



# Thematic Maps with cartography

Timothée Giraud



useR! 2019 - Toulouse - 2019/07/09

# Produce publication ready thematic maps



# Standing on the Shoulders of Giants

Initially based on **sp** & **rgeos**...

... and updated to **sf** with version 2.0.0 (Sep, 2017)



# Main Features: Symbologies



Choropleth  
`choroLayer(x = mtq, var = "myvar",  
method = "quantile", nclass = 8)`



Typology  
`typoLayer(x = mtq, var = "myvar")`



Proportional Symbols  
`propSymbolsLayer(x = mtq, var = "myvar",  
inches = 0.1, symbols = "circle")`



Colorized Proportional Symbols (relative data)  
`propSymbolsChoroLayer(x = mtq, var = "myvar",  
var2 = "myvar2")`



`propSymbolsTypoLayer(x = mtq, var = "myvar",  
var2 = "myvar2")`



Double Proportional Symbols  
`propTrianglesLayer(x = mtq, var1 = "myvar",  
var2 = "myvar2")`



OpenStreetMap Basemap (see rosm package)  
`tiles <- getTiles(x = mtq, type = "osm")  
tilesLayer(tiles)`



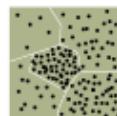
Isopleth (see SpatialPosition package)  
`smoothLayer(x = mtq, var = "myvar",  
typefct = "exponential", span = 500,  
beta = 2)`



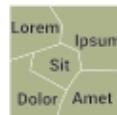
Discontinuities  
`discLayer(x = mtq.borders, df = mtq,  
var = "myvar", threshold = 0.5)`



Flows  
`propLinkLayer(x = mtq_link, df = mtq_df,  
var = "fij")`

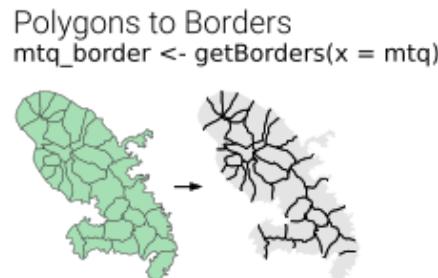
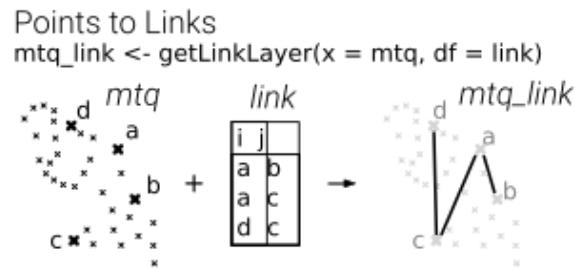


Dot Density  
`dotDensityLayer(x = mtq, var = "myvar")`

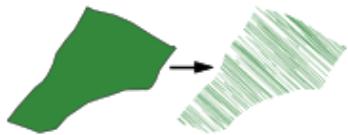


Labels  
`labelLayer(x = mtq, txt = "myvar",  
halo = TRUE, overlap = FALSE)`

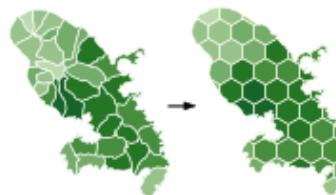
# Main Features: Transformations



Polygons to Pencil Lines  
`mtq_pen <- getPencilLayer(x = mtq)`

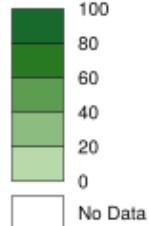


Polygons to Grid  
`mtq_grid <- getGridLayer(x = mtq, cellsize = 3.6e+07, type = "hexagonal", var = "myvar")`



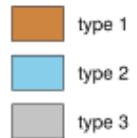
# Main Features: Map Layout

legendChoro()



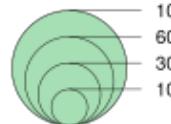
```
100 legendChoro(pos = "topleft",
80   title.txt = "legendChoro()",  
60   breaks = c(0,20,40,60,80,100),  
40   col = carto.pal("green.pal", 5),  
20   nodata = TRUE, nodata.txt = "No Data")  
0
```

legendTypo()



```
legendTypo(title.txt = "legendTypo()",  
           col = c("peru", "skyblue", "gray77"),  
           categ = c("type 1", "type 2", "type 3"),  
           nodata = FALSE)
```

legendCirclesSymbols()

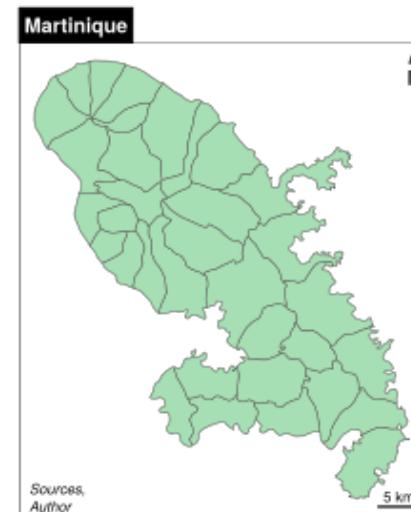


```
100 legendCirclesSymbols(var = c(10,100),
60   title.txt = "legendCirclesSymbols()",  
30   col = "#a7dfb4ff", inches = 0.3)  
10
```

North Arrow:  
north(pos = "topright")

Scale Bar:  
barscale(size = 5)

Full Layout:  
layoutLayer(  
 title = "Martinique",  
 tabtitle = TRUE,  
 frame = TRUE,  
 author = "Author",  
 sources = "Sources",  
 north = TRUE,  
 scale = 5)

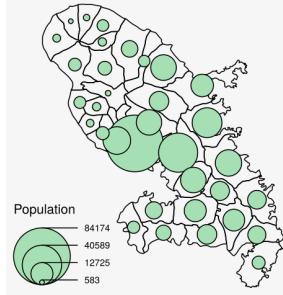


# Cheat Sheet

## Thematic maps with cartography :: CHEAT SHEET

Use cartography with spatial objects from sf or sp packages to create thematic maps.

