

# Méthodes Quantitatives en Géographie

Hugo Périlleux

2024-10-18



# Contents

<b>Introduction</b>	<b>5</b>
<b>1 Régressions</b>	<b>7</b>
1.1 Préparation . . . . .	7
1.2 Analyse de régression . . . . .	9
1.3 Analyse géographique . . . . .	21
<b>2 Analyse en Composantes Principales</b>	<b>37</b>
2.1 Objectif: . . . . .	37
2.2 Données: . . . . .	37
2.3 Importation des données . . . . .	38
2.4 Recodage . . . . .	41
2.5 Première visualisation des données . . . . .	41
2.6 Réaliser l'ACP . . . . .	50
2.7 Visualiser les résultats . . . . .	55
2.8 Ajouter des variables supplémentaires . . . . .	68
2.9 Aller plus loin . . . . .	72
2.10 Ressources utilisées: . . . . .	73
<b>3 Classification</b>	<b>75</b>
3.1 Objectifs . . . . .	75
3.2 Préparation . . . . .	75
3.3 ACP . . . . .	77
3.4 Classification ascendante hiérarchique . . . . .	78
3.5 Visualisation des résultats . . . . .	81
3.6 Ressources utilisées . . . . .	87



# Introduction

Ce manuel est le matériel des travaux pratiques pour le cours GEOG-F-404  
Méthode quantitative en géographie à l'Université Libre de Bruxelles.

Il contient 3 chapitres:

1. Régression linéaires
2. Analyse en composantes principales
3. Classification



# Chapter 1

## Régressions

données utilisées:

- immoweb\_louer.csv
- census\_2011\_logements.xls
- cantons\_judiciaires\_bxl\_2018.gpkg

### 1.1 Préparation

Pour manipuler les données on utilisera les packages suivants:

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## vforcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr    1.3.1
## v purrr    1.0.2
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()   masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
library(readxl)
library(sf)

## Linking to GEOS 3.11.2, GDAL 3.8.2, PROJ 9.3.1; sf_use_s2() is TRUE
library(mapsf)
library(modelr)
```

On va utiliser les nouveaux packages suivants:

```
library(jtools)
library(huxtable)

## 
## Attachement du package : 'huxtable'

## L'objet suivant est masqué depuis 'package:dplyr':
##
##     add_rownames

## L'objet suivant est masqué depuis 'package:ggplot2':
##
##     theme_grey

library(ggstats)
library(performance)

## 
## Attachement du package : 'performance'

## Les objets suivants sont masqués depuis 'package:huxtable':
##
##     print_html, print_md

## Les objets suivants sont masqués depuis 'package:modelr':
##
##     mae, mse, rmse
```

Pour réaliser cet exemple, on utilisera les données issues d'un scraping de Immoweb des logements à louer pour Bruxelles. Les adresses ont été géocodées avec Phacochr.

```
loyers_data<-read_delim("data/immoweb_louer.csv", delim = ";")

## Rows: 17735 Columns: 9
## -- Column specification -----
## Delimiter: ;
## chr (4): type, PEB, nb_chambres, cd_secteur
## dbl (4): surface, loyer, x_31370, y_31370
## dttm (1): date_request
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

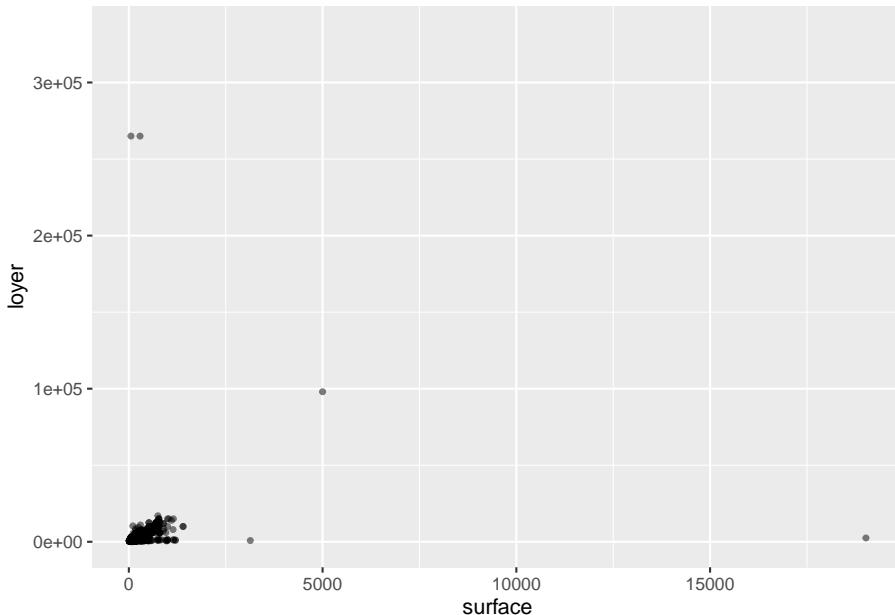
## 1.2 Analyse de régression

### 1.2.1 Visualiser et supprimer les outliers

On peut analyser la relation entre la variable loyer et la variable surface, on peut simplement réaliser un graphique avec un geom\_point:

```
loyers_data %>%
  ggplot( aes(surface, loyer)) +
  geom_point(alpha=0.5,cex=1)

## Warning: Removed 2066 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



Pour analyser la relation entre les deux variables, il peut être utile de supprimer les valeurs abérantes (outliers). Ceci peut se faire de façon simple en retirant 1% de chaque côté de la distribution en appliquant un filtre et en conservant les loyer par surface supérieur au quantile 0.01 et inférieur au quantile 0.99:

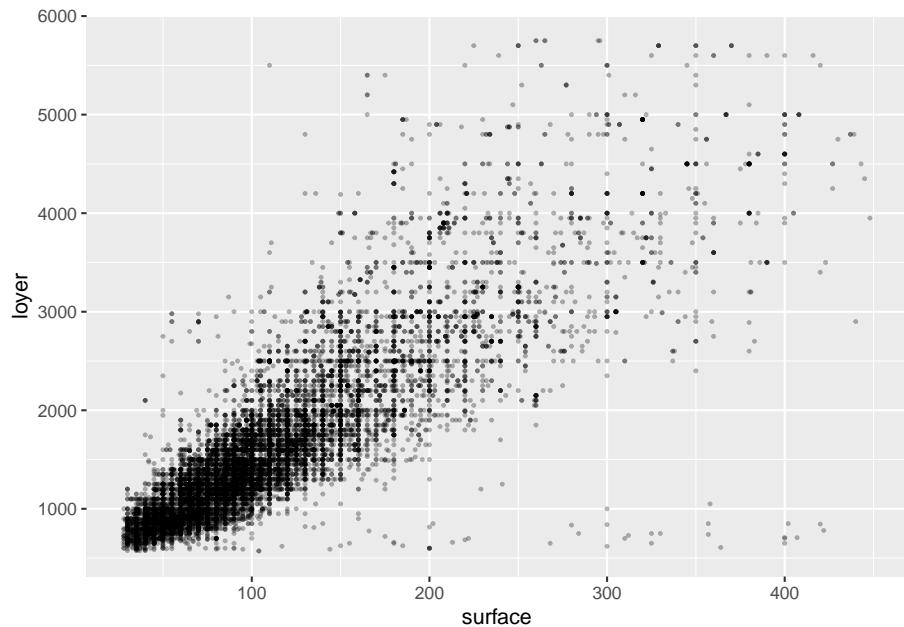
```
loyers_data<- loyers_data %>%
  filter(
    loyer > quantile(loyer, prob = 0.01, na.rm = TRUE),
    loyer < quantile(loyer, prob = 0.99, na.rm = TRUE),
    surface > quantile(surface, prob = 0.01, na.rm = TRUE),
    surface < quantile(surface, prob = 0.99, na.rm = TRUE))
```

Pour supprimer les outliers, il est possible également d'utiliser de filtre plus

“réfléchi” en analysant les données ou en utilisant des méthodes plus complexes : <https://delladata.fr/comment-detecter-les-outliers-avec-r/>

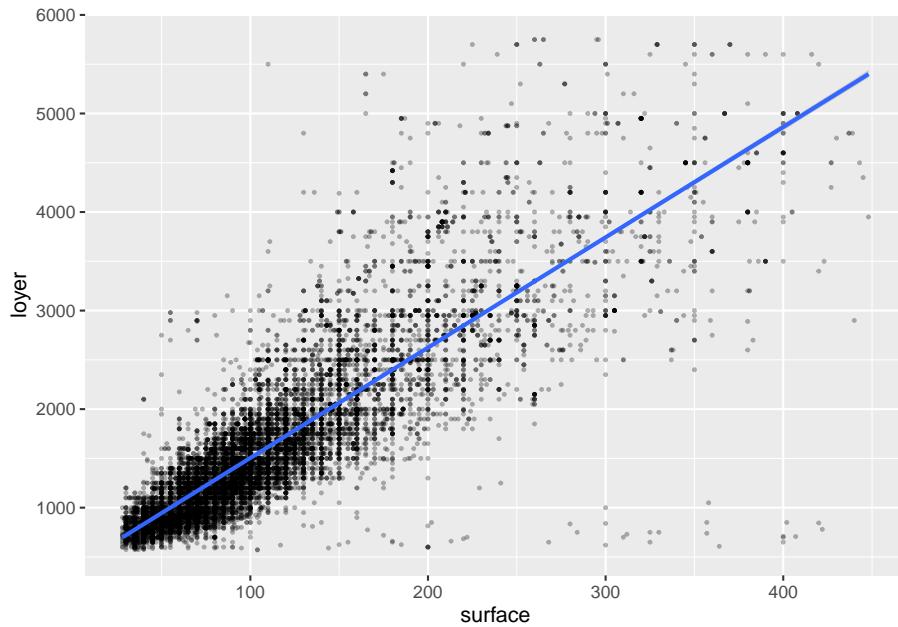
On peut alors visualiser les données et on observe une relation beaucoup plus claire:

```
loyers_data %>%
  ggplot( aes(surface, loyer)) +
  geom_point(alpha=0.3,cex=0.5)
```



On peut ajouter une droite de régression sur le graphique grâce à geom\_smooth:

```
loyers_data %>%
  ggplot( aes(surface, loyer)) +
  geom_point(alpha=0.3,cex=0.5) +
  geom_smooth(formula = y ~ x, method = "lm")
```



### 1.2.2 Régression simple

On peut réaliser une analyse de régression grâce à la fonction lm où  $Y = aX + b$  ce traduit par  $Y \sim X$ . Dans notre cas, on tente d'évaluer le loyer ( $Y$ ) en fonction de la surface ( $X$ ):

```
model1<-lm(loyer ~ surface, data=loyers_data)
```

Pour afficher un résumé des résultats on utilise la fonction summary sur l'objet créé par la fonction lm:

```
summary(model1)

##
## Call:
## lm(formula = loyer ~ surface, data = loyers_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -4328.9  -210.7   -50.8   147.6  3883.4 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 385.3549    7.0466  54.69 <2e-16 ***
## surface     11.1933    0.0557 200.97 <2e-16 ***
## ---
```

```
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 435.8 on 15222 degrees of freedom
## Multiple R-squared: 0.7263, Adjusted R-squared: 0.7263
## F-statistic: 4.039e+04 on 1 and 15222 DF, p-value: < 2.2e-16
```

Notez que l'objet model est un objet list qui contient une série de choses qu'il est possible d'aller chercher grâce au \$ : model1\$coefficients les coefficients model1\$residuals les résidus model1\$fitted.values les valeurs prédites

De même il est possible d'aller rechercher des éléments du résultat de summary appliqué sur le modèle: summary(model1)\$r.squared le  $R^2$

### 1.2.3 Régression multiple

On peut décider de rajouter des variables dans le modèle en modifiant la formule  $Y = X_1 + X_2 + X_3 + \dots$

```
model2<-lm(loyer ~ surface + type + PEB + nb_chambres, data=loyers_data)
summary(model2)
```

```
##
## Call:
## lm(formula = loyer ~ surface + type + PEB + nb_chambres, data = loyers_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -4185.1  -212.7   -38.6   154.0  2892.6 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 408.4328   27.3727  14.921 < 2e-16 ***
## surface     11.1487    0.1127  98.950 < 2e-16 ***
## typeHOUSE  21.0105   19.6294   1.070 0.284480    
## PEBB        102.9783   23.7831   4.330 1.50e-05 ***
## PEBC        11.3038   22.9773   0.492 0.622762    
## PEBD       -69.1262   22.7818  -3.034 0.002417 **  
## PEBE       -82.0685   23.5270  -3.488 0.000488 ***  
## PEBF       -118.1271   25.2917  -4.671 3.04e-06 ***  
## PEBG       -99.2732   24.5295  -4.047 5.22e-05 ***  
## nb_chambres1 -12.0105   19.0896  -0.629 0.529254    
## nb_chambres2    4.8550   19.6849   0.247 0.805194    
## nb_chambres3  156.4296   23.2023   6.742 1.64e-11 ***  
## nb_chambres4   79.7030   31.7783   2.508 0.012153 *  
## nb_chambres5+ -135.7740   42.2787  -3.211 0.001325 ** 
## ---                                                        
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```

## 
## Residual standard error: 410.4 on 11090 degrees of freedom
##   (4120 observations effacées parce que manquantes)
## Multiple R-squared:  0.7633, Adjusted R-squared:  0.763
## F-statistic:  2751 on 13 and 11090 DF,  p-value: < 2.2e-16

```

## 1.2.4 Régression polynomiale

### 1.2.4.1 Logarithme

Plutôt que de modéliser loyer en fonction de la surface, on pourrait avoir envie de modéliser le loyer selon le logarithme de la surface.

Alors on soit créer une nouvelle variable et réaliser la régression:

```

loyers_data$log_surface <- log(loyers_data$surface)

model3<- lm(loyer ~ log_surface + type + PEB + nb_chambres, data=loyers_data)
summary(model3)

```

```

## 
## Call:
## lm(formula = loyer ~ log_surface + type + PEB + nb_chambres,
##      data = loyers_data)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3415.2  -263.3   -50.6   200.8  2883.8
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -4067.02    67.47  -60.282 < 2e-16 ***
## log_surface  1320.08   16.31   80.923 < 2e-16 ***
## typeHOUSE    98.95    21.29    4.649  3.38e-06 ***
## PEBB         92.62    25.88    3.578  0.000347 ***
## PEBC        11.21    25.01    0.448  0.653853  
## PEBD        -56.46    24.79   -2.277  0.022783 *  
## PEBE        -67.66    25.60   -2.643  0.008226 ** 
## PEBF       -105.00    27.52   -3.816  0.000137 *** 
## PEBG        -72.71    26.68   -2.725  0.006443 ** 
## nb_chambres1 -281.90   21.42  -13.161 < 2e-16 ***
## nb_chambres2 -458.09   24.36  -18.804 < 2e-16 ***
## nb_chambres3 -249.66   29.24  -8.538 < 2e-16 ***
## nb_chambres4 -137.41   37.64  -3.650  0.000263 *** 
## nb_chambres5+ 130.42   46.17   2.825  0.004743 ** 
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```

## 
## Residual standard error: 446.6 on 11090 degrees of freedom
##   (4120 observations effacées parce que manquantes)
## Multiple R-squared:  0.7198, Adjusted R-squared:  0.7194
## F-statistic:  2191 on 13 and 11090 DF, p-value: < 2.2e-16

soit indiquer le logarithme directement dans la formule du lm

model3<- lm(loyer ~ log(surface) + type + PEB + nb_chambres, data=loyers_data)
summary(model3)

## 
## Call:
## lm(formula = loyer ~ log(surface) + type + PEB + nb_chambres,
##      data = loyers_data)
## 
## Residuals:
##    Min     1Q Median     3Q    Max 
## -3415.2 -263.3 -50.6  200.8 2883.8 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -4067.02    67.47 -60.282 < 2e-16 ***
## log(surface) 1320.08   16.31  80.923 < 2e-16 ***
## typeHOUSE    98.95    21.29   4.649 3.38e-06 ***
## PEBC         92.62    25.88   3.578 0.000347 *** 
## PEBC          11.21    25.01   0.448 0.653853  
## PEBC         -56.46    24.79  -2.277 0.022783 *  
## PEBC         -67.66    25.60  -2.643 0.008226 ** 
## PEBC        -105.00    27.52  -3.816 0.000137 *** 
## PEBC         -72.71    26.68  -2.725 0.006443 ** 
## nb_chambres1 -281.90   21.42 -13.161 < 2e-16 ***
## nb_chambres2 -458.09   24.36 -18.804 < 2e-16 ***
## nb_chambres3 -249.66   29.24  -8.538 < 2e-16 ***
## nb_chambres4 -137.41   37.64  -3.650 0.000263 *** 
## nb_chambres5+ 130.42   46.17   2.825 0.004743 ** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

## Residual standard error: 446.6 on 11090 degrees of freedom
##   (4120 observations effacées parce que manquantes)
## Multiple R-squared:  0.7198, Adjusted R-squared:  0.7194
## F-statistic:  2191 on 13 and 11090 DF, p-value: < 2.2e-16

```

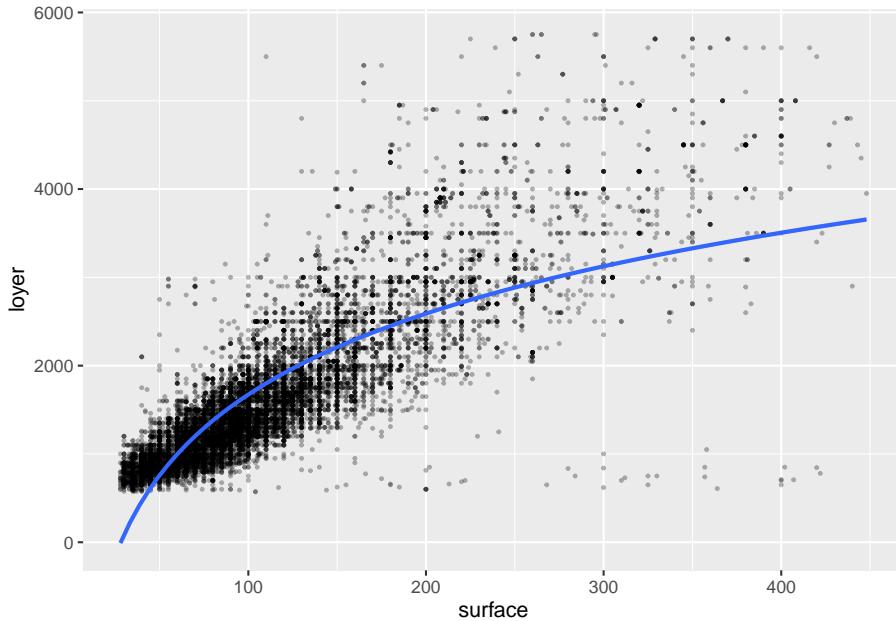
On peut faire un graphique de la façon suivante:

```

loyers_data %>%
  ggplot( aes(surface, loyer)) +

```

```
geom_point(alpha=0.3,cex=0.5)+  
geom_smooth(formula = y ~ log(x), method = "lm")
```



#### 1.2.4.2 Division

On peut vouloir modéliser le loyer par surface en fonction de la surface:

```
loyers_data <- loyers_data %>%  
  mutate(loyer_surface= loyer/surface,  
        inv_surface= 1/surface)  
  
model4<- lm(loyer_surface ~ inv_surface + type + PEB + nb_chambres, data=loyers_data)  
summary(model4)  
  
##  
## Call:  
## lm(formula = loyer_surface ~ inv_surface + type + PEB + nb_chambres,  
##      data = loyers_data)  
##  
## Residuals:  
##     Min      1Q  Median      3Q     Max  
## -11.2378 -2.1955 -0.4891  1.6736 31.3294  
##  
## Coefficients:  
##             Estimate Std. Error t value Pr(>|t|)
```

```

## (Intercept) 8.3490 0.3133 26.647 < 2e-16 ***
## inv_surface 513.5564 9.6383 53.283 < 2e-16 ***
## typeHOUSE 0.3212 0.1522 2.111 0.0348 *
## PEBB 0.8469 0.1857 4.561 5.14e-06 ***
## PEBC -0.0173 0.1793 -0.097 0.9231
## PEBD -0.7036 0.1776 -3.961 7.51e-05 ***
## PEBE -0.9432 0.1835 -5.139 2.80e-07 ***
## PEBF -1.1262 0.1973 -5.708 1.17e-08 ***
## PEBG -1.3072 0.1915 -6.827 9.11e-12 ***
## nb_chambres1 0.9402 0.1631 5.765 8.38e-09 ***
## nb_chambres2 1.8815 0.1948 9.659 < 2e-16 ***
## nb_chambres3 3.2194 0.2225 14.472 < 2e-16 ***
## nb_chambres4 3.0069 0.2709 11.099 < 2e-16 ***
## nb_chambres5+ 2.3830 0.3180 7.494 7.17e-14 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.204 on 11090 degrees of freedom
## (4120 observations effacées parce que manquantes)
## Multiple R-squared: 0.3567, Adjusted R-squared: 0.356
## F-statistic: 473.1 on 13 and 11090 DF, p-value: < 2.2e-16

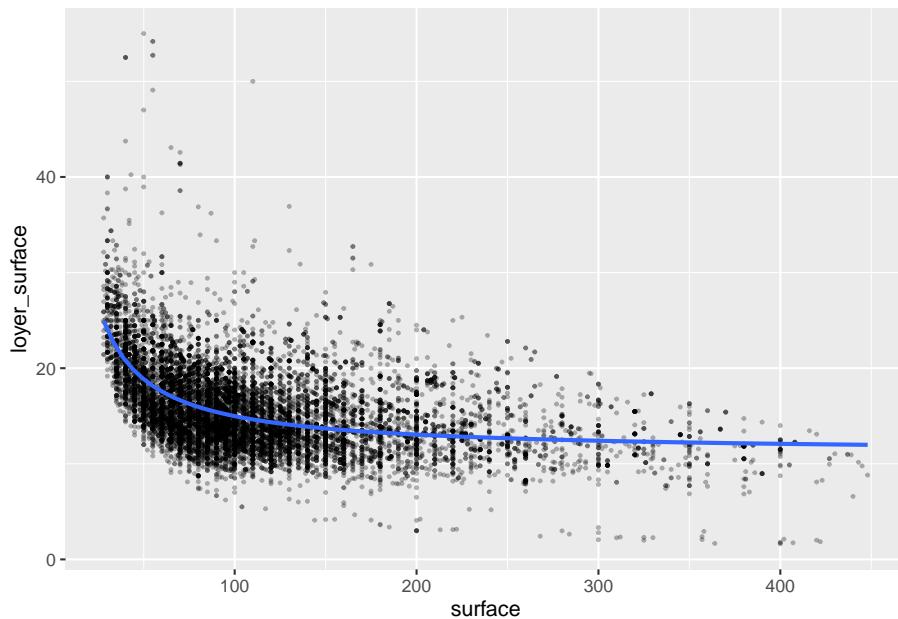
```

Le graphique avec le I dans le formule:

```

loyers_data %>%
  ggplot( aes(surface, loyer_surface)) +
  geom_point(alpha=0.3,cex=0.5) +
  geom_smooth(formula = y ~ I(1/x), method = "lm")

```



#### 1.2.4.3 Exposants

On pourrait directement placer ces changements dans la formule mais alors il faut utiliser la fonction `I()` qui permet de réaliser ces opérations. La fonction `I` ("come si" / "as is") permet d'indiquer qu'il s'agit d'une formule et non pas une opération sur un vecteur et le ~ "est dépendant de".

