.6503
##
## $Explained.inertia
## [1] 0.3018186
```

Note, **geocmeans** also proposes a so-called generalized version of the c-means algorithm. It is known to accelerate convergence and yield less fuzzy results by adjusting the membership matrix at each iteration. It requires an extra *beta* parameter controlling the strength of the modification. The modification only affects the formula updating the membership matrix.

$$u_{ik} = \frac{(||x_k - v_i||^2 - \beta_k)^{(-1/(m-1))}}{\sum_{j=1}^{c}(||x_k - v_j||^2 - \beta_k)^{(-1/(m-1))}}$$

with $\beta_k = min(||x_k - v||^2)$ and $0 \leq \beta \leq 1$.

To select an appropriate value for this parameter, we will try all the possible values between 0 and 1 with a step of 0.05.

```
beta_values <- selectParameters("GFCM",data = Data, k = 4, m = 1.5,
                                beta = seq(0,1,0.05), spconsist = F,
                                tol = 0.00001, seed = 456)
```

```
knitr::kable(beta_values[c("beta","Silhouette.index","XieBeni.index","Explained.inertia")],
             col.names = c("beta", "silhouette index",
                           "Xie and Beni index", "explained inertia"),digits = 3)
```

beta	silhouette index	Xie and Beni index	explained inertia
0.00	0.363	1.352	0.302
0.05	0.365	1.268	0.310
0.10	0.366	1.193	0.319
0.15	0.365	1.128	0.327
0.20	0.361	1.071	0.336
0.25	0.360	1.020	0.345
0.30	0.363	0.984	0.354
0.35	0.361	0.963	0.363
0.40	0.370	0.943	0.372
0.45	0.369	0.924	0.381
0.50	0.366	0.907	0.390
0.55	0.365	0.891	0.399
0.60	0.360	0.876	0.409
0.65	0.358	0.863	0.418
0.70	0.358	0.852	0.428
0.75	0.353	0.843	0.437
0.80	0.347	0.838	0.447
0.85	0.340	0.837	0.456
0.90	0.332	0.844	0.464
0.95	0.323	0.861	0.471
1.00	0.317	0.919	0.475

Considering the table above, we select $beta = 0.7$, it maintains a good silhouette index, and increases Xie and Beni index and explained inertia. Let us compare the results of GFCM and FCM.

```
GCmean <- GCMeans(Data,k = 4,m = 1.5, beta = 0.7,500,standardize = FALSE, seed=456,
                  tol = 0.00001, verbose = FALSE)
r1 <- calcqualityIndexes(Data,GCmean$Belongings,m=1.5)
r2 <- calcqualityIndexes(Data,Cmean$Belongings,m=1.5)
df <- cbind(unlist(r1), unlist(r2))

knitr::kable(df,
             digits = 3,col.names = c("GFCM", "FCM"))
```

	GFCM	FCM
Silhouette.index	0.358	0.363
Partition.entropy	0.323	0.732
Partition.coeff	0.833	0.607
Modified.partition.coeff	0.778	0.476
XieBeni.index	0.852	1.352
FukuyamaSugeno.index	219.641	910.650
Explained.inertia	0.428	0.302

The results indicate that the GFCM provides a solution that is less fuzzy (higher explained inertia and lower partition entropy) but keeps a good silhouette index and an even better Xie and Beni index.

We can now map the two membership matrices and the most likely group for each observation. To do so, we use the function *mapClusters* from **geocmeans**. We propose here to define a threshold of 0.45. If an observation only has values below this probability in a membership matrix, it will be labeled as "undecided" (represented with transparency on the map).

We can compare the maps of the classical c-means and the generalized version.

```
cmeansMaps<- mapClusters(LyonIris,Cmean$Belongings,undecided = 0.45)
GcmeansMaps<- mapClusters(LyonIris,GCmean$Belongings,undecided = 0.45)

ggarrange(cmeansMaps$ProbaMaps[[1]],GcmeansMaps$ProbaMaps[[1]],
          nrow = 1, ncol = 2, common.legend = TRUE, legend = "bottom")
```

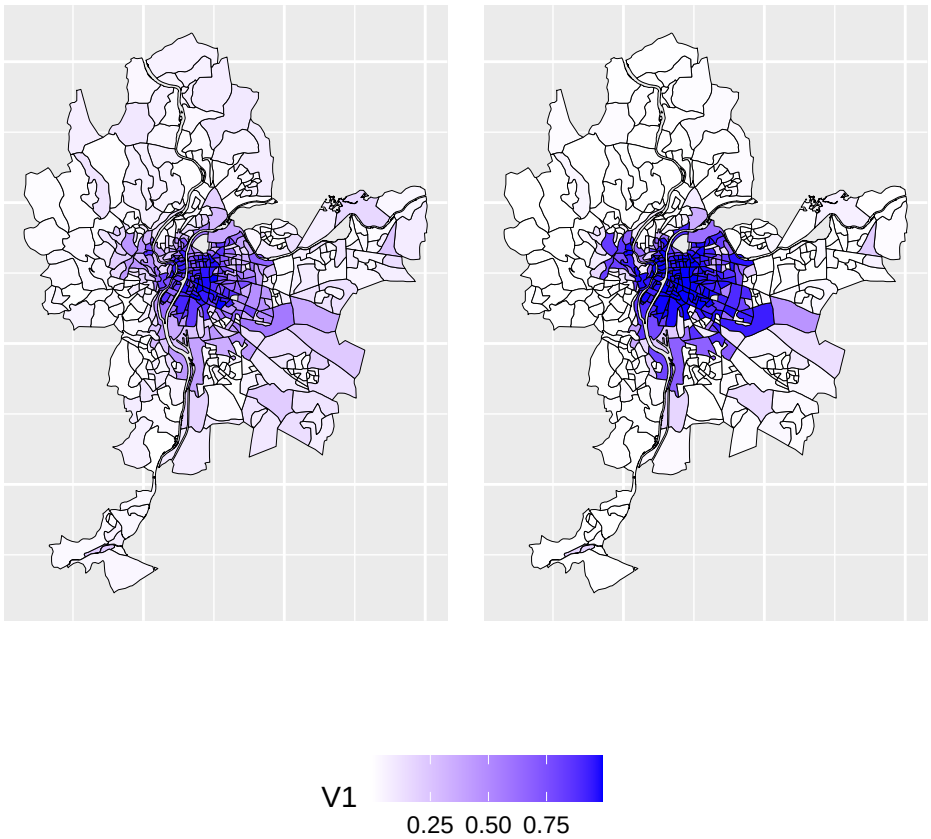


Figure 3: Probability of belonging to cluster 1

