Guidelines on areal interpolation methods (supplementary material)

V. H. Do, T. Laurent, A. Vanhems

Last update: 2020-10-20

Contents

1 The data **2**
1.1 Targets . . . 2
1.2 1st kind of sources: data at the cell level . . . 6
1.3 2nd kind of sources: data at the iris level . . . 10
1.4 Auxiliary information . . . 12
1.5 Figure 1 in the book chapter . . . 13

2 Point-in-polygon method **16**
2.1 Illustration . . . 16
2.2 Extensive variables . . . 17
2.3 Intensive variables . . . 20
2.4 Limitation of the point-in polygon method . . . 24

3 Areal weighting interpolation (DAW) method **25**
3.1 Extensive variable . . . 25
3.2 Comparaison between PIP and DAW . . . 28
3.3 Intensive variable . . . 30

4 Dasymetric method with auxiliary variable X (DAX) **36**
4.1 Extensive variables . . . 36
4.2 Comparaison between DAW and DAX . . . 38
4.3 Intensive variables . . . 40
4.4 Comparaison between DAX and DAW . . . 44

5 Dasymetric method with control zones **45**
5.1 Step 1: DAW method . . . 45
5.2 Step 2: DAX method . . . 46

6 Regression Modelling **49**
6.1 Construction of the covariates . . . 49
6.2 Exploratory Analysis . . . 51
6.3 Regression modelling . . . 53

Packages needed:

```
install.packages(c("sf", # spatial data structure
                   "tidyverse", # tidyverse universe
                   "gridExtra", # ggplot2 extra plot
                   "ggspatial",  # plot fanzy map
                   "readxl", # import excel files
                   "corrplot", # correlation plot
```

```
                    "rpart", # regression tree
                    "rattle" # fanzy regression tree plot
                    )
)
```

```
library(ggspatial)
library(readxl)
library(sf)
library(tidyverse)
library(corrplot)
library(rattle)
library(rpart)
```

This document presents the **R** codes used to obtain the computational results included in the chapter book. It also contains tables and additional graphs which were not included in the chapter book.

1 The data

The results of the election are released by the Ministry of Interior in open access (source: https://www.data.gouv.fr/fr/datasets/elections-departementales-2015-resultats-par-bureaux-de-vote/) at the polling place scale.

1.1 Targets

Our targets are the polling place in the 2015'Departemental elections. To obtain the administrative boundaries of the polling place, we use the maps provided by the Cartelec project (http://cartelec.univ-rouen.fr/).

We will focus here on the polling places in Toulouse, but our codes could be used for any other regions.

```
# we define the name of the city
city_name_chapter <- "Toulouse"

# import the data
bv <- read_sf("./data/BV_grandes_villes_2015/BV_grandes_villes_2015.shp")

# We only keep the variables **NOM** and **BUREAU** in this database
bv <- select(bv, CODE, NOM, BUREAU)

# We extract the rows corresponding to the chosen city :
bv_sample <- bv[grep(city_name_chapter, 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 = ""))
```

There are $T = 256$ targets (polling places) in Toulouse. We print the summary statistics of the area.

```
summary(st_area(bv_sample))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   23846  136342  209436  464303  395699 8164842
```

The same thing can be done for the other geometries. Table 1 presents the areas of sources and targets (the code used to obtain the Table is presented hereafter, once all the data have been imported).

Table 1: Statistics on the areas (in m^2) with respect to sources and targets

	Min	Q1	Median	Mean	Q3	Max
Polling places (targets)	23 846	136 342	209 436	464 303	395 699	8 164 842
Small cells (sources)	40 021	40 024	40 026	40 026	40 027	40 030
Big cells (sources)	40 021	40 025	40 026	277 540	40 028	4 002 961
Iris (sources)	79 843	237 291	344 533	770 352	725 516	8 326 068

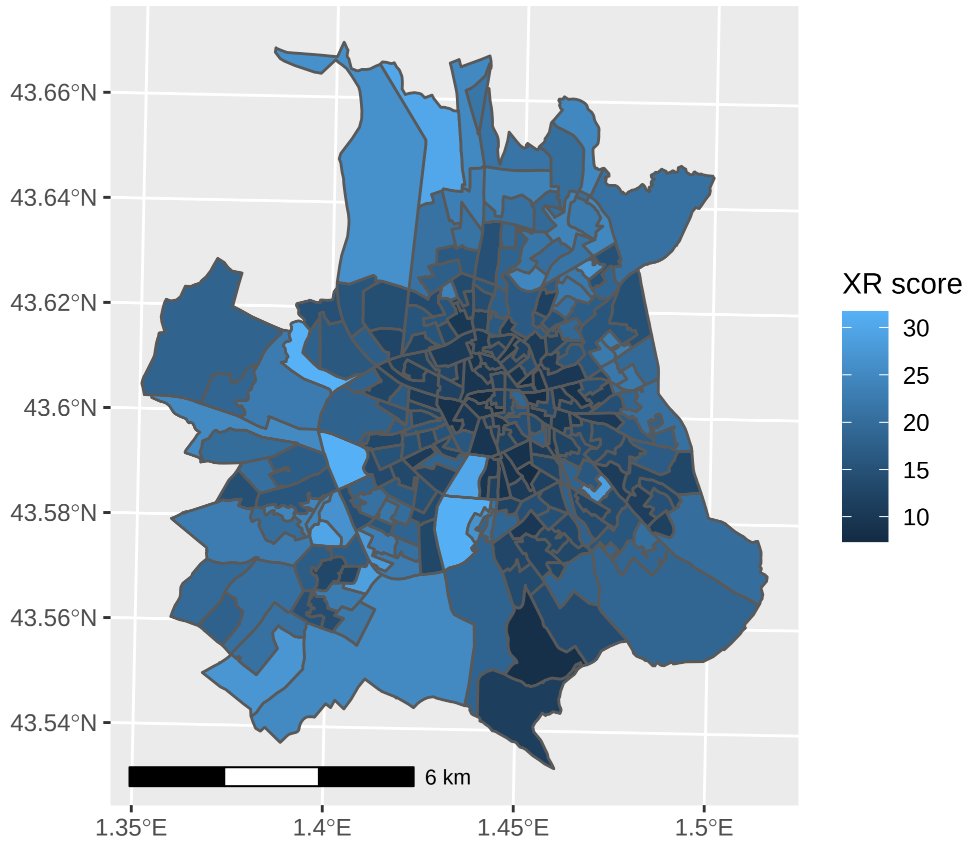
The next step is to associate the "Departementales 2015" election results to that geographical data basis. We import the "Departementales 2015" election results directly from that link https://www.data.gouv.fr/fr/datasets/elections-departementales-2015-resultats-par-bureaux-de-vote/ :

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

# We properly clean up 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_chapter, don_dep_2015$LIBSUBCOM),
                                      grep(city_name_chapter, don_dep_2015$LIBCAN)), ]

# 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 <- sample_2015_T1 %>%
  select(CODDPT, CODCAN, CODGEO, CODBURVOT, NBRINS, CODNUA, NBRVOIX) %>%
  pivot_wider(names_from = "CODNUA",
              values_from = "NBRVOIX")

# We verify that the number of polling place in the chosen city is the same
# as in the previous data basis:
# nrow(sample_2015_T1_bv) == nrow(bv_sample)

# We change the name of some variables:
ind_BC <- grep("BC", names(sample_2015_T1_bv))
names(sample_2015_T1_bv)[ind_BC] <- sapply(strsplit(names(sample_2015_T1_bv)[ind_BC], "-"),
                                           function(x) paste(x, collapse = "_"))

# 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")

# We determine the total number of voters by summing the results obtained by each party represented in
bv_sample$nb_voters <- bv_sample %>%
  st_set_geometry(NULL) %>%
  select(contains("BC_")) %>%
  rowSums(na.rm = TRUE)

# We determine the percentage of radical right votes and the percentage of turnout
bv_sample <- bv_sample %>%
  mutate(taux_fn = BC_FN / nb_voters * 100,
         turnout = (1 - nb_voters / NBRINS) * 100)

# Define the CRS
st_crs(bv_sample) <- "+proj=lcc +lat_1=49 +lat_2=44 +lat_0=46.5 +lon_0=3
                      +x_0=700000 +y_0=6600000 +ellps=GRS80 +units=m +no_defs"
```

At this step, we have created the following variables that can be represented in maps (not presented in the chapter of the book).

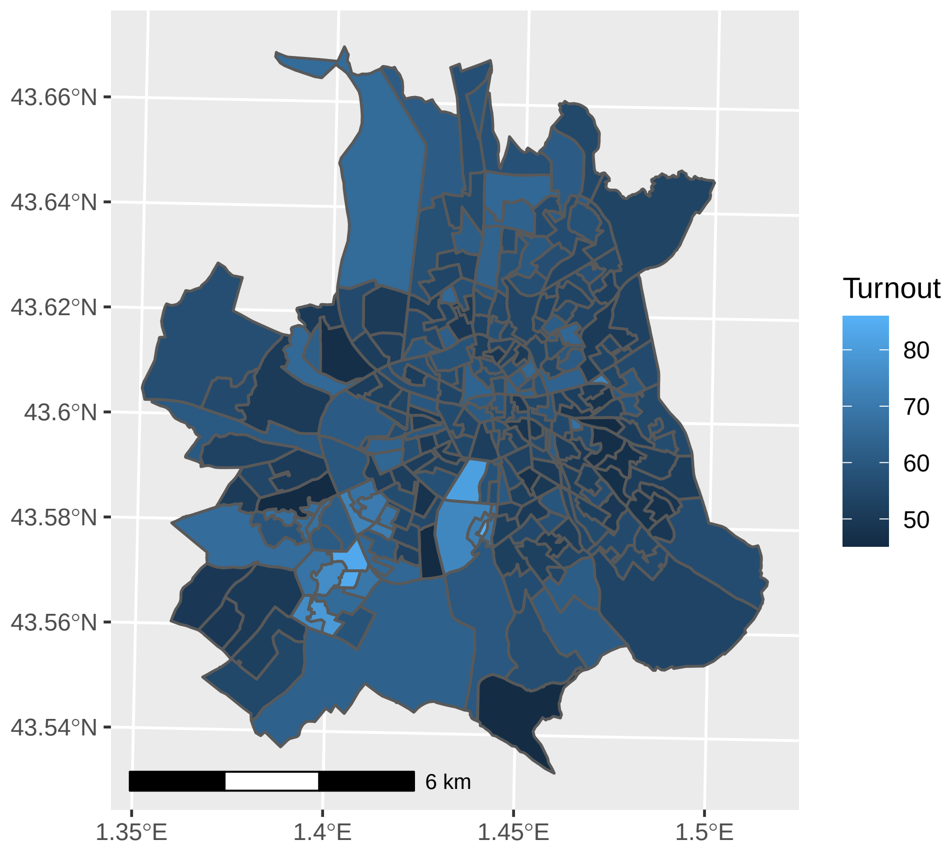
- **taux_fn**: the dependent variable,

```
ggplot(data = bv_sample)  +
  labs(fill = "XR score") +
  geom_sf(aes(fill = taux_fn)) +
    annotation_scale(location = "bl", width_hint = 0.5)
```



- **turnout**: the percentage of turnout.

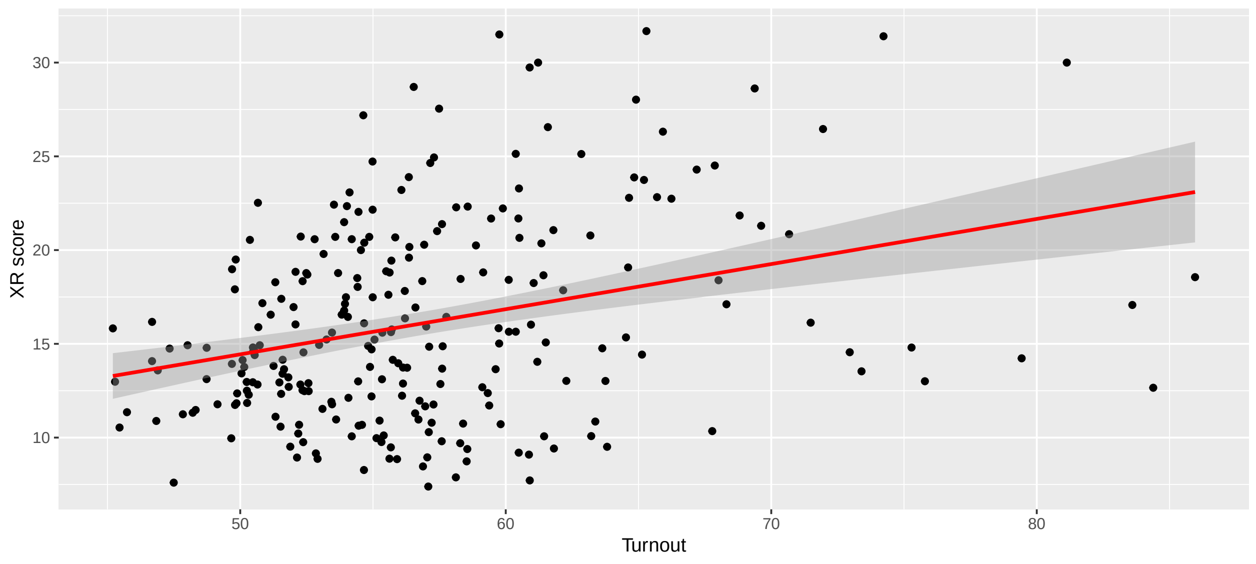
```
ggplot(data = bv_sample) +
  labs(fill = "Turnout") +
  geom_sf(aes(fill = turnout)) +
    annotation_scale(location = "bl", width_hint = 0.5)
```



We also represent the scatter plot of the two variables, and it seems that the turnout has a positive effect on the extreme right score:

```
ggplot(data = bv_sample) +
  aes(x = turnout, y = taux_fn) +
  geom_point() +
  geom_smooth(method = "lm",
              col = "red")  +
  labs(x = "Turnout", y = "XR score")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Summary statistics of the two variables are presented in Table 2.

```r
stargazer::stargazer(round(t(sapply(st_drop_geometry(bv_sample[,
    c("taux_fn", "turnout")]), function(x) c(mean(x), sd(x), min(x), max(x)))), 2))
```

Table 2: Summary statistics of the variables observed at the target levels: extreme right score (dependent variable) and turnout (explanatory variable)

Variable	Description	Mean	Sd	Min	Max
taux_fn	Extreme Right vote (in percentage)	16.09	5.29	7.39	31.68
turnout	Turnout (in percentage)	56.86	6.90	45.20	85.96

1.2 1st kind of sources: data at the cell level

The INSEE data are obtained thanks to the income tax files (source: https://www.insee.fr/fr/statistiques/4176281?sommaire=4176305). There are two different levels of cells.

- The finest level is the cell of dimension $200m \times 200m$. There are $S = 2\ 027$ different sources. The areas of the cells are all the same and equal to $40\ 000m^2$.

```r
cell_200m <- read_sf("./data/Filosofi2015/Filosofi2015_carreaux_200m_metropole.shp")

# select the cells falling in Toulouse
cell_200m <- cell_200m %>%
  filter(Depcom == "31555") %>%
  select(-Id_carr1km, -Id_carr_n, -Id_car2010, -Groupe, -Depcom)

# change the crs
cell_200m <- st_transform(cell_200m, st_crs(bv_sample))
```

- The second level is more aggregated so that the sources have enough inhabitants to solve the confidentiality problems. The number of sources is equal in that case to $S = 591$, and the cells have different sizes (473 cells of size $200m \times 200m$, 109 cells of size $1000m \times 1000m$ and 9 cells of size $2000m \times 2000m$):

```r
cell_big <- read_sf("./data/Filosofi2015_naturel/Filosofi2015_carreaux_niveau_naturel_metropole.shp")
```

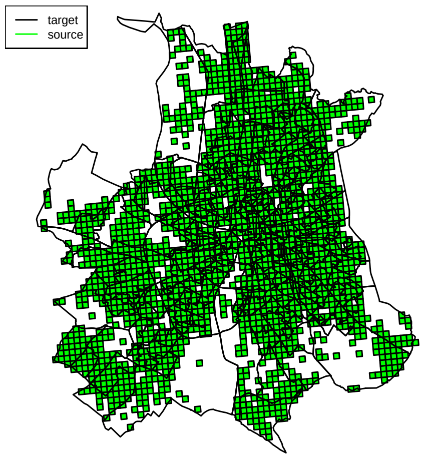


Figure 1: Sources at the finest level

```
# change the crs
cell_big <- st_transform(cell_big, st_crs(bv_sample))

ind_touches <- st_intersects(cell_big, bv_sample)
ind_touches <- apply(ind_touches, 1, any)
cell_big <- cell_big[ind_touches, ]
# change the crs
cell_big <- st_transform(cell_big, st_crs(bv_sample))
```

1.2.1 Which levels should we choose?

For the small cells, the total area is equal to :

```
sum(st_area(cell_200m))
```

```
## 81132570 [m^2]
```

whereas in the case of the large cells, it is equal to:

```
sum(st_area(cell_big))
```

```
## 164026135 [m^2]
```

The total number of inhabitants observed on the small cells is equal to :

```
sum(cell_200m$Ind)
```

```
## [1] 404497
```

whereas in the case of the large cells, it is equal to:

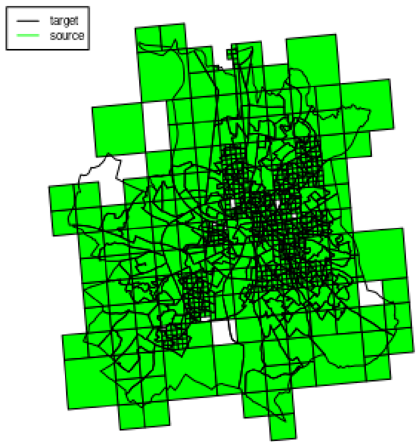


Figure 2: Sources at the aggregated level

```
sum(cell_big$Ind)
```

```
## [1] 457031.5
```

1.2.2 Variables observed

The variables of interest provided by the INSEE are presented in Table 3.

```
stargazer::stargazer(round(t(sapply(st_drop_geometry(cell_200m[, c("Ind", "Men",
  "Men_pauv", "Men_prop", "Men_coll", "Men_mais", "Log_ap90", "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")]),
  function(x) c(mean(x), sd(x), min(x), max(x)))), 0))
```

The intensive variables that we consider are:

- **prop_Ind_under_18**, the proportion of inhabitants under 18 years old, which is the ratio of the two extensive variables : **Ind_0_17** (number of inhabitants less than 18 years old) and **Ind** (number of inhabitants).
- **pop_dens**, the population density which corresponds to the number of inhabitants **Ind** divided by the area of the source **area** that we obtain with the variable **t_maille** which contains the length of a cell.

Here, we create the variables **Ind_0_17**, **area**, **prop_Ind_under_18** and **pop_dens**:

```
cell_200m$area <- as.numeric(st_area(cell_200m)) / 1000 ^ 2
cell_big$area <- as.numeric(st_area(cell_big)) / 1000 ^ 2
cell_200m <- cell_200m %>%
  mutate(
    Ind_0_17 = (Ind_0_3 + Ind_4_5 + Ind_6_10 + Ind_11_17),
```

Table 3: Summary statistics of variables of interest (target variables) provided by INSEE at the small cells level

Extensive variables	Description	Mean	Sd	Min	Max
Ind	Number of individuals	200	199	1	1 700
Men	Number of households	101	107	0	909
Men_pauv	Number of poor households	19	26	0	227
Men_prop	Number of owner households	39	43	0	477
Men_coll	Number of households in collective dwellings	82	106	0	909
Men_mais	Number of house households	19	20	0	104
Log_ap90	Number of dwellings built since 1990	37	60	0	614
Log_soc	Number of social housing	17	43	0	378
Ind_0_3	Number of individuals aged 0 to 3	11	13	0	106
Ind_4_5	Number of individuals aged 4 to 5	5	6	0	55
Ind_6_10	Number of individuals 6 to 10 years old	11	13	0	124
Ind_11_17	Number of individuals aged 11 to 17	13	15	0	156
Ind_18_24	Number of individuals aged 18 to 24	15	18	0	126
Ind_25_39	Number of individuals aged 25 to 39	56	65	0	579
Ind_40_54	Number of individuals aged 40 to 54	37	36	0	322
Ind_55_64	Number of individuals aged 55 to 64	20	20	0	173
Ind_65_79	Number of individuals aged 65 to 79	19	22	0	177
Ind_80p	Number of individuals aged 80 and above	10	13	0	109
Ind_inc	Number of individuals whose age is unknown	4	5	0	34

```r
    prop_Ind_under_18 = Ind_0_17 / Ind,
    pop_dens = Ind / area)
cell_big <- cell_big %>%
  mutate(
    Ind_0_17 = (Ind_0_3 + Ind_4_5 + Ind_6_10 + Ind_11_17),
    prop_Ind_under_18 = Ind_0_17 / Ind,
    pop_dens = Ind / area)
```

Summary statistics :

```r
stargazer::stargazer(round(t(sapply(st_drop_geometry(cell_200m[,
    c("prop_Ind_under_18", "pop_dens")]),
    function(x) c(mean(x), sd(x), min(x), max(x)))), 2))
```

Table 4: Summary statistics for the two intensive variables created for pedagogical purposes

Intensive variables	Description	Mean	Sd	Min	Max
prop_Ind_under_18	% of inhabitants under 18 years old	19.9	7.32	0	47.2
pop_dens	Population density (Inhabitants / km^2)	4 985	4 965	24	42 487

We define the labels of the extensive variables on one hand and the label of the intensive variable on other hand:

```r
var_extensive <- c("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", "Ind_0_17", "Ind", "area")
var_intensive <- c("prop_Ind_under_18", "pop_dens")
```

1.2.3 Variables to estimate

Inspired by the work of Nguyen et al. (2018), and thanks to this data set at the cell levels, our aim is to determine the following covariates to explain the extreme right vote :

- population density (ratio of **Ind** to the area),
- percentage of individuals less than 18 (Sum of **Ind__0__3**, **Ind__4__5**, **Ind__6__10**, **Ind__11__17** divided by **Ind**),
- percentage of individuals between 18 and 40 (Sum of **Ind__18__24**, **Ind__25__39** divided by **Ind**),
- percentage of individuals between 40 and 64 (Sum of **Ind__40__54**, **Ind__55__64** divided by **Ind**),
- percentage of individuals above 65 (Sum of **Ind__65__79**, **Ind__80p** divided by **Ind**),
- percentage of poor households (ratio of **Men__pauv** to **Men**),
- percentage of owners (ratio of **Men__prop** to **Men**),
- percentage of recent dwellings (ratio of **Log__ap90** to Sum of **Men__coll**, **Men__mais**),

1.3 2nd kind of sources: data at the iris level

The data can be found here:

- Geographical boundaries: https://public.opendatasoft.com/explore/dataset/contours-iris-2015/export/?flg=fr&sort=iris&refine.nom_com=Toulouse
- data set 1 which corresponds to the population in 2015: https://www.insee.fr/fr/statistiques/3627376#consulter
- data set 2 which corresponds to the residents in 2015 (particularly the unemployed): https://www.insee.fr/fr/statistiques/2386631#consulter

We first import the data:

```r
# import the boundaries
iris <- read_sf("./data/iris/contours-iris-2015.geojson")
iris <- iris %>%
  mutate(IRIS = paste0(insee_com, iris))
# Import data 1
iris_don <- read_excel("./data/iris/base-ic-evol-struct-pop-2015.xls",
                       skip = 5)
# Import data 2
iris_don_2 <- read_excel("./data/iris/base-ic-activite-residents-2013.xls",
                         skip = 5)
# Filter the data
iris_don <- iris_don %>%
  filter(COM == "31555")

iris_don_2 <- iris_don_2 %>%
  filter(COM == "31555") %>%
  select(P13_CHOM1564, P13_ACT1564, IRIS)
# Merge polygons and data
```

```r
iris <- merge(iris, iris_don, by = "IRIS")
iris <- merge(iris, iris_don_2, by = "IRIS")
# Change the CRS
iris <- st_transform(iris, st_crs(bv_sample))
```

The number of sources is equal in that case to $S = 153$. The distribution of the area of the sources is equal to:

```r
summary(st_area(iris))
```

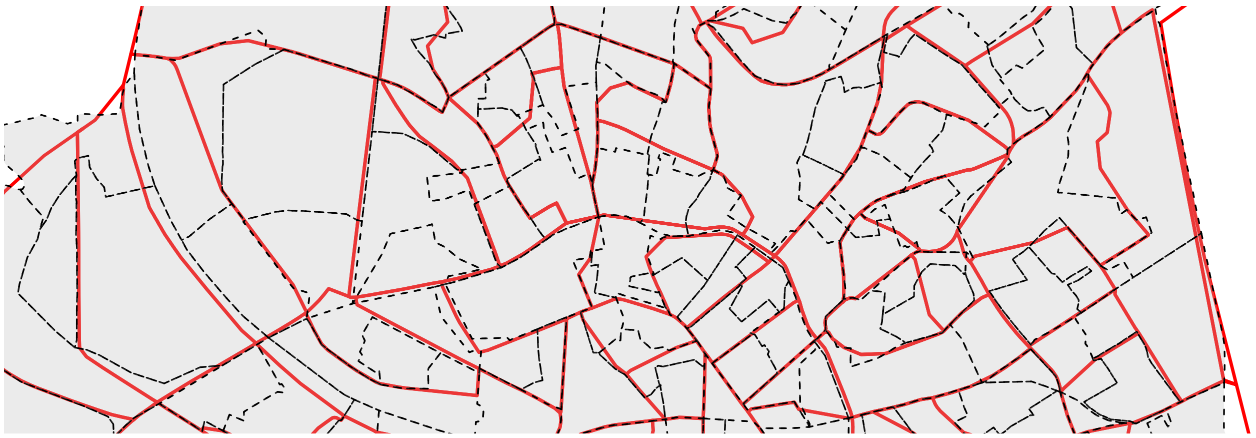
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   79843  237291  344533  770352  725516 8326068
```

Code to obtain Table 1.

