Example 4.5

Disease mapping: from foundations to multidimensional modeling

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

This document reproduces the analysis made at Example 4.5 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

```
# 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
```

```r
# 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")
```

R function for calculating the DIC criterion 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. See annex material for Example 4.3.

```r
# 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")
}
```

INLA implementation of the BYM model

```r
sdunif = "expression:
  logdens = -log_precision/2;
  return(logdens)"

data = data.frame(O = Obs.muni, E = Exp.muni, id.node = 1:540)
# See the annex material of Example 3.6 for details on how the VR.graph
# file is generated
form = O ~ f(id.node, model = "bym", hyper = list(prec.spatial = list(prior = sdunif),
    prec.unstruct = list(prior = sdunif)), graph = "../Data/VR.graph")
resul.BYM.inla = inla(form, family = "poisson", data = data, E = E, control.compute = list(dic = TRUE))
# Computing time
resul.BYM.inla$cpu.used
```

```
##  Pre-processing    Running inla Post-processing           Total
##       1.7429600       5.9972811       0.3877769       8.1280179
```

INLA implementation of some alternative spatial models

```
# PCAR distributed random effects
formProper = O ~ f(id.node, model = "besagproper", hyper = list(prec = list(prior = sdunif)),
    graph = "../Data/VR.graph")
resul.PCAR.inla <- inla(formProper, 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 'besagpro
##   Use this model with extra care!!! Further warnings are disabled.
```

```
# Computing time
resul.PCAR.inla$cpu.used
```

```
##  Pre-processing    Running inla Post-processing           Total
##       1.2524300       4.8354459       0.1359711       6.2238469
```

```
# HCAR distributed random effects Cmatrix exportation as required by
# INLA
write.table(cbind(rep(1:540, VR.wb$num), VR.wb$adj, rep(1, length(VR.wb$adj))),
    "../Data/Cmat.inla", row.names = FALSE, col.names = FALSE)
formGeneric = O ~ f(id.node, model = "generic1", hyper = list(prec = list(prior = sdunif)),
    Cmatrix = "../Data/Cmat.inla")
resul.HCAR.inla <- inla(formGeneric, family = "poisson", data = data, E = E,
    control.compute = list(dic = TRUE))
# Computing time
resul.HCAR.inla$cpu.used
```

```
##  Pre-processing    Running inla Post-processing           Total
##        1.043143        4.392911        0.112426        5.548480
```

Proper CAR model fit with WinBUGS

```
ModelProperCAR = function() {
    for (i in 1:n) {
        O[i] ~ dpois(lambda[i])
        log(lambda[i]) <- log(E[i]) + log.theta[i]
        log.theta[i] <- sp[i]
        sSMR[i] <- 100 * exp(log.theta[i])
        P.sSMR[i] <- step(sSMR[i] - 100)
    }

    # The proper CAR 'piece'
    sp[1:n] ~ car.proper(mu.sp[], C[], adj[], num[], M[], tau, rho)
    for (i in 1:n) {
        mu.sp[i] <- mu
    }

    # Prior distributions
    rho ~ dunif(rho.low, rho.up)
    rho.low <- min.bound(C[], adj[], num[], M[])
    rho.up <- max.bound(C[], adj[], num[], M[])
    tau <- pow(sd.sp, -2)
    sd.sp ~ dunif(0, 10)
```

```
    mu ~ dflat()
}

data <- list(O = Obs.muni, E = Exp.muni, n = 540, num = VR.wb$num, adj = VR.wb$adj,
    M = 1/VR.wb$num, C = rep(1/VR.wb$num, VR.wb$num))
inits <- function() {
    list(sp = rnorm(540), sd.sp = runif(1), mu = rnorm(1))
}
param <- c("sSMR", "mu", "sd.sp", "rho")
# BEWARE: The following line could take long to run.
time.ProperCAR <- system.time(ResulProperCAR <- pbugs(data = data, inits = inits,
    param = param, model.file = ModelProperCAR, n.iter = 150000, n.burnin = 20000,
    bugs.seed = 1, DIC = FALSE))
# Computing time
ResulProperCAR$exec_time
```

```
## Time difference of 14.31505 mins
```

```
# Result summaries
summary(ResulProperCAR$summary[, "Rhat"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.001   1.001   1.001   1.002   1.024
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     250    1500    2000    1711    2000    2000
```

```
round(ResulProperCAR$summary[c("mu", "sd.sp", "rho"), ], 1)
```

```
##       mean  sd 2.5%  25%  50%  75% 97.5% Rhat n.eff
## mu    -0.1 0.1 -0.3 -0.2 -0.1 -0.1   0.1    1  2000
## sd.sp  0.5 0.1  0.4  0.4  0.5  0.5   0.6    1  2000
## rho    1.0 0.0  0.9  1.0  1.0  1.0   1.0    1  2000
```

