How to explain voting in 2015 for populist radical right in some parts of France? A spatial econometric approach

Thibault Laurent and Christine Thomas-Agnan

14 February of 2018

Contents

The pdf version of this document is available at this link : preparation_base.pdf.

Packages needed :

```
require("sf") # new package for spatial data
require("spdep") # package for spatial econometric modelling
require("dplyr") # Wickham package
require("reshape") # Wickham package
require("cartography") # mapping
require("ggplot2") # for plottig
require("readxl") # Wickham package
require("readr") # import data
require("rgeos") # geographic boundaries manipulation
require("foreign") # importing dbf file
```

About this session :

```
sessionInfo()
```

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.3 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblasp-r0.2.18.so
##
## locale:
##  [1] LC_CTYPE=fr_FR.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=fr_FR.UTF-8        LC_COLLATE=fr_FR.UTF-8
##  [5] LC_MONETARY=fr_FR.UTF-8    LC_MESSAGES=fr_FR.UTF-8
##  [7] LC_PAPER=fr_FR.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=fr_FR.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
```

```
## other attached packages:
##  [1] foreign_0.8-69     rgeos_0.3-26       readr_1.1.1
##  [4] readxl_1.0.0       ggplot2_2.2.1.9000 cartography_2.0.2
##  [7] reshape_0.8.7      dplyr_0.7.4        spdep_0.7-4
## [10] spData_0.2.7.4     Matrix_1.2-11      sp_1.2-7
## [13] sf_0.6-0
##
## loaded via a namespace (and not attached):
##  [1] gtools_3.5.0      splines_3.4.3     lattice_0.20-35
##  [4] colorspace_1.3-2  expm_0.999-2      htmltools_0.3.6
##  [7] yaml_2.1.16       rlang_0.1.4       e1071_1.6-8
## [10] pillar_1.0.1      glue_1.1.1        DBI_0.7
## [13] bindrcpp_0.2      bindr_0.1         plyr_1.8.4
## [16] stringr_1.2.0     munsell_0.4.3     gtable_0.2.0
## [19] cellranger_1.1.0  coda_0.19-1       evaluate_0.10.1
## [22] knitr_1.19        class_7.3-14      Rcpp_0.12.14
## [25] udunits2_0.13     backports_1.1.1   scales_0.5.0.9000
## [28] classInt_0.1-24   gdata_2.18.0      deldir_0.1-14
## [31] hms_0.3           digest_0.6.12     stringi_1.0-1
## [34] gmodels_2.16.2    grid_3.4.3        rprojroot_1.2
## [37] tools_3.4.3       LearnBayes_2.15   magrittr_1.5
## [40] lazyeval_0.2.1    tibble_1.3.4      pkgconfig_2.0.1
## [43] MASS_7.3-48       assertthat_0.2.0  rmarkdown_1.8
## [46] R6_2.2.2          boot_1.3-20       units_0.4-6
## [49] nlme_3.1-131      compiler_3.4.3
```

1 The data

We provide informations coming from three data bases :

- the boundaries of the polling place (source : http://cartelec.univ-rouen.fr/).
- the results of the French "Departementales 2015" election (https://www.data.gouv.fr/fr/datasets/elections-departementales-2015-resultats-par-bureaux-de-vote/) at the polling place level.
- the results of the French census survey in 2014 given by INSEE at the iris level.

The idea is to focus on the polling places corresponding to one of the biggest cities in France. Students have to choose one city among : "Lille", "Marseille", "Montpellier", "Nantes", "Nice", "Strasbourg", "Toulon" (there are no data available for Lyon neither Paris because there are not concerned by this election) and realize a spatial econometric analysis on that region.

In this document, we present the codes which permit to create the final data basis on the city of Toulouse. Thus, students can reproducce the different preliminary steps of data management by changing the following argument by the name of the chosen city :

```
city_name_project <- "Toulouse"
```

Students are permitted to work on Toulouse if and only if they are able to create at least three new variables coming from another sources of data (such as OpenStreetMap, Toulouse Metropole, etc.).

1.1 The boundaries of the polling place

Most communes in France have only one polling place ("bureau de vote"). They usually correspond to communes with a few inhabitants. In the perspective of explaining voting for populist radical right by some socio-economic variables, we could easily use census results given by INSEE at the commune levels for these polling places.

However in the biggest communes, for obvious organisational reasons, there are several polling places and there are usually not related to any administrative boundary (such as IRIS, which is the finest level used by INSEE for diffusing the population census results).

For these communes, the authors of http://cartelec.univ-rouen.fr/ found the boundaries associated to each polling place which is a huge work. Indeed, to do this task, the authors had to look at the official documents published by the "prefecture", which indicate how are affected the different streets of a commune to a polling place.

Please, note that the boundary can change every year before a year of election which means that these boundaries are only true for the 2015 election year.

These boundaries are available at that link :

http://cartelec.univ-rouen.fr/telechargement/BV__grandes__villes__2015.rar

Download the file and unzip it.

1.1.1 Importation

```
bv <- read_sf("BV_grandes_villes_2015/BV_grandes_villes_2015.shp")
```

We only keep in this data basis the variables **NOM** and **BUREAU** :

```
bv <- select(bv, CODE, NOM, BUREAU)
```

We extract the rows corresponding to the chosen city :

```
bv_sample <- bv[grep(city_name_project, bv$NOM), ]
```

We change the codification of the **BUREAU** variable such that it is the same across the different data bases:

```
bv_sample$BUREAU <- sapply(strsplit(bv_sample$BUREAU, "_"), function (x) x[2])
bv_sample$BUREAU <- ifelse(nchar(bv_sample$BUREAU) == 4, bv_sample$BUREAU,
                           paste("0", bv_sample$BUREAU, sep = ""))
```

We map the different polling places :

```
plot(st_geometry(bv_sample))
```

The next step is to associate the "Departementales 2015" election results to that spatial data basis.

1.2 "Departementales 2015" election results

1.2.1 Importation

We import the "Departementales 2015" election results directly from that link https://www.data.gouv.fr/fr/datasets/elections-departementales-2015-resultats-par-bureaux-de-vote/ :

```
link_bv <- "https://www.data.gouv.fr/s/resources/elections-departementales-2015-resultats-par-bureaux-d
don_dep_2015 <- read_csv2(paste0(link_bv, "/20150925-104418/DP15_Bvot_T1T2.txt"))
```

We clean up properly the geographic code of the communes :

```
don_dep_2015$CODGEO <- paste(don_dep_2015$CODDPT,
                        ifelse(nchar(don_dep_2015$CODSUBCOM)==1,
                               paste("00", don_dep_2015$CODSUBCOM, sep = ""),
                        ifelse(nchar(don_dep_2015$CODSUBCOM)==2,
                               paste("0", don_dep_2015$CODSUBCOM, sep = ""),
                                      don_dep_2015$CODSUBCOM)), sep = "")
```

We also focus on the polling places corresponding to the chosen city :