```
model5<- lm(loyer~ surface + I(surface^2) + type + PEB + nb_chambres, data=loyers_data)
summary(model5)
```

```
##
## Call:
## lm(formula = loyer ~ surface + I(surface^2) + type + PEB + nb_chambres,
##      data = loyers_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4072.8  -213.4   -37.2   159.6  2881.1
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.562e+02  2.943e+01 12.105 < 2e-16 ***
## surface     1.259e+01  3.200e-01 39.332 < 2e-16 ***
## I(surface^2) -3.933e-03 8.187e-04 -4.804 1.58e-06 ***
## typeHOUSE   2.305e+01  1.961e+01  1.175 0.239960
## PEBB        9.977e+01  2.377e+01  4.197 2.72e-05 ***
##
```

```

## PEBC      6.240e+00  2.298e+01  0.272  0.785963
## PEBD     -7.414e+01  2.278e+01 -3.254  0.001141 ** 
## PEBE     -8.608e+01  2.352e+01 -3.660  0.000253 *** 
## PEBF    -1.216e+02  2.528e+01 -4.811  1.52e-06 *** 
## PEBG    -1.019e+02  2.451e+01 -4.156  3.27e-05 *** 
## nb_chambres1 -3.119e+01  1.948e+01 -1.601  0.109455
## nb_chambres2 -4.329e+01  2.207e+01 -1.961  0.049890 *  
## nb_chambres3  9.303e+01  2.667e+01  3.488  0.000489 *** 
## nb_chambres4  1.865e+01  3.420e+01  0.545  0.585428
## nb_chambres5+ -1.391e+02  4.224e+01 -3.293  0.000993 *** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 410 on 11089 degrees of freedom
##   (4120 observations effacées parce que manquantes)
## Multiple R-squared:  0.7638, Adjusted R-squared:  0.7635
## F-statistic:  2561 on 14 and 11089 DF,  p-value: < 2.2e-16

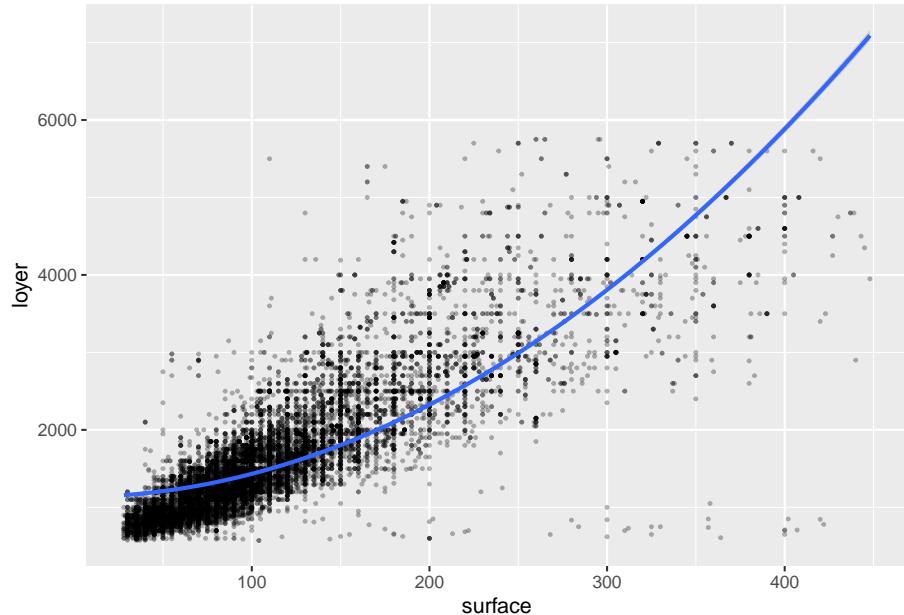
```

le graphique, de nouveau il faut bien utiliser I dans la formule:

```

loyers_data %>%
  ggplot( aes(surface, loyer)) +
  geom_point(alpha=0.3,cex=0.5) +
  geom_smooth(formula = y ~ I(x+x^2), method = "lm")

```



### 1.2.5 Visualisation

Il existe beaucoup de package pour faciliter la visualisation des résultats. # Par exemple pour visualiser rapidement le résultat de plusieurs modèles jtools permet de réaliser un tableau synthétique utile:

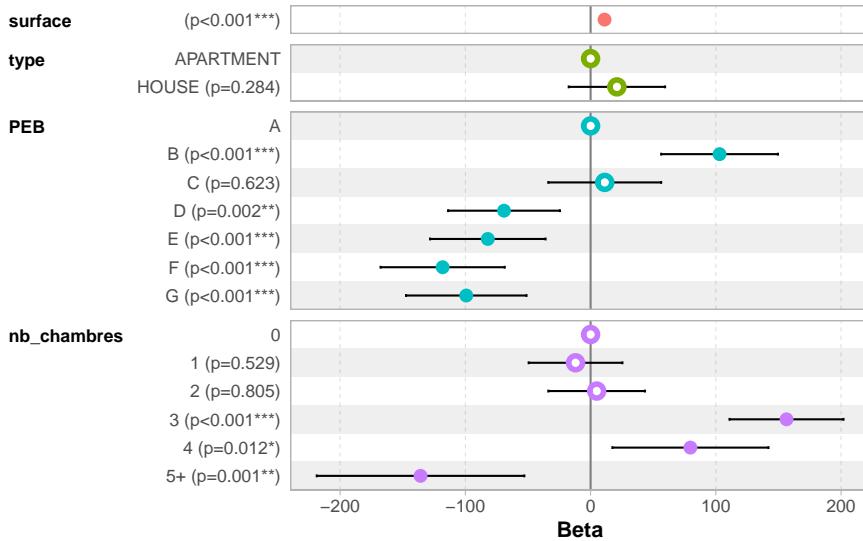
```
library(jtools)
export_summs(model1, model2, model3, model4, model5)
```

On peut également exporter ce résultat en format pdf de la façon suivante:

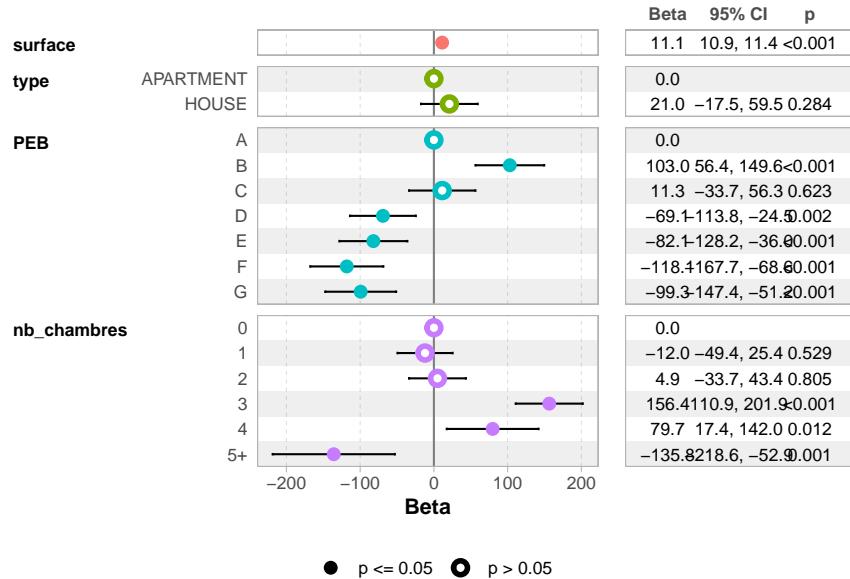
```
# export_summs(model1, model2, model3, model4, to.file = "html", file.name = "TP08/tableau_regression.html")
```

On peut visualiser un forest plot de cette façon grâce au package ggstats:

```
library(ggstats)
ggcoef_model(model2)
```



```
ggcoef_table(model2)
```



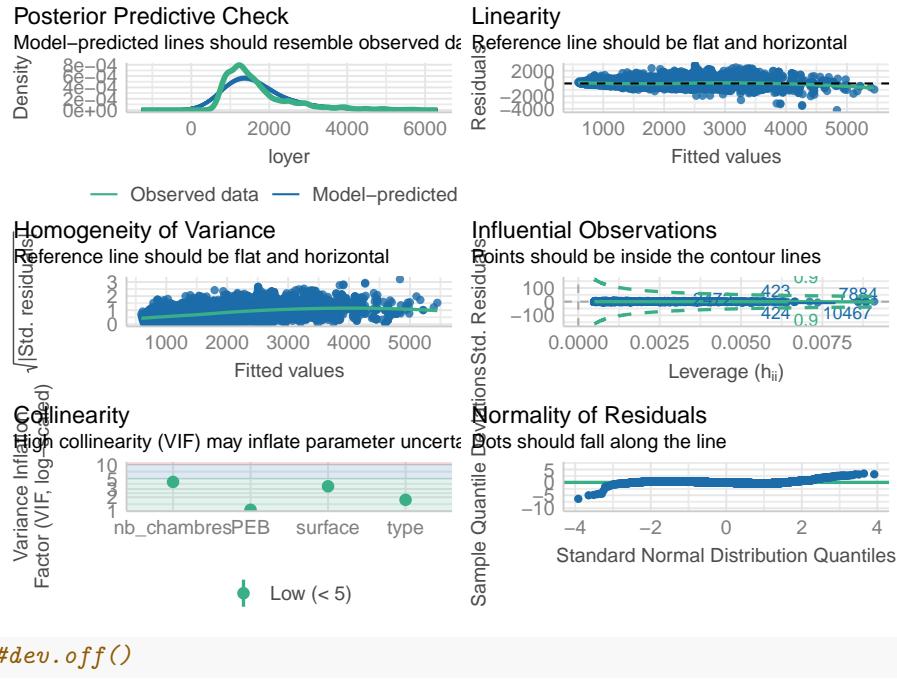
voir ici pour plus de détails: < [https://cran.r-project.org/web/packages/ggstats/vignettes/ggcoef\\_model.htm](https://cran.r-project.org/web/packages/ggstats/vignettes/ggcoef_model.htm) >

### 1.2.6 Réaliser les tests des hypothèses

La réalisation de modèle de régression s'appuie sur une série d'hypothèses qu'il s'agit de vérifier dont: - la linéarité des résidus, - la constance de la variance des résidus, - la faible influence d'outliers, - la non colinéarité entre les variables explicatives, - la normalité des résidus.

< <https://easystats.github.io/performance/index.html> >

```
library(performance)
#png("TP08/check_model.png", height= 1000, width=1000)
check_model(model2)
```



## 1.3 Analyse géographique

### 1.3.1 Analyse géographique simple

On peut réaliser une carte du loyer moyen par secteur statistique. On calculer par secteur statistique le nombre d'annonces et le loyer moyen:

```
loyer_moyen<-loyers_data%>%
  group_by(cd_sector) %>%
  summarise(
    n=n(),
    loy_moyen= mean(loyer, na.rm=T)
  )
```

On charge les secteurs statistiques

```
secteurs_stats<- st_read ("data/sh_statbel_statistical_sectors_31370_20230101.gpkg")

## Reading layer `secteurs_stats2023` from data source
##   `C:\Users\hugop\Nextcloud\git\book\data\sh_statbel_statistical_sectors_31370_20230101.gpkg'
##   using driver `GPKG'
## Simple feature collection with 19795 features and 31 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
```

```
## Bounding box: xmin: 21991.63 ymin: 21162.16 xmax: 295167.1 ymax: 244027.2
## Projected CRS: BD72 / Belgian Lambert 72
```

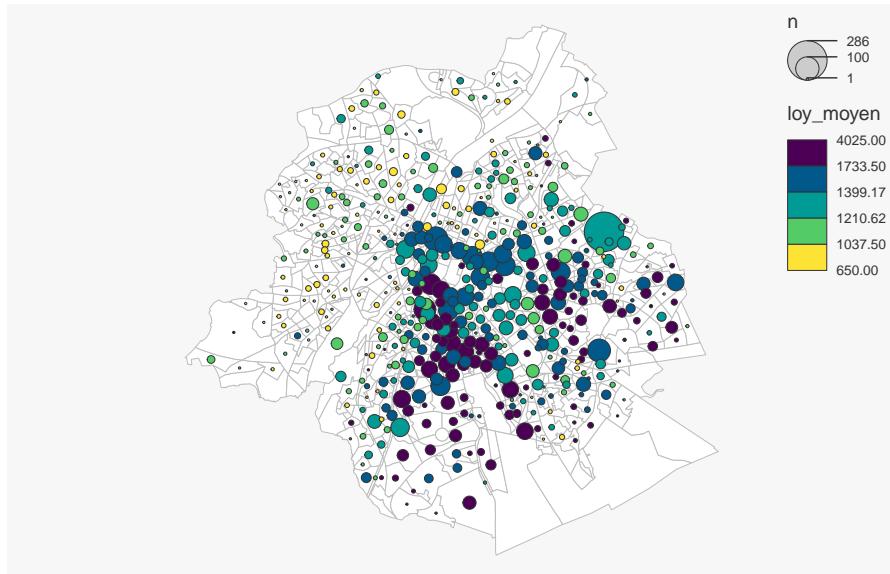
On fait une jointure entre le résultat du calcul précédent et les secteurs statistiques

```
loyer_moyen_sec<-secteurs_stats %>%
  left_join(loyer_moyen, by=c("cd_sector")) %>%
  filter(tx_rgn_descr_fr=="Région de Bruxelles-Capitale")
```

On réalise la carte

```
mf_map(x = loyer_moyen_sec,col = "white", border = "grey")
mf_map(loyer_moyen_sec,
       var= c("n", "loy_moyen"),
       type="prop_choro",
       pal= "Viridis",
       inches=0.14,
       nbreaks=5,
       add=T)

## 196 'NA' values are not plotted on the map.
```



### 1.3.2 Cartographier les résidus

Les résidus de la régression se trouvent dans l'objet model produit par la fonction lm. Néanmoins il s'agit d'un objet list où le valeur manquantes ont été

supprimée. Le package `modelr` permet de faire facilement la jointure entre les données de base et les résidus et calculer une moyenne des résidus par secteurs statistiques:

```
library(modelr)

resid<-loyers_data %>%
  add_residuals( model2) %>%
  group_by(cd_sector) %>%
  summarise(moyenne_residus= mean(resid, na.rm=T),
            n=n())
```

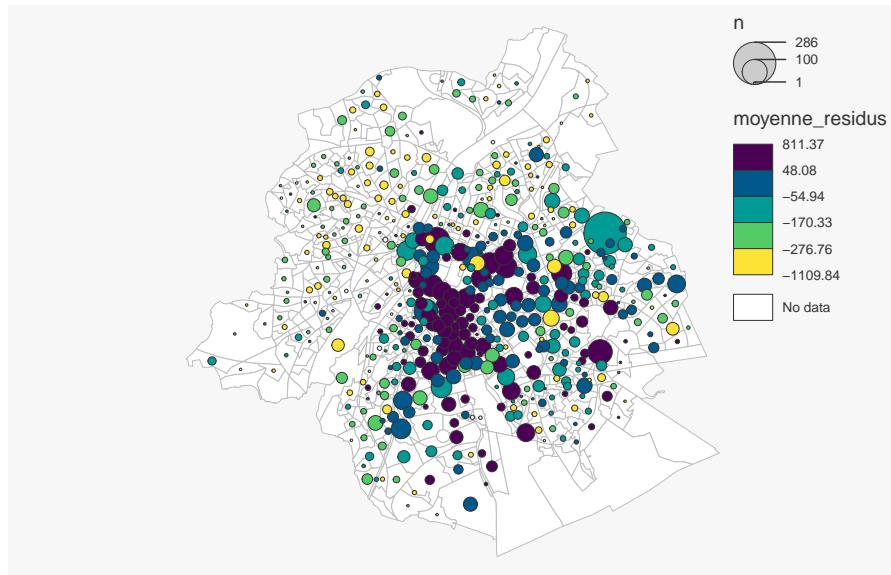
On joint les résidus au fichier des secteurs statistiques

```
resid_sec<-secteurs_stats%>%
  left_join(resid, by=c("cd_sector")) %>%
  filter(tx_rgn_descr_fr=="Région de Bruxelles-Capitale")
```

On fait une carte des résidus moyen par secteurs stats

```
mf_map(x = resid_sec,col = "white", border = "grey")
mf_map(resid_sec,
       var= c("n", "moyenne_residus"),
       type="prop_choro",
       pal= "Viridis",
       inches=0.15,
       nbreaks=5,
       add=T)
```

## 196 'NA' values are not plotted on the map.



On peut alors observer là où le loyer est en moyenne plus élevé ou moins élevé que prédit par le modèle.

### 1.3.3 Ajouter une variable géographique

On peut ajouter une variable géographique comme l'indice de difficulté par secteur statistique de 2010 pour Bruxelles qui une bonne approximation de la division sociale de l'espace bruxellois:

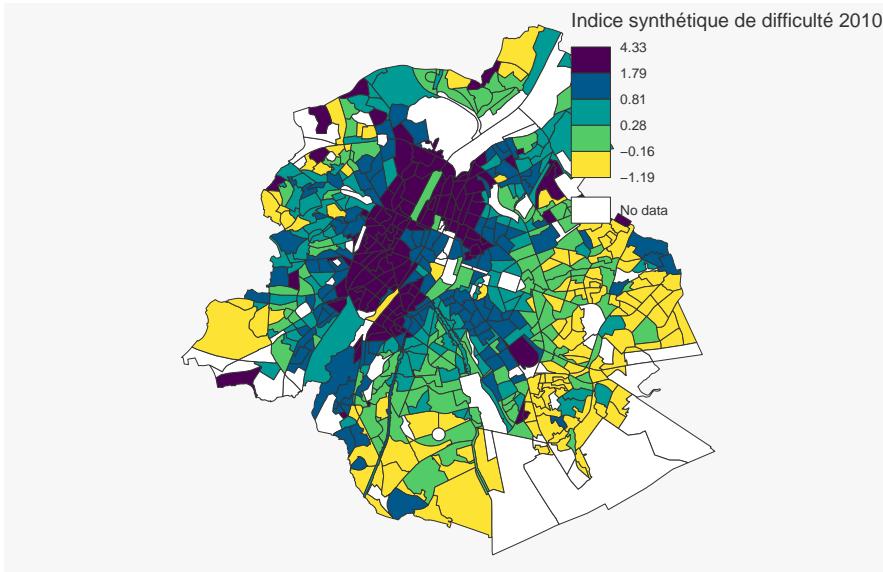
```
indice <- read_delim("data/indice_synthetique.csv", delim= ";")

## #> Rows: 7752 Columns: 2
## #> -- Column specification -----
## #> Delimiter: ";"
## #> chr (1): Secteur statistique
## #> dbl (1): Indice synthétique de difficulté 2010
## #>
## #> i Use `spec()` to retrieve the full column specification for this data.
## #> i Specify the column types or set `show_col_types = FALSE` to quiet this message.

indice_sec<-secteurs_stats%>%
  left_join(indice, by=c("cd_sector"="Secteur statistique")) %>%
  filter(tx_rgn_descr_fr=="Région de Bruxelles-Capitale")

mf_map(indice_sec,
       var="Indice synthétique de difficulté 2010",
```

```
type="choro",
pal= "Viridis",
nbreaks=5)
```



On peut alors ajouter cet indice dans un troisième modèle. Pour réaliser ceci on doit d'abord joindre les données

```
loyers_data<-loyers_data %>%
  left_join(indice, by=c("cd_sector"="Secteur statistique"))

model6<-lm(loyer ~ surface + PEB+type+nb_chambres +`Indice synthétique de difficulté 2010`, data=
  export_summs(model1, model2, model6)
```

Notez que ici le modèle semble se détériorer puisque le R2 diminue. Ceci peut s'expliquer par le fait que le modèle est réalisé sur un nombre plus petit de variables.

#### 1.3.4 Analyse de corrélation spatiale

On peut également réaliser des régressions sur des entités spatiales. Ici on prend les données expulsions et on va analyser la correlation avec la division sociale de l'espace et les cantons pour mesurer un possible “effet juge”.

On importe les données:

```

expulsions <- read_delim("data/expulsions_2018_secteursstat.csv", delim= ";")

## Rows: 698 Columns: 7
## -- Column specification -----
## Delimiter: ;"
## chr (1): ID_SS_bis
## dbl (6): ID_SS, exp_ais, exp_individu, exp_public, exp_société, exp_total
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

indice <- read_delim("data/indice_synthetique.csv", delim= ";")

## Rows: 7752 Columns: 2
## -- Column specification -----
## Delimiter: ;"
## chr (1): Secteur statistique
## dbl (1): Indice synthétique de difficulté 2010
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

logements <- read_excel("data/census_2011_logements.xls")
secteurs_stats<- st_read ("data/sh_statbel_statistical_sectors_31370_20230101.gpkg")

## Reading layer `secteurs_stats2023` from data source
##   `C:\Users\hugop\Nextcloud\git\book\data\sh_statbel_statistical_sectors_31370_20230101.gpkg'
##   using driver `GPKG'
## Simple feature collection with 19795 features and 31 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:  xmin: 21991.63 ymin: 21162.16 xmax: 295167.1 ymax: 244027.2
## Projected CRS: BD72 / Belgian Lambert 72
cantons<- st_read ("data/cantons_judiciaires_bxl_2018.gpkg") %>%
  st_zm()

## Reading layer `cantons_bxl_2018` from data source
##   `C:\Users\hugop\Nextcloud\git\book\data\cantons_judiciaires_bxl_2018.gpkg'
##   using driver `GPKG'
## Simple feature collection with 20 features and 2 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XYZ
## Bounding box:  xmin: 141192.7 ymin: 161464.7 xmax: 158003.9 ymax: 178175.9
## z_range:        zmin: 0 zmax: 0
## Projected CRS: BD72 / Belgian Lambert 72

```

On réalise la jointure spatiale entre les canton et les centroïdes des secteurs

statistiques:

```
canton_secteurs<-cantons %>%
  st_join(st_point_on_surface(secteurs_stats), join= st_intersects) %>%
  as.data.frame() %>%
  select(CANTON, cd_sector)

