Thematic Maps with cartography

Timothée Giraud

useR! 2019 - Toulouse - 2019/07/09

Produce publication ready thematic maps

Standing on the Shoulders of Giants

Initialy 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")

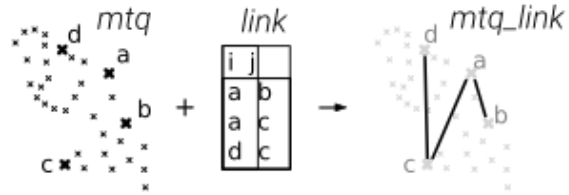
Dot Density
dotDensityLayer(x = mtq, var = "myvar")

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

Main Features: Transformations

Points to Links

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

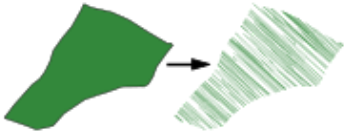


Polygons to Borders

mtq_border <- getBorders(x = mtq)

Polygons to Pencil Lines

mtq_pen <- getPencilLayer(x = mtq)

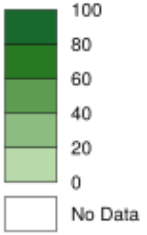


Polygons to Grid

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

Main Features: Map Layout

legendChoro()




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

legendTypo()


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

legendCirclesSymbols()


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

North Arrow:

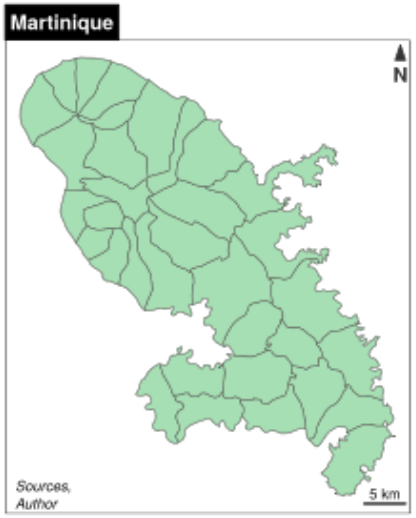
```
north(pos = "topright")
```

Scale Bar:

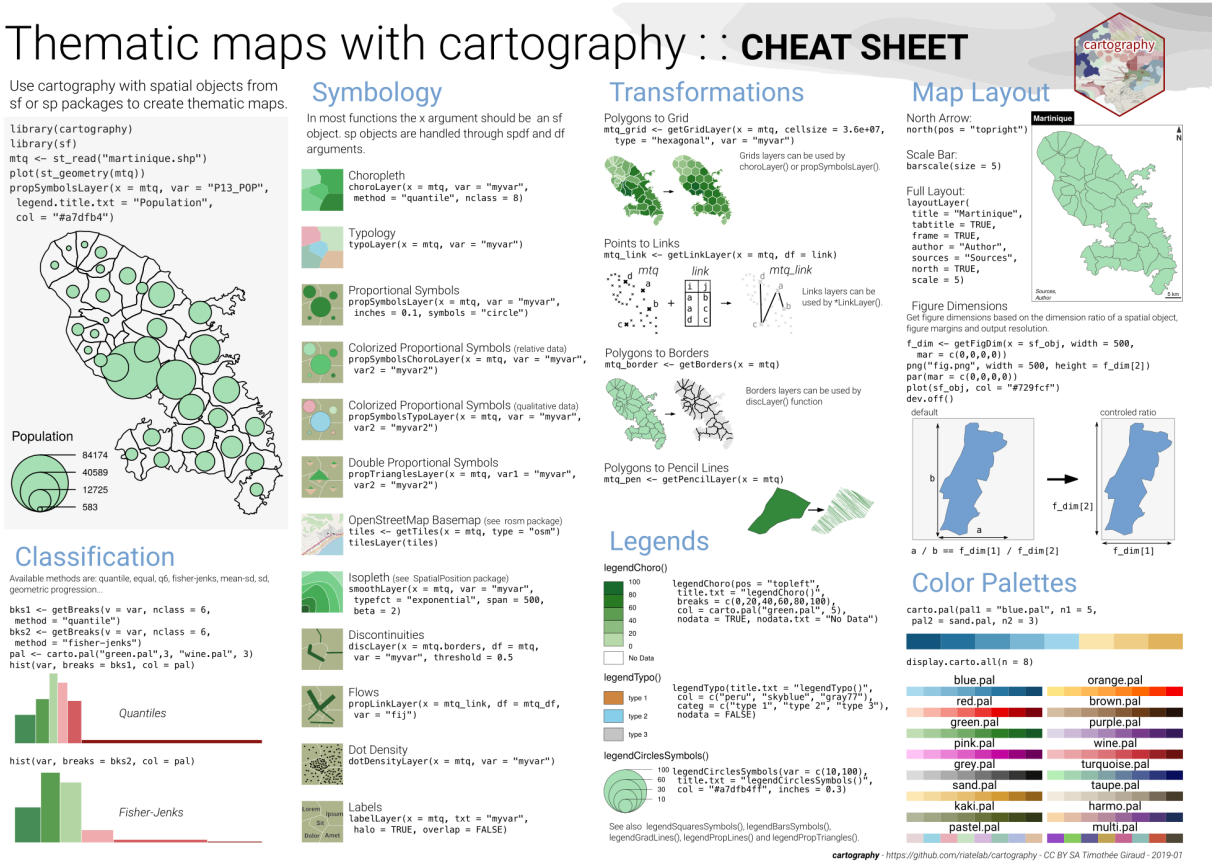
```
barscale(size = 5)
```

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
cartography
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))
propSymbolsLayer(x = mtq, var = "P13_POP",
legend.title.txt = "Population",
col = "#a7dfb4")
Population
84174
40589
12725
583
Classification
Available methods are: quantile, equal, q6, fisher-jenks, mean-sd, 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)
Quantiles
hist(var, breaks = bks2, col = pal)
Fisher-Jenks
Symbology
In most functions the x argument should be an sf object. sp objects are handled through spdf and df arguments.
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")
Colorized Proportional Symbols (qualitative data)
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)
Transformations
Polygons to Grid
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)
mtq
link
mtq_link
Links layers can be used by *LinkLayer().
Polygons to Borders
mtq_border <- getBorders(x = mtq)
Borders layers can be used by discLayer() function
Polygons to Pencil Lines
mtq_pen <- getPencilLayer(x = mtq)
Legends
legendChoro()
legendChoro(pos = "topleft",
title.txt = "legendChoro()",
breaks = c(0,20,40,60,80,100),
col = carto.pal("green.pal", 5),
nodata = TRUE, nodata.txt = "No Data")
No Data
legendTypo()
legendTypo(title.txt = "legendTypo()",
col = c("peru", "skyblue", "gray77"),
categ = c("type 1", "type 2", "type 3"),
nodata = FALSE)
type 1
type 2
type 3
legendCirclesSymbols()
legendCirclesSymbols(var = c(10,100),
title.txt = "legendCirclesSymbols()",
col = "#a7dfb4ff", inches = 0.3)
See also legendSquaresSymbols(), legendBarsSymbols(), legendGradLines(), legendPropLines() and legendPropTriangles().
Map Layout
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)
Martinique
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))
png("fig.png", width = 500, height = f_dim[2])
par(mar = c(0,0,0,0))
plot(sf_obj, col = "#729fcf")
dev.off()
default
controled ratio
f_dim[2]
a / b == f_dim[1] / 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
orange.pal
red.pal
brown.pal
green.pal
purple.pal
pink.pal
wine.pal
grey.pal
turquoise.pal
sand.pal
taupe.pal
kaki.pal
harmo.pal
pastel.pal
multi.pal
cartography - https://github.com/riatelab/cartography - CC BY SA Timothée Giraud - 2019-01

Website

cartography 2.2.1 | Get started | Reference | Articles | Changelog

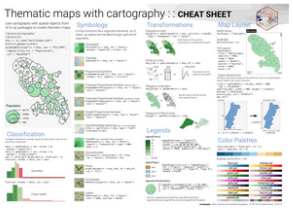
cartography

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.

