bigDM

Erick Orozco-Acosta, Aritz Adin and María Dolores Ugarte

2020-07-10

Introduction

Disease mapping is the field of spatial epidemiology that studies the link between geographic locations and the occurrence of diseases, focusing on the estimation of the spatial and/or spatio-temporal distribution of disease incidence or mortality patterns. The great variability inherent to classical risk estimation measures, such as standardized mortality/incidence ratios or crude rates, makes necessary the use of statistical models to estimate smooth spatial risk surfaces borrowing information from neighbouring regions. The information adquired from these analyses is invaluable for health researchers and policy-makers as it helps to formulate hypothesis about the etiology of a disease, to look for risk factors and and also to allocate funds efficiently in hot spot areas.

Spatial models in disease mapping

Let us assume that the spatial domain of interest is divided into n contiguous small areas labeled as $i = 1, \ldots, n$. For a given area i, let O_i and E_i denote the observed and expected number of disease cases, respectively. Using these quantities, the *standardized mortality/incidence ratio* (SMR or SIR) is defined as the ratio of observed and expected cases for the corresponding areal unit. Although its interpretation is very simple, these measures are extremely variable when analyzing rare diseases or low-populated areas, as it is the case of high-dimensional data. To cope with this situation, it is necessary to use statistical models that stabilize the risks (rates) borrowing information from neighbouring regions.

Generalized linear mixed models (GLMM) are typically used for the analysis of count data within a hierarchical Bayesian framework. Conditional to the relative risk r_i, the number of observed cases in the ith area is assumed to be Poisson distributed with mean $\mu_i = E_i r_i$. That is,

$$\begin{array}{rcl} O_i|r_i & \sim & Poisson(\mu_i = E_i r_i),\ i = 1, \ldots, n \\ \log \mu_i & = & \log E_i + \log r_i, \end{array}$$

% where $\log E_i$ is an offset. Depending on the specification of the log-risks different models are defined. Here we assume that %

$$\log r_i = \alpha + \xi_i, \tag{1}$$

% where α is an intercept representing the overall log-risk and $\xi = (\xi_1, \ldots, \xi_n)'$ is a spatial random effect for which a conditional autoregressive (CAR) prior is usually assumed. The spatial correlation between CAR random effects is determined by the neighbouring structure (represented as an undirected graph) of the areal units. Let $\mathbf{W} = (w_{ij})$ be a binary $n \times n$ adjacency matrix, whose ijth element is equal to one if areas j and k are defined as neighbours (usually if they share a common border), and it is zero otherwise. Here, the prior distribution proposed by Leroux et. al.(2000) has been considered, which is given by

$$\xi \sim N(0, Q_\xi^{-1}), \quad \text{where} \quad Q_\xi = \tau_\xi[\lambda_\xi(D_W - W) + (1 - \lambda_\xi)I_n]$$

% and $\tau_\xi = 1/\sigma_\xi^2$ is the precision parameter, $\lambda_\xi \in [0, 1)$ is a spatial smoothing parameter, $D_W = diag(w_{1+}, \ldots, w_{n+})$ and $w_{i+} = \sum_j w_{ij}$ is the ith row sum of W, and I_n is the $n \times n$ identity matrix. % We will refer to this model as the *Global model*.

A initial step: Merge cartography file with datasets

This package has as input files of the class `sf` or `spatialpolygonsdataframe`, but not always the user has this type of file. For example: in the case of Spain, we have the following cartography files:

- MGN_MPIO_POLITICO.shp
- MGN_MPIO_POLITICO.shx
- MGN_MPIO_POLITICO.prj
- MGN_MPIO_POLITICO.dbf

To convert this cartography in `SpatialPolygonsDataFrame` the following is required:

```
#### import to spatialpolygonsdataframe ####
library(rgdal)
#> Loading required package: sp
#> rgdal: version: 1.5-12, (SVN revision 1018)
#> Geospatial Data Abstraction Library extensions to R successfully loaded
#> Loaded GDAL runtime: GDAL 2.4.2, released 2019/06/28
#> Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/4.0/Resources/library/rgdal/gdal
#> GDAL binary built with GEOS: FALSE
#> Loaded PROJ runtime: Rel. 5.2.0, September 15th, 2018, [PJ_VERSION: 520]
#> Path to PROJ shared files: /Library/Frameworks/R.framework/Versions/4.0/Resources/library/rgdal/proj
#> Linking to sp version:1.4-2
#> Overwritten PROJ_LIB was /Library/Frameworks/R.framework/Versions/4.0/Resources/library/rgdal/proj
zones_spdf <-
  readOGR(system.file("shape/Carto_SpainMUN.shp", package = "bigDM"),
          layer = "Carto_SpainMUN")
#> OGR data source with driver: ESRI Shapefile
#> Source: "/private/var/folders/yh/2lzsb6gs5bvd1cvvgjxztm8r0000gp/T/RtmpFoz4kk/temp_libpathc7893f2722e
#> with 7907 features
#> It has 7 fields
```

