Example 4.3

Disease mapping: from foundations to multidimensional modeling

Martinez-Beneito M.A. and Botella-Rocamora P.

This document reproduces the analysis made at Example 4.3 of the book: "Disease mapping: from foundations to multidimensional modeling" by Martinez-Beneito M.A. and Botella-Rocamora P., published by CRC press in 2019. You can watch the analysis made with full detail at this pdf document, or even execute it if you want with the material available at https://github.com/MigueBeneito/DMBook. Anyway, this pdf file should be enough for following most of the details of the analysis made for this example.

The statistical analysis below has been run in `R`, by additionally using the library `Rmarkdown`, so be sure that you have this software installed if you want to reproduce by yourself the content of this document. In that case we advise you to download first the annex material at https://github.com/MigueBeneito/DMBook, open with `Rstudio` the corresponding `.Rproj` file that you will find at the folder corresponding to this example and compile the corresponding `.Rmd` document. This will allow you to reproduce the whole statistical analysis below.

This document has been executed with real data that are not provided in order to preserve their confidentiality. Slightly modified data are provided instead, as described in Chapter 1 of the book. Thus, when reproducing this document you will not obtain exactly the same results, although they should be very close to those shown here.

Libraries and data loading

```r
# Libraries loading
#-----------------
if (!require(RColorBrewer)) {
    install.packages("RColorBrewer")
    library(RColorBrewer)
}
if (!require(rgdal)) {
    install.packages("rgdal")
    library(rgdal)
}
if (!require(INLA)) {
    install.packages("INLA", repos = c(getOption("repos"), INLA = "https://inla.r-inla-download.org/R/s
        dep = TRUE)
    library(INLA)
}
if (!require(pbugs)) {
    if (!require(devtools)) {
        install.packages("devtools")
        devtools::install_github("fisabio/pbugs")
    } else {
        install_github("fisabio/pbugs")
    }
}

# Data loading
#------------
# For reproducing the document, the following line should be changed to
# load('../Data/ObsOral-mod.Rdata') since that file contains the
# modified data making it possible to reproduce this document.
load("../Data/ObsOral.Rdata")
# load('../Data/ObsOral-mod.Rdata')
load("../Data/ExpOral.Rdata")
load("../Data/VR.Rdata")
```

Grid of knots for defining the Gaussian basis of functions

```r
# Centroids of the municipalities in the Valencian regions
centroids <- t(mapply(VR.cart@polygons, FUN = function(x) {
    apply(x@Polygons[[1]]@coords, 2, mean)
}))
# summaries of the centroids
c(min(centroids[, 1]), max(centroids[, 1]))/1000
```

```
## [1] 631.3658 791.6110
```

```r
c(min(centroids[, 2]), max(centroids[, 2]))/1000
```

```
## [1] 4197.586 4513.863
```

```r
max(centroids[, 1]) - min(centroids[, 1])/1000
```