```
ggarrange(cmeansMaps$ProbaMaps[[2]],GcmeansMaps$ProbaMaps[[2]],
          nrow = 1, ncol = 2, common.legend = TRUE, legend = "bottom")
```

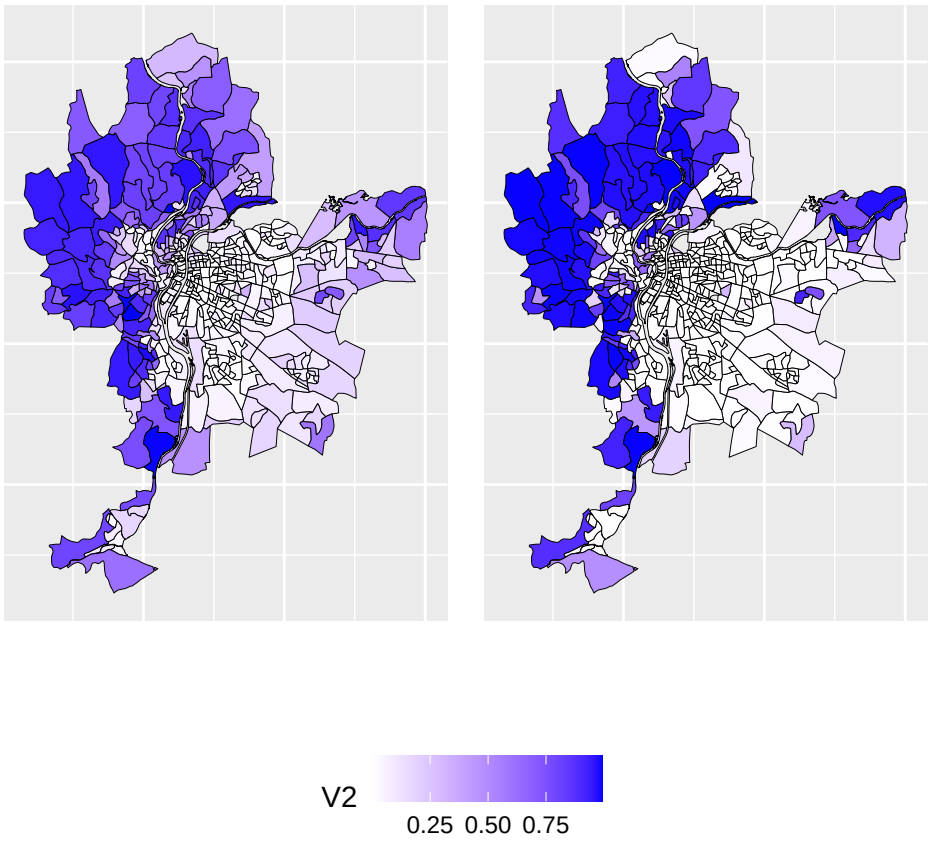


Figure 4: Probability of belonging to cluster 2

```
ggarrange(cmeansMaps$ProbaMaps[[3]],GcmeansMaps$ProbaMaps[[3]],
          nrow = 1, ncol = 2, common.legend = TRUE, legend = "bottom")
```

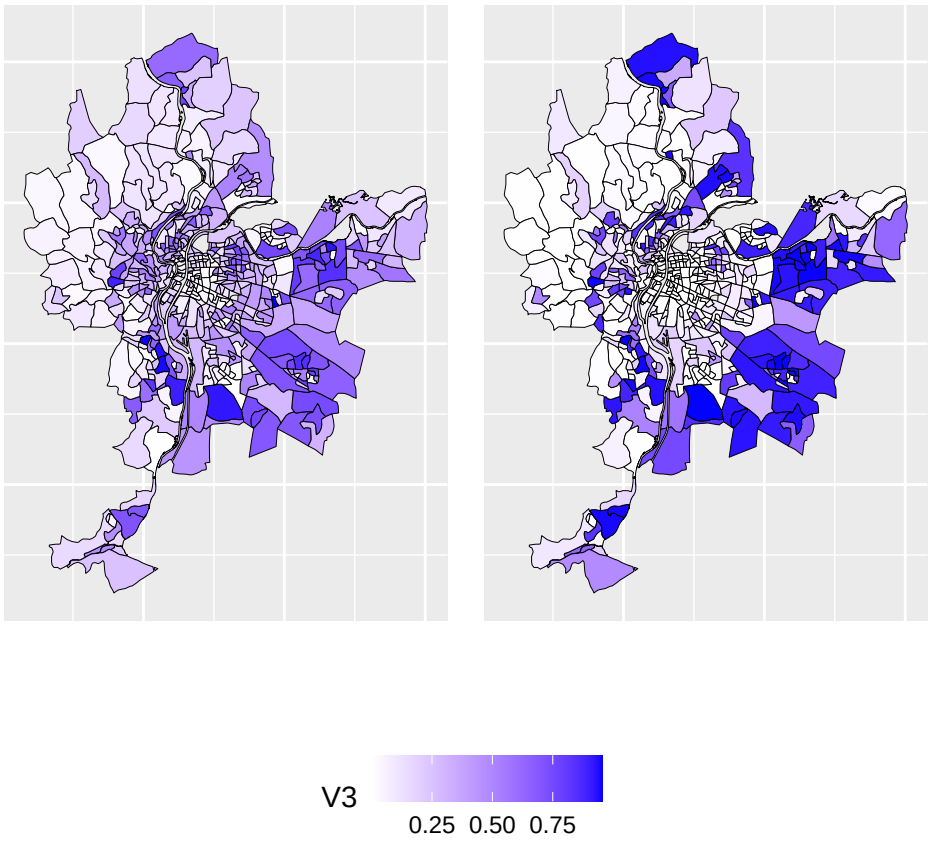


Figure 5: Probability of belonging to cluster 3

```
ggarrange(cmeansMaps$ProbaMaps[[4]],GcmeansMaps$ProbaMaps[[4]],
          nrow = 1, ncol = 2, common.legend = TRUE, legend = "bottom")
```

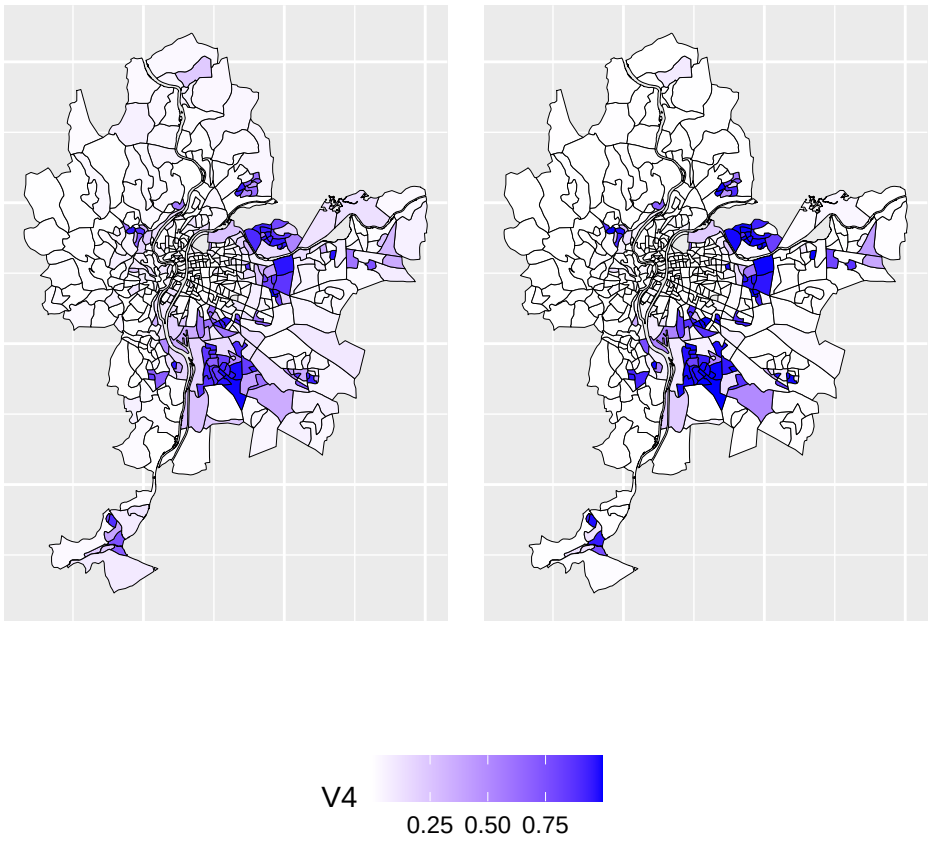


Figure 6: Probability of belonging to cluster 4