## Warning: st_point_on_surface assumes attributes are constant over geometries
```

#### 1.3.4.1 Analyse visuelle

On peut réaliser une cartographie des trois jeux de données

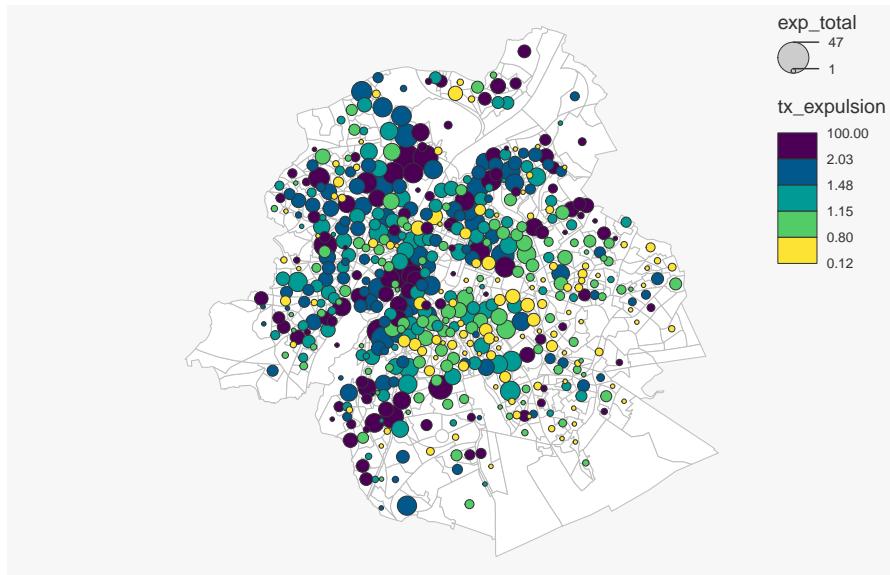
```
secteurs_stats_epulsions<-secteurs_stats %>%
  left_join(expulsions, by=c("cd_sector"="ID_SS_bis")) %>%
  left_join(logements, by= c("cd_sector"="Secteur statistique")) %>%
  filter(tx_rgn_descr_fr=="Région de Bruxelles-Capitale") %>%
  mutate(tx_expulsion=100*exp_total /`Logements loués`)

mf_map(x = secteurs_stats_epulsions,col = "white", border = "grey")
mf_map(secteurs_stats_epulsions,
       var=c("exp_total", "tx_expulsion"),
       type="prop_choro",
       pal= "Viridis",
       inches=0.11,
       nbreaks=5,
       add=T)
```

##### 1.3.4.1.1 Les expulsions

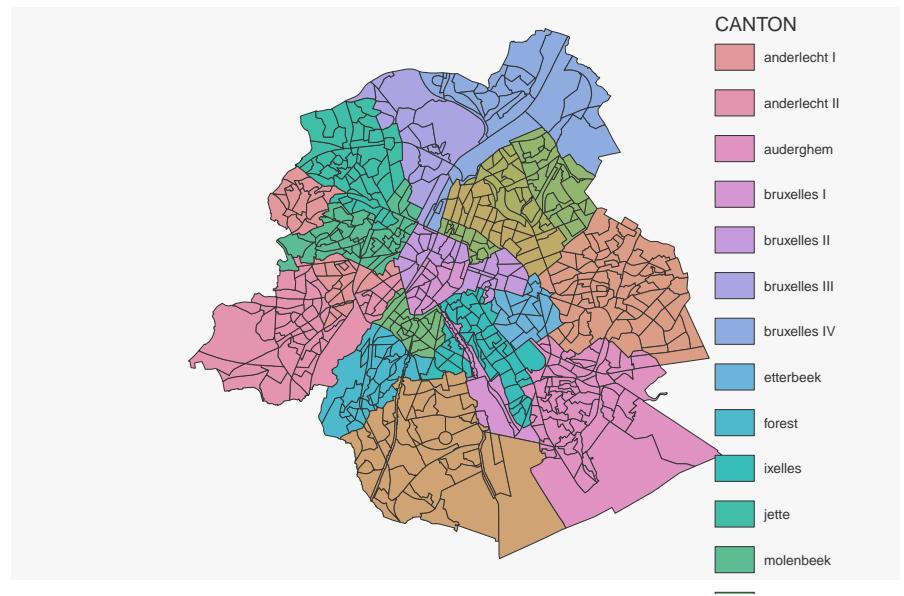
```
## 26 'NA' values are not plotted on the map.

## 140 '0' values are not plotted on the map.
```



```
secteurs_stats_cantons<-secteurs_stats %>%
  left_join(canton_secteurs, by="cd_sector") %>%
  filter(tx_rgn_descr_fr=="Région de Bruxelles-Capitale")

mf_map(secteurs_stats_cantons,
       var= "CANTON",
       type="typo")
```



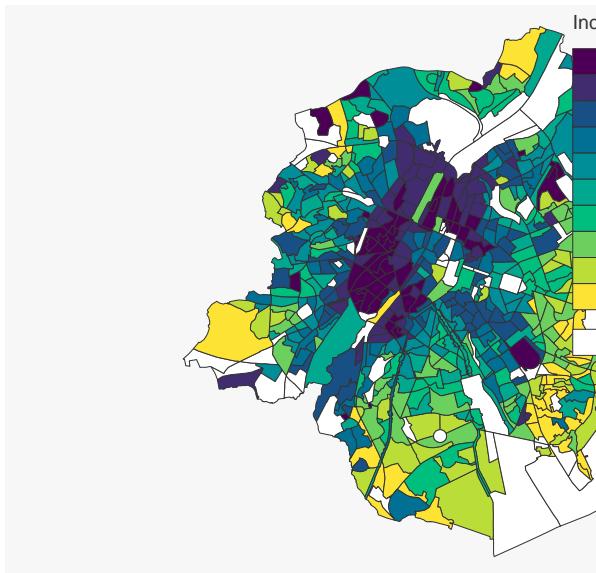
#### 1.3.4.1.2 Les cantons:

```

secteurs_stats_indice<-secteurs_stats %>%
  left_join(indice, by= c("cd_sector"="Secteur statistique")) %>%
  filter(tx_rgn_descr_fr=="Région de Bruxelles-Capitale")

mf_map(secteurs_stats_indice,
       var= "Indice synthétique de difficulté 2010",
       type="choro",
       pal= "Viridis")

```



#### 1.3.4.1.3 L'indice synthétique de difficulté:

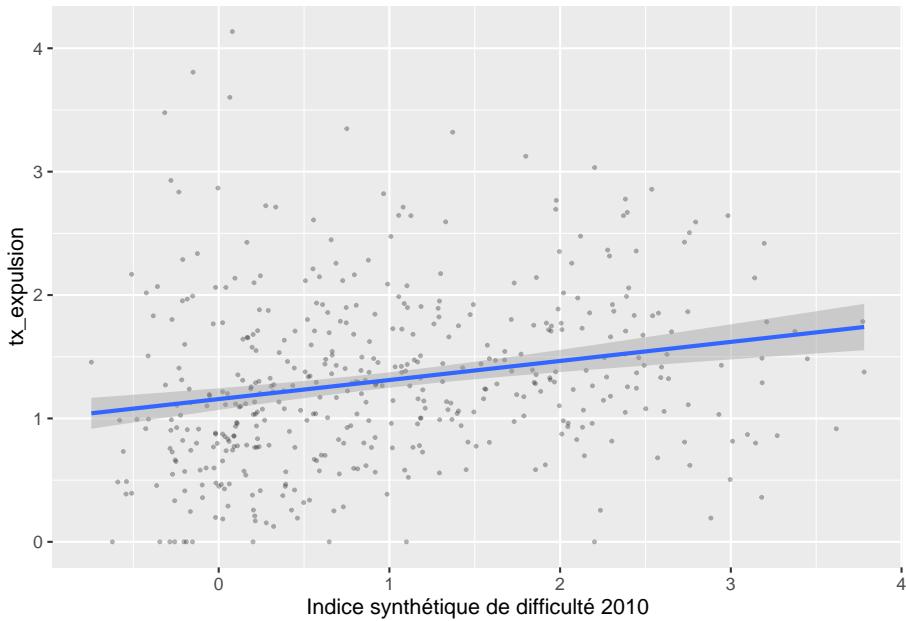
#### 1.3.4.2 Régression

On réalise les jointures, on calcul un taux d'expulsions et on ne garde que les secteurs statistiques bruxellois qui ont plus de 200 logements loués

```
secteurs_stats_expulsions<-secteurs_stats %>%
  left_join(expulsions, by=c("cd_sector"="ID_SS_bis")) %>%
  left_join(logements, by= c("cd_sector"="Secteur statistique")) %>%
  left_join(indice, by= c("cd_sector"="Secteur statistique")) %>%
  left_join(canton_secteurs,by= c("cd_sector")) %>%
  mutate(tx_expulsion=100*exp_total /`Logements loués`) %>%
  filter(tx_rgn_descr_fr=="Région de Bruxelles-Capitale") %>%
  filter (`Logements loués`>200)
```

On peut analyser le lien entre indice synthétique et le taux d'expulsion:

```
secteurs_stats_expulsions%>%
  ggplot( aes(`Indice synthétique de difficulté 2010`, tx_expulsion)) +
  geom_point(alpha=0.3,cex=0.5)+
  geom_smooth(formula = y ~ x, method = "lm")
```



Et réaliser une régression

```
model1<-lm(tx_expulsion~ `Indice synthétique de difficulté 2010`,data= secteurs_stats_expulsions
summary(model1)
```

```
##
## Call:
## lm(formula = tx_expulsion ~ `Indice synthétique de difficulté 2010`,
##     data = secteurs_stats_expulsions)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.49704 -0.43992 -0.08241  0.37774  2.96598
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  1.15698   0.04469 25.887 < 2e-16
## `Indice synthétique de difficulté 2010` 0.15447   0.03208  4.815 2.02e-06
##
## (Intercept)                 ***
## `Indice synthétique de difficulté 2010` ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6788 on 450 degrees of freedom
## Multiple R-squared:  0.04899,    Adjusted R-squared:  0.04688
```

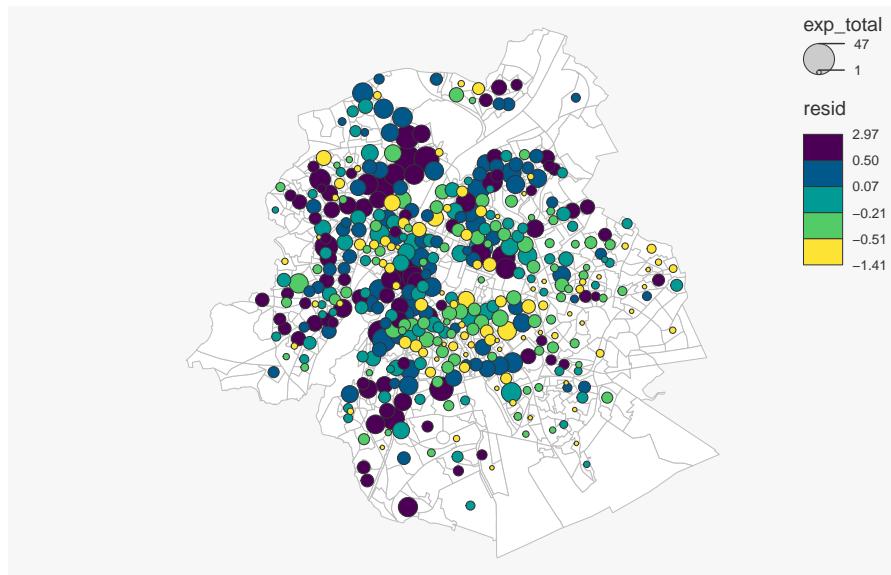
```
## F-statistic: 23.18 on 1 and 450 DF, p-value: 2.018e-06
```

On peut à nouveau cartographier les résidus

```
secteurs_stats_expulsions1<-secteurs_stats_expulsions %>%
  add_residuals( model1)
```

```
mf_map(x = filter(secteurs_stats, tx_rgn_descr_fr=="Région de Bruxelles-Capitale"),
       col = "white", border = "grey")
mf_map(secteurs_stats_expulsions1,
       var=c("exp_total", "resid"),
       type="prop_choro",
       pal= "Viridis",
       inches=0.11,
       nbreaks=5,
       add=T)
```

```
## 11 '0' values are not plotted on the map.
```



On peut pondérer la régression par le nombre de logements loués. Dans le cas où on a des entités de tailles très différentes cela peut avoir du sens

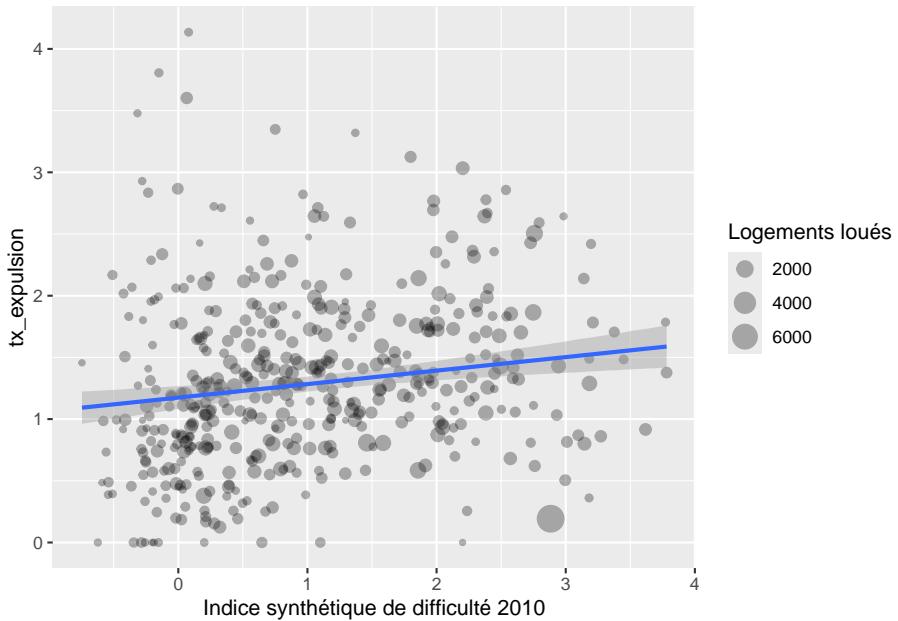
```
model2<-lm(tx_expulsion~ `Indice synthétique de difficulté 2010`, weights = `Logements
summary(model2)
```

```
##
## Call:
```

```

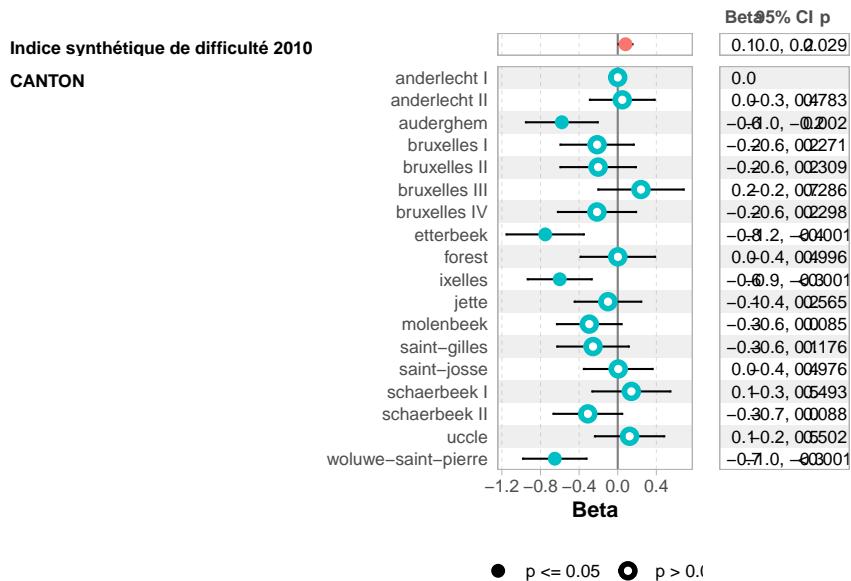
## lm(formula = tx_expulsion ~ `Indice synthétique de difficulté 2010`,
##     data = secteurs_stats_expulsions, weights = `Logements loués`)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -110.573  -9.817  -1.243   9.429   63.777
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  1.17420   0.04720 24.879 < 2e-16
## `Indice synthétique de difficulté 2010` 0.10944   0.03095  3.536 0.000448
## 
## (Intercept)                 ***
## `Indice synthétique de difficulté 2010` ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.14 on 450 degrees of freedom
## Multiple R-squared:  0.02704,    Adjusted R-squared:  0.02488
## F-statistic:  12.5 on 1 and 450 DF,  p-value: 0.000448
secteurs_stats_expulsions %>%
  ggplot( aes(x= `Indice synthétique de difficulté 2010`,y= tx_expulsion)) +
  geom_point(alpha=0.3, aes( size= `Logements loués` )) +
  geom_smooth(formula = y ~ x, method = "lm", mapping = aes(weight = `Logements loués`))

```



On peut ajouter la variable canton pour mesurer “l’effet juge”

```
model3<-lm(tx_expulsion~`Indice synthétique de difficulté 2010`+CANTON,data= secteur)
library(ggstats)
ggcoef_table(model3)
```



	Model 1	Model 2	Model 3	Model 4	Model 5
(Intercept)	385.35 *** (7.05)	408.43 *** (27.37)	-4067.02 *** (67.47)	8.35 *** (0.31)	356.21 *** (29.43)
surface	11.19 *** (0.06)	11.15 *** (0.11)			12.59 *** (0.32)
typeHOUSE		21.01 (19.63)	98.95 *** (21.29)	0.32 * (0.15)	23.05 (19.61)
PEBB		102.98 *** (23.78)	92.62 *** (25.88)	0.85 *** (0.19)	99.77 *** (23.77)
PEBC		11.30 (22.98)	11.21 (25.01)	-0.02 (0.18)	6.24 (22.98)
PEBD		-69.13 ** (22.78)	-56.46 * (24.79)	-0.70 *** (0.18)	-74.14 ** (22.78)
PEBE		-82.07 *** (23.53)	-67.66 ** (25.60)	-0.94 *** (0.18)	-86.08 *** (23.52)
PEBF		-118.13 *** (25.29)	-105.00 *** (27.52)	-1.13 *** (0.20)	-121.61 *** (25.28)
PEBG		-99.27 *** (24.53)	-72.71 ** (26.68)	-1.31 *** (0.19)	-101.86 *** (24.51)
nb_chambres1		-12.01 (19.09)	-281.90 *** (21.42)	0.94 *** (0.16)	-31.19 (19.48)
nb_chambres2		4.86 (19.68)	-458.09 *** (24.36)	1.88 *** (0.19)	-43.29 * (22.07)
nb_chambres3		156.43 *** (23.20)	-249.66 *** (29.24)	3.22 *** (0.22)	93.03 *** (26.67)
nb_chambres4		79.70 * (31.78)	-137.41 *** (37.64)	3.01 *** (0.27)	18.65 (34.20)
nb_chambres5+		-135.77 ** (42.28)	130.42 ** (46.17)	2.38 *** (0.32)	-139.12 *** (42.24)
log(surface)			1320.08 *** (16.31)		

	Model 1	Model 2	Model 3
(Intercept)	385.35 *** (7.05)	408.43 *** (27.37)	504.53 *** (36.16)
surface	11.19 *** (0.06)	11.15 *** (0.11)	9.58 *** (0.16)
typeHOUSE		21.01 (19.63)	75.26 ** (25.58)
PEBB		102.98 *** (23.78)	90.67 ** (30.53)
PEBC		11.30 (22.98)	-30.36 (29.94)
PEBD		-69.13 ** (22.78)	-79.09 ** (29.90)
PEBE		-82.07 *** (23.53)	-111.85 *** (30.57)
PEBF		-118.13 *** (25.29)	-119.83 *** (32.58)
PEBG		-99.27 *** (24.53)	-138.68 *** (31.69)
nb_chambres1		-12.01 (19.09)	7.97 (23.97)
nb_chambres2		4.86 (19.68)	73.96 ** (24.99)
nb_chambres3		156.43 *** (23.20)	250.02 *** (29.66)
nb_chambres4		79.70 * (31.78)	283.29 *** (40.85)
nb_chambres5+		-135.77 ** (42.28)	75.90 (54.97)
'Indice synthétique de difficulté 2010'		4.95 (6.16)	

## Chapter 2

# Analyse en Composantes Principales

### 2.1 Objectif:

L'objectif de ce TP est de réaliser une analyse en composante principale sur base de données électorales.

### 2.2 Données:

- données des élections présidentielles 2022 par communes : pres2022comm.csv (<https://unehistoireduconflitpolitique.fr/telecharger.html>)

Ce fichier contient les résultats des élections présidentielles de 2022 en format csv. Ces fichiers ont été générés à partir des résultats électoraux disponibles sur le site du Ministère de l'Intérieur et sur data.gouv.fr. J. Cagé et T. Piketty (2023) : Une histoire du conflit politique. Élections et inégalités sociales en France, 1789-2022. Le Seuil.

- communes: communes-20220101.shp (1)
- arrondissement municipaux: arrondissements\_municipaux-20180711.shp (2)
- table commune - région : commune2021.csv (3)
- revenus par commune : FILO2021\_DEC\_COM.xlsx (4)
- population par commune: population\_insee\_2021.xlsx (5)

(1) INSEE: <https://www.data.gouv.fr/fr/datasets/decoupage-administratif-communal-francais-issu-d-openstreetmap/>

- (2) INSEE: <https://www.data.gouv.fr/fr/datasets/decoupage-administratif-communal-francais-issu-d-openstreetmap/>
- (3) INSEE: <https://www.insee.fr/fr/information/2560452>
- (4) INSEE: <https://www.insee.fr/fr/statistiques/7756855?sommaire=7756859>
- (5) INSEE : <https://www.insee.fr/fr/statistiques/7739582?sommaire=7728826>

Pour ce TP nous utiliserons les packages suivants:

De façon maintenant habituelle:

```
library(tidyverse)
library(sf)
library(mapsf)
library(readxl)
library(remote)
```

Et deux packages développés pour réaliser des ACP et visualiser les résultats:

```
#install_version("estimability", "1.4.1")
#install.packages("FactoMineR")
#install.packages("factoextra")
library(FactoMineR)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https:// goo.g1/ve3
```

Les objectifs pour réaliser une ACP sont généralement :

1. Débroussailler un large set de données avec plein de variables
2. Réaliser un indicateur synthétique
3. Préparer à une classification

Dans le cas ici, on va plutôt réaliser une ACP suivant le premier objectif sur les votes exprimés aux élections en Île de France aux élections présidentielles de 2022. L'objectif sera de voir si on retrouve des structures dans la géographie de cette élection.

## 2.3 Importation des données

### 2.3.1 Élections 2022

Importation des données:

```
data<-read_delim("data/France/pres2022comm.csv", delim=",") %>%
  select(1:19) %>%
  filter(codecommune!="75056") # %>% # On enlève la communes de Paris
```

```
## Rows: 34867 Columns: 97
## -- Column specification -----
## Delimiter: ","
## chr (4): dep, nomdep, codecommune, nomcommune
## dbl (93): inscrits, votants, exprimes, voixARTHAUD, voixPOUTOU, voixROUSSEL, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

On prend une table pour ne sélectionner que l'Île de France:

```
communes_table<-read_delim("data/France/commune2021.csv")
```

```
## Rows: 37742 Columns: 12
## -- Column specification -----
## Delimiter: ","
## chr (11): TYPECOM, COM, REG, DEP, CTCD, ARR, NCC, NCCENR, LIBELLE, CAN, COMP...
## dbl (1): TNCC
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

data<-data %>%
  left_join(communes_table, by=c("codecommune"="COM"))%>%
  filter(REG==11) %>%
  select(-c(20:30))
```

### 2.3.2 Communes

```
##   using driver `ESRI Shapefile'
## Simple feature collection with 45 features and 4 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:  xmin: 2.224122 ymin: 43.19714 xmax: 5.532476 ymax: 48.90216
## Geodetic CRS:  WGS 84
communes<-bind_rows(communes,arrond_lyon_mars_paris) # On joint communes et arrondissements
rm(arrond_lyon_mars_paris)
```

On ne garde que les communes de l'Ile de France:

```
communes<-communes %>%
  left_join(communes_table, by=c("insee"="COM"))%>%
  filter(REG==11) %>%
  select(-c(5:15))

## Warning in sf_column %in% names(g): Detected an unexpected many-to-many relationship
## i Row 17 of `x` matches multiple rows in `y`.
## i Row 31675 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

On calcul la densité par communes:

```
communes<-communes %>%
  mutate(surface=as.numeric(st_area(geom))/10000) # On calcul le superficie de chaque commune
```

On créée un objet département pour la cartographie:

```
departements_Paris<-communes %>%
  left_join(communes_table, by=c("insee"="COM")) %>%
  filter(REG==11) %>%
  group_by(DEP,REG) %>%
  summarise(geom=st_union(geom))

## Warning in sf_column %in% names(g): Detected an unexpected many-to-many relationship
## i Row 17 of `x` matches multiple rows in `y`.
## i Row 31675 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.