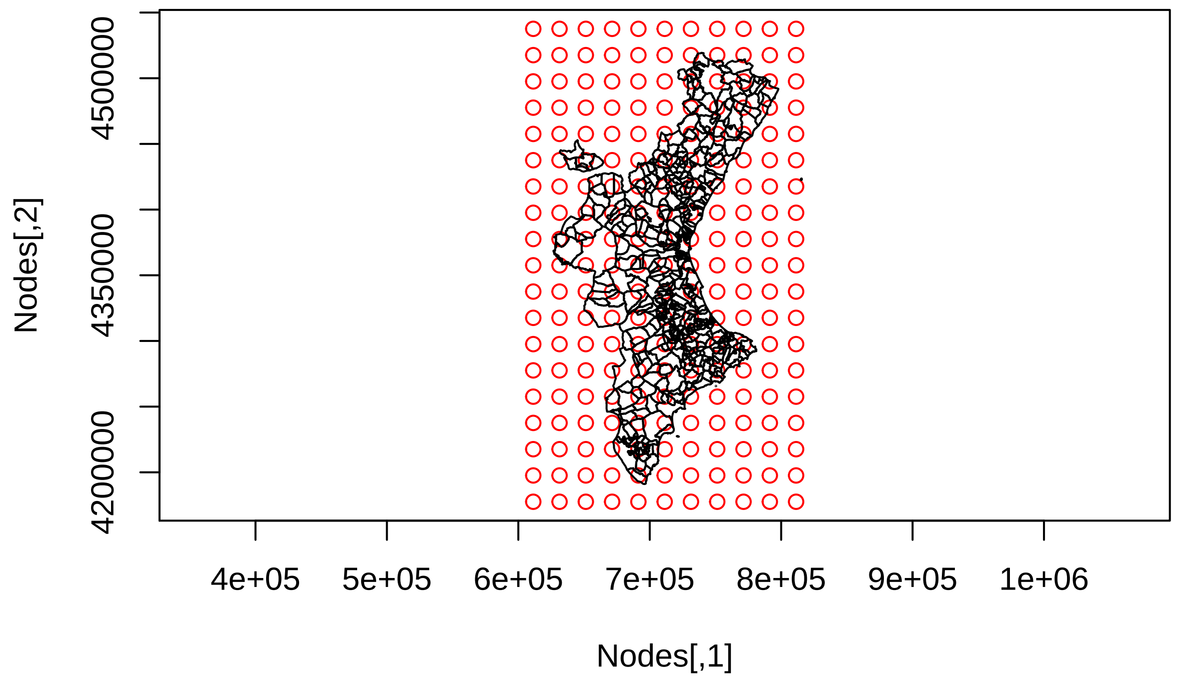
```
## [1] 790979.7
```

```r
max(centroids[, 2]) - min(centroids[, 2])/1000
```

```
## [1] 4509665
```

```r
# The Valencian region is approximately twice taller than wider.
# According to these summaries we are going to consider the following
# grid of knots: Knots will be separated 20 kilometers for both
# coordinate axes. Thus we will consider 9(+2 outer knots) for the
# longitudes and 17(+2 outer knots) for the latitudes. The outer knots
# are intended to avoid edge effects in the analysis.
NodesX <- min(centroids[, 1]) + ((-1):9) * 20000
NodesY <- min(centroids[, 2]) + ((-1):17) * 20000
Nodes <- cbind(rep(NodesX, 19), rep(NodesY, each = 11))

# Plot of the grid of knots and the Valencian Region
plot(Nodes, col = "red", asp = 1)
plot(VR.cart, add = TRUE)
```



Trimming of the grid of knots for improving the computational performance of the MCMC

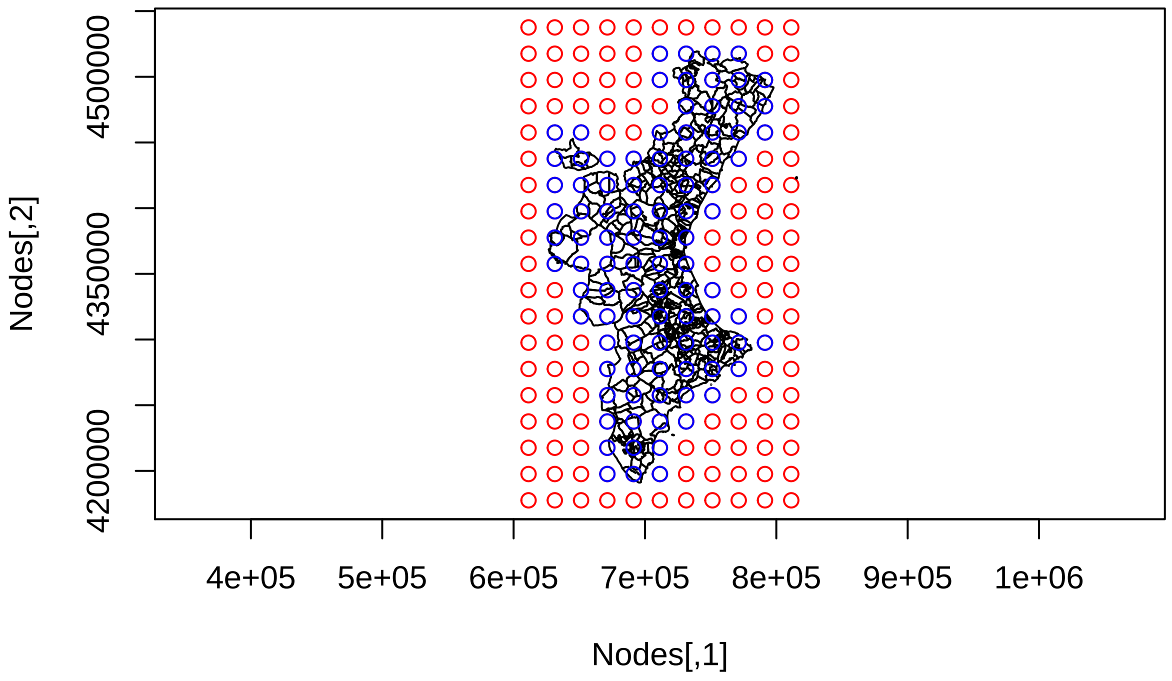
```r
# Distances between muncicipalities (centroids) and knots. distance=1
# corresponds to locations placed at a knot of distance (20 kilometers)
distances <- matrix(nrow = 540, ncol = 19 * 11)
for (j in 1:(19 * 11)) {
    distances[, j] <- sqrt((centroids[, 1] - Nodes[j, 1])^2 + (centroids[,
        2] - Nodes[j, 2])^2)/20000
}

# Trimming of the grid of knots. We keep just those knots having a
# municipality closer than 20 kilometers away. This improves the
# computational performance of the MCMC.
distances2 <- distances[, which(apply(distances, 2, function(x) {
    any(x < 1)
}))]
# Final number of knots for the basis of functions
ncol(distances2)
```

```
## [1] 95
```

```r
# Plot of the trimmed knots
plot(Nodes, col = "red", asp = 1)
plot(VR.cart, add = TRUE)
```

```
points(Nodes[which(apply(distances, 2, function(x) {
    any(x < 1)
})), ], col = "blue")
```



