Procédure de sélection des flux à cartographier selon un critère spatial |

Le voisinage spatial : la contiguité ordinale (CKij)

F. Bahoken, Janvier, 2016 - mise en ligne avril, 2016

1. Présentation :

Ce document fait partie d'une série de procédures permettant de sélectionner les flux selon des critères liés à l'espacement des lieux. Le présent document (2/4) porte sur la construction d'une matrice de voisinage ordinal, notée (CKij). Il s'agit d'une matrice de contiguité d'ordre k={1,2,4,…} entre les couples de lieux (i,j). Sa valeur correspond au nombre de limites de zones (frontières) qu'il convient de franchir pour aller de (i) à (j).

On va ensuite construire ici trois matrices de contigüité : d'ordres 1, 2 et 4.

On mobilise pour cela un fond de carte surfacique. Le cas d'illustration est le découpage communal de l'Isère. La matrice de flux associée est celle des flux domicile-travail de l'Isère que l'on réduira aux seuls flux s'exprimant entre des couples de lieux séparé de 1, de 2 ou de 4 limites de zones : (Fij)<-(CKij, avec k=n)

Données exemple

isere_com.shp : fond de carte des communes de l'Isère **matrice_isere.csv** : matrice de flux de l'Isère (format matriciel)

2. Préparation

Déclaration des packages et connection

```
#purge environnement
rm(list=ls())
```

```
setwd("C:/FRANCOISE/R/2-2_contig_CKij_ok")

library(rgeos)
library(maptools)
library(proj4)
library(reshape2)
library(rCarto)
library(rgdal)
```

Chargement du fond de carte (fdc)

```
fdc<- readShapeSpatial("./isere_com.shp")
colnames(fdc@data)[4]<-"INSEE_COM"
```

3) Matrice de contiguité ordinale (CKij) d'ordre (k=1)

Construction de CKij_1

```
# la fonction gintersect agit sur la geometrie du fdc
contig<-gIntersects(fdc, byid = TRUE, prepared=TRUE)
row.names(contig)<-fdc@data$INSEE_COM
colnames(contig)<-fdc@data$INSEE_COM

#mise à 1 ou 0 de la matrice contig
for (i in 1:nrow(contig))
  for (j in 1:ncol(contig))
  {if (contig[i,j]==TRUE) {contig[i,j]<-1}
   if (contig[i,i]!=0)  {contig[i,i]<-0}
  }

#Transformation de la matrice contig au format liste
tab <-melt(contig)
names(tab) = c("CODE_i", "CODE_j","cij")
head(tab)
```

```
##   CODE_i CODE_j cij
## 1  38226  38226   0
## 2  38206  38226   0
## 3  38301  38226   0
```

```
## 4  38448  38226   0
## 5  38287  38226   0
## 6  38153  38226   0
```

On renomme la matrice (pour préparer l'export en CKij_1)

```
# Contiguité d'ordre 1
ordre_1<-tab

# Suppression des valeurs égales à 0
ordre_1<-ordre_1[ordre_1[,"cij"]!=0,]

#export
write.table(ordre_1,file="CKij_1.csv",sep=";",eol="\n")
```

4) Matrice de contiguité ordinale (CKij) d'ordre (k=2)

Construction de CKij_2

Cette matrice d'ordre 2 est construite à partir de la matrice d'ordre 1 précédente.

L'objectif est de lancer une boucle grâce à laquelle on va rechercher, pour une entité (i) quelconque, le voisin d'ordre 1 de son voisin d'ordre 1 qui sera alors le voisin d'ordre 2 de l'unité spatiale (i) de départ.

```
CKij_1<-ordre_1
CKij_1$CODE_i1<-0
CKij_1$CODE_j1<-0
CKij_1$CODE_j2<-0
CKij_1$cij2<-0

for (k in 1:nrow(CKij_1))
{
  if (CKij_1$cij[k]==1)
  {
    col_i<-CKij_1$CODE_i[k]
    col_j<-CKij_1$CODE_j[k]

    for (v in 1:nrow(CKij_1))
    {
      if (CKij_1$CODE_i[v]==col_j && CKij_1$CODE_i[v]!=CKij_1$CODE_j[v])
      {
        CKij_1$CODE_i1[k]<-col_i  # i ordre 1
        CKij_1$CODE_j1[k]<-col_j   #j ordre 1
        CKij_1$CODE_j2[k]<- CKij_1$CODE_j[v]  #j ordre 2
        CKij_1$cij2[k]<-2
      }
    }
  }
}

head(CKij_1)
```

```
##     CODE_i CODE_j cij CODE_i1 CODE_j1 CODE_j2 cij2
## 16   38366  38226   1   38366   38226   38456    2
## 96   38125  38226   1   38125   38226   38456    2
## 142  38321  38226   1   38321   38226   38456    2
## 154  38127  38226   1   38127   38226   38456    2
## 378  38403  38226   1   38403   38226   38456    2
## 444  38456  38226   1   38456   38226   38456    2
```

(CODE_i)=(CODEi1) et (CODEJ)=(CODE_J1) à l'ordre 1 et (CODE_J2)= lieu de destination à l'ordre 2. Ces apparents doublons permettent de vérifier l'opération.

