Example 5.2

Disease mapping: from foundations to multidimensional modeling

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

This document reproduces the analysis made at Example 5.2 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(spdep)) {
    install.packages("spdep")
    library(spdep)
}
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/AIDS-mod.Rdata') since that file contains the modified
```

```r
# data making it possible to reproduce this document.
load("../Data/AIDS.Rdata")
# The data loaded contain the following objects: carto.val:
# SpatialPolygonsDataFrame with the census tracts of Valencia city
# Deprivation: Deprivation index for the census tracts of Valencia city
# Obs.AIDS: Observed cases per census tract Exp.AIDS: Observed cases
# per census tract
```

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")
}
```

Data management

```r
# Neighbouring structure for the Valencia city census tracts
carto.val.nb = poly2nb(carto.val, snap = 1)
carto.val.wb = nb2WB(carto.val.nb)

# Categorization of the deprivation index in 5 and 25 groups
cuts.5 <- quantile(Deprivation, prob = c(0, 0.2, 0.4, 0.6, 0.8, 1))
Deprivation.5 <- as.numeric(cut(Deprivation, cuts.5, include.lowest = TRUE))
cuts.25 <- quantile(Deprivation, prob = (0:25/25))
Deprivation.25 <- as.numeric(cut(Deprivation, cuts.25, include.lowest = TRUE))
```

Ecological regression with linear term

```
RegEcoLinear = function() {
    for (i in 1:n) {
        O[i] ~ dpois(lambda[i])
        log(lambda[i]) <- log(E[i]) + log.theta[i]
        log.theta[i] <- mu + beta * Deprivation[i] + sd.sp * sp[i] + sd.h *
            het[i]
        sSMR.withoutDep[i] <- 100 * exp(mu + sd.sp * sp[i] + sd.h * het[i])
        sSMR.withDep[i] <- 100 * exp(log.theta[i])
        het[i] ~ dnorm(0, 1)
    }
    sp[1:n] ~ car.normal(adj[], weights[], num[], 1)

    for (i in 1:nVec) {
        weights[i] <- 1
    }

    # Prior distributions
    mu ~ dflat()
    sd.sp ~ dunif(0, 5)
    sd.h ~ dunif(0, 5)
    beta ~ dflat()
}

data = list(O = Obs.AIDS, E = Exp.AIDS, Deprivation = Deprivation, n = length(Obs.AIDS),
    nVec = length(carto.val.wb$adj), num = carto.val.wb$num, adj = carto.val.wb$adj)
inits = function() {
    list(mu = rnorm(1), beta = rnorm(1), sd.sp = runif(1), sd.h = runif(1),
        sp = rnorm(length(Obs.AIDS)), het = rnorm(length(Obs.AIDS)))
}
param = c("sSMR.withDep", "sSMR.withoutDep", "mu", "beta", "sd.sp", "sd.h")
ResulLin = pbugs(data = data, inits = inits, parameters = param, model.file = RegEcoLinear,
    n.iter = 10000, n.burnin = 1000, DIC = F, bugs.seed = 1)
# Computing time
ResulLin$exec_time
```

```
## Time difference of 1.452312 mins
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9995  1.0003  1.0013  1.0019  1.0029  1.0120
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   180.0   730.0  1000.0   859.7  1000.0  1000.0
```

```
# Results summaries
round(ResulLin$summary["beta", ], 3)
```

```
##     mean       sd     2.5%      25%      50%      75%    97.5%     Rhat
##    0.524    0.055    0.422    0.487    0.521    0.559    0.636    1.000
##    n.eff
```

```
## 1000.000
```

```
exp(ResulLin$mean$beta * (max(Deprivation) - min(Deprivation)))
```

```
## [1] 20.9166
```