```
ggarrange(cmeansMaps$ClusterPlot,GcmeansMaps$ClusterPlot,
          nrow = 1, ncol = 2, common.legend = TRUE, legend = "bottom")
```

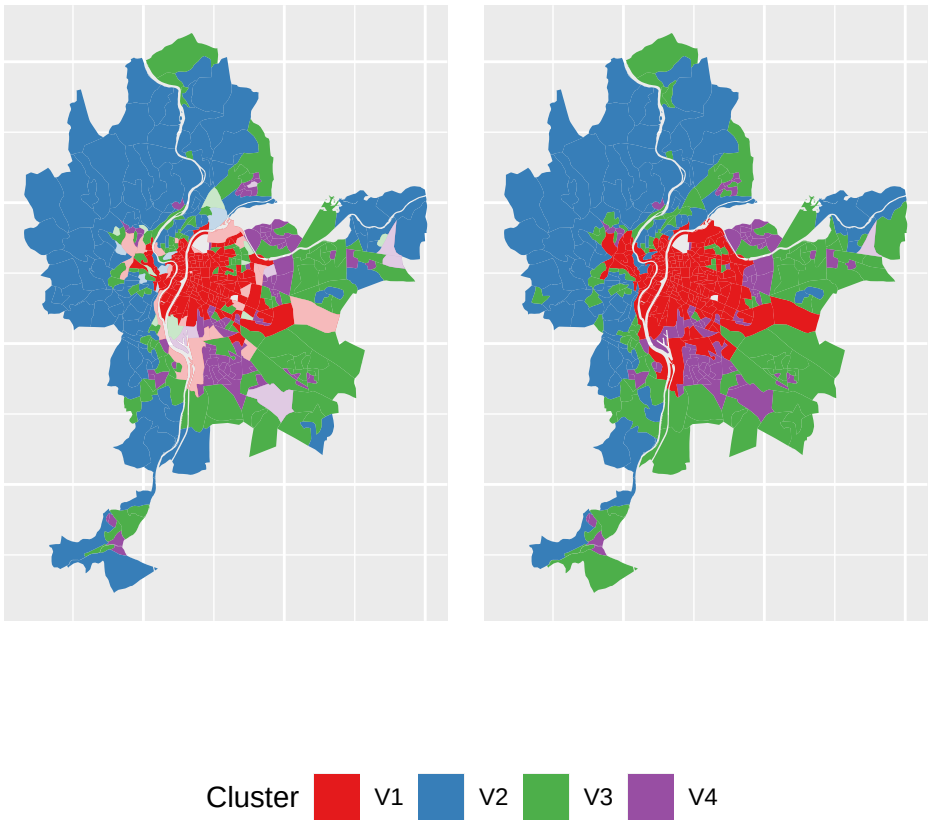


Figure 7: Most likely clusters and undecided units

As expected, the results are very similar, but the generalized version provides a more clear-cut classification.

Spatial c-means and generalized c-means

Selecting *alpha* for SFCM

Now we can use the *SFCM* function to perform a spatial c-means. The first step is to define a spatial weight matrix indicating which observations are neighbours and the strength of their relationship. We propose here to use a basic queen neighbour matrix (built with **spdep**). The matrix must be row-standardized to ensure that the interpretation of all the parameters remains clear.

```
library(spdep)

Neighbours <- poly2nb(LyonIris,queen = TRUE)
WMat <- nb2listw(Neighbours,style="W",zero.policy = TRUE)
```

The main challenge with the SFCM method is to select the parameter *alpha*. It represents the weight of the spatial dimension (lagged values) in the calculus of the membership matrix and the cluster centers.

- If alpha=0, then we end up with a classical c-means algorithm.
- If alpha=1, then the original and the lagged values have the same weight
- If alpha=2, then the lagged values are twice more important than the original values
- end so on...

The two following formulas show how the functions updating the membership matrix and the centers of the clusters are modified.

$$u_{ik} = \frac{(||x_k - v_i||^2 + \alpha||\bar{x_k} - v_i||^2)^{(-1/(m-1))}}{\sum_{j=1}^{c}(||x_k - v_j||^2 + \alpha||\bar{x_k} - v_j||^2)^{(-1/(m-1))}}$$

$$v_i = \frac{\sum_{k=1}^{N} u_{ik}^m (x_k + \alpha\bar{x_k})}{(1+\alpha)\sum_{k=1}^{N} u_{ik}^m}$$

with $\bar{x}$ the lagged version of x and $\alpha \geq 0$

As the formula suggests, the *SFCM* can be seen as a spatially smoothed version of the classical c-means and *alpha* controls the degree of spatial smoothness. This smoothing can be interpreted as an attempt to reduce spatial overfitting of the classical c-means.

To select *alpha*, we propose to check all possible values between 0 and 2 with a step of 0.05.

```
DFindices_SFCM <- selectParameters(algo = "SFCM", data = Data,
                                   k = 4, m = 1.5, alpha = seq(0,2,0.05),
                                   nblistw = WMat, standardize = FALSE,
                                   tol = 0.0001, verbose = FALSE, seed = 456)
```

Now we are able to check the indices to select the best *alpha*. The goal is to reduce spatial inconsistency as much as possible and to maintain a good classification quality.

Let us start with the spatial inconsistency. This indicator (developed for this package) calculates the sum of the squared differences between each observation and its neighbours on the membership matrix. Thus, the maximum for each observation is $k * j$ with j the number of neighbours for the observation and k the number of groups. A maximum is reached if each observation has 100% chance belonging to a cluster that is different from all its neighbours. So, when we sum up the values obtained for all the observations, we obtain a quantity of spatial inconsistency. This quantity is divided by the quantity obtained when randomly permuting the rows of the membership matrix. This second quantity represents the spatial inconsistency that we might expect if the observations were randomly scattered in space. We can repeat the permutation step (Monte Carlo approach) and keep the mean of the ratios to have a more robust indicator (see help(spConsistency) for details).

A smaller value indicates a smaller spatial inconsistency and thus a greater spatial consistency. 0 meaning that all observations have exactly the same values in the membership matrix as their neighbours (perfect spatial consistency).

```
ggplot(DFindices_SFCM)+
  geom_smooth(aes(x=alpha,y=spConsistency), color = "black")+
  geom_point(aes(x=alpha,y=spConsistency), color = "red")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

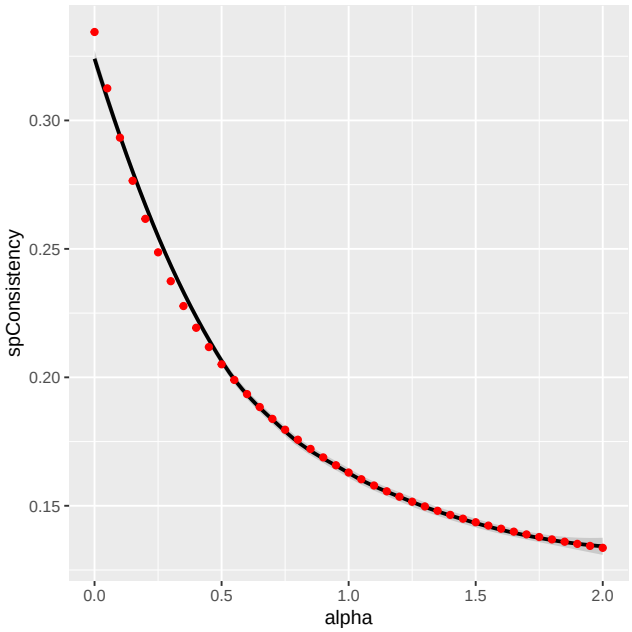


Figure 8: Link between alpha and spatial inconsistency

Not surprisingly, increasing *alpha* leads to a decrease of the spatial inconsistency. This gain follows an inverse function.

