Representing mass data (inhabitants, livestock,…) on a map in not always easy : choropleth maps are clearly a no-go, except if you normalize with area and then you stumble on the [MAUP](https://en.wikipedia.org/wiki/Modifiable_areal_unit_problem)… A nice solution is smoothing, producing a raster. You even get freebies like (potential) statistical confidentiality, a better geographic synthesis and easy multiple layers computations.

The kernel smoothing should not be confused with interpolation or [kriging](https://en.wikipedia.org/wiki/Kriging) : the aim here is to « spread » and sum point values.

We’ll use the btb package (Santos *et al*. [2018](http://r.iresmi.net/2019/05/11/kernel-spatial-smoothing-transforming-points-pattern-to-continuous-coverage/#santos2018)) which has the great advantage of providing a way to specify a geographical study zone, avoiding our values to bleed in another country or in the sea for example.

We’ll map the french population :

* the data is available on the [IGN](http://www.professionnels.ign.fr/adminexpress) site
* a 7z decompress utility must be available in your $PATH ;
* the shapefile COMMUNE.shp has a POPULATION field ;
* this is a polygon coverage, so we’ll take the « centroids ».

library(raster) # load before dplyr (against select conflict)

library(tidyverse)

library(httr)

library(sf)

library(btb)

# download and extract data

zipped\_file <- tempfile()

GET("ftp://Admin\_Express\_ext:Dahnoh0eigheeFok@ftp3.ign.fr/ADMIN-EXPRESS\_2-0\_\_SHP\_\_FRA\_2019-03-14.7z.001",

write\_disk(zipped\_file),

progress())

system(paste("7z x -odata", zipped\_file))

# create a unique polygon for France (our study zone)

fr <- read\_sf("data/ADMIN-EXPRESS\_2-0\_\_SHP\_\_FRA\_2019-03-14/ADMIN-EXPRESS/1\_DONNEES\_LIVRAISON\_2019-03-14/ADE\_2-0\_SHP\_LAMB93\_FR/REGION.shp") %>%

st\_union() %>%

st\_sf() %>%

st\_set\_crs(2154)

# load communes ; convert to points

comm <- read\_sf("data/ADMIN-EXPRESS\_2-0\_\_SHP\_\_FRA\_2019-03-14/ADMIN-EXPRESS/1\_DONNEES\_LIVRAISON\_2019-03-14/ADE\_2-0\_SHP\_LAMB93\_FR/COMMUNE.shp")%>%

st\_set\_crs(2154) %>%

st\_point\_on\_surface()

We create a function :

#' Kernel weighted smoothing with arbitrary bounding area

#'

#' @param df sf object (points)

#' @param field weigth field in sf

#' @param bandwith kernel bandwidth (map units)

#' @param resolution output grid resolution (map units)

#' @param zone sf study zone (polygon)

#' @param out\_crs EPSG (should be an equal-area projection)

#'

#' @return a raster object

#' @import btb, raster, fasterize, dplyr, plyr, sf

lissage <- function(df, field, bandwidth, resolution, zone, out\_crs = 3035) {

if (st\_crs(zone)$epsg != out\_crs) {

message("reprojecting data...")

zone <- st\_transform(zone, out\_crs)

}

if (st\_crs(df)$epsg != out\_crs) {

message("reprojecting study zone...")

df <- st\_transform(df, out\_crs)

}

zone\_bbox <- st\_bbox(zone)

# grid generation

message("generating reference grid...")

zone\_xy <- zone %>%

dplyr::select(geometry) %>%

st\_make\_grid(cellsize = resolution,

offset = c(plyr::round\_any(zone\_bbox[1] - bandwidth, resolution, f = floor),

plyr::round\_any(zone\_bbox[2] - bandwidth, resolution, f = floor)),

what = "centers") %>%

st\_sf() %>%

st\_join(zone, join = st\_intersects, left = FALSE) %>%

st\_coordinates() %>%

as\_tibble() %>%

dplyr::select(x = X, y = Y)

# kernel

message("computing kernel...")

kernel <- df %>%

cbind(., st\_coordinates(.)) %>%

st\_set\_geometry(NULL) %>%

dplyr::select(x = X, y = Y, field) %>%

btb::kernelSmoothing(dfObservations = .,

sEPSG = out\_crs,

iCellSize = resolution,

iBandwidth = bandwidth,

vQuantiles = NULL,

dfCentroids = zone\_xy)

