Agrégation de données carroyées sur un zonage à façon avec R

23 mars 2022

Introduction

Ce document présente un exemple de code R dont le principe consiste à déterminer, pour un **zonage** donné, l'**ensemble des carreaux de 200m qui le recouvrent** puis à **calculer les agrégats des données carroyées issues de Filosofi** sur cet ensemble de carreaux.

Pour ce faire, l'utilisateur doit disposer d'un **contour géographique** de sa zone (par exemple au format .shp ou .gpkg).

La table des résultats contient une ligne pour chaque agrégat calculé sur le zonage. L'ensemble de carreaux étant en général plus large que la zone, les agrégats obtenus seront des estimations qui tendent à surestimer les valeurs réelles.

Programme

Chargement des librairies

Pour commencer, on installe (si cela n'est pas encore fait) puis importe les librairies nécessaires.

```
lpackages <- list("data.table","dplyr","tidyr","sf","stringr","knitr")

for (pack in lpackages) {
  if(!require(pack,character.only = T)) {
    install.packages(pack)
    require(pack,character.only = T)
    }
}
```

Chargement de la base géographique des carreaux de Filosofi

On définit la liste des indicateurs pour lesquels on souhaite calculer des agrégats sur une zone à façon.

```
listeIndic <- c(
  "Ind", "Men", "Men_pauv", "Men_1ind", "Men_5ind", "Men_prop",
```

```
  "Men_fmp", "Ind_snv",  "Men_surf",  "Men_coll",  "Men_mais",
  "Log_av45",  "Log_45_70",  "Log_70_90",  "Log_ap90",  "Log_inc",
  "Log_soc",  "Ind_0_3",  "Ind_4_5",  "Ind_6_10", "Ind_11_17",
  "Ind_18_24", "Ind_25_39", "Ind_40_54", "Ind_55_64", "Ind_65_79",
  "Ind_80p", "Ind_inc"
)
```

On ajoute des colonnes qui seront nécessaires par la suite :

- `Idcar_200m` : l'identifiant du carreau de 200m ;
- `lcog_geo` : l'ensemble des codes commune qui intersectent le carreau ;
- `Ind` : le nombre d'individus du carreau.

```
listeIndic <- unique(c(listeIndic, "Idcar_200m","lcog_geo", "Ind"))
```

On importe la table des carreaux.

```
cheminFichier <- "CHEMIN_DU_FICHIER_DES_CARREAUX_GPKG_OU_SHP"
carreaux <- st_read(cheminFichier)[,listeIndic]
```

Filtrage pour limiter la taille des bases de données [optionnel]

Si on dispose d'une liste de communes, ou de départements dans lesquels se trouve la zone à façon, on peut ne conserver que les carreaux de la grille qui intersectent le territoire qui nous intéresse.

Ceci permet de limiter la taille de la base manipulée et donc les temps de calcul à venir pour la sélection géographique des carreaux.

Pour rappel, la variable `lcog_geo` est la concaténation des COG de chaque commune intersectée par le carreau (cf. documentation).

L'exemple ci-dessous permet de sélectionner tous les carreaux intersectant les communes de Montrouge (`92049`), Malakoff (`92046`), Châtillon (`92020`) ou de Bagneux (`92007`) dans les Hauts-de-Seine.

```
liste_depcom <- c("92049","92046","92020","92007")

# Liste de booléens qui renseignent s'il faut conserver ou non chaque
# ligne (c'est-à-dire chaque carreau) de la base de données
position <- strsplit(carreaux$lcog_geo,'(?<=.{5})', perl=TRUE) %>%
   lapply(function(x) any(x%in% liste_depcom)) %>%
   unlist()

# Sélection des carreaux à conserver
carreaux_select <- carreaux[position,]
```

Attention : il faut s'assurer que les codes désignant les communes correspondent :

- *au code officiel géographique (COG), et non au code postal ;*
- *à la géographique en vigueur au 1er janvier 2018.*