Let us now check the explained inertia

```
ggplot(DFindices_SFCM)+
  geom_smooth(aes(x=alpha,y=Explained.inertia), color = "black")+
  geom_point(aes(x=alpha,y=Explained.inertia), color = "red")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

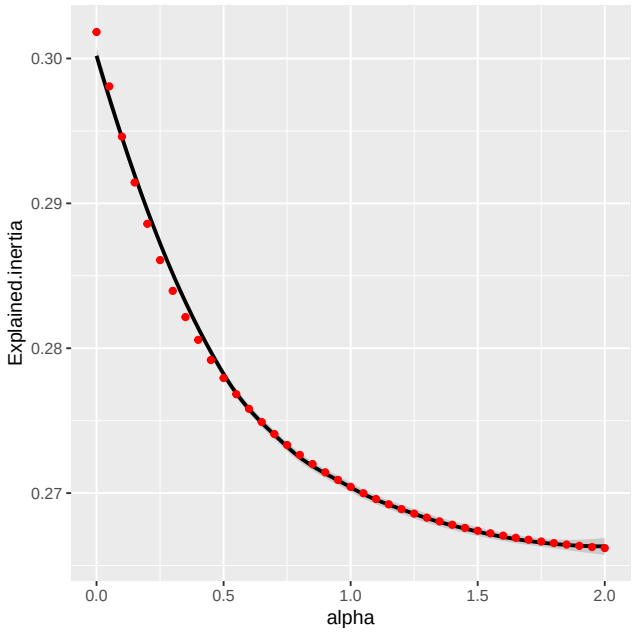


Figure 9: Link between alpha and explained inertia

As expected, the explained inertia decreases when alpha increases and again follows an inverse function. The classification has to find a compromise between the original values and the lagged values. However, the loss is very small here: only 3% between alpha = 0 and alpha = 2.

To finish here, we can observe the silhouette and Xie and Beni indicators.

```
ggplot(DFindices_SFCM)+
  geom_smooth(aes(x=alpha,y=Silhouette.index), color = "black")+
  geom_point(aes(x=alpha,y=Silhouette.index), color = "red")
```

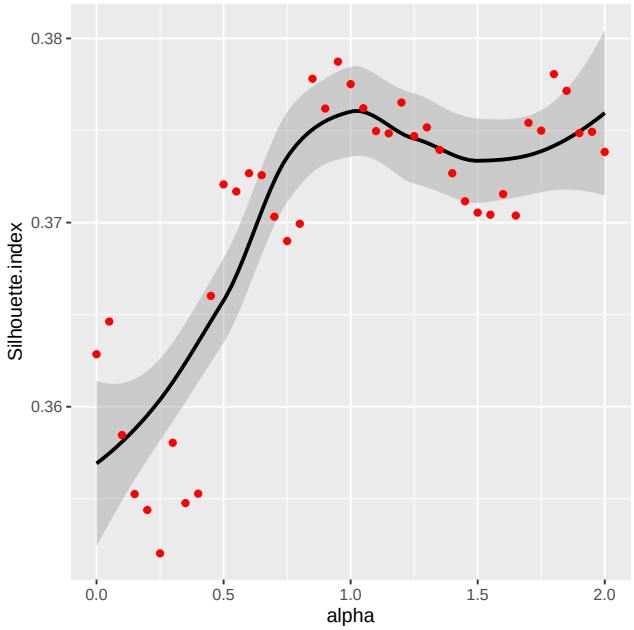


Figure 10: Link between alpha and silhouette index

```
ggplot(DFindices_SFCM)+
  geom_smooth(aes(x=alpha,y=XieBeni.index), color = "black")+
  geom_point(aes(x=alpha,y=XieBeni.index), color = "red")
```

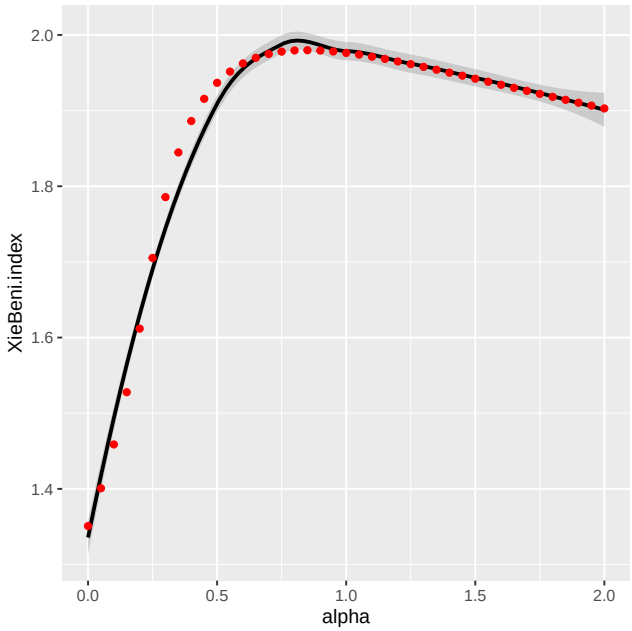


Figure 11: Link between alpha and Xie and Beni index

The detail of the meaning of these indicators is beyond the scope of this vignette. Let us just stress that a larger silhouette index indicates a better classification, and a smaller Xie and beni index indicates a better classification.

After considering all the previous charts, we decide to keep alpha = 0.7 as it seems to provide a good balance between spatial consistency and classification quality in this case.

```
SFCM <- SFCMeans(Data, WMat, k = 4, m = 1.5, alpha = 0.7,
                 tol = 0.0001, standardize = FALSE,
                 verbose = FALSE, seed = 456)
```

Selecting alpha and beta for SGFCM

It is also possible to use the so-called generalized version of the spatial c-means. In that case, we must define both *alpha* and *beta*.

The next formula shows how the membership matrix is updated at each iteration. Note that the centres of the clusters are updated with the same formula as SFCM.

$$u_{ik} = \frac{(||x_k - v_i||^2 - \beta_k + \alpha||\bar{x_k} - v_i||^2)^{(-1/(m-1))}}{\sum_{j=1}^{c}(||x_k - v_j||^2 - \beta_k + \alpha||\bar{x_k} - v_j||^2)^{(-1/(m-1))}}$$

Because we select a high resolution for our grid search of *alpha* and *beta*, we will use a multiprocessing approach.

```
future::plan(future::multiprocess(workers=4))
DFindices_SFGCM <- selectParameters.mc(algo = "SGFCM", data = Data,
                                k = 4, m = 1.5, alpha = seq(0,2,0.05),
```

```
                                beta = seq(0,0.85,0.05),
                                nblistw = WMat, standardize = FALSE, chunk_size = 50,
                                tol = 0.0001, verbose = FALSE, seed = 456)
```

```
ggplot(DFindices_SFGCM) +
  geom_raster(aes(x = alpha, y = beta, fill = Silhouette.index),  size = 5) +
  scale_fill_viridis() +
  coord_fixed(ratio=1)
```

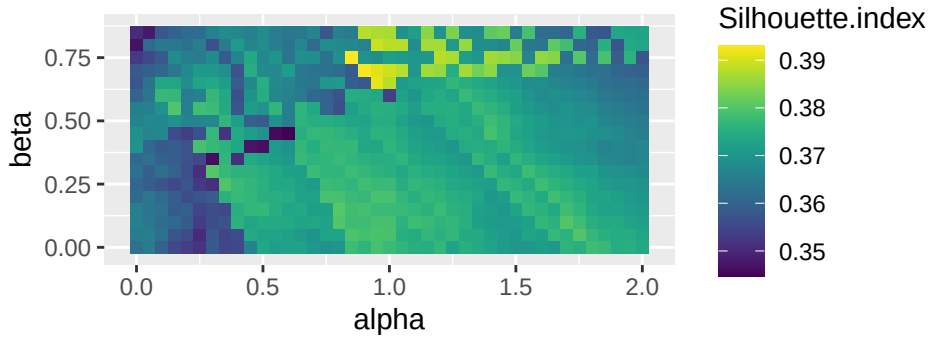


Figure 12: Impact of beta and alpha on silhouette index

```
ggplot(DFindices_SFGCM) +
  geom_raster(aes(x = alpha, y = beta, fill = XieBeni.index),  size = 5) +
  scale_fill_viridis() +
  coord_fixed(ratio=1)
```

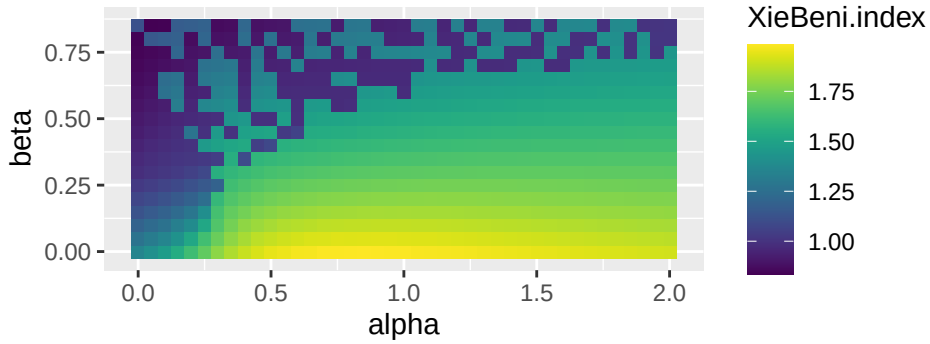


Figure 13: Impact of beta and alpha on Xie and Beni index