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

```
## D= 1704.507 pD= 83.67534 DIC= 1788.182
```

BYM model fit with WinBUGS

```
ModelBYM = function() {
    for (i in 1:n) {
        O[i] ~ dpois(lambda[i])
        log(lambda[i]) <- log(E[i]) + log.theta[i]
        log.theta[i] <- mu + sp[i] + het[i]
        het[i] ~ dnorm(0, tau.het)
        sSMR[i] <- 100 * exp(log.theta[i])
        P.sSMR[i] <- step(sSMR[i] - 100)
    }

    # The intrinsic CAR 'piece'
```

```
    sp[1:n] ~ car.normal(adj[], weights[], num[], tau.sp)
    for (i in 1:nadj) {
        weights[i] <- 1
    }

    # Prior distributions
    tau.het <- pow(sd.het, -2)
    sd.het ~ dunif(0, 10)
    tau.sp <- pow(sd.sp, -2)
    sd.sp ~ dunif(0, 10)
    mu ~ dflat()
}

data <- list(O = Obs.muni, E = Exp.muni, n = 540, num = VR.wb$num, adj = VR.wb$adj,
    nadj = length(VR.wb$adj))
inits <- function() {
    list(sp = rnorm(540), het = rnorm(540), sd.sp = runif(1), sd.het = runif(1),
        mu = rnorm(1))
}
param <- c("sSMR", "mu", "sd.sp", "sd.het")
# BEWARE: The following line could take long to run.
ResulBYM <- pbugs(data = data, inits = inits, param = param, model.file = ModelBYM,
    n.iter = 150000, n.burnin = 20000, bugs.seed = 1, DIC = FALSE)
# Computing time
ResulBYM$exec_time
```

```
## Time difference of 22.34423 mins
```

```
# Result summaries
summary(ResulBYM$summary[, "Rhat"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.001   1.001   1.002   1.002   1.045
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     130    1200    2000    1616    2000    2000
```

```
round(ResulBYM$summary[c("mu", "sd.sp", "sd.het"), ], 1)
```

```
##        mean  sd 2.5%  25%  50%  75% 97.5% Rhat n.eff
## mu     -0.2 0.0 -0.2 -0.2 -0.2 -0.1  -0.1    1  2000
## sd.sp   0.4 0.1  0.3  0.4  0.4  0.5   0.5    1  1200
## sd.het  0.1 0.1  0.0  0.0  0.1  0.1   0.2    1   130
```

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

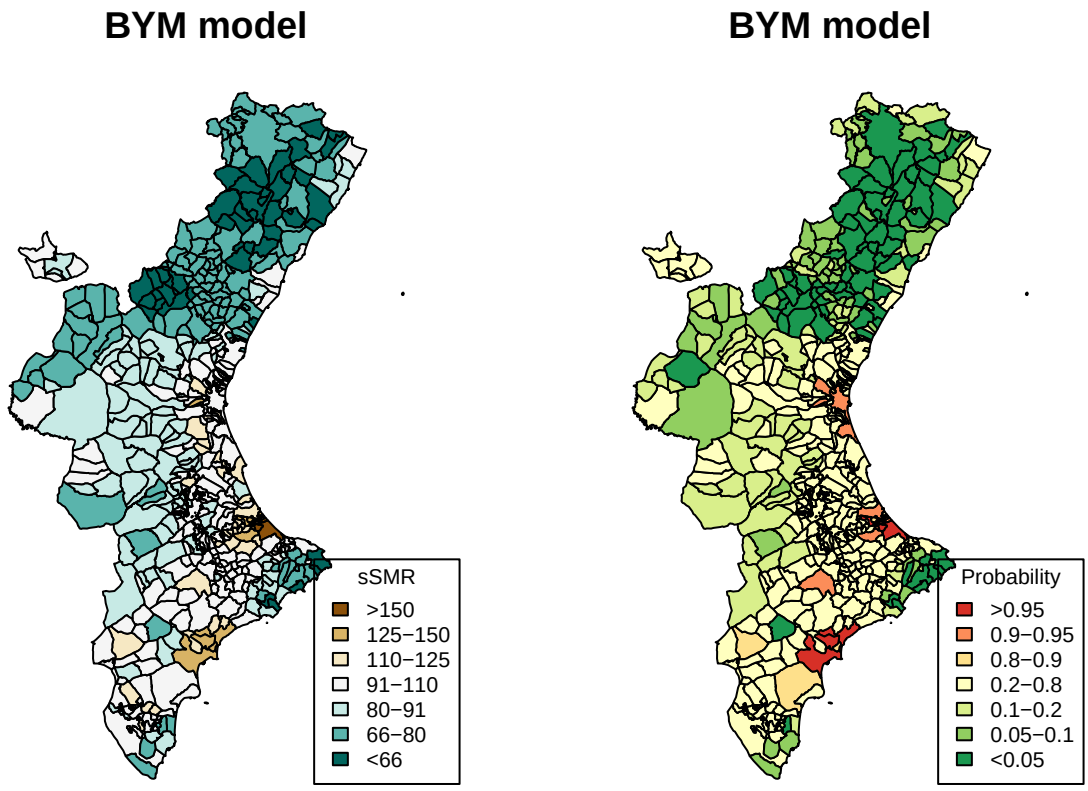
```
## D= 1703.222 pD= 83.26077 DIC= 1786.483
```

```
# Highest sSMR
max(ResulBYM$mean$sSMR)
```

```
## [1] 170.2413
```