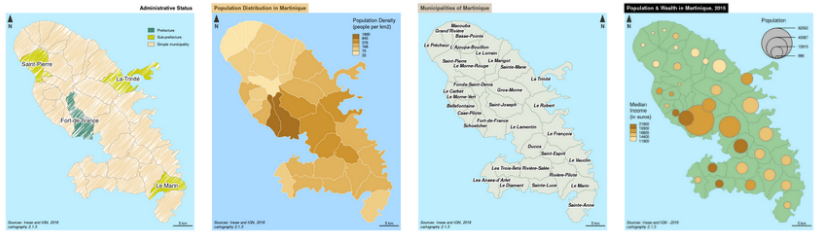
Cheat Sheet

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

Browse source code at https://github.com/riatelab/cartography

Report a bug at https://github.com/riatelab/cartography/issues/

License

GPL-3

Citation

Citing cartography

Developers

Timothée Giraud
Maintainer, author

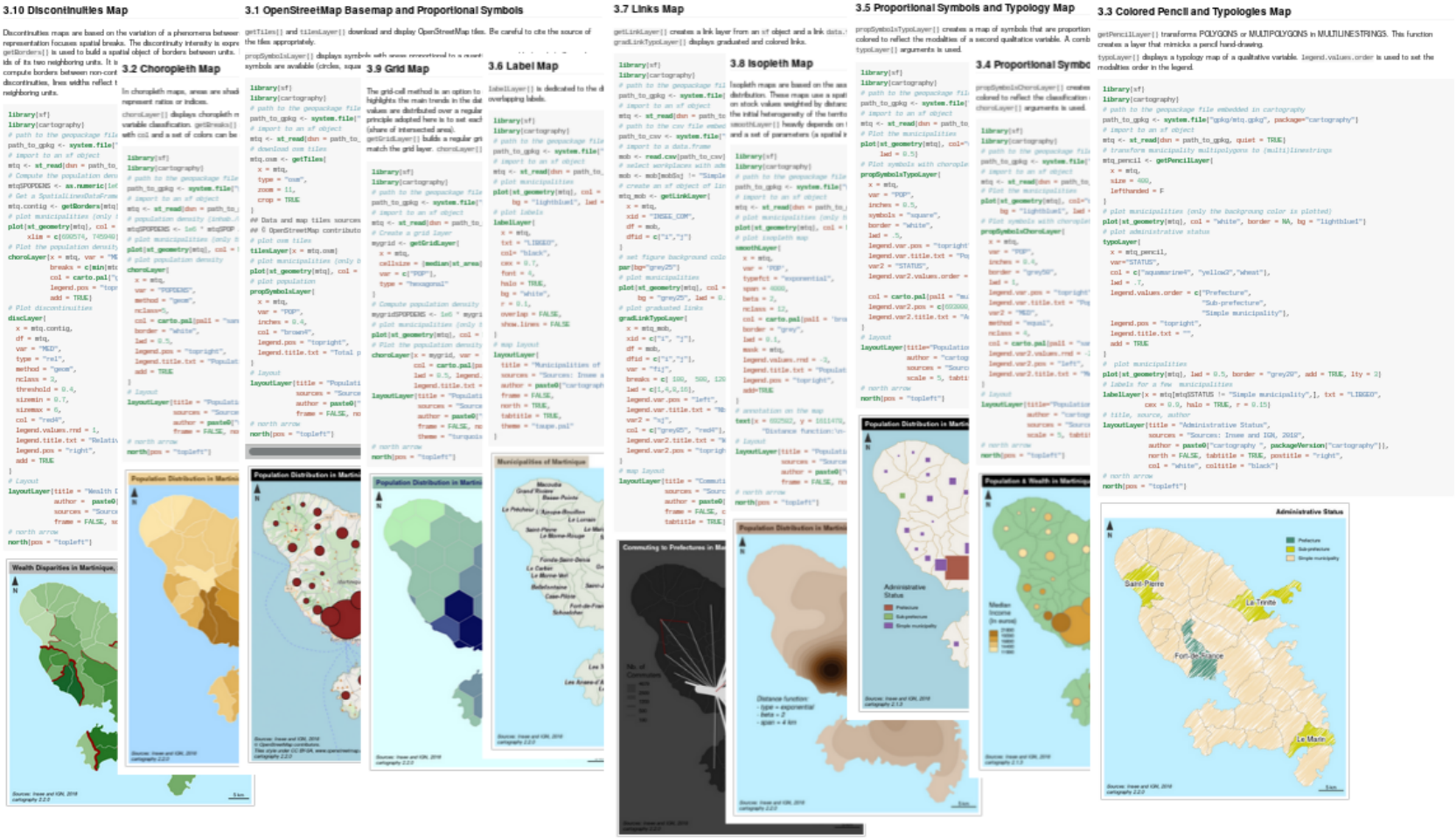
Nicolas Lambert
Author

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
3.1 OpenStreetMap Basemap and Proportional Symbols
3.7 Links Map
3.5 Proportional Symbols and Typology Map
3.3 Colored Pencil and Typologies Map
3.2 Choropleth Map
3.9 Grid Map
3.6 Label Map
3.8 Isopleth Map
3.4 Proportional Symbo

Alternative Solutions

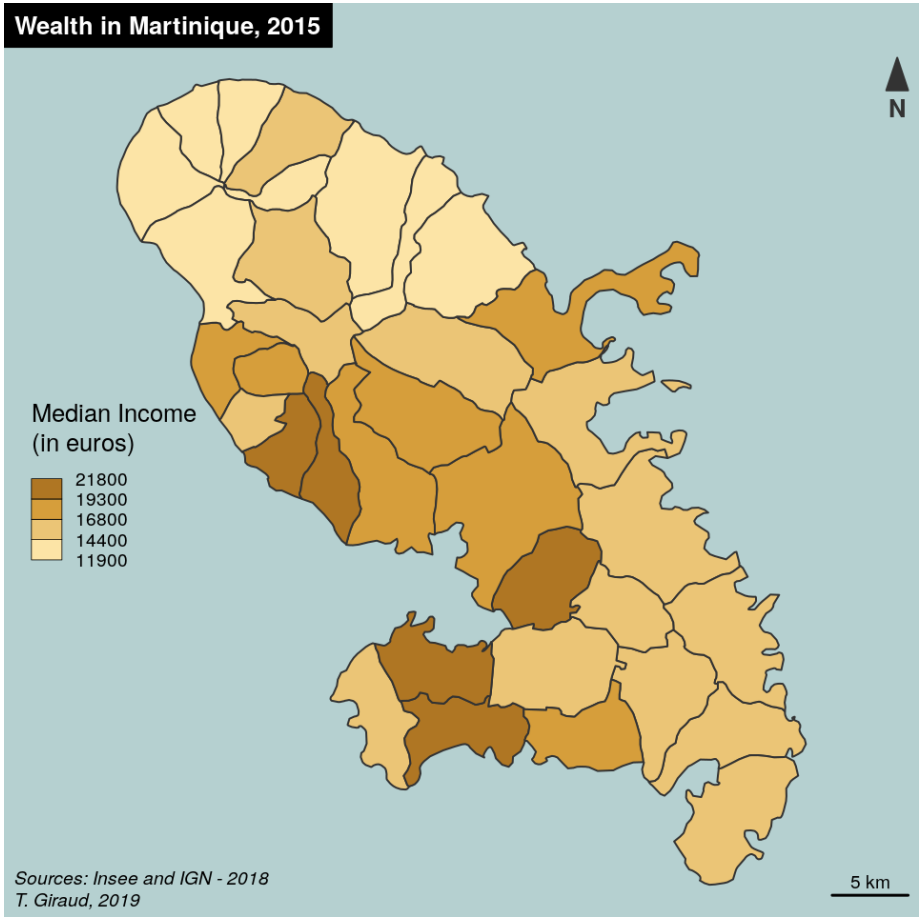
`ggplot2` (Wickham, 2016) + `ggspatial` (Dunnington, 2018)