```
ggplot(DFindices_SFGCM) +
  geom_raster(aes(x = alpha, y = beta, fill = spConsistency),  size = 5) +
  scale_fill_viridis() +
  coord_fixed(ratio=1)
```

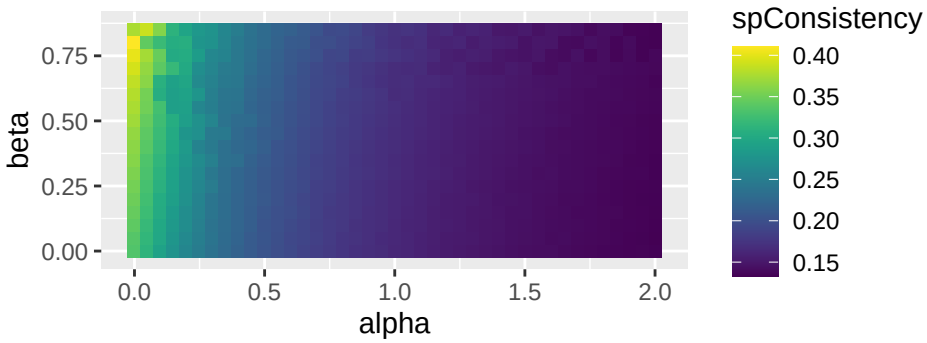


Figure 14: Impact of beta and alpha on spatial inconsistency

The first two plots indicate that some pairs of *alpha* and *beta* in particulat yield good results in the range 0.8 $<$ *alpha* $<$ 1.2 and 0.6 $<$ *beta* $<$ 0.8. The last plot shows that the selection of *beta* has no impact on the spatial consistency.

Considering the previous plots, we decide to retain the solution with $beta = 0.65$ and $alpha = 0.95$ which yield very good results for all the indices considered.

```
SGFCM <- SGFCMeans(Data,WMat,k = 4,m=1.5, alpha=0.95, beta = 0.65,
                   tol=0.0001, standardize = FALSE, verbose = FALSE, seed = 456)
```

Again, we compare here the generalized and the classical version of the spatial c-means algorithm.

```
r1 <- calcqualityIndexes(Data, SFCM$Belongings,m = 1.5)
r2 <- calcqualityIndexes(Data, SGFCM$Belongings,m = 1.5)

diagSFCM <- spatialDiag(belongmatrix = SFCM$Belongings, nblistw = WMat,
                        undecided = 0.45,nrep = 500)
diagSGFCM <- spatialDiag(belongmatrix = SGFCM$Belongings, nblistw = WMat,
                         undecided = 0.45,nrep = 500)

df <- cbind(
  c(unlist(r1),diagSFCM$SpConsist),
  c(unlist(r2),diagSGFCM$SpConsist)
)
row.names(df)[length(row.names(df))] <- "sp.consistency"

knitr::kable(df,digits = 3,col.names = c("SFCM","SGFCM"))
```

	SFCM	SGFCM
Silhouette.index	0.370	0.391
Partition.entropy	0.765	0.544
Partition.coeff	0.592	0.715
Modified.partition.coeff	0.456	0.620
XieBeni.index	1.975	0.971
FukuyamaSugeno.index	1100.085	739.161
Explained.inertia	0.274	0.347
sp.consistency	0.183	0.171

The solution of the SGFCM is better on the semantic and the spatial aspects.

We can compare the maps

```
SFCMMaps <- mapClusters(geodata = LyonIris,belongmatrix = SFCM$Belongings,undecided = 0.45)
SGFCMMaps <- mapClusters(geodata = LyonIris,belongmatrix = SGFCM$Belongings,undecided = 0.45)

ggarrange(SFCMMaps$ProbaMaps[[1]],SGFCMMaps$ProbaMaps[[1]], nrow = 1, ncol = 2,
          common.legend = TRUE, legend = "bottom")
```

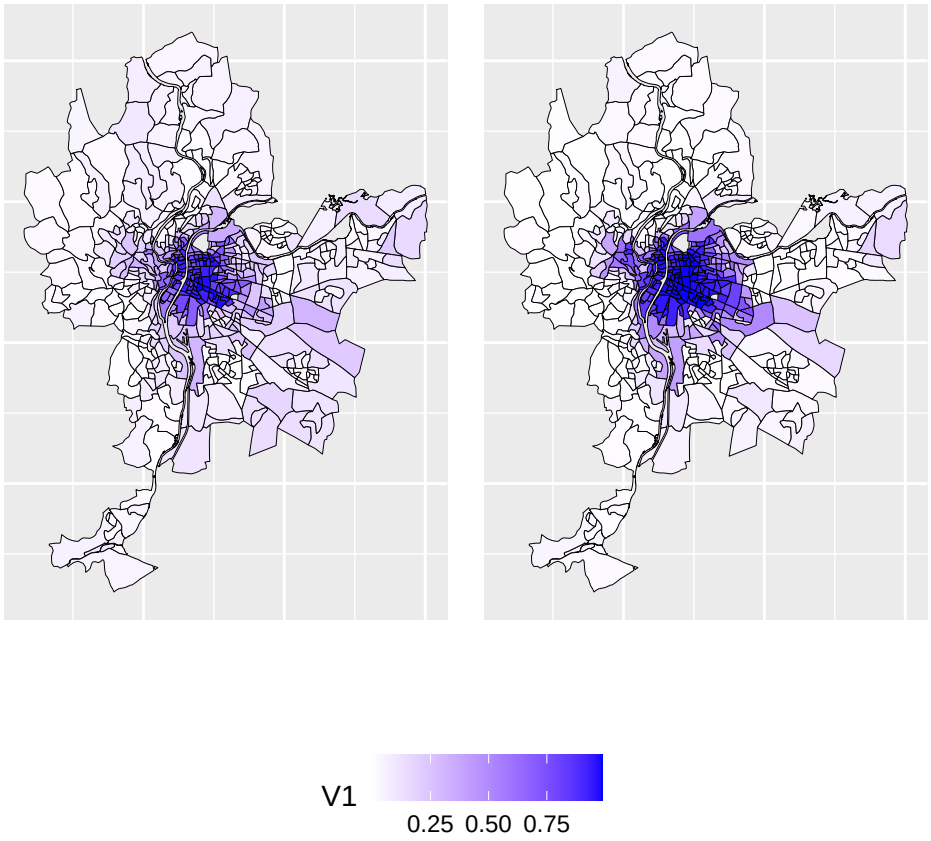


Figure 15: Probability of belonging to cluster 1