Alternatively, an `sf` can be used:

```
#### import to sf####
library(sf)
#> Linking to GEOS 3.7.2, GDAL 2.4.2, PROJ 5.2.0
zones_sf <-
  st_read(system.file("shape/Carto_SpainMUN.shp", package = "bigDM"))
#> Reading layer `Carto_SpainMUN' from data source `/private/var/folders/yh/2lzsb6gs5bvd1cvvgjxztm8r000
#> Simple feature collection with 7907 features and 7 fields
#> geometry type:  MULTIPOLYGON
#> dimension:      XY
#> bbox:           xmin: -13952 ymin: 3987408 xmax: 1021167 ymax: 4859444
#> proj4string:    +proj=utm +zone=30 +ellps=GRS80 +units=m +no_defs
```

To merge the observed and expected data with the imported cartography it is necessary to have the data organized by a common identifier. Using function as `merge()` or logical relations it is possible to have in a unique file cartography and data.

```
#### data with expected and observed cases in csv file ####
library(readr)
data <- read_csv(system.file("csv/data.csv", package = "bigDM"),
                 col_types = cols(ID = col_character(),
                                  exp = col_number(),
                                  obs = col_number()))
zones_sf<- merge (zones_sf, data, by.x="ID", by.y="ID")
```

```
head(zones_sf)
#> Simple feature collection with 6 features and 9 fields
#> geometry type:  MULTIPOLYGON
#> dimension:      XY
#> bbox:           xmin: 485318 ymin: 4727428 xmax: 543317 ymax: 4779153
#> proj4string:    +proj=utm +zone=30 +ellps=GRS80 +units=m +no_defs
#>      ID                                    name     lat    long     area
#> 1 01001                          Alegria-Dulantzi 4742974 4742974 19913794
#> 2 01002                                   Amurrio 4765886 4765886 96145595
#> 3 01003                                   Aramaio 4766225 4766225 73338806
#> 4 01004                                Artziniega 4775453 4775453 27506468
#> 5 01006                                   Arminon 4729884 4729884 10559721
#> 6 01008 Arrazua-Ubarrundia (San Martin de Ania) 4752714 4752714 57502811
#>   perimeter     region obs        exp                       geometry
#> 1  34372.11 Pais Vasco   4  3.0237149 MULTIPOLYGON (((538259 4737...
#> 2  63352.32 Pais Vasco  27 20.8456682 MULTIPOLYGON (((503520 4760...
#> 3  41430.46 Pais Vasco   6  3.7527301 MULTIPOLYGON (((533286 4759...
#> 4  22605.22 Pais Vasco   5  3.2093191 MULTIPOLYGON (((491260 4776...
#> 5  17847.35 Pais Vasco   0  0.4817391 MULTIPOLYGON (((509851 4727...
#> 6  64968.81 Pais Vasco   1  1.9643891 MULTIPOLYGON (((534678 4746...
```

Disjoint model

This model is characterized by the area of interest D is formed by a partition of sub-areas d_i such that $\bigcup_{i=1}^{n} d_i = D$ and $\bigcap_{i=1}^{n} d_i = \emptyset$. This partition can be determined by an administrative or natural region (provinces, municipalities,...), although it can also be determined randomly. $\mathbf{E}_d$ and $\mathbf{O}_d$ are expected and observed cases, respectively in each sub-area $d_i \in D$. Then we wil have the log-risk vector $\log \mathbf{r}_d$, overall log-risk vector α_d and spatial random effect vector ξ_d. The steps for fitting a model whose small area is large are illustrated below, for colorectal cancer in Spain with simulated data.

Stage 1: Build disjoint partition.

The disjointed partition is carried out taking as a reference the autonomous regions of Spain.

