Example 4.6

Disease mapping: from foundations to multidimensional modeling

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

This document reproduces the analysis made at Example 4.6 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(R2WinBUGS)) {
    install.packages("R2WinBUGS")
    library(R2WinBUGS)
}
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")
    }
}
```

```r
# 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")
```

Running of the BYM (original and reparameterized) models

```r
# Number of iterations to run the models
n.iters = c(5000, 10000, 25000)

# Original BYM model
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.h)
        sSMR[i] <- 100 * exp(log.theta[i])
        P.sSMR[i] <- step(sSMR[i] - 100)
    }
    sp[1:n] ~ car.normal(adj[], weights[], num[], tau.sp)
    mu ~ dflat()
    tau.h <- pow(sd.h, -2)
    sd.h ~ dunif(0, 10)
    tau.sp <- pow(sd.sp, -2)
    sd.sp ~ dunif(0, 10)
}

# BYM.naive run of the model
data <- list(O = Obs.muni, E = Exp.muni, n = 540, num = VR.wb$num, weights = VR.wb$weights,
    adj = VR.wb$adj)
inits <- function() {
    list(het = rnorm(540), sp = rnorm(540), sd.h = runif(1), sd.sp = runif(1),
        mu = rnorm(1))
}
param <- c("mu", "sd.h", "sd.sp")

time.Naive <- matrix(nrow = 10, ncol = 3)
Resul.Naive <- list()
for (i in 1:10) {
    Resul.Naive[[i]] <- list()
    for (j in 1:3) {
        n.iter <- n.iters[j]
        time.Naive[i, j] <- system.time(Resul.Naive[[i]][[j]] <- bugs(data = data,
            inits = inits, param = param, model.file = ModelBYM, n.iter = n.iter,
            n.burnin = n.iter/10, DIC = TRUE, bugs.seed = ((j - 1) * 10 +
```

```
                i)))[3]
    }
}

# BYM.DIC run of the model
time.DIC <- matrix(nrow = 10, ncol = 3)
Resul.DIC <- list()
for (i in 1:10) {
    Resul.DIC[[i]] <- list()
    for (j in 1:3) {
        n.iter <- n.iters[j]
        time.DIC[i, j] <- system.time(Resul.DIC[[i]][[j]] <- bugs(data = data,
            inits = inits, param = param, model.file = ModelBYM, n.iter = n.iter,
            n.burnin = n.iter/10, DIC = FALSE, bugs.seed = ((j - 1) * 10 +
                i)))[3]
    }
}

# BYM.DIC.pbugs run of the model
time.DIC.pbugs <- matrix(nrow = 10, ncol = 3)
Resul.DIC.pbugs <- list()
for (i in 1:10) {
    Resul.DIC.pbugs[[i]] <- list()
    for (j in 1:3) {
        n.iter <- n.iters[j]
        time.DIC.pbugs[i, j] <- system.time(Resul.DIC.pbugs[[i]][[j]] <- pbugs(data = data,
            inits = inits, param = param, model.file = ModelBYM, n.iter = n.iter,
            n.burnin = n.iter/10, DIC = FALSE, bugs.seed = ((j - 1) * 10 +
                i)))[3]
    }
}

# Reparameterized BYM model
ModelBYM.Reparam <- function() {
    for (i in 1:n) {
        O[i] ~ dpois(lambda[i])
        log(lambda[i]) <- log(E[i]) + log.theta[i]
        log.theta[i] <- mu + sd.h * het[i] + sd.sp * sp[i]
        het[i] ~ dnorm(0, 1)
        sSMR[i] <- 100 * exp(log.theta[i])
        P.sSMR[i] <- step(sSMR[i] - 100)
    }
    sp[1:n] ~ car.normal(adj[], weights[], num[], 1)
    mu ~ dflat()
    sd.h ~ dunif(0, 10)
    sd.sp ~ dunif(0, 10)
}