Ecological regression with 5 independent categories

```
RegEco5Cat = function() {
    for (i in 1:n) {
        O[i] ~ dpois(lambda[i])
        log(lambda[i]) <- log(E[i]) + log.theta[i]
        log.theta[i] <- mu + beta[Deprivation.5[i]] + sd.sp * sp[i] + sd.h *
            het[i]
        sSMR.withoutDep[i] <- 100 * exp(mu + sd.sp * sp[i] + sd.h * het[i])
        sSMR.withDep[i] <- 100 * exp(log.theta[i])
        het[i] ~ dnorm(0, 1)
    }
    sp[1:n] ~ car.normal(adj[], weights[], num[], 1)

    for (i in 1:nVec) {
        weights[i] <- 1
    }

    # Prior distributions
    mu ~ dflat()
    sd.sp ~ dunif(0, 5)
    sd.h ~ dunif(0, 5)
    beta[1] <- -sum(beta[2:5])
    for (i in 2:5) {
        beta[i] ~ dflat()
    }
}

data = list(O = Obs.AIDS, E = Exp.AIDS, Deprivation.5 = Deprivation.5,
    n = length(Obs.AIDS), nVec = length(carto.val.wb$adj), num = carto.val.wb$num,
    adj = carto.val.wb$adj)
inits <- function() {
    list(mu = rnorm(1), beta = c(NA, rnorm(4)), sd.sp = runif(1), sd.h = runif(1),
        sp = rnorm(length(Obs.AIDS)), het = rnorm(length(Obs.AIDS)))
}
param = c("sSMR.withDep", "sSMR.withoutDep", "mu", "beta", "sd.sp", "sd.h")
Resul5Cat = pbugs(data = data, inits = inits, parameters = param, model.file = RegEco5Cat,
    n.iter = 10000, n.burnin = 1000, DIC = F, bugs.seed = 1)
# Computing time
Resul5Cat$exec_time
```

```
## Time difference of 1.535038 mins
```

```
# Convergence checking
summary(Resul5Cat$summary[, "Rhat"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9995  1.0003  1.0014  1.0020  1.0030  1.0155
```

```
summary(Resul5Cat$summary[, "n.eff"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   140.0   700.0  1000.0   847.3  1000.0  1000.0
```

Ecological regression with 25 independent categories

```
RegEco25Cat = 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.beta * beta[Deprivation.25[i]] + sd.sp *
            sp[i] + sd.h * het[i]
        sSMR.withoutDep[i] <- 100 * exp(mu + sd.sp * sp[i] + sd.h * het[i])
        sSMR.withDep[i] <- 100 * exp(log.theta[i])
        het[i] ~ dnorm(0, 1)
    }
    sp[1:n] ~ car.normal(adj[], weights[], num[], 1)

    for (i in 1:nVec) {
        weights[i] <- 1
    }

    beta[1:25] ~ car.normal(adj.beta[], weights.beta[], num.beta[], 1)
    for (i in 1:nVec.beta) {
        weights.beta[i] <- 1
    }

    # Prior distributions
    mu ~ dflat()
    sd.sp ~ dunif(0, 5)
    sd.h ~ dunif(0, 5)
    sd.beta ~ dunif(0, 5)
}

num.beta = c(1, rep(2, 23), 1)
adj.beta = c(2)
for (i in 2:24) {
    adj.beta = c(adj.beta, i - 1, i + 1)
}
adj.beta = c(adj.beta, 24)

data = list(O = Obs.AIDS, E = Exp.AIDS, Deprivation.25 = Deprivation.25,
    n = length(Obs.AIDS), nVec = length(carto.val.wb$adj), num = carto.val.wb$num,
    adj = carto.val.wb$adj, nVec.beta = length(adj.beta), num.beta = num.beta,
    adj.beta = adj.beta)
inits = function() {
    list(mu = rnorm(1), beta = rnorm(25), sd.sp = runif(1), sd.beta = runif(1),
        sd.h = runif(1), sp = rnorm(length(Obs.AIDS)), het = rnorm(length(Obs.AIDS)))
}
param = c("sSMR.withDep", "sSMR.withoutDep", "mu", "beta", "sd.sp", "sd.h",
    "sd.beta")
```

```
Resul25Cat = pbugs(data = data, inits = inits, parameters = param, model.file = RegEco25Cat,
    n.iter = 10000, n.burnin = 1000, DIC = F, bugs.seed = 1)
# Computing time
Resul25Cat$exec_time
```

```
## Time difference of 1.574422 mins
```

```
# Convergence checking
summary(Resul25Cat$summary[, "Rhat"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9995  1.0004  1.0013  1.0019  1.0027  1.0162
```

```
summary(Resul25Cat$summary[, "n.eff"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   150.0   770.0  1000.0   859.8  1000.0  1000.0
```