```
ggarrange(SFCMMaps$ProbaMaps[[2]],SGFCMMaps$ProbaMaps[[2]], nrow = 1, ncol = 2,
          common.legend = TRUE, legend = "bottom")
```

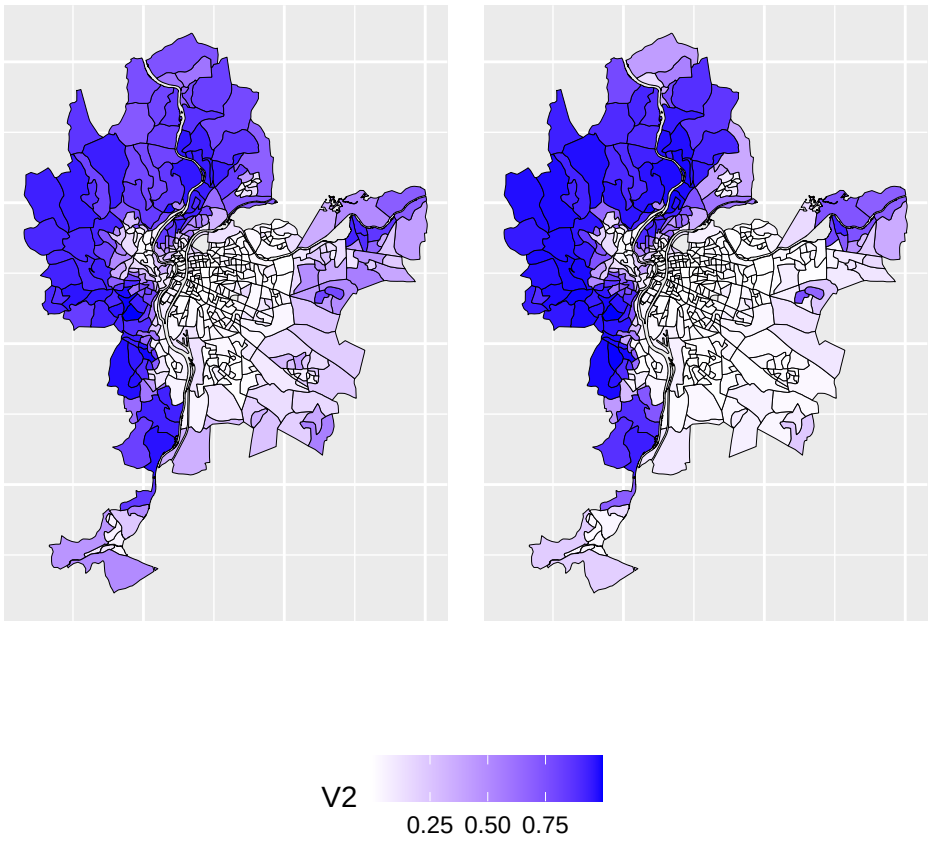


Figure 16: Probability of belonging to cluster 2

```
ggarrange(SFCMMaps$ProbaMaps[[3]],SGFCMMaps$ProbaMaps[[3]], nrow = 1, ncol = 2,
          common.legend = TRUE, legend = "bottom")
```

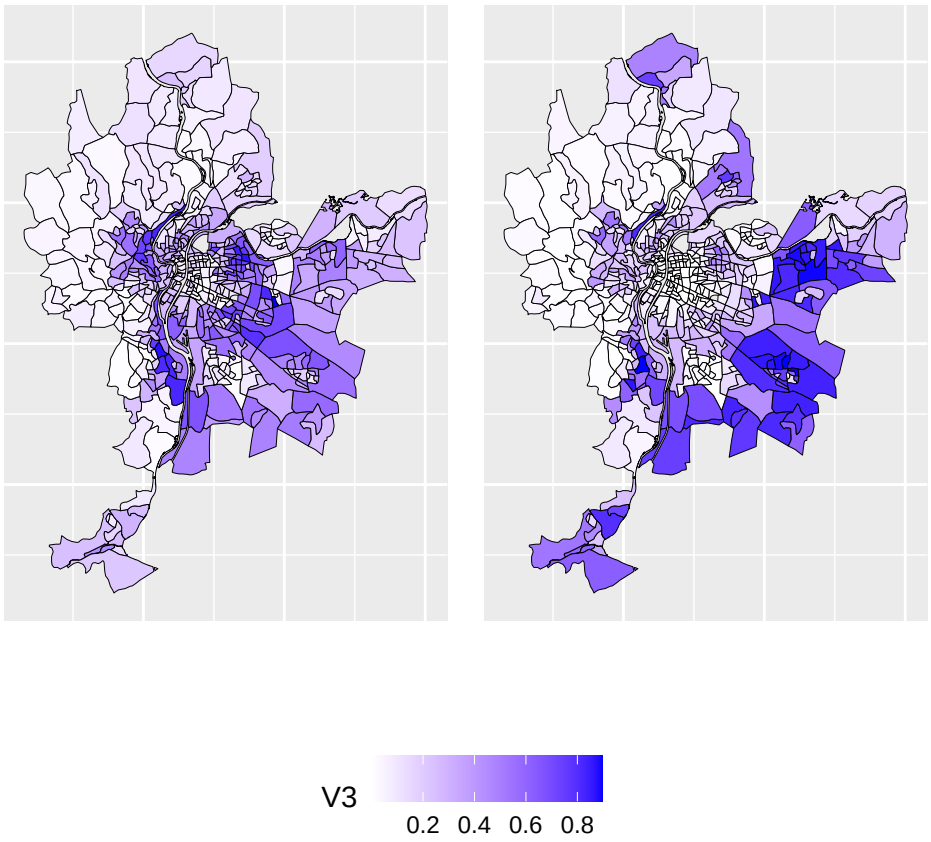


Figure 17: Probability of belonging to cluster 3

```
ggarrange(SFCMMaps$ProbaMaps[[4]],SGFCMMaps$ProbaMaps[[4]], nrow = 1, ncol = 2,
          common.legend = TRUE, legend = "bottom")
```

```
ggarrange(SFCMMaps$ClusterPlot,SGFCMMaps$ClusterPlot, nrow = 1, ncol = 2,
          common.legend = TRUE, legend = "bottom")
```

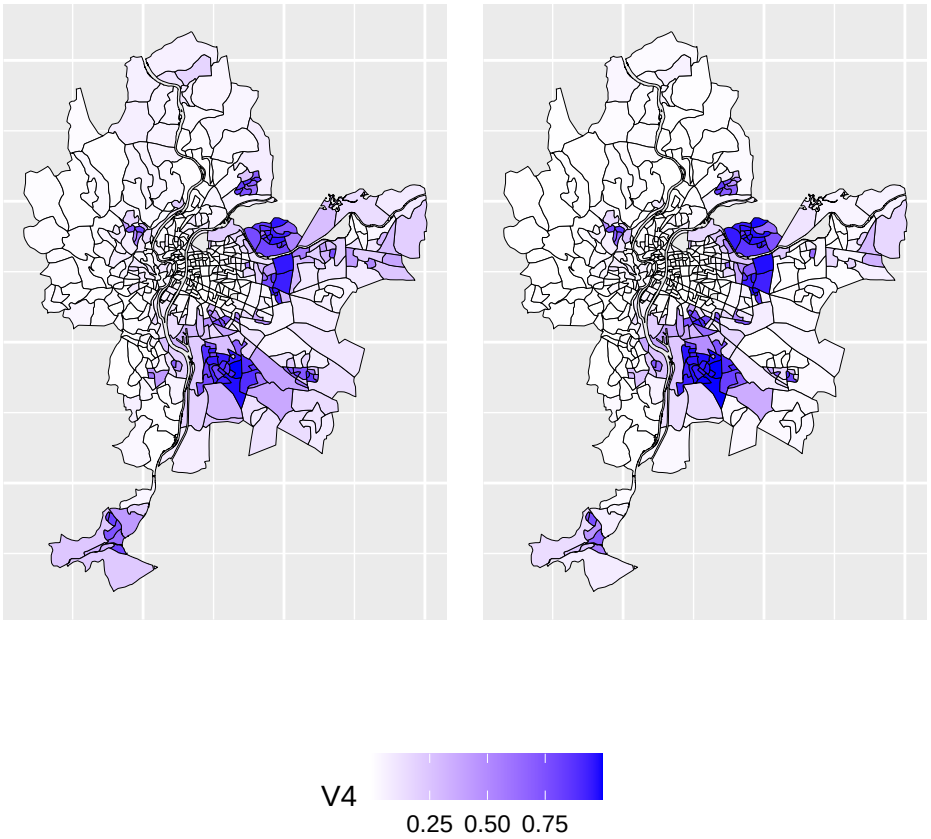


Figure 18: Probability of belonging to cluster 4

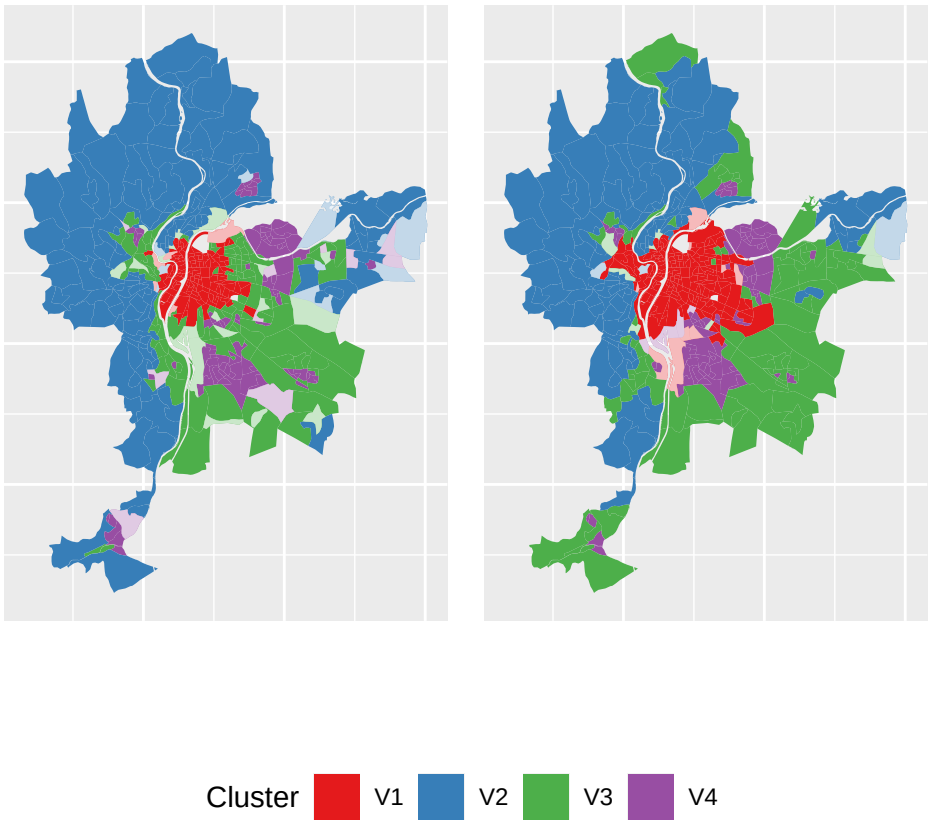


Figure 19: Most likely cluster and undecided units

Comparing the spatial consistency of FCM, GFCM, SFCM and SGFCM

Now, we can do a deeper spatial analysis and compare the spatial consistency of the four classifications realized (FCM, GFCM, SFCM, SGFCM).