# BYM.Reparam run of the model
time.Reparam <- matrix(nrow = 10, ncol = 3)
Resul.Reparam <- list()
for (i in 1:10) {
    Resul.Reparam[[i]] <- list()
```

```
    for (j in 1:3) {
        n.iter <- n.iters[j]
        time.Reparam[i, j] <- system.time(Resul.Reparam[[i]][[j]] <- pbugs(data = data,
            inits = inits, param = param, model.file = ModelBYM.Reparam,
            n.iter = n.iter, n.burnin = n.iter/10, DIC = FALSE, bugs.seed = ((j -
                1) * 10 + i)))[3]
    }
}
```

Computing times

```
# BYM.naive
apply(time.Naive, 2, mean, na.rm = T)
```

```
## [1]  91.957 147.744 318.657
```

```
# BYM.DIC
apply(time.DIC, 2, mean, na.rm = T)
```

```
## [1]  59.644 118.728 289.288
```

```
# BYM.DIC.pbugs
apply(time.DIC.pbugs, 2, mean, na.rm = T)
```

```
## [1]  27.140  51.194 121.299
```

```
# BYM.Reparam
apply(time.Reparam, 2, mean, na.rm = T)
```

```
## [1]  32.133  61.395 149.595
```

Median Brooks-Gelman-Rubin statistic

```
Rhat.sd.sp.5000 <- matrix(nrow = 10, ncol = 4)
Rhat.sd.sp.10000 <- matrix(nrow = 10, ncol = 4)
Rhat.sd.sp.25000 <- matrix(nrow = 10, ncol = 4)
for (i in 1:10) {
    Rhat.sd.sp.5000[i, ] <- c(Resul.Naive[[i]][[1]]$summary["sd.sp", "Rhat"],
        Resul.DIC[[i]][[1]]$summary["sd.sp", "Rhat"], Resul.DIC.pbugs[[i]][[1]]$summary["sd.sp",
            "Rhat"], Resul.Reparam[[i]][[1]]$summary["sd.sp", "Rhat"])
    Rhat.sd.sp.10000[i, ] <- c(Resul.Naive[[i]][[2]]$summary["sd.sp", "Rhat"],
        Resul.DIC[[i]][[2]]$summary["sd.sp", "Rhat"], Resul.DIC.pbugs[[i]][[2]]$summary["sd.sp",
            "Rhat"], Resul.Reparam[[i]][[2]]$summary["sd.sp", "Rhat"])
    Rhat.sd.sp.25000[i, ] <- c(Resul.Naive[[i]][[3]]$summary["sd.sp", "Rhat"],
        Resul.DIC[[i]][[3]]$summary["sd.sp", "Rhat"], Resul.DIC.pbugs[[i]][[3]]$summary["sd.sp",
            "Rhat"], Resul.Reparam[[i]][[3]]$summary["sd.sp", "Rhat"])
}

Rhat.sd.h.5000 <- matrix(nrow = 10, ncol = 4)
Rhat.sd.h.10000 <- matrix(nrow = 10, ncol = 4)
Rhat.sd.h.25000 <- matrix(nrow = 10, ncol = 4)
for (i in 1:10) {
    Rhat.sd.h.5000[i, ] <- c(Resul.Naive[[i]][[1]]$summary["sd.h", "Rhat"],
```