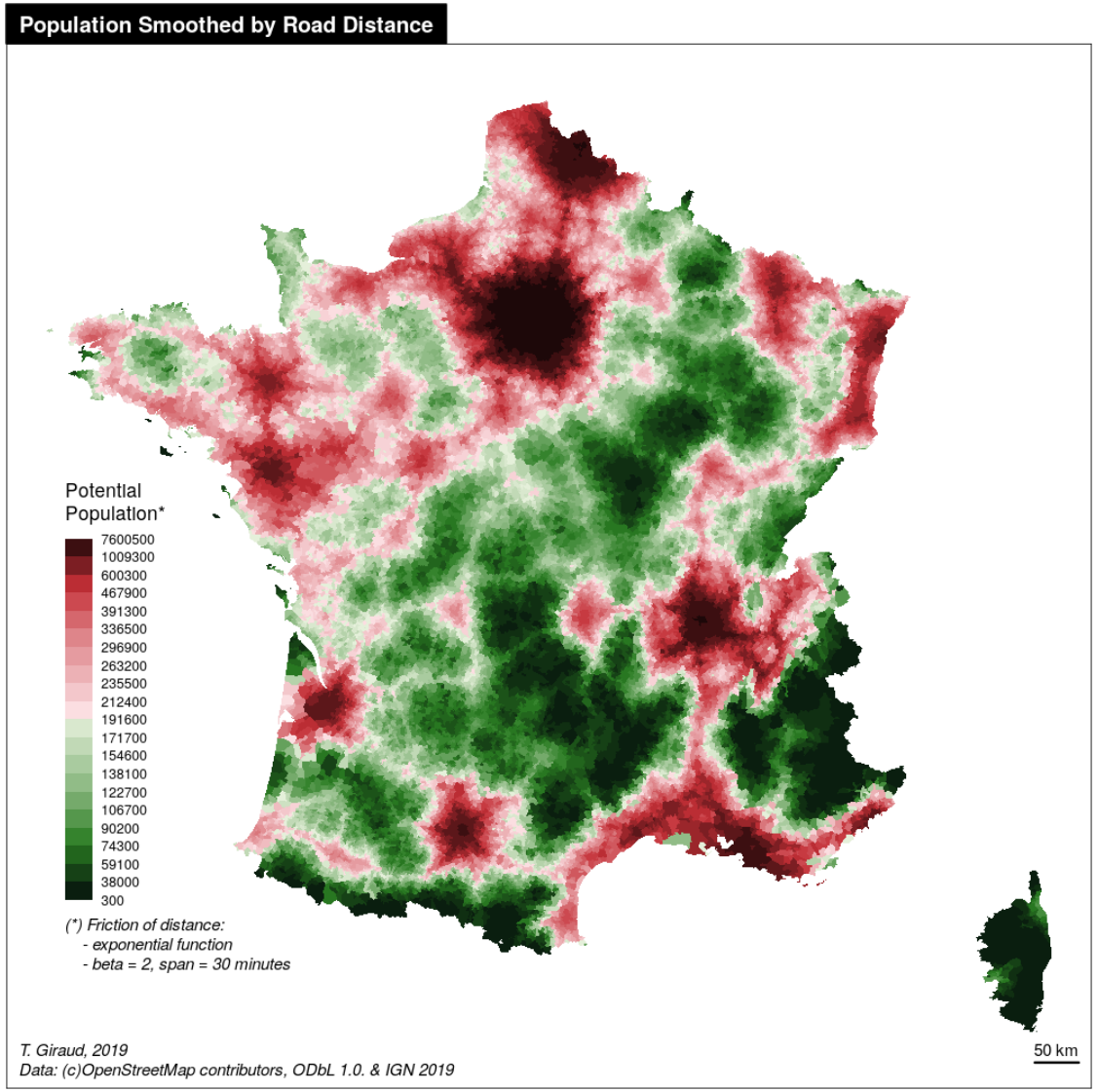
- A general purpose graphic library

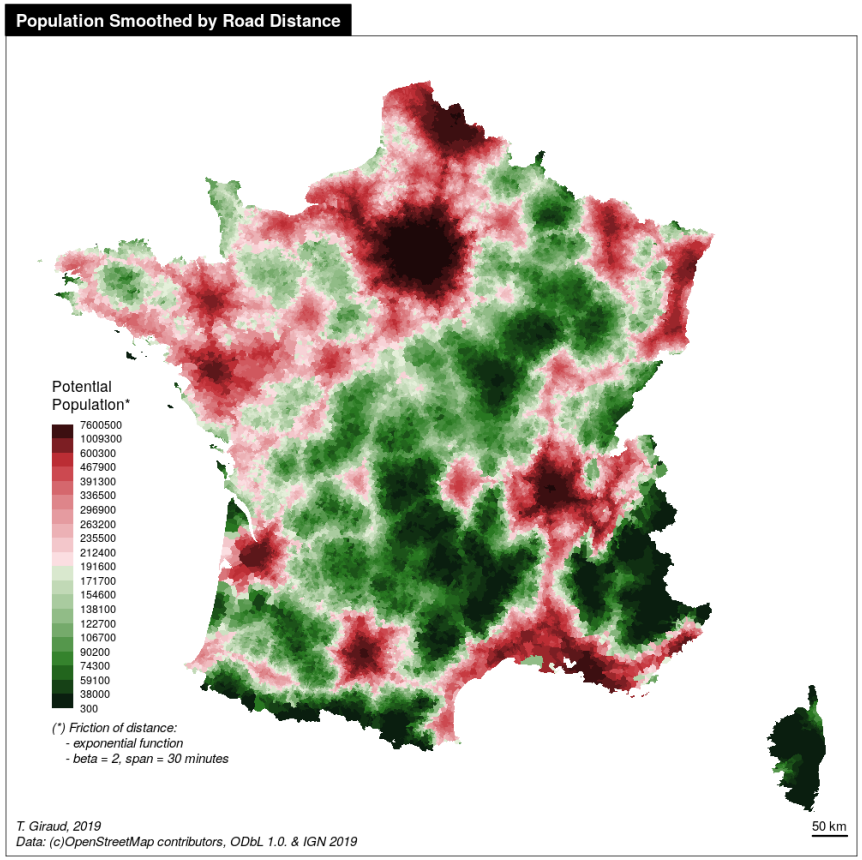
`tmap` (Tennekes, 2018)