```
spdiag_1 <- spatialDiag(Cmean$Belongings, nblistw = WMat,nrep=250)
spdiag_2 <- spatialDiag(GCmean$Belongings, nblistw = WMat,nrep=250)
spdiag_3 <- spatialDiag(SFCM$Belongings, nblistw = WMat,nrep=250)
spdiag_4 <- spatialDiag(SGFCM$Belongings, nblistw = WMat,nrep=250)

#looking at the moran I values for each group
moran_table <- data.frame(cbind(spdiag_1$MoranValues$MoranI,
                     spdiag_2$MoranValues$MoranI,
                     spdiag_3$MoranValues$MoranI,
                     spdiag_4$MoranValues$MoranI
                     ))
row.names(moran_table) <- paste("cluster ",1:4,sep="")
knitr::kable(moran_table, digits = 3,
             col.names = c("FCM","GFCM","SFCM","SGFCM"),
             caption = "Moran I index for the columns of the membership matrix"
             )
```

Table 4: Moran I index for the columns of the membership matrix

	FCM	GFCM	SFCM	SGFCM
cluster 1	0.835	0.789	0.908	0.913
cluster 2	0.723	0.624	0.860	0.865
cluster 3	0.389	0.408	0.645	0.704
cluster 4	0.547	0.489	0.749	0.741

Not surprisingly, the Moran I values calculated on the membership matrices are higher for SFCM and SGFCM, indicating stronger spatial structures in the classifications.

```
print(c(spdiag_1$SpConsist, spdiag_2$SpConsist,spdiag_3$SpConsist,spdiag_4$SpConsist))
```

```
## [1] 0.3344419 0.3933803 0.1837956 0.1713605
```

Considering the values of spatial inconsistency, we could check if the value obtained for SGFCM is significantly lower than the one of SFCM. Considering the previous 250 permutated values, we can calculate a pseudo p-value:

```
sum(spdiag_4$SpConsist > spdiag_3$SpConsistSamples) / length(spdiag_3$SpConsistSamples)
```

```
## [1] 0
```

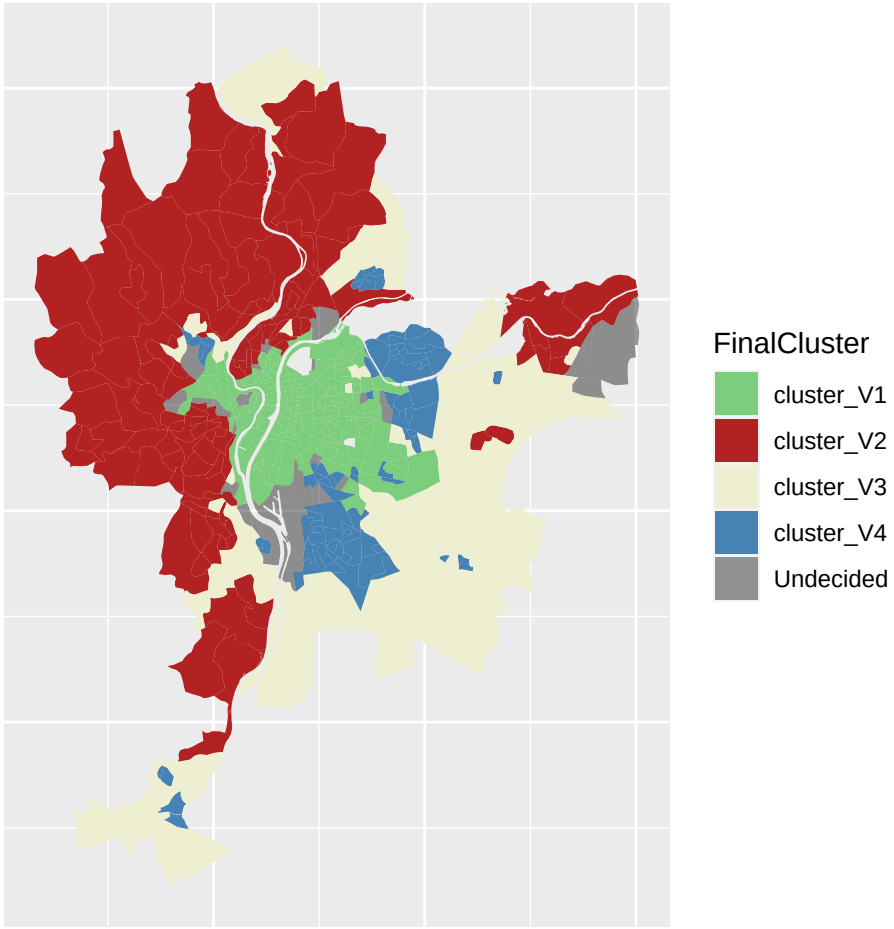
It appears that out of 250 permutations, the observed values of spatial inconsistency of SGFCM are always lower than that of SFCM. The difference is significant at the threshold 0.004 (=1/250)

We can map the undecided observations of the final solution. These entities should be analyzed more precisely. Selecting them is easy with the function *undecidedUnits* of the **geocmeans** package.