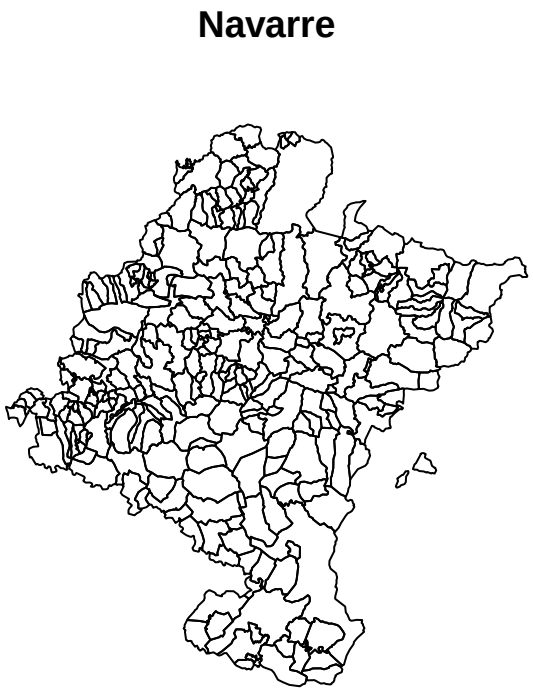
```
#### libraries and data####
library(bigDM)
data("Carto_SpainMUN")
head(Carto_SpainMUN)
#> Simple feature collection with 6 features and 9 fields
#> geometry type:  MULTIPOLYGON
#> dimension:      XY
#> bbox:           xmin: 485318 ymin: 4727428 xmax: 543317 ymax: 4779153
#> CRS:            +proj=utm +zone=30 +ellps=GRS80 +units=m +no_defs
#>      ID                                    name     lat    long     area
#> 1 01001                          Alegria-Dulantzi 4742974 4742974 19913794
#> 2 01002                                   Amurrio 4765886 4765886 96145595
#> 3 01003                                   Aramaio 4766225 4766225 73338806
#> 4 01004                                Artziniega 4775453 4775453 27506468
#> 5 01006                                   Arminon 4729884 4729884 10559721
#> 6 01008 Arrazua-Ubarrundia (San Martin de Ania) 4752714 4752714 57502811
#>   perimeter obs        exp     region                       geometry
#> 1  34372.11   4  3.0237149 Pais Vasco MULTIPOLYGON (((538259 4737...
#> 2  63352.32  27 20.8456682 Pais Vasco MULTIPOLYGON (((503520 4760...
#> 3  41430.46   6  3.7527301 Pais Vasco MULTIPOLYGON (((533286 4759...
```

```
#> 4  22605.22   5  3.2093191 Pais Vasco MULTIPOLYGON (((491260 4776...
#> 5  17847.35   0  0.4817391 Pais Vasco MULTIPOLYGON (((509851 4727...
#> 6  64968.81   1  1.9643891 Pais Vasco MULTIPOLYGON (((534678 4746...

#### disjoint partition ####
carto.d <- divide_carto(carto=Carto_SpainMUN, ID.group="region", k=0)
data.d <- lapply(carto.d, function(x) sf::st_set_geometry(x, NULL))
```

The returned object is list with 15 elements composed with the autonomus regions except insular zone, Ceuta and Melilla. Plotting Navarre region:

```
plot(carto.d[[14]]$geometry, main = "Navarre")
```