Optimisation du temps de chargement La fonction `charger_carreaux` permet de charger de manière rapide des carreaux sur un ensemble de communes (ou, au choix, de départements) à partir du fichier au format CSV.

```
charger_carreaux <- function(cheminFichierCSV,
                             listeIndic = NULL,
                             tailleCarreaux = 200,
                             Idcar = "Idcar_200m",
                             com_selec = NULL,
```

```
                              dep_selec = NULL
){

  # Lecture du fichier CSV
  carreaux <- data.table::fread(cheminFichierCSV, showProgress = FALSE)
  if(!is.null(listeIndic)){
    carreaux <- carreaux[ ,listeIndic, with=FALSE]
  }

  # Filtrer les lignes qui concernent les communes / départements sélectionnés
  if(!is.null(dep_selec)){
    position <- strsplit(carreaux$lcog_geo,'(?<=.{2})', perl=TRUE) %>%
     lapply(function(x) any(x%in% dep_selec)) %>%
     unlist()
  } else if(!is.null(com_selec)){
    position <- strsplit(carreaux$lcog_geo,'(?<=.{5})', perl=TRUE) %>%
     lapply(function(x) any(x%in% com_selec)) %>%
     unlist()
  } else{
    position = rep(TRUE, nrow(carreaux))
  }
  carreaux <- carreaux[position,]


  # Récupération de l'identifiant carreau, sa projection, ses coordonnées (coin inférieur gauche).
  cIdInspire <- carreaux[[Idcar]]
  epsg <- as.integer(str_sub(str_extract(cIdInspire[1], "CRS\\d+"), 4))
  ordonneesCarreaux <- as.integer(str_sub(str_extract(cIdInspire, "N\\d+"), 2))
  abscissesCarreaux <- as.integer(str_sub(str_extract(cIdInspire, "E\\d+"), 2))


  # Création d'une colonne geometry (coordonnées des contours des carreaux)
  carreaux$geometry <- sprintf("POLYGON ((%i %i, %i %i, %i %i, %i %i, %i %i))",
                               abscissesCarreaux, ordonneesCarreaux,
                               abscissesCarreaux + tailleCarreaux,
                               ordonneesCarreaux,
                               abscissesCarreaux + tailleCarreaux,
                               ordonneesCarreaux + tailleCarreaux,
                               abscissesCarreaux, ordonneesCarreaux +
                                 tailleCarreaux,
                               abscissesCarreaux, ordonneesCarreaux)

  # Transformation en objets géométriques à l'aide du package sf
  carreauxSf <- sf::st_as_sf(carreaux, wkt = "geometry", crs = epsg)

  return(carreauxSf)

}
```

On reproduit donc grâce à cette fonction les actions de chargement qui avaient été effectuées précédemment.

```
liste_depcom <- c("92049","92046","92020","92007")
cheminFichierCSV <- "CHEMIN_DU_FICHIER_DES_CARREAUX_CSV"
carreaux_select <- charger_carreaux(cheminFichierCSV, listeIndic,
```

```
                                com_selec = liste_depcom)
```

Chargement de la zone à façon

À présent, on importe la zone à façon. Dans cet exemple, la zone à façon est directement stockée sous forme vectorielle (fichier .shp, .gpkg, .kml, etc.).

```
zaf <- st_read("CHEMIN_DE_LA_ZONE_A_FACON_EN_SHP_OU_GPKG")
```

La zone à façon fictive que nous utilisons ici correspond à un disque de rayon 1 km au centre de l'amas de carreaux.

