Example 7.2

Disease mapping: from foundations to multidimensional modeling

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

This document reproduces the analysis made at Example 7.2 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 data 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-ET-mod.Rdata') since that file contains the
# modified data making it possible to reproduce this document.
load("../Data/ObsOral-ET.Rdata")
# load('../Data/ObsOral-mod.Rdata')
load("../Data/ExpOral-ET.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")
}
```

WinBUGS execution of the model with linear time trends

```r
# Model with linear time trends
LinearTrends = function() {
    for (i in 1:nRegions) {
        for (j in 1:nPeriods) {
            Obs[i, j] ~ dpois(lambda[i, j])
            log(lambda[i, j]) <- log(Exp[i, j]) + log.theta[i, j]
            log.theta[i, j] <- (mu.alpha + alpha[i]) + (mu.beta + beta[i]) *
                (j - (nPeriods + 1)/2)
            sSMR[i, j] <- 100 * exp(log.theta[i, j])
        }
        # BYM components in the coefficients of the linear predictor
        alpha[i] <- sd.alpha.spat * alpha.spat[i] + sd.alpha.het * alpha.het[i]
        beta[i] <- sd.beta.spat * beta.spat[i] + sd.beta.het * beta.het[i]
        # Heterogenous random effects
        alpha.het[i] ~ dnorm(0, 1)
```

```
        beta.het[i] ~ dnorm(0, 1)
    }
    # Spatial random effects
    alpha.spat[1:nRegions] ~ car.normal(adj[], w[], num[], 1)
    beta.spat[1:nRegions] ~ car.normal(adj[], w[], num[], 1)

    # Prior distributions
    mu.alpha ~ dflat()
    mu.beta ~ dflat()
    sd.alpha.spat ~ dunif(0, 5)
    sd.alpha.het ~ dunif(0, 5)
    sd.beta.spat ~ dunif(0, 5)
    sd.beta.het ~ dunif(0, 5)
}

data = list(Obs = ObsOral, Exp = ExpOral, nRegions = 540, nPeriods = 12,
    w = rep(1, length(VR.wb$adj)), num = VR.wb$num, adj = VR.wb$adj)
inits = function() {
    list(mu.alpha = rnorm(1), mu.beta = rnorm(1), sd.alpha.spat = runif(1,
        0, 2), sd.beta.spat = runif(1, 0, 2), sd.alpha.het = runif(1, 0,
        2), sd.beta.het = runif(1, 0, 2), alpha.spat = rnorm(540, 0, 1),
        beta.spat = rnorm(540, 0, 1), alpha.het = rnorm(540, 0, 1), beta.het = rnorm(540,
            0, 1))
}

param = c("log.theta", "mu.alpha", "mu.beta", "sd.alpha.spat", "sd.beta.spat",
    "sd.alpha.het", "sd.beta.het")

ResulLinear = pbugs(data = data, inits = inits, parameters = param, model.file = LinearTrends,
    n.iter = 10000, n.burnin = 1000, n.sims = 3000, DIC = F, bugs.seed = 1)

# Computing time
ResulLinear$exec_time
```

```
## Time difference of 27.36483 mins
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.001   1.001   1.001   1.002   1.006
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     390    1800    3000    2419    3000    3000
```

```
# DIC The matrices of observed and expected cases have been tansposed
# so that their elements are arranged in the same order than the first
# argument of the DICPoisson function.
DICPoisson(100 * exp(ResulLinear$sims.matrix[, grep("log.theta", dimnames(ResulLinear$sims.matrix)[[2]]
    as.vector(t(ObsOral)), as.vector(t(ExpOral)))
```

```
## D= 7251.954 pD= 88.88962 DIC= 7340.844
```

Variance decomposition for the linear model in time