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

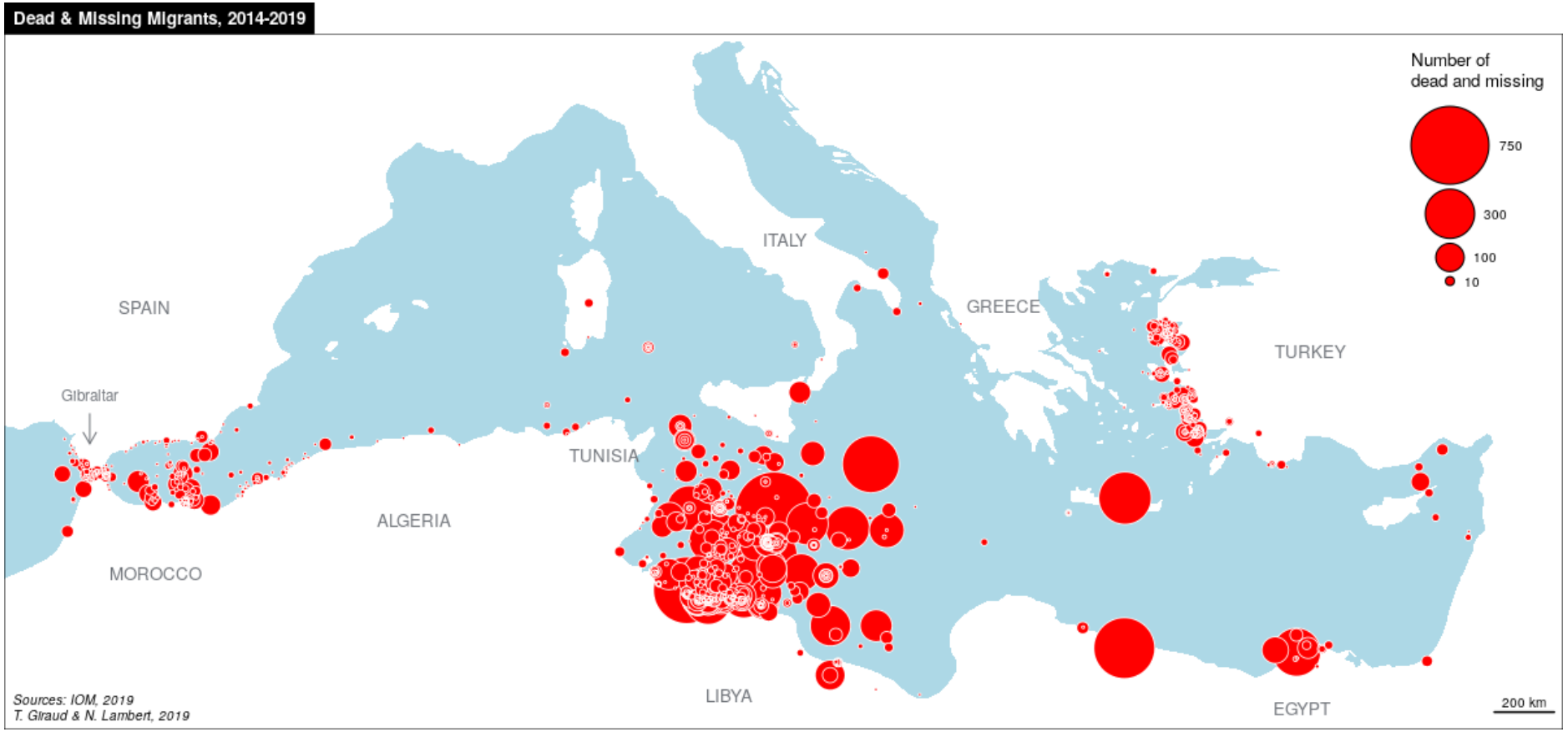
```r
library(sf)
library(cartography)
# path to the geo file embedded in cartography
path <- system.file("gpkg/mtq.gpkg",
                    package = "cartography")
# import to an sf object
mtq <- st_read(dsn = path, quiet = TRUE)
# Set figure margins
par(mar = c(0,0,1.2,0), bg = "#b5d0d0")
# Plot 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
)
```



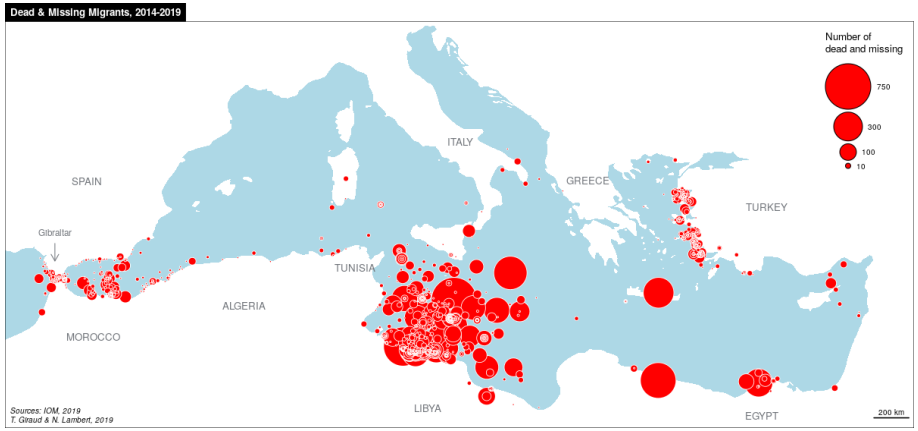

Population Smoothed by Road Distance
Potential Population*
7600500
1009300
600300
467900
391300
336500
296900
263200
235500
212400
191600
171700
154600
138100
122700
106700
90200
74300
59100
38000
300
(*) Friction of distance:
- exponential function
- beta = 2, span = 30 minutes
T. Giraud, 2019
Data: (c)OpenStreetMap contributors, ODbL 1.0. & IGN 2019
50 km