```
sample_2015 <- don_dep_2015[intersect(grep(city_name_project, don_dep_2015$LIBSUBCOM),
                                      grep(city_name_project, don_dep_2015$LIBCAN)), ]
```

We look at the data :

```
sample_2015
```

```
## # A tibble: 1,801 x 15
##    NUMTOUR CODDPT CODSUBCOM LIBSUBCOM CODBURVOT CODCAN     LIBCAN NBRINS
##      <int>  <chr>     <chr>     <chr>     <chr>  <chr>      <chr>  <int>
##  1       1     31       555  Toulouse      0001     17 Toulouse-3   1093
##  2       1     31       555  Toulouse      0001     17 Toulouse-3   1093
##  3       1     31       555  Toulouse      0001     17 Toulouse-3   1093
##  4       1     31       555  Toulouse      0001     17 Toulouse-3   1093
##  5       1     31       555  Toulouse      0001     17 Toulouse-3   1093
##  6       1     31       555  Toulouse      0002     17 Toulouse-3    946
##  7       1     31       555  Toulouse      0002     17 Toulouse-3    946
##  8       1     31       555  Toulouse      0002     17 Toulouse-3    946
##  9       1     31       555  Toulouse      0002     17 Toulouse-3    946
## 10       1     31       555  Toulouse      0002     17 Toulouse-3    946
## # ... with 1,791 more rows, and 7 more variables: NBRVOT <int>,
## #   NBREXP <int>, NUMDEPCAND <int>, LIBLISEXT <chr>, CODNUA <chr>,
## #   NBRVOIX <int>, CODGEO <chr>
```

Note : this is raw data. Each line corresponds to a pair of candidates (variable LIBLISEXT), which represent a party (variable CODNUA) and the corresponding votes (variable CODNUA) into a polling place (variable CODBURVOT). It is also interesting to mention the variables NBINSC (number of registered electors in the polling place), NBRVOT (number of voters), NBEXP (number of voters who vote for a candidate). To obtain more informations about that election, visit https://fr.wikipedia.org/wiki/%C3%89lections_d%C3%A9partementales_fran%C3%A7aises_de_2015

Thus, the idea is to transform the raw data basis into a new data basis which contains infotmation at the polling places. For that, we first separate the results from the tour 1 and tour 2

```
sample_2015_T1 <- filter(sample_2015, NUMTOUR == 1)
sample_2015_T2 <- filter(sample_2015, NUMTOUR == 2)
```

We reshape the data :

```
sample_2015_T1_bv <- cast(sample_2015_T1[, c("CODDPT", "CODCAN", "CODGEO",
                                              "CODBURVOT", "NBRINS", "CODNUA", "NBRVOIX")],
                           CODDPT + CODCAN + CODGEO + CODBURVOT + NBRINS ~ CODNUA, sum)
```

```
## Using NBRVOIX as value column.  Use the value argument to cast to override this choice
```

We check that the number of polling place in the city chosen is the same than in the previous data basis :

```
nrow(sample_2015_T1_bv) == nrow(bv_sample)
```

```
## [1] TRUE
```

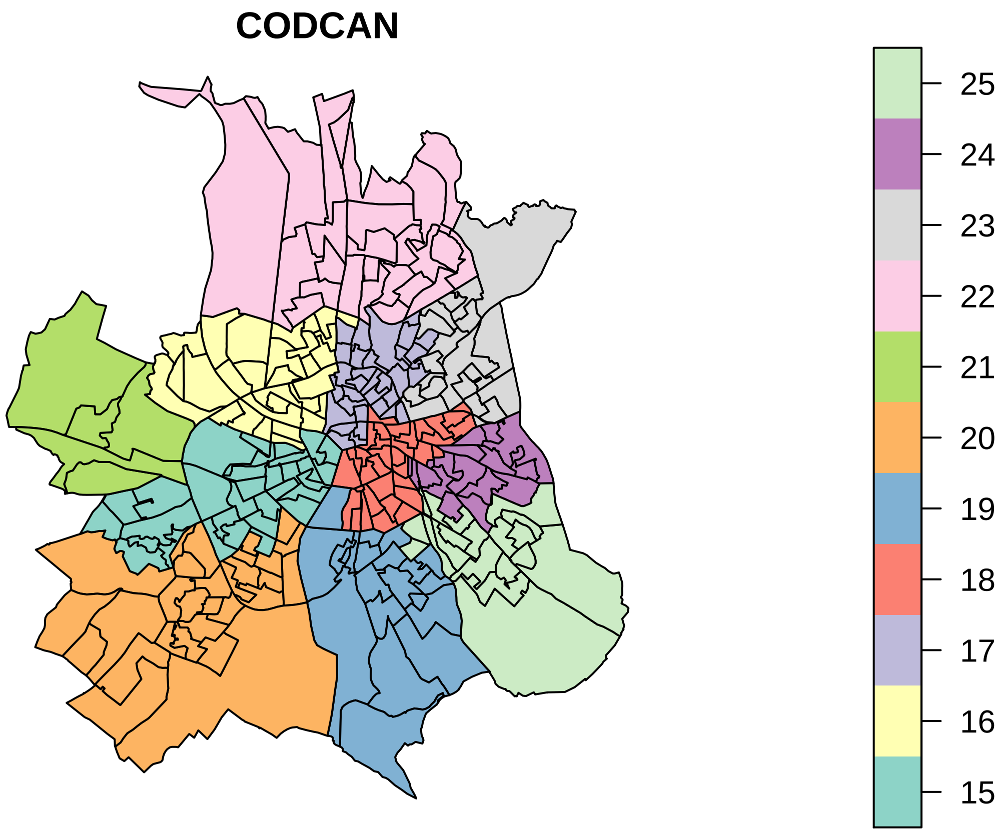
1.2.2 Merging

We now merge the geographical data with the election data.

```
bv_sample <- merge(bv_sample, sample_2015_T1_bv, by.x = "BUREAU", by.y = "CODBURVOT")
```

Verification : we plot the variable "CODCAN" (election data basis) associated to the spatial boundaries (cartelect databasis). If the levels of that variable are contiguous on the map, that would mean that the merging step was done correctly.