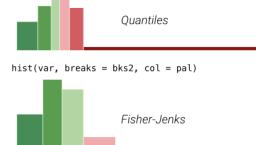
```
library(cartography)
library(sf)
mtq <- st_read("martinique.shp")
plot(st_geometry(mtq))
propSymbolLayer(x = mtq, var = "P13_POP",
legend.title.txt = "Population",
col = "#a7dfbd")
```



### Classification

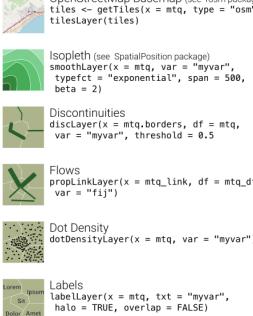
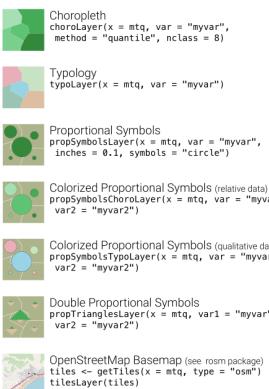
Available methods are: quantile, equal, q5, fisher-jenks, mean-sd, geometric progression...

```
bks1 <- getBreaks(v = var, nclass = 6,
method = "quantile")
bks2 <- getBreaks(v = var, nclass = 6,
method = "fisher-jenks")
pal <- carto.pal("green.pal",3,"wine.pal", 3)
hist(var, breaks = bks1, col = pal)
```



### Symbology

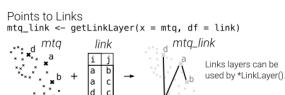
In most functions the x argument should be an sf object. sp objects are handled through spdf and df arguments.



### Transformations

```
mtq_grid <- getGridLayer(x = mtq, cellsize = 3.6e+07,
type = "hexagonal", var = "myvar")
```

Grids layers can be used by chorolayer() or propSymbolsLayer().



Points to Links

```
mtq_link <- getLinkLayer(x = mtq, df = link)
```

Links layers can be used by \*Linklayer()



Polylines to Borders

```
mtq_border <- getBorders(x = mtq)
```

Borders layers can be used by discLayer() function



Polylines to Pencil Lines

```
mtq_pen <- getPencilLayer(x = mtq)
```

Pencil lines layers can be used by discLayer() function

### Legends

```
legendChoro()
```

100
80
60
40
20
0
No Data

```
legendChoro(pgs = "topleft",
title.txt = "legendChoro()",
```

```
breaks = c(0, 40, 80, 120),
col = carto.pal("green.pal", 5),
nodata = TRUE, nodata.txt = "No Data")
```

```
legendType()
```

```
LegendType(title.txt = "legendType()",
```

```
col = c("peru", "skyblue", "gray77"),
```

```
category = c("type 1", "type 2", "type 3", "type 4"),
```

```
nodata = FALSE)
```

```
legendCirclesSymbols()
```

```
LegendCirclesSymbols(var = c(10, 100),
```

```
title.txt = "legendCirclesSymbols()",
```

```
col = "#a7dfbd", inches = 0.3)
```

See also: legendSquaresSymbols(), legendBarsSymbols(),

legendGradLines(), legendPropLines() and legendPropTriangles()

### Map Layout

North Arrow:

```
north$pos = "topright")
```

Scale Bar:

```
bar$scale(size = 5)
```

Full Layout:

```
layoutLayer(
```

```
title = "Martinique",
```

```
title = TRUE,
```

```
frame = TRUE,
```

```
author = "Author",
```

```
sources = "Sources",
```

```
north = TRUE,
```

```
scale = 5)
```

Figure Dimensions

Get figure dimensions based on the dimension ratio of a spatial object, figure margins and output resolution.

```
f.dim <- getFigDim(x = sf_obj, width = 500,
```

```
mar = c(0,0,0,0),
```

```
plot.dim$width = 500, height = f.dim[2])
```

```
par(mar = c(0,0,0,0))
```

```
plot(sf_obj, col = "#729fcf")
```

```
dev.off()
```

default

```
a / b == f.dim[1] / f.dim[2]
```

controlled ratio

```
a / b == f.dim[2] / f.dim[1]
```

### Color Palettes

```
carto.pal(pal1 = "blue.pal", n1 = 5,
pal2 = sand.pal, n2 = 3)
```

display.carto.all(n = 8)

blue.pal

red.pal

green.pal

pink.pal

grey.pal

sand.pal

kaki.pal

pastel.pal

orange.pal

brown.pal

purple.pal

wine.pal

turquoise.pal

taupe.pal

harmo.pal

multi.pal

cartography - <https://github.com/niatlab/cartography> - CC BY SA Timothée Giraud - 2019-01

# Website

cartography 2.2.1 [Home](#) Get started Reference Articles Changelog [Feedback](#)

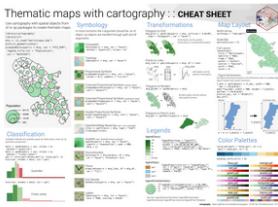
## cartography

Create and integrate maps in your R workflow!

This package helps to design **cartographic representations** such as proportional symbols, choropleth, typology, flows or discontinuities maps. It also offers several features that improve the graphic presentation of maps, for instance, map palettes, layout elements (scale, north arrow, title...), labels or legends.

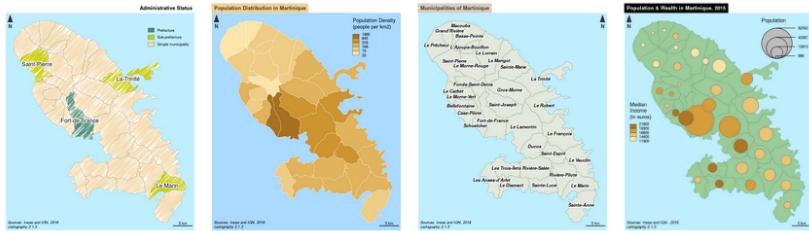
### Cheat Sheet

The [cheat sheet](#) displays a quick overview of `cartography`'s main features.