```
Undecided <- undecidedUnits(SGFCM$Belongings,0.45)
LyonIris$FinalCluster <- ifelse(Undecided=="Undecided",
                                "Undecided",paste("cluster",Undecided,sep="_"))

#mapping the groups
DFmapping <- merge(FortiData,LyonIris,by.x="id",by.y="OID")

ggplot(data=DFmapping)+
  geom_polygon(aes(x=long,y=lat,group=group,fill=FinalCluster),color=rgb(0,0,0,0))+
  coord_fixed(ratio = 1)+
  scale_fill_manual(name="FinalCluster",values = c("cluster_V1"="palegreen3",
                                "cluster_V2"="firebrick",
                                "cluster_V3"="lightyellow2",
                                "cluster_V4"="steelblue",
                                "cluster_V5"="pink",
                                "Undecided"=rgb(0,0,0,0.4)))+
  theme( axis.title = element_blank(),
    axis.text = element_blank(),
    axis.ticks = element_blank()
    )
```



Interpreting the results of the final classification

One can obtain a lot of descriptive information about the final groups with three functions of **geocmeans** :

- *summarizeClusters* : calculate summary statistics for each group for a given dataset by using the membership matrix as weights (sticking with the fuzzy spirit).
- *spiderPlots* : display a spider plot allowing to compare quickly the differences between groups.
- *violinPlots* : display a violin plot for each variable in a given dataset. Observations must be grouped before.

```
summarizeClusters(LyonIris@data[AnalysisFields],belongmatrix = SGFCM$Belongings,
                  weighted = T,dec = 3)
```

```
## $Cluster_1
##      Lden   NO2    PM25   VegHautPrt Pct0_14 Pct_65 Pct_Img TxChom1564
## Q5   50.589 27.985 16.784 6.338      8.32    6.771  6.047   7.786
## Q10  51.585 29.229 17.273 7.78       10.332  8.408  6.904   8.578
## Q25  53.746 30.867 18.333 10.335     12.47   10.739 8.826   11.115
## Q50  56.385 35.038 18.952 13.857     15.027  13.893 11.984  13.119
## Q75  59.121 38.887 19.583 18.095     17.456  17.5   15.251  15.498
## Q90  62.476 41.161 20.007 26.105     19.679  20.491 18.64   18.43
## Q95  64.181 44.368 20.208 31.449     21.197  23.01  20.204  20.58
## Mean 56.628 35.057 18.77  15.294     14.91   14.085 12.45   13.567
## Std  4.37   5.765  1.208  7.716      4.416   5.184  5.642   5.298
##      Pct_brevet NivVieMed
```

```
## Q5   7.221      17420.35
## Q10  8.474      18717.83
## Q25  10.738     20126.62
## Q50  13.996     22573
## Q75  19.773     24717.29
## Q90  25.564     28049.48
## Q95  30.338     31053.11
## Mean 15.936     22868.49
## Std  8.182      4004.567
##
## $Cluster_2
##      Lden   NO2    PM25   VegHautPrt Pct0_14 Pct_65 Pct_Img TxChom1564
## Q5   44.821 14.772 12.649 12.107     12.776  12.691 3.787   6.723
## Q10  45.822 15.692 12.92  14.655     14.297  13.379 4.259   6.999
## Q25  49.653 18.685 13.741 21.738     16.697  16.145 5.738   7.934
## Q50  52.309 22.073 14.555 28.635     18.605  19.207 7.466   9.603
## Q75  54.808 25.969 15.695 36.479     20.816  22.61  10.08   12
## Q90  58.109 28.994 16.533 42.037     22.335  27.455 14.169  15.038
## Q95  60.151 32.378 17.276 44.887     22.834  29.574 17.307  17.687
## Mean 52.16  22.437 14.748 28.342     18.401  19.835 8.509   10.619
## Std  4.802  5.595  1.514  10.237     3.598   5.877  4.842   5.629
##      Pct_brevet NivVieMed
## Q5   10.061     19832.14
## Q10  11.653     20783.02
## Q25  14.719     22402
## Q50  18.431     24529.42
## Q75  23.228     28398.37
## Q90  29.315     31030.27
## Q95  32.435     34173.22
## Mean 19.708     25380.93
## Std  8.288      4369.776
##
## $Cluster_3
##      Lden   NO2    PM25   VegHautPrt Pct0_14 Pct_65 Pct_Img TxChom1564
## Q5   49.797 17.778 13.828 6.065      12.749  9.203  5.735   7.013
## Q10  51.341 19.266 14.049 7.238      14.851  11.218 7.02    8.041
## Q25  53.118 21.733 15.145 10.737     17.283  13.914 9.307   10.339
## Q50  55.598 25.369 15.653 14.946     19.507  17.316 12.743  13.099
## Q75  58.772 30.374 17.072 20.605     21.754  20.906 18.389  16.649
## Q90  62.576 35.15  18.58  29.923     24.503  23.812 22.798  20.812
## Q95  63.925 38.5   18.881 34.802     26.287  25.475 28.386  24.275
## Mean 56.112 26.14  16.017 16.576     19.359  17.264 14.365  14.03
## Std  4.53   6.41   1.613  8.737      4.516   5.483  7.41    7.064
##      Pct_brevet NivVieMed
## Q5   13.487     15635.57
## Q10  17.018     16541.31
## Q25  20.78      18807.17
## Q50  25.832     20430.87
## Q75  32.012     23021.89
## Q90  38.263     25203.61
## Q95  43.712     26271.93
## Mean 27.218     20877.38
## Std  10.51      3534.917
##
```

```
## $Cluster_4
##      Lden   NO2    PM25   VegHautPrt Pct0_14 Pct_65 Pct_Img TxChom1564
## Q5   51.602 20.07  14.136 6.455      16.459  7.549  10.481  11.133
## Q10  52.383 21.383 14.56  7.648      18.483  8.95   16.124  13.787
## Q25  55.067 23.266 15.907 10.844     21.452  11.048 21.008  17.872
## Q50  57.244 26.534 16.538 14.219     24.399  13.815 27.058  23.004
## Q75  59.531 31.85  17.734 18.39      27.6    17.126 32.865  31.831
## Q90  63.444 37.194 18.757 24.836     30.977  20.536 38.251  34.398
## Q95  64.732 39.632 19.069 28.611     32.411  23.234 41.015  37.971
## Mean 57.516 27.992 16.693 15.476     24.402  14.223 26.758  24.188
## Std  4.253  6.623  1.518  7.25       6.315   5.024  9.457   9.479
##      Pct_brevet NivVieMed
## Q5   21.535     12404.92
## Q10  26.813     12891.25
## Q25  32.703     13852.98
## Q50  38.41      15648.34
## Q75  45.03      18118.53
## Q90  49.572     19546.11
## Q95  53.704     21902.5
## Mean 38.607     16111.26
## Std  11.927     3117.786
```

```
spiderPlots(LyonIris@data[AnalysisFields], SGFCM$Belongings,
            chartcolors = c("darkorange3","grey4","darkgreen","royalblue"))
violinPlots(LyonIris@data[AnalysisFields], SGFCM$Groups)
```

That's all, folks ! Following are the enhancements for the next version

- introduce other methods of spatial c-means
- open some other parameters to the user (such as the function defining the convergence criterion)
- work on documentation
- improve calculus speed by dropping some *apply* in the code
- adding the manhattan distance (useful when data have a high dimensionality)

Cai, Weiling, Songcan Chen, and Daoqiang Zhang. 2007. "Fast and Robust Fuzzy c-Means Clustering Algorithms Incorporating Local Information for Image Segmentation." *Pattern Recognition* 40 (3): 825–38.

Gelb, Jérémy, and Philippe Apparicio. 2021. "Apport de La Classification Floue c-Means Spatiale En géographie: Essai de Taxinomie Socio-résidentielle Et Environnementale à Lyon." *Cybergeo: European Journal of Geography*.

Zhao, Feng, Licheng Jiao, and Hanqiang Liu. 2013. "Kernel Generalized Fuzzy c-Means Clustering with Spatial Information for Image Segmentation." *Digital Signal Processing* 23 (1): 184–99.