```
plot(bv_sample[, "CODCAN"])
```



1.2.3 Creation of new variables

We create the total number of voters by summing the results obtained by each party represented in the chosen city :

```
bv_sample$nb_voters <- bv_sample %>%
  st_set_geometry(NULL) %>%
  select(contains("BC.")) %>%
  rowSums(na.rm = TRUE)
```

Important remark : the list of the parties observed in one city is unique. Indeed, even if the main parties are usually represented in the biggest cities, it usually happens that local parties have candidates.

We create the percentage of radical right votes :

```
bv_sample$taux_fn <- bv_sample$BC.FN/bv_sample$nb_voters*100
```

We also create the percentage of turnout :

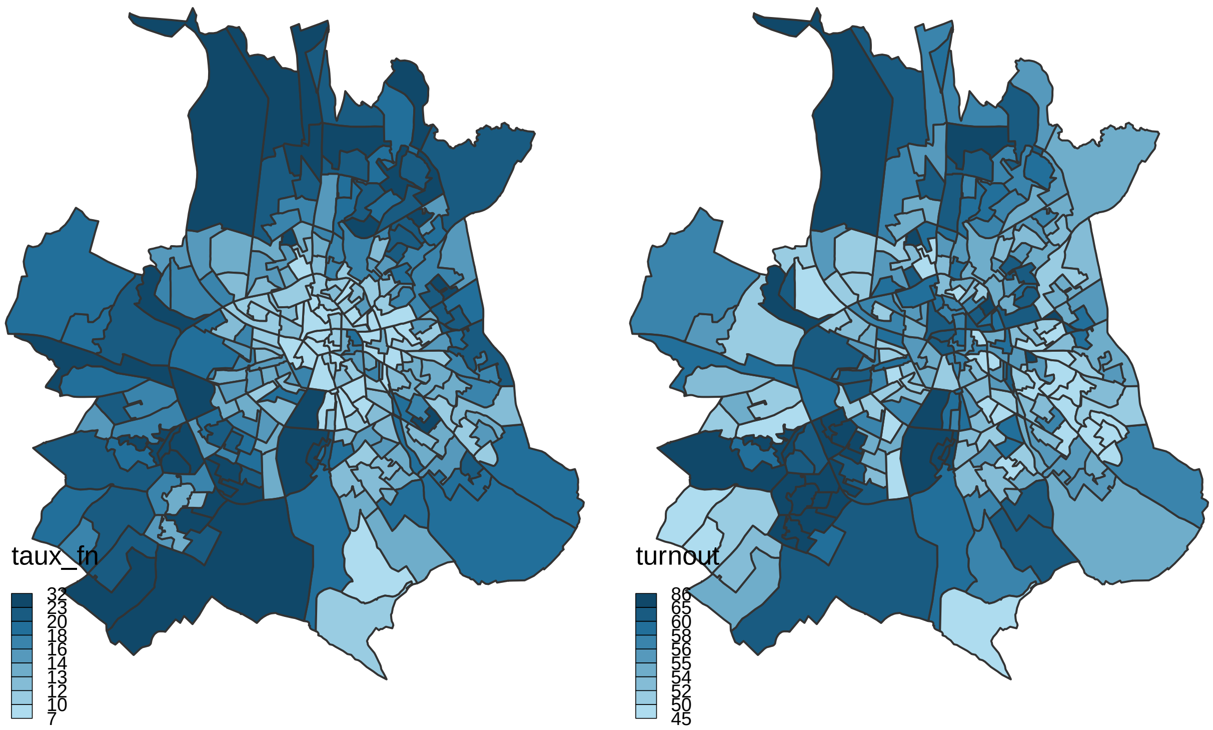
```
bv_sample$turnout <- (1 - bv_sample$nb_voters/bv_sample$NBRINS)*100
```

At this step, we have created these following variables :

- **taux_fn**: the depend variable,
- **turnout**: the percentage of turnout.

We map these 2 variables :

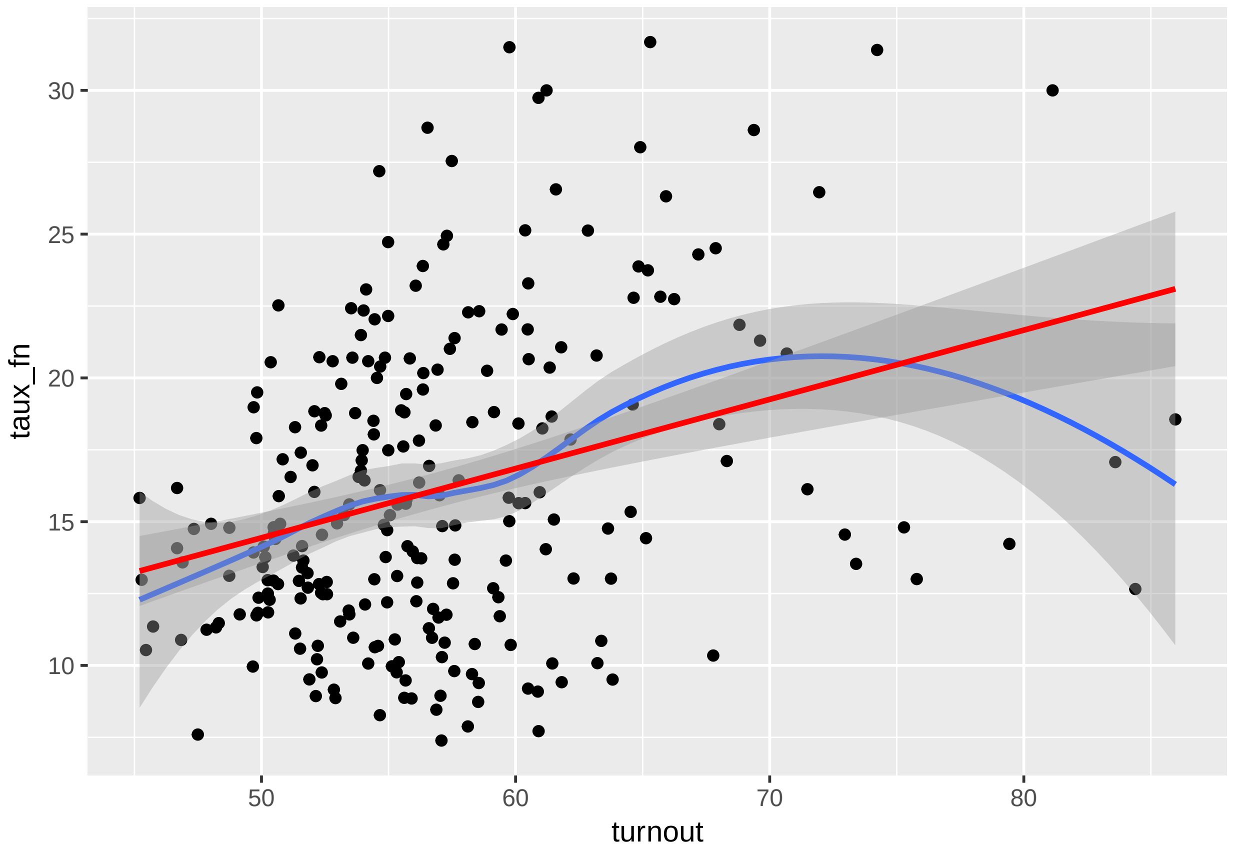
```
op <- par(mfrow = c(1, 2), oma = c(0, 0, 0, 0), mar = c(0, 0, 1.5, 0))
choroLayer(bv_sample, var = "taux_fn")
choroLayer(bv_sample, var = "turnout")
```