```r
        Resul.DIC[[i]][[1]]$summary["sd.h", "Rhat"], Resul.DIC.pbugs[[i]][[1]]$summary["sd.h",
            "Rhat"], Resul.Reparam[[i]][[1]]$summary["sd.h", "Rhat"])
    Rhat.sd.h.10000[i, ] <- c(Resul.Naive[[i]][[2]]$summary["sd.h", "Rhat"],
        Resul.DIC[[i]][[2]]$summary["sd.h", "Rhat"], Resul.DIC.pbugs[[i]][[2]]$summary["sd.h",
            "Rhat"], Resul.Reparam[[i]][[2]]$summary["sd.h", "Rhat"])
    Rhat.sd.h.25000[i, ] <- c(Resul.Naive[[i]][[3]]$summary["sd.h", "Rhat"],
        Resul.DIC[[i]][[3]]$summary["sd.h", "Rhat"], Resul.DIC.pbugs[[i]][[3]]$summary["sd.h",
            "Rhat"], Resul.Reparam[[i]][[3]]$summary["sd.h", "Rhat"])
}
cbind(apply(Rhat.sd.sp.5000, 2, median), apply(Rhat.sd.sp.10000, 2, median),
    apply(Rhat.sd.sp.25000, 2, median))
```

```
##          [,1]     [,2]     [,3]
## [1,] 1.037687 1.021526 1.012036
## [2,] 1.046921 1.032819 1.006148
## [3,] 1.048982 1.028960 1.007393
## [4,] 1.008103 1.001985 1.002385
```

```r
cbind(apply(Rhat.sd.h.5000, 2, median), apply(Rhat.sd.h.10000, 2, median),
    apply(Rhat.sd.h.25000, 2, median))
```

```
##          [,1]     [,2]     [,3]
## [1,] 1.170305 1.170357 1.148411
## [2,] 1.146205 1.323612 1.072519
## [3,] 1.353096 1.174368 1.076684
## [4,] 1.011610 1.000789 1.003800
```

Median efective sample size

```r
Neff.sd.sp.5000 <- matrix(nrow = 10, ncol = 4)
Neff.sd.sp.10000 <- matrix(nrow = 10, ncol = 4)
Neff.sd.sp.25000 <- matrix(nrow = 10, ncol = 4)
for (i in 1:10) {
    Neff.sd.sp.5000[i, ] <- c(Resul.Naive[[i]][[1]]$summary["sd.sp", "n.eff"],
        Resul.DIC[[i]][[1]]$summary["sd.sp", "n.eff"], Resul.DIC.pbugs[[i]][[1]]$summary["sd.sp",
            "n.eff"], Resul.Reparam[[i]][[1]]$summary["sd.sp", "n.eff"])
    Neff.sd.sp.10000[i, ] <- c(Resul.Naive[[i]][[2]]$summary["sd.sp", "n.eff"],
        Resul.DIC[[i]][[2]]$summary["sd.sp", "n.eff"], Resul.DIC.pbugs[[i]][[2]]$summary["sd.sp",
            "n.eff"], Resul.Reparam[[i]][[2]]$summary["sd.sp", "n.eff"])
    Neff.sd.sp.25000[i, ] <- c(Resul.Naive[[i]][[3]]$summary["sd.sp", "n.eff"],
        Resul.DIC[[i]][[3]]$summary["sd.sp", "n.eff"], Resul.DIC.pbugs[[i]][[3]]$summary["sd.sp",
            "n.eff"], Resul.Reparam[[i]][[3]]$summary["sd.sp", "n.eff"])
}
Neff.sd.h.5000 <- matrix(nrow = 10, ncol = 4)
Neff.sd.h.10000 <- matrix(nrow = 10, ncol = 4)
Neff.sd.h.25000 <- matrix(nrow = 10, ncol = 4)
for (i in 1:10) {
    Neff.sd.h.5000[i, ] <- c(Resul.Naive[[i]][[1]]$summary["sd.h", "n.eff"],
        Resul.DIC[[i]][[1]]$summary["sd.h", "n.eff"], Resul.DIC.pbugs[[i]][[1]]$summary["sd.h",
            "n.eff"], Resul.Reparam[[i]][[1]]$summary["sd.h", "n.eff"])
    Neff.sd.h.10000[i, ] <- c(Resul.Naive[[i]][[2]]$summary["sd.h", "n.eff"],
        Resul.DIC[[i]][[2]]$summary["sd.h", "n.eff"], Resul.DIC.pbugs[[i]][[2]]$summary["sd.h",
```