```r
area_df <- rbind(
  target = round(as.numeric(summary(st_area(bv_sample))), 0),
  cell_200m = round(as.numeric(summary(st_area(cell_200m))), 0),
  cell_big = round(as.numeric(summary(st_area(cell_big))), 0),
  iris = round(as.numeric(summary(st_area(iris))), 0)
)
stargazer::stargazer(area_df)
```

In the following figure, we zoom in on the maps to represent the few sources in red and the targets in dotted line. It appears that the sources are in general larger than the targets, and in that case, the situation can be considered as a disaggregation problem. In other terms, the variables on the targets should first be split to the intersection zones, and finally to the sources.

```r
plot(st_geometry(iris[40:50, ]), lwd = 2, border = "red")
plot(st_geometry(iris), lwd = 2, border = "red", add = T)
plot(st_geometry(bv_sample),
     add = T, col = scales::alpha("grey", alpha = 0.3), lty = 2)
```



The variables of interest in this data set are presented in Table 6. To produce the Table, we use:

```r
stargazer::stargazer(round(t(sapply(st_drop_geometry(iris[, var_extensive_rp]),
                                    function(x) c(mean(x), sd(x), min(x), max(x)))), 0))
```

These variables are all extensive.

```r
var_extensive_rp <- c("C15_POP15P", "C15_POP15P_CS1", "C15_POP15P_CS2",
                      "C15_POP15P_CS3", "C15_POP15P_CS4", "C15_POP15P_CS5",
                      "C15_POP15P_CS6",
                      "P15_POP", "P15_POP_IMM", "P13_CHOM1564",
                      "P13_ACT1564")
```

Table 5: Descriptive statistics for the variables of interest (target variables) provided by INSEE at the iris levels

Extensive variables	Description	Mean	Sd	Min	Max
POP15P	Inhabitants (up to 15 y.o.)	2 634	1 053	593	6 120
POP15P_CS1	Farmers (up to 15 y.o.)	1	2	0	11
POP15P_CS2	Self-employed workers (up to 15 y.o.)	73	38	10	236
POP15P_CS3	Highly qualified workers (up to 15 y.o.)	459	272	9	1 193
POP15P_CS4	Intermediate (up to 15 y.o.)	431	235	36	1 536
POP15P_CS5	Lower supervisory and technical (up to 15 y.o.)	400	216	69	1 525
POP15P_CS6	Workers/labour (up to 15 y.o.)	222	134	21	902
POP	Population number	3 085	1 211	693	7 812
POP_IMM	Immigrates	453	264	93	1 644
CHOM1564	Unemployed	269	124	40	731
ACT1564	Active people	1 562	677	205	4 152

Thanks to these variables, we aim to estimate for the targets the following variables that we suspect to be related to the extreme right score:

- **prop_unemploy**: corresponding to the ratio of variable **P13_CHOM1564** to **P13_ACT1564**
- **prop_immi**: corresponding to the ratio of variable **P15_POP_IMM** to **P15_POP**
- **prop_csp_1**: corresponding to the ratio of variable (**C15_POP15P_CS1** + **C15_POP15P_CS2** to **P15_POP**)
- **prop_csp_2**: corresponding to the ratio of variable (**C15_POP15P_CS3** + **C15_POP15P_CS4** to **P15_POP**)
- **prop_csp_3**: corresponding to the ratio of variable (**C15_POP15P_CS5** + **C15_POP15P_CS6** to **P15_POP**)

Remark: In this work, we illustrate the dasymetric method control zone on this data set.

1.4 Auxiliary information

INSEE contains individual data concerning housing (see https://www.insee.fr/fr/metadonnees/definition/c1815) that could be used to obtain information at the intersection levels, but it is not open access.

To obtain data at the intersection levels, we can use, for instance, road data (source : https://www.data.gouv.fr/fr/datasets/filaire-de-voirie-toulouse-metropole/) or OSM data. These data are not areal: they are given at the point or the line level, but it is easy to transpose the data at the intersection levels. If data are points, a user can obtain the information at the intersection levels by using the PIP method. For the road data, we simply compute the length of the roads in the intersection zones.

By using these data, we make the assumption that the INSEE variables are correlated with the density of the roads. In other terms, the more roads there are, the more people there are.

```r
# We import the boundaries of the road maps in Toulouse:
voieries <- read_sf("./data/voierie/filaire-de-voirie.geojson")
voieries <- st_transform(voieries, crs = st_crs(bv_sample))
```

Finally the variables we want to estimate are:

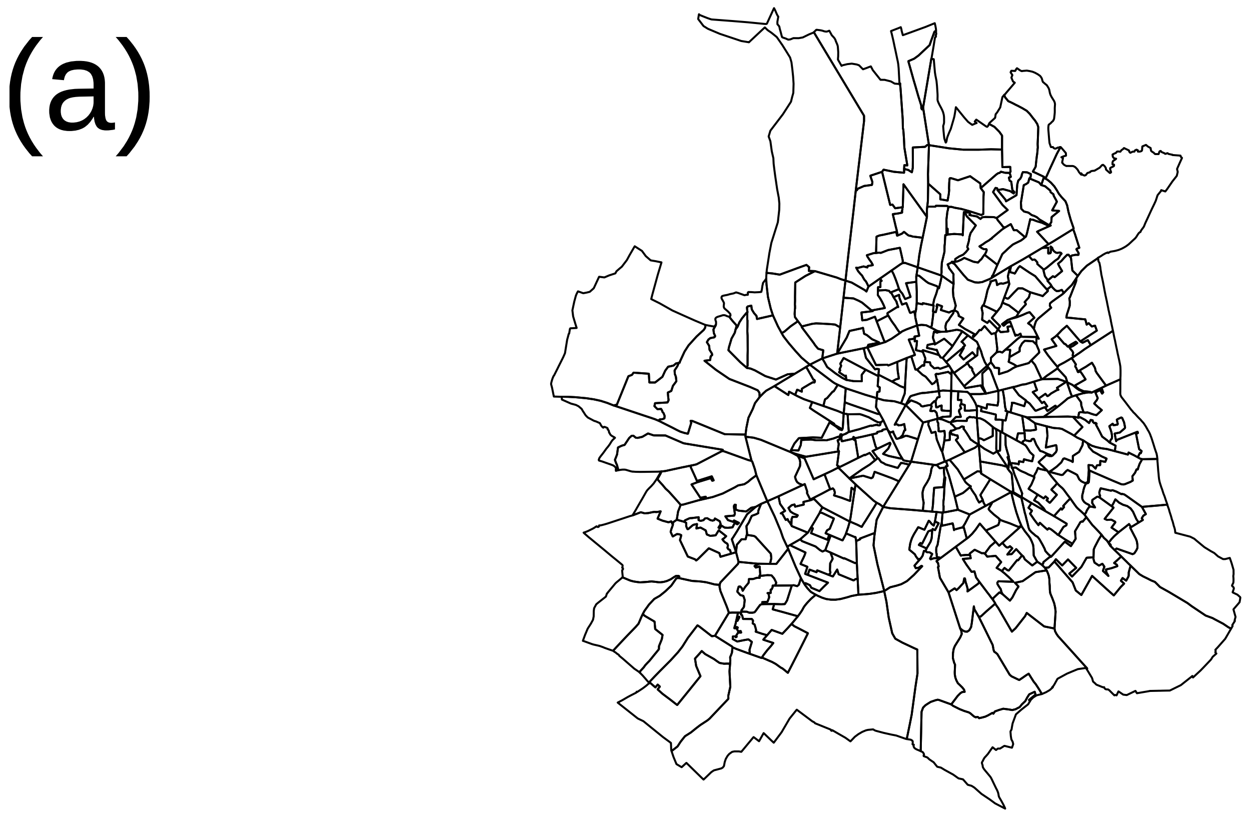
Table 6: Variables to estimate

Variable	Source	Description
pop_dens	small cells	Population density
prop_pour_hos	small cells	% of poor households
prop_owner	small cells	% of owners
prop_recent	small cells	% of recent dwellings
prop_Ind_18_40	small cells	% of ind. between 18 and 40
prop_Ind_40_64	small cells	% of ind. between 40 and 64
prop_Ind_up_65	small cells	% of ind. up to 65
prop_unemploy	iris	% of unemployment
prop_immi	iris	% of immigrates
prop_csp_1	iris	% of farmers and self-employed
prop_csp_2	iris	% of intermediate and High. qualif.
prop_csp_3	iris	% Lower supervisory and workers

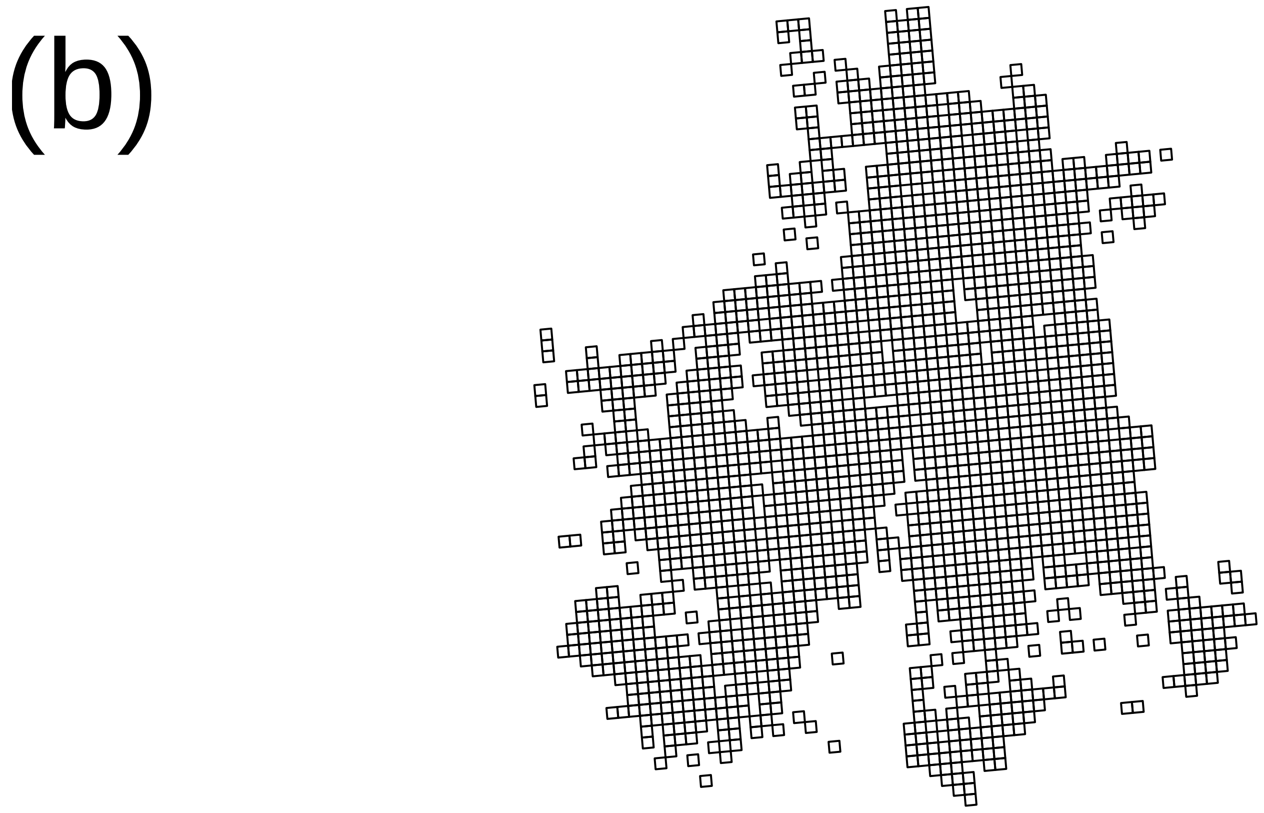
1.5 Figure 1 in the book chapter

To get Figure 1:

```r
temp <- st_intersection(
  cell_200m[,  "IdINSPIRE"], voieries[, "id_troncon"])

par(oma = c(0, 0, 0, 0), mar = c(0, 0, 0, 0))
plot(st_geometry(bv_sample))
legend("topleft", legend = "", title = "(a)", cex = 4, bty = "n")
```

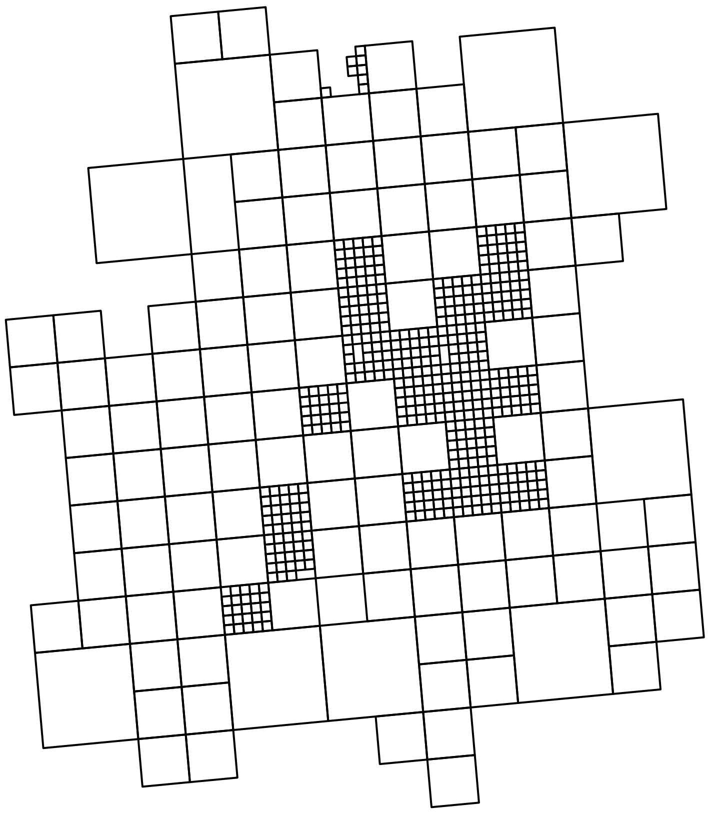


```r
plot(st_geometry(cell_200m))
legend("topleft", legend = "", title = "(b)", cex = 4, bty = "n")
```



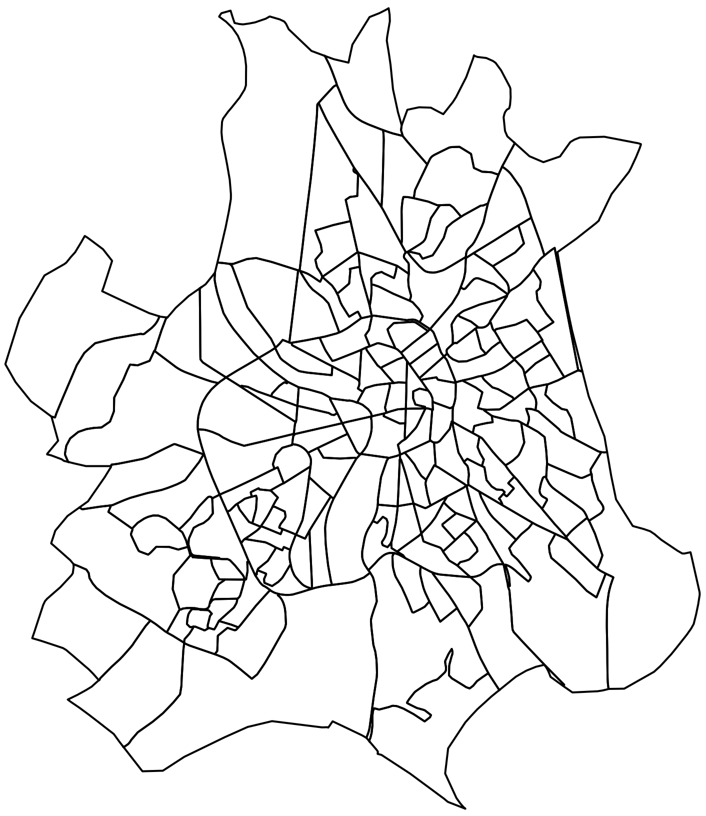
```r
plot(st_geometry(cell_big))
legend("topleft", legend = "", title = "(c)", cex = 4, bty = "n")
```

(c)

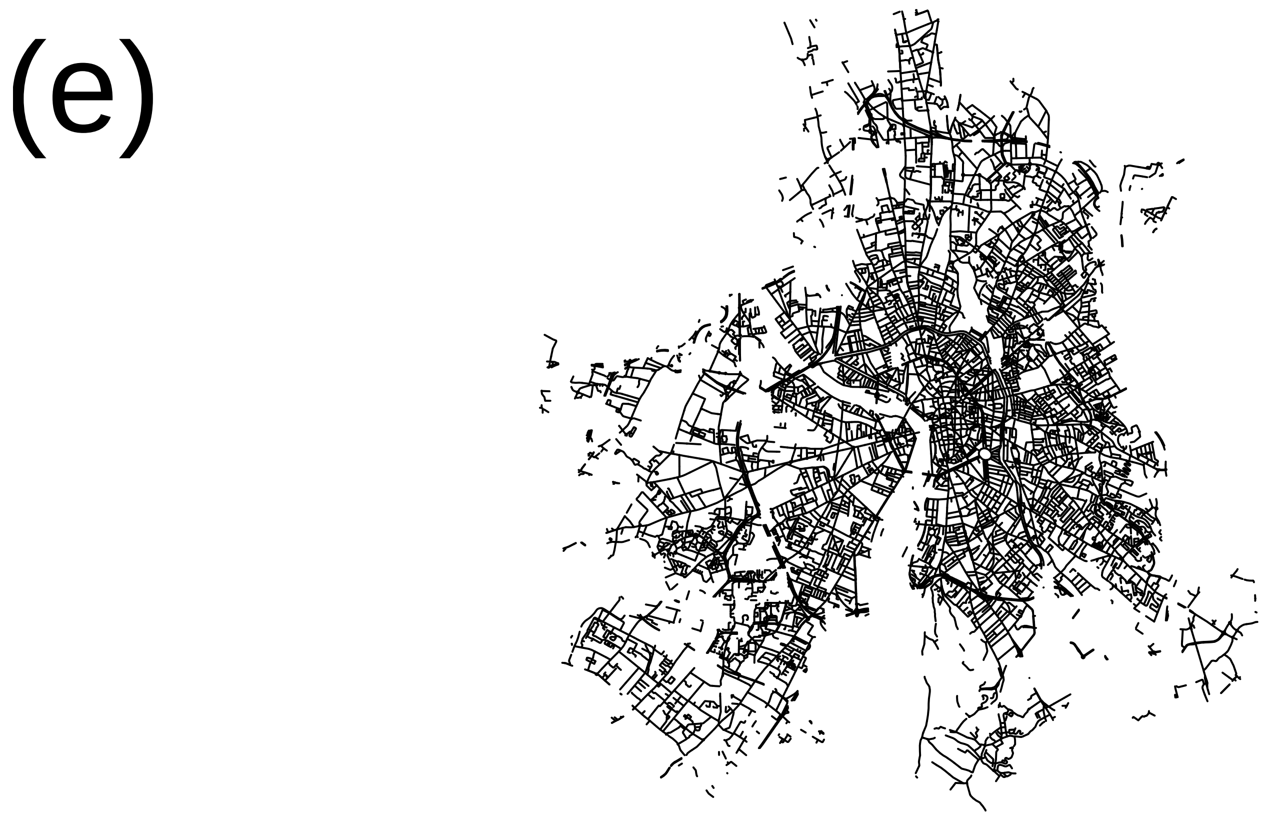


```r
plot(st_geometry(iris))
legend("topleft", legend = "", title = "(d)", cex = 4, bty = "n")
```

(d)



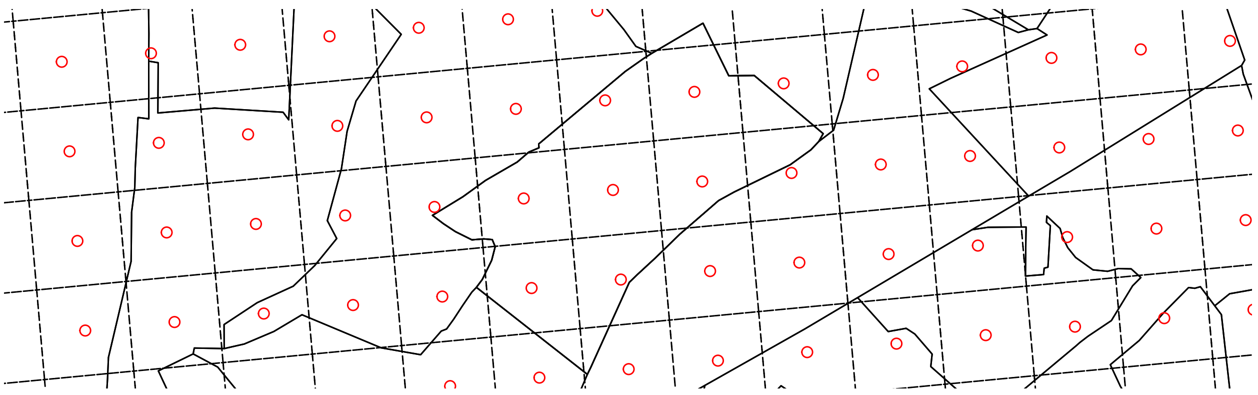
```r
plot(st_geometry(temp))
legend("topleft", legend = "", title = "(e)", cex = 4, bty = "n")
```



2 Point-in-polygon method

2.1 Illustration

In the following figure, when we consider the small cells as the sources, we observe that the targets (in full lines) are larger than the sources (the cells in dashed lines). The method point-in-polygon is usually appropriated for those kinds of geometries.



For this method, we consider the cell as a point by choosing the centroid of the cell to use the point-in-polygon method (this is actually what is done by the INSEE, which has provided an **R** program for estimating the variables at different target levels). The function *st_centroid()* permits us to transform the sources from cell to points, and the function *st_intersects()* permits us to determine if a source intersects with a target or not.

2.2 Extensive variables

In the case of extensive variables, it is obvious that the aggregation consists simply in summing the values of the points falling within the same target. For the case of the regular cells, it consists in executing this code:

```
# convert the cell to points
cell_200m_as_points <- st_centroid(cell_200m)

# intersection of points and polygons
temp_inters_cell_200m <- st_intersects(bv_sample,
                              cell_200m_as_points, sparse = F)

# detect the null intersections
intersect_yes <- apply(temp_inters_cell_200m, 1, function (x) any(x != F))

# aggregate the data
for (i in which(intersect_yes)) {
  bv_sample[i, paste0(var_extensive, "_pip_cells")] <-
    apply(st_drop_geometry(cell_200m_as_points)
              [temp_inters_cell_200m[i,], var_extensive], 2, sum)
}

# NA value for the targets with no values
bv_sample[!intersect_yes, paste0(var_extensive, "_pip_cells")] <- NA
```

We have performed the same estimates for the large cells, but we do not print the code because it is identical to the previous.

2.2.1 Border effects

It could be the case that some points do not fall into any sources due to border effects, as presented in the next figure. This is something which may occur when the targets are not nested into the sources, which is the case in our data, especially for the large cells.