```
par(op)
```

We look at the relationship between these two variables :

```r
ggplot(bv_sample) +
  aes(x = turnout, y = taux_fn) +
  geom_point() +
  geom_smooth(method = "loess") +
  geom_smooth(method = "lm", col = "red")
```



We are now going to add socio-demographic variable provided by INSEE to our data basis.

1.3 The information at the iris level

Here, our objective is to build a data basis at the polling vote level with some socio-demographic variables, by using informations given by INSEE in the French census. However, the results of the French survey are not given at the polling vote level : indeed, INSEE publishes its results at the region, departement, commune or IRIS level, but not at the polling vote level. In this section, we import data from the INSEE.

1.3.1 Importation of the IRIS boundaries

First download the boundaries of the iris level-data at https://www.data.gouv.fr/fr/datasets/contour-des-iris-insee-tout-en-un/#_ (the size of this file is 350Mo) and import it into R :

```r
iris <- read_sf("iris-2013-01-01/iris-2013-01-01.shp")
```

We still continue to focus on the chosen city :

```r
iris_sample <- iris[iris$DEPCOM %in% unique(bv_sample$CODE), ]
```

We transform the data into the good CRS :

```r
iris_sample <- st_transform(iris_sample, 2154)
```

The number of iris in Tlse is:

```r
nrow(iris_sample)
```

```
## [1] 1530
```

Remark: this number of iris is wrong whatever the considered city (this can be checked on INSEE data basis). This is an issue due to the data basis which contain boundaries which appear several times. To takle this problem, we select here one boundary per iris:

```r
iris_name <- unique(iris_sample$NOM_IRIS)
ind_good <- numeric(length(iris_name))
for (k in 1:length(ind_good)) {
  ind_good[k] <- which(iris_sample$NOM_IRIS %in% iris_name[k])[1]
}
iris_sample <- iris_sample[ind_good,]
```

1.3.2 Iris INSEE data bases

In this section, we import the data from INSEE at the IRIS level. We can find these data bases at four different links :

- Housing data : https://www.insee.fr/fr/statistiques/3137421
- Professional situation https://www.insee.fr/fr/statistiques/3137415
- Diploma data basis : https://www.insee.fr/fr/statistiques/3137418
- Age : https://www.insee.fr/fr/statistiques/3137412

In our example, we selected only a few variables per data basis. To anwser to the initial problematic, students are strongly encouraged to select other potentials significant variables. The previous links give the definition of the set of variables.

1.3.2.1 Housing data

```r
df_logement <- read_excel("insee/base-ic-logement-2014.xls", skip = 5)
```

We keep the following variables :

- **P14__RP__LOC** : nombre de résidences principales occupées par des locataires,
- **P14__RP__LOCHLMV** : nombre de résidences principales HLM loué vide

and extract only the iris which concern the chosen city :

```r
df_logement_sample <- df_logement[df_logement$COM %in% unique(bv_sample$CODE),
                                  c("IRIS", "P14_RP_LOC", "P14_RP_LOCHLMV")]
```

We merge the INSEE data with the geographical data (iris):

```r
iris_sample_with_X <- merge(iris_sample, df_logement_sample,
                    by.x = "DCOMIRIS", by.y = "IRIS")
```

We remove the useless data :

```r
rm(df_logement, df_logement_sample)
```

1.3.2.2 Professional situation

We import the data :

```
df_activite <- read_excel("insee/base-ic-activite-residents-2014.xls", skip = 5)
```

We choose to keep the following variables :

- **P14_CHOM1564** : nombre de chômeurs de 15 à 64 ans,
- **P14_ACT1564** : nombre de personnes actives de 15 à 64 ans,
- **C14_ACT1564_CS1** : nombre d'agriculteurs exploitants actifs de 15 à 64 ans,
- **C14_ACT1564_CS2** : nombre d'artisans, commerçants, chefs d'entreprise actifs de 15 à 64 ans,
- **C14_ACT1564_CS3** : nombre de cadres et professions intellectuelles supérieures actifs de 15 à 64 ans,
- **C14_ACT1564_CS4** : nombre de professions intermédiaires actives de 15 à 64 ans,
- **C14_ACT1564_CS5** : nombre d'employés actifs de 15 à 64 ans,
- **C14_ACT1564_CS6** : nombre d'ouvriers actifs de 15 à 64 ans

and extract only the iris which concern the chosen city :

```
df_activite_sample <- df_activite[df_activite$COM %in% unique(bv_sample$CODE),
                                 c("IRIS", "P14_CHOM1564", "P14_ACT1564",
                                   paste("C14_ACT1564_CS", 1:6, sep=""))]
```

We merge the INSEE data with the geographical data (iris):

```
iris_sample_with_X <- merge(iris_sample_with_X, df_activite_sample,
                   by.x = "DCOMIRIS", by.y = "IRIS")
```

We remove the useless data :

```
rm(df_activite, df_activite_sample)
```

1.3.2.3 Diploma data basis

We import the data :

```
df_diplome <- read_excel("insee/base-ic-diplomes-formation-2014.xls", skip = 5)
```

We choose to keep the following variables :

- **P14_NSCOL15P** : nombre de personnes non scolarisées de 15 ans ou plus
- **P14_NSCOL15P_DIPLMIN** : nombre de personnes non scolarisées de 15 ans ou plus titulaires d'aucun diplôme ou au plus un BEPC, brevet des collèges ou DNB
- **P14_NSCOL15P_CAPBEP** : nombre de personnes non scolarisées de 15 ans ou plus titulaires d'un CAP ou d'un BEP
- **P14_NSCOL15P_BAC** : nombre de personnes non scolarisées de 15 ans ou plus titulaires d'un baccalauréat (général, technologique, professionnel)
- **P14_NSCOL15P_SUP** : nombre de personnes non scolarisées de 15 ans ou plus titulaires d'un diplôme de l'enseignement supérieur

We focus on Tlse only

```
df_diplome_sample <- df_diplome[df_diplome$COM %in% unique(bv_sample$CODE),
                                c("IRIS", "P14_NSCOL15P", "P14_NSCOL15P_DIPLMIN",
                                  "P14_NSCOL15P_CAPBEP", "P14_NSCOL15P_BAC", "P14_NSCOL15P_SUP")]
```

We merge the INSEE data with the geographical data (iris):

```
iris_sample_with_X <- merge(iris_sample_with_X, df_diplome_sample,
                          by.x = "DCOMIRIS", by.y = "IRIS")
```

We remove the useless data :