### Vignette

The [vignette](#) contains commented scripts on how to build various types of maps with `cartography`:



**Links**

Download from CRAN at [https://CRAN.R-project.org/  
package=cartography](https://CRAN.R-project.org/package=cartography)

Browse source code at <https://github.com/rlatlab/cartography>

Report a bug at [https://github.com/rlatlab/cartography/  
Issues/](https://github.com/rlatlab/cartography/issues/)

**License**

GPL-3

**Citation**

[Citing cartography](#)

**Developers**

Timothée Giraud  
Maintainer, author

Nicolas Lambert  
Author

[All authors...](#)

**Dev status**

[CRAN 2.2.0](#)  
[downloads 1603/month](#)  
[build passing](#)  
 [codecov 99%](#)  
[JOSS 10.21105/joss.00054](#)

# Vignette

## 3.10 Discontinuities Map

Discontinuity maps are based on the variation of a phenomena between representation focuses spatial breaks. The discontinuity intensity is expressed by colors. The `getdiscontinuity()` is used to build a spatial object of breaks between units.

```
library(sf)
library(cartography)
# path to the shapefile
path_to_gpkg <- system.file("shapefiles/Discontinuity", package = "cartography")
path_to_gpkg <- file.path(path_to_gpkg, "discontinuity.gpkg")
# read the shapefile
stg <- st_read(path_to_gpkg)
```

If the two neighboring units, it is considered borders between non-contiguous areas with the reflect the neighboring units.

```
# Import the population data
stgPOPDIS <- st_as_sf(stg)
# Get a SpatialPolygonsDataFrame
stg,coords <- geometry(stg)
stg$area <- area(stg)
plot(st_geometry(stg), col = "#1f77b4", lwd = 1)
# Plot the population density
choropleth <- stg
stg$pop <- stg$area * stg$pop
stg$pop <- stg$pop / 1000000
plot(st_geometry(stg), col = "#1f77b4", lwd = 1)
# Plot the population density choropleth
x <- stg
var <- "POP"
inches <- 0.4
col <- "#1f77b4"
legend.title <- "Population"
legend.title.txt <- "Population"
method <- "mean"
relax <- 2,
threshold <- 0.4,
minsize <- 0.7,
sizecols <- 6,
col <- "#1f77b4",
legend.values <- 1,
legend.title.txt <- "Relativ"
legend.title.col <- "right",
add <- TRUE
}
# Layout
layeratlas[title = "Health I
author = "pastel0"
sources = "Source
frame = FALSE, at
# north arrow
north[pos = "topleft"]]
```

The `getdiscontinuity()` and `tileraster()` download and display OpenStreetMap files. Be careful to cite the source of the tile appropriately.

```
library(cartography)
# path to the shapefile
path_to_gpkg <- system.file("shapefiles/Discontinuity", package = "cartography")
# read the shapefile
stg <- st_read(path_to_gpkg)
```

**3.2 Choropleth Map**

This figure displays three choropleth maps of Martinique showing population distribution. The first map uses a single color (#1f77b4) for the entire island. The second map shows a more detailed distribution with varying shades of green and yellow. The third map highlights discontinuities with a grid overlay, showing where population density changes abruptly between adjacent administrative units. All maps include a north arrow and a scale bar.

## 3.1 OpenStreetMap Basemap and Proportional Symbols

OpenStreetMap basemaps and proportional symbols are available with `getosmLayer()` and `tileraster()`.

```
library(sf)
library(cartography)
# path to the shapefile
path_to_gpkg <- system.file("shapefiles/Population", package = "cartography")
# read the shapefile
stg <- st_read(path_to_gpkg)
```

**3.2 Grid Map**

This figure shows a choropleth map of Martinique's population distribution with a grid overlay. The grid highlights areas where the population density varies significantly between adjacent administrative units. The map includes a north arrow and a scale bar.

## 3.6 Label Map

Labels layer is an option to highlights the main trends in the data values are distributed over a regular grid. It is possible to set the share of intersected areas.

```
library(sf)
library(cartography)
# path to the shapefile
path_to_gpkg <- system.file("shapefiles/Population", package = "cartography")
# read the shapefile
stg <- st_read(path_to_gpkg)
```

**3.7 Links Map**

This figure shows a choropleth map of Martinique's population distribution with a grid overlay and labels. The labels indicate the percentage of the total area that intersects with each grid cell. The map includes a north arrow and a scale bar.

## 3.5 Isopleth Map

`getisoplethLayer()` creates a link layer from an sf object and a link data.

```
library(sf)
library(cartography)
# path to the shapefile
path_to_gpkg <- system.file("shapefiles/Population", package = "cartography")
# read the shapefile
stg <- st_read(path_to_gpkg)
```

**3.8 Isopleth Map**

This figure shows a choropleth map of Martinique's population distribution with an isopleth overlay. The isopleth highlights specific regions where the population density reaches a certain threshold. The map includes a north arrow and a scale bar.

## 3.3 Colored Pencil and Typologies Map

`getpencilLayer()` creates a link layer that mimicks a pencil hand-drawing.

```
library(sf)
library(cartography)
# path to the shapefile
path_to_gpkg <- system.file("shapefiles/Population", package = "cartography")
# read the shapefile
stg <- st_read(path_to_gpkg)
```

**3.4 Proportional Symbols**

This figure shows a choropleth map of Martinique's population distribution with a colored pencil overlay. The overlay mimics a hand-drawn style, with colors and patterns applied to different regions. The map includes a north arrow and a scale bar.

9/22

# Alternative Solutions

`ggplot2` (Wickham, 2016) + `ggspatial` (Dunnington, 2018)