Stage 2: Object's neighbours and spatial neighbourhood matrix

In this step, we obtain a neighbours list of class `nb` and its associated spatial adjacency matrix computed by adding to isolated areas its nearest neighbour for each region. LCAR prior has been considered, $\xi_a \sim N(\mathbf{0}, \mathbf{Q}_{\xi_d}^{-1})$, with $\mathbf{Q}_{\xi_d} = \tau_{\xi_d}[\lambda_{\xi_d}(D_W - W) + (1 - \lambda_{\xi_d})I_{n_d}]$. The matrix $\mathbf{Q}_{\xi_d}$ is computed as follows:

```
#### list of object's neighbours and spatial neighbourhood matrix ####
library(Matrix)
aux <- lapply(carto.d, function(x) connect_subgraphs(x, ID.area = "ID"))
#> Searching for isolated areas:
#>   0 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
#>  No disjoint connected subgraphs
#> Searching for isolated areas:
#>   0 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
#>  No disjoint connected subgraphs
#> Searching for isolated areas:
```

```
#>   0 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
#>  No disjoint connected subgraphs
#> Searching for isolated areas:
#>   1 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
#>  No disjoint connected subgraphs
#> Searching for isolated areas:
#>   0 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
#>  No disjoint connected subgraphs
#> Searching for isolated areas:
#>   0 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
#>   2 disjoint connected subgraphs
#> Searching for isolated areas:
#>   1 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
#>  No disjoint connected subgraphs
#> Searching for isolated areas:
#>   0 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
#>   2 disjoint connected subgraphs
#> Searching for isolated areas:
#>   0 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
#>  No disjoint connected subgraphs
#> Searching for isolated areas:
#>   0 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
#>  No disjoint connected subgraphs
#> Searching for isolated areas:
#>   0 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
#>  No disjoint connected subgraphs
#> Searching for isolated areas:
#>   0 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
#>  No disjoint connected subgraphs
#> Searching for isolated areas:
#>   0 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
```

```
#>  No disjoint connected subgraphs
#> Searching for isolated areas:
#>   1 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
#>  No disjoint connected subgraphs
#> Searching for isolated areas:
#>   0 region(s) with no links
#>
#> Searching for disjoint connected subgraphs:
#>  No disjoint connected subgraphs
Wd <- lapply(aux, function(x) x$W)
nd <- lapply(Wd, function(x) nrow(x))
Rd <- mapply(function(x,y){Diagonal(x,colSums(y))-y}, x=nd, y=Wd)
Rd.Leroux <- mapply(function(x,y){Diagonal(x)-y}, x=nd, y=Rd)
```

Stage 3: Fit of the models

The datasets for fit the models are as follows:

```
#### List of datasets for INLA ####
data.INLA <-
  mapply(function(x, y) {
    data.frame(
      O = x[, "obs"],
      E = x[, "exp"],
      Area = x[, "ID"],
      ID.area = seq(1, y)
    )
  },
  x = data.d,
  y = nd,
  SIMPLIFY = FALSE)
D <- length(data.INLA)
```

The hyperparameter prior assumed:

```
#### Priors distributions of hiperparameters ####
sdunif = "expression: logdens=-log_precision/2; return(logdens)"
lunif = "expression: a = 1; b = 1; beta = exp(theta)/(1+exp(theta));
        logdens = lgamma(a+b)-lgamma(a)-lgamma(b)+(a-1)*log(beta)+(b-1)*log(1-beta);
        log_jacobian = log(beta*(1-beta));
        return(logdens+log_jacobian)"
```

We fit the model selection criteria in a classic way,

Stage 4: Merging disjoint models

The merging of disjoint models is carried out with the function `mergeINLA`. We have merge the models in three ways:

Computing only the models selection criteria By default, the function extracts the model selection criteria with `compute.DIC = TRUE`.

```
#### Merge models ####
disjoint <-
```

```
  mergeINLA(
    inla.models = Models,
    k=0,
    ID.area = "Area",
    O = "O",
    E = "E",
    seed=1234,
    compute.fixed = FALSE,
    compute.DIC = TRUE
  )

#### Computational times ####
disjoint$cpu.used
#>       Pre   Running      Post   Merging     Total
#> 16.016739 36.954650  6.090857 23.686000 82.748246

#### DIC ####
disjoint$dic$dic
#> [1] 28205.62

#### WAIC ####
disjoint$waic$waic
#> [1] 28181.98
```

Aditionaly computing the overall fixed effect. To compute the overall fixed effect `compute.fixed=TRUE`.

```
#### Merge models with \alpha ####
disjoint <- mergeINLA(
  inla.models = Models,
  k = 0,
  ID.area = "muni",
  O = "O",
  E = "E",
  seed = 1234,
  n.sample = 1000,
  compute.fixed = TRUE,
  compute.DIC = TRUE
)
summary(disjoint)
```