```
rm(df_diplome, df_diplome_sample)
```

1.3.2.4 Age data basis

```
df_age <- read_excel("insee/base-ic-couples-familles-menages-2014.xls", skip = 5)
```

We choose to keep the following variables :

- **P14_POPMEN15P** : nombre de personnes des ménages de 15 ans ou plus
- **P14_POPMEN1524** : nombre de personnes des ménages de 15 à 24 ans
- **P14_POPMEN2554** : nombre de personnes des ménages de 25 à 54 ans
- **P14_POPMEN5579** : nombre de personnes des ménages de 55 à 79 ans
- **P14_POPMEN80P** : nombre de personnes des ménages de 80 ans ou plus

We focus on Tlse only :

```
df_age_sample <- df_age[df_age$COM %in% unique(bv_sample$CODE),
                             c("IRIS", "P14_POPMEN15P", "P14_POPMEN1524",
                               "P14_POPMEN2554", "P14_POPMEN5579", "P14_POPMEN80P")]
```

We merge the INSEE data with the geographical data (iris):

```
iris_sample_with_X <- merge(iris_sample_with_X, df_age_sample,
                            by.x = "DCOMIRIS", by.y = "IRIS")
```

We remove the useless data :

```
rm(df_age, df_age_sample)
```

We transform **sf** object into **Spatial** object :

```
iris_sample_with_X_spatial <- as(iris_sample_with_X, "Spatial")
bv_sample_spatial <- as(bv_sample, "Spatial")
```

Since we got the information at the IRIS level, we use in the next section Areal Interpolation Methods to estimate socio-demographic variables at the polling vote level by using information provided by INSEE at the IRIS level.

1.4 Areal Interpolation Methods

Objective: we have information at the IRIS levels that we call Sources and we want to estimate that information at the polling places level that we call Targets. For doing that, we use Areal Interpolation Methods. To have an overview on existing methods on that topic, we refer the students to the thesis of Van Huyen Do (http://publications.ut-capitole.fr/18749/1/DoVanHuyen2015.pdf).

In this section, we are going to use the following two methods :

- DAW : Areal weighting interpolation,
- DAX : Ordinary dasymetric weighting.

The programs are given at this link AIM.R

```
source("AIM.R")
```

Before using this method, we are going to import an auxiliary information provided by INSEE at the very small level : a grid made of cells of size $200m \times 200m$ which give the population observed in each cell. This information will be used in the Areal Interpolation Methods to obtain better estimates.

1.4.1 Import population data at the grid level

The population data is given by INSEE at the grid data with cell of size $200m \times 200m$. This data basis can be downloaded at this link : https://www.insee.fr/fr/statistiques/fichier/2520034/200m-carreaux-metropole.zip (size of the file : 86Mo).

We import the data (this operation could take a few minutes) :

```
grid_rp <- st_read(dsn = "200m-carreaux-metropole/car_m.mif")
```

```
## Reading layer `car_m' from data source `/home/laurent/Documents/M2/2017-2018/projet_christine/200m-ca
## Simple feature collection with 2278213 features and 2 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 48385.79 ymin: 1620790 xmax: 1197778 ymax: 2676806
## epsg (SRID):    NA
## proj4string:    NA
```

We select the good CRS :

```
st_crs(grid_rp) <- 27572
```

We transform the data into another CRS :

```
grid_rp_2154 <- st_transform(grid_rp, "+init=epsg:2154")
```

We only select the grid falling into the chosen city :

```
ind_sample <- st_intersects(iris_sample_with_X, grid_rp_2154)
grid_sample <- grid_rp_2154[unique(unlist(ind_sample)), ]
```

We import the population data observed at the gridded level :

```
car_db <- read.dbf("200m-carreaux-metropole/car_m.dbf")
grid_sample <- merge(grid_sample, car_db, all.x = T)
```

We delete the useless data :

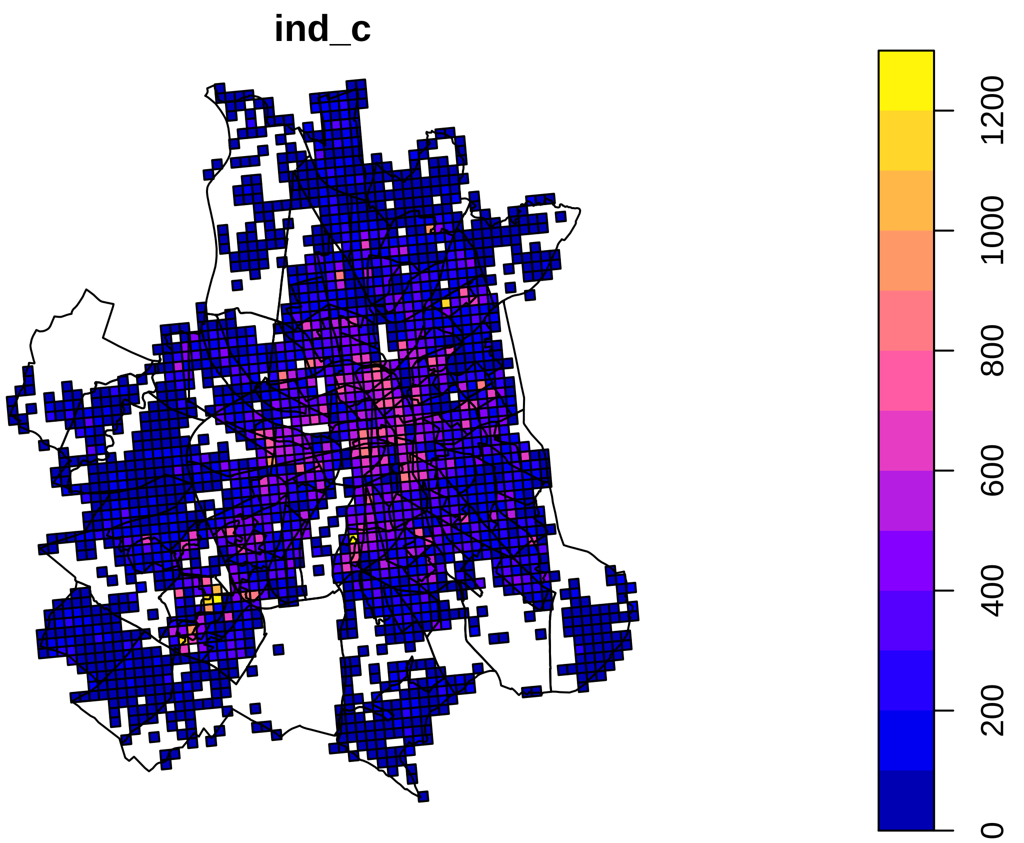
```
rm(grid_rp)
rm(car_db)
```

We transform the **sf** object into a **Spatial** object :