Choropleth maps and some summaries

```
ColorSMR <- brewer.pal(7, "BrBG")[7:1]
ColorProb <- brewer.pal(7, "RdYlGn")[7:1]
par(mfrow = c(1, 2))
par(mar = c(1, 1, 2, 1) + 0.1)
plot(VR.cart, col = ColorSMR[as.numeric(cut(ResulBYM$mean$sSMR, 100 * c(-0.1,
    1/1.5, 1/1.25, 1/1.1, 1.1, 1.25, 1.5, 100)))])
title("BYM model", cex = 0.75)
legend(x = "bottomright", fill = ColorSMR[7:1], legend = c(">150", "125-150",
    "110-125", "91-110", "80-91", "66-80", "<66"), cex = 0.65, inset = 0.03,
    title = "sSMR")
plot(VR.cart, col = ColorProb[as.numeric(cut(apply(ResulBYM$sims.list$sSMR,
    2, function(x) {
        mean(x > 100)
    }), c(-0.1, 0.05, 0.1, 0.2, 0.8, 0.9, 0.95, 1.1)))])
title("BYM model", cex = 0.75)
legend(x = "bottomright", fill = ColorProb[7:1], legend = c(">0.95", "0.9-0.95",
    "0.8-0.9", "0.2-0.8", "0.1-0.2", "0.05-0.1", "<0.05"), cex = 0.65,
    inset = 0.03, title = "Probability")
```




```
# Correlations between spatial patterns of the sSMRs
cor(cbind(ResulProperCAR$mean$sSMR, ResulBYM$mean$sSMR))
```

```
##           [,1]      [,2]
## [1,] 1.0000000 0.9891443
```

```
## [2,] 0.9891443 1.0000000
```

```
# standard deviations of the sSMRs for each spatial pattern
sd(ResulProperCAR$mean$sSMR)
```

```
## [1] 17.31201
```

```
sd(ResulBYM$mean$sSMR)
```

```
## [1] 18.54342
```

Standard deviations and correlations between different sSMRs patterns

```
# standard deviation for the spatial patterns fitted
100 * sd(resul.BYM.inla$summary.fitted.values[, 1])
```

```
## [1] 18.58966
```

```
100 * sd(resul.PCAR.inla$summary.fitted.values[, 1])
```

```
## [1] 17.30716
```

```
100 * sd(resul.HCAR.inla$summary.fitted.values[, 1])
```

```
## [1] 11.71754
```

```
sd(ResulBYM$mean$sSMR)
```

```
## [1] 18.54342
```

```
sd(ResulProperCAR$mean$sSMR)
```

```
## [1] 17.31201
```

```
# Correlations between patterns
round(cor(cbind(resul.BYM.inla$summary.fitted.values[, 1], resul.PCAR.inla$summary.fitted.values[,
    1], resul.HCAR.inla$summary.fitted.values[, 1], ResulBYM$mean$sSMR,
    ResulProperCAR$mean$sSMR)), 4)
```

```
##        [,1]   [,2]   [,3]   [,4]   [,5]
## [1,] 1.0000 0.9881 0.8072 0.9995 0.9893
## [2,] 0.9881 1.0000 0.8516 0.9877 0.9993
## [3,] 0.8072 0.8516 1.0000 0.8077 0.8432
## [4,] 0.9995 0.9877 0.8077 1.0000 0.9891
## [5,] 0.9893 0.9993 0.8432 0.9891 1.0000
```