Immediately without steps by steps with `Car_INLA`. The `Car_INLA` function allows without the previous steps to obtain a merged model with several variants in the parameters. It is possible to retrieve elements from the previous steps such as save the sub-models with `save.models = TRUE`. The `prior = "Leroux"` specifies the prior distribution considered for the spatial random effect in this case Leroux et. al.(2000), but we can choose another: "intrinsic", "BYM" or "BYM2". The `PCpriors` is a functionality is developed in Simpson et. al. (2017) for scaling intrinsic Gaussian Markov Random Fields (GMRFs). If PC priors are used for the precision of a Gaussian random effect, the parameters U and α must be specified so that $P(\sigma > U) = \alpha$ or $P(\lambda_\xi > U) = \alpha$, as appropriate. If `PCpriors = TRUE`, the spatial random effects must be "intrinsic" or "BYM2", with values $(U, \alpha) = (1, 0.01)$ and $(U, \alpha) = (0.5, 0.5)$.

```
disjoint_1 <- CAR_INLA(
  carto = Carto_SpainMUN,
  ID.area = "ID",
  ID.group = "region",
```

```
  O = "obs",
  E = "exp",
  prior = "Leroux",
  model = "partition",
  k = 0,
  seed = 1234,
  strategy = "gaussian",
  PCpriors = FALSE,
  compute.fixed = FALSE,
  compute.DIC = TRUE,
  save.models = FALSE
)
#> STEP 1: Pre-processing data
#> STEP 2: Fitting partition (k=0) model with INLA
#> + Model 1 of 15
#> + Model 2 of 15
#> + Model 3 of 15
#> + Model 4 of 15
#> + Model 5 of 15
#> + Model 6 of 15
#> + Model 7 of 15
#> + Model 8 of 15
#> + Model 9 of 15
#> + Model 10 of 15
#> + Model 11 of 15
#> + Model 12 of 15
#> + Model 13 of 15
#> + Model 14 of 15
#> + Model 15 of 15
#> STEP 3: Merging the results

#### Computational times ####
disjoint_1$cpu.used
#>       Pre   Running      Post   Merging     Total
#> 15.869339 36.742623  4.425221 20.712000 77.749184

#### DIC ####
disjoint_1$dic$dic
#> [1] 28205.59

#### WAIC ####
disjoint_1$waic$waic
#> [1] 28181.96
```

K-order neighbours model

This model is characterized by the area of interest D is formed by a partition of sub-areas d_i such that $\bigcup_{i=1}^{n} d_i = D$ and $\bigcap_{i=1}^{n} d_i \neq \emptyset$. This implies that there are regions with multiple estimates. Below is how to extract the cartography list.

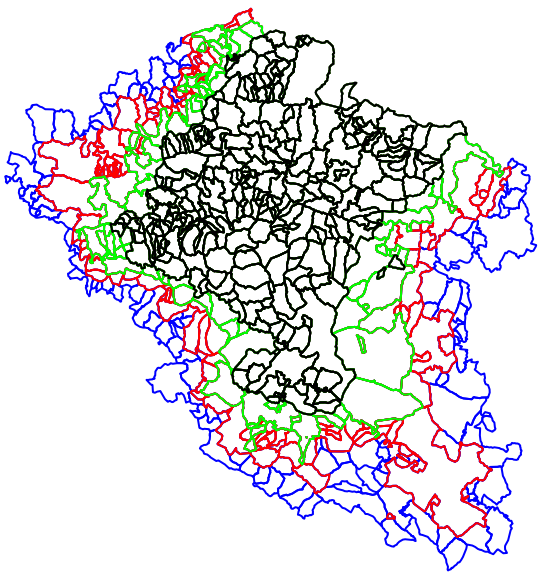
```
#### 1st order neighbours ####
a <-
  divide_carto(
    carto = Carto_SpainMUN,
```

```
    ID.group = "region",
    k = 1,
    plot = FALSE
  )

#### 2nd order neighbours ####
b <-
  divide_carto(
    carto = Carto_SpainMUN,
    ID.group = "region",
    k = 2,
    plot = FALSE
  )

#### 3rd order neighbours ####
c <-
  divide_carto(
    carto = Carto_SpainMUN,
    ID.group = "region",
    k = 3,
    plot = FALSE
  )

#### Navarre with 1st, 2nd and 3rd order neighbours ####
plot(c[[14]]$geometry, border = "blue",main="Navarre with 1st, 2nd and 3rd order neighbours")
plot(b[[14]]$geometry, border = "red", add = TRUE)
plot(a[[14]]$geometry, border = "green", add = TRUE)
plot(carto.d[[14]]$geometry, add = TRUE)
```