# rasterization

message("\nrasterizing...")

raster::raster(xmn = plyr::round\_any(zone\_bbox[1] - bandwidth, resolution, f = floor),

ymn = plyr::round\_any(zone\_bbox[2] - bandwidth, resolution, f = floor),

xmx = plyr::round\_any(zone\_bbox[3] + bandwidth, resolution, f = ceiling),

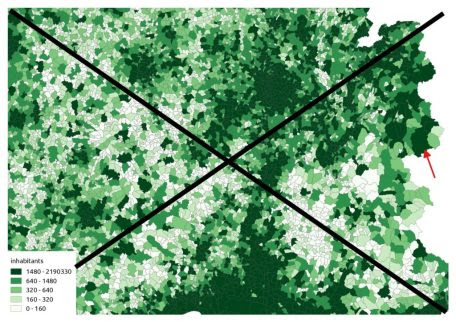
ymx = plyr::round\_any(zone\_bbox[4] + bandwidth, resolution, f = ceiling),

resolution = resolution) %>%

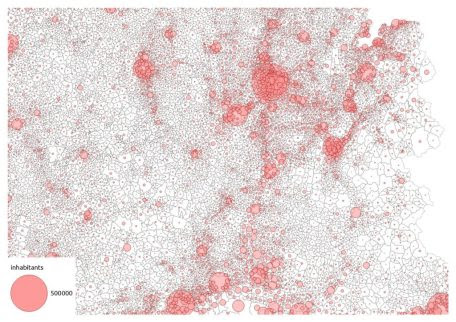
fasterize::fasterize(kernel, ., field = field)

}

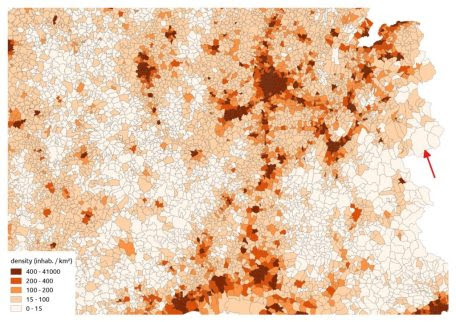
Instead of a raw choropleth map like this (don’t do this at home) :

[](https://i2.wp.com/r.iresmi.net/wp-content/uploads/2019/05/inhabitants.jpeg)Inhabitants, quantile classification ; see the red arrow : a big commune with a somewhat low population (2100 inhabitants) pops out due to its big area

… we should use a proportional symbol. But it’s quite cluttered in urban areas :

[](https://i2.wp.com/r.iresmi.net/wp-content/uploads/2019/05/points-1.jpeg)You’ve got measles

We can also get the polygon density :

[](https://i0.wp.com/r.iresmi.net/wp-content/uploads/2019/05/density.jpeg)Density, quantile classification. Our previous commune is now more coherent, however the map is not very synthetic due to the heterogeneous size of the communes

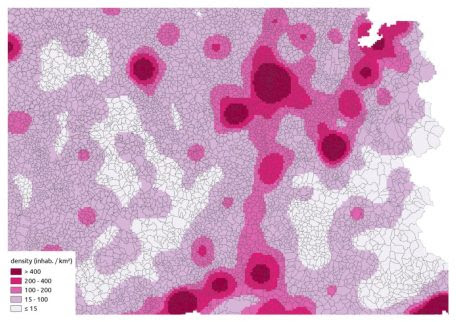
We just have to run our function for example with a bandwidth of 20 km and a cell size of 2 km. We use an equal area projection : the LAEA Europa (EPSG:3035).

comm %>%

lissage("POPULATION", 20000, 2000, fr, 3035) %>%

raster::writeRaster("pop.tif")

And lastly our smoothing :

[](https://i2.wp.com/r.iresmi.net/wp-content/uploads/2019/05/smoothing.jpeg)Smoothing, discrete quantile classification

That’s a nice synthetic representation !

After that it’s easy in R to do raster algebra ; for example dividing a grid of crop yields by a grid of agricultural area, create a percent change between dates, etc.