## `summarise()` has grouped output by 'DEP'. You can override using the `.groups` argument.
```

```
rm(communes_table)
```

## 2.4 Recodage

Pour réaliser l'ACP il faut d'abord recoder les variables. On utilise généralement des variables en proportion:

```
data<-data %>%
  mutate(ARTHAUD=voixARTHAUD/exprimes,
        POUTOU=voixPOUTOU/exprimes,
        ROUSSEL=voixROUSSEL/exprimes,
        MELENCHON=voixMELENCHON/exprimes,
        JADOT=voixJADOT/exprimes,
        HIDALGO=voixHIDALGO/exprimes,
        LASSALLE=voixLASSALLE/exprimes,
        MACRON=voixMACRON/exprimes,
        PECRESSE=voixPECRESSE/exprimes,
        ZEMMOUR=voixZEMMOUR/exprimes,
        DUPONTAIGNAN=voixDUPONTAIGNAN/exprimes,
        MLEPEN=voixMLEPEN/exprimes)

# De façon raccourcie, il est possible de réaliser de la façon suivante:

# - en utilisant des pivot (tidyverse) et un group by
# data<-data %>%
#   pivot_longer(cols= 8:19,names_to ="candidat", values_to = "voix" ) %>%
#   mutate(voix=voix/exprimes) %>%
#   pivot_wider(values_from = "voix", names_from = "candidat")

# - de façon encore plus synthétique, en utilisant un across:
# data<-data %>%
#   mutate(across(8:19, ~ . /exprimes))
```

## 2.5 Première visualisation des données

```
communes_voix<-communes %>%
  left_join(data, by=c("insee"="codecommune"))
```

### 2.5.1 Une carte par variable

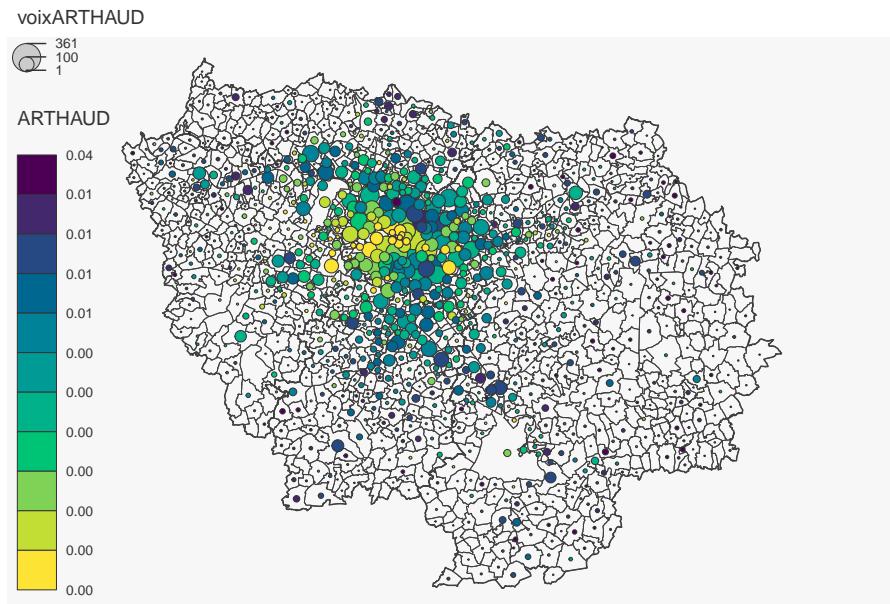
On peut réaliser une carte pour chaque candidat:

```

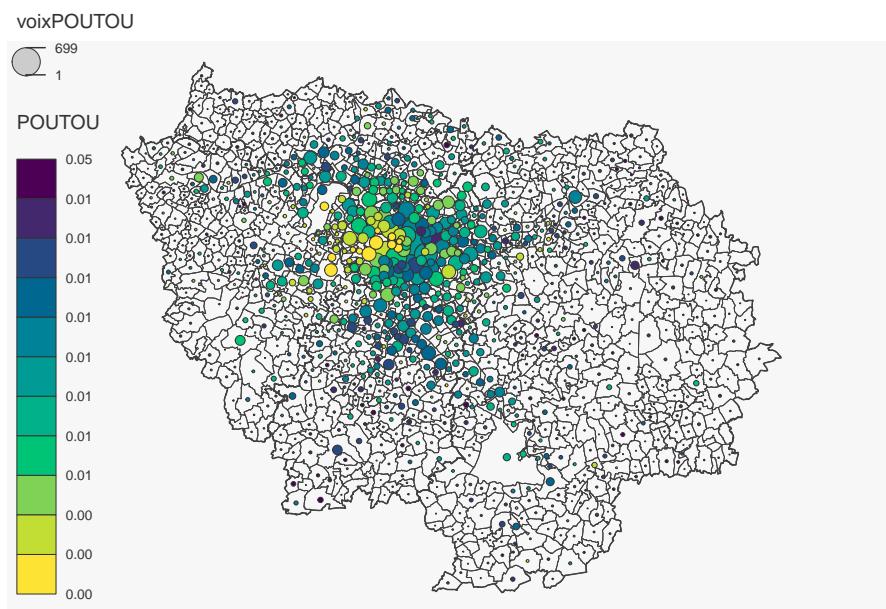
for (i in 12:23) {
  # mf_export(x = communes_voix ,
  #           filename =paste0("TP09/cartes_candidats/carte_",
  #                           names(communes_voix)[i],
  #                           width = 900)
  mf_map(x = communes_voix, col = NA, border = "gray25", lwd = 0.1)
  mf_map(x = departements_Paris, col = NA, border = "gray25", lwd = 1, add=T)
  mf_map(communes_voix ,
         var= c(names(communes_voix)[i], names(communes_voix)[i+12]),
         #val_max=max(unlist(communes_voix[,11:22])),
         type="prop_choro",
         pal="Viridis",
         inches=0.1,
         add=T)
  #dev.off()
}

```

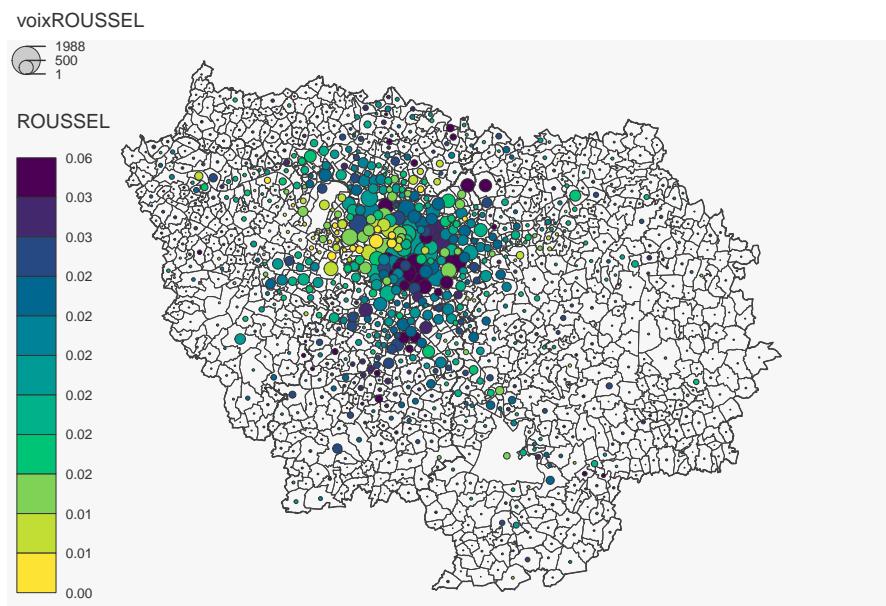
## 184 '0' values are not plotted on the map.

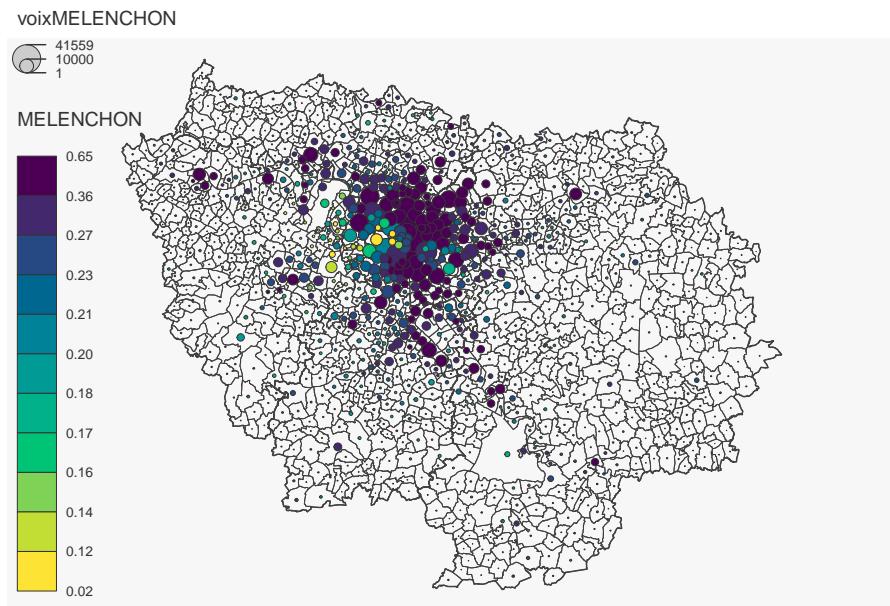


## 101 '0' values are not plotted on the map.

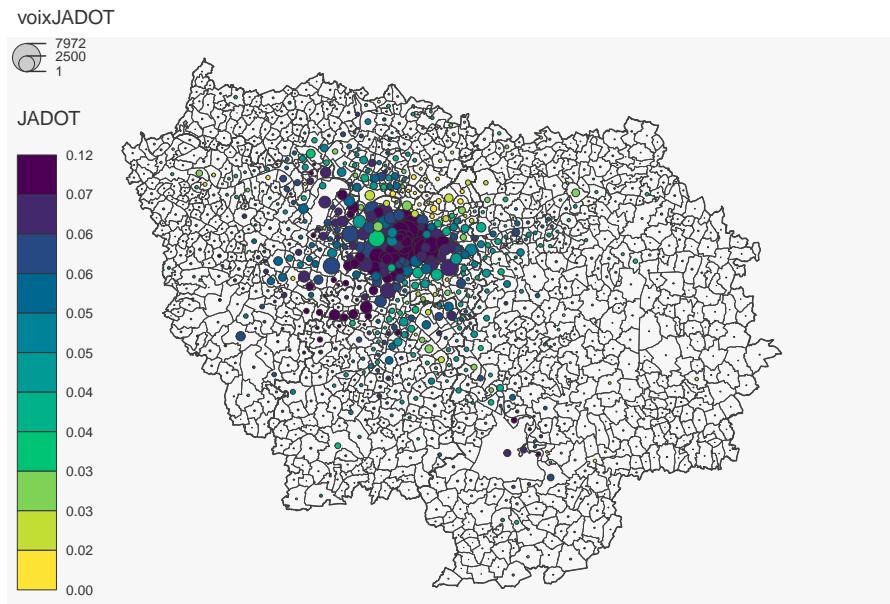


## 42 '0' values are not plotted on the map.

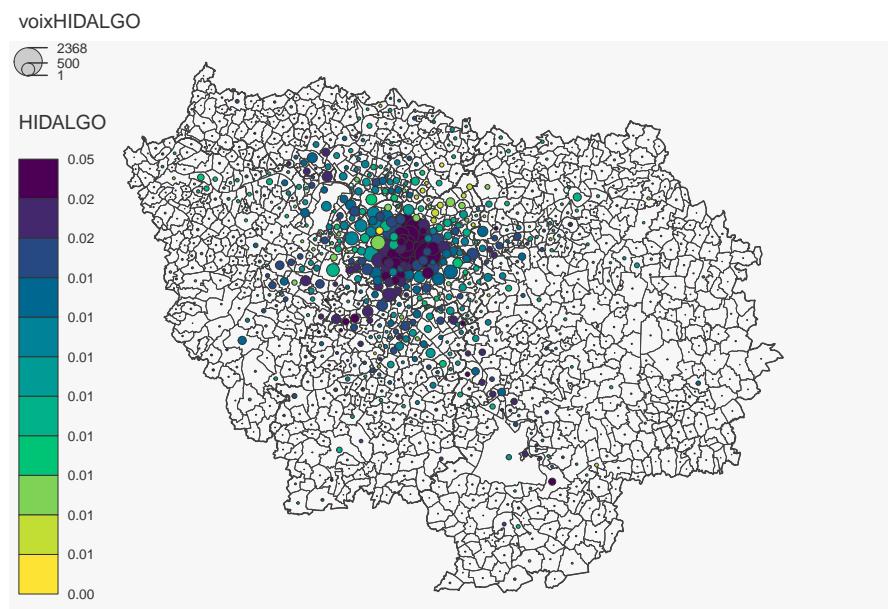




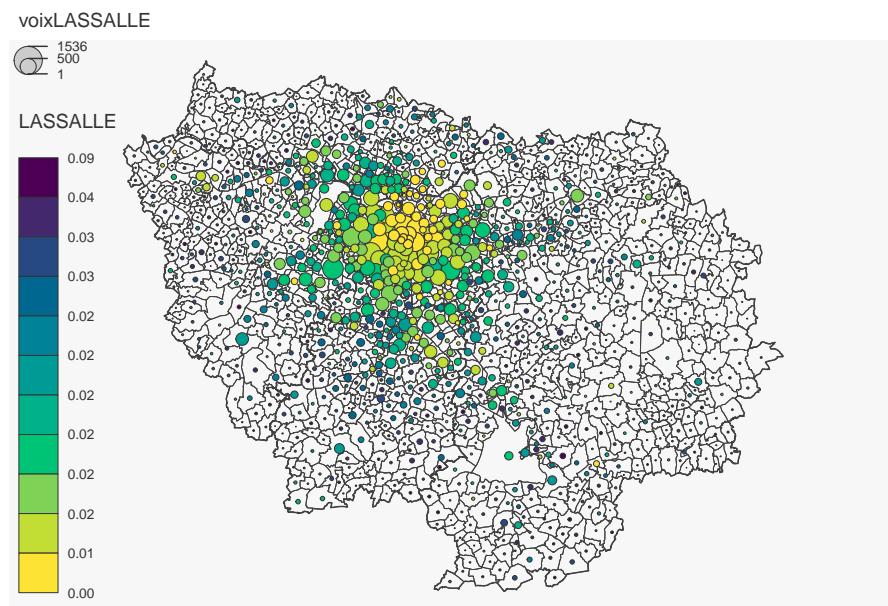
```
## 8 '0' values are not plotted on the map.
```

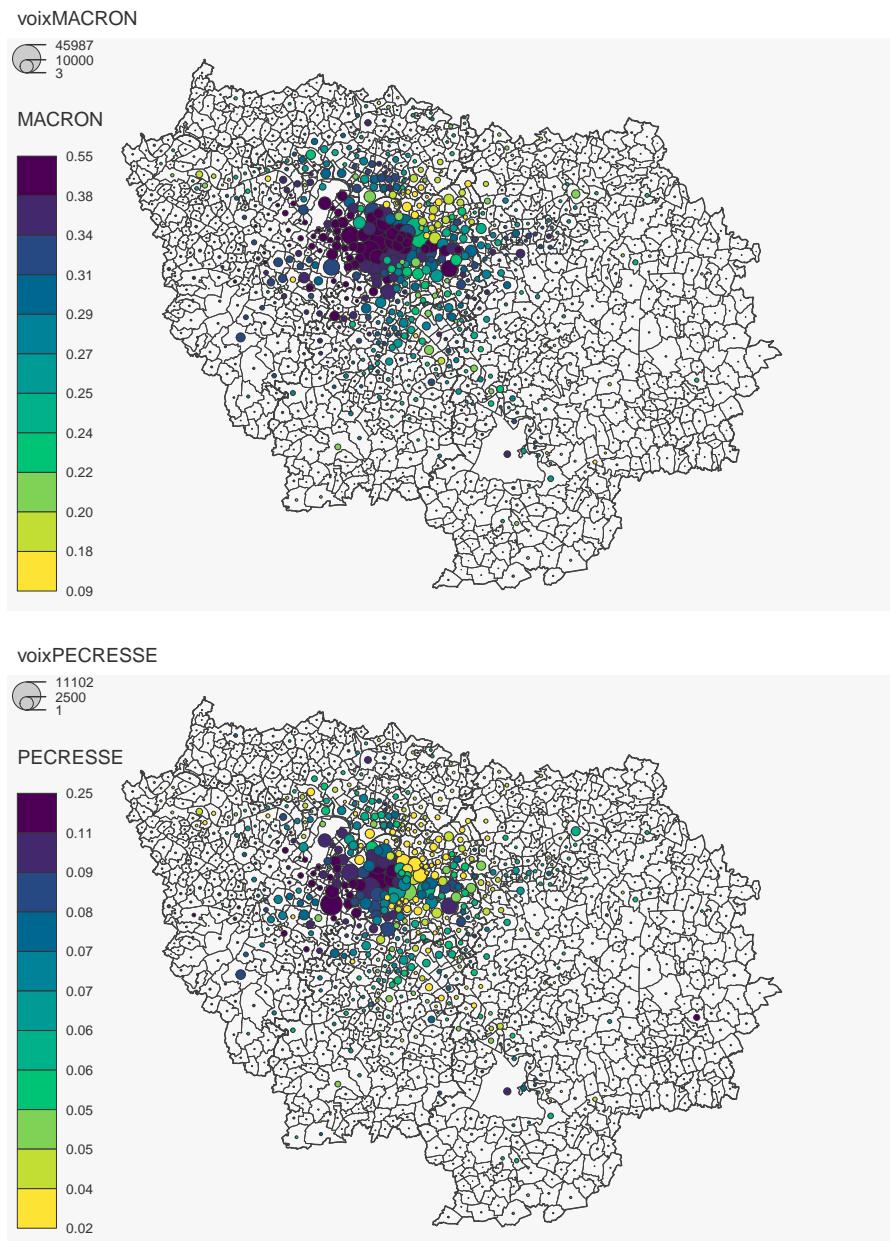


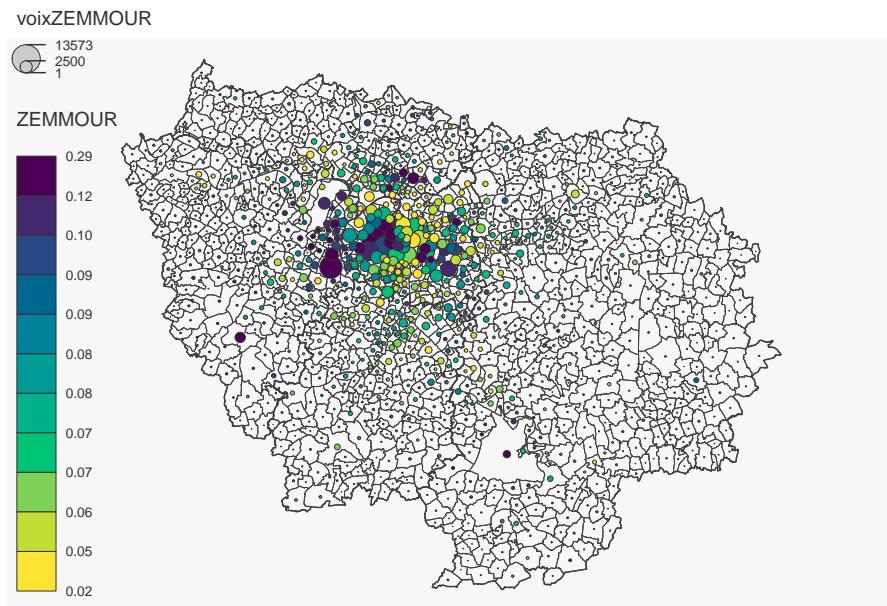
```
## 97 '0' values are not plotted on the map.
```



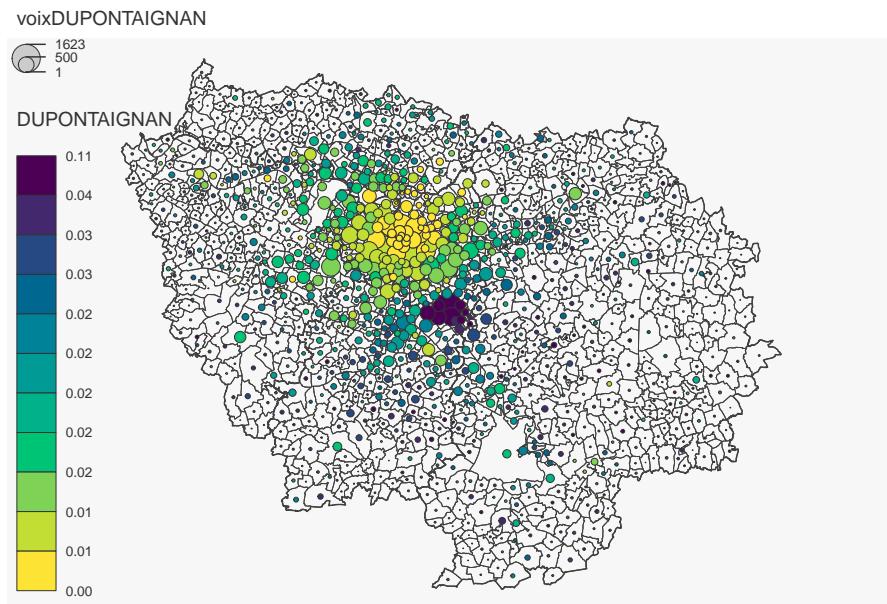
## 18 '0' values are not plotted on the map.

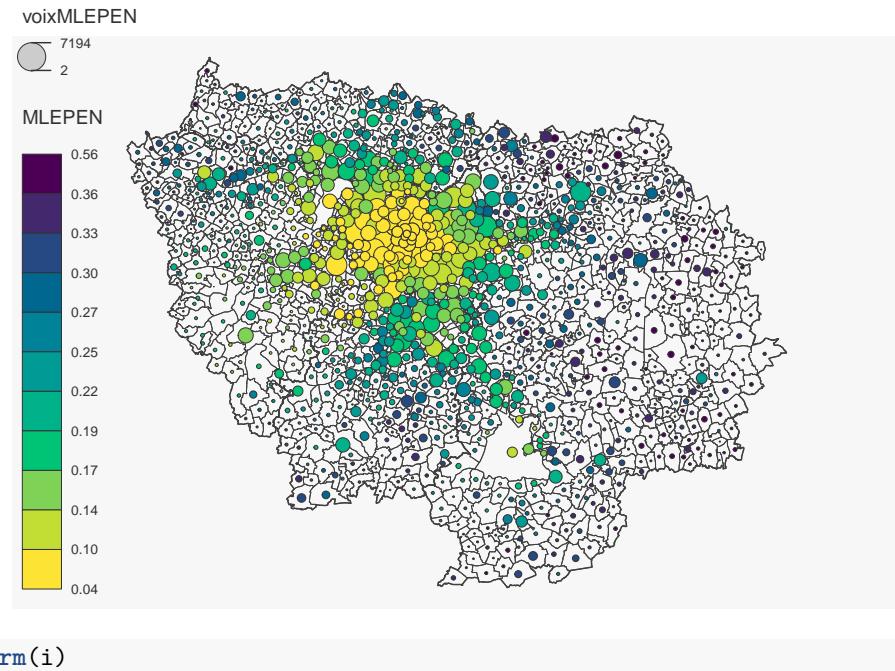