```
decompLin = matrix(nrow = 3, ncol = 3, dimnames = list(c("2.5%", "50%",
    "97.5%"), c("S", "T", "ST")))
decompLin.mean = vector(length = 3)
nIter = dim(ResulLinear$sims.list$log.theta)[1]
components = matrix(nrow = nIter, ncol = 4)
for (i in 1:nIter) {
    m = mean(ResulLinear$sims.list$log.theta[i, , ])
    S = apply(ResulLinear$sims.list$log.theta[i, , ], 1, mean) - m
    T = apply(ResulLinear$sims.list$log.theta[i, , ], 2, mean) - m
    ST = ResulLinear$sims.list$log.theta[i, , ] - (m + matrix(rep(S, length(T)),
        ncol = length(T)) + matrix(rep(T, length(S)), ncol = length(T),
        byrow = T))
    components[i, ] = c(m, var(S), var(T), var(as.vector(ST)))
}
aux = components[, c(2:4)]/apply(components[, c(2:4)], 1, sum)
decompLin = apply(aux, 2, quantile, c(0.025, 0.5, 0.975))
decompLin.mean = apply(aux, 2, mean) * 100
decompLin.mean
```

```
## [1] 82.738297 14.206895  3.054809
```

```
decompLin
```

```
##             [,1]       [,2]         [,3]
## 2.5%  0.7070494 0.05900442 0.0008178234
## 50%   0.8332700 0.13762278 0.0213318084
## 97.5% 0.9149423 0.24904211 0.1100614063
```

`WinBUGS` execution of the model with quadratic time trends

```
# Model with quadratic time trends
QuadTrends = function() {
    for (i in 1:nRegions) {
        for (j in 1:nPeriods) {
            Obs[i, j] ~ dpois(lambda[i, j])
            log(lambda[i, j]) <- log(Exp[i, j]) + log.theta[i, j]
            # The linear predictor is truncated to prevent numerical overflows when
            # the log transformated is reverted
            log.theta[i, j] <- min(max((mu.alpha + alpha[i]) + (mu.beta +
                beta[i]) * (j - (nPeriods + 1)/2) + (mu.delta + delta[i]) *
                pow(j - (nPeriods + 1)/2, 2), -10), 10)
            sSMR[i, j] <- 100 * exp(log.theta[i, j])
        }
        # BYM components in the coefficients of the linear predictor
        alpha[i] <- sd.alpha.spat * alpha.spat[i] + sd.alpha.het * alpha.het[i]
        beta[i] <- sd.beta.spat * beta.spat[i] + sd.beta.het * beta.het[i]
        delta[i] <- sd.delta.spat * delta.spat[i] + sd.delta.het * delta.het[i]
        # Heterogenous random effects
        alpha.het[i] ~ dnorm(0, 1)
        beta.het[i] ~ dnorm(0, 1)
        delta.het[i] ~ dnorm(0, 1)
```

```
    }
    # Spatial random effects
    alpha.spat[1:nRegions] ~ car.normal(adj[], w[], num[], 1)
    beta.spat[1:nRegions] ~ car.normal(adj[], w[], num[], 1)
    delta.spat[1:nRegions] ~ car.normal(adj[], w[], num[], 1)

    # Prior distributions
    mu.alpha ~ dflat()
    mu.beta ~ dflat()
    mu.delta ~ dflat()
    sd.alpha.spat ~ dunif(0, 5)
    sd.alpha.het ~ dunif(0, 5)
    sd.beta.spat ~ dunif(0, 5)
    sd.beta.het ~ dunif(0, 5)
    sd.delta.spat ~ dunif(0, 5)
    sd.delta.het ~ dunif(0, 5)
}