- A general purpose graphic library

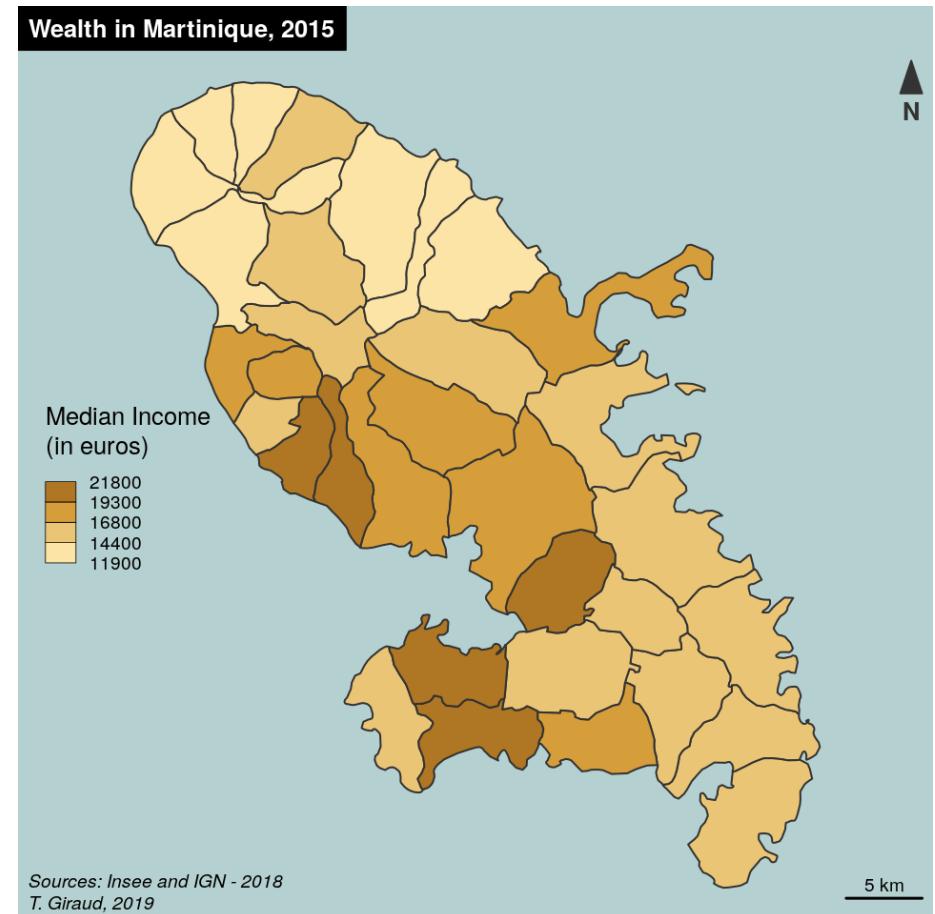
`tmap` (Tennekes, 2018)

- A mapping library with similar functionalities
- Uses a different grammar (*à la* `ggplot2`)
- Allows interactive maps

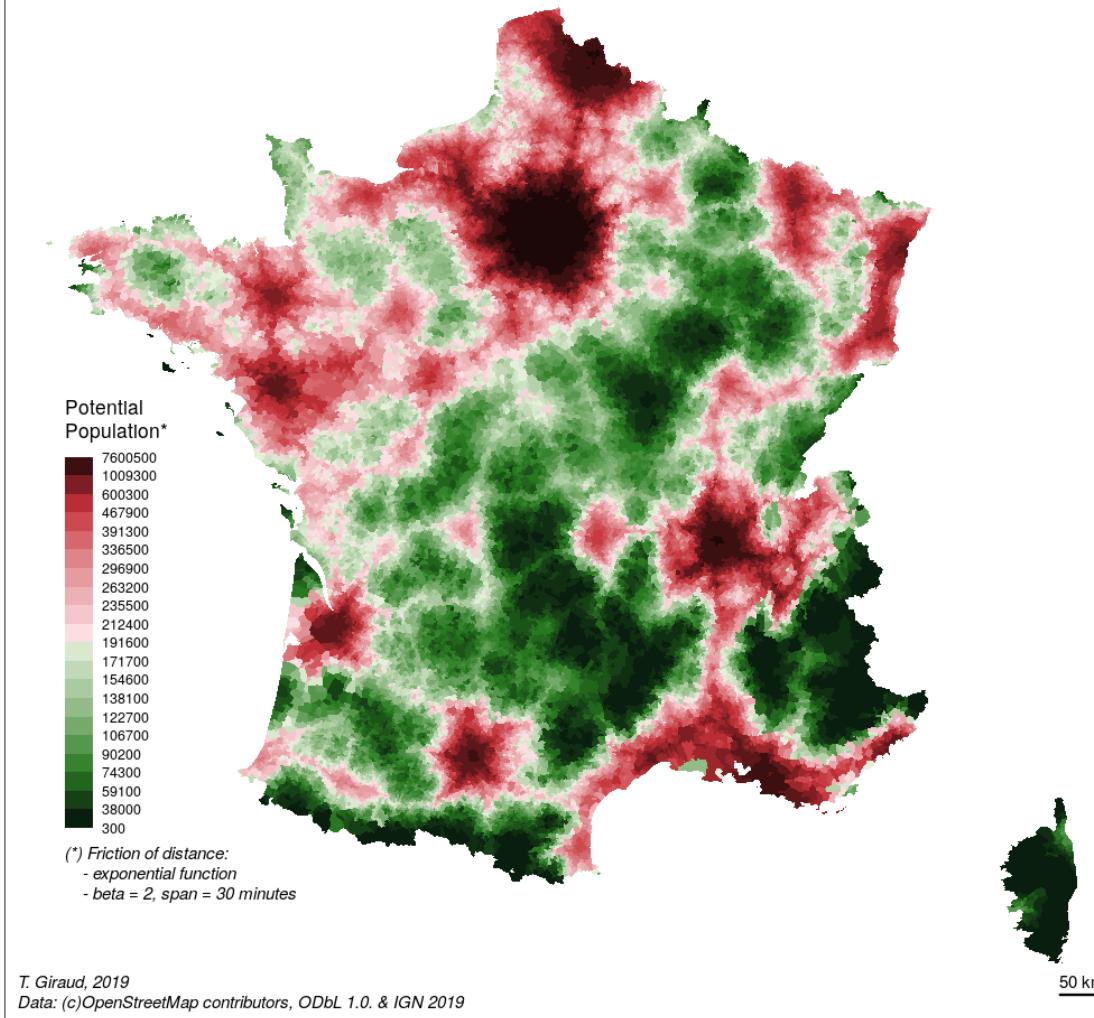
```

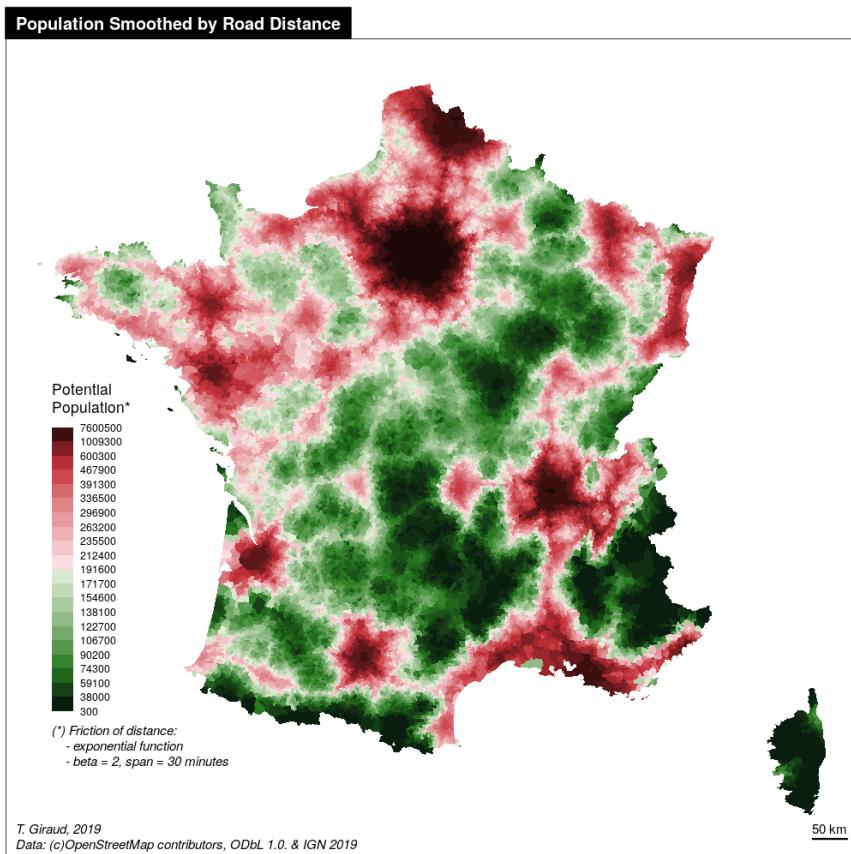
library(sf)
library(cartography)
# path to the geo file embedded in cartography
path <- system.file("gpkg/mtq.gpkg",
                     package = "cartography")
# import to an sf object
mtq <- st_read(dsn = path, quiet = TRUE)
# Set figure margins
par(mar = c(0,0,1.2,0), bg = "#b5d0d0")
# Plot the choropleth map
choroLayer(
  x = mtq, var = "MED",
  method = "equal", nclass = 4,
  col = carto.pal(pal1 = "sand.pal", n1 = 4),
  legend.values.rnd = -2, legend.pos = "left",
  legend.title.txt = "Median Income\n(in euros)"
)
# Plot a layout
layoutLayer(
  title="Wealth in Martinique, 2015",
  author = "T. Giraud, 2019",
  sources = "Sources: Insee and IGN - 2018",
  scale = 5, north = TRUE,
  tabtitle = TRUE, frame = FALSE
)