Fixed effects Gaussian Basis model

```
# Fixed effects model
ModelFixedEf <- function() {
    for (i in 1:n) {
        O[i] ~ dpois(lambda[i])
        log(lambda[i]) <- log(E[i]) + log.theta[i]
        log.theta[i] <- mu + inprod2(basis[i, ], gamma[])
        sSMR[i] <- 100 * exp(log.theta[i])
        P.sSMR[i] <- step(log.theta[i])
    }
    for (j in 1:nbasis) {
        gamma[j] ~ dflat()
    }
    mu ~ dflat()
}

# WinBUGS call to the model above
data <- list(O = Obs.muni, E = Exp.muni, basis = round(exp(-distances2^2),
    10), n = 540, nbasis = ncol(distances2))
```

```
inits <- function() {
    list(gamma = rnorm(data$nbasis), mu = rnorm(1, 0, 0.3))
}
param <- c("sSMR", "mu", "gamma")
# BEWARE: This sentence may take quite a long time to run (hours).
ResulFE <- pbugs(data = data, inits = inits, param = param, model.file = ModelFixedEf,
    n.iter = 55000, n.burnin = 5000, bugs.seed = 1)
ResulFE$exec_time
```

```
## Time difference of 4.597479 hours
```

```
# Convergence checking
summary(ResulFE$summary[, "Rhat"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9995  1.0019  1.0056  1.0293  1.0152  1.4446
```

```
summary(ResulFE$summary[, "n.eff"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     8.0   140.0   380.0   492.4  1000.0  1000.0
```

Fixed effects Gaussian Basis model

```
# Random effects model
ModelRandomEf <- function() {
    for (i in 1:n) {
        O[i] ~ dpois(lambda[i])
        log(lambda[i]) <- log(E[i]) + log.theta[i]
        log.theta[i] <- mu + inprod2(basis[i, ], gamma[])
        sSMR[i] <- 100 * exp(log.theta[i])
        P.sSMR[i] <- step(log.theta[i])
    }
    for (j in 1:nbasis) {
        gamma[j] ~ dnorm(0, tau)
    }
    mu ~ dflat()
    tau <- pow(sd.g, -2)
    sd.g ~ dunif(0, 10)
}

# WinBUGS call to the model above
data <- list(O = Obs.muni, E = Exp.muni, basis = round(exp(-distances2^2),
    10), n = 540, nbasis = ncol(distances2))
inits <- function() {
    list(gamma = rnorm(data$nbasis), mu = rnorm(1, 0, 0.3), sd.g = runif(1))
}
param <- c("sSMR", "mu", "sd.g", "gamma")
ResulRE <- pbugs(data = data, inits = inits, param = param, model.file = ModelRandomEf,
    n.iter = 5500, n.burnin = 500, bugs.seed = 1)
ResulRE$exec_time
```

```
## Time difference of 27.38678 mins
```

```r
# Convergence checking
summary(ResulRE$summary[, "Rhat"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.9995  1.0003  1.0015  1.0020  1.0030  1.0107
```

```r
summary(ResulRE$summary[, "n.eff"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   180.0   612.5  1000.0   821.6  1000.0  1000.0
```