On renomme la variable définissant la géométrie (elle peut varier selon le type de fichier importé).

```
zaf <- st_sf(geometry = st_geometry(zaf))
```

Si besoin, on reprojette la zone à façon dans le même système que la couche carroyée.

```
# Les carreaux sont en EPSG 3035
epsg_carreaux <- st_crs(carreaux_select)$epsg
epsg_carreaux
```

```
## [1] 3035
```

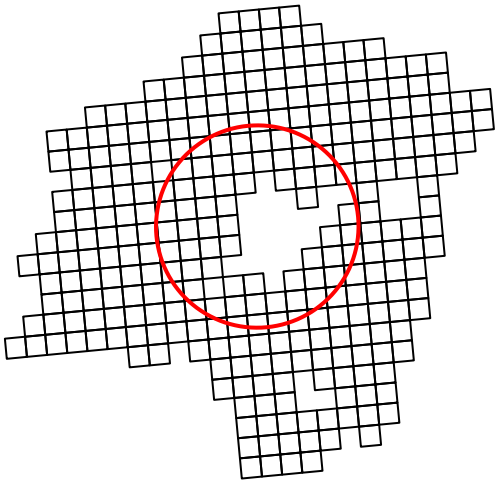
```
# La zone à façon est en EPSG 2154
epsg_zaf <- st_crs(zaf)$epsg
epsg_zaf
```

```
## [1] 2154
```

```
# On transforme (si besoin) les carreaux et la zone en EPSG 2154
zaf <- zaf %>% st_transform(2154)
carreaux_select <- carreaux_select %>% st_transform(2154)
```

On affiche les carreaux sélectionnés ainsi que la zone à façon en rouge.

```
plot(st_geometry(carreaux_select))
plot(st_geometry(zaf), border = "red", lwd = 2, add = TRUE)
```

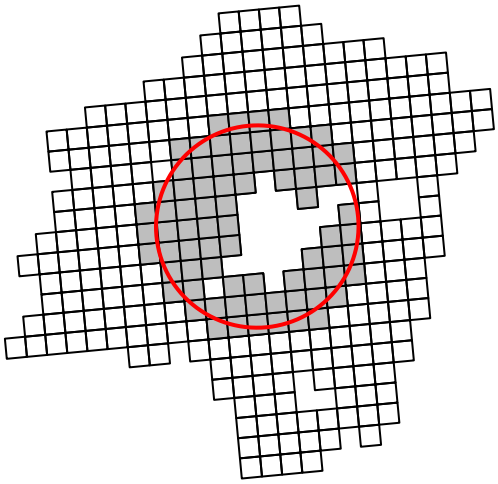


Calcul des agrégats

On sélectionne les carreaux intersectant la zone à façon. Ils apparaissent en gris dans la carte ci-dessous.

```
agregatsZaf <- carreaux_select %>%
    st_join(zaf, join = st_intersects, left = FALSE)

# Visualisation cartographique
plot(st_geometry(carreaux_select))
plot(st_geometry(agregatsZaf), col = "grey", add = TRUE)
plot(st_geometry(zaf), border = "red", lwd = 2, add = TRUE)
```



On calcule les agrégats sur les variables souhaitées.

```
agregatsZaf <- agregatsZaf %>%
    st_set_geometry(NULL) %>% # Suppression de la colonne géométrie devenue inutile
    select(-Idcar_200m, -lcog_geo) %>% # Suppression de 2 colonnes
    summarise_all(sum) %>% # Somme de tous les indicateurs
    pivot_longer(everything(), # Mise au format vertical
                 names_to = "Indicateur",
                 values_to = "Agregats")
```

Voici la table produite :

```
kable(agregatsZaf)
```

Indicateur	Agregats
Ind	45273.0
Men	20595.0
Men_pauv	2453.0
Men_1ind	8471.0
Men_5ind	1340.9
Men_prop	6367.0
Men_fmp	2639.0
Ind_snv	1163281825.0
Men_surf	1214235.9
Men_coll	19239.9
Men_mais	1355.1
Log_av45	3175.9

Indicateur	Agregats
Log_45_70	8369.1
Log_70_90	3181.0
Log_ap90	5845.0
Log_inc	24.0
Log_soc	10024.0
Ind_0_3	2672.5
Ind_4_5	1298.4
Ind_6_10	2817.9
Ind_11_17	3536.9
Ind_18_24	3063.6
Ind_25_39	11131.5
Ind_40_54	9264.0
Ind_55_64	5043.1
Ind_65_79	4441.1
Ind_80p	1913.0
Ind_inc	91.0