```
grid_sample_spatial <- as(grid_sample, "Spatial")
```

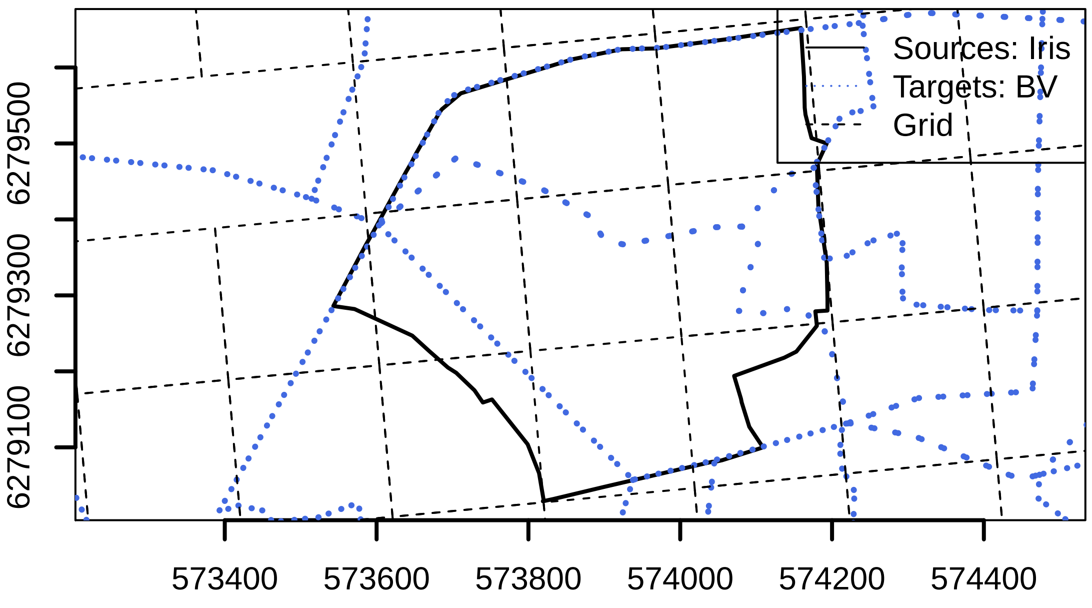
We plot the the population data in the chosen city :

```
plot(grid_sample[, "ind_c"])
plot(iris_sample_with_X_spatial, add = T, col = NULL)
```



1.4.2 Illustration of our problem

We have information at the sources level (data from INSEE at the iris level) and we would like to estimate this information at the targets level (polling place) :



1.4.3 Method 1 : DAW, proportionnal to area

We use the function *daw()* (including in AIM.R) :

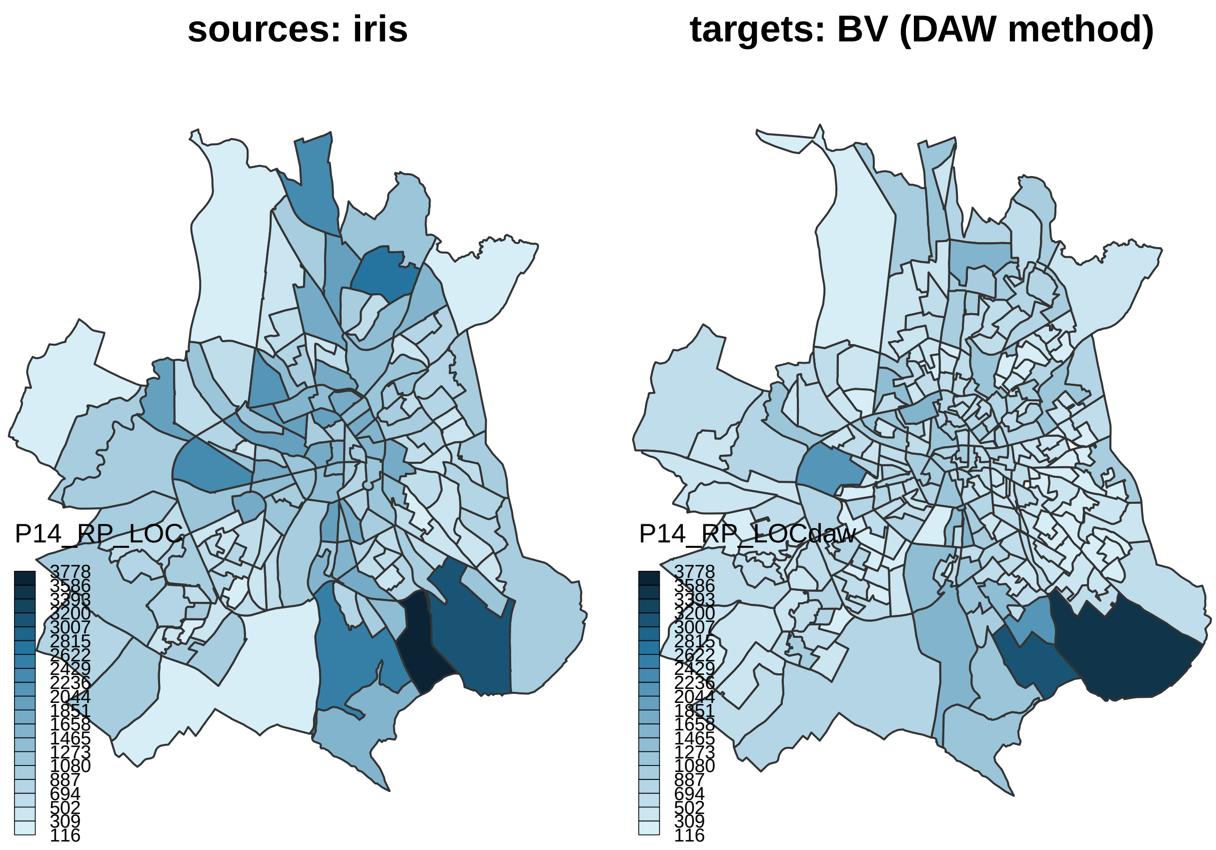
```
bv_sample_spatial_with_X <- daw(sources = iris_sample_with_X_spatial,
                                targets = bv_sample_spatial,
                                y = c("P14_RP_LOC", "P14_RP_LOCHLMV", "P14_CHOM1564", "P14_ACT1564",
                                  "C14_ACT1564_CS1", "C14_ACT1564_CS2", "C14_ACT1564_CS3",
                                  "C14_ACT1564_CS4", "C14_ACT1564_CS5", "C14_ACT1564_CS6",
                                  "P14_NSCOL15P", "P14_NSCOL15P_DIPLMIN", "P14_NSCOL15P_CAPBEP",
                                  "P14_NSCOL15P_BAC", "P14_NSCOL15P_SUP", "P14_POPMEN15P", "P14_POPMEN1
                                  "P14_POPMEN2554", "P14_POPMEN5579", "P14_POPMEN80P"),
                                nature = "extensive", scaling = F)
```