Finalisation et export de CKij_2

avec (i) lieu d'origine et (j2) le lieu de destination à l'ordre 2 cij2=ordre_2, elle contient la modalité 2 choisie ordre_2 correspond bien à CKij avec (k=2). On supprime ensuite les valeurs nulles ou impossibles.

```
ordre_2<-data.frame(CKij_1$CODE_i,CKij_1$CODE_j2,CKij_1$cij2)
names(ordre_2) = c("i", "j2","cij2")

#suppression de la diagonale sur ckij ordre 2
for (z in 1:nrow(ordre_2))
  if (ordre_2$i[z] == ordre_2$j2[z]) ordre_2$cij2[z]<-0
```

```
#Suppression des valeurs égales à 0
ordre_2<-ordre_2[ordre_2[,"cij2"]!=0,]

#export
write.table(ordre_2,file="./CKij_2.csv",sep=";",eol="\n")
```

NOTES : - La représentation graphique de cette matrice CKij_2.csv conduit à un graphe de voisinage d'ordre 2 qu'il est possible de représenter ; la procédure est décrite ici : https://elementr.hypotheses.org/489

- Pour cartographier ces flux qui parcourent une distance inférieure à CKij, avec k=2, il faut:

1. l'ajouter à l'ordre 1 (CKij=1) pour conserver le principe de continuité spatiale.
2. réduire (Fij) en fonction de la matrice de voisinage qui en résulte Fij <-[(Ckij), (k<2)], comme indiqué dans le billet suivant : https://elementr.hypotheses.org/370

Il est, bien entendu, possible de ne représenter que les flux qui franchissement 2 et seulement 2 limites de zones Fij <- (CKij, k=2) mais cela me semble moins intéressant.

5) Matrice de contiguité ordinale (CKij) d'ordre (k=4)

Construction de CKij_4

Nous reproduisons l'opération précédente, en considérant que pour tout (i), le voisin d'ordre 2 de son voisin d'ordre 2 est le voisin d'ordre 4 de (i).

Construction de la matrice d'ordre 4 (ordre_4) à partir de la matrice d'ordre 2 (ordre_2)

```
ordre_2$CODE_i<-0
ordre_2$CODE_j2<-0
ordre_2$CODE_j4<-0
ordre_2$cij4<-0

for (k in 1:nrow(ordre_2))
{
  if (ordre_2$cij2[k]==2)
  {
    col_i4<-ordre_2$i[k]
    col_j4<-ordre_2$j2[k]

    for (v in 1:nrow(ordre_2))
    {
      if (ordre_2$i[v]==col_j4 && ordre_2$i[v]!=ordre_2$j2[v])
      {
        ordre_2$CODE_i[k]<-col_i4
        ordre_2$CODE_j2[k]<-col_j4
        ordre_2$CODE_j4[k]<-ordre_2$j2[v]
        ordre_2$cij4[k]<-4
      }
    }
  }
}

head(ordre_2) # pour vérification en cours de route
```

```
##       i    j2 cij2 CODE_i CODE_j2 CODE_j4 cij4
## 1 38366 38456    2  38366   38456   38224    4
## 2 38125 38456    2  38125   38456   38224    4
## 3 38321 38456    2  38321   38456   38224    4
## 4 38127 38456    2  38127   38456   38224    4
## 5 38403 38456    2  38403   38456   38224    4
## 7 38005 38547    2  38005   38547   38397    4
```

Finalisation et export de CKij_4

```
ordre_4<-data.frame(ordre_2$CODE_i,ordre_2$CODE_j4,ordre_2$cij4)
names(ordre_4) = c("i", "j4","cij4")

#suppression de la diagonale sur CKij ordre 4
for (z in 1:nrow(ordre_4))
  if (ordre_4$i[z] == ordre_4$j4[z]) ordre_4$cij4[z]<-0

# Suppression des valeurs égales à 0
ordre_4<-ordre_4[ordre_4[,"cij4"]!=0,]

# export
write.table(ordre_4,file="./CKij_4.csv",sep=";",eol="\n")
```

NOTES : Comme précédemment, il est possible de visualiser le graphe de voisinage correspondant, voir le billet : https://elementr.hypotheses.org/489

6) Cartographie des flux qui s'expriment à différents ordres de proximités

Pour conserver / respecter le principe de continuité* spatiale, la cartographie de ces flux qui s'expriment dans un voisinage de (k), ou qui parcourent une distance inférieure ou égale à [(CKij), avec (k=4)], notée (CKij_4), il convient : (1) d'ajouter cette matrice (CKij_4) aux matrices précédentes : à l'ordre 1 (CKij_1) et à l'ordre 2 (CKij_2) ; (2) réduire (Fij) en fonction de la matrice de voisinage qui en résulte : [(Fij) <- (CKij), k<4)] ; Cette procédure de réduction de (Fij) est présentée ici : https://elementr.hypotheses.org/370

*Bien entendu, il est également possible de ne représenter que les flux qui franchissement 4 et seulement 4 limites de zones (Fij <- [(CKij, k=4)] mais cela me semble moins intéressant.