DIC comparison of the previous three models

```
DICPoisson(ResulLin$sims.matrix[, grep("sSMR.withDep", dimnames(ResulLin$sims.matrix)[[2]])],
    Obs.AIDS, Exp.AIDS)
```

```
## D= 1352.322 pD= 100.7711 DIC= 1453.094
```

```
DICPoisson(Resul5Cat$sims.matrix[, grep("sSMR.withDep", dimnames(Resul5Cat$sims.matrix)[[2]])],
    Obs.AIDS, Exp.AIDS)
```

```
## D= 1356.105 pD= 113.6098 DIC= 1469.714
```

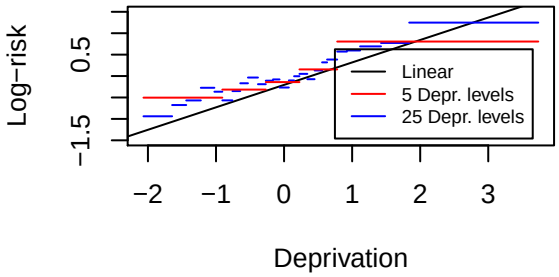
```
DICPoisson(Resul25Cat$sims.matrix[, grep("sSMR.withDep", dimnames(Resul25Cat$sims.matrix)[[2]])],
    Obs.AIDS, Exp.AIDS)
```

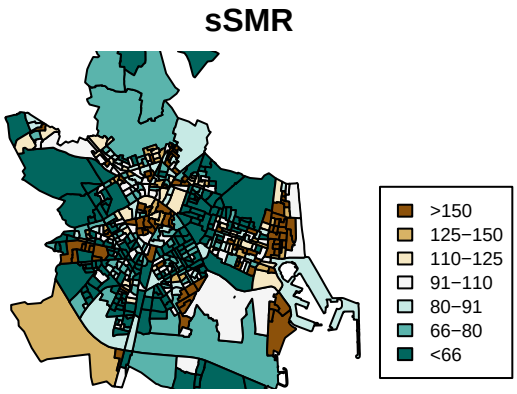
```
## D= 1350.958 pD= 106.023 DIC= 1456.981
```

