bigDM

Erick Orozco-Acosta, Aritz Adin and María Dolores Ugarte

6/25/2020

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 Colombian case, if you download the 'Nivel Geográfico Municipio' in year 2018 from `https://tinyurl.com/srt6xuu` (National Administrative Department Statistical-DANE), the downloaded and uncompressed files are:

- MGN_MPIO_POLITICO.shp
- MGN_MPIO_POLITICO.shx
- MGN_MPIO_POLITICO.sbx
- MGN_MPIO_POLITICO.prj
- MGN_MPIO_POLITICO.cpj
- MGN_MPIO_POLITICO.sbn
- MGN_MPIO_POLITICO.dbf

To convert this cartography in `spatialpolygonsdataframe` and `sf` the following is required.

```
#### libraries ####
library(rgdal)
```

```
## Loading required package: sp
```

```
## rgdal: version: 1.5-10, (SVN revision 1006)
## 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
```

```
library(sf)
```

```
## Linking to GEOS 3.7.2, GDAL 2.4.2, PROJ 5.2.0
```

```
#### import to spatialpolygonsdataframe ####
zones <-
        readOGR(dsn = "/Users/erick.orozco/MGN2018_MPIO_POLITICO/MGN_MPIO_POLITICO.shp",
                layer = "MGN_MPIO_POLITICO")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "/Users/erick.orozco/MGN2018_MPIO_POLITICO/MGN_MPIO_POLITICO.shp", layer: "MGN_MPIO_POLITICO
## with 1122 features
## It has 10 fields
```

```
#### convert spatialpolygonsdataframe in sf ####
zones <- st_as_sf(zones)
```

Now, is possible convert the `.shp` files in `sf` directly:

```
#### library ####
library(sf)

#### import to sf ####
zones <-
        st_read(dsn = "/Users/erick.orozco/MGN2018_MPIO_POLITICO/MGN_MPIO_POLITICO.shp",
                layer = "MGN_MPIO_POLITICO")
```

```
## Reading layer `MGN_MPIO_POLITICO' from data source `/Users/erick.orozco/MGN2018_MPIO_POLITICO/MGN_MPI
## Simple feature collection with 1122 features and 10 fields
```

```
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -81.73562 ymin: -4.229406 xmax: -66.84722 ymax: 13.39473
## CRS:            4326
```

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 ####
library(readr)
data <- read_delim("data.txt",
                   "\t",
                   escape_double = FALSE,
                   col_types = cols(ID = col_character(),
                                    exp = col_number(),
                                    obs = col_number()),
                   trim_ws = TRUE)

#### Merging data with cartography ####
zones<- merge (zones, data, by.x="MPIO_CCNCT", by.y="ID")
```

Disjoint model

This model is characterized by the area of interest A is formed by a partition of sub-areas a_i such that $\bigcup_{i=1}^{n} a_i = A$ and $\bigcap_{i=1}^{n} a_i = \emptyset$. This partition can be determined by an administrative or natural region (provinces, municipalities,...), although it can also be determined randomly. $\mathbf{E}_a$ and $\mathbf{O}_a$ are expected and observed cases, respectively in each sub-area $a_i \in A$. Then we wil have the log-risk vector $\log \mathbf{r}_a$, overall log-risk vector α_a and spatial random effect vector ξ_a. 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 ####
regions <-
  divide_carto(
    carto = Carto_SpainMUN,
    ID.area = "ID",
    ID.group = "region",
    k = 0,
    plot = FALSE
  )
```

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

```
plot(regions[[8]]$geometry, main = "Valencian community")
```

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.

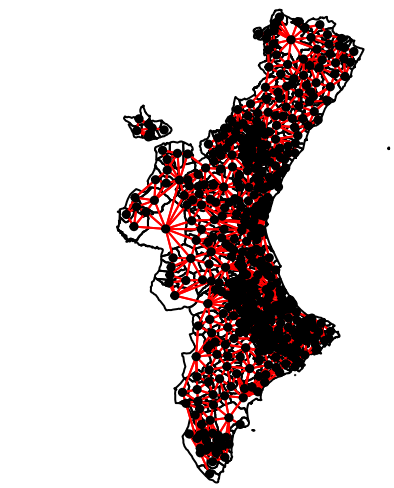
```
#### list of object's neighbours and spatial neighbourhood matrix ####
regions.nb <-lapply(regions, function(x)
  add_neighbour(x, plot = FALSE))
```

```
##   0 region(s) with no links
##   0 region(s) with no links
##   0 region(s) with no links
##   1 region(s) with no links
##   0 region(s) with no links
##   0 region(s) with no links
##   1 region(s) with no links
##   0 region(s) with no links
##   0 region(s) with no links
```

```
##   0 region(s) with no links
##   0 region(s) with no links
##   0 region(s) with no links
##   0 region(s) with no links
##   1 region(s) with no links
##   0 region(s) with no links
```