data = list(Obs = ObsOral, Exp = ExpOral, nRegions = 540, nPeriods = 12,
    w = rep(1, length(VR.wb$adj)), num = VR.wb$num, adj = VR.wb$adj)
inits = function() {
    list(mu.alpha = rnorm(1), mu.beta = rnorm(1), mu.delta = rnorm(1),
        sd.alpha.spat = runif(1, 0, 2), sd.beta.spat = runif(1, 0, 2),
        sd.delta.spat = runif(1, 0, 2), sd.alpha.het = runif(1, 0, 2),
        sd.beta.het = runif(1, 0, 2), sd.delta.het = runif(1, 0, 2), alpha.spat = rnorm(540,
            0, 1), beta.spat = rnorm(540, 0, 1), delta.spat = rnorm(540,
            0, 1), alpha.het = rnorm(540, 0, 1), beta.het = rnorm(540,
            0, 1), delta.het = rnorm(540, 0, 1))
}
param = c("log.theta", "mu.alpha", "mu.beta", "mu.delta", "sd.alpha.spat",
    "sd.beta.spat", "sd.delta.spat", "sd.alpha.het", "sd.beta.het", "sd.delta.het")
ResulQuad = pbugs(data = data, inits = inits, parameters = param, model.file = QuadTrends,
    n.iter = 10000, n.burnin = 1000, DIC = F, n.sims = 3000, bugs.seed = 1)

# Computing time
ResulQuad$exec_time
```

```
## Time difference of 43.07116 mins
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.001   1.002   1.003   1.003   1.038
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      77     760    1600    1744    3000    3000
```

```
# DIC The matrices of observed and expected cases have been tansposed
# so that their elements are arranged in the same order than the first
# argument of the DICPoisson function.
DICPoisson(100 * exp(ResulQuad$sims.matrix[, grep("log.theta", dimnames(ResulQuad$sims.matrix)[[2]])]),
    as.vector(t(ObsOral)), as.vector(t(ExpOral)))
```

```
## D= 7227.981 pD= 106.3559 DIC= 7334.337
```

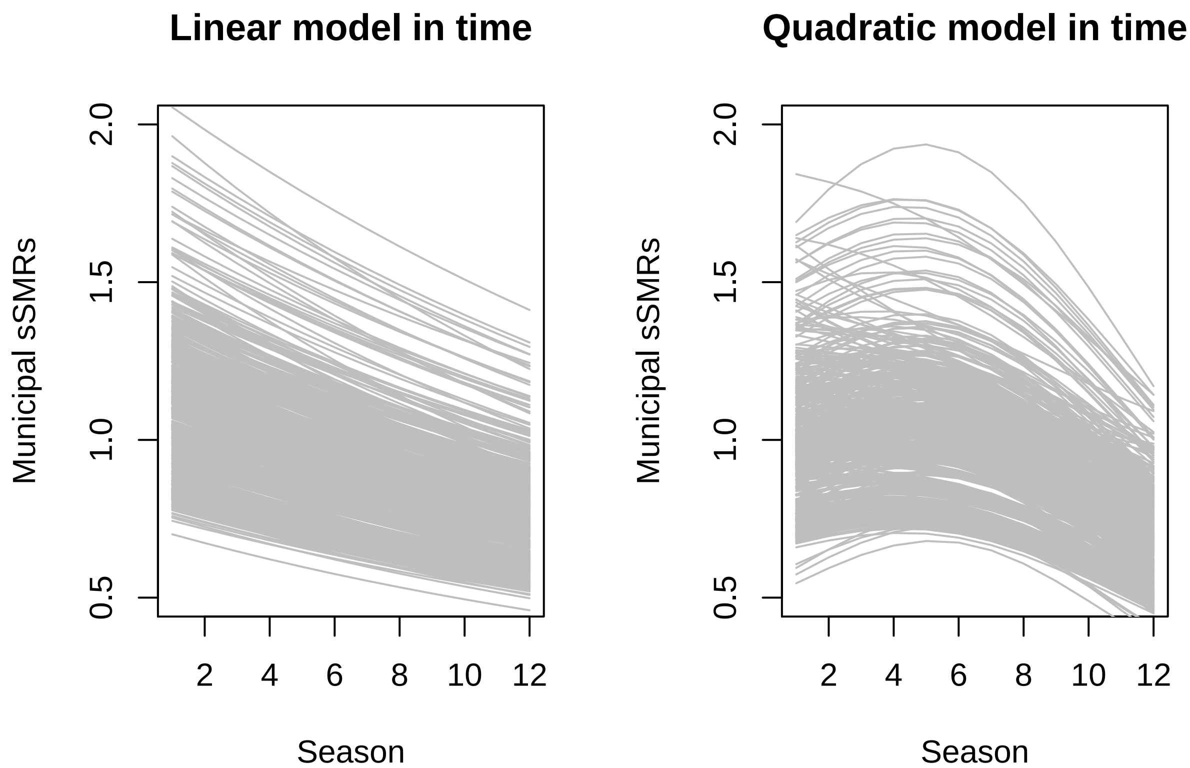
Variance decomposition for the quadratic model in time