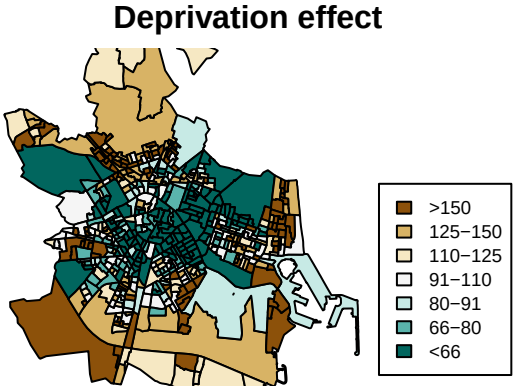
Figure in the example

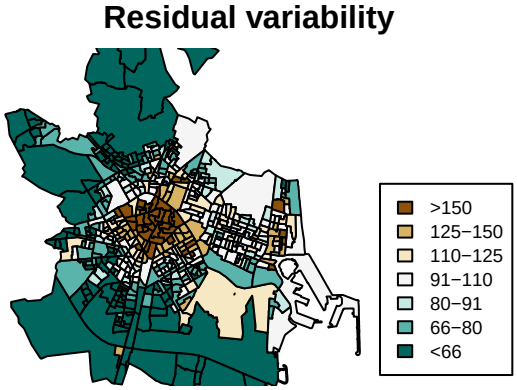
```
par(mfrow = c(2, 2))
# top-left plot
from = range(Deprivation)[1]
to = range(Deprivation)[2]
plot(from, cuts.25[2], ylim = c(-1.5, 1.5), xlim = range(Deprivation),
    type = "n", ylab = "Log-risk", xlab = "Deprivation")
abline(c(ResulLin$mean$mu, ResulLin$mean$beta))

y = mean(Resul25Cat$sims.matrix[, "sd.beta"] * Resul25Cat$sims.matrix[,
    "beta[1]"])
segments(x0 = from, x1 = cuts.25[2], y0 = y, y1 = y, col = "blue")
for (i in 2:24) {
    beta.1 = which(dimnames(Resul25Cat$sims.matrix)[[2]] == "beta[1]")
    y = mean(Resul25Cat$sims.matrix[, "sd.beta"] * Resul25Cat$sims.matrix[,
        beta.1 + i - 1])
    segments(x0 = cuts.25[i], x1 = cuts.25[i + 1], y0 = y, y1 = y, col = "blue")
}
```

```
y = mean(Resul25Cat$sims.matrix[, "sd.beta"] * Resul25Cat$sims.matrix[,
    "beta[25]"])
segments(x0 = cuts.25[25], x1 = to, y0 = y, y1 = y, col = "blue")

segments(x0 = from, x1 = cuts.5[2], y0 = Resul5Cat$mean$beta[1], y1 = Resul5Cat$mean$beta[1],
    col = 2)
for (i in 2:4) {
    segments(x0 = cuts.5[i], x1 = cuts.5[i + 1], y0 = Resul5Cat$mean$beta[i],
        y1 = Resul5Cat$mean$beta[i], col = 2)
}
segments(x0 = cuts.5[5], x1 = to, y0 = Resul5Cat$mean$beta[5], y1 = Resul5Cat$mean$beta[5],
    col = 2)
legend(x = "bottomright", lty = 1, col = c(1:2, "blue"), legend = c("Linear",
    "5 Depr. levels", "25 Depr. levels"), inset = 0.05, cex = 0.7)

# Top-right plot
sSMR.withDep = ResulLin$mean$sSMR.withDep
sSMR.withoutDep = ResulLin$mean$sSMR.withoutDep

par(mar = c(0, 0, 1, 0) + 1)
colors = brewer.pal(7, "BrBG")[7:1]
plot(carto.val, xlim = c(721500, 736000), ylim = c(4368000, 4378000), col = colors[as.numeric(cut(sSMR.w
    100 * c(-0.1, 1/1.5, 1/1.25, 1/1.1, 1.1, 1.25, 1.5, 100)))])
title("sSMR", 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)

# Bottom-left plot
plot(carto.val, xlim = c(721500, 736000), ylim = c(4368000, 4378000), col = colors[as.numeric(cut(100 *
    exp(ResulLin$mean$beta * Deprivation), 100 * c(-0.1, 1/1.5, 1/1.25,
    1/1.1, 1.1, 1.25, 1.5, 100)))])
title("Deprivation effect", 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)

# Bottom-right plot
plot(carto.val, xlim = c(721500, 736000), ylim = c(4368000, 4378000), col = colors[as.numeric(cut(sSMR.w
    100 * c(-0.1, 1/1.5, 1/1.25, 1/1.1, 1.1, 1.25, 1.5, 100)))])
title("Residual variability", 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)
```









Variance decomposition

```
# Variance explained by the deprivation component
var.deprivation = mean(sapply(ResulLin$sims.list$beta, function(x) {
    var(x * Deprivation)
}))
# Variance explained by the random effects
var.residual = mean(apply(log(ResulLin$sims.list$sSMR.withoutDep), 1, var))
# Percentage of variance explained by deprivation
c(sqrt(var.deprivation), sqrt(var.residual), var.deprivation/(var.deprivation +
    var.residual))
```

```
## [1] 0.5270549 0.5680555 0.4626127
```