```



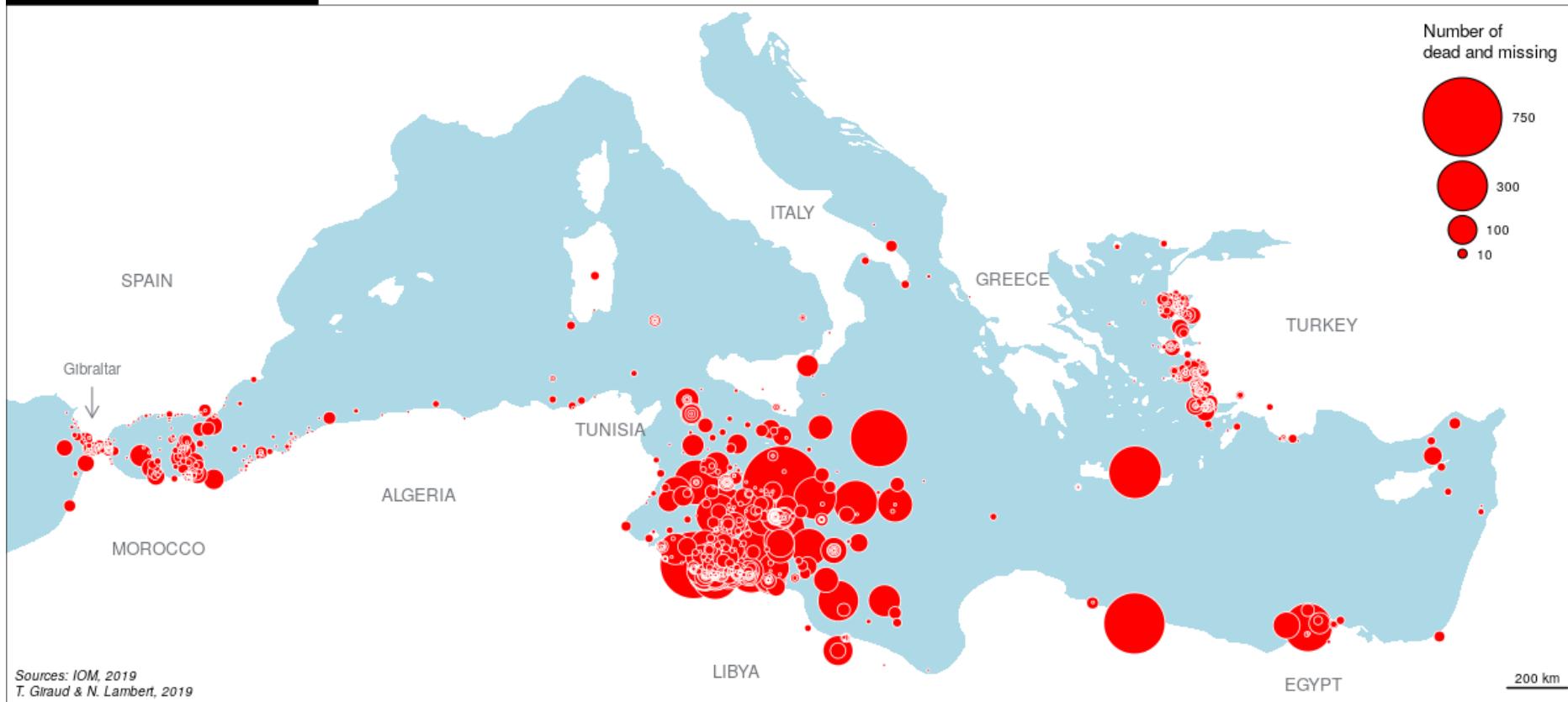
### Population Smoothed by Road Distance