```r
decompQuad = matrix(nrow = 3, ncol = 3, dimnames = list(c("2.5%", "50%",
    "97.5%"), c("S", "T", "ST")))
decompQuad.mean = vector(length = 3)
nIter = dim(ResulQuad$sims.list$log.theta)[1]
components = matrix(nrow = nIter, ncol = 4)
for (i in 1:nIter) {
    m = mean(ResulQuad$sims.list$log.theta[i, , ])
    S = apply(ResulQuad$sims.list$log.theta[i, , ], 1, mean) - m
    T = apply(ResulQuad$sims.list$log.theta[i, , ], 2, mean) - m
    ST = ResulQuad$sims.list$log.theta[i, , ] - (m + matrix(rep(S, length(T)),
        ncol = length(T)) + matrix(rep(T, length(S)), ncol = length(T),
        byrow = T))
    components[i, ] = c(m, var(S), var(T), var(as.vector(ST)))
}
aux = components[, c(2:4)]/apply(components[, c(2:4)], 1, sum)
decompQuad = apply(aux, 2, quantile, c(0.025, 0.5, 0.975))
decompQuad.mean = apply(aux, 2, mean) * 100
decompQuad.mean
```

```
## [1] 74.486077 17.335557  8.178365
```

Plot of the linear and quadratic time trends per municipality

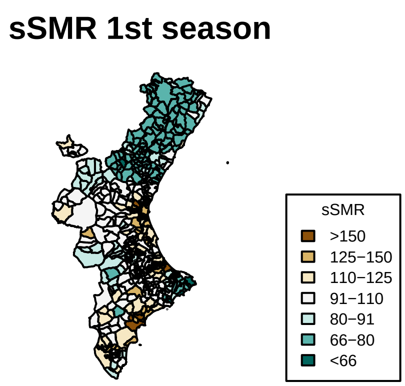
```r
par(mfrow = c(1, 2))
plot(1:12, apply(exp(ResulLinear$sims.list$log.theta[, 1, ]), 2, mean),
    type = "n", ylim = c(0.5, 2), xlab = "Season", ylab = "Municipal sSMRs")
title("Linear model in time")
for (i in 1:540) {
    lines(1:12, apply(exp(ResulLinear$sims.list$log.theta[, i, ]), 2, mean),
        col = "grey")
}
plot(1:12, apply(exp(ResulQuad$sims.list$log.theta[, 1, ]), 2, mean), type = "n",
    ylim = c(0.5, 2), xlab = "Season", ylab = "Municipal sSMRs")
title("Quadratic model in time")
for (i in 1:540) {
    lines(1:12, apply(exp(ResulQuad$sims.list$log.theta[, i, ]), 2, mean),
        col = "grey")
}
```

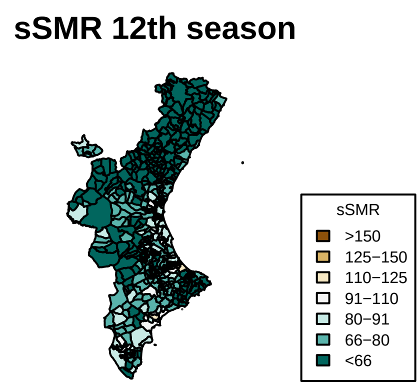


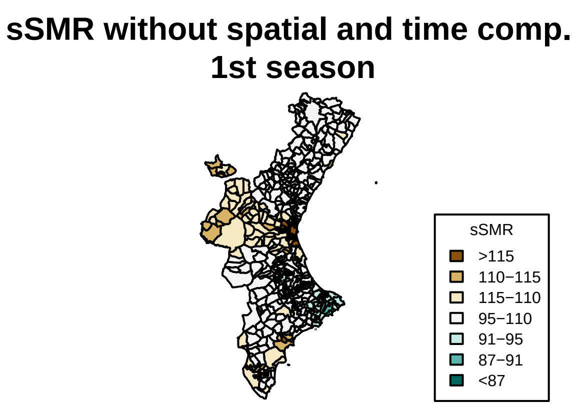
Choropleth maps with and without temporal component