Fitting of the random effects model in INLA

```r
# Uniform prior on the standard deviation of random effects
sdunif = "expression:
\tlogdens = -log_precision/2;
\treturn(logdens)"

data = data.frame(O = Obs.muni, E = Exp.muni, id.node = 1:540)
basis <- round(exp(-distances2^2), 10)
form = O ~ f(id.node, model = "z", Z = basis, hyper = list(prec = list(prior = sdunif)))
resul.INLA = inla(form, family = "poisson", data = data, E = E, control.compute = list(dic = TRUE))
```

```
## Warning in inla.model.properties.generic(inla.trim.family(model), (mm[names(mm) == : Model 'z' in se
##   Use this model with extra care!!! Further warnings are disabled.
```

```r
# Computing time
resul.INLA
```

```
## 
## Call:
## c("inla(formula = form, family = \"poisson\", data = data, E = E, ",  "    control.compute = list(di
## 
## Time used:
##  Pre-processing    Running inla Post-processing           Total 
##        1.383235        2.443830        0.127068        3.954133 
## 
## Integration Strategy: Model contains 1 hyperparameters 
## The model contains 1 fixed effect (including a possible intercept)
## 
## Likelihood model: poisson
## 
## The model has 1 random effects:
## 1.'id.node' is a Z model
```

```r
# random effects model with the most accurate strategy='laplace' option
resul.INLA2 = inla(form, family = "poisson", data = data, E = E, control.compute = list(dic = TRUE),
    control.inla = list(strategy = "laplace"))
# Computing time
resul.INLA2
```

```
## 
## Call:
## c("inla(formula = form, family = \"poisson\", data = data, E = E, ",  "    control.compute = list(di
## 
```

```
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##       0.9615881      17.2434180       0.1191969      18.3242030
##
## Integration Strategy: Model contains 1 hyperparameters
## The model contains 1 fixed effect (including a possible intercept)
##
## Likelihood model: poisson
##
## The model has 1 random effects:
## 1.'id.node' is a Z model
```

DIC comparison of the models fitted

The function below computes the DIC criterion for disease mapping models fitted with `WinBUGS`. It returns DIC values comparable to those reported by `INLA`, in contrast to `WinBUGS`.

```
# Arguments: Simu.sSMRs: matrix of dimensions n.IterXn.Units where
# n.Iter are the number of MCMC iterations saved and n.Units the number
# of spatial units in the analysis. You will typically find this as a
# submatrix of the sims.matrix element of any bugs object. O: Vector of
# length n.Units with the observed deaths per spatial unit. E: Vector
# of length n.Units with the expected deaths per spatial unit.
DICPoisson = function(Simu.sSMRs, O, E) {
    mu = t(apply(Simu.sSMRs/100, 1, function(x) {
        x * E
    }))
    D = apply(mu, 1, function(x) {
        -2 * sum(O * log(x) - x - lfactorial(O))
    })
    Dmean = mean(D)
    mumean = apply(Simu.sSMRs/100, 2, mean) * E
    DinMean = -2 * sum(O * log(mumean) - mumean - lfactorial(O))
    # if(save==TRUE){return(c(Dmedia,Dmedia-DenMedia,2*Dmedia-DenMedia))}
    cat("D=", Dmean, "pD=", Dmean - DinMean, "DIC=", 2 * Dmean - DinMean,
        "\n")
}
DICPoisson(ResulRE$sims.matrix[, grep("sSMR", dimnames(ResulRE$sims.matrix)[[2]])],
    Obs.muni, Exp.muni)
```

```
## D= 1761.272 pD= 29.14557 DIC= 1790.417
```

```
DICPoisson(ResulFE$sims.matrix[, grep("sSMR", dimnames(ResulFE$sims.matrix)[[2]])],
    Obs.muni, Exp.muni)
```

```
## D= 1746.605 pD= 98.04447 DIC= 1844.65
```

```
resul.INLA$dic$dic
```

