The French Ministry for the Ecological and Inclusive Transition (for which I’m currently working) is ongoing a process of opening data related to energy consumption. Each year, we publish data for every neighborhood in France (at the [iris](https://www.insee.fr/en/metadonnees/definition/c1523) statistical level, even adresses in some cases) and to the nature of the final consumer (a household, an industry, a shop…). These data are available [here](https://www.statistiques.developpement-durable.gouv.fr/donnees-locales-de-consommation-denergie) (website in French – [direct link to 2018 electricity consumption data](https://www.statistiques.developpement-durable.gouv.fr/sites/default/files/2020-05/donnees-electricite-2018.zip)).

Making a map to have an overlook at the situation isn’t easy, because the administrative boundaries of France are very diverse, and a direct mapping will reflect this situation. In order to overcome this issue, a solution is to use a different way to represent the country. In this post, we’re going to talk about hex map

What’s a hex map? First, we need to talk about [chloropleth maps](https://en.wikipedia.org/wiki/Choropleth_map); despite their unusual names, they’re perfectly common maps where each area is shaded or colored differently according to the values taking by a variable. They’re quite similar to maps [such as this one](https://www.nytimes.com/interactive/2019/12/13/world/europe/uk-general-election-results.html) showing which political party won an election on each voting district, but for continuous variable, and required a scale. To determine the color used for each area, we need to compute the average (or any statistical summary) of the values of every individual belonging in this area.

Hex map are entirely composed of hexagons ([something perfectly suited for France!](https://french.stackexchange.com/questions/28886/using-hexagone-to-describe-france)); they’re chloropleth maps, so each hexagon is colored depending on the values taking by a variable. The main difference is that the hexagon aren’t usual geographical areas so we need a way to attribute each value in our data set to one of the hexagons before calculating averages, and therefore colors.

In order to do that, we’re going to load some usual R-packages:

library(tidyverse)

library(viridis)

library(ggplot2)

library(stringr)

Our objective will be to make an hex map of residential consumption of electricity in 2018. First, we need to import the data at iris level (doing this, we miss some of the electricity consumption only associated to larger geographical area, but it’s marginal); as the data is coded as strings with some missing values, we start by cleaning the data.

elec <- read.csv2("donnees\_elec\_iris\_2018.csv",stringsAsFactors = F)

elec <- elec[elec$CONSO != "s" & elec$PDL != "s",] #"s" are missing values

# CONSO (electricity consumption) is coded with a french comma separator ; we need to substitute that and convert to numeric CONSO and PDL (number of energy delivery points)

elec$CONSO <- str\_replace(elec$CONSO,",",".")

elec$CONSO <- as.numeric(elec$CONSO)

elec$PDL <- as.numeric(elec$PDL)

# Select only residential sector

res <- elec[elec$CODE\_GRAND\_SECTEUR == "R",]

Each data point is then associated to the coordinates of the center of the neighborhood, using a iris shapefile (available [here](https://www.data.gouv.fr/fr/datasets/contours-iris/)). The *res* dataset obtained looks like this:

CODE\_IRIS CONSO PDL x y

100020000 843.8162 124 48.25 4.69

100030000 10374.7166 1987 48.24 3.72

100040000 679.8085 88 48.59 4.12

100050000 769.8268 145 48.29 4.49

100060000 7318.2055 1534 48.53 4.14

100070000 429.9674 61 48.16 4.73

100080000 343.1404 62 48.25 4.60

100090000 262.1539 55 48.05 4.28

100100000 140.5366 28 48.54 4.61

100110000 700.1244 113 48.27 4.74

We’re using a [ggplot2](https://ggplot2.tidyverse.org/) environment to create our hex map. The aesthetics used are *aes(x,y)*, the spatial coordinates. We begin by these line of code in order to use a simpler theme (as we’re creating a map, we don’t need any axis) and to specify the positions (longitude and latitude) we want to map.

c <- ggplot(res, aes(x, y))

c +

xlim(-9, 12) +

ylim(40, 52) +

theme\_void()

The main argument of the [*stat\_summary\_hex* function](https://www.rdocumentation.org/packages/ggplot2/versions/1.0.1/topics/stat_summary_hex?tap_a=5644-dce66f&tap_s=10907-287229) we’re going to use is *bins*. This parameter allows us to choose how many hexagons will be displayed on the map; the more hexagon, the more small local variations are shown. For the other parameters, we specify *aes(z = CONSO)* as we want to color the hexagons in relation to the *CONSO* (Consumption in french) of the data points contained in this area. The *fun = “sum”* parameter means that the choice of the color of the hexagon depends on the total consumption inside. Lastly, we add a *colour=”grey”* parameter, which defines the color of every hexagon’s borders, because using white borders on my computer leads to some graphical artefacts .