```
## 19 '0' values are not plotted on the map.
```





### 2.5.2 Matrice de corrélation

On peut réaliser une matrice de corrélation:

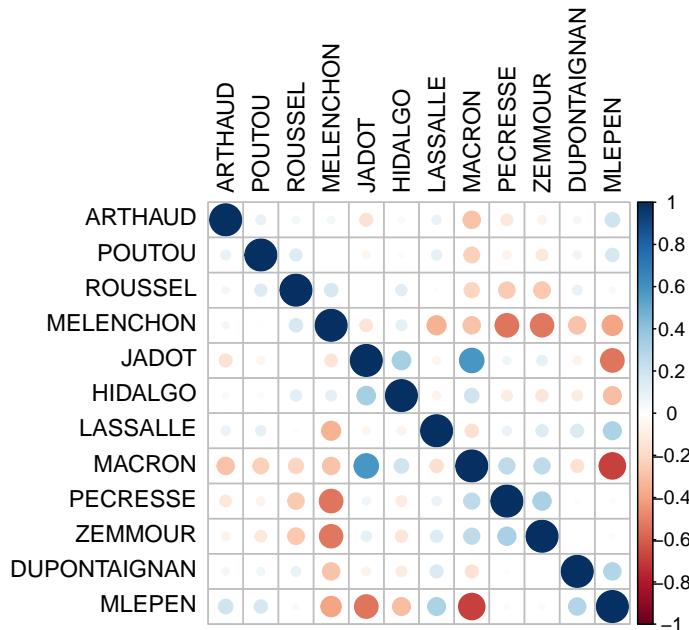
```
cor.mat <- round(cor(data[,20:31], use="complete.obs"),2) # round(,2) permet d'arrondir
cor.mat
```

	ARTHAUD	POUTOU	ROUSSEL	MELENCHON	JADOT	HIDALGO	LASSALLE	MACRON
## ARTHAUD	1.00	0.09	0.05	0.05	-0.16	-0.04	0.08	-0.29
## POUTOU	0.09	1.00	0.14	0.00	-0.06	-0.03	0.10	-0.24
## ROUSSEL	0.05	0.14	1.00	0.17	-0.01	0.12	0.01	-0.22
## MELENCHON	0.05	0.00	0.17	1.00	-0.15	0.11	-0.35	-0.29
## JADOT	-0.16	-0.06	-0.01	-0.15	1.00	0.33	-0.06	0.58
## HIDALGO	-0.04	-0.03	0.12	0.11	0.33	1.00	-0.07	0.20
## LASSALLE	0.08	0.10	0.01	-0.35	-0.06	-0.07	1.00	-0.17
## MACRON	-0.29	-0.24	-0.22	-0.29	0.58	0.20	-0.17	1.00
## PECRESSE	-0.13	-0.07	-0.26	-0.54	0.06	-0.11	0.08	0.25
## ZEMMOUR	-0.07	-0.13	-0.27	-0.53	0.10	-0.14	0.14	0.25
## DUPONTAIGNAN	0.05	0.06	0.09	-0.29	-0.07	-0.11	0.16	-0.16
## MLEPEN	0.20	0.17	0.03	-0.40	-0.54	-0.31	0.31	-0.68
##	PECRESSE	ZEMMOUR	DUPONTAIGNAN	MLEPEN				
## ARTHAUD	-0.13	-0.07			0.05	0.20		
## POUTOU	-0.07	-0.13			0.06	0.17		
## ROUSSEL	-0.26	-0.27			0.09	0.03		

```
## MELENCHON      -0.54   -0.53      -0.29   -0.40
## JADOT          0.06    0.10      -0.07   -0.54
## HIDALGO        -0.11   -0.14      -0.11   -0.31
## LASSALLE        0.08    0.14       0.16   0.31
## MACRON         0.25    0.25      -0.16   -0.68
## PECRESSE       1.00    0.32      -0.02   -0.03
## ZEMMOUR         0.32    1.00      -0.01   -0.03
## DUPONTAIGNAN   -0.02   -0.01      1.00   0.29
## MLEPEN          -0.03   -0.03      0.29   1.00
```

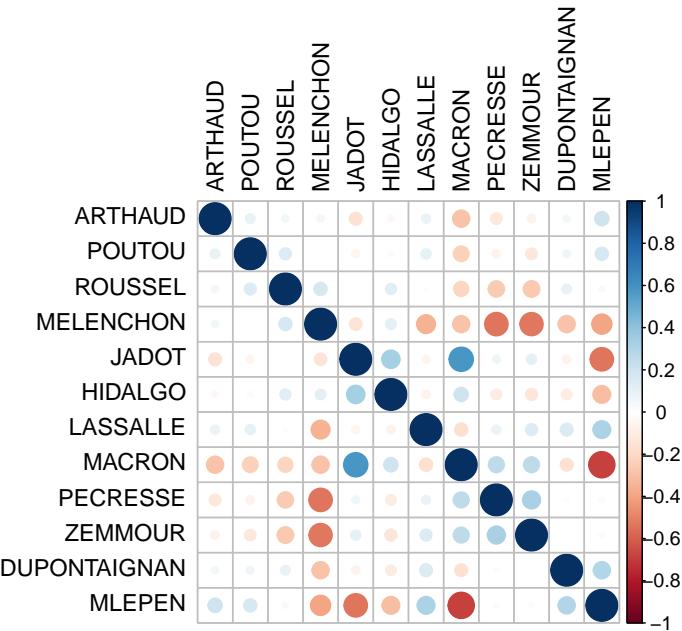
Le package corrplot permet de visualiser cette matrice de corrélation de façon plus intuitive:

```
library("corrplot")
## corrplot 0.94 loaded
corrplot(cor.mat, tl.col="black")
```



On peut exporter le résultat avec la fonction png de la façon suivante:

```
#png(file="TP09/matrice_correlation.png", width = 1000, height = 1000)
corrplot(cor.mat, tl.col="black")
```



```
#dev.off()
```

On voit entre autres des corrélations :

- positives entre :
  - le vote pour Hidalgo, Jadot et Macron,
  - le vote Pécresse, Zemmour et Macron
- négative entre:
  - le vote Marine Le Pen et le vote Jadot-Macron
  - le vote Mélenchon et le vote Pécresse-Zemmour

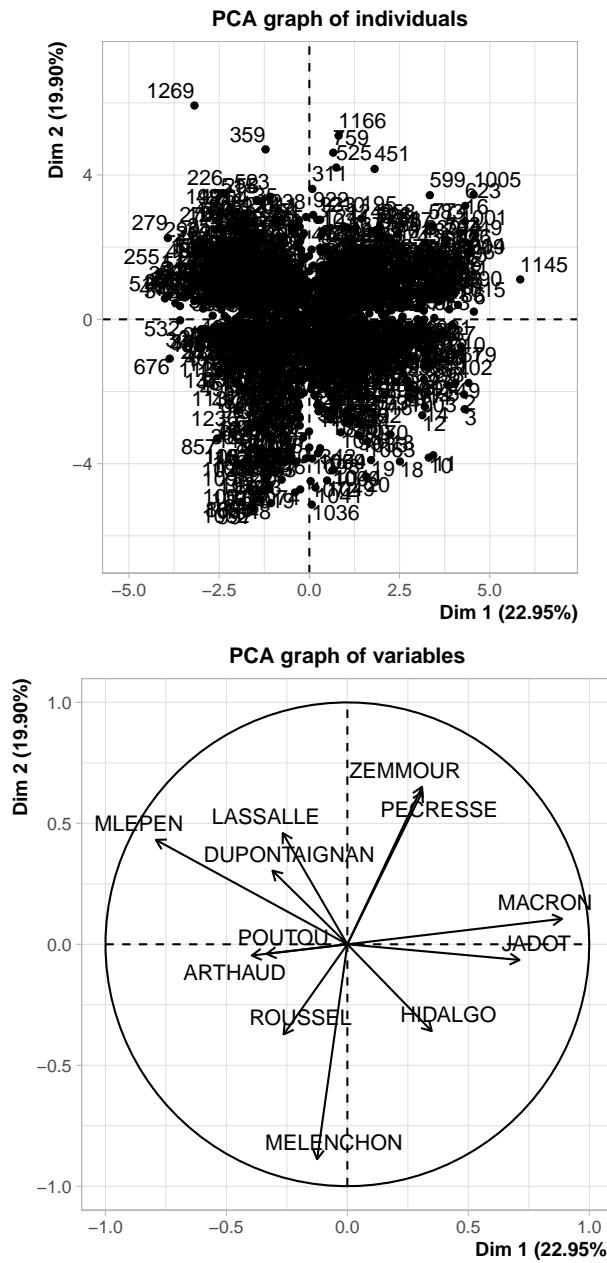
Très peu de corrélation entre les votes Zemmour et Marine Le Pen.

```
rm(cor.mat)
```

## 2.6 Réaliser l'ACP

### 2.6.1 Une ACP non pondérée

```
pca<-PCA(data[,20:31], graph=T, ncp = NULL)
```



L'objet `pca` est une liste dans lequel se trouve tous les résultats de l'ACP. On peut y accéder directement ou utiliser des fonctions préfaites pour visualiser les résultats:

```

# name           description
# 1  "$eig"       "eigenvalues"
# 2  "$var"        "results for the variables"
# 3  "$var$coord"  "coord. for the variables"
# 4  "$var$cor"    "correlations variables - dimensions"
# 5  "$var$cos2"   "cos2 for the variables"
# 6  "$var$contrib" "contributions of the variables"
# 7  "$ind"         "results for the individuals"
# 8  "$ind$coord"  "coord. for the individuals"
# 9  "$ind$cos2"   "cos2 for the individuals"
# 10 "$ind$contrib" "contributions of the individuals"
# 11 "$call"        "summary statistics"
# 12 "$call$centre" "mean of the variables"
# 13 "$call$ecart.type" "standard error of the variables"
# 14 "$call$row.w"  "weights for the individuals"
# 15 "$call$col.w"  "weights for the variables"

```

### 2.6.2 Pondérer le calcul de l'ACP

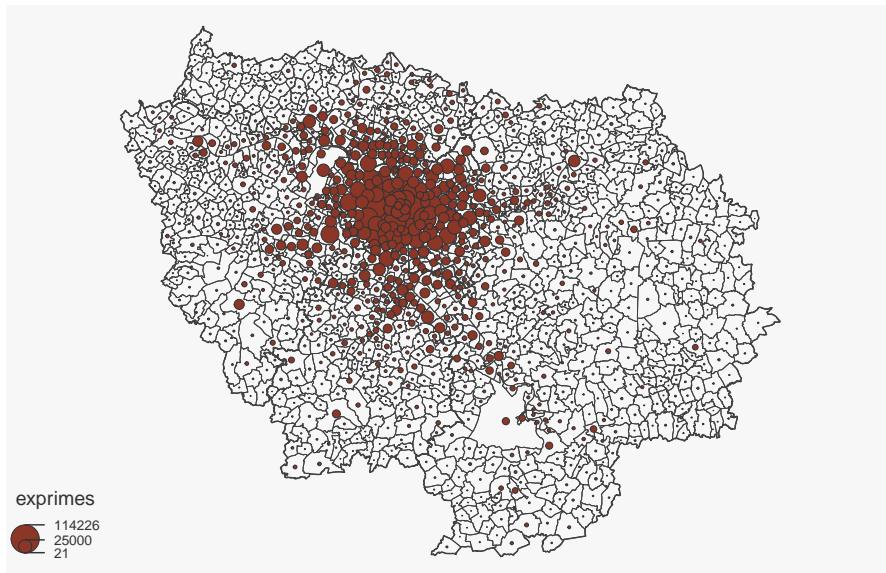
Dans le cas précédent chacuns des individus à le même poid et chaque variable également. On peut décider de modifier les poids.

Pourtant, il existe de grandes différences entre communes:

```

# mf_export(x = communes_voix ,
#            filename =paste0("TP09/cartes_votes_exprimés.png" ),
#            width = 900)
mf_map(x = communes_voix, col = NA, border = "gray25", lwd = 0.1)
mf_map(x = departements_Paris, col = NA, border = "gray25", lwd = 1, add=T)
mf_map(communes_voix ,
       var= "exprimés",
       type="prop",
       inches=0.1,
       add=T)

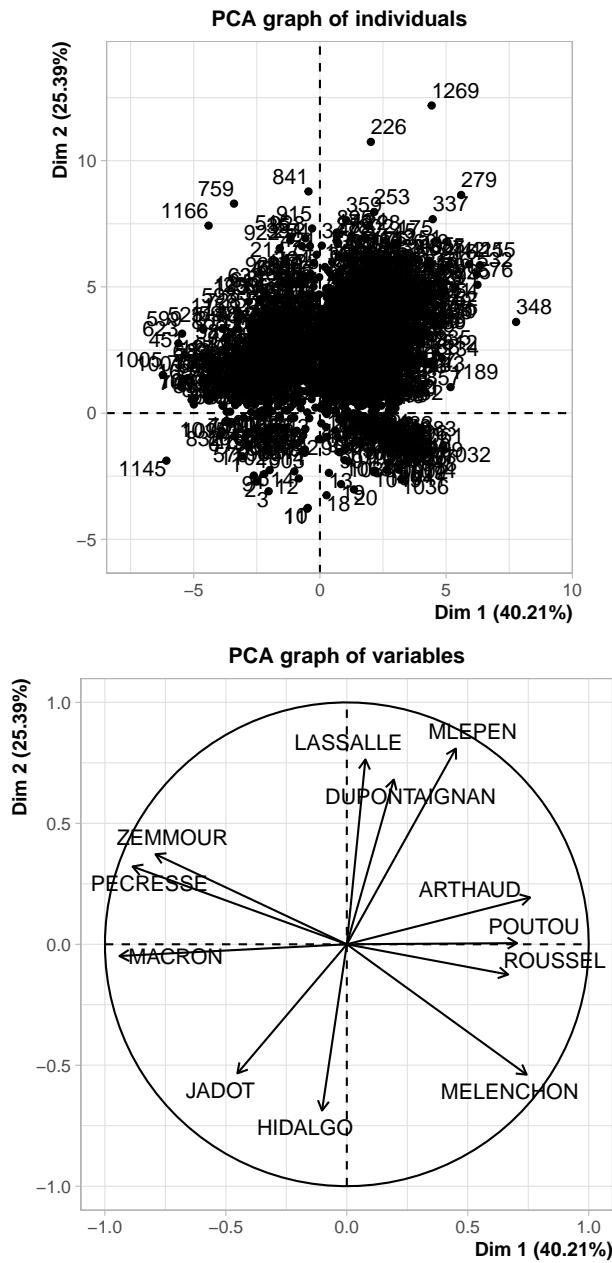
```



```
#dev.off()  
rm(communes_voix)
```

Si on souhaite pondérer les entités on peut utiliser “row.w =”:

```
pca<-PCA(data[,20:31], row.w =data$exprimes,ncp = NULL)
```



Pour pondérer les variables on peut utiliser col.w Attention: pour pondérer, il faut que tous les poids soient  $>0$ . Dans le cas présent ça ne pose pas de problème.

## 2.7 Visualiser les résultats

### 2.7.1 Cercles de corrélation

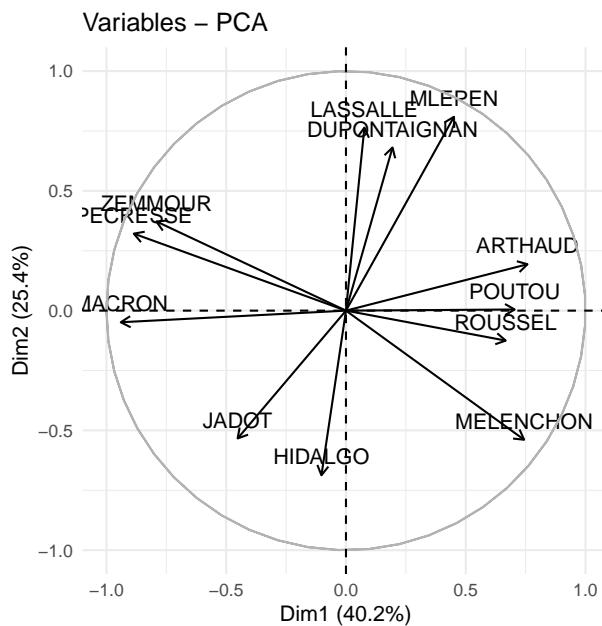
Ce résultat sont stocker dans l'objet pca:

```
pca$var$coord
```

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
## ARTHAUD	0.75783970	0.193700662	0.053886556	-0.33202051	0.22476966
## POUTOU	0.70415312	0.004765184	0.418058777	-0.07460159	0.23692366
## ROUSSEL	0.66687033	-0.124984377	0.299327347	0.62640195	0.14923271
## MELENCHON	0.74370403	-0.538863968	-0.340415783	-0.02007377	-0.09595007
## JADOT	-0.45229731	-0.533488252	0.663311687	-0.03924331	-0.01890221
## HIDALGO	-0.10246245	-0.687312206	0.620349032	-0.17878940	-0.04634065
## LASSALLE	0.07686699	0.764044366	0.449976355	-0.03134172	0.14026444
## MACRON	-0.94011924	-0.048504652	0.166318959	0.02380460	0.02352660
## PECRESSE	-0.88585300	0.321419606	-0.009943942	0.04293183	0.14109764
## ZEMMOUR	-0.79129943	0.372205374	-0.100007056	0.06375980	0.19413646
## DUPONTAIGNAN	0.19419953	0.681063286	0.357058840	0.04326915	-0.55375208
## MLEPEN	0.45088539	0.809295495	0.102917819	-0.06620236	0.04182531
	Dim.6	Dim.7	Dim.8	Dim.9	Dim.10
## ARTHAUD	-0.270976655	0.38461093	-0.05343325	-0.0045179775	-0.008370156
## POUTOU	0.499863557	0.02458871	-0.12931404	-0.0007009491	0.015475982
## ROUSSEL	-0.131302409	0.12543543	0.02976306	-0.0269964797	0.028203308
## MELENCHON	0.007735581	-0.03917968	-0.02553644	0.1644913917	0.010914950
## JADOT	-0.069747306	0.01101491	-0.04680005	-0.0019169799	-0.225853682
## HIDALGO	-0.040984394	-0.01196939	0.24982650	0.0054539159	0.183340122
## LASSALLE	-0.191952534	-0.29553028	-0.08181916	0.2339225539	0.025984130
## MACRON	-0.043682377	0.04938810	-0.18414230	-0.1216818436	0.031798367
## PECRESSE	0.017812438	0.11278338	-0.11425240	-0.0094676328	0.152069481
## ZEMMOUR	0.158100636	0.20993077	0.27169444	0.1793777244	-0.079291052
## DUPONTAIGNAN	0.082126654	0.22599626	-0.03151295	0.0577028315	0.017115750
## MLEPEN	-0.004296971	-0.15138365	0.19561196	-0.2456237780	-0.040599884
	Dim.11	Dim.12			
## ARTHAUD	-0.0140271624	3.282270e-10			
## POUTOU	-0.0156209607	3.891439e-10			
## ROUSSEL	-0.0056358853	1.336890e-09			
## MELENCHON	0.0401760796	2.360485e-08			
## JADOT	0.1036719950	3.831603e-09			
## HIDALGO	-0.0166654983	1.030643e-09			
## LASSALLE	-0.0303200827	9.593980e-10			
## MACRON	-0.1759761971	1.521580e-08			
## PECRESSE	0.2019841257	5.393004e-09			
## ZEMMOUR	-0.0469535055	5.553946e-09			
## DUPONTAIGNAN	0.0000869299	1.563518e-09			
## MLEPEN	0.0428961375	1.323804e-08			

Les cercle de corrélation permettent de visualiser entre les variables et les dimensions produites par l'ACP.

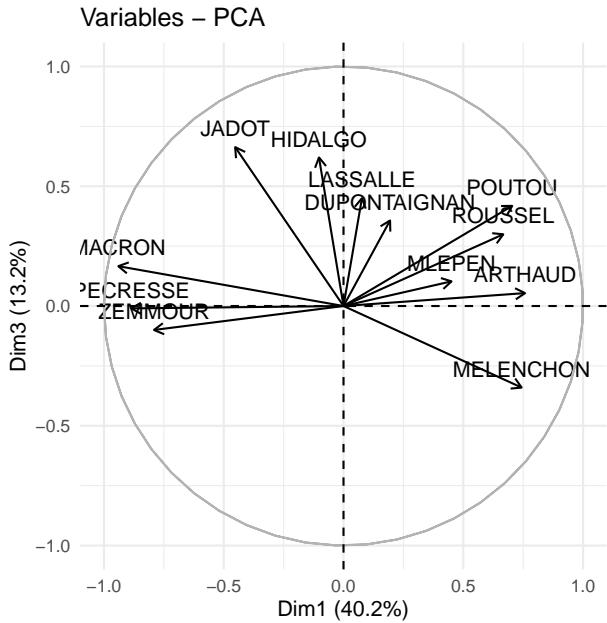
```
cercle_1_2<-fviz_pca_var(pca, axes = c(1, 2))+
  theme_minimal()
cercle_1_2
```