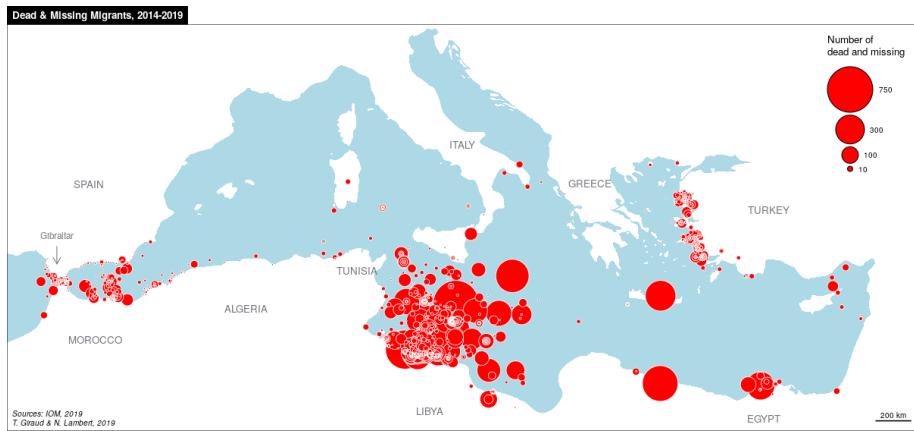


- osrm: Gets road distances (time) between each municipalities.
- SpatialPosition: Computes gravitationnal accessibility based on road distances.
- cartography::getBreaks(): Classifies data
- cartography::carto.pal(): Uses color palettes from the package
- cartography::choroLayer(): Plots the choropleth map

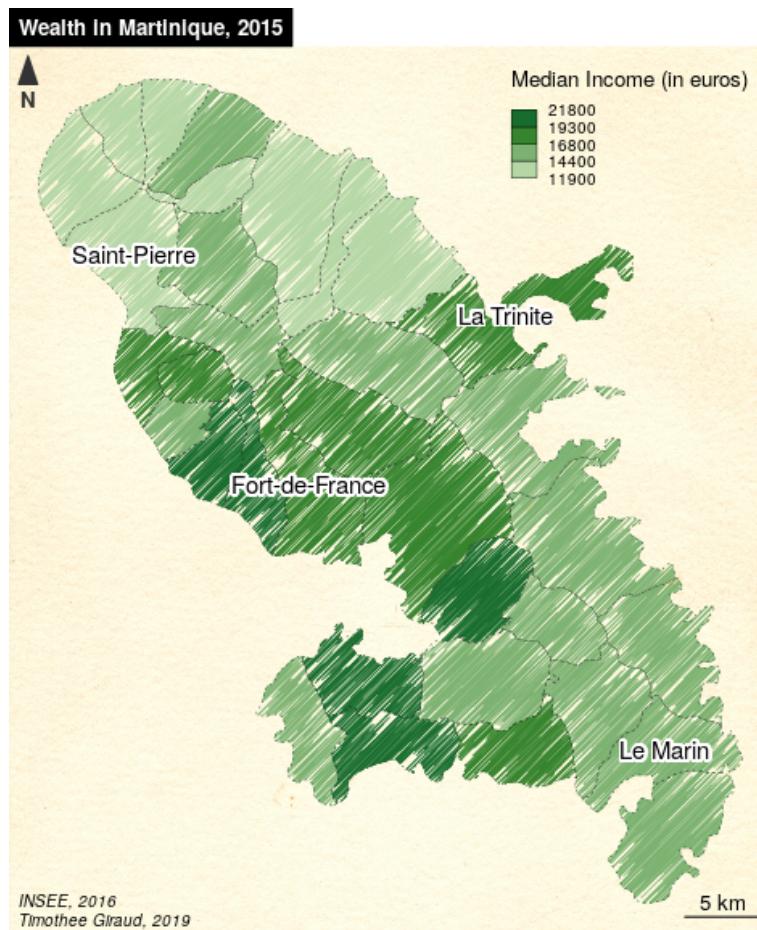
**Dead & Missing Migrants, 2014-2019**



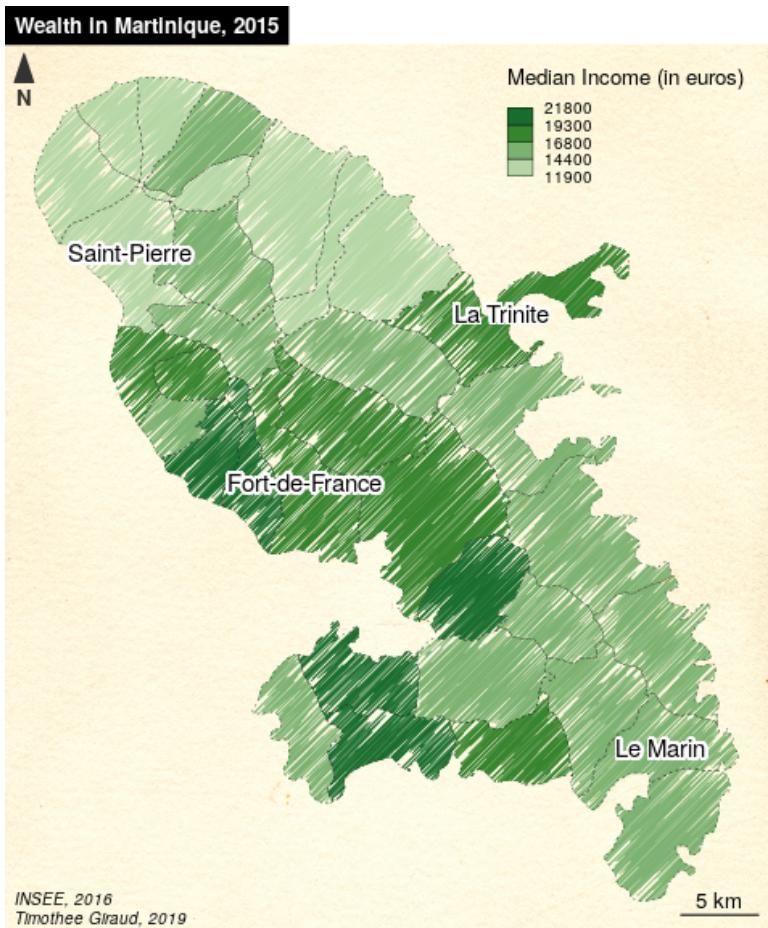
[riatelab.github.io/MDM](http://riatelab.github.io/MDM)