To fit the models, we have the following:

```
#### 1st order neighbours model ####
order1<-CAR_INLA(
carto = Carto_SpainMUN,
ID.area = "ID",
ID.group = "region",
O = "obs",
E = "exp",
prior = "Leroux",
model = "partition",
k = 1,
strategy = "gaussian")

#### 2nd order neighbours model ####
order2<-CAR_INLA(
carto = Carto_SpainMUN,
ID.area = "ID",
ID.group = "region",
O = "obs",
E = "exp",
prior = "Leroux",
model = "partition",
k = 2,
strategy = "gaussian")

#### 3rd order neighbours model ####
order3<-CAR_INLA(
carto = Carto_SpainMUN,
ID.area = "ID",
ID.group = "region",
O = "obs",
E = "exp",
prior = "Leroux",
model = "partition",
k = 3,
strategy = "gaussian")
```

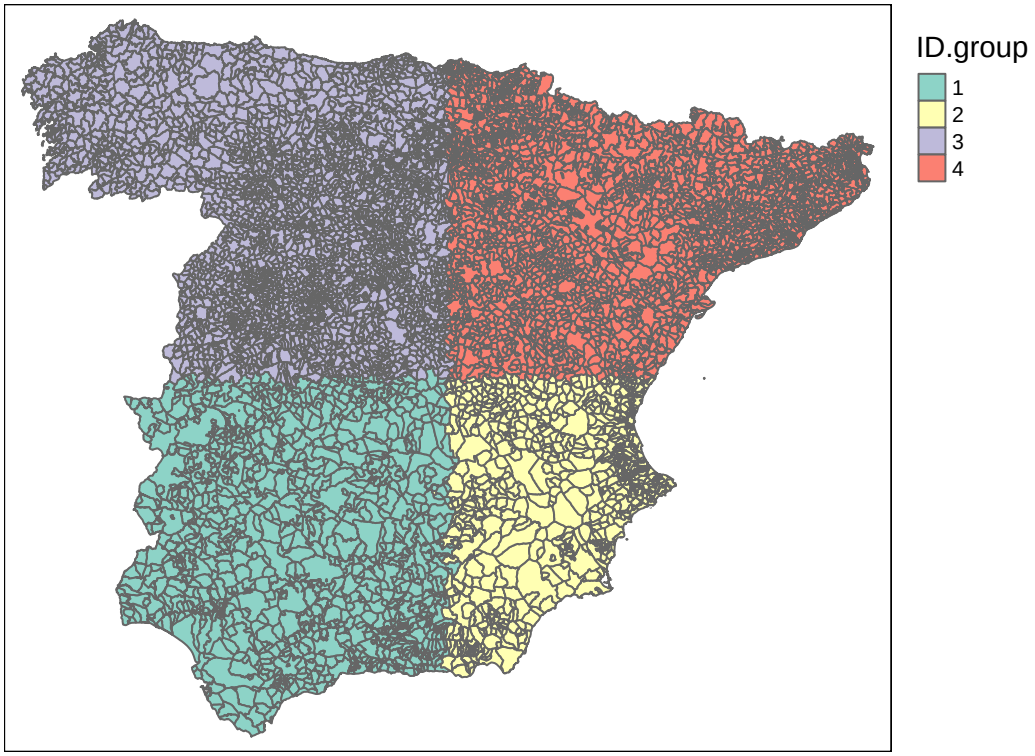
Random partition

Many times, it is not possible to divide by a natural or administrative region. In this case, a random grid partition could be generated by specifying the desired rows and columns.

```
library(tmap)

## define a random partition based on a 2x2 regular grid ##
carto_random <- random_partition(carto=Carto_SpainMUN, rows=2, columns=2)

## plot of the grouping variable 'ID.group' ##
tm_shape(carto_random) +
tm_polygons(col="ID.group") +
  tm_layout(legend.outside=TRUE)
```



References

Leroux, Brian G., Xingye Lei, and Norman Breslow. 2000. "Estimation of Disease Rates in Small Areas: A New Mixed Model for Spatial Dependence." In *Statistical Models in Epidemiology, the Environment, and Clinical Trials*, 179–91. Springer New York. https://doi.org/10.1007/978-1-4612-1284-34.

Simpson, Daniel, Håvard Rue, Andrea Riebler, Thiago G. Martins, and Sigrunn H. Sørbye. 2017. "Penalising Model Component Complexity: A Principled, Practical Approach to Constructing Priors." *Statist. Sci.* 32 (1): 1–28. https://doi.org/10.1214/16-STS576.