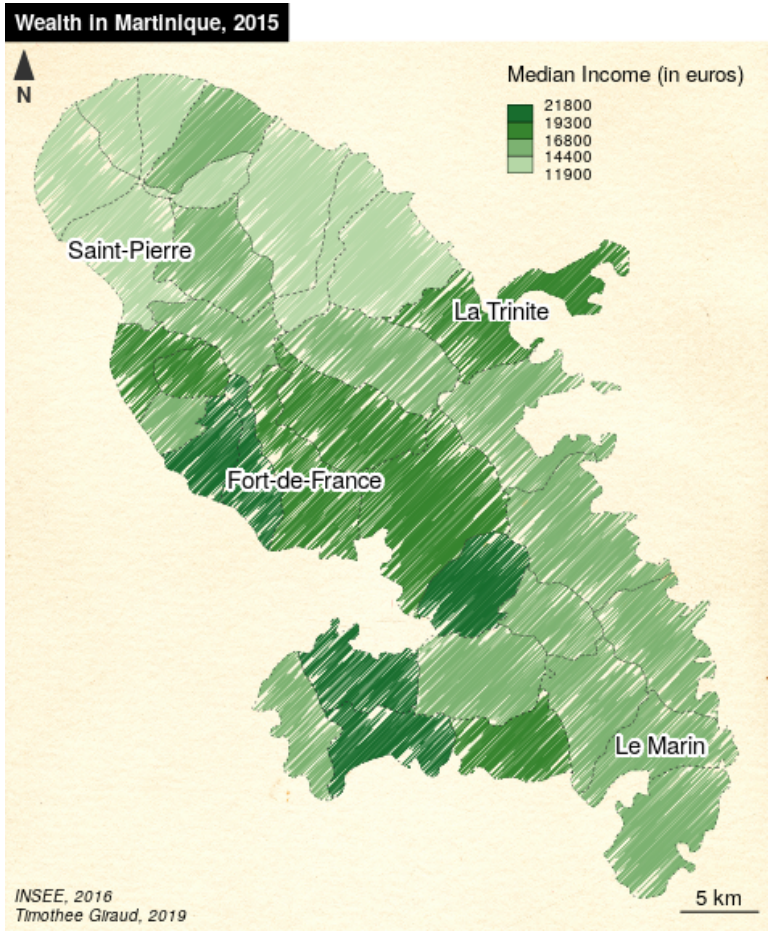
- `osrm`: Gets road distances (time) between each municipalities.
- `SpatialPosition`: Computes gravitationnal accessibility based on road distances.
- `cartography::getBreaks()`: Classifies data
- `cartography::carto.pal()`: Uses color palettes from the package
- `cartography::choroLayer()`: Plots the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