- **sf:** Provides geodata transformations
- **cartography::propSymbolsLayer():** Plots the proportional symbols
- **cartography::layoutLayer():** Plots the map layout

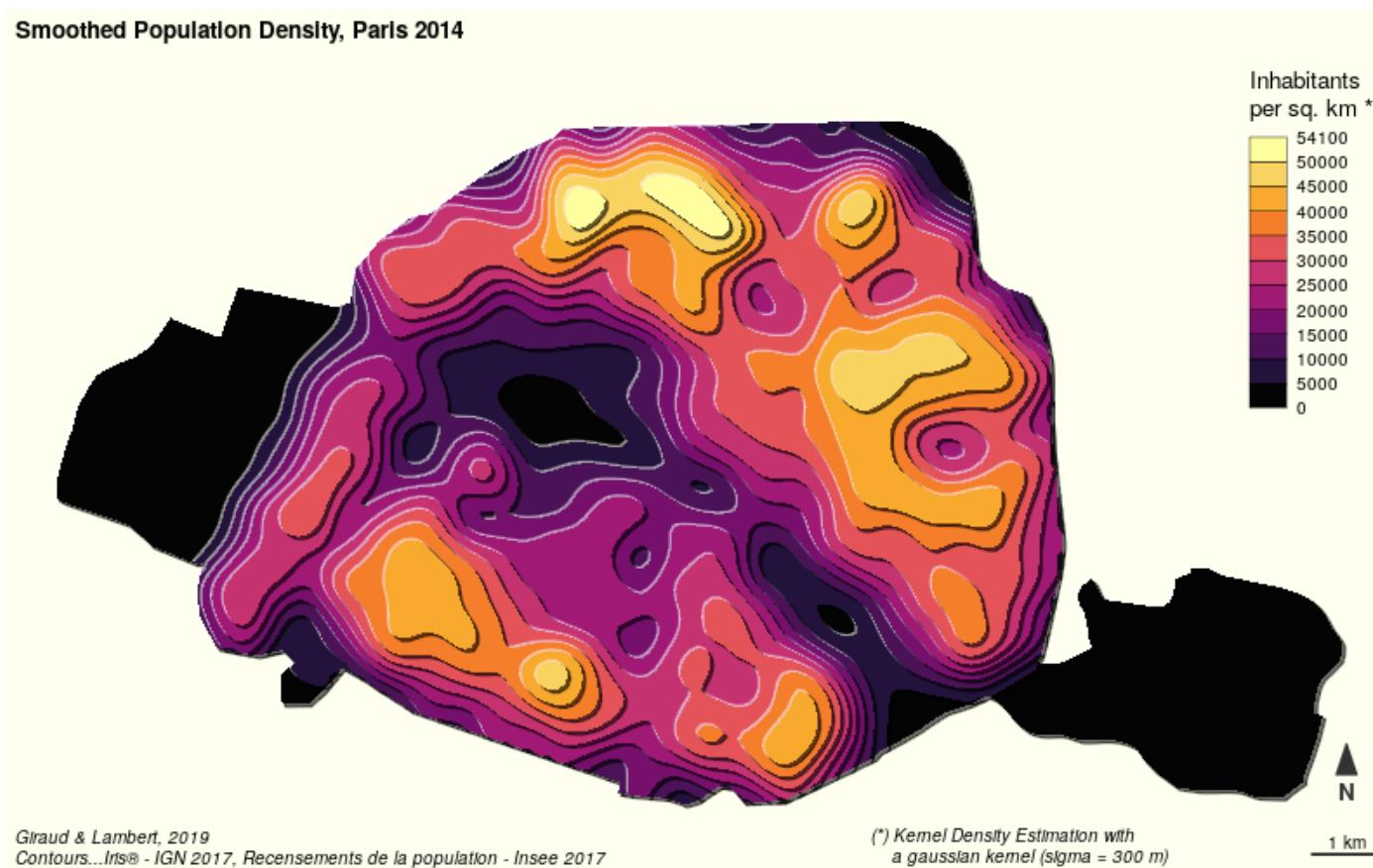


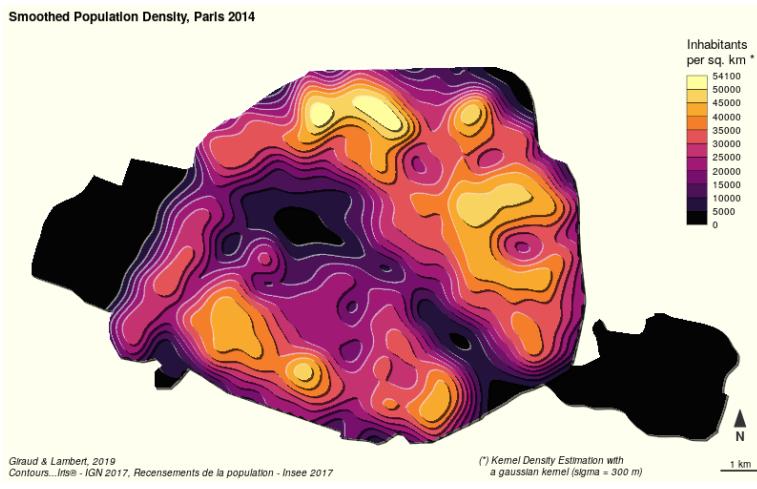
[gist](#)



- `cartography::getPencilLayer()`: Transforms polygons to hand-drawn polylines
- `cartography::choroLayer()`: Plots the choropleth map