```
plot(st_geometry(cell_big_as_points[1:5, ]))
plot(st_geometry(cell_big[1:5, ]), add = T)
plot(st_geometry(bv_sample), add = T)
```

In that case, there are two options for the point-in-polygon method:

- **exclude** : the user supposes that these sources should not be included in the targets because they are outside the space of targets. In that case, the sum of the extensive variables obtained for the targets should be lower than the sum of the extensive variables obtained for the sources.
- **include** : the user supposes that every source should be associated with a target. It is possible to use the algorithm of the nearest neighbour to attribute a source to its closest target.

In that second case, we use the following two steps:

- Identify the sources which do not fall into targets:

```
temp_inters_mat <- st_intersects(cell_big_as_points, bv_sample, sparse = T)
ind_with_no_inter <- which(sapply(temp_inters_mat, length) == 0)
```

- use function *st_nearest_feature()* from **sf** to detect the closest neighbour and add the values to the concerned targets:

```
bv_sample[, paste0(var_extensive, "_pip_big_2")] <- bv_sample[,
                                          paste0(var_extensive, "_pip_big")]
for (k in ind_with_no_inter) {
  ind_k <- st_nearest_feature(cell_big_as_points[k, ], bv_sample)
  bv_sample[ind_k, paste0(var_extensive, "_pip_big_2")] <-
    rowSums(cbind(
      as.numeric(st_drop_geometry(bv_sample[ind_k, paste0(var_extensive, "_pip_big_2")])),
      as.numeric(st_drop_geometry(cell_big_as_points)[k, var_extensive])
      ), na.rm = T)
}
```

We have done the same thing for the small cells, but we do not present the codes because they are similar to the previous codes.

We compare the total number of inhabitants (variable **Ind**) in the sources and the targets with respect to the method proposed above. In the case of **exclude**, the estimated number of inhabitants is lower than the total number of inhabitants observed at the source level, whereas in the case of **include**, all inhabitants have affected the targets.

For the small cells, the total number of individuals observed on the sources is equal to:

```
sum(cell_200m$Ind)
```

```
## [1] 404497
```

The total number of individuals observed on the targets is equal to the total on sources when including and slightly lower when excluding:

```
sum(bv_sample$Ind_pip_cells, na.rm = T)
```

```
## [1] 403729
```

```
sum(bv_sample$Ind_pip_cells_2, na.rm = T)
```

```
## [1] 404497
```

For the large cells, the total number of individuals observed on the sources is equal to:

```
sum(cell_big$Ind)
```

```
## [1] 457031.5
```

Because the cells are larger, the total number of individuals observed on the targets is much lower when we exclude the sources outside the delimited area of the sources.

```r
sum(bv_sample$Ind_pip_big, na.rm = T)
```

```
## [1] 409717
```

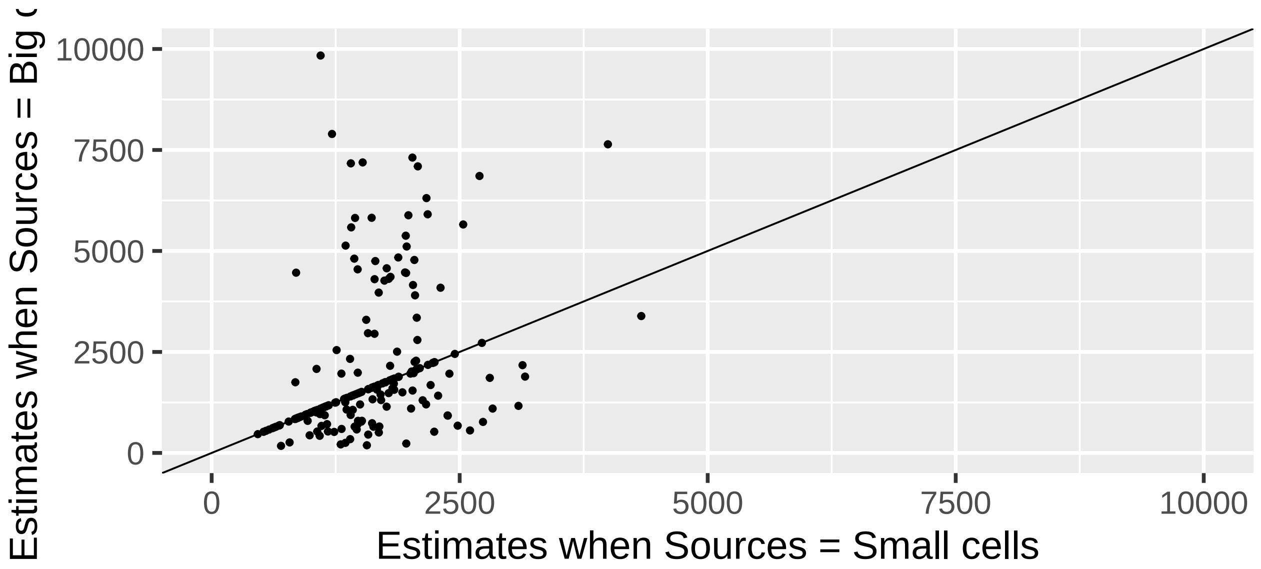
```r
sum(bv_sample$Ind_pip_big_2, na.rm = T)
```

```
## [1] 457031.5
```

2.2.2 Comparaison between small and big cells taken as sources

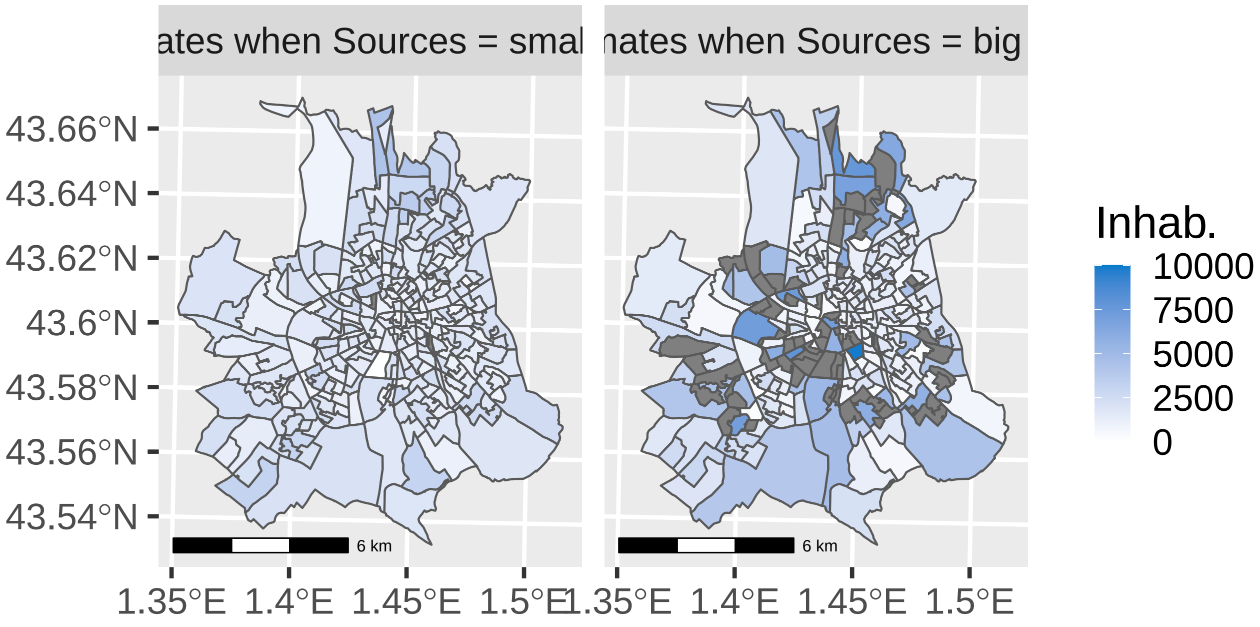
We compare the results obtained when considering the small cells or the large cells as sources. We have kept the method which has excluded the points falling outside the zone delimited by the sources. We focus on the variable number of inhabitants to obtain Figure 2 in the article.

```r
ggplot(bv_sample) +
  aes(x = Ind_pip_cells, y = Ind_pip_big) +
  geom_point() +
  xlim(0, 10000) +
  ylim(0, 10000) +
  geom_abline(intercept = 0, slope = 1) +
  labs(x = "Estimates when Sources = Small cells", y = "Estimates when Sources = Big cells") +
   theme_grey(base_size = 22)
```



We also represent in the following maps the estimates when the sources are the small (resp. large) cells.

```r
library(tidyr)
nc2 <- bv_sample %>% select(Ind_pip_cells, Ind_pip_big, geometry) %>%
  rename(Ind_pip_1 = Ind_pip_cells,
         Ind_pip_2 = Ind_pip_big) %>%
  gather(VAR, Ind_pip_, -geometry)
ggplot() +
  geom_sf(data = nc2, aes(fill = Ind_pip_)) +
  scale_fill_gradient('Inhab.', low='#ffffff', high = '#007acc', limits = c(0, 10000)) +
  facet_wrap(~VAR, ncol = 2, labeller = labeller(VAR =
    c("Ind_pip_1" = "Estimates when Sources = small cells",
      "Ind_pip_2" = "Estimates when Sources = big cells")
  )) +
    annotation_scale(location = "bl", width_hint = 0.5) +
   theme_grey(base_size = 22)
```



2.3 Intensive variables

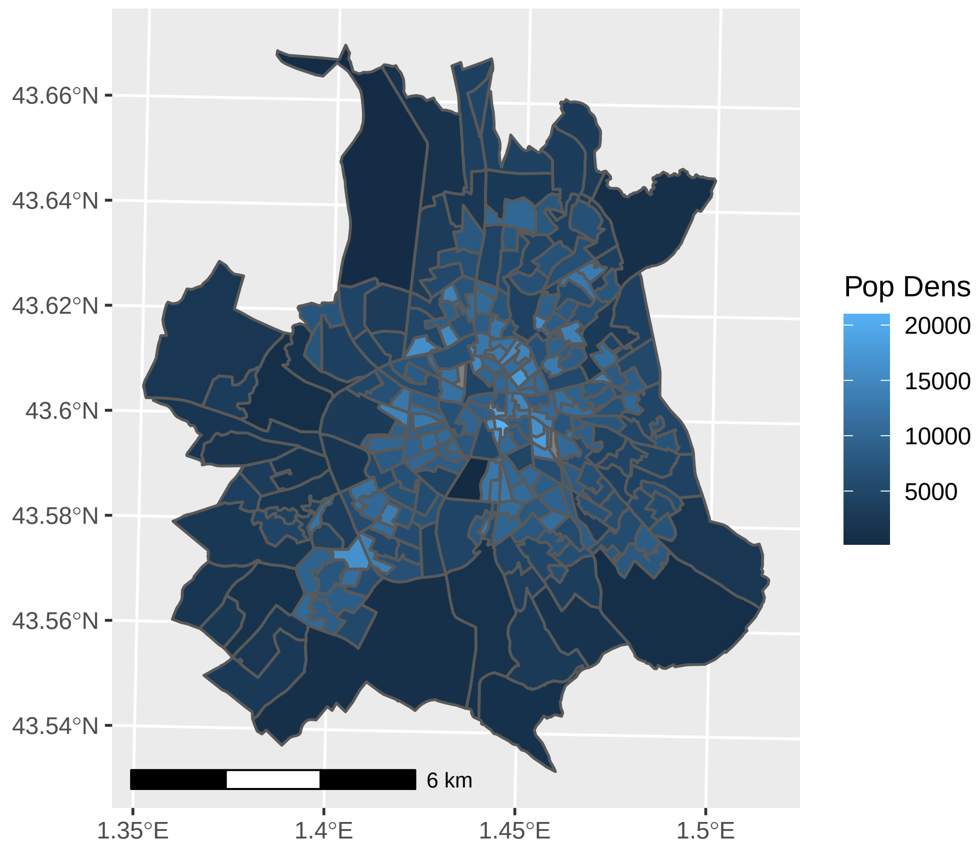
2.3.1 The extensive variables defining the intensive variable are known

Once the aggregation has been conducted on the extensive variables, if an intensive variable is defined as the ratio of two extensive variables, it is then possible to produce the ratio of the two estimates. For example, we estimate the PIP method for the two variables number of inhabitants lower than 18 years old (**prop_Ind_under_18**) and population density (**pop_dens**):

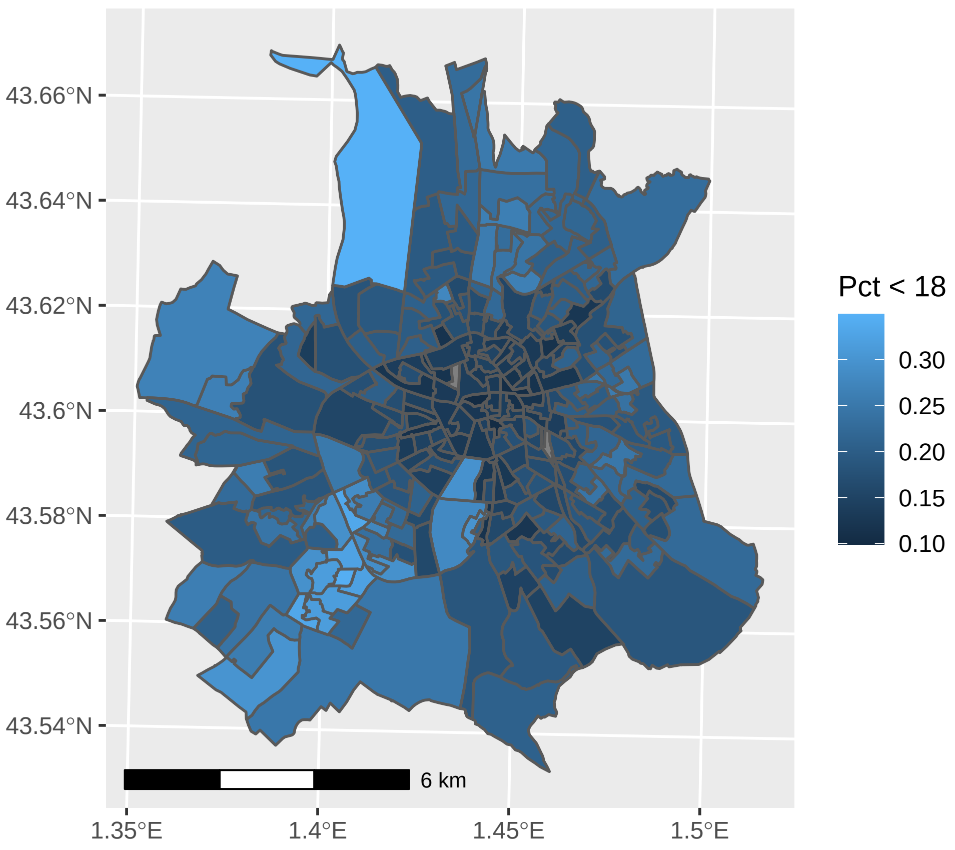
```
bv_sample <- bv_sample %>%
  mutate(
    prop_Ind_under_18_pip_cells  = Ind_0_17_pip_cells / Ind_pip_cells ,
    pop_dens_pip_cells   = Ind_pip_cells / area_pip_cells,
    prop_Ind_under_18_pip_big   = Ind_0_17_pip_big / Ind_pip_big,
    pop_dens_pip_big   = Ind_pip_big / area_pip_big
  )
```

We represent the estimates of the two intensive variables computed when the sources correspond to the small cells.

```
ggplot(data = bv_sample) +
  geom_sf(aes(fill = pop_dens_pip_cells)) +
           labs(fill = "Pop Dens") +
    annotation_scale(location = "bl", width_hint = 0.5)
```



```
ggplot(data = bv_sample) +
  geom_sf(aes(fill = prop_Ind_under_18_pip_cells)) +
    annotation_scale(location = "bl", width_hint = 0.5)  +
           labs(fill = 'Pct < 18')
```

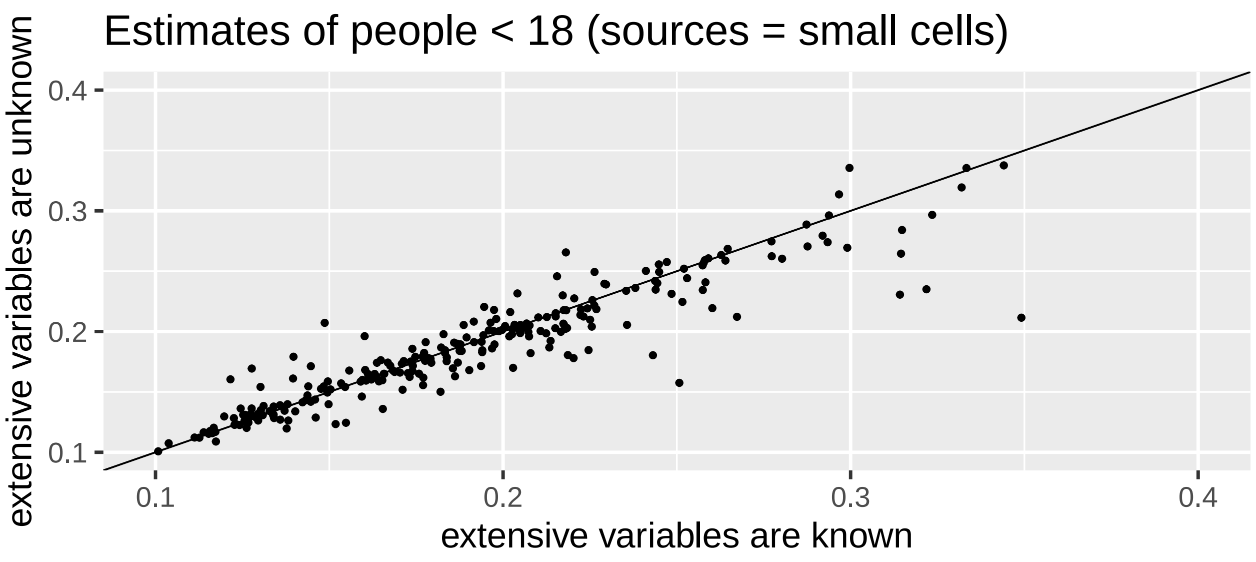


2.3.2 The extensive variables defining the intensive variable are unknown

In the following codes, we estimate the intensive variables without considering the weights and we compare them to those obtained previously. Hence, we remark that with no weight, the estimates provide different results.

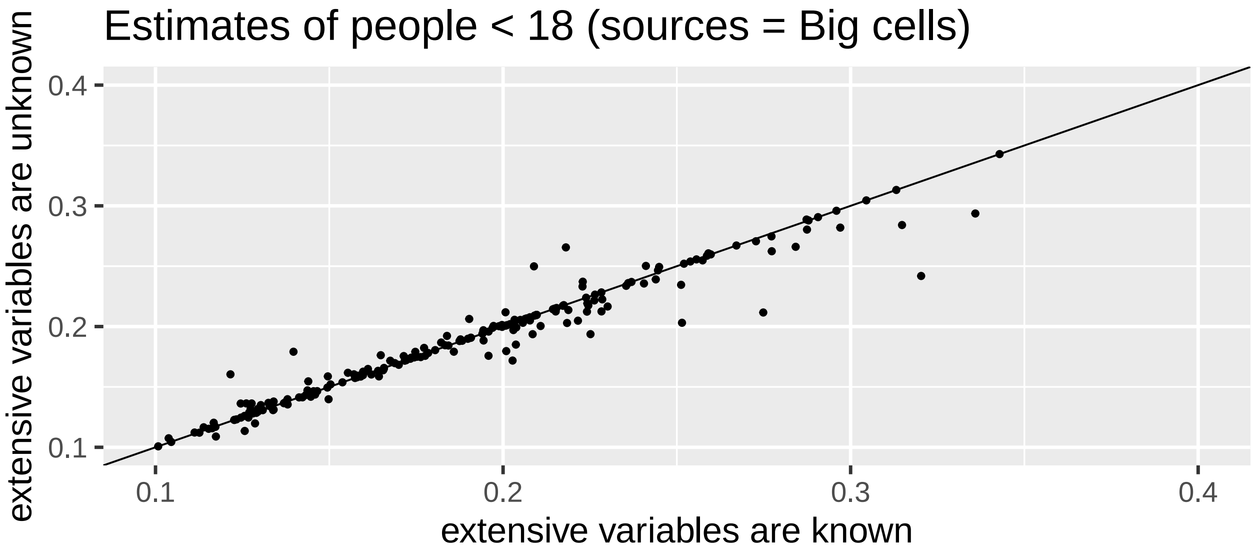
In the case of the small cells taken as sources, even if the results are correlated, the estimates are slightly different. Figure 3 in the article is obtained.