c <- ggplot(res, aes(x, y))

c + stat\_summary\_hex(bins=40,aes(z = CONSO),fun = "sum", colour="grey") +

xlim(-9, 12) +

ylim(40, 52) +

theme\_void()

We then define a scale (both numeric and in terms of colors) using the [*scale\_fill\_viridis* function](https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html). In order to better see large differences, we use the *trans=”log”* option which means that transitions between colors are logarithmic and not arithmetic progressions. The breaks points used are 1 GWh, 10 GWh, 100 GWh and 1 TWh.

c <- ggplot(res, aes(x, y))

c + stat\_summary\_hex(bins=40,aes(z = CONSO),fun = "sum", colour="grey") +

xlim(-9, 12) +

ylim(40, 52) +

theme\_void() +

scale\_fill\_viridis(

option="A",

trans = "log",

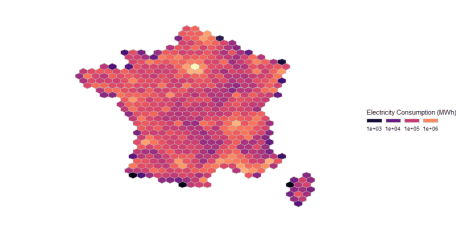
breaks = c(0,1000,10000,100000,1000000,99999999999),

name="Electricity Consumption",

guide = guide\_legend(keyheight = unit(2.5, units = "mm"), keywidth=unit(4, units = "mm"),

label.position = "bottom",

title.position = 'top', nrow=1) )



Using the following code with *bins = 80* results in this another map:

c <- ggplot(res, aes(x, y))

c + stat\_summary\_hex(bins=80,aes(z = CONSO),fun = "sum", colour="grey") +

xlim(-9, 12) +

ylim(40, 52) +

theme\_void() +

scale\_fill\_viridis(

option="A",

trans = "log",

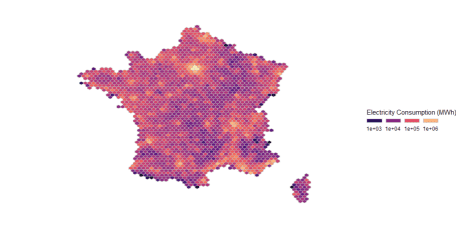
breaks = c(0,1000,10000,100000,1000000,99999999999),

name="Electricity Consumption",

guide = guide\_legend(keyheight = unit(2.5, units = "mm"), keywidth=unit(4, units = "mm"),

label.position = "bottom",

title.position = 'top', nrow=1) )



These map highlight the position of urban centers with high population density. A more useful information is the total electricity consumption per household (or, more accurately in this case, per energy delivery point, whose quantity per iris is *PDL* in the dataset), defined this way:

res$ElectricityPerHousehold <- res$CONSO/res$PDL

The code used to generate the map is slighty similar; the stat\_summary\_hex parameter are modified to *aes(z = ElectricityPerHousehold)* and *fun=”mean”*, and, *scale\_fill\_viridis* parameters are switched to *option=”E”* (leading to a new color scheme), new breakpoints of 2, 4, 6 and 8 GWh per household, and a new legend name.

c <- ggplot(res, aes(x, y))

c + stat\_summary\_hex(bins=80,aes(z = ElectricityPerHousehold),fun = "mean", colour="grey") +

xlim(-9, 12) +

ylim(40, 52) +

theme\_void() +

scale\_fill\_viridis(

option="E",

trans = "log",

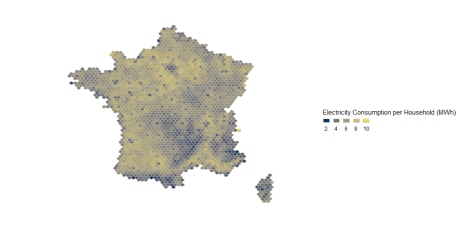
breaks = c(0,2,4,6,8,10),

name="Electricity Consumption per Household",

guide = guide\_legend(keyheight = unit(2.5, units = "mm"), keywidth=unit(4, units = "mm"),

label.position = "bottom",

title.position = 'top', nrow=1) )



This map better hightlights places with large electricity consumptions; Paris and its suburbs are associated to lower consumptions per household, which contrasts with an unusually high total consumption, due to high population density. An high consumption per household is usually associated to electric heating and/or large houses, and parisian dwellings are mostly smaller flats.