```
#ggsave(cercle_1_2,filename="TP09/cercle_correlation_1_2.png", width=7, height=7, bg="white")
#rm(cercle_1_2)
```

On peut décider de visualiser d'autres axes en modifiant le paramètre axes:

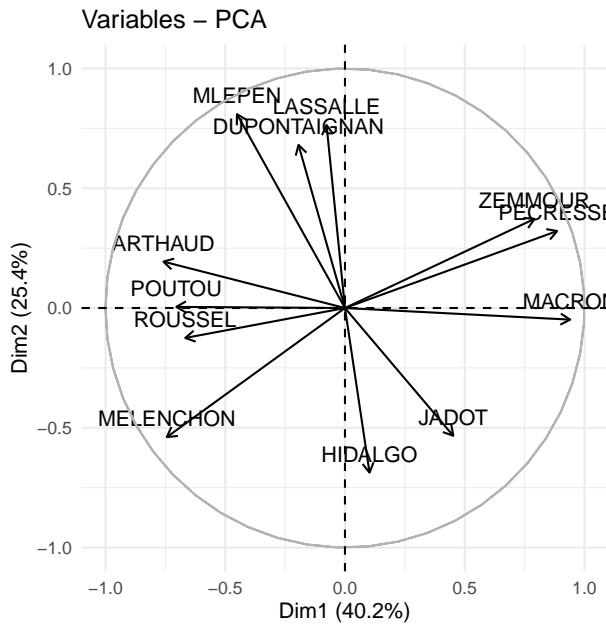
```
cercle_1_3<-fviz_pca_var(pca, axes = c(1, 3))+
  theme_minimal()
cercle_1_3
```



```
#ggsave(cercle_1_3,filename="TP09/cercle_correlation_1_3.png", width=7, height=7, bg="white")
#rm(cercle_1_3)
```

Pour faciliter la lecture, on peut inverser un axe en multipliant par -1 (dans notre cas, il est plus intuitif d'avoir la gauche politique à gauche et la droite à droite):

```
pca$var$coord[,1] <- -1*pca$var$coord[,1]
cercle_1_2<-fviz_pca_var(pca, axes = c(1, 2) )+
  theme_minimal()
cercle_1_2
```



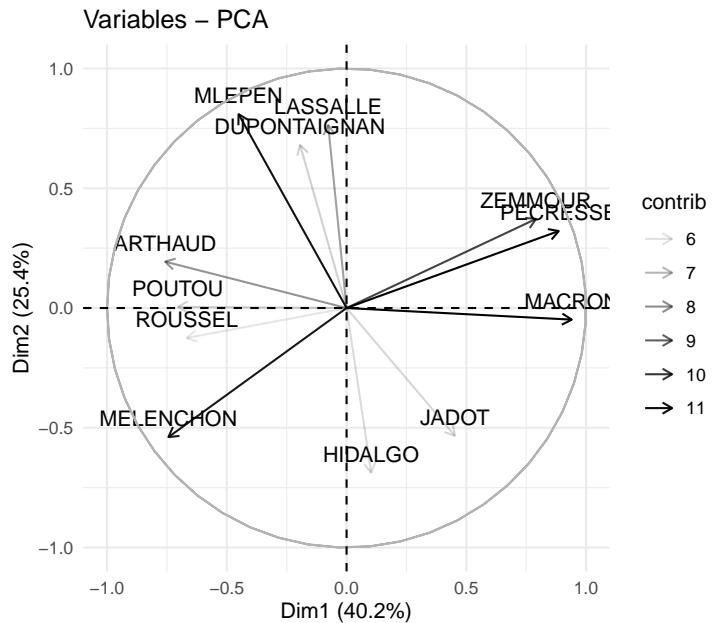
```
#ggsave(cercle_1_2,filename="TP09/cercle_correlation_1_2.png", width=7, height=7, bg="white")
```

Il faut alors aussi le faire sur les coordonnées des individus:

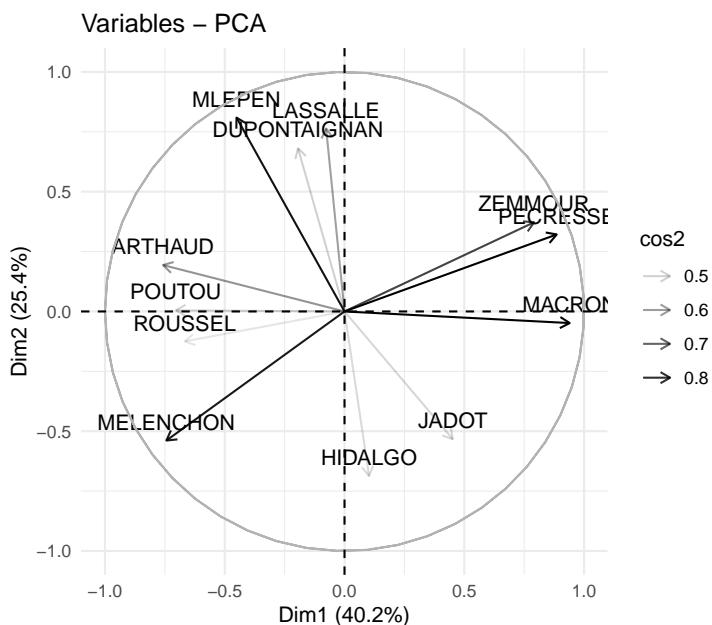
```
pca$ind$coord[,1] <- -1*pca$ind$coord[,1]
```

On peut décider de ne pas afficher toutes les variables (si il y en a trop ça devient illisible). Ici, on décide d'afficher surtout (en modifiant la transparence) celles qui ont le plus contribué à la construction de les deux axes (`alpha.var="contrib"`) ou celle qui sont le plus prise en compte par les deux axes (`alpha.var="cos2"`), soit encore de sélectionner celles qui ont un `cos2` supérieur à un seuil (`select.var = list(cos2 = 0.75)`) ou les 3 qui contribuent le plus (`select.var = list(contrib = 3))`):

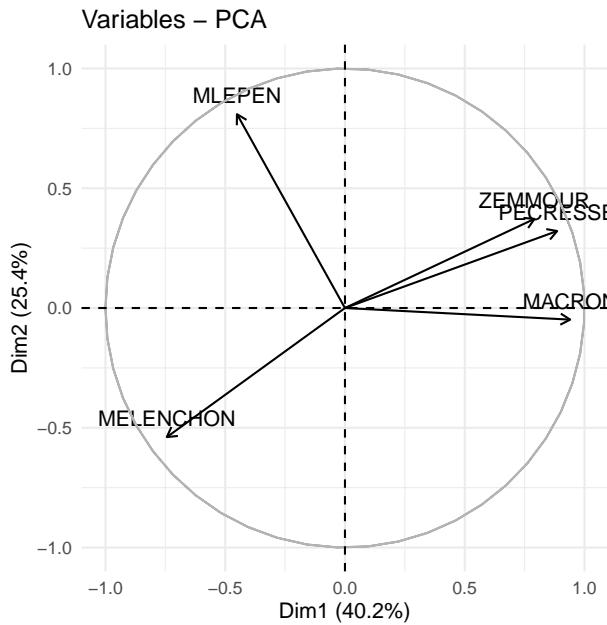
```
fviz_pca_var(pca, axes = c(1, 2), alpha.var="contrib")+theme_minimal()
```



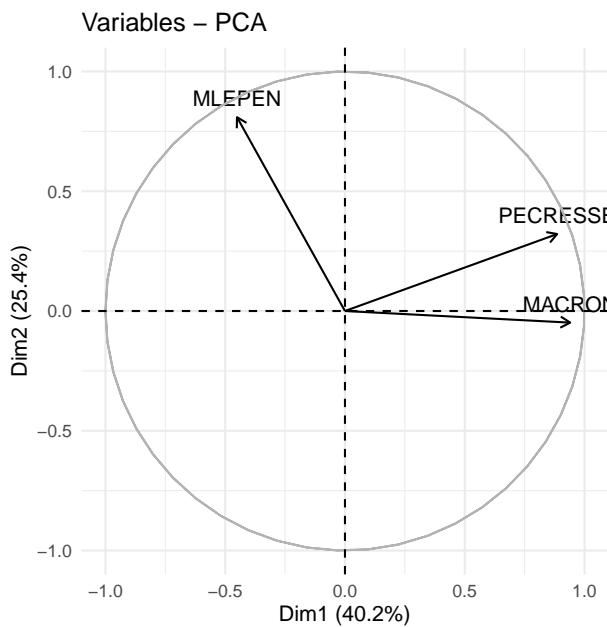
```
fviz_pca_var(pca, axes = c(1, 2), alpha.var="cos2") + theme_minimal()
```



```
fviz_pca_var(pca, axes = c(1, 2), select.var = list(cos2 = 0.75)) + theme_minimal()
```

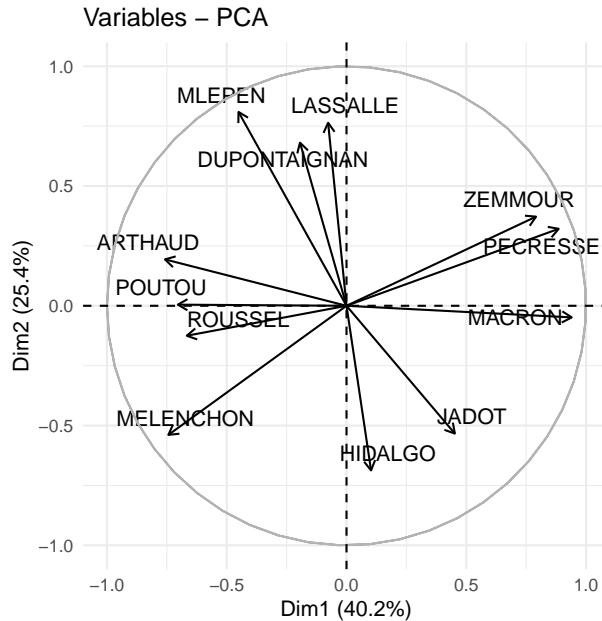


```
fviz_pca_var(pca, axes = c(1, 2), select.var = list(contrib = 3)) + theme_minimal()
```



Il est peut aussi être pertinent d'utiliser l'argument `repel=T` pour utiliser le package `repel` pour les étiquettes et éviter qu'elle ne se chevauchent:

```
fviz_pca_var(pca, axes = c(1, 2), repel=T ) + theme_minimal()
```



### 2.7.2 Contributions de chaque variables

La contribution de chaque variable à la construction de chacune de dimension:

```
pca$var$contrib
```

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
## ARTHAUD	11.9038406	1.231246e+00	0.183052896	19.83392132	9.58930308
## POUTOU	10.2770061	7.451467e-04	11.017687192	1.00132471	10.65438844
## ROUSSEL	9.2175433	5.126181e-01	5.648180142	70.59679126	4.22707266
## MELENCHON	11.4639072	9.528856e+00	7.305250657	0.07249975	1.74743980
## JADOT	4.2401420	9.339684e+00	27.736460944	0.27708309	0.06781672
## HIDALGO	0.2176015	1.550210e+01	24.259841866	5.75125091	0.40760167
## LASSALLE	0.1224650	1.915665e+01	12.764245064	0.17673574	3.73427942
## MACRON	18.3188531	7.720568e-02	1.743810408	0.10195308	0.10505823
## PECRESSE	16.2650617	3.390215e+00	0.006233513	0.33161753	3.77877609
## ZEMMOUR	12.9781893	4.546192e+00	0.630488306	0.73142974	7.15362007
## DUPONTAIGNAN	0.7816796	1.522150e+01	8.037025154	0.33684917	58.20260409
## MLEPEN	4.2137107	2.149298e+01	0.667723858	0.78854370	0.33203971
	Dim.6	Dim.7	Dim.8	Dim.9	Dim.10
## ARTHAUD	17.567987257	38.26088046	1.1309610	1.056017e-02	0.05892929
## POUTOU	59.780728472	0.15638066	6.6239329	2.541885e-04	0.20145614
## ROUSSEL	4.124807104	4.06960429	0.3508970	3.770484e-01	0.66905864

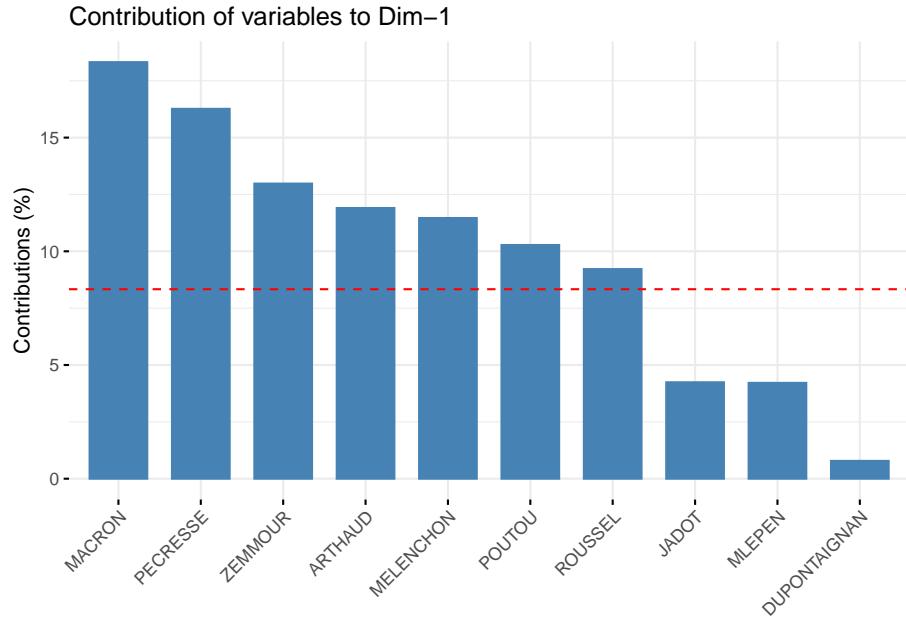
```

## MELENCHON      0.014316739  0.39703930  0.2583124  1.399810e+01  0.10020919
## JADOT          1.163893177  0.03138149  0.8675954  1.901157e-03  42.90604037
## HIDALGO         0.401878970  0.03705574  24.7230208  1.538862e-02  28.27347030
## LASSALLE        8.815479728  22.58997116  2.6517620  2.830917e+01  0.56791122
## MACRON         0.456531526  0.63089386  13.4317208  7.660100e+00  0.85049882
## PECRESSE        0.075911050  3.29004564  5.1707694  4.637305e-02  19.45128139
## ZEMMOUR         5.980334606  11.39892589  29.2405834  1.664638e+01  5.28825575
## DUPONTAIGNAN   1.613713804  13.21034744  0.3933712  1.722571e+00  0.24640866
## MLEPEN          0.004417566  5.92747407  15.1570737  3.121215e+01  1.38648022
##                           Dim.11      Dim.12
## ARTHAUD         2.190093e-01  0.01030897
## POUTOU          2.716054e-01  0.01449063
## ROUSSEL         3.535471e-02  0.17102445
## MELENCHON       1.796628e+00  53.31743705
## JADOT            1.196316e+01  1.40484268
## HIDALGO          3.091431e-01  0.10164445
## LASSALLE         1.023255e+00  0.08807739
## MACRON           3.446917e+01  22.15420622
## PECRESSE         4.541062e+01  2.78309654
## ZEMMOUR           2.453912e+00  2.95168558
## DUPONTAIGNAN    8.411265e-06  0.23392274
## MLEPEN            2.048139e+00  16.76926331

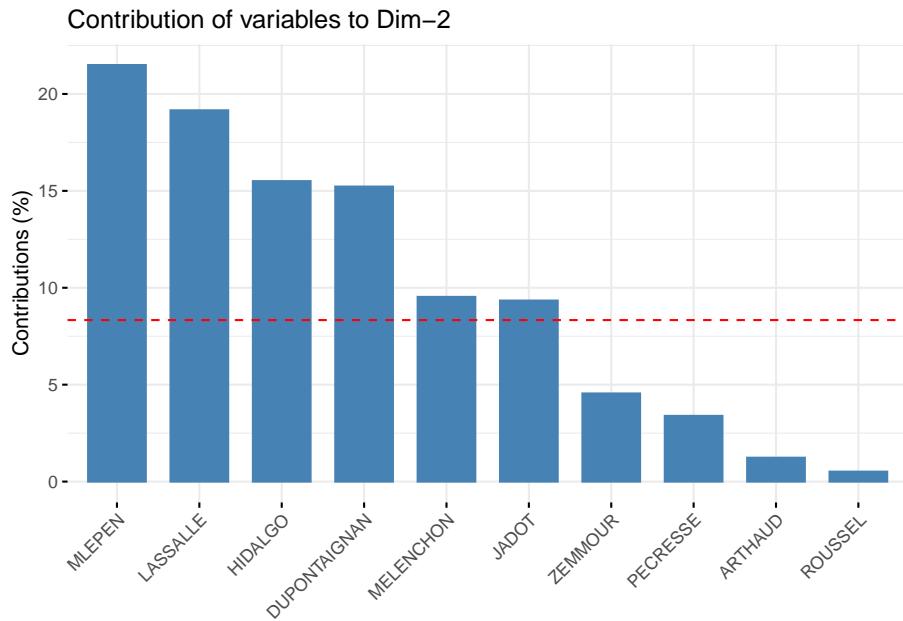
```

On peut également afficher de façon graphique avec fviz\_contrib:

```
fviz_contrib(pca, choice = "var", axes = 1, top = 10)
```



```
fviz_contrib(pca, choice = "var", axes = 2, top = 10)
```



### 2.7.3 Valeurs propres et % de la variance expliquée

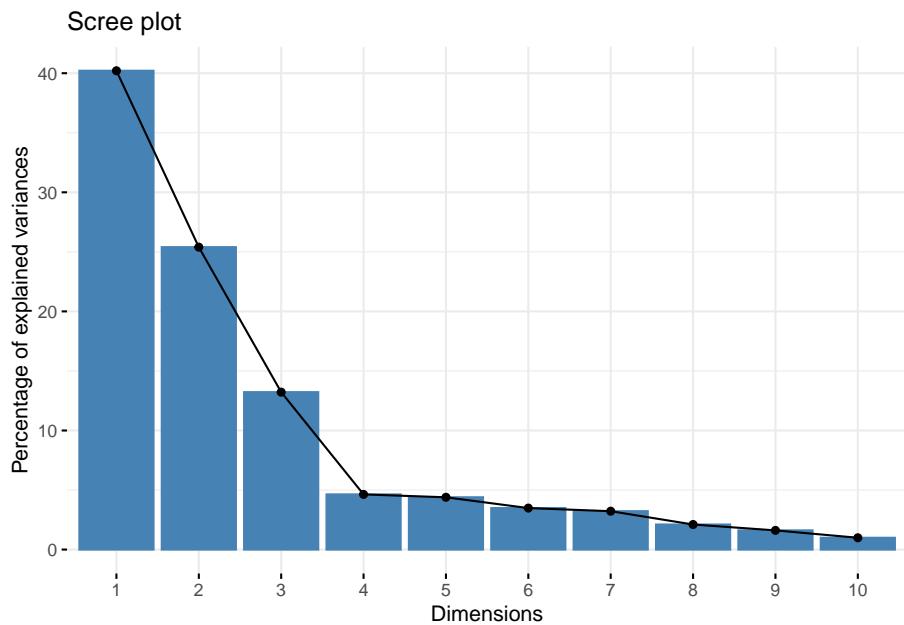
Pour afficher les valeurs propres (eigen value) et le pourcentage de variance expliquée, on peut simplement afficher le tableau compris dans l'objet pca

```
pca$eig
```

	eigenvalue	percentage of variance	cumulative percentage of variance
## comp 1	4.824670e+00	4.020558e+01	40.20558
## comp 2	3.047316e+00	2.539430e+01	65.59988
## comp 3	1.586296e+00	1.321913e+01	78.81902
## comp 4	5.558034e-01	4.631695e+00	83.45071
## comp 5	5.268516e-01	4.390430e+00	87.84114
## comp 6	4.179668e-01	3.483056e+00	91.32420
## comp 7	3.866235e-01	3.221863e+00	94.54606
## comp 8	2.524500e-01	2.103750e+00	96.64981
## comp 9	1.932934e-01	1.610779e+00	98.26059
## comp 10	1.188874e-01	9.907285e-01	99.25132
## comp 11	8.984151e-02	7.486793e-01	100.00000
## comp 12	1.045041e-15	8.708673e-15	100.00000

On peut faire un graphique de la variance expliquées:

```
fviz_screeplot(pca, ncp=10)
```



Ces informations sont utiles pour nous aider à décider combien de composante retenir. Généralement, on s'arrange avec ces règles non-strictes: - eigenvalue  $> 1$  - variance cumulée  $> 70\%$  - gain faible d'une composante supplémentaire (ajouter une dimension supplémentaire ne permet pas de gagner beaucoup en variance expliquée)

#### 2.7.4 Carte de scores

Chaque individus est reprojeté sur les nouvelles dimensions. Leurs scores sont stocké dans l'objet pca:

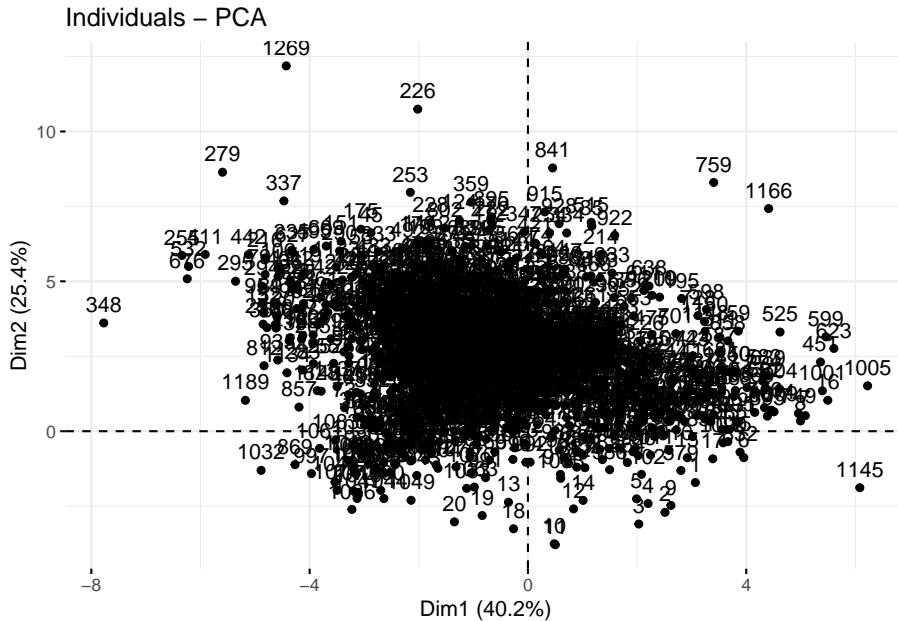
```
head(pca$ind$coord)
```

```
##      Dim.1     Dim.2     Dim.3     Dim.4     Dim.5     Dim.6
## 1 3.067555 -1.7114152  0.08150264 -0.185735850 -0.41247012 -0.1498315
## 2 2.511828 -2.7064534  0.70245008 -0.193923389 -0.70891924 -0.4567220
## 3 2.031107 -3.0966114  1.07311078 -0.404287195 -0.18691460 -0.5270521
## 4 2.201343 -2.4124210  1.18680228 -0.001301227 -0.20907736  0.2881203
## 5 1.991101 -2.2577563  1.02941720 -0.014047196 -0.02204558 -0.1995434
## 6 3.955340 -0.8826657 -0.20675170  0.106337247  0.01854102 -0.1812643
##          Dim.7     Dim.8     Dim.9     Dim.10    Dim.11    Dim.12
## 1 -0.31386218  0.51897558 -0.4392267  0.16799431 -0.45550607 -2.122990e-09
## 2 -0.53505432  0.36256495 -0.6205415 -0.13143940 -0.64944655 -2.768542e-09
## 3 -0.32566793  0.04083631 -0.7457277 -0.17322620 -0.51307636 -6.765777e-09
```