```
            "n.eff"], Resul.Reparam[[i]][[2]]$summary["sd.h", "n.eff"])
    Neff.sd.h.25000[i, ] <- c(Resul.Naive[[i]][[3]]$summary["sd.h", "n.eff"],
        Resul.DIC[[i]][[3]]$summary["sd.h", "n.eff"], Resul.DIC.pbugs[[i]][[3]]$summary["sd.h",
            "n.eff"], Resul.Reparam[[i]][[3]]$summary["sd.h", "n.eff"])
}
cbind(apply(Neff.sd.sp.5000, 2, median), apply(Neff.sd.sp.10000, 2, median),
    apply(Neff.sd.sp.25000, 2, median))
```

```
##       [,1] [,2] [,3]
## [1,] 105.0  140  210
## [2,]  64.5  104  325
## [3,]  54.0  105  315
## [4,] 345.0 1000 1000
```

```
cbind(apply(Neff.sd.h.5000, 2, median), apply(Neff.sd.h.10000, 2, median),
    apply(Neff.sd.h.25000, 2, median))
```

```
##       [,1]   [,2]   [,3]
## [1,]  18.0   17.5   23.5
## [2,]  20.5   11.5   49.5
## [3,]  10.0   22.5   48.5
## [4,] 440.0 1000.0 1000.0
```

INLA BYM implementation

```
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.l <- inla(form, family = "poisson", data = data, E = E,
    control.compute = list(dic = TRUE), control.inla = list(strategy = "laplace"))
# Computing time
resul.BYM.inla.l$cpu.used
```

```
##  Pre-processing    Running inla Post-processing           Total
##       1.5337510      62.6700380       0.1328881      64.3366771
```

```
resul.BYM.inla.sl <- inla(form, family = "poisson", data = data, E = E,
    control.compute = list(dic = TRUE))
# Computing time
resul.BYM.inla.sl$cpu.used
```

```
##  Pre-processing    Running inla Post-processing           Total
##        1.078692        5.515544        0.129699        6.723935
```

```
resul.BYM.inla.g <- inla(form, family = "poisson", data = data, E = E,
    control.compute = list(dic = TRUE), control.inla = list(strategy = "gaussian"))
```

```r
# Computing time
resul.BYM.inla.g$cpu.used
```

```
##  Pre-processing    Running inla Post-processing           Total
##        1.084395        4.980822        0.132905        6.198122
```

```r
# Posterior summaries of the precisions in the model
resul.BYM.inla.g$summary.hyperpar
```

```
##                                                 mean         sd 0.025quant
## Precision for id.node (iid component)     201.301127 315.810140   29.26644
## Precision for id.node (spatial component)   6.309793   2.030998    3.16668
##                                              0.5quant 0.975quant       mode
## Precision for id.node (iid component)     111.214629  927.54776 54.335758
## Precision for id.node (spatial component)   6.037931   11.07038  5.520207
```

```r
resul.BYM.inla.sl$summary.hyperpar
```

```
##                                                 mean         sd 0.025quant
## Precision for id.node (iid component)     201.301127 315.810140   29.26644
## Precision for id.node (spatial component)   6.309793   2.030998    3.16668
##                                              0.5quant 0.975quant       mode
## Precision for id.node (iid component)     111.214629  927.54776 54.335758
## Precision for id.node (spatial component)   6.037931   11.07038  5.520207
```

```r
resul.BYM.inla.l$summary.hyperpar
```

```
##                                                 mean         sd 0.025quant
## Precision for id.node (iid component)     201.301127 315.810140   29.26644
## Precision for id.node (spatial component)   6.309793   2.030998    3.16668
##                                              0.5quant 0.975quant       mode
## Precision for id.node (iid component)     111.214629  927.54776 54.335758
## Precision for id.node (spatial component)   6.037931   11.07038  5.520207
```