```r
# aggregate the data
for (i in 1:nrow(bv_sample)) {
  bv_sample[i, paste0(var_intensive, "_pip_cells_bad")] <-
    apply(st_drop_geometry(cell_200m)
                          [temp_inters_cell_200m[i,], var_intensive], 2, mean)
}
ggplot(bv_sample) +
  aes(x = prop_Ind_under_18_pip_cells, y = prop_Ind_under_18_pip_cells_bad) +
  geom_point() +
  labs(title = "Estimates of people < 18 (sources = small cells)",
       x = "extensive variables are known",
       y = "extensive variables are unknown") +
  xlim(0.1, 0.4) +
  ylim(0.1, 0.4) +
  geom_abline(intercept = 0, slope = 1) +
   theme_grey(base_size = 20)
```



For the large cells, it seems that there are fewer differences. This could be due to the fact that the number of sources falling inside the targets is lower than for the small cells. Indeed, let us suppose that exactly one source falls inside one target. In that case, the two methods would be equivalent.

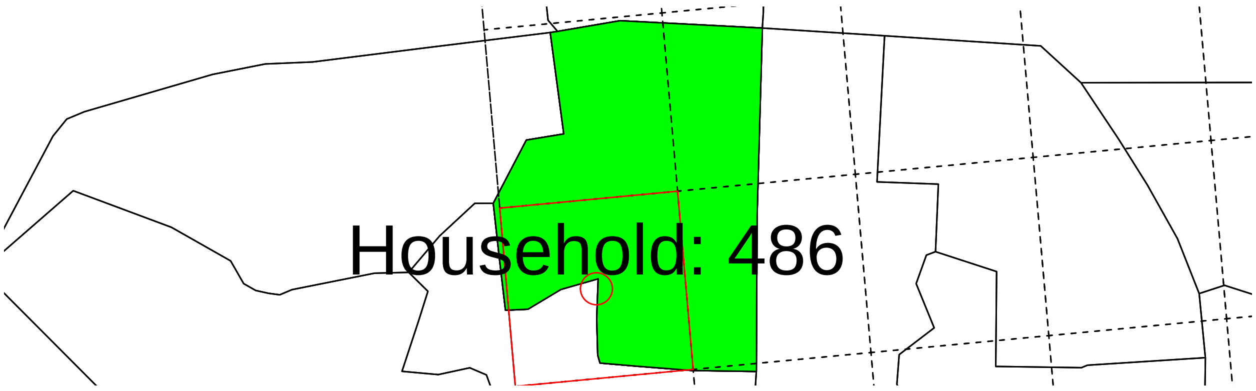
```
# aggregate the data
for (i in 1:nrow(bv_sample)) {
  bv_sample[i, paste0(var_intensive, "_pip_big_bad")] <-
    apply(st_drop_geometry(cell_big)
                          [temp_inters_big[i, ], var_intensive], 2, mean)
}
ggplot(bv_sample) +
  aes(x = prop_Ind_under_18_pip_big, y = prop_Ind_under_18_pip_big_bad) +
  geom_point() +
  labs(title = "Estimates of people < 18 (sources = Big cells)",
       x = "extensive variables are known",
       y = "extensive variables are unknown") +
    xlim(0.1, 0.4) +
  ylim(0.1, 0.4) +
  geom_abline(intercept = 0, slope = 1) +
   theme_grey(base_size = 20)
```



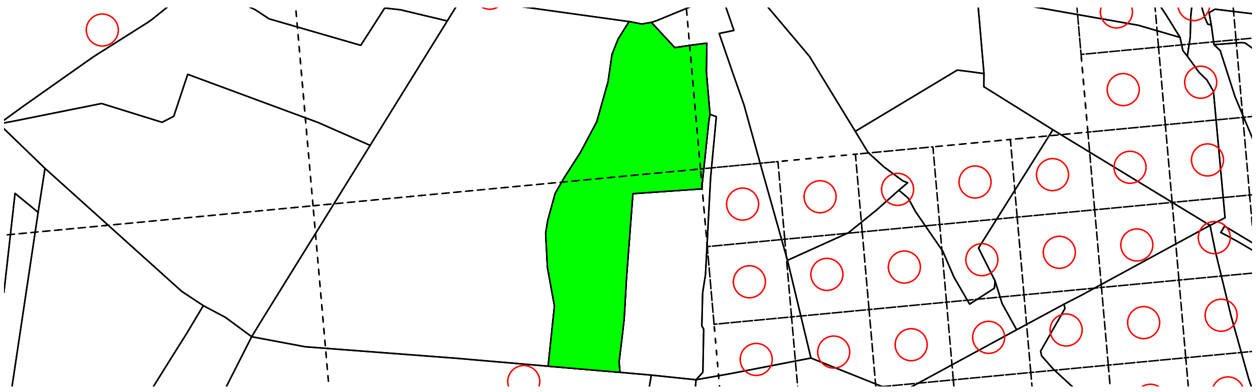
2.4 Limitation of the point-in polygon method

The limitations of this method are illustrated in Figure 4 in the book chapter.

- In the first figure, approximately 75% of a source (represented in red) is falling within the target in green. However, as the centroid is falling into the target neighbour, the number of households (486 hereafter) is given for the target neighbour. In this example, the sizes of the targets and the sources are close. In that case, it is preferable to use another method, like DAW or DAX.



- In the second figure, we can see that the estimates in the target in green equal 0, because there are no red points falling inside the green polygon. One more time, this is due to the fact that the large cells ($1km \times 1km$ and $2km \times 2km$) are larger than the targets.



In our application, 57 (resp. 2) targets have not been estimated when using the large (resp. small) cells as sources.

```r
length(which(is.na(bv_sample[, "pop_dens_pip_big"])))
```

```
## [1] 57
```

3 Areal weighting interpolation (DAW) method

3.1 Extensive variable

3.1.1 Border effect issue

As for the point-in-polygon method, the issue of border effect still occurs. For example, if a source intersects 30% with target 1, 40% with target 2, and 30% with an empty zone, how should we disaggregate the variable $x = 100$ inhabitants? There are two possibilities :

- **exclude** : 30 inhabitants are affected for target 1, 40 inhabitants are affected for target 2, and 30 inhabitants are not affected. In that case, the sum of x on the targets will be lower than the sum of x on the sources.
- **include** : $\frac{30}{30+40} \times 100 = 42.86$ inhabitants are affected for target 1 and $\frac{40}{30+40} \times 100 = 57.14$ inhabitants are affected for target 2.

In our illustration, we compare the two options. We present here the codes in the case of the small cells taken as sources:

```r
# intersection of sources and targets
temp_inters_cell_200m <- st_intersection(bv_sample[, "BUREAU"],
                  cell_200m[, c(var_extensive, "IdINSPIRE")])

# compute the areas of the intersections divided by the area of the source
temp_inters_cell_200m$Area_intersect <-
  as.numeric(st_area(temp_inters_cell_200m)) / 1000 ^ 2

# compute the sum of the intersected zones per source
sum_intersect_cell_200m <- temp_inters_cell_200m %>%
  select(Area_intersect, IdINSPIRE) %>%
  group_by(IdINSPIRE) %>%
  summarise(sum_area = sum(Area_intersect))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
# merge with the interesected zones
temp_inters_cell_200m <-  merge(temp_inters_cell_200m,
      st_drop_geometry(sum_intersect_cell_200m), by = "IdINSPIRE")
temp_inters_cell_200m$Area_share_1 <- temp_inters_cell_200m$Area_intersect /
  temp_inters_cell_200m$area
temp_inters_cell_200m$Area_share_2 <- temp_inters_cell_200m$Area_intersect /
  temp_inters_cell_200m$sum_area

# Multiply the variables by the proportions
# * for method 1
temp_inters_cell_200m[, paste0(var_extensive, "_1")] <-
  sapply(st_drop_geometry(temp_inters_cell_200m[, var_extensive]),
         function(x) x * temp_inters_cell_200m$Area_share_1)

# Aggregate the variables by target
temp_targets <- aggregate(temp_inters_cell_200m[, paste0(var_extensive, "_1")],
                          by = list(temp_inters_cell_200m$BUREAU),
                          FUN = sum)
# Rename the variables
temp_targets <- temp_targets %>%
  rename_at(vars(paste0(var_extensive, "_1")), ~
              paste0(var_extensive, "_daw_cells"))
# Merge with the targets
bv_sample <- merge(bv_sample, st_drop_geometry(temp_targets),
                   by.x = "BUREAU", by.y = "Group.1")
# * for method 2
temp_inters_cell_200m[, paste0(var_extensive, "_2")] <- sapply(
  st_drop_geometry(temp_inters_cell_200m[, var_extensive]),
  function(x) x * temp_inters_cell_200m$Area_share_2)
# Aggregate the variables by target
temp_targets <- aggregate(temp_inters_cell_200m[, paste0(var_extensive, "_2")],
                          by = list(temp_inters_cell_200m$BUREAU),
                          FUN = sum)
# Rename the variables
temp_targets <- temp_targets %>%
  rename_at(vars(paste0(var_extensive, "_2")), ~
              paste0(var_extensive, "_daw_cells_2"))
# Merge with the targets
bv_sample <- merge(bv_sample, st_drop_geometry(temp_targets),
                   by.x = "BUREAU", by.y = "Group.1")
```

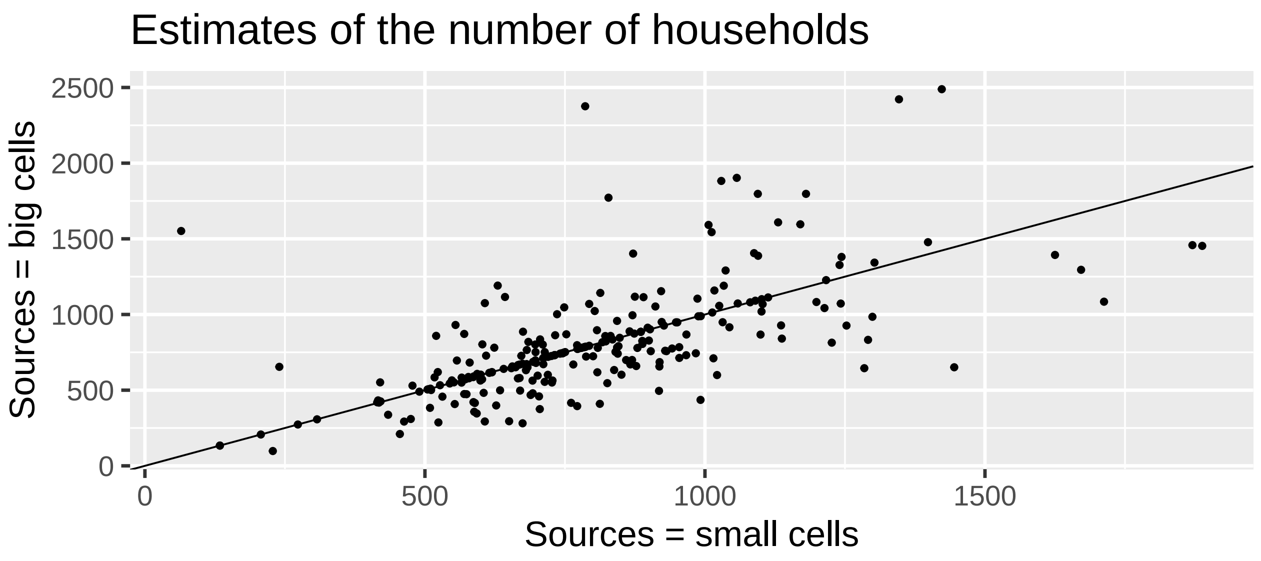
We have done the same thing for the large cells but we do not present the codes here as it is the same than the one seen previously.

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

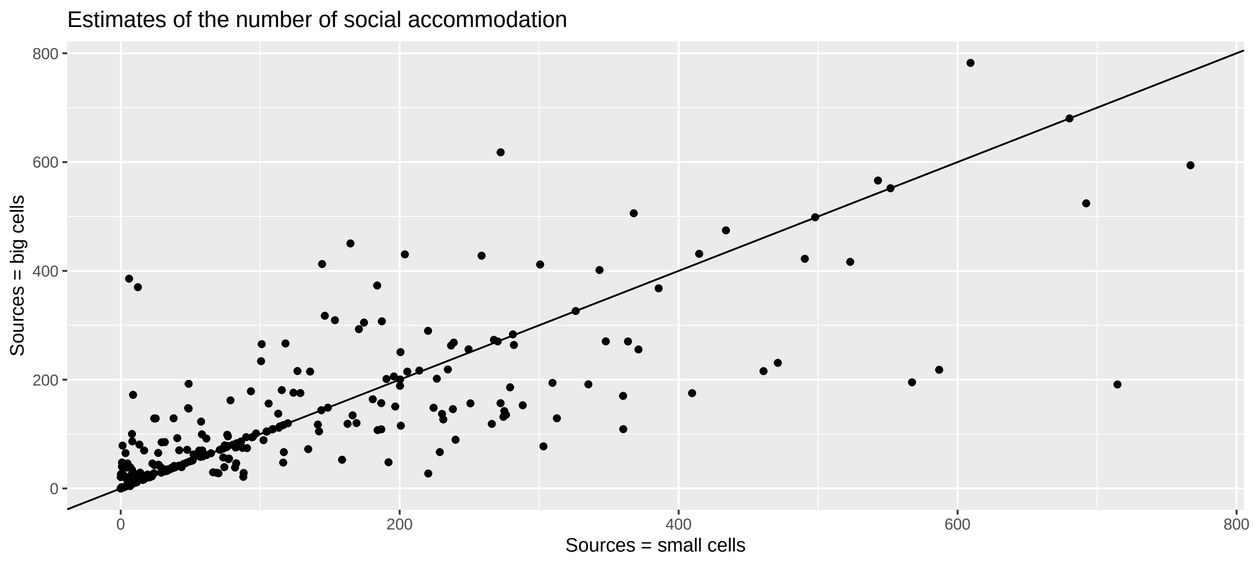
3.1.2 Comparaison between the two sources of data

Now, we compare the DAW results with respect to the choice of the sources (small cells or large cells). Few observations fit with the regression line $y = x$, but we observe differences between the two estimates. As for the PIP method, we recommend to use the sources with the most detailed geographical level if possible.

```r
ggplot(bv_sample) +
  aes(x = Men_daw_cells, y = Men_daw_big) +
  geom_point() +
  labs(title = "Estimates of the number of households",
       x = "Sources = small cells",
       y = "Sources = big cells") +
  geom_abline(intercept = 0, slope = 1) +
   theme_grey(base_size = 20)
```



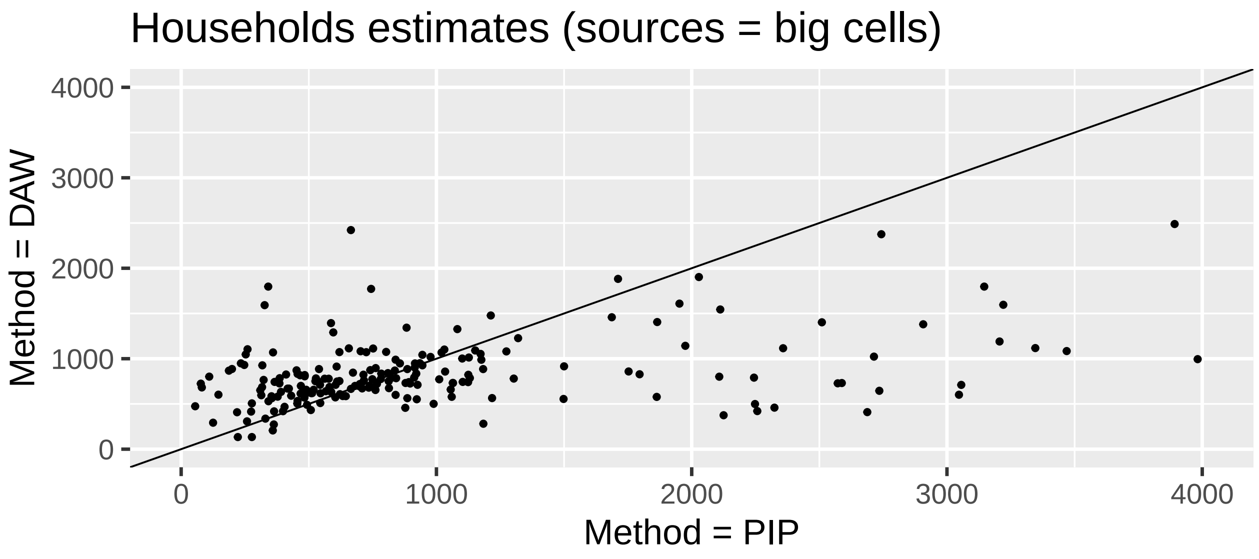
```
ggplot(bv_sample) +
  aes(x = Log_soc_daw_cells, y = Log_soc_daw_big) +
  geom_point()  +
  labs(title = "Estimates of the number of social accommodation",
       x = "Sources = small cells",
       y = "Sources = big cells") +
  geom_abline(intercept = 0, slope = 1)
```



3.2 Comparaison between PIP and DAW

We compare the results obtained with point-in-polygon and with DAW for the variable number of households. First, we present the figure in the case where the large cells have been utilized as sources to produce Figure 5 in the book chapter.

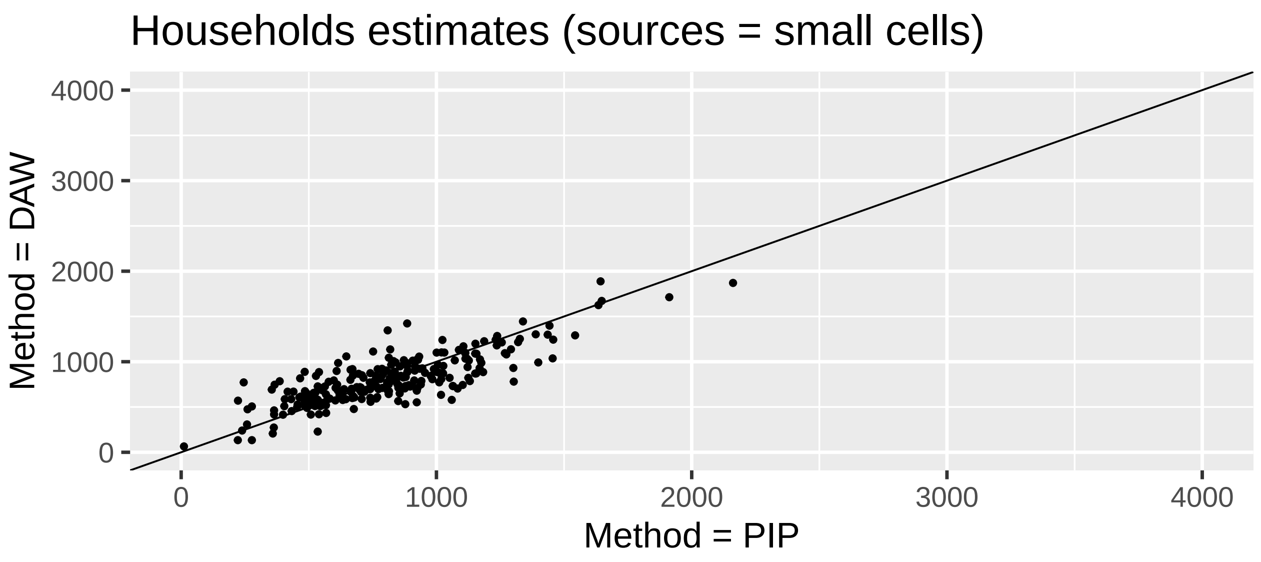
```r
ggplot(bv_sample) +
  aes(x = Men_pip_big, y = Men_daw_big) +
  labs(title = "Households estimates (sources = big cells)",
       x = "Method = PIP",
       y = "Method = DAW") +
  xlim(0, 4000) +
  ylim(0, 4000) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1) +
   theme_grey(base_size = 20)
```



Hence, it appears that there are extensive differences. It seems that the DAW method corrects the approximation made by the PIP method (1 source is affected by the 1 and only target).

We now look at the differences in the case of the small cells taken as sources:

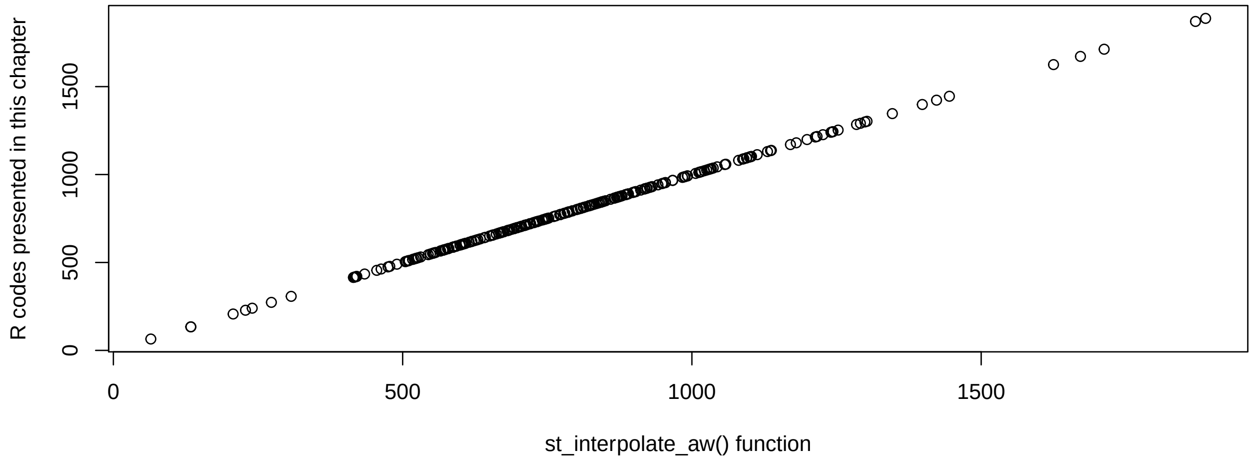
```r
ggplot(bv_sample) +
  aes(x = Men_pip_cells, y = Men_daw_cells) +
  labs(title = "Households estimates (sources = small cells)",
       x = "Method = PIP",
       y = "Method = DAW") +
    xlim(0, 4000) +
  ylim(0, 4000) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1) +
   theme_grey(base_size = 20)
```



In the case of small cells, the differences are less obvious because the approximation made by the PIP method (1 source is affected by the 1 and only target) has less impact when the sources are smaller than the targets, because 1 source has more chance to fall fully inside 1 target. In other terms, the smaller the sources are as compared to the targets, the more similar the PIP and DAW methods should be.

Remark: The DAW method has been implemented by Pebsema (2018) in function *st_interpolate_aw()*. It can be used like this. The scatter plot shows that the program coincides with what we have done in the case where we exclude the non-intersected zones. However, *st_interpolate_aw()* does not permit to keeping the identification of the sources and targets which define the intersected zones.