```
## 4 -0.18494092 0.55795434 -0.3835738 0.18158093 -0.51829968 -1.631543e-09
## 5 -0.07027549 0.24404548 -0.1594431 0.07828135 -0.03678451 -1.845831e-09
## 6 0.07415939 -0.23014907 -0.3167121 0.16569411 -0.35174294 -2.268829e-09
```

On peut souhaiter les visualiser dans un graphique à deux dimensions:

```
fviz_pca_ind(pca)
```



Forcément cela à du sens que si on a peu d'individus. Si les individus sont des entités géographiques, on peut souhaiter en faire des cartes.

On peut récupérer les score de chaque individus en collant avec bind\_rows les coordonnées avec le set de données initiale. On doit utiliser ici bind\_rows plutôt qu'une jointure (left\_join, right\_join) parce que l'agorithme de FactoMineR ne conserve pas d'ID lors de la réalisation de l'ACP mais il conserve bien l'ordre des lignes mise en entrée:

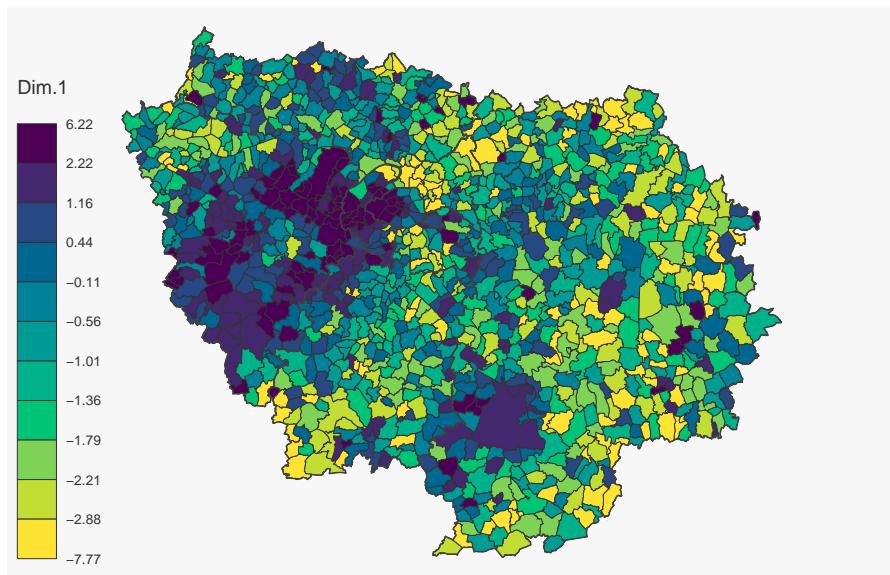
```
data_pca<-bind_cols(data,
                      pca$ind$coord)
```

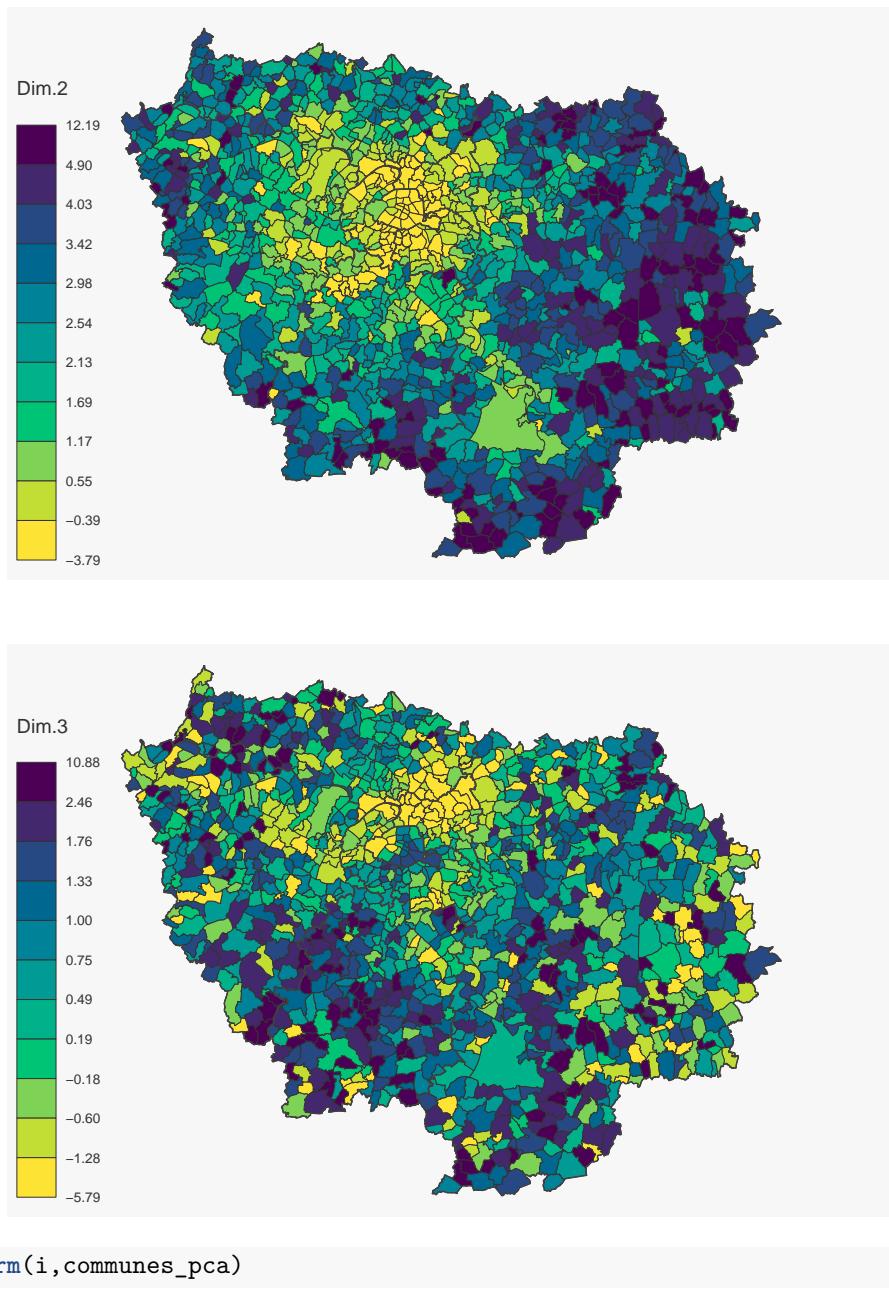
On réalise une boucle pour visualiser les 3 premières dimensions en faisant d'abord la jointure entre nos résultats et l'objet commune:

```
communes_pca<-communes %>%
  inner_join(data_pca, by=c("insee"="codecommune"))

for (i in 1:3){
```

```
# mf_export(x = communes_pca ,
#            filename =paste0("TP09/carte_acp_dim",i,".png" ),
#            width = 900)
mf_map(communes_pca ,
       var= paste0("Dim.",i),
       type="choro",
       pal= "Viridis",
       lwd=0.001)
mf_map(x = departements_Paris, col = NA, border = "gray25", lwd = 1, add=T)
# dev.off()
}
```





## 2.8 Ajouter des variables supplémentaires

Pour aider à l’interprétation des axes, il peut être utile d’importer des nouvelles variables qui ne seront pas utilisées dans l’ACP mais qu’on pourra néanmoins utiliser comme repère.

### 2.8.1 Importation des données

#### 2.8.1.1 Population

On importe les données de population:

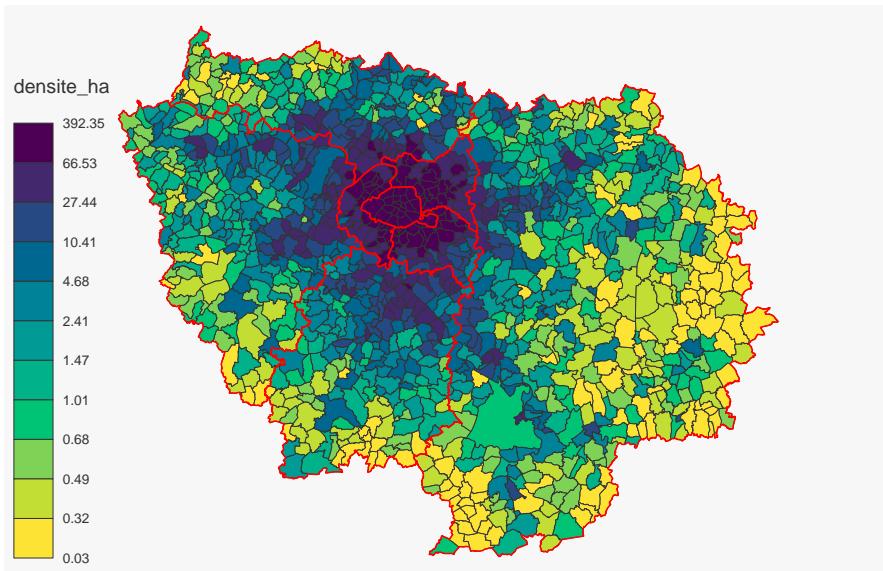
```
pop<-read_excel("data/France/population_insee_2021.xlsx", sheet="Communes") %>%
  mutate(codecommune=paste0(`Code département`, `Code commune`)) %>%
  filter(`Code région`=="11") # on garde que la France métropolitaine
```

On joint avec le fichier spatial des communes et calcul la densité de population

```
communes_pop<- communes %>%
  left_join(pop, by=c("insee"="codecommune")) %>%
  mutate(densite_ha=`Population totale`/surface) %>%
  select(insee, densite_ha)
```

On peut réaliser une carte de la densité de population

```
# mf_export(x = communes_pop ,
#            filename =paste0("TP09/carte_densite.png" ),
#            width = 900)
mf_map(communes_pop ,
       var= "densite_ha",
       type="choro",
       pal= "Viridis",
       lwd=0.001)
mf_map(x = departements_Paris, col = NA, border = "red", lwd = 1, add=T)
```



```
#dev.off()
```

On enlève transforme l'objet en datafram et on enlève la colonne geom:

```
communes_pop_df<-communes_pop %>%
  as.data.frame() %>%
  select(-geom)
```

On réalise la jointure avec les données de base:

```
data<-data %>%
  left_join(communes_pop_df, by=c("codecommune"="insee"))

rm(pop,communes_pop, communes_pop_df)
```

### 2.8.1.2 Revenus

On importe les données de revenus, on conserve les champ qui nous intéresse et on transforme le champ revenu médian en “numeric”:

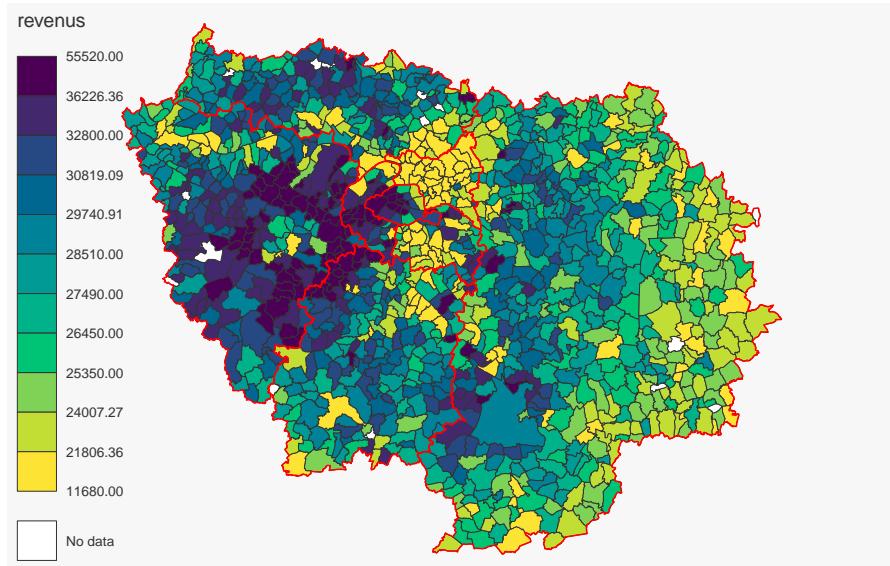
```
revenus<-read_excel("data/France/FIL02021_DEC_COM.xlsx", sheet=2) %>%
  mutate(revenus=as.numeric(revenu_median)) %>%
  select(CODGEO, revenus)
```

On joint avec le fichier spatial des communes et les revenus:

```
communes_revenus<- communes %>%
  left_join(revenus, by=c("insee"="CODGEO"))
```

On peut réaliser une carte de revenu:

```
# mf_export(x = communes_revenus ,
#           filename = paste0("TP09/carte_revenus.png" ),
#           width = 900)
mf_map(communes_revenus ,
       var= "revenus",
       type="choro",
       pal= "Viridis",
       lwd=0.001)
mf_map(x = departements_Paris, col = NA, border = "red", lwd = 1, add=T)
```



```
#dev.off()
rm(communes_revenus)
```

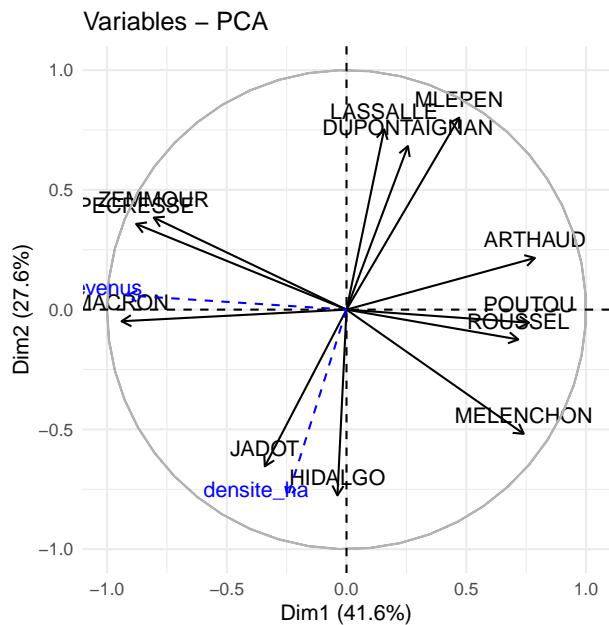
On réalise la jointure avec les données de base:

```
data<-data %>%
  left_join(revenus, by=c("codecommune"="CODGEO"))
rm(revenus)
```

### 2.8.2 ACP avec les variables supplémentaires

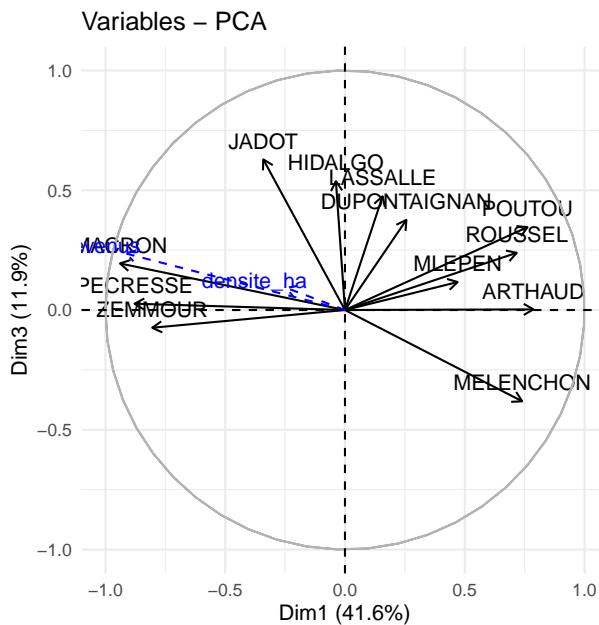
```
pca<-PCA(data[,20:33], quanti.sup = 13:14, row.w =data$exprimes, graph=F, ncp = NULL)
```

```
## Warning in PCA(data[, 20:33], quanti.sup = 13:14, row.w = data$exprimes, :
## Missing values are imputed by the mean of the variable: you should use the
## imputePCA function of the missMDA package
cercle_1_2<-fviz_pca_var(pca, axes = c(1, 2))+
  theme_minimal()
cercle_1_2
```



```
#ggsave(cercle_1_2,filename="TP09/cercle_correlation_1_2_revenu_pop.png", width=7, height=7, bg=
rm(cercle_1_2)

cercle_1_3<-fviz_pca_var(pca, axes = c(1, 3))+
  theme_minimal()
cercle_1_3
```



```
#ggsave(cercle_1_3,filename="TP09/cercle_correlation_1_3_revenu_pop.png", width=7, height=7)
rm(cercle_1_3)
```

On observe des fortes corrélations entre la première dimension qui dépeind une opposition gauche-droite et la variable de revenus, les revenus élevés étant associés aux communes où le vote de droite est important. La seconde dimension est quant à elle fortement corrélée à la densité de population et permet d'interpréter cette dernière comme l'opposition centre urbain-périmétrie

## 2.9 Aller plus loin

Il est possible de faire des ACP sur des données d'enquêtes soit en utilisant les poids comme présenté précédemment soit grâce au package :svyprcomp (<https://r-survey.r-forge.r-project.org/survey/html/svyprcomp.html>)

On peut vouloir réaliser des légères rotations des axes pour améliorer la lecture des axes: <https://stats.stackexchange.com/questions/59213/how-to-compute-varimax-rotated-principal-components-in-r> <https://dimension.usherbrooke.ca/pages/87>

<https://sites.google.com/site/rgraphiques/4--stat/machine-learning-biostatistiques-analyse-de-donn%C3%A9es/analyse-en-composantes-principales/la-rotation-varimax>

## 2.10 Ressources utilisées:

- [http://www.sthda.com/english/wiki/wiki.php?id\\_contents=7851](http://www.sthda.com/english/wiki/wiki.php?id_contents=7851)
- [http://factominer.free.fr/index\\_fr.html](http://factominer.free.fr/index_fr.html)



# Chapter 3

## Classification

### 3.1 Objectifs

Ce tp a pour objectif de réaliser une classification ascendante hiérarchique des secteurs statistiques de Bruxelles en utilisant des variables de la structure de la propriété des logements mis en location.

### 3.2 Préparation

De façon maintenant habituelle:

```
library(tidyverse) # Manipulation des données
library(sf) # Manipulation des données spatiales
library(mapsf) # Réalisation de carte
library(readxl) # Importation de données Excel
library(kableExtra) # Réalisation de tableau

## 
## Attachement du package : 'kableExtra'

## L'objet suivant est masqué depuis 'package:huxtable':
##
##     add_footnote

## L'objet suivant est masqué depuis 'package:dplyr':
##
##     group_rows
```

Et deux packages développés pour réaliser des ACP et visualiser les résultats:

```
library(FactoMineR)
library(factoextra)
```

Pour ce TP, on utilisera également le packages JLutils:

```
#library(devtools)
#devtools::install_github("lamarange/JLutils")
library(JLutils)

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

##
## Attachement du package : 'JLutils'
## Les objets suivants sont masqués depuis 'package:ggstats':
##
##     ggsurvey, signif_stars
```

Pour cette classification, on utiliser des indicateurs construits à partir des données du cadastre (1 janvier 2015) utilisée dans ma thèse (<https://difusion.ulb.ac.be/vufind/Record/ULB-DIPOT:oai:dipot.ulb.ac.be:2013/359086/Holdings>)

Les variables utilisées sont:

- cd\_sector = code secteur statistique
- NB\_LOG = nombre de logements total
- NB\_LOGLOUE\_TOT = nombre de logements loués total
- NB\_LOCDOM = nombre de logements loués à domicile
- prop\_occup = proportion de propriétaires occupants
- prop\_LOCDOM = proportion de logements loués à domicile
- prop\_LOC\_HBX = proportion de logements loués par des propriétaires habitants en dehors de la région de Bruxelles
- dist\_200 = proportion de logements loués par des propriétaires domiciliés à moins de 200 m
- cinq\_plus = proportion de logements loués possédés par des propriétaires possédants 5 logements ou plus à Bruxelles
- vingt\_plus = proportion de logements loués possédés par des propriétaires possédants 20 logements ou plus à Bruxelles
- log\_sociaux = proportion des logements sociaux parmi les logements loués
- entrep\_sprl\_sa = proportion des logements possédés par des entreprises privées parmi les logements loués

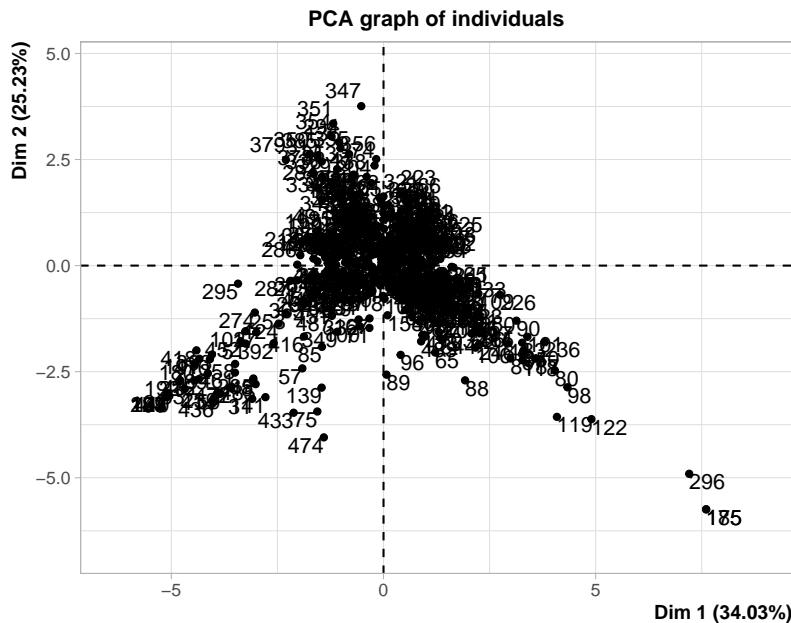
```
cadastre <- read_delim( "data/cadastre_2015_secteurs.csv", delim=";") %>%
  filter(NB_LOGLOUE_TOT>=200)
```

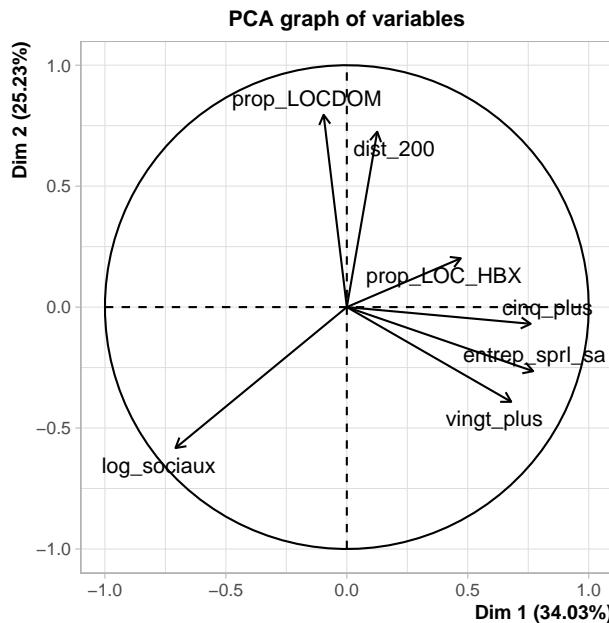
```
## Rows: 724 Columns: 12
## -- Column specification -----
## Delimiter: ";"
## chr (1): cd_sector
## dbl (11): NB_LOG, NB_LOGLOUE_TOT, NB_LOCDOC, prop_occup, prop_LOCDOC, prop_L...
## 
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

