Agrégation de données carroyées sur un zonage à façon avec R

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 ;

```
listeIndic <- c(listeIndic, "idcar_200m","lcog_geo")
```

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 <- stringr::str_sub(carreaux$lcog_geo, 1, 5) %in% liste_depcom

# 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 2020.*

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 <- stringr::str_sub(carreaux$lcog_geo, 1, 2) %in% dep_selec
} else if(!is.null(com_selec)){
 position <- stringr::str_sub(carreaux$lcog_geo, 1, 5) %in% com_selec
} 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(stringr::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 <- sf::st_sf(geometry = sf::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
```

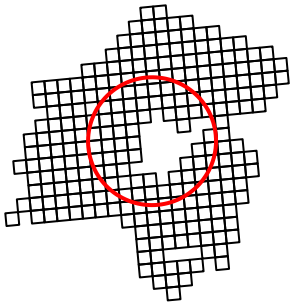
```
# La zone à façon est en EPSG 2154
epsg_zaf <- st_crs(zaf)$epsg
epsg_zaf
```

```
[1] 2154
```

```r
# 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.

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



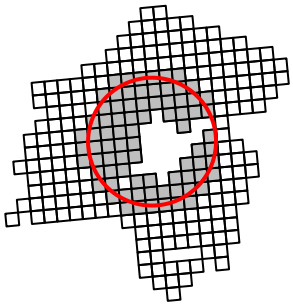
Calcul des agrégats

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

```r
agregatsZaf <- carreaux_select |>
    sf::st_join(zaf, join = st_intersects, left = FALSE)

# Visualisation cartographique
plot(sf::st_geometry(carreaux_select))
plot(sf::st_geometry(agregatsZaf), col = "grey", add = TRUE)
```

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



On calcule les agrégats sur les variables souhaitées.

```
agregatsZaf <- agregatsZaf |>
    sf::st_set_geometry(NULL) |> # Suppression colonne géo 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	44452.5
men	20593.0
men_pauv	2589.0

Indicateur	Agregats
men_1ind	8827.0
men_5ind	1280.0
men_prop	6822.0
men_fmp	2533.0
ind_snv	1199612313.0
men_surf	1204601.1
men_coll	19157.1
men_mais	1435.9
log_av45	3602.0
log_45_70	7648.9
log_70_90	3218.1
log_ap90	6079.0
log_inc	45.0
log_soc	9021.0
ind_0_3	2433.6
ind_4_5	1164.0
ind_6_10	2780.5
ind_11_17	3421.0
ind_18_24	2988.5
ind_25_39	10873.5
ind_40_54	8980.9
ind_55_64	5043.5
ind_65_79	4706.0
ind_80p	2000.0
ind_inc	61.0