```
a1 <- st_interpolate_aw(cell_200m["Men"], bv_sample, extensive = TRUE)
plot(a1$Men, bv_sample$Men_daw_cells,
     xlab = "st_interpolate_aw() function",
     ylab = "R codes presented in this chapter")
```



3.3 Intensive variable

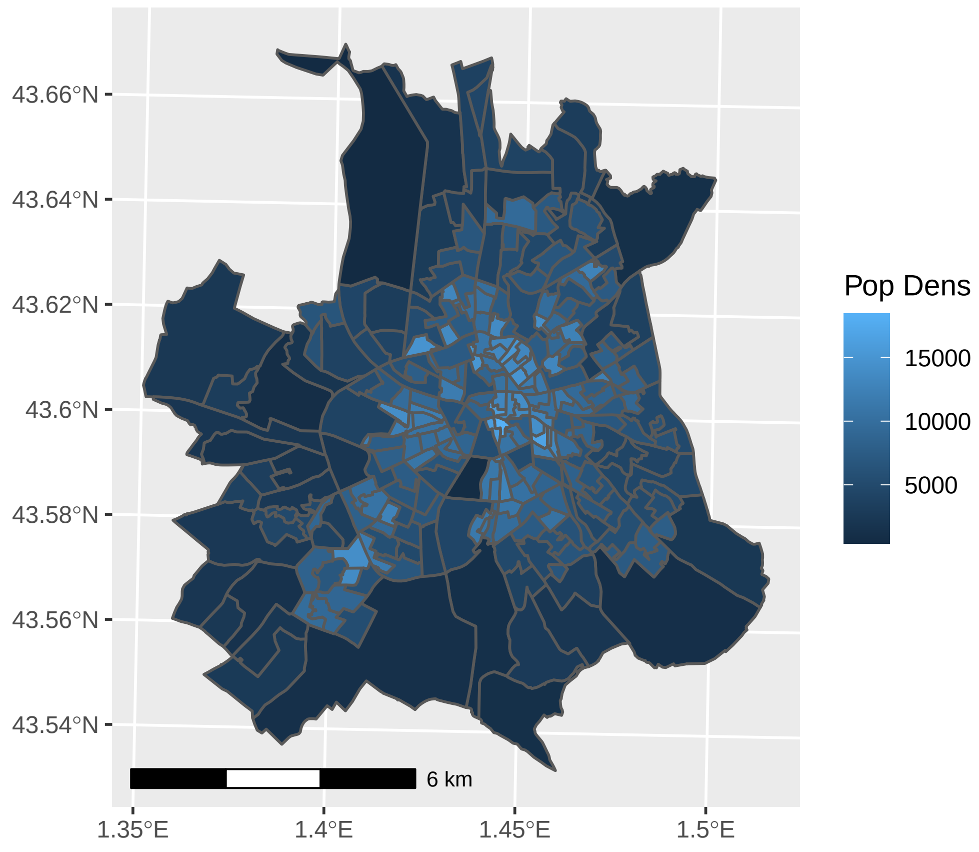
3.3.1 Extensive variables are known

If the extensive variables which define the intensive variables are known, the DAW method is applied first on the extensive variable to obtain the estimates at the target levels, and then intensive variables are re-created on the estimates. For example, we estimate the DAW method for the two variables number of inhabitants under 18 years old (**prop_Ind_under_18**) and population density (**pop_dens**):

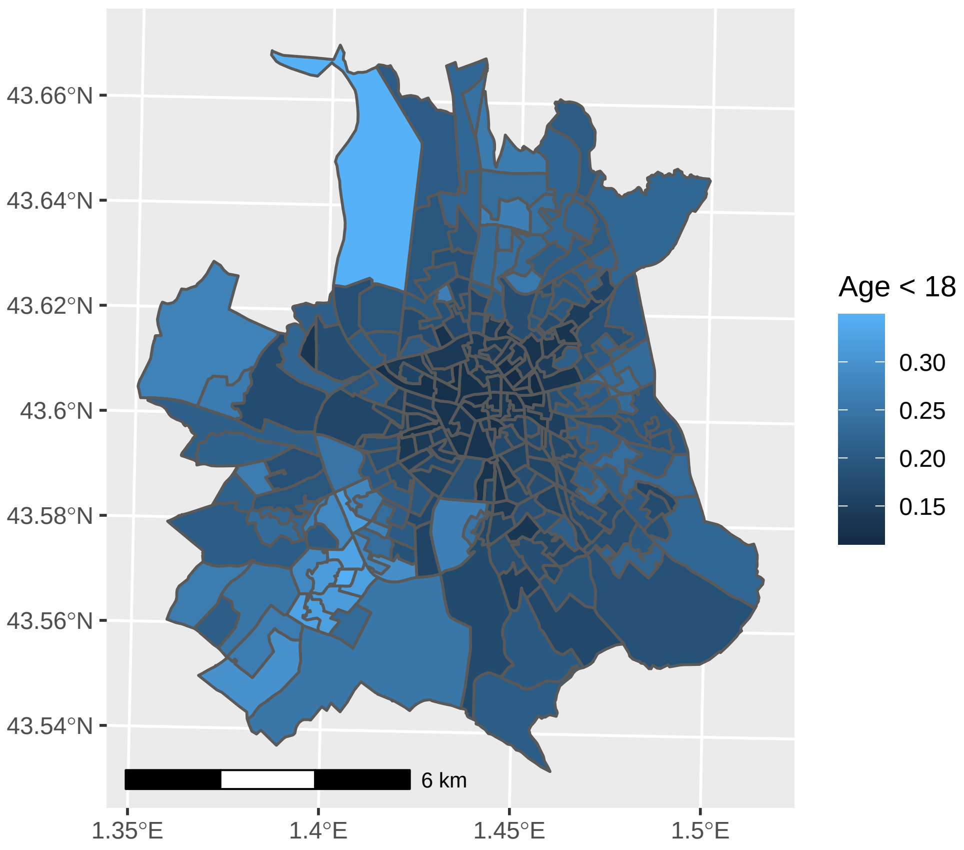
```r
bv_sample <- bv_sample %>%
  mutate(
    prop_Ind_under_18_daw_cells = Ind_0_17_daw_cells / Ind_daw_cells,
    pop_dens_daw_cells = Ind_daw_cells / area_daw_cells,
    prop_Ind_under_18_daw_big = Ind_0_17_daw_big / Ind_daw_big,
    pop_dens_daw_big = Ind_daw_big / area_daw_big
  )
```

We represent the estimates of the two variables by using the small cells as sources.

```r
ggplot(data = bv_sample) +
  geom_sf(aes(fill = pop_dens_daw_cells)) +
           labs(fill = "Pop Dens") +
    annotation_scale(location = "bl", width_hint = 0.5)
```



```r
ggplot(data = bv_sample) +
  geom_sf(aes(fill = prop_Ind_under_18_daw_cells)) +
    annotation_scale(location = "bl", width_hint = 0.5)  +
           labs(fill = 'Age < 18')
```



3.3.2 Extensive variables are unknown

If it is not the case and the extensive variables are not known, we suppose that $\hat{Y}_{s,t} = Y_s$. Then, $\hat{Y}_t = \sum \frac{|A_{s,t}|}{|T_t|} Y_s$.

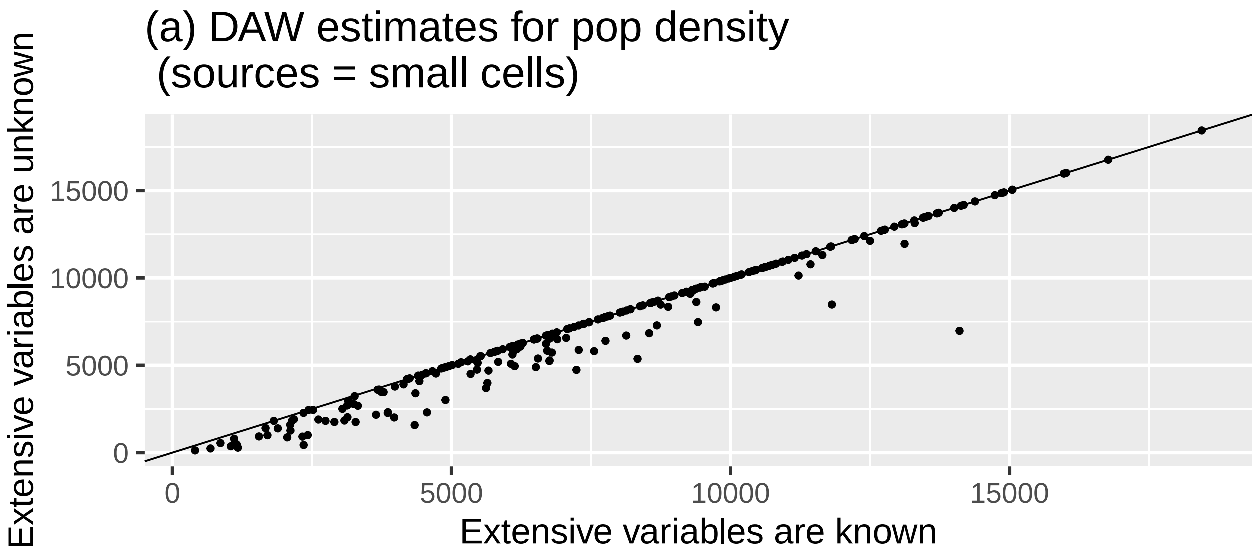
```
# intersection of sources and targets
temp_inters_cells <- st_intersection(bv_sample[, "BUREAU"],
                     cell_200m[, c(var_intensive, "area", "IdINSPIRE")])

# compute the areas of the intersections divided by the area of the source
temp_inters_cells$Area_intersect <- as.numeric(st_area(temp_inters_cells)) /
  1000 ^ 2

# compute the sum of the intersected zones per targets
sum_targets <- bv_sample %>%
  mutate(sum_area = as.numeric(st_area(bv_sample)) / 1000 ^ 2)

# merge with the intersected zones
temp_inters_cells <-  merge(temp_inters_cells,
      st_drop_geometry(sum_targets), by = "BUREAU")
temp_inters_cells$Area_share <- temp_inters_cells$Area_intersect /
  temp_inters_cells$sum_area
temp_inters_cells[, var_intensive] <- sapply(
  st_drop_geometry(temp_inters_cells[, var_intensive]),
  function(x) x * temp_inters_cells$Area_share)
# Aggregate the variables by target
temp_targets <- aggregate(temp_inters_cells[, var_intensive],
                              by = list(temp_inters_cells$BUREAU),
                              FUN = sum)
# Rename the variables
temp_targets <- temp_targets %>%
  rename_at(vars(var_intensive), ~ paste0(var_intensive, "_daw_cells_bad"))

# Merge with the targets
bv_sample <- merge(bv_sample, st_drop_geometry(temp_targets),
                   by.x = "BUREAU", by.y = "Group.1")
```

We have done the same things when the sources are the large cells, but we do not present the codes because they are the same as the ones presented above.

3.3.3 Comparing the two methods

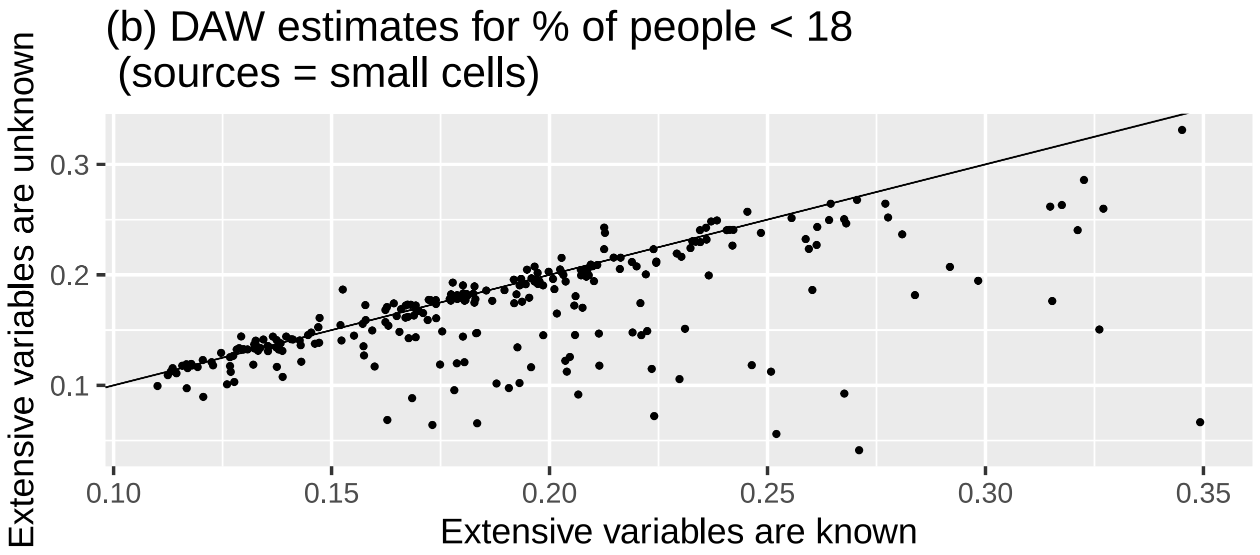
3.3.3.1 Sources = small cells We compare the estimates when using “extensive variables are known” and “extensive variables are unknown”. We do this first for the case where the sources are the small cells, and for the population density variable. We observe that the two methods seem similar. We create Figure 6 in the article through these actions.

```
ggplot(bv_sample) +
  aes(x = pop_dens_daw_cells, y = pop_dens_daw_cells_bad) +
  geom_point()   +
  labs(title = "(a) DAW estimates for pop density \n (sources = small cells)",
       x = "Extensive variables are known",
       y = "Extensive variables are unknown") +
  geom_abline(intercept = 0, slope = 1)    +
   theme_grey(base_size = 20)
```



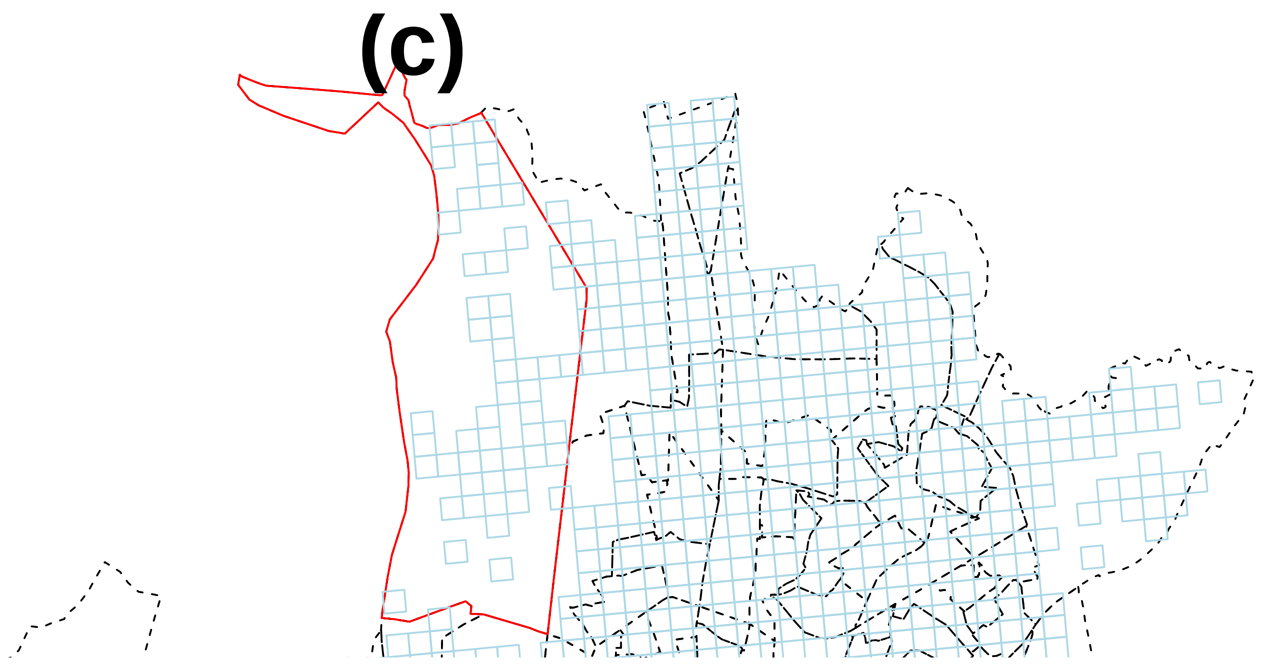
We now find the proportion of people less than 18 years old. The method “extensive variables are unknown” seems clearly underestimated for some targets.

```
ggplot(bv_sample) +
  aes(x = prop_Ind_under_18_daw_cells, y = prop_Ind_under_18_daw_cells_bad) +
  geom_point()   +
  labs(title = "(b) DAW estimates for % of people < 18 \n (sources = small cells)",
      x = "Extensive variables are known",
      y = "Extensive variables are unknown") +
  geom_abline(intercept = 0, slope = 1) +
   theme_grey(base_size = 20)
```



We try to better understand what is happening for these cases. We have represented in red a target where the percentage of young people is estimated as 34.9% with the case "extensive variables are known" and 6.7% with the case "extensive variables are unknown".

```
op <- par(oma= c(3, 2, 1.5, 0), mar = c(0, 0, 0.7, 0))
plot(st_geometry(bv_sample[187,]), border = "red")
plot(st_geometry(bv_sample), border = "black", add = T, lty = 2)
plot(st_geometry(bv_sample[187, ]), border = "red", add = T)
plot(st_geometry(cell_200m), border = "lightblue", add = T)
title("(c)", cex.main = 3, line = -1)
```



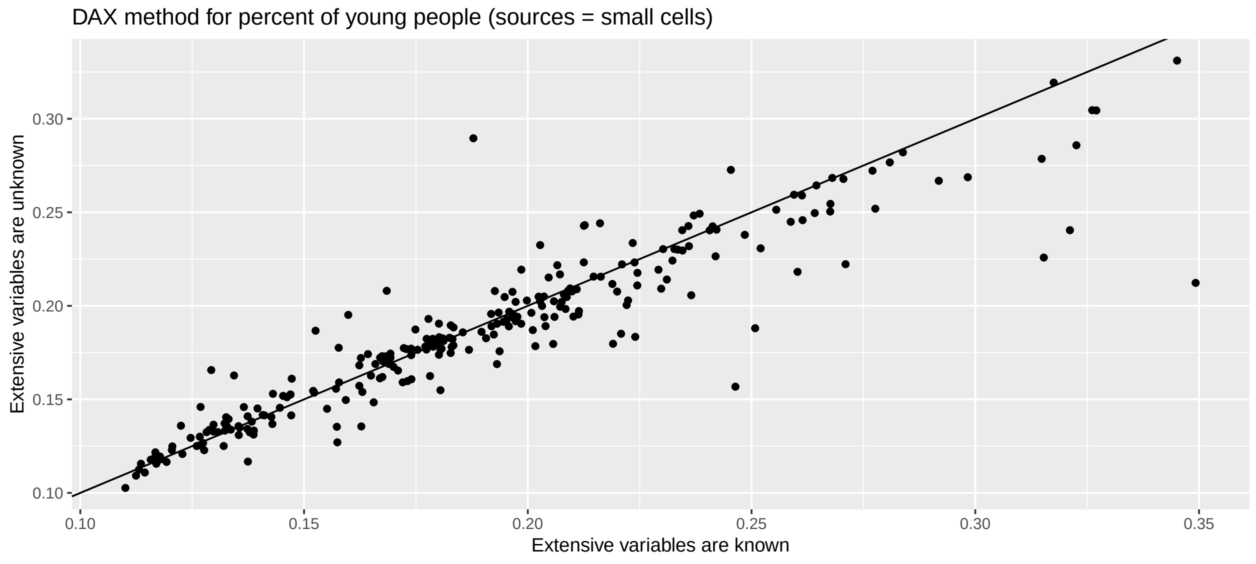
```
par(op)
```

When we consider an intensive variable which does not depend on the area (like the proportion of people less than 18 years old), the formula creates biases. One issue should consist in replacing the term $|T_t|$ by $\sum_s |A_{s,t}|$. This is actually what is done by the function *st_interpolate_aw()* with the option **extensive=F**.

```
a1 <- st_interpolate_aw(cell_200m["prop_Ind_under_18"],
                        bv_sample, extensive = FALSE)
bv_sample$prop_Ind_under_18_daw_sf <- a1$prop_Ind_under_18
```

In that case, we observe fewer differences between the cases “extensive variables are known” or “extensive variables are unknown”.

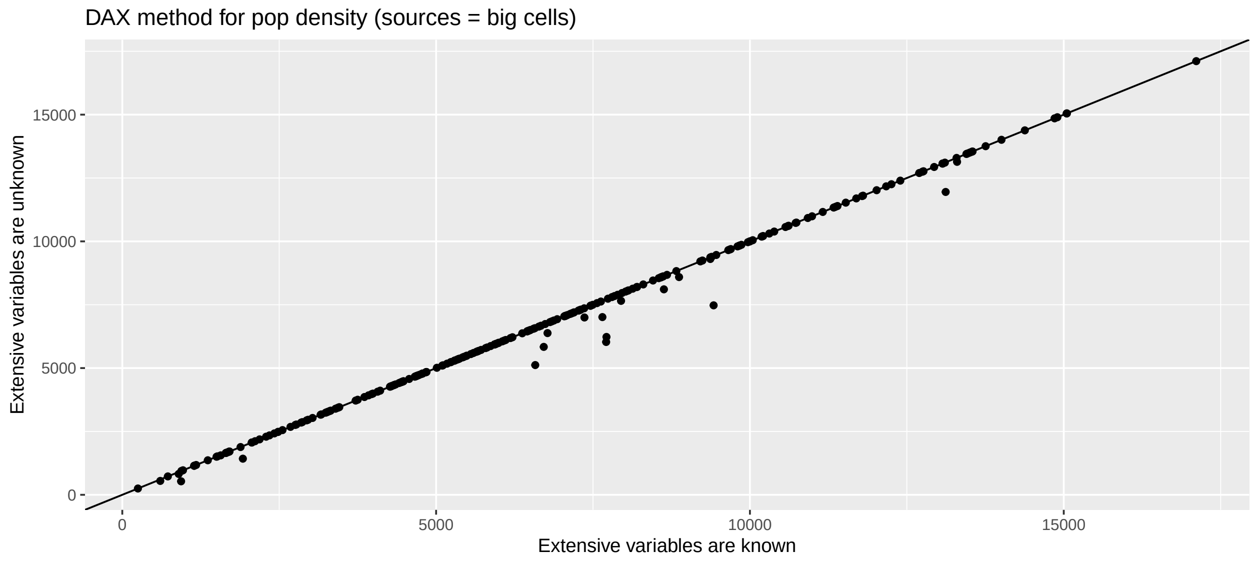
```
ggplot(bv_sample) +
  aes(x = prop_Ind_under_18_daw_cells, y = prop_Ind_under_18_daw_sf) +
  geom_point()   +
  labs(title = "DAX method for percent of young people (sources = small cells)",
       x = "Extensive variables are known",
       y = "Extensive variables are unknown") +
  geom_abline(intercept = 0, slope = 1)
```



This is actually what is performed by the function *st_interpolate_aw()* with the option **extensive=F**.

3.3.3.2 Sources = big cells When the sources are the large cells, the correlation seems stronger. This confirms the remark which has been produced with the PIP method. When the variable is intensive, if sources and targets have more or less the same size, the method "extensive variables are known" vs. "extensive variables are unknown" should give similar results. For example, for the population density:

```
ggplot(bv_sample) +
  aes(x = pop_dens_daw_big, y = pop_dens_daw_big_bad) +
  geom_point()   +
  labs(title = "DAX method for pop density (sources = big cells)",
       x = "Extensive variables are known",
       y = "Extensive variables are unknown") +
  geom_abline(intercept = 0, slope = 1)
```

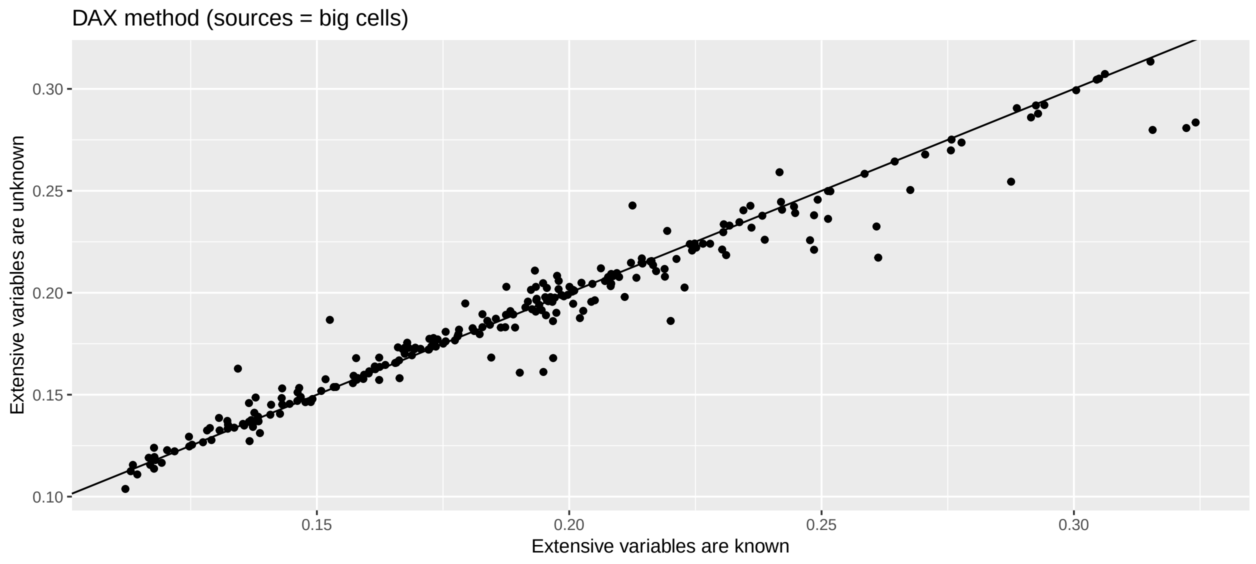


For the variable "percentage of people less than 18 years old", we use the function *st_interpolate_aw()* with the option **extensive=F**.