```
## Warning in daw(sources = iris_sample_with_X_spatial, targets =
## bv_sample_spatial, : Sources and Targets do not have the same CRS
```

```
## Warning in RGEOSBinTopoFunc(spgeom1, spgeom2, byid, id, drop_lower_td,
## unaryUnion_if_byid_false, : spgeom1 and spgeom2 have different proj4
## strings
```

We plot the results for the variable **P14_RP_LOC** :

```
op <- par(mfrow = c(1, 2), oma = c(0, 0, 0, 0), mar = c(0, 0, 1.5, 0))
choroLayer(iris_sample_with_X, var = "P14_RP_LOC",
           breaks = seq(min(iris_sample_with_X$P14_RP_LOC),
                        max(iris_sample_with_X$P14_RP_LOC),
                        length.out = 20))
title("sources: iris")
choroLayer(spdf = bv_sample_spatial_with_X, var = "P14_RP_LOCdaw",
           breaks = seq(min(iris_sample_with_X$P14_RP_LOC),
                        max(iris_sample_with_X$P14_RP_LOC),
                        length.out = 20))
title("targets: BV (DAW method)")
```



```r
par(op)
```

1.4.4 Method 2 : DAX with variable population

First, we have to know the population at the sources, targets and intersection levels. For obtaining this information, we use the information at the grid level.

We define the intersection between sources and targets :

```r
st_inter <- intersect.spdf(sources = iris_sample_with_X_spatial,
                           targets = bv_sample_spatial,
                           out = "spdf")
```

```
## Warning in RGEOSBinTopoFunc(spgeom1, spgeom2, byid, id, drop_lower_td,
## unaryUnion_if_byid_false, : spgeom1 and spgeom2 have different proj4
## strings
```

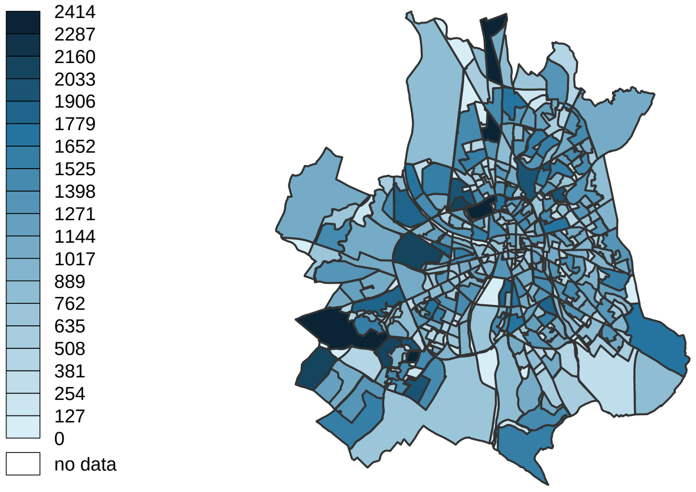
```r
row.names(st_inter) <- as.character(1:length(st_inter))
```

We apply a DAW method to obtain the estimated population at the intersection level :

```r
daw_pop_intersect <- daw(sources = grid_sample_spatial,
                         targets = st_inter,
                         y = "ind_c",
                         nature = "extensive", scaling = F)
```

We obtain this estimation at the intersection level:

```r
choroLayer(spdf = daw_pop_intersect, var = "ind_cdaw",
           breaks = seq(min(daw_pop_intersect$ind_cdaw, na.rm = T),
                        max(daw_pop_intersect$ind_cdaw, na.rm = T),
                        length.out = 20))
```



There are some missing values which we replace by 0 values :

```r
daw_pop_intersect@data$ind_cdaw[is.na(daw_pop_intersect@data$ind_cdaw)] <- 0
```

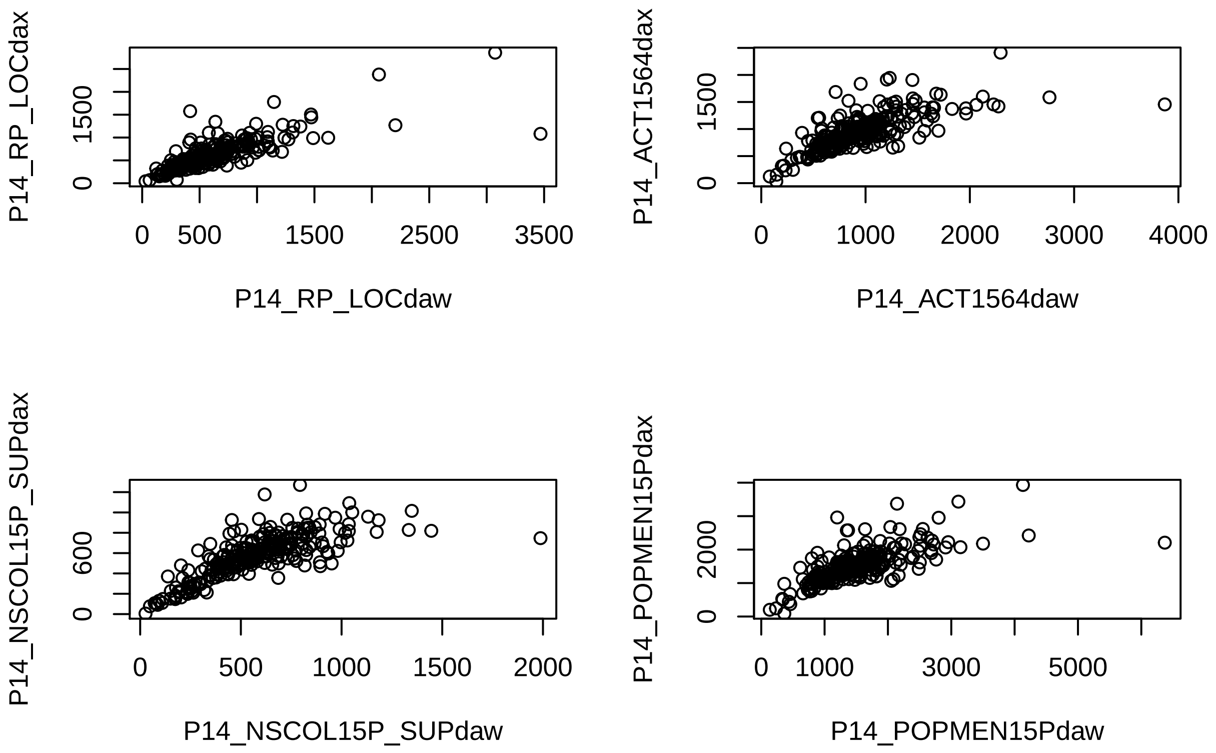
We finally use the DAX method :