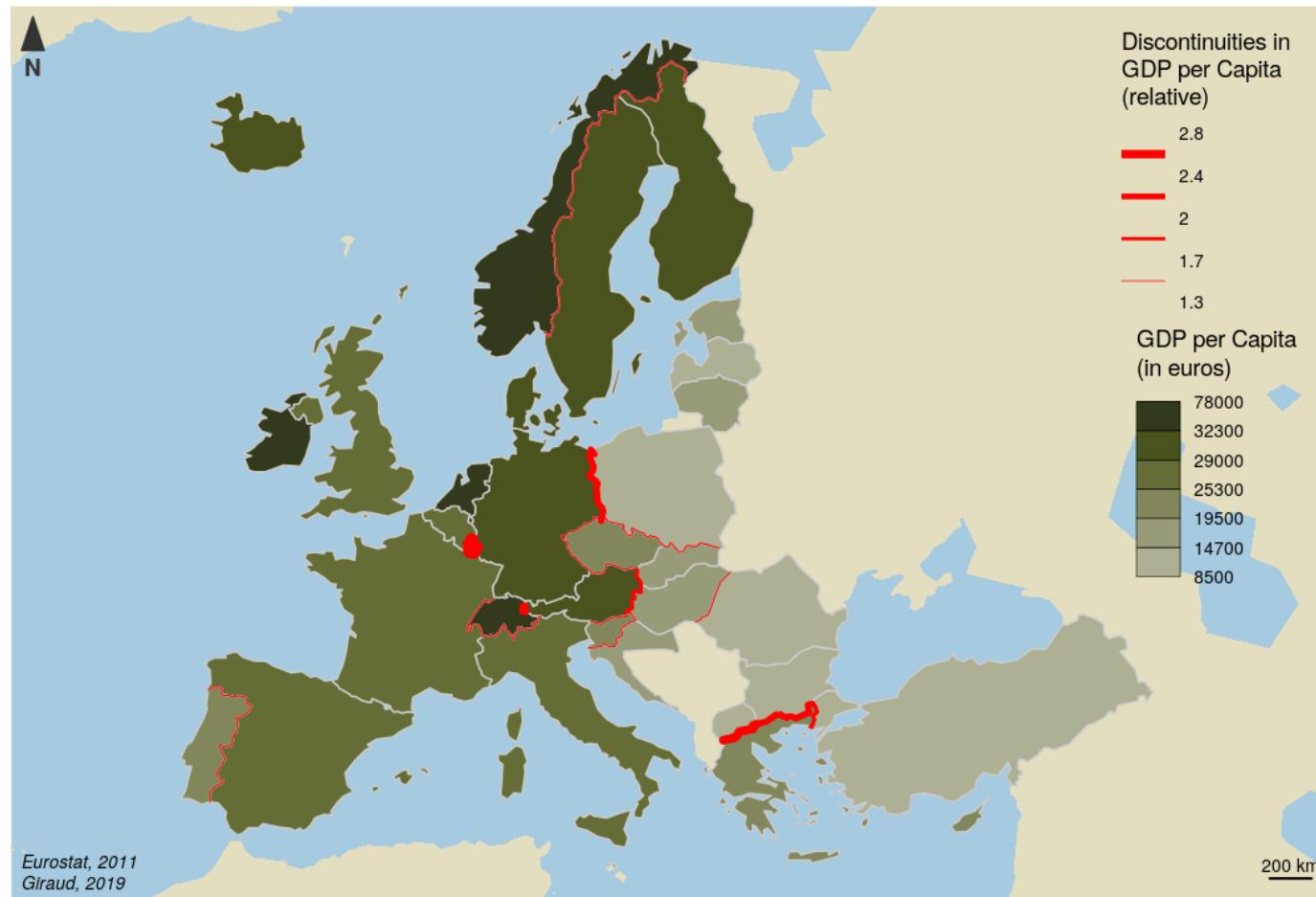
Smoothed Population Density, Paris 2014



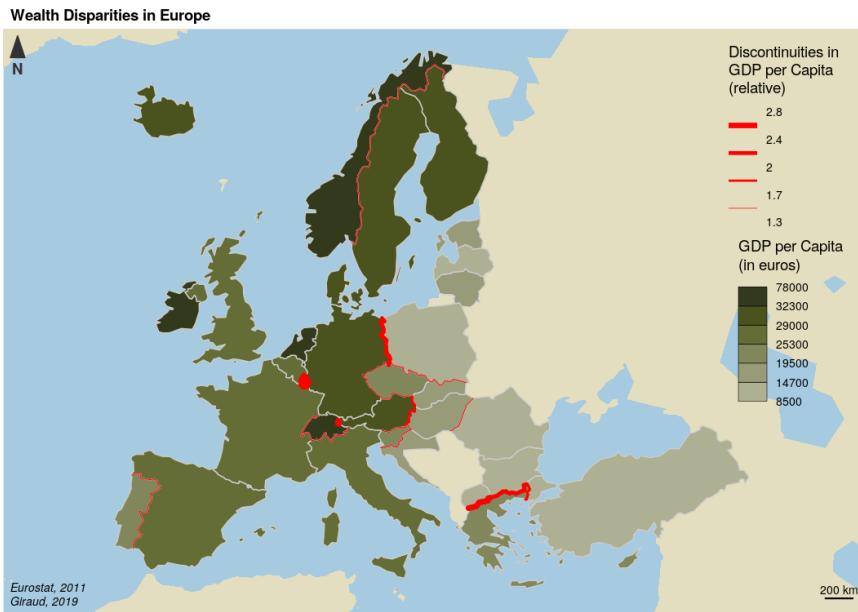


- spatstat: Computes Kernel Density Estimation
- tanaka: Plots shaded contour lines
- cartography::LegendChoro(): Plots legend
- cartography::layoutLayer(): Plots map layout

### Wealth Disparities in Europe



gist



- `cartography::choroLayer()`: Plots the choropleth map
- `cartography::getBorders()`: Extract borders between countries
- `cartography::discLayer()`: Plots discontinuities on borders
- `cartography::layoutLayer()`: Plots the map layout

# Thank You



[frama.link/cartography](https://frama.link/cartography)



[github.com/riatelab/cartography](https://github.com/riatelab/cartography)



[@rgeomatic](https://twitter.com/rgeomatic)



[rgeomatic.hypotheses.org](https://rgeomatic.hypotheses.org)