```r
a1 <- st_interpolate_aw(cell_big["prop_Ind_under_18"],
                        bv_sample, extensive = FALSE)
bv_sample$prop_Ind_under_18_daw_big_sf <- a1$prop_Ind_under_18
```

The two methods “extensive variables are known” vs. “extensive variables are unknown” are also correlated.

```r
ggplot(bv_sample) +
  aes(x = prop_Ind_under_18_daw_big, y = prop_Ind_under_18_daw_big_sf) +
  geom_point()   +
  labs(title = "DAX method (sources = big cells)",
       x = "Extensive variables are known",
       y = "Extensive variables are unknown") +
  geom_abline(intercept = 0, slope = 1)
```



4 Dasymetric method with auxiliary variable X (DAX)

4.1 Extensive variables

Here, we do not take into account the sources which do not intersect with any target (in other terms, we keep the case **exclude** seen previously).

We present here the codes used in the DAX method when the sources are the small cells.

```r
# We find the intersections between the following:
# - (intersection of sources and targets) and road maps
# intersection of sources and targets
temp_inters_cells <- st_intersection(bv_sample[, "BUREAU"],
                     cell_200m[, c(var_extensive, "IdINSPIRE")])
temp_inters_sourcestargets_roads <- st_intersection(
  temp_inters_cells[, c("BUREAU", "IdINSPIRE")],
  voieries[, "id_troncon"])
# compute the length of the roads
temp_inters_sourcestargets_roads$length <-
     as.numeric(st_length(temp_inters_sourcestargets_roads))
```

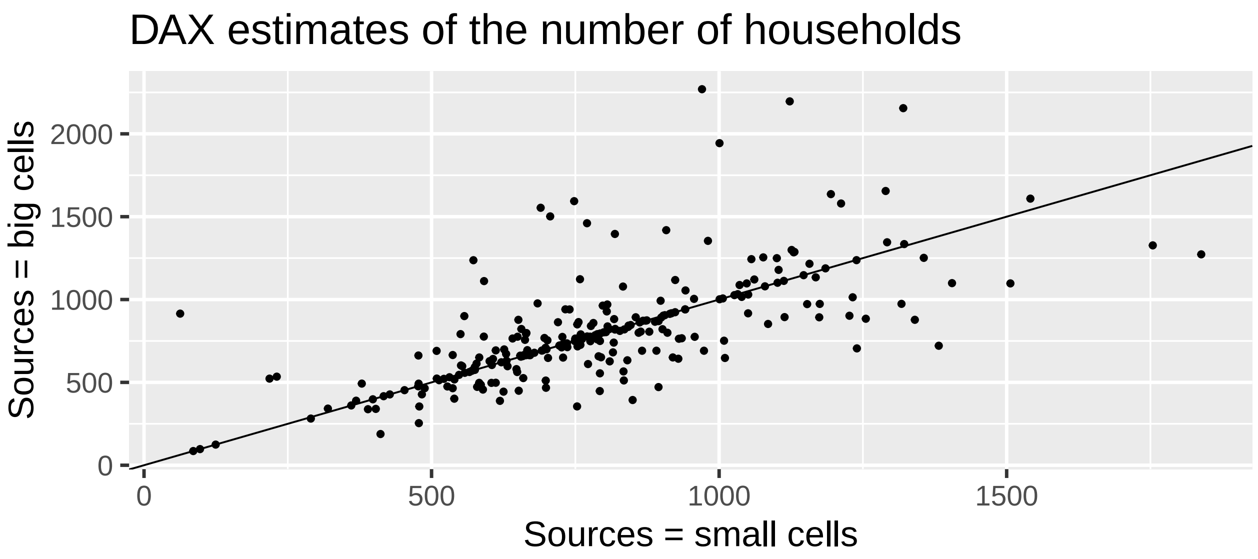
```r
# aggregate the lengths by sources
temp_inters_sources_roads <- st_intersection(
  cell_200m[,  "IdINSPIRE"], voieries[, "id_troncon"])
temp_inters_sources_roads$length <-
     as.numeric(st_length(temp_inters_sources_roads))
temp_sources <- aggregate(temp_inters_sources_roads[, "length"],
          by = list(temp_inters_sources_roads$IdINSPIRE),
          FUN = sum)
temp_sources <- temp_sources %>%
  rename(length_s = length)
# aggregate the lengths by (intersection of sources and targets)
temp_intersects <- aggregate(temp_inters_sourcestargets_roads[, "length"],
           by = list(temp_inters_sourcestargets_roads$BUREAU,
                     temp_inters_sourcestargets_roads$IdINSPIRE),
           FUN = sum)
# add the variable Xst
temp_inters_cells <- merge(temp_inters_cells,
                           st_drop_geometry(temp_intersects),
                     by.x = c("BUREAU", "IdINSPIRE"),
                     by.y = c("Group.1", "Group.2"))
# add the variable Xs
temp_inters_cells <- merge(temp_inters_cells,
                           st_drop_geometry(temp_sources),
                     by.x = "IdINSPIRE",
                     by.y = "Group.1")
temp_inters_cells <- temp_inters_cells %>%
  mutate(ratio = length / length_s)
# apply the ratio with extensive variables
# * for method 2
temp_inters_cells[, paste0(var_extensive)] <- sapply(
  st_drop_geometry(temp_inters_cells[, var_extensive]),
  function(x) x * temp_inters_cells$ratio)

# Aggregate the variables by target
temp_targets <- aggregate(temp_inters_cells[, var_extensive], by =
                            list(temp_inters_cells$BUREAU),
                          FUN = sum)
# Rename the variables
temp_targets <- temp_targets %>%
  rename_at(vars(var_extensive), ~ paste0(var_extensive, "_dax_cells"))

# Merge with the targets
bv_sample <- merge(bv_sample, st_drop_geometry(temp_targets),
                   by.x = "BUREAU", by.y = "Group.1")
```

We have performed the same computations for the large cells taken as sources, but we do not present the codes here because they are similar to previous codes.

4.1.1 Comparaison between the two sources of data

We compare the DAX results with respect to the choice of the sources (small cells or large cells) for the variable "Number of households". Few observations fit with the regression line $y = x$, but we observe differences between the two estimates. As for the PIP or DAW method, we recommend using the sources with the most detailed geographical level.

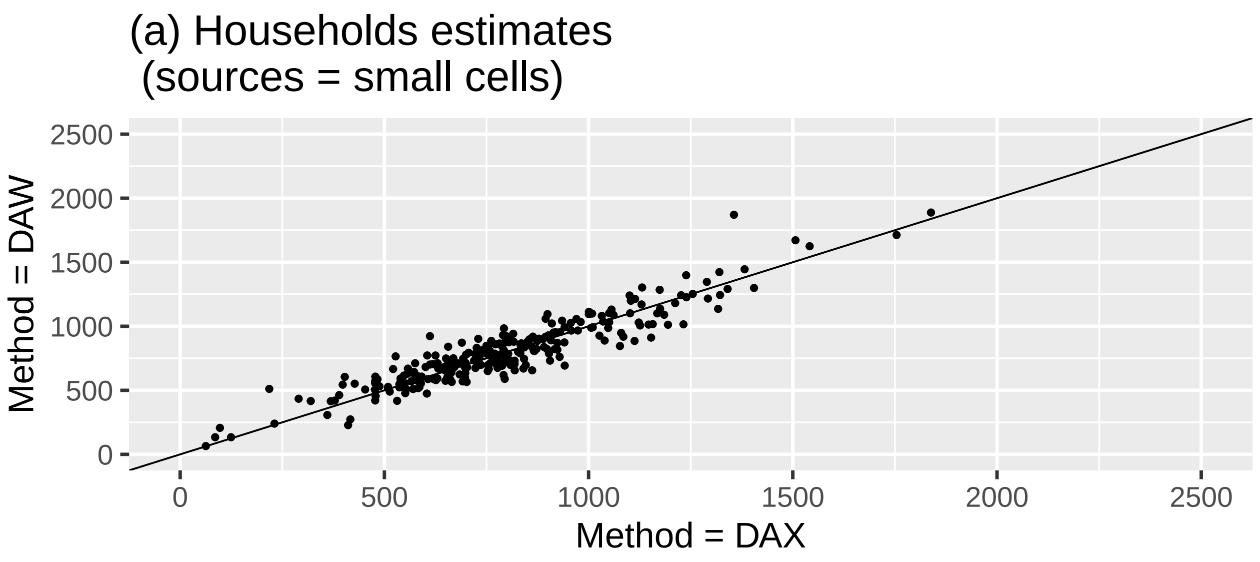
```
ggplot(bv_sample) +
  aes(x = Men_dax_cells, y = Men_dax_big) +
  geom_point() +
  labs(title = "DAX estimates of the number of households",
       x = "Sources = small cells",
       y = "Sources = big cells") +
  geom_abline(intercept = 0, slope = 1) +
   theme_grey(base_size = 20)
```



4.2 Comparaison between DAW and DAX

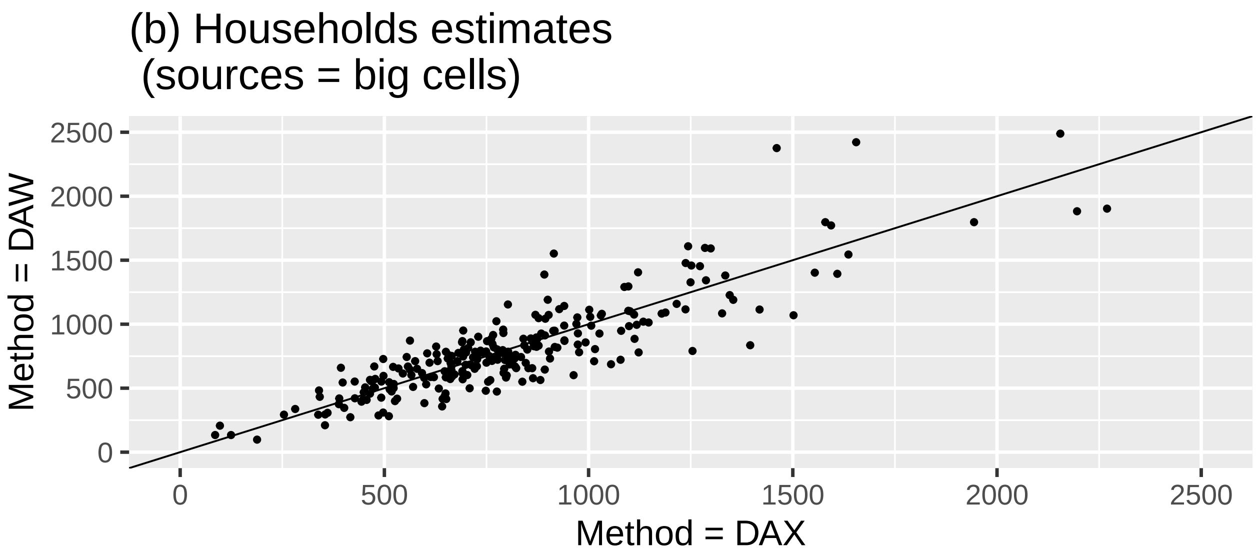
We compare the results obtained with DAX and with DAW on the variable number of households. First, we present the figure in the case where the small cells have been taken as sources. It corresponds to Figure 7 in the article.

```
ggplot(bv_sample) +
  aes(x = Men_dax_cells, y = Men_daw_cells) +
  labs(title = "(a) Households estimates \n (sources = small cells)",
       x = "Method = DAX",
       y = "Method = DAW") +
  geom_point() +
  xlim(0, 2500) +
  ylim(0, 2500) +
  geom_abline(intercept = 0, slope = 1)  +
   theme_grey(base_size = 20)
```



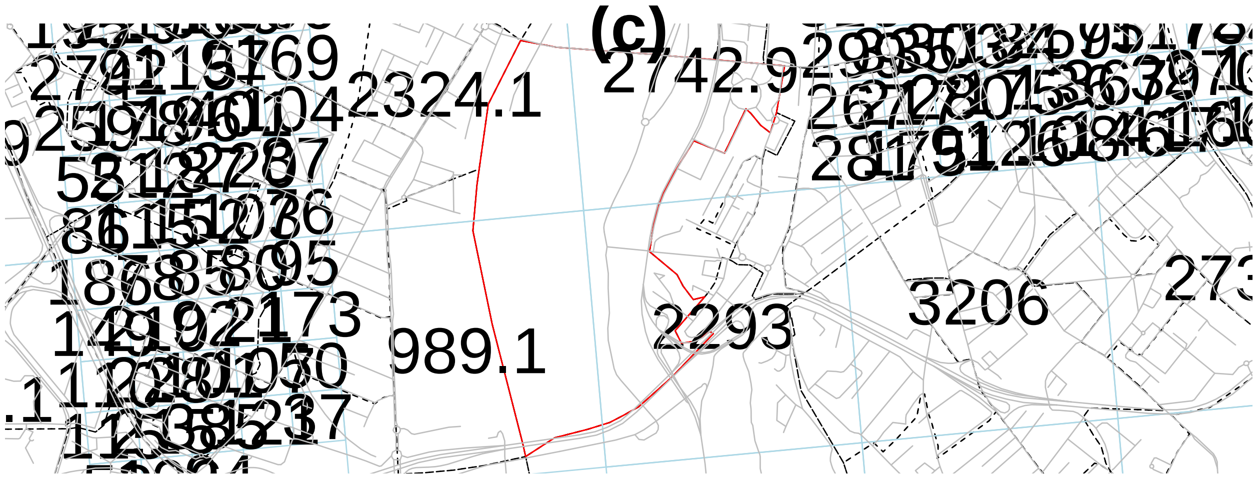
DAW and DAX seem to produce similar results. We now look at the case of the large cells taken as sources.

```r
ggplot(bv_sample) +
  aes(x = Men_dax_big, y = Men_daw_big) +
  labs(title = "(b) Households estimates \n (sources = big cells)",
       x = "Method = DAX",
       y = "Method = DAW") +
  xlim(0, 2500) +
  ylim(0, 2500) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1) +
   theme_grey(base_size = 20)
```



In that case, it appears that the variability between the two methods is higher. Let us try to better understand the differences between the two methods. We have represented in red the target which has the higher difference between DAX and DAW estimates. For this target, the value estimated is 1460 with DAX and 2376 with DAW. We observe that the density of the roads is very low in the target, whereas the area covered by the target is important. This explains why the value is underestimated with DAX as compared to the DAW method.

```
op <- par(oma= c(3, 2, 1.5, 0), mar = c(0, 0, 0.7, 0))
plot(st_geometry(bv_sample[70,]), border = "red")
plot(st_geometry(bv_sample), border = "black", add = T, lty = 2)
plot(st_geometry(bv_sample[70,]), border = "red", add = T)
plot(st_geometry(cell_big), border = "lightblue", add = T)
text(st_coordinates(st_centroid(cell_big))[, 1],
     st_coordinates(st_centroid(cell_big))[, 2],
     cell_big$Men, cex = 3)
plot(st_geometry(voieries), col = "grey", add = T)
title("(c)", cex.main = 3, line = -1)
```



4.3 Intensive variables

4.3.1 Extensive variables are known

If the extensive variables which define the intensive variables are known, the DAW method is applied first on the extensive variable to obtain the estimates at the target levels, and then intensive variables are re-created for the estimates.

```
bv_sample <- bv_sample %>%
  mutate(
    prop_Ind_under_18_dax_cells = Ind_0_17_dax_cells / Ind_dax_cells,
    pop_dens_dax_cells = Ind_dax_cells / area_dax_cells,
    prop_Ind_under_18_dax_big = Ind_0_17_dax_big / Ind_dax_big,
    pop_dens_dax_big = Ind_dax_big / area_dax_big
  )
```

4.3.2 Intensive variable

This works similarly to the DAW method except that we replace the area by the auxiliary information. The formula is $\hat{Y}_t = \sum \frac{x_{s,t}}{\sum_s x_{st}} Y_s$. Note that we have chosen $\sum_s x_{st}$ instead of x_t for the reason that we have seen in the DAW section.

```
# We find the intersections between the following:
# - (intersection of sources and targets) and road maps
# intersection of sources and targets
```

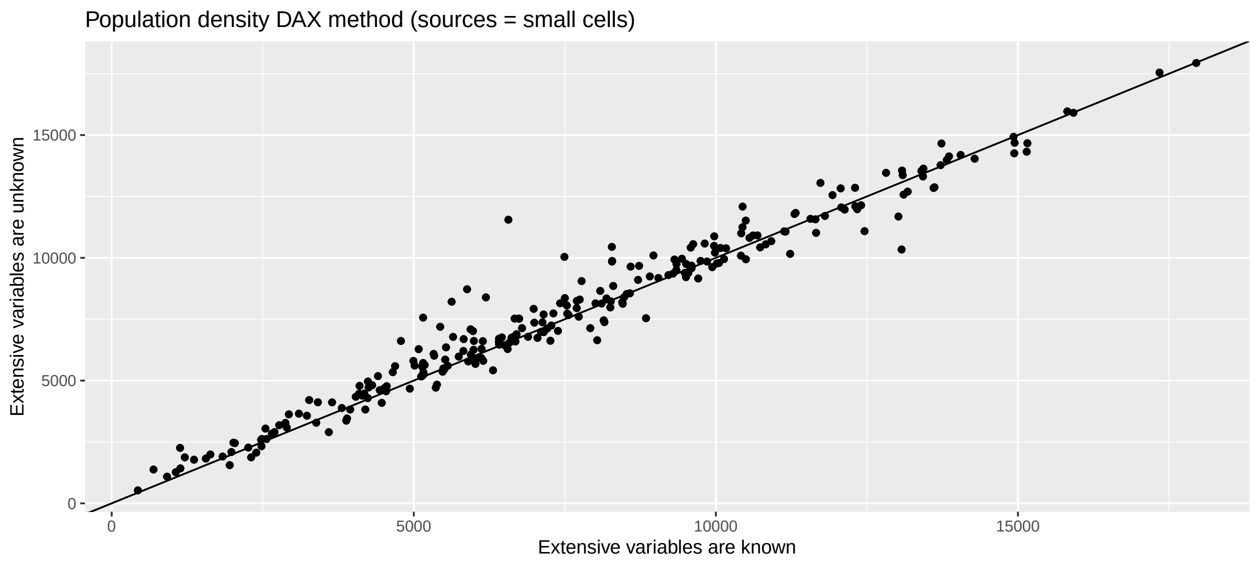
```r
temp_inters_cells <- st_intersection(bv_sample[, "BUREAU"],
                  cell_200m[, c(var_intensive, "IdINSPIRE")])
temp_inters_sourcestargets_roads <- st_intersection(
  temp_inters_cells[, c("BUREAU", "IdINSPIRE")],
  voieries[, "id_troncon"])
# compute the length of the roads
temp_inters_sourcestargets_roads$length <-
     as.numeric(st_length(temp_inters_sourcestargets_roads))

# aggregate the lengths by targets
temp_targets <- aggregate(temp_inters_sourcestargets_roads[, "length"],
                          by = list(temp_inters_sourcestargets_roads$BUREAU),
                          sum)
temp_targets <- temp_targets %>%
  rename(length_t = length)

# aggregate the lengths by (intersection of sources and targets)
temp_intersects <- aggregate(temp_inters_sourcestargets_roads[, "length"],
                 by = list(temp_inters_sourcestargets_roads$BUREAU,
                           temp_inters_sourcestargets_roads$IdINSPIRE),
                 FUN = sum)
# add the variable Xst
temp_inters_cells <- merge(temp_inters_cells, st_drop_geometry(temp_intersects),
                     by.x = c("BUREAU", "IdINSPIRE"),
                     by.y = c("Group.1", "Group.2"))
# add the variable Xs
temp_inters_cells <- merge(temp_inters_cells, st_drop_geometry(temp_targets),
                     by.x = "BUREAU",
                     by.y = "Group.1")
temp_inters_cells <- temp_inters_cells %>%
  mutate(ratio = length / length_t)
# apply the ratio with extensive variables
temp_inters_cells[, paste0(var_intensive)] <- sapply(
  st_drop_geometry(temp_inters_cells[, var_intensive]),
  function(x) x * temp_inters_cells$ratio)

# Aggregate the variables by target
temp_targets <- aggregate(temp_inters_cells[, var_intensive], by =
                            list(temp_inters_cells$BUREAU),
                          FUN = sum)
# Rename the variables
temp_targets <- temp_targets %>%
  rename_at(vars(var_intensive), ~ paste0(var_intensive, "_dax_cells_bad"))

# Merge with the targets
bv_sample <- merge(bv_sample, st_drop_geometry(temp_targets),
                   by.x = "BUREAU",
                   by.y = "Group.1")
```

We have performed the same computations for the small cells taken as sources, but we do not present the codes here because they are similar to what has been done previously.

4.3.3 Comparaison between the two methods

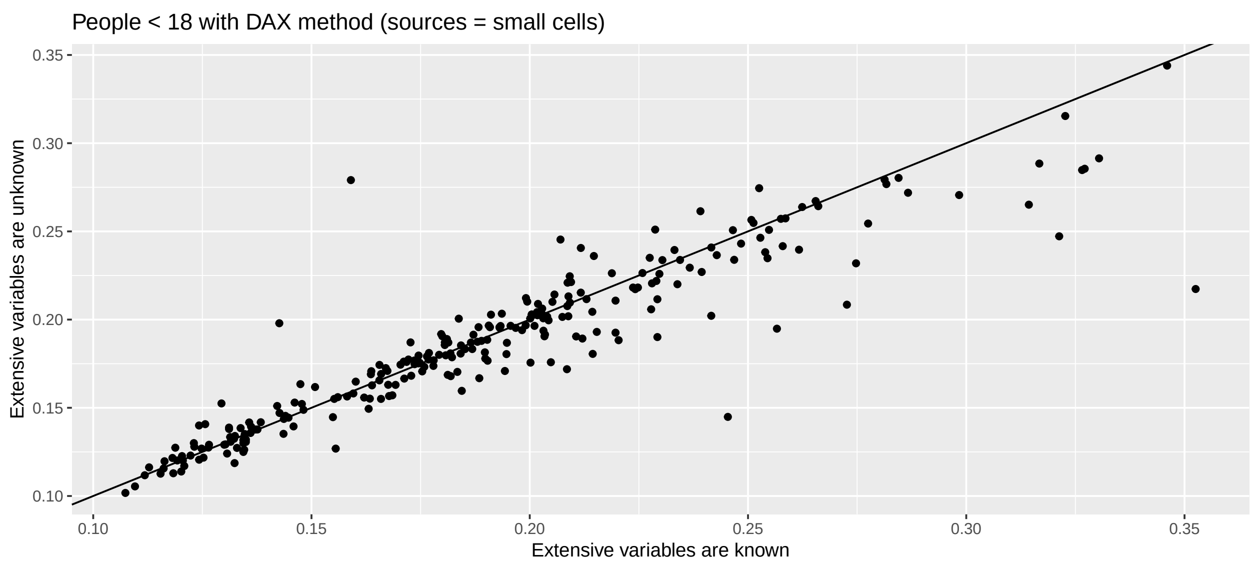
4.3.3.1 Sources = small cells We compare the estimates when using “extensive variables are known” versus “extensive variables are unknown”. We do this first for the variable population density.