### 3.3 ACP

Pour réaliser la classification, on va d'abord réaliser une classification. Cela nous permet d'éliminer le bruit.

```
pca<-PCA(cadastre[,6:12], row.w =cadastre$NB_LOGLOUE_TOT,ncp = NULL)
```





On se demande combien de composante garder: les 3 premières représentent 80% de la variance.

```
fviz_screeplot(pca, ncp=10)
pca$eig
```

On peut joindre les socres des ACP aux données

```
cadastre<-bind_cols(cadastre,pca$ind$coord)
```

## 3.4 Classification ascendente hiérarchique

### 3.4.1 Calcul de la classification

On extrait les scores des trois premières composantes qu'on va utiliser dans la classification:

```
score<-pca$ind$coord[,1:3]
```

On calcul une matrice de distance euclidienne:

```
dist <- dist(score, method = "euclidean")
```

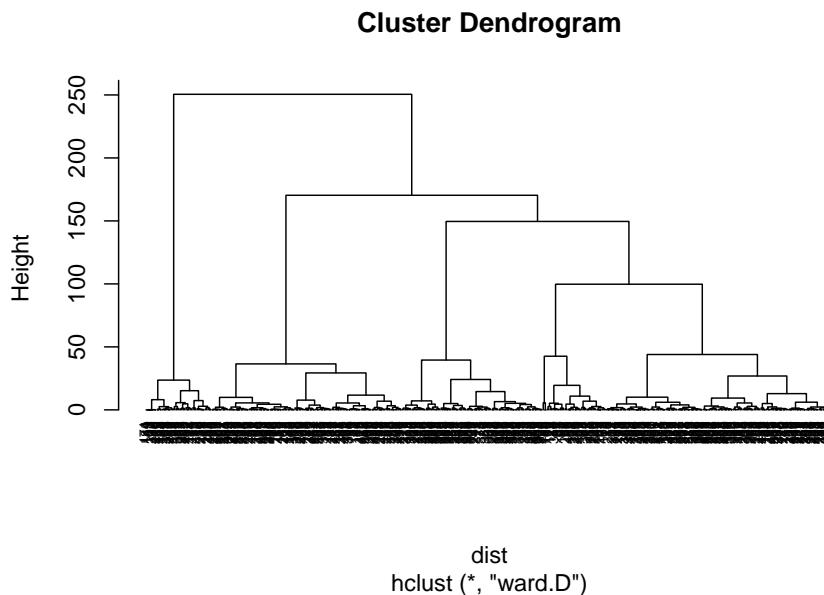
On réalise la classification avec la méthode de Ward:

```
hcpc <- hclust(dist, method = "ward.D" )
```

### 3.4.2 Choix du nombre de classe

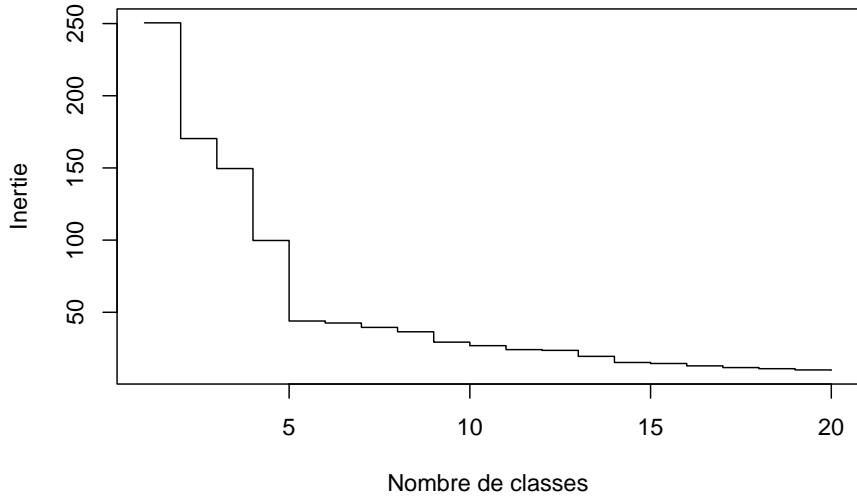
On peut réaliser le dendrogramme :

```
plot(hcpc, cex = 0.6, hang = -1)
```



Ainsi que le graphique d'inertie:

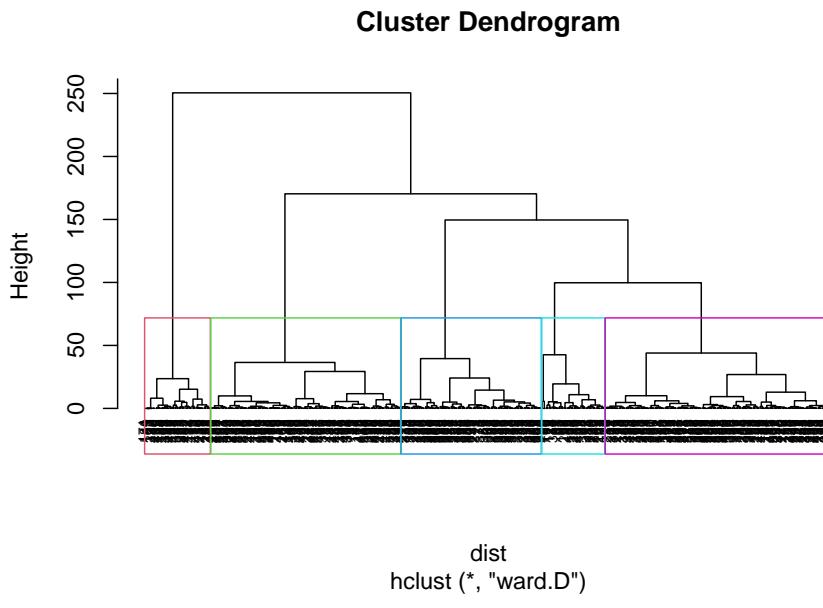
```
inertie <- sort(hcpc$height, decreasing = TRUE)
plot(inertie[1:20], type = "s", xlab = "Nombre de classes", ylab = "Inertie")
```



Il apparaît ici que le gain pour ajouter une classe supplémentaire diminue fortement après la 5ème classe

On définit donc le paramètre  $k=5$ :

```
#png(filename = "TP10/dendrogramme.png")
plot(hcpc, cex = 0.6, hang = -1)
rect.hclust(hcpc , k = 5, border = 2:6)
```



Il existe une fonction de JLutils qui permet de claculer le nombre de classes “optimale”.

```
best.cutree(hcpc)
```

```
## [1] 4
```

## 3.5 Visualisation des résultats

### 3.5.1 Extraire les groupes

On peut alors associer chaque individus (ici les secteurs statistiques) à chaque un groupe:

```
hcpc_indiv <- cutree(hcpc, k = 5)
```

On peut alors rapatrier les résultats sur les données de bases:

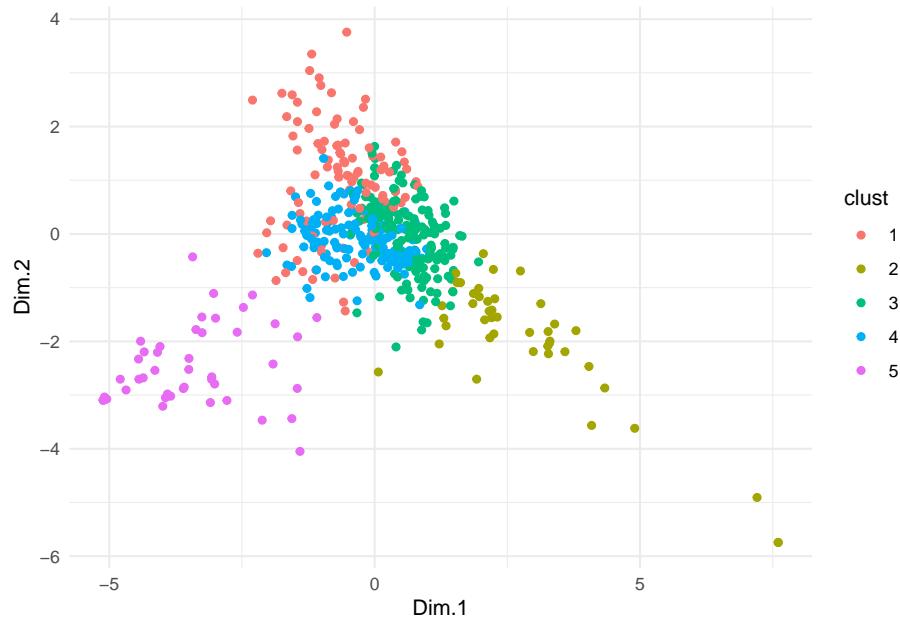
```
cadastre<-cadastre %>%
  mutate(clust= as.factor(hcpc_indiv))
```

### 3.5.2 Un graphique selon les groupes

On peut réaliser un graphique où on place chaque individus (ici les secteurs statistiques) sur les deux dimensions construites lors de l'ACP et on colorie les

points selon la classe:

```
ggplot(data=cadastre,
       aes(x=Dim.1, y=Dim.2, color=clust)) +
  geom_point() +
  theme_minimal()
```



### 3.5.3 Une carte des groupes

On importe les secteurs statistiques et garde que les secteurs bruxellois:

```
secteurs_stats<- st_read ("data/sh_statbel_statistical_sectors_31370_20230101.gpkg") %>
  filter(tx_rgn_descr_fr=="Région de Bruxelles-Capitale")

## Reading layer `secteurs_stats2023' from data source
##   `C:\Users\hugop\Nextcloud\git\book\data\sh_statbel_statistical_sectors_31370_20230101.gpkg'
##     using driver `GPKG'
##   Simple feature collection with 19795 features and 31 fields
##   Geometry type: MULTIPOLYGON
##   Dimension:      XY
##   Bounding box:  xmin: 21991.63 ymin: 21162.16 xmax: 295167.1 ymax: 244027.2
##   Projected CRS: BD72 / Belgian Lambert 72
```

On construit un objet spatial communes pour la cartographie:

```
communes<-secteurs_stats %>%
  group_by(tx_munty_descr_fr) %>%
```

```
summarise(geom=st_union(geom))
```

On joint les groupes aux secteurs statistiques:

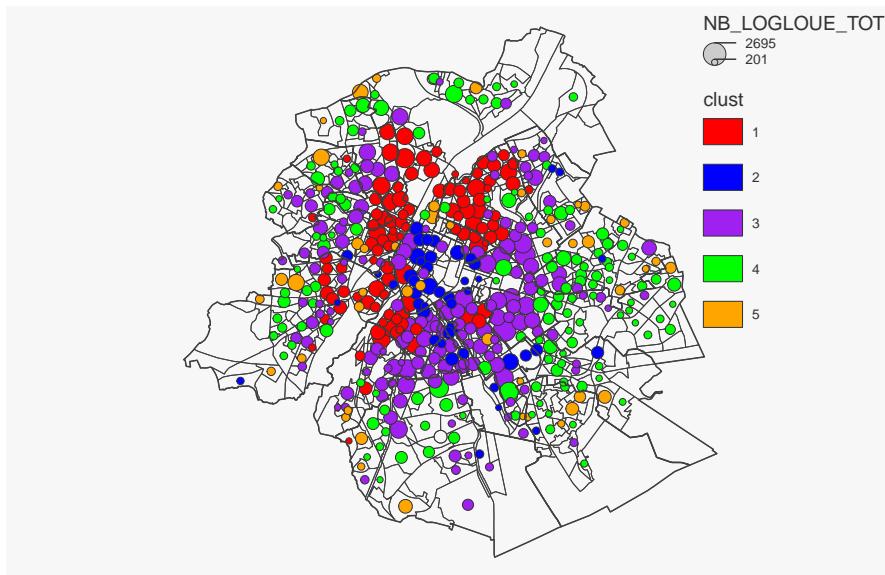
```
secteurs_stats_cad <- secteurs_stats %>%
  left_join(cadastre, by="cd_sector")
```

On réalise la carte:

```
# On peut aussi définir une palette de couleur pour faciliter la lecture
pal <- c("red", "blue", "purple", "green", "orange")

# mf_export(x = secteurs_stats_cad ,
#            filename =paste0("TP10_typo/carte_typo.png" ),
#            width = 900)
mf_map(x = secteurs_stats, col = NA, border = "gray25", lwd = 0.3)
mf_map(secteurs_stats_cad ,
       var= c("NB_LOGLOUE_TOT","clust"),
       type="prop_typo",
       inches = 0.08,
       pal= pal,
       lwd=0.3,
       add=T)

## 225 'NA' values are not plotted on the map.
mf_map(x = communes, col = NA, border = "gray25", lwd = 1, add=T)
```



#dev.off()

### 3.5.4 Un tableau de statistique pour chaque groupes

On réalise des statistiques (ici moyenne pondérée par le nombre de logements loués) par groupes:

```

tab<-cadastre %>%
  group_by(clust) %>%
  summarise(
    log_sociaux= weighted.mean(log_sociaux,NB_LOGLOUE_TOT ),
    entrep_sprl_sa= weighted.mean(entrep_sprl_sa,NB_LOGLOUE_TOT ),
    prop_LOCDOM= weighted.mean(prop_LOCDOM,NB_LOGLOUE_TOT ),
    prop_LOC_HBX = weighted.mean(prop_LOC_HBX,NB_LOGLOUE_TOT ),
    dist_200 = weighted.mean(dist_200,NB_LOGLOUE_TOT ),
    cinq_plus= weighted.mean(cinq_plus,NB_LOGLOUE_TOT ),
    vingt_plus= weighted.mean(vingt_plus,NB_LOGLOUE_TOT ) )

# On ajoute une ligne pour la moyenne:

moyenne_total<-cadastre %>%
  summarise(
    log_sociaux= weighted.mean(log_sociaux,NB_LOGLOUE_TOT ),
    entrep_sprl_sa= weighted.mean(entrep_sprl_sa,NB_LOGLOUE_TOT ),
    prop_LOCDOM= weighted.mean(prop_LOCDOM,NB_LOGLOUE_TOT ),
    prop_LOC_HBX = weighted.mean(prop_LOC_HBX,NB_LOGLOUE_TOT ),
    dist_200 = weighted.mean(dist_200,NB_LOGLOUE_TOT ),
    cinq_plus= weighted.mean(cinq_plus,NB_LOGLOUE_TOT ),
    vingt_plus= weighted.mean(vingt_plus,NB_LOGLOUE_TOT ) )

# On place le résultat dans la ligne donc le numéro est équivalent au nombre de ligne
tab[nrow(tab)+1 , 2:8]<-moyenne_total

#On multiplie toutes les cellules par 100 et on arrondi avec un 1 chiffre après la virgule
tab[,2:8]<-round(100*tab[,2:8],1)

```

On peut alors donner des noms aux groupes

```
tab$clust<-as.character(tab$clust)
tab<- tab %>%
  mutate(clust=case_when (
    clust ==1~"1. Propriétaires à proximité",
    clust ==2~"2. Grands propriétaires et entreprises",
    clust ==3~"3. Type moyen",
    clust ==4~"4. Propriétaires non-Bruxellois",
```

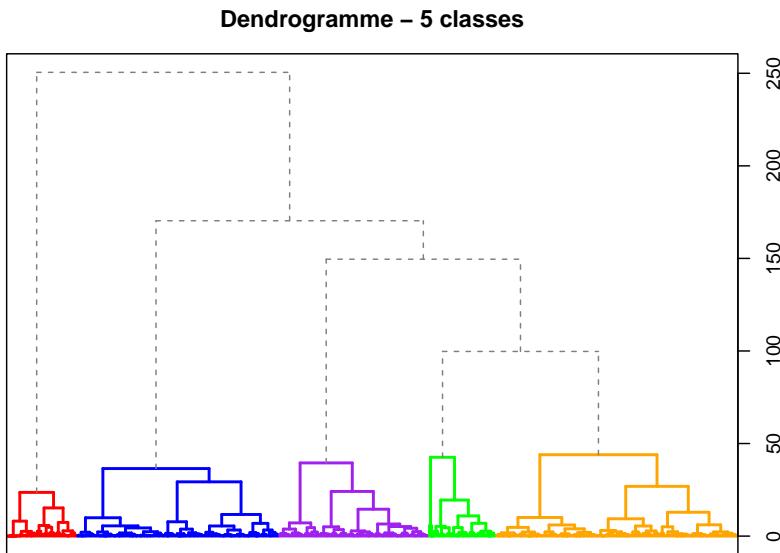
```
clust ==5~"5. Logements sociaux",
is.na(clust)~"Moyenne"))

kable(tab)
```

clust	log_sociaux	entrep_sprl_sa	prop_LOCDOM	prop_LOC_HBX	...
1. Propriétaires à proximité	10.2	6.7	27.6	19.8	
2. Grands propriétaires et entreprises	3.4	26.0	5.8	26.4	
3. Type moyen	2.8	13.1	14.0	32.1	
4. Propriétaires non-Bruxellois	3.1	10.4	9.6	39.2	
5. Logements sociaux	78.0	1.9	2.4	7.1	
Moyenne	10.4	11.1	15.0	28.1	

Le package JLutils permet également de colorer le dendrogramme:

```
# library(JLutils)
A2Rplot(hcpc, k =5, boxes = T, show.labels = F,col.up = "gray50",
        col.down = pal, main="Dendrogramme - 5 classes")
```



### 3.5.5 Comparaison avec une autre variable

On importe l'indice synthétique de difficulté:

```
ind_synth <-read_delim("data/indice_synthetique.csv", delim=";")
```

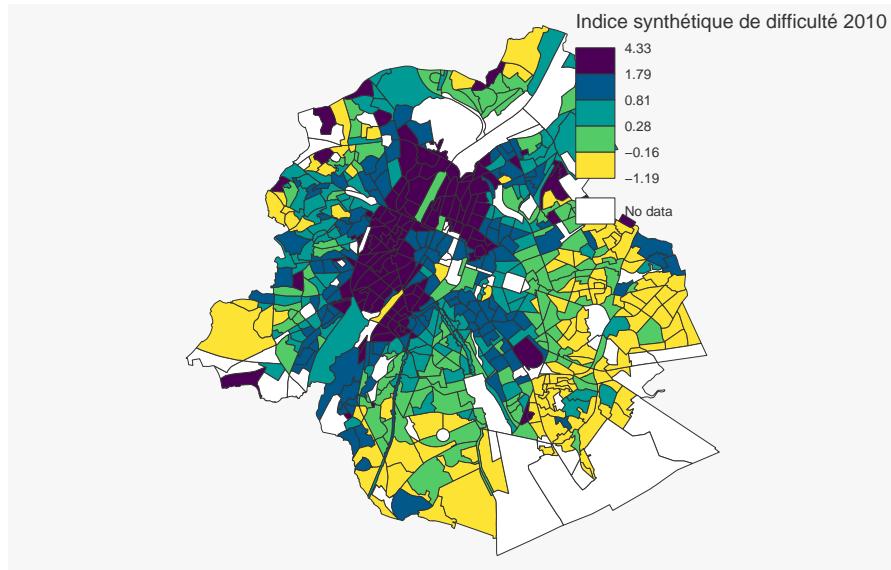
```
## Rows: 7752 Columns: 2
```

```
## -- Column specification -----
## Delimiter: ";"
```

## chr (1): Secteur statistique  
## dbl (1): Indice synthétique de difficulté 2010  
##  
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

```
indice_sec<-secteurs_stats %>%
  left_join(ind_synth, by=c("cd_sector"="Secteur statistique")) %>%
  filter(tx_rgn_descr_fr=="Région de Bruxelles-Capitale")
```

```
mf_map(indice_sec,
       var="Indice synthétique de difficulté 2010",
       type="choro",
       pal= "Viridis",
       nbreaks=5)
```

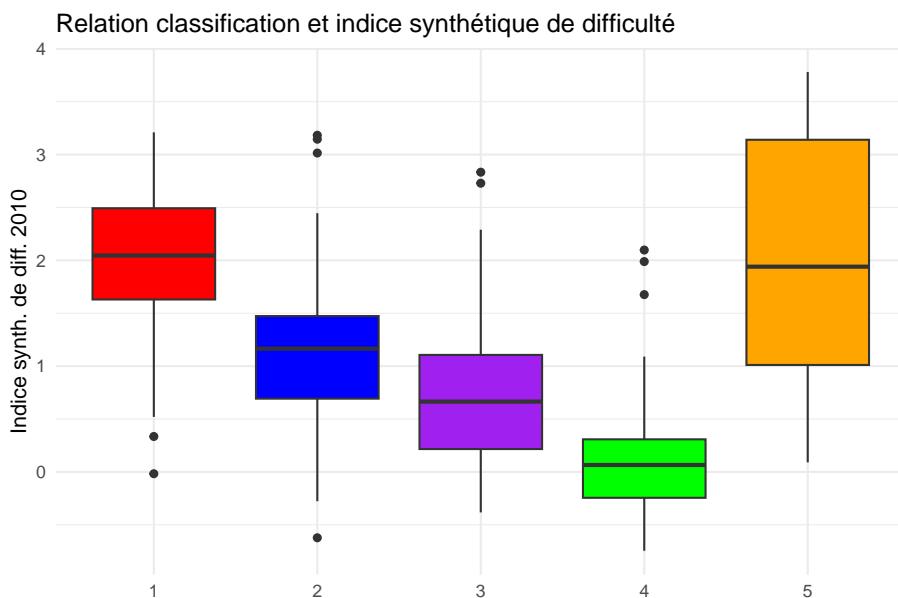


On joint l'indice synthétique de difficulté avec les données sur base du secteur statistique:

```
cadastre<- cadastre %>%
  left_join(ind_synth,by= c("cd_sector"= "Secteur statistique"))
```

On réalise un graphique avec des boxplot:

```
ggplot(data=cadastre,aes(x=clust,
                           y="Indice synthétique de difficulté 2010",
                           weight=NB_LOGLOUE_TOT,
                           fill=clust)) +
  geom_boxplot(fill=pal) +
  labs(title ="Relation classification et indice synthétique de difficulté",
       x= " ",
       y= "Indice synth. de diff. 2010")+
  theme_minimal()
```



On peut en conclure qu'il y a un lien entre la division sociale de l'espace et la structure de la propriété des logements loués. Les secteurs du groupe 1 sont plus en difficulté que les autres secteurs, surtout les 3 et 4. Le secteurs centraux (2) ont des indices intermédiaires et certains secteurs de logements sociaux (5) sont très mixtes d'où leur forte dispersion.

## 3.6 Ressources utilisées

- <https://larmarange.github.io/analyse-R/classification-ascendante-hierarchique.html>
- <http://factominer.free.fr/factomethods/classification-hierarchique-sur-composantes-principales.html>
- <http://www.sthda.com/english/articles/31-principal-component-analysis-PCA-in-R/>

[methods-in-r-practical-guide/117-hcpc-hierarchical-clustering-on-principal-components-essentials/](#)