```r
bv_sample_spatial_with_X <- dax(sources = iris_sample_with_X_spatial,
                                targets = bv_sample_spatial_with_X,
                                y = c("P14_RP_LOC", "P14_RP_LOCHLMV", "P14_CHOM1564", "P14_ACT1564",
                                    "C14_ACT1564_CS1", "C14_ACT1564_CS2", "C14_ACT1564_CS3", "C14_ACT15
                                    "C14_ACT1564_CS5", "C14_ACT1564_CS6", "P14_NSCOL15P", "P14_NSCOL15P_
                                    "P14_NSCOL15P_CAPBEP", "P14_NSCOL15P_BAC", "P14_NSCOL15P_SUP", "P14_
                                    "P14_POPMEN1524", "P14_POPMEN2554", "P14_POPMEN5579", "P14_POPMEN80
                                st.df = daw_pop_intersect@data,
                                x = "ind_cdaw",
                                scaling = F)
```

```
## Warning in dax(sources = iris_sample_with_X_spatial, targets =
## bv_sample_spatial_with_X, : Sources and Targets do not have the same CRS
```

Comparaison of DAW and DAX :

```r
op <- par(mfrow = c(2, 2))
plot(P14_RP_LOCdax ~ P14_RP_LOCdaw , data = bv_sample_spatial_with_X)
plot(P14_ACT1564dax ~ P14_ACT1564daw , data = bv_sample_spatial_with_X)
plot(P14_NSCOL15P_SUPdax ~ P14_NSCOL15P_SUPdaw , data = bv_sample_spatial_with_X)
plot(P14_POPMEN15Pdax ~ P14_POPMEN15Pdaw , data = bv_sample_spatial_with_X)
```



```r
par(op)
```

The two methods are usually correlated. In our case, we decide to keep estimates ontained the with DAX method.

1.5 Final data basis

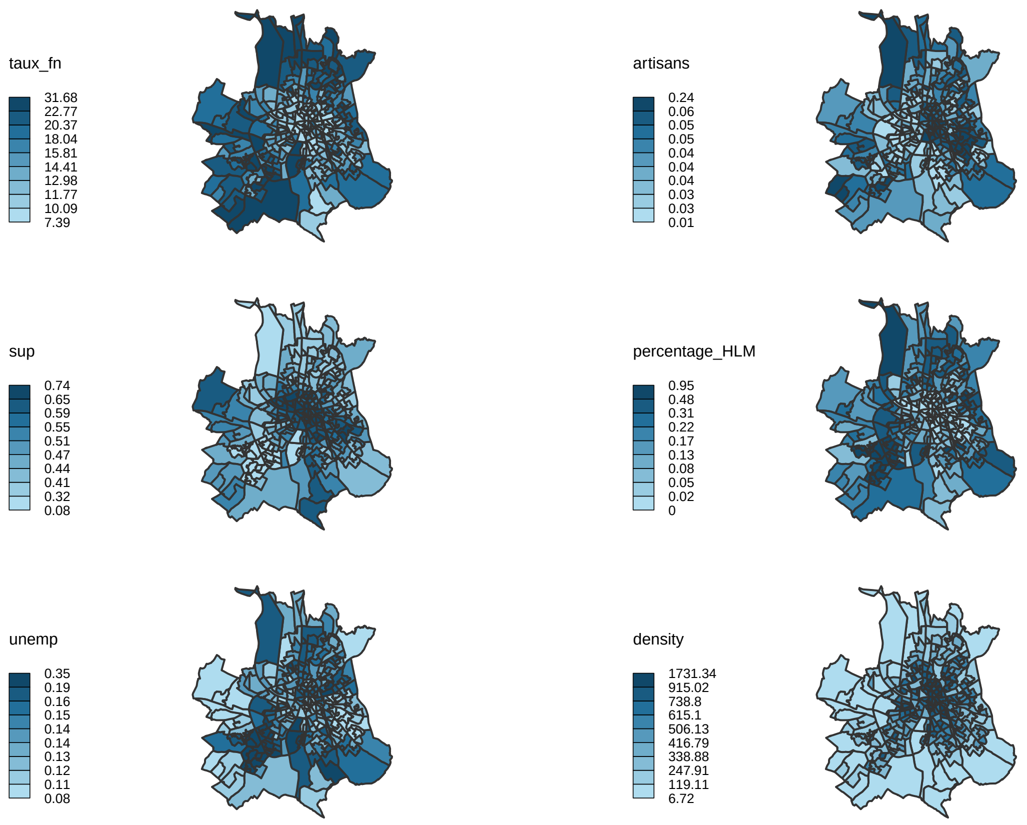
We created the following variables thanks to the estimates obtained with the DAX method :

- variable percentage of HLM located,
- variable percentage of unemployement,
- variable percentage of CSP “artisan commercant”,
- variable percentage of “cadre sup”,
- variable population density

```r
bv_sample_spatial_with_X$percentage_HLM <- bv_sample_spatial_with_X$P14_RP_LOCHLMVdax/
  bv_sample_spatial_with_X$P14_RP_LOCdax
bv_sample_spatial_with_X$unemp <- bv_sample_spatial_with_X$P14_CHOM1564dax/
  (bv_sample_spatial_with_X$P14_CHOM1564dax + bv_sample_spatial_with_X$P14_ACT1564dax)
bv_sample_spatial_with_X$artisans <- bv_sample_spatial_with_X$C14_ACT1564_CS2dax/
  bv_sample_spatial_with_X$P14_ACT1564dax
bv_sample_spatial_with_X$sup <- bv_sample_spatial_with_X$P14_NSCOL15P_SUPdax/
  bv_sample_spatial_with_X$P14_NSCOL15Pdax
bv_sample_spatial_with_X$density <- bv_sample_spatial_with_X@data$NBRINS/
  gArea(bv_sample_spatial_with_X, byid = T)*10^5
```

We look at the plot :

```r
op <- par(mfrow = c(3, 2), oma = c(0, 0, 0, 0), mar = c(0, 0, 1.5, 0))
choroLayer(spdf = bv_sample_spatial_with_X, var = "taux_fn", legend.values.rnd = 2)
choroLayer(spdf = bv_sample_spatial_with_X, var = "artisans", legend.values.rnd = 2)
choroLayer(spdf = bv_sample_spatial_with_X, var = "sup", legend.values.rnd = 2)
choroLayer(spdf = bv_sample_spatial_with_X, var = "percentage_HLM", legend.values.rnd = 2)
choroLayer(spdf = bv_sample_spatial_with_X, var = "unemp", legend.values.rnd = 2)
choroLayer(spdf = bv_sample_spatial_with_X, var = "density", legend.values.rnd = 2)
```



```r
par(op)
```

At this step, we recommand to save the data into a ".RData" file.

```r
save(bv_sample_spatial_with_X, file = "spatial_ecoX.RData")
```

For the spatial econometric analysis, students should load the data directly from the ".RData" file rather than compiling all the previous code.

```r
load("spatial_ecoX.RData")
```