```r
ggplot(bv_sample) +
  aes(x = pop_dens_dax_cells, y = pop_dens_dax_cells_bad) +
  geom_point() +
  labs(title = "Population density DAX method (sources = small cells)",
       x = "Extensive variables are known",
       y = "Extensive variables are unknown") +
  geom_abline(intercept = 0, slope = 1)
```



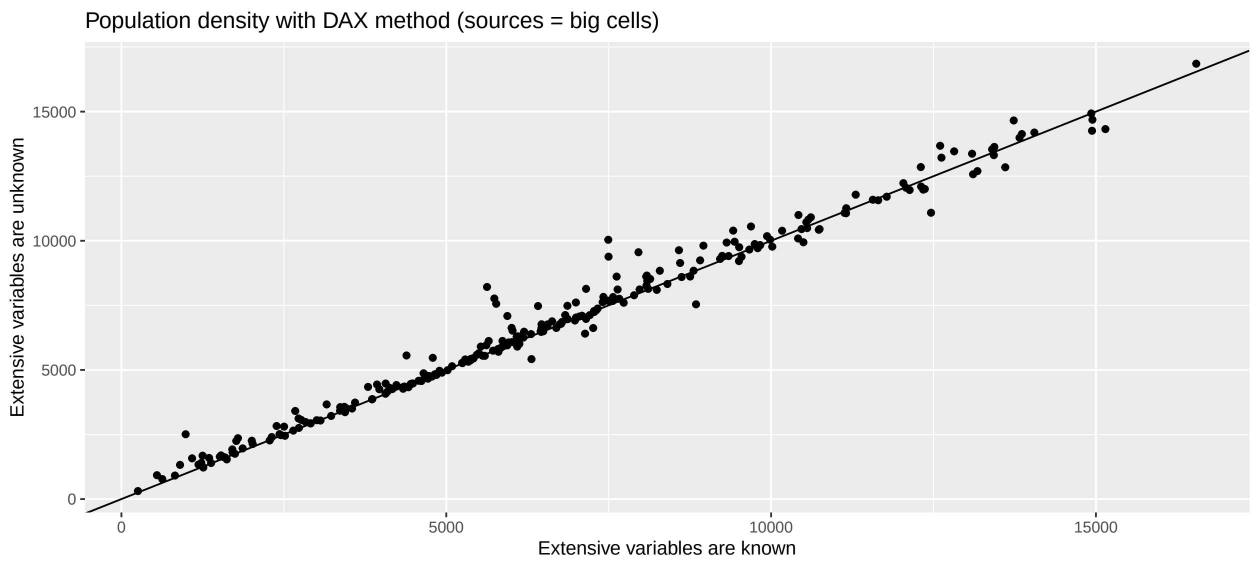
We remark that the scatter plot fits well around the curve $y = x$. The same remark can be offered for the variable “Percentage of people under 18” even if there is more variability here.

```r
ggplot(bv_sample) +
  aes(x = prop_Ind_under_18_dax_cells, y = prop_Ind_under_18_dax_cells_bad) +
  geom_point() +
  labs(title = "People < 18 with DAX method (sources = small cells)",
       x = "Extensive variables are known",
       y = "Extensive variables are unknown") +
  geom_abline(intercept = 0, slope = 1)
```

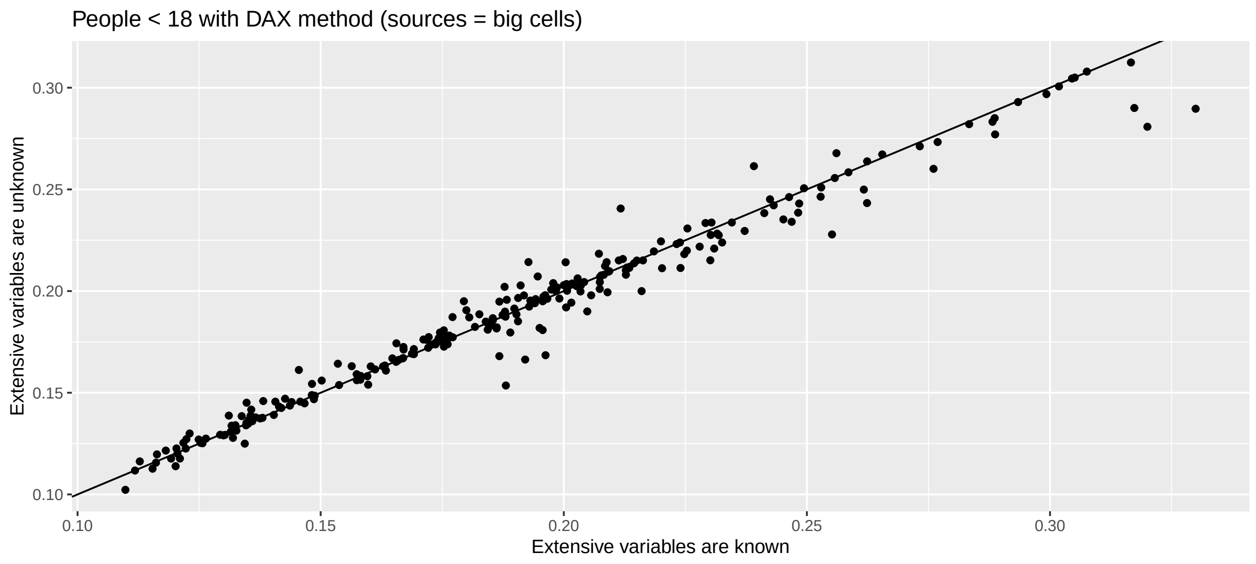


4.3.3.2 Sources = large cells When the sources are the large cells, the two methods "extensive variables are known" and "extensive variables are unknown" fit quite well.

```
ggplot(bv_sample) +
  aes(x = pop_dens_dax_big, y = pop_dens_dax_big_bad) +
  geom_point() +
  labs(title = "Population density with DAX method (sources = big cells)",
       x = "Extensive variables are known",
       y = "Extensive variables are unknown") +
  geom_abline(intercept = 0, slope = 1)
```



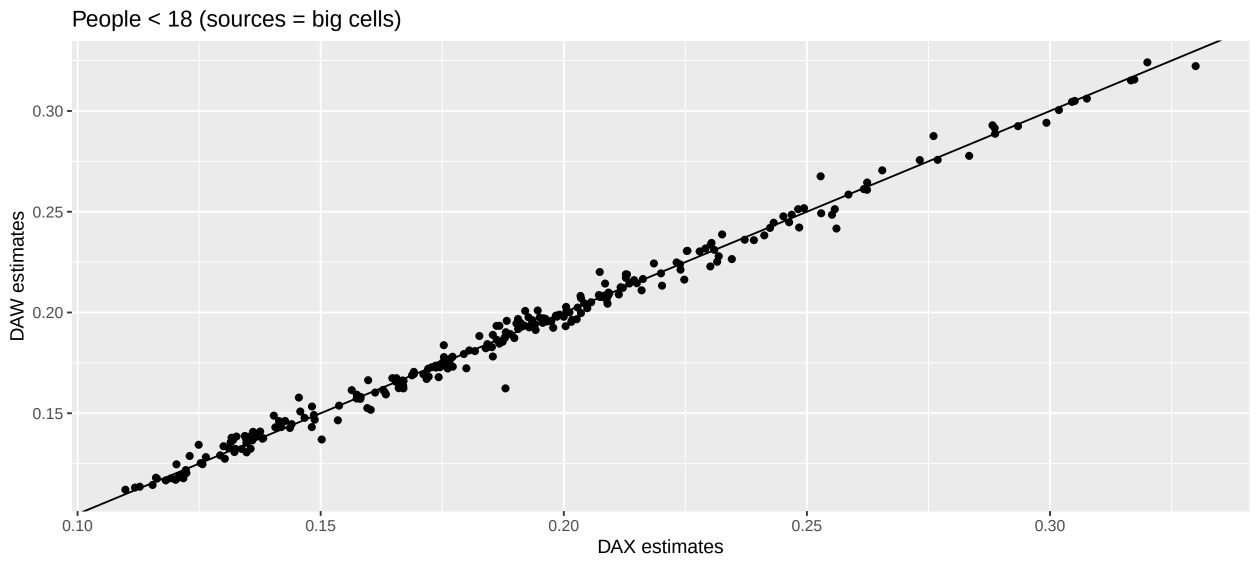
```
ggplot(bv_sample) +
  aes(x = prop_Ind_under_18_dax_big, y = prop_Ind_under_18_dax_big_bad) +
  geom_point() +
  labs(title = "People < 18 with DAX method (sources = big cells)",
       x = "Extensive variables are known",
       y = "Extensive variables are unknown") +
  geom_abline(intercept = 0, slope = 1)
```



4.4 Comparaison between DAX and DAW

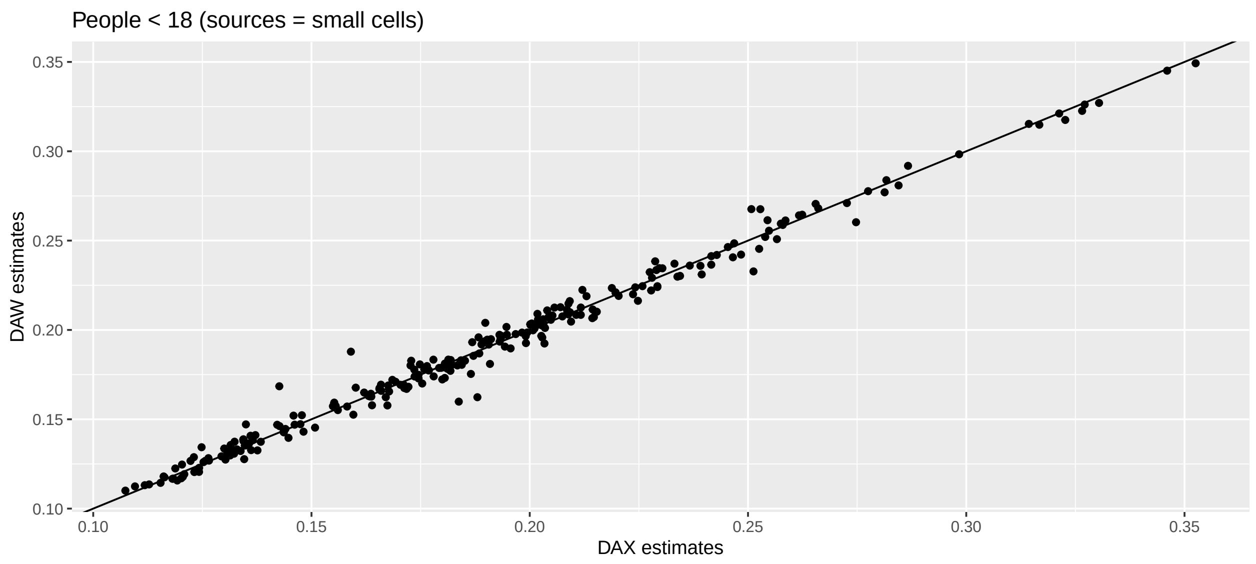
For the large cells:

```
ggplot(bv_sample) +
  aes(x = prop_Ind_under_18_dax_big, y = prop_Ind_under_18_daw_big) +
  geom_point() +
  labs(title = "People < 18 (sources = big cells)",
       x = "DAX estimates",
       y = "DAW estimates") +
  geom_abline(intercept = 0, slope = 1)
```



For the small cells:

```r
ggplot(bv_sample) +
  aes(x = prop_Ind_under_18_dax_cells, y = prop_Ind_under_18_daw_cells) +
  geom_point() +
  labs(title = "People < 18 (sources = small cells)",
       x = "DAX estimates",
       y = "DAW estimates") +
  geom_abline(intercept = 0, slope = 1)
```



5 Dasymetric method with control zones

5.1 Step 1: DAW method

The aim is to apply the DAW method by using the control zones as sources and the intersections between iris and polling places as targets. The variable of interest is the “number of individuals”. We first define the geometries of the intersections between iris and polling places:

```r
# The targets
control <- st_intersection(bv_sample[, "BUREAU"],
                                    iris[, c(var_extensive_rp, "IRIS")])
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

```r
control <- control %>%
  mutate(ID = paste0(IRIS, "_", BUREAU))
```

We can then apply the DAW method by considering the small cells as the sources and the intersections between iris and polling places as the targets.

```r
# intersection of sources and targets
temp_inters_cell_200m <- st_intersection(control[, "ID"],
                  cell_200m[, c("Ind", "area", "IdINSPIRE")])
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

```
# compute the areas of the intersections divided by the area of the source
temp_inters_cell_200m$Area_intersect <-
  as.numeric(st_area(temp_inters_cell_200m)) / 1000 ^ 2

# compute the sum of the intersected zones per source
sum_intersect_cell_200m <- temp_inters_cell_200m %>%
  select(Area_intersect, IdINSPIRE) %>%
  group_by(IdINSPIRE) %>%
  summarise(sum_area = sum(Area_intersect))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# merge with the interesected zones
temp_inters_cell_200m <-  merge(temp_inters_cell_200m,
                                st_drop_geometry(sum_intersect_cell_200m),
                                by = "IdINSPIRE")
temp_inters_cell_200m$Area_share_1 <- temp_inters_cell_200m$Area_intersect /
  temp_inters_cell_200m$area

# Multiply the variables by the proportions
temp_inters_cell_200m[, "Ind"] <-
  sapply(st_drop_geometry(temp_inters_cell_200m[, "Ind"]),
         function(x) x * temp_inters_cell_200m$Area_share_1)

# Aggregate the variables by target
temp_targets <- aggregate(temp_inters_cell_200m[, "Ind"],
                          by = list(temp_inters_cell_200m$ID),
                          FUN = sum)
# Rename the variables
temp_targets <- temp_targets %>%
  rename(Ind_daw = Ind)

# Merge with the targets
control <- merge(control, st_drop_geometry(temp_targets),
                 by.x = "ID", by.y = "Group.1")
```

5.2 Step 2: DAX method

The aim is to use the iris as sources and the polling places as targets. We use here the auxiliary information created in step 1: the number of inhabitants estimated at the intersection levels.

We apply these methods on the extensive variables.

```
# aggregate the X by sources
control_sources <- aggregate(control[, "Ind_daw"],
                          by = list(control$IRIS),
                          FUN = sum)
control_sources <- control_sources %>%
  rename(Ind_daw_s = Ind_daw)

# aggregate the lengths by (intersection of sources and targets)
# add the variable Xs
control <- merge(control, st_drop_geometry(control_sources),
                         by.x = "IRIS",
```

```r
                          by.y = "Group.1")
control <- control %>%
  mutate(ratio = Ind_daw / Ind_daw_s)

# apply the ratio with extensive variables
control[, paste0(var_extensive_rp)] <- sapply(
  st_drop_geometry(control[, var_extensive_rp]),
  function(x) x * control$ratio)

# Aggregate the variables by target
temp_targets <- aggregate(control[, var_extensive_rp], by =
                             list(control$BUREAU),
                           FUN = sum)
# Rename the variables
temp_targets <- temp_targets %>%
  rename_at(vars(var_extensive_rp), ~ paste0(var_extensive_rp, "_dac"))
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(var_extensive_rp)` instead of `var_extensive_rp` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```r
# Merge with the targets
bv_sample <- merge(bv_sample, st_drop_geometry(temp_targets),
                   by.x = "BUREAU",
                   by.y = "Group.1")
```

5.2.1 Comparaison between DAC and DAX

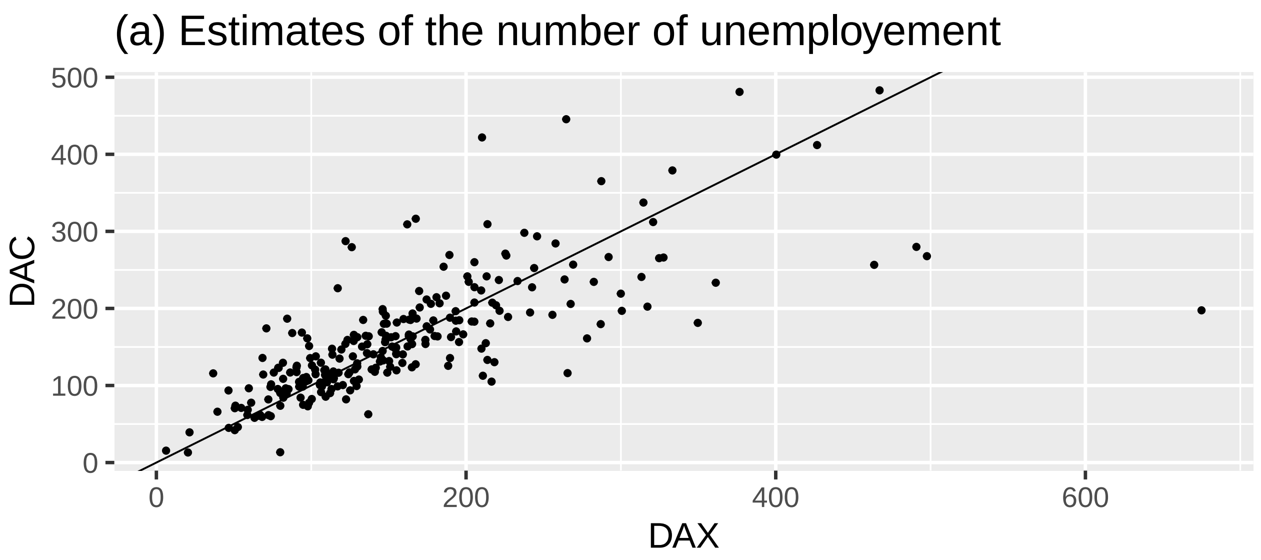
We have also computed the estimates for the DAX method. We do not present the codes because they are similar to those presented in the previous section.

```r
source("codes_DAX_iris.R")
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries

## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries

## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

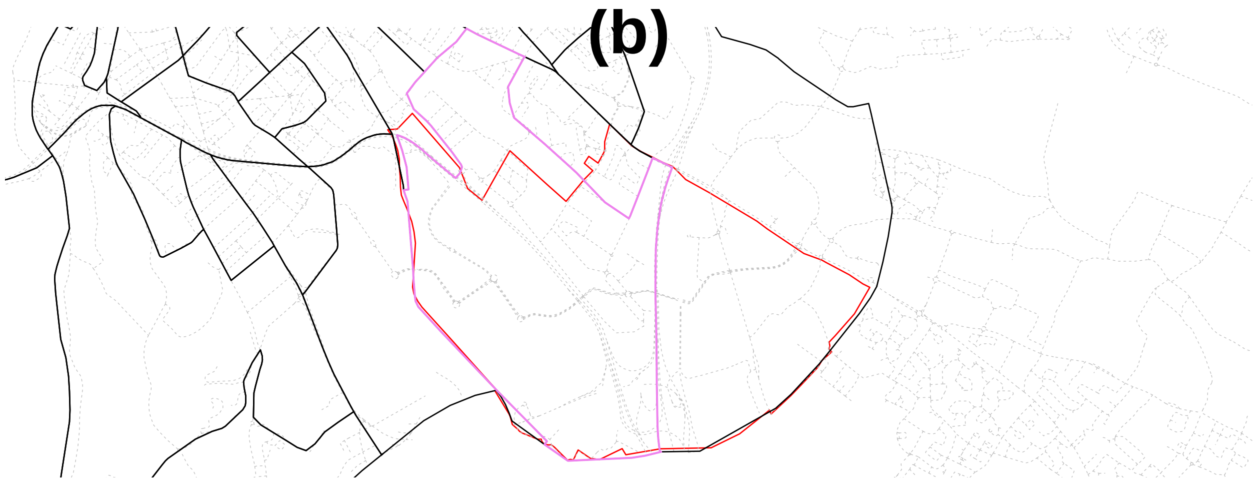
We compare the results for the two methods DAX and DAW for the variable “number of unemployed persons” to obtain Figure 8 in the article.

```r
# Comparaison DAC VS DAX
ggplot(bv_sample) +
  aes(x = P13_CHOM1564_dax, y = P13_CHOM1564_dac) +
  geom_point()  +
  labs(title = "(a) Estimates of the number of unemployement",
       x = "DAX",
       y = "DAC") +
  geom_abline(intercept = 0, slope = 1) +
   theme_grey(base_size = 20)
```



We observe some targets with very different estimates. Let us have a look at what is happening for one of these targets represented in red in the following figure. The estimated value with DAX is equal to 674 and to 197 with DAC. In the DAX method, the main contribution to this value comes from the source represented in violet which shares 82% of the roads represented in grey with the target.

```r
op <- par(oma= c(3, 2, 1.5, 0), mar = c(0, 0, 0.7, 0))
plot(st_geometry(bv_sample[c(239, 241), ]), lty = 2, border = "white")
plot(st_geometry(bv_sample[239, ]), border = "red", add = T)
plot(st_geometry(voieries), col = "grey", add = T, lty = 2, lwd = 0.5)
plot(st_geometry(iris), border = "black", add = T)
plot(st_geometry(iris[iris$IRIS == "315554702", ]),
     border = "violet", add = T, lwd = 1.5)
title("(b)", cex.main = 3, line = -1)
```

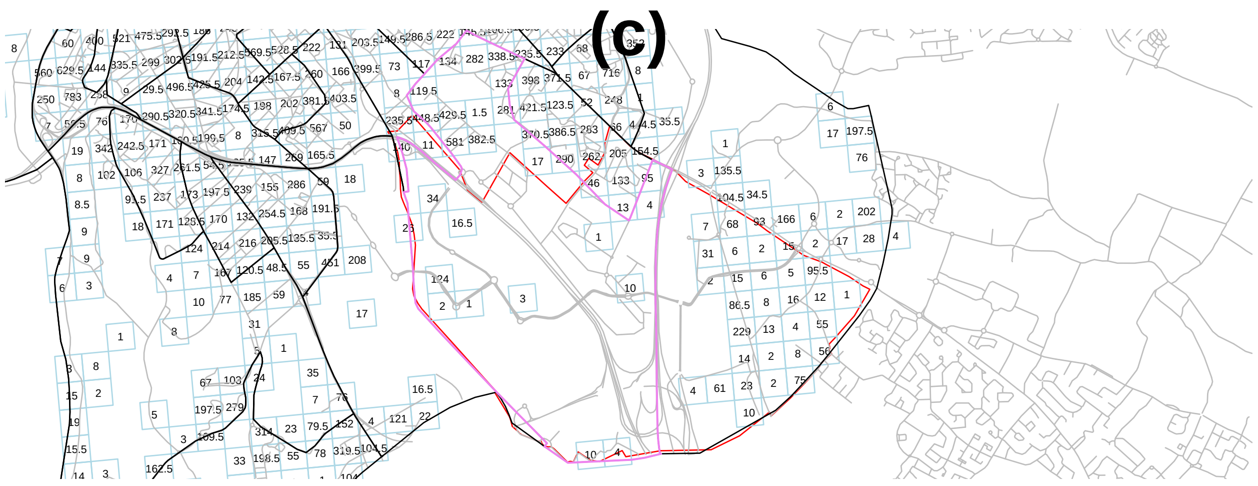


What is happening with the DAC 2 step method? We remark in the following figure that the intersected zone between the target in red and the source in violet does not contain a large population with regard to the control zones represented in light blue. In that case, the source in violet will attribute the main part of its value to the non-intersected zone with the target, which is a zone with a large population.

```
op <- par(oma= c(3, 2, 1.5, 0), mar = c(0, 0, 0.7, 0))
plot(st_geometry(bv_sample[c(239, 241), ]), lty = 2, border = "white")
plot(st_geometry(bv_sample[239, ]), border = "red", add = T)
plot(st_geometry(cell_200m), border = "lightblue", add = T)
text(st_coordinates(st_centroid(cell_200m))[, 1],
     st_coordinates(st_centroid(cell_200m))[, 2],
     paste0(cell_200m$Ind), cex = 0.5)
```

```
## Warning in st_centroid.sf(cell_200m): st_centroid assumes attributes are
## constant over geometries of x

## Warning in st_centroid.sf(cell_200m): st_centroid assumes attributes are
## constant over geometries of x
```

```
plot(st_geometry(voieries), col = "grey", add = T)
plot(st_geometry(iris), border = "black", add = T)
plot(st_geometry(iris[iris$IRIS == "315554702", ]),
     border = "violet", add = T, lwd = 1.5)
title("(c)", cex.main = 3, line = -1)
```



6 Regression Modelling

6.1 Construction of the covariates

We construct the covariates population density, percentage of individuals less than 18, percentage of individuals between 18 and 40, percentage of individuals between 40 and 64, percentage of individuals above 65, percentage of poor households, percentage of owners, and percentage of recent dwellings thanks to the variables provided by the INSEE at the cell level by using the DAX method.