This output show that there are sub-areas without neighbours. Now, we plotted the neighbors of each area for the Valencian community with no-conected graphs.

```
plot(regions[[8]]$geometry, main="Original neighbourhood graph")
plot(regions.nb[[8]][[1]], st_centroid(st_geometry(regions[[8]]), of_largest_polygon=TRUE),
pch=19, cex=0.5, col="red", add=TRUE)
```

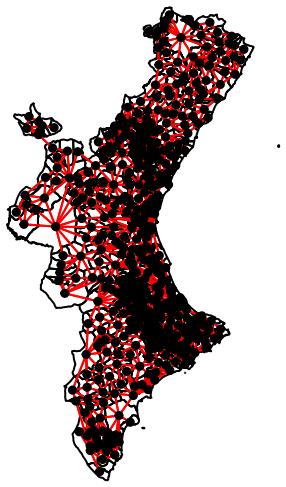


Stage 3: Merge disjoint graphs (if required)

To solve the problem about the no-conected graphs or no neighbous zones, we need to use the function `conect_subgraphs`.

This output shows that all 15 sub-regions have their connected graphs. Here is the plot again of the Valencian community with the no-connected graph merged.

```
plot(regions[[8]]$geometry, main="Modified neighbourhood graph")
plot(regions.nb[[8]][[1]], st_centroid(st_geometry(regions[[8]]), of_largest_polygon=TRUE),
pch=19, cex=0.5, col="red", add=TRUE)
```



Stage 4: Fit of the models

To fit the models the LCAR prior has been considered, $\xi_a \sim N(\mathbf{0}, \mathbf{Q}_{\xi_a}^{-1})$, with $\mathbf{Q}_{\xi_a} = \tau_{\xi_a}[\lambda_{\xi_a}(D_W - W) + (1-\lambda_{\xi_a})I_{n_a}]$. The matrix $\mathbf{Q}_{\xi_a}$ is computed as follows:

```
#### Cmatrix list for INLA ####
Q<-lapply(c(1:D), function(x) {diag(dim(regions.nb[[x]][[2]])[1])-regions.nb[[x]][[2]]})
```

The datasets for fit the models are as follows:

```
#### List of datasets for INLA ####
data.inla <- lapply(regions, function(x)
  data.frame(
    O = x$obs,
    E = x$exp,
    ID.area = seq(1, dim(x)[1]),
    muni = x$ID
  ))
```

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)"
```

Fitting the models with the model selection criteria

```
#### Fit of the models ####
library(INLA)
models<-vector("list",D)
for (i in 1:D) {
  Q.leorux<-Q[[i]]
  formula <- O ~ f(ID.area, model="generic1", Cmatrix=Q.leorux, constr=TRUE,
                   hyper=list(prec=list(prior=sdunif),beta=list(prior=lunif,initial=0)))
  models[[i]] <- inla(formula, family="poisson", data=data.inla[[i]], E=E,
```

```
                      control.predictor=list(compute=TRUE,cdf=c(log(1))),
                      control.compute=list(dic=TRUE, cpo=TRUE, waic=TRUE, config=TRUE),
                      control.inla=list(strategy="gaussian"))
}
```

Stage 5: Merging disjoint models

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

```
#### Merge models ####
disjoint <-
  mergeINLA(
    inla.models = models,
    k = 0,
    ID.area = "muni",
    O = "O",
    E = "E",
    seed = 20191127
  )

#### Computational times ####
disjoint$cpu.used
```

Computing only the models selection criteria

```
##       Pre   Running      Post   Merging     Total
## 18.774858 43.985599  6.386464 24.720000 93.866921
```

```
#### DIC ####
disjoint$dic$dic
```

```
## [1] 28333.62
```

```
#### WAIC ####
disjoint$waic$waic
```

```
## [1] 28334
```

Aditionaly computing the overall fixed effect.