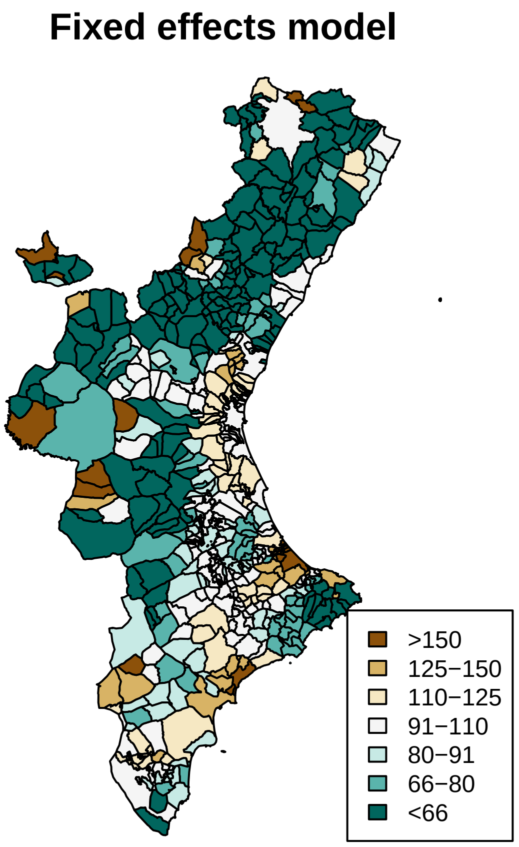
```
## [1] 1792.151
```

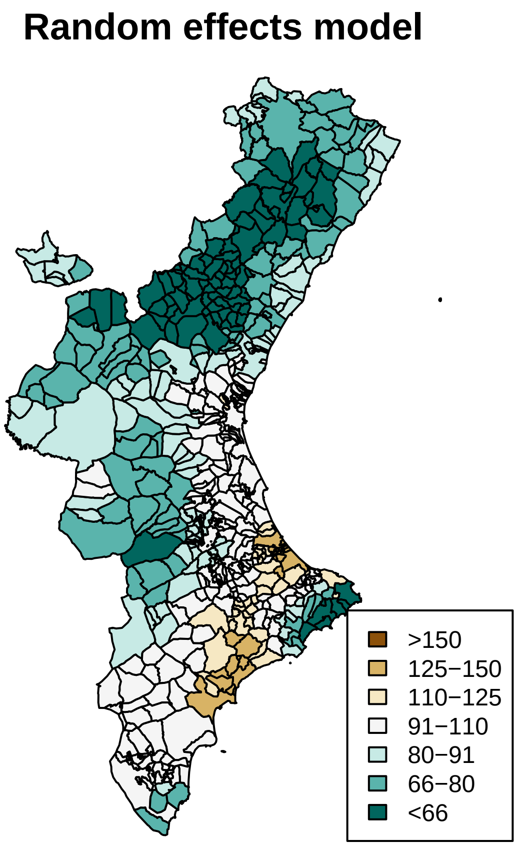
```
resul.INLA2$dic$dic
```

```
## [1] 1791.892
```

Choropleth maps

```r
colors <- brewer.pal(7, "BrBG")[7:1]
par(mfrow = c(1, 2))
par(mar = c(0, 0, 1, 0) + 0.1)
plot(VR.cart, col = colors[as.numeric(cut(ResulFE$mean$sSMR, 100 * c(-0.1,
    1/1.5, 1/1.25, 1/1.1, 1.1, 1.25, 1.5, 100)))])
title("Fixed effects model", cex = 0.75)
legend(x = "bottomright", fill = colors[7:1], legend = c(">150", "125-150",
    "110-125", "91-110", "80-91", "66-80", "<66"), cex = 0.75, inset = 0.03)
plot(VR.cart, col = colors[as.numeric(cut(ResulRE$mean$sSMR, 100 * c(-0.1,
    1/1.5, 1/1.25, 1/1.1, 1.1, 1.25, 1.5, 100)))])
title("Random effects model", cex = 0.75)
legend(x = "bottomright", fill = colors[7:1], legend = c(">150", "125-150",
    "110-125", "91-110", "80-91", "66-80", "<66"), cex = 0.75, inset = 0.03)
```


Fixed effects model
>150
125–150
110–125
91–110
80–91
66–80
<66


Random effects model
>150
125–150
110–125
91–110
80–91
66–80
<66