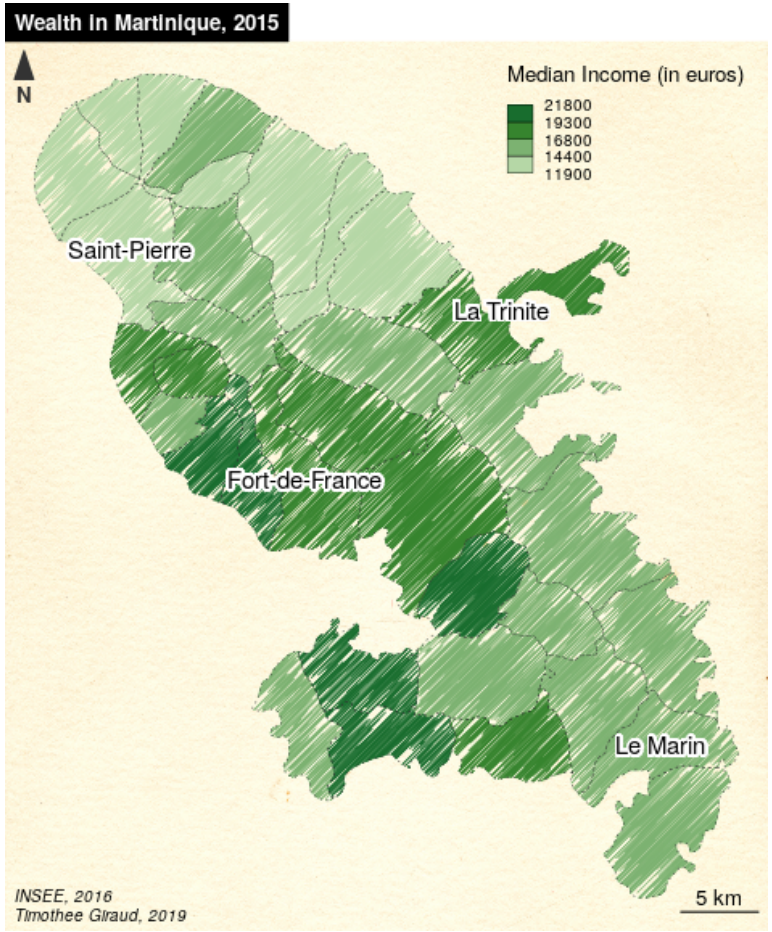
riatelab.github.io/MDM



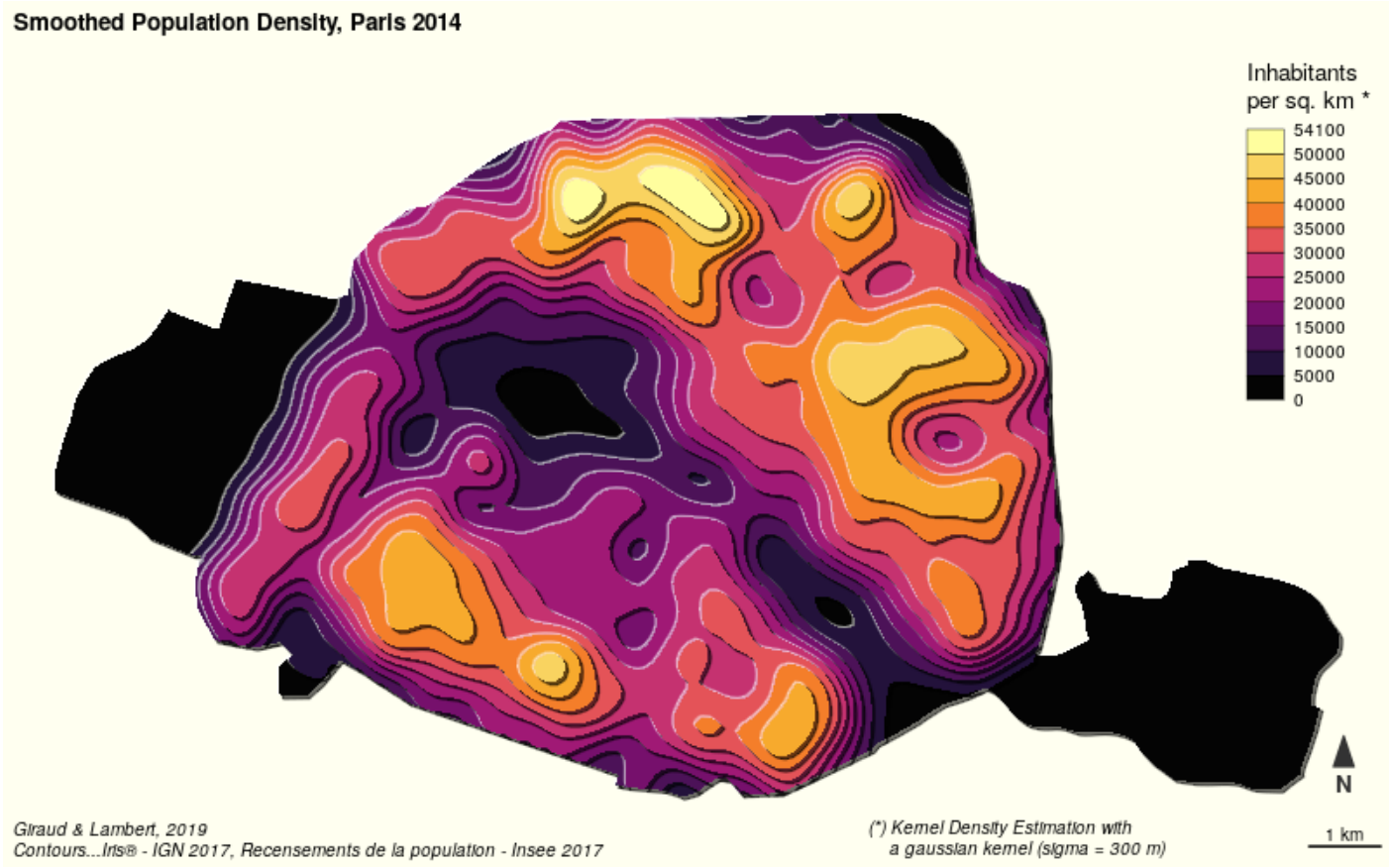
- `sf`: Provides geodata transformations
- `cartography::propSymbolsLayer()`: Plots the proportional symbols
- `cartography::layoutLayer()`: Plots the map layout


Wealth in Martinique, 2015
N
Median Income (in euros)
21800
19300
16800
14400
11900
Saint-Pierre
La Trinite
Fort-de-France
Le Marin
INSEE, 2016
Timothee Giraud, 2019
5 km