```
disjoint<-CAR_INLA(
carto = Carto_SpainMUN,
ID.area = "ID",
ID.group = "region",
O = "obs",
E = "exp",
prior = "Leroux",
model = "partition",
k = 0,
strategy = "gaussian")
```

Immediately without steps by steps with `CarINLA`.

```
## 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$cpu.used
```

```
##       Pre   Running      Post   Merging     Total
## 18.749560 41.080498  5.158764 23.353000 88.341822
```

```
#### DIC ####
disjoint$dic$dic
```

```
## [1] 28204.78
```

```
#### WAIC ####
disjoint$waic$waic
```

```
## [1] 28181.6
```

K-order neighbours model

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

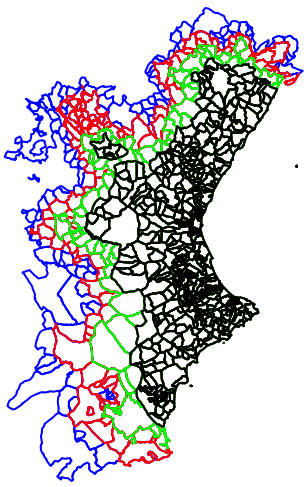
```
#### 1st order neighbours ####
a <-
  divide_carto(
    carto = Carto_SpainMUN,
    ID.group = "region",
    k = 1,
    plot = FALSE
  )

#### 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
  )

#### Valencian community with 1st, 2nd and 3rd order neighbours ####
plot(c[[8]]$geometry, border = "blue",main="Valencian community with 1st, 2nd and 3rd order neighbours")
plot(b[[8]]$geometry, border = "red", add = TRUE)
plot(a[[8]]$geometry, border = "green", add = TRUE)
plot(regions[[8]]$geometry, add = TRUE)
```



To fit the models, we have the following:

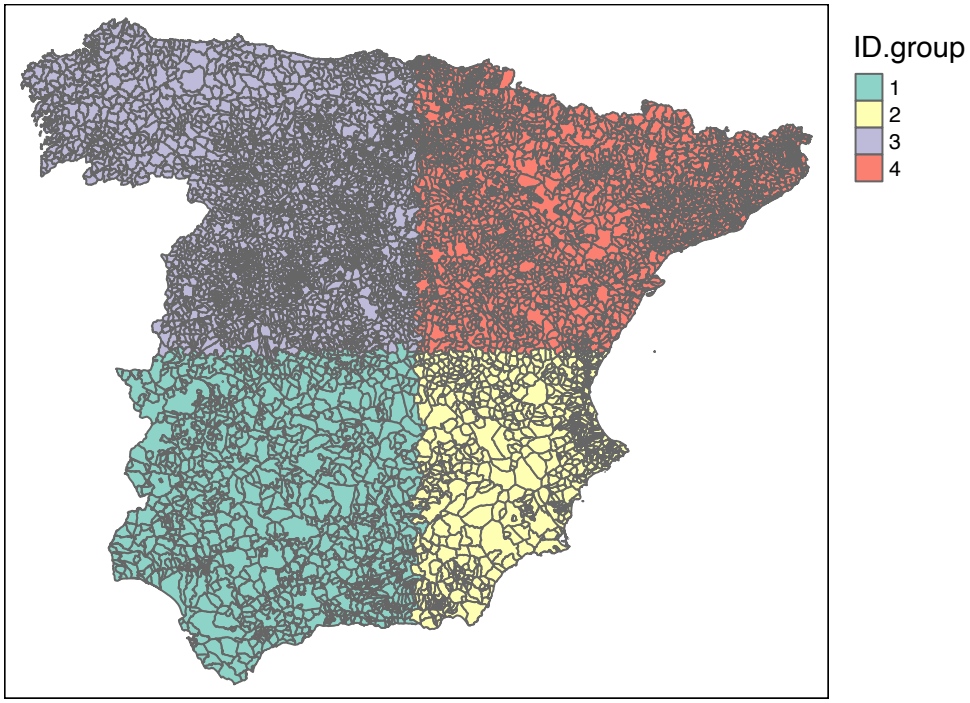
Random partition

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.rp <- random_partition(carto=Carto_SpainMUN, rows=2, columns=2)

## plot of the grouping variable 'ID.group' ##
tm_shape(carto.rp) +
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.