We use the estimates obtained for the extensive variables and then compute the ratio. For the small cells:

```
bv_sample <- bv_sample %>%
  mutate(
    pop_dens = Ind_dax_cells / area_dax_cells,
    prop_Ind_18_40 = (Ind_18_24_dax_cells + Ind_25_39_dax_cells) / Ind_dax_cells,
    prop_Ind_40_64 = (Ind_40_54_dax_cells + Ind_55_64_dax_cells) / Ind_dax_cells,
    prop_Ind_up_65 = (Ind_65_79_dax_cells + Ind_80p_dax_cells) / Ind_dax_cells,
    prop_pour_hos = Men_pauv_dax_cells / Men_dax_cells,
    prop_owner = Men_prop_dax_cells / Men_dax_cells,
    prop_recent = Log_ap90_dax_cells / Men_dax_cells
  )
```

We do the same for PIP method:

```
bv_sample_pip <- bv_sample
bv_sample_pip <- bv_sample_pip %>%
  mutate(
    pop_dens = Ind_pip_cells / area_pip_cells,
    prop_Ind_18_40 = (Ind_18_24_pip_cells + Ind_25_39_pip_cells) / Ind_pip_cells,
    prop_Ind_40_64 = (Ind_40_54_pip_cells + Ind_55_64_pip_cells) / Ind_pip_cells,
    prop_Ind_up_65 = (Ind_65_79_pip_cells + Ind_80p_pip_cells) / Ind_pip_cells,
    prop_pour_hos = Men_pauv_pip_cells / Men_pip_cells,
    prop_owner = Men_prop_pip_cells / Men_pip_cells,
    prop_recent = Log_ap90_pip_cells / Men_pip_cells
  )
```

We do the same for DAW method:

```
bv_sample_daw <- bv_sample
bv_sample_daw <- bv_sample_daw %>%
  mutate(
    pop_dens = Ind_daw_cells / area_daw_cells,
    prop_Ind_18_40 = (Ind_18_24_daw_cells + Ind_25_39_daw_cells) / Ind_daw_cells,
    prop_Ind_40_64 = (Ind_40_54_daw_cells + Ind_55_64_daw_cells) / Ind_daw_cells,
    prop_Ind_up_65 = (Ind_65_79_daw_cells + Ind_80p_daw_cells) / Ind_daw_cells,
    prop_pour_hos = Men_pauv_daw_cells / Men_daw_cells,
    prop_owner = Men_prop_daw_cells / Men_daw_cells,
    prop_recent = Log_ap90_daw_cells / Men_daw_cells
  )
```

For the large cells:

```
bv_sample_big <- bv_sample
bv_sample_big <- bv_sample_big %>%
  mutate(
    pop_dens = Ind_dax_big / area_dax_big,
    prop_Ind_18_40 = (Ind_18_24_dax_big + Ind_25_39_dax_big) / Ind_dax_big,
    prop_Ind_40_64 = (Ind_40_54_dax_big + Ind_55_64_dax_big) / Ind_dax_big,
    prop_Ind_up_65 = (Ind_65_79_dax_big + Ind_80p_dax_big) / Ind_dax_big,
    prop_pour_hos = Men_pauv_dax_big / Men_dax_big,
    prop_owner = Men_prop_dax_big/ Men_dax_big,
    prop_recent = Log_ap90_dax_big / Men_dax_big
  )
```

We do the same for PIP method:

```
bv_sample_pip_big <- bv_sample
bv_sample_pip_big <- bv_sample_pip_big %>%
  mutate(
      pop_dens = Ind_pip_big / area_pip_big,
    prop_Ind_18_40 = (Ind_18_24_pip_big + Ind_25_39_pip_big) / Ind_pip_big,
    prop_Ind_40_64 = (Ind_40_54_pip_big + Ind_55_64_pip_big) / Ind_pip_big,
    prop_Ind_up_65 = (Ind_65_79_pip_big + Ind_80p_pip_big) / Ind_pip_big,
    prop_pour_hos = Men_pauv_pip_big / Men_pip_big,
    prop_owner = Men_prop_pip_big/ Men_pip_big,
    prop_recent = Log_ap90_pip_big / Men_pip_big
  )
```

We do the same for DAW method:

```
bv_sample_daw_big <- bv_sample
bv_sample_daw_big <- bv_sample_daw_big %>%
  mutate(
      pop_dens = Ind_daw_big / area_daw_big,
    prop_Ind_18_40 = (Ind_18_24_daw_big + Ind_25_39_daw_big) / Ind_daw_big,
    prop_Ind_40_64 = (Ind_40_54_daw_big + Ind_55_64_daw_big) / Ind_daw_big,
    prop_Ind_up_65 = (Ind_65_79_daw_big + Ind_80p_daw_big) / Ind_daw_big,
    prop_pour_hos = Men_pauv_daw_big / Men_daw_big,
    prop_owner = Men_prop_daw_big/ Men_daw_big,
    prop_recent = Log_ap90_daw_big / Men_daw_big
  )
```

We then construct the covariates unemployment rate, immigration rate, percentage of farmers, percentage of intermediate/highly qualified jobs, percentage of workers, and percentage of retired people thanks to the variables provided by the INSEE at the iris level by using the DAC 2 step method.

We use the estimates obtained on the extensive variables and then compute the ratio.

```
bv_sample <- bv_sample %>%
  mutate(prop_immi = P15_POP_IMM_dac / P15_POP_dac,
         prop_unemploy = P13_CHOM1564_dac / P13_ACT1564_dac,
         prop_csp_1 = (C15_POP15P_CS1_dac + C15_POP15P_CS2_dac) / C15_POP15P_dac,
         prop_csp_2 = (C15_POP15P_CS3_dac + C15_POP15P_CS4_dac) / C15_POP15P_dac,
         prop_csp_3 = (C15_POP15P_CS5_dac + C15_POP15P_CS6_dac) / C15_POP15P_dac
  )
```

6.2 Exploratory Analysis

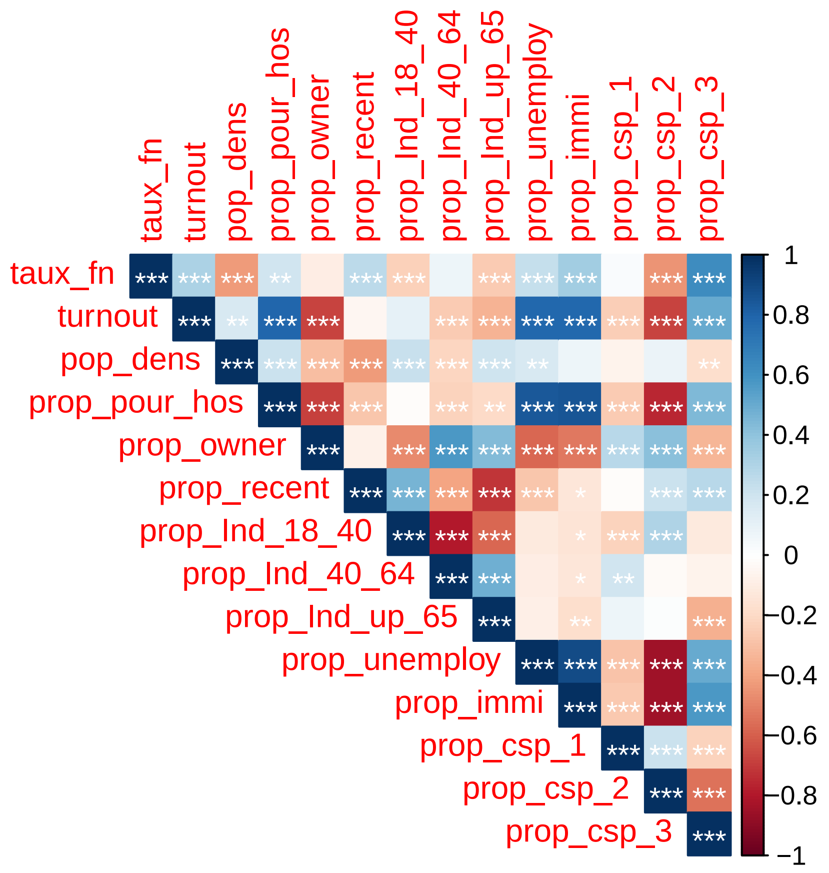
The correlation plot indicates that the extreme right vote is **positively** correlated with:

- proportion of people younger than 18,
- proportion of immigrants,
- turnout,
- proportion of workers,
- proportion of recent housing

It is **negatively** correlated with

- percentage of high qualified professions
- population density
- percentage of people between 18 and 40
- percentage of people up to 65

```
my_data <- st_drop_geometry(bv_sample[, c("taux_fn", "turnout", "pop_dens", "prop_pour_hos",
                       "prop_owner", "prop_recent",
                       "prop_Ind_18_40", "prop_Ind_40_64", "prop_Ind_up_65",
                       "prop_unemploy", "prop_immi", "prop_csp_1", "prop_csp_2",
                       "prop_csp_3"), ])
M <- cor(my_data)
res1 <- cor.mtest(my_data, conf.level = .95)
corrplot(M, p.mat = res1$p, method = "color", type = "upper",
         sig.level = c(.001, .01, .05), pch.cex = .9,
         insig = "label_sig", pch.col = "white")
```

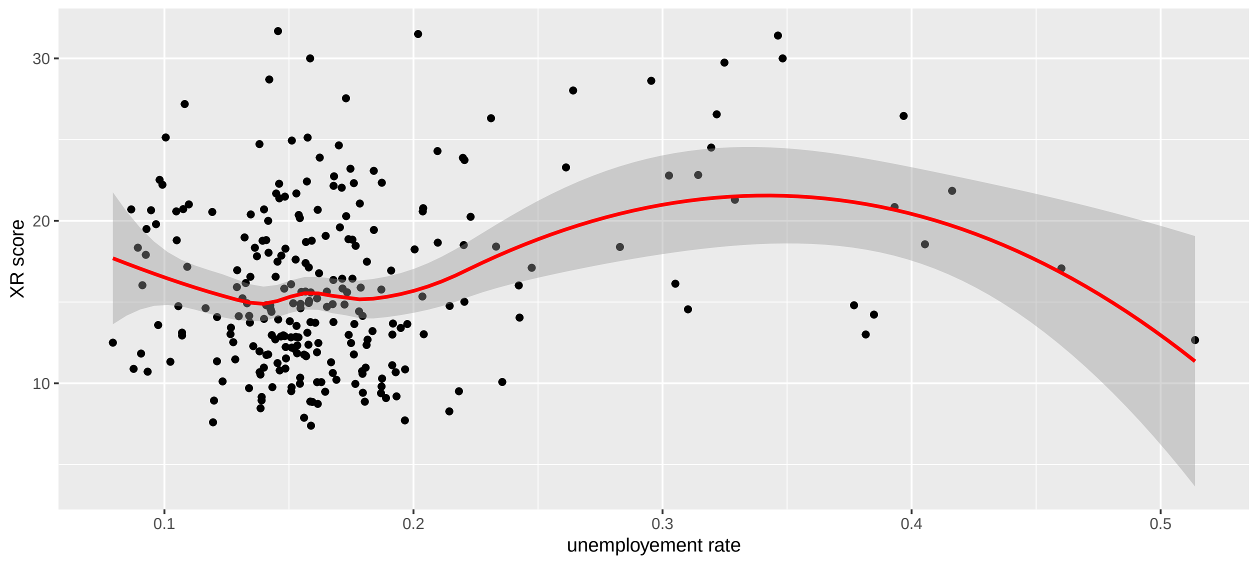


Remark 1: the covariates are strongly correlated, which lets us presume that the linear model should include collinearity.

Remark 2: the relationships between the extreme right vote and the covariates are not necessarily linear. For example, if we look at the scatter plot of the extreme right voting with respect to the unemployment rate, the relationship is not linear.

```
ggplot(data = bv_sample) +
  aes(x = prop_unemploy, y = taux_fn) +
  geom_point() +
  geom_smooth(method = "loess",
              col = "red")  +
  labs(x = "unemployement rate", y = "XR score")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Linear modelling is based on the hypothesis that covariates are supposed as linearly independent, which is not the case here. For that reason, we will propose two regression modelling approaches:

- Linear modelling
- Regression tree

6.3 Regression modelling

6.3.1 Linear modelling

```
res_lm_1_big <- lm(taux_fn ~ turnout + pop_dens + prop_pour_hos +
                prop_owner + prop_recent +
                prop_Ind_18_40 +
                prop_Ind_40_64 +
                prop_Ind_up_65,
            data = bv_sample_pip_big)
 res_lm_2_big <- lm(taux_fn ~ turnout + pop_dens + prop_pour_hos +
                prop_owner + prop_recent +
                prop_Ind_18_40 +
                prop_Ind_40_64 +
                prop_Ind_up_65,
            data = bv_sample_daw_big)
 res_lm_3_big <- lm(taux_fn ~ turnout + pop_dens + prop_pour_hos +
                prop_owner + prop_recent +
                prop_Ind_18_40 +
                prop_Ind_40_64 +
                prop_Ind_up_65,
            data = bv_sample_big)

 res_lm_1 <- lm(taux_fn ~ turnout + pop_dens + prop_pour_hos +
                prop_owner + prop_recent +
                prop_Ind_18_40 +
                prop_Ind_40_64 +
                prop_Ind_up_65,
            data = bv_sample_pip)
```

```
  res_lm_2 <- lm(taux_fn ~ turnout + pop_dens + prop_pour_hos +
                 prop_owner + prop_recent +
                 prop_Ind_18_40 +
                 prop_Ind_40_64 +
                 prop_Ind_up_65,
             data = bv_sample_daw)
  res_lm_3 <- lm(taux_fn ~ turnout + pop_dens + prop_pour_hos +
                 prop_owner + prop_recent +
                 prop_Ind_18_40 +
                 prop_Ind_40_64 +
                 prop_Ind_up_65,
             data = bv_sample)
```

```
  res_lm <- lm(taux_fn ~ turnout + pop_dens + prop_pour_hos +
                 prop_owner + prop_recent +
                 prop_Ind_18_40 +
                 prop_Ind_40_64 +
                 prop_Ind_up_65 +
                 prop_unemploy + prop_immi +
                 prop_csp_1 + prop_csp_2 + prop_csp_3,
             data = bv_sample)
```

```
stargazer::stargazer(res_lm)
```

```
stargazer::stargazer(res_lm_1_big, res_lm_2_big, res_lm_3_big,
                     res_lm_1, res_lm_2, res_lm_3, res_lm)
```

6.3.1.1 Full Model

We also compute the MSE:

```
round(c(
  mean(residuals(res_lm_1_big) ^ 2),
  mean(residuals(res_lm_2_big) ^ 2),
  mean(residuals(res_lm_3_big) ^ 2),
  mean(residuals(res_lm_1) ^ 2),
  mean(residuals(res_lm_2) ^ 2),
  mean(residuals(res_lm_3) ^ 2),
  mean(residuals(res_lm) ^ 2)), 3)
```

```
## [1] 17.511 15.794 15.773 16.162 15.640 15.578 11.475
```

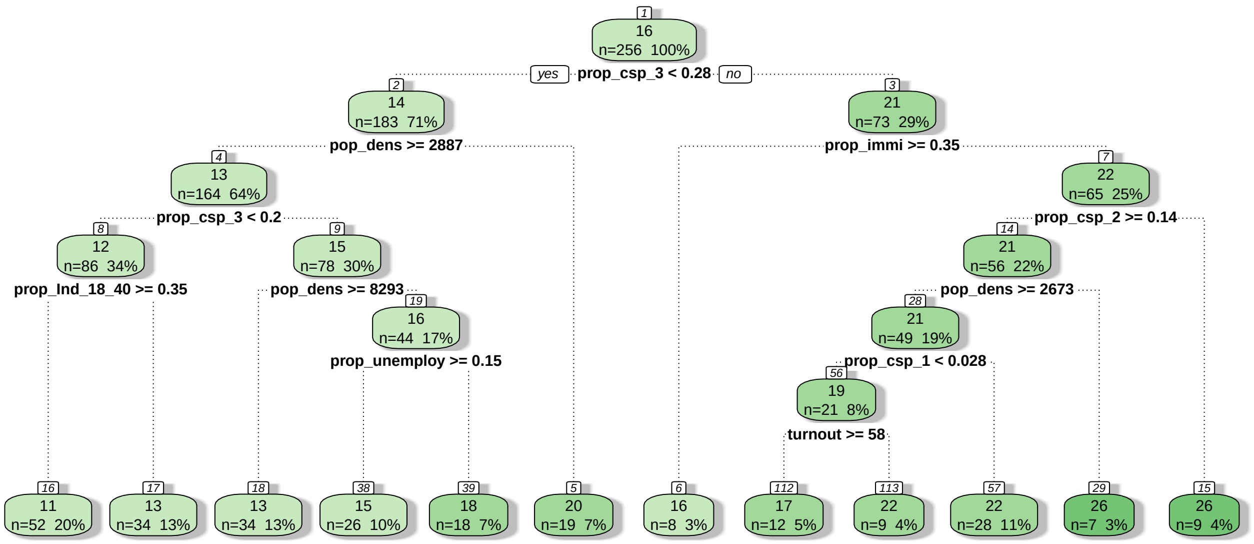
6.3.2 Regression tree

```
my_tree <- rpart(taux_fn ~ turnout + pop_dens + prop_pour_hos + prop_owner +
    prop_recent +
      prop_Ind_18_40 +
      prop_Ind_40_64 +
      prop_Ind_up_65 +
      prop_unemploy + prop_immi + prop_csp_1 +
    prop_csp_2 + prop_csp_3, data = bv_sample)
fancyRpartPlot(my_tree, sub = "")
```

Table 7: Results of the linear regression

	Dependent variable: Extreme Right Vote						
	Sources = Big cells			Sources = Small cells			Full model
	PIP	DAW	DAX	PIP	DAW	DAX	$DAX + DAC$
turnout	0.293***	0.267***	0.249***	0.248***	0.257***	0.267***	0.194***
	(0.072)	(0.058)	(0.058)	(0.065)	(0.066)	(0.065)	(0.063)
pop_dens	−0.0001	−0.0002*	−0.0002**	−0.0003***	−0.0004***	−0.0004***	−0.0003***
	(0.0001)	(0.0001)	(0.0001)	(0.0001)	(0.0001)	(0.0001)	(0.0001)
prop_pour_hos	0.450	−1.165	−0.856	−7.152	−8.461	−8.676	−23.477***
	(8.157)	(8.921)	(8.866)	(6.799)	(7.582)	(7.543)	(8.293)
prop_owner	4.655	3.136	0.688	−2.146	−5.446	−5.887	−3.898
	(3.796)	(5.106)	(5.090)	(3.311)	(3.828)	(3.790)	(3.380)
prop_recent	7.942***	9.193***	8.974***	4.082**	5.010**	5.352**	−2.503
	(2.363)	(2.734)	(2.707)	(1.987)	(2.261)	(2.261)	(2.216)
prop_Ind_18_40	−38.371***	−45.299***	−43.100***	−42.835***	−42.579***	−40.747***	−13.038*
	(7.950)	(9.066)	(9.182)	(6.804)	(7.395)	(7.225)	(7.425)
prop_Ind_40_64	−22.593	−41.592**	−30.479	−39.130**	−29.572*	−26.661	−15.637
	(17.958)	(20.682)	(20.607)	(15.121)	(16.615)	(16.301)	(14.640)
prop_Ind_up_65	−20.026*	−13.328	−14.481	−27.081**	−20.248	−17.904	−17.953
	(11.755)	(15.473)	(15.746)	(10.651)	(12.277)	(12.569)	(11.660)
prop_unemploy							−33.265***
							(9.415)
prop_immi							7.567
							(9.339)
prop_csp_1							49.223**
							(24.657)
prop_csp_2							−25.087***
							(5.960)
prop_csp_3							34.222***
							(4.648)
Constant	18.818*	28.023**	26.418**	35.439***	32.869***	30.730***	29.907***
	(10.253)	(11.612)	(11.658)	(9.024)	(9.684)	(9.368)	(9.347)
Observations	199	256	256	254	256	256	256
R^2	0.397	0.433	0.434	0.418	0.438	0.441	0.588
Adjusted R^2	0.371	0.414	0.415	0.399	0.420	0.422	0.566
MSE	17.511	15.794	15.773	16.162	15.640	15.578	11.475

Note: *p<0.1; **p<0.05; ***p<0.01



```r
pred_tree <- predict(my_tree)
```

MSE :

```r
mean(residuals(my_tree) ^ 2)
```

```
## [1] 8.456495
```

Plotting of the first and last leaf.

```r
par(mar= c(0,0,0,0), mfrow = c(1, 2))
plot(st_geometry(bv_sample))
plot(st_geometry(bv_sample[pred_tree == max(pred_tree), ]),
     add = T, col = "red")
plot(st_geometry(bv_sample))
plot(st_geometry(bv_sample[pred_tree == min(pred_tree), ]),
     add = T, col = "red")
```