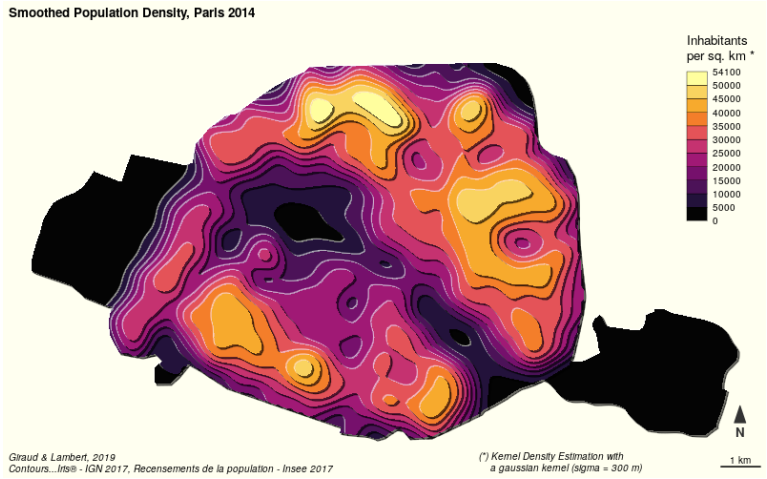
gist



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

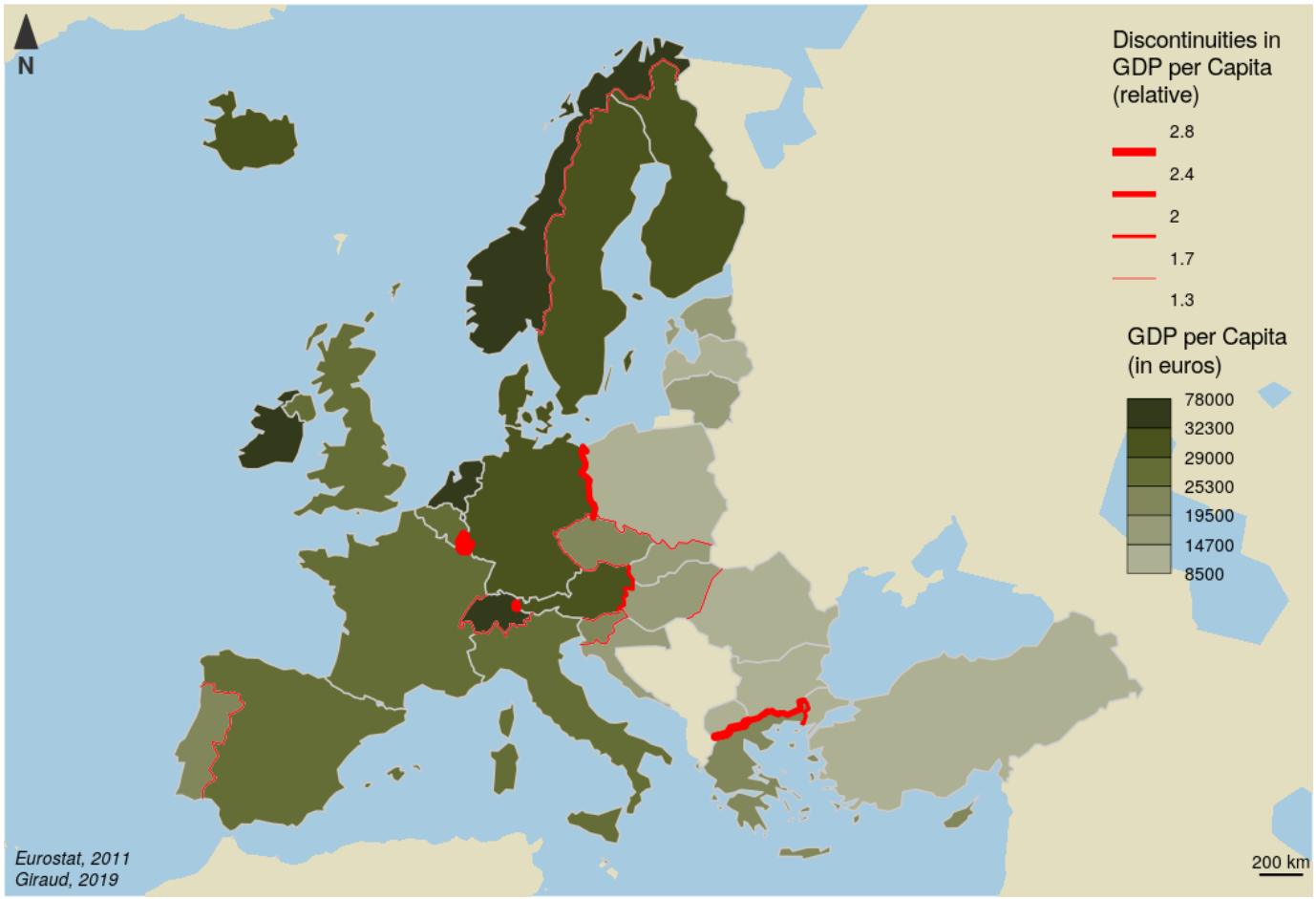


rgeomatic.hypotheses.org

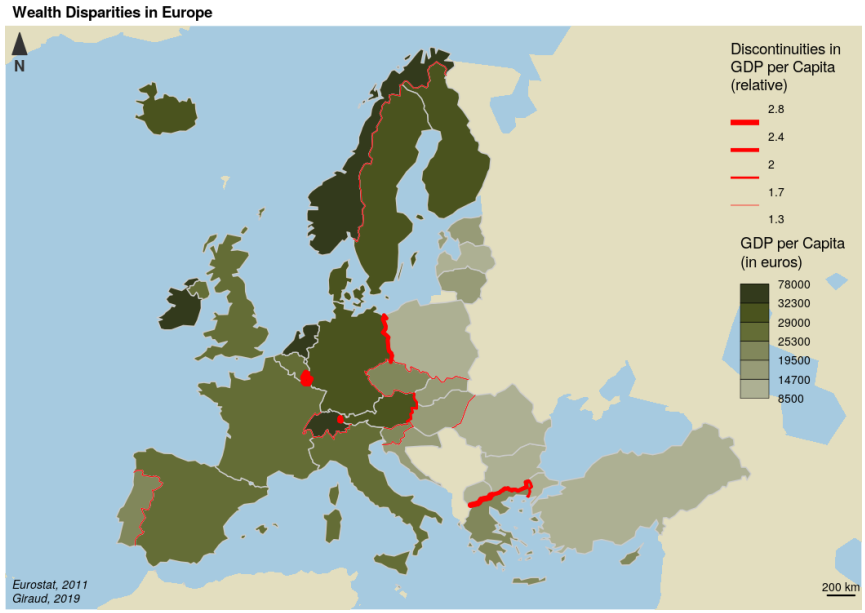


- `spatstat`: Computes Kernel Density Estimation
- `tanaka`: Plots shaded contour lines
- `cartography::LegendChoro()`: Plots legend
- `cartography::layoutLayer()`: Plots map layout

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

github.com/riatelab/cartography

@rgeomatic

rgeomatic.hypotheses.org