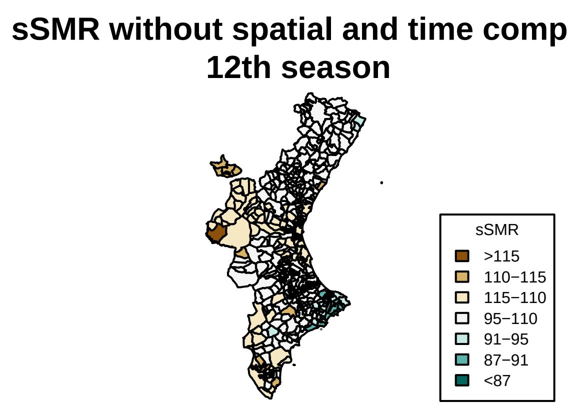
```
par(mfrow = c(2, 2))
par(mar = c(1, 1, 2, 1) + 0.1)
per = 1
sSMR.cut = cut(apply(exp(ResulQuad$sims.list$log.theta[, , per]), 2, mean),
    c(0, 0.66, 0.8, 0.91, 1.1, 1.25, 1.5, 10))
plot(VR.cart, col = brewer.pal(7, "BrBG")[7:1][sSMR.cut])
title("sSMR 1st season", cex = 0.75)
legend(x = "bottomright", fill = brewer.pal(7, "BrBG"), legend = c(">150",
    "125-150", "110-125", "91-110", "80-91", "66-80", "<66"), cex = 0.65,
    inset = 0.03, title = "sSMR")
per = 12
sSMR.cut = cut(apply(exp(ResulQuad$sims.list$log.theta[, , per]), 2, mean),
    c(0, 0.66, 0.8, 0.91, 1.1, 1.25, 1.5, 10))
plot(VR.cart, col = brewer.pal(7, "BrBG")[7:1][sSMR.cut])
title("sSMR 12th season", cex = 0.75)
legend(x = "bottomright", fill = brewer.pal(7, "BrBG"), legend = c(">150",
    "125-150", "110-125", "91-110", "80-91", "66-80", "<66"), cex = 0.65,
    inset = 0.03, title = "sSMR")

# Choropleth maps removing the temporal component
ST.iter = array(dim = c(3000, 540, 12))
```

```r
for (i in 1:nIter) {
    m = mean(ResulQuad$sims.list$log.theta[i, , ])
    S = apply(ResulQuad$sims.list$log.theta[i, , ], 1, mean) - m
    T = apply(ResulQuad$sims.list$log.theta[i, , ], 2, mean) - m
    ST.iter[i, , ] = ResulQuad$sims.list$log.theta[i, , ] - (m + matrix(rep(S,
        length(T)), ncol = length(T)) + matrix(rep(T, length(S)), ncol = length(T),
        byrow = T))
}
ST = apply(exp(ST.iter), c(2, 3), mean)
per = 1
STwithoutT.1.cut = cut(ST[, 1], c(0, 0.87, 0.91, 0.95, 1.05, 1.1, 1.15,
    10))
plot(VR.cart, col = brewer.pal(7, "BrBG")[7:1][STwithoutT.1.cut])
title("sSMR without spatial and time comp.\n 1st season", cex = 0.75)
legend(x = "bottomright", fill = brewer.pal(7, "BrBG"), legend = c(">115",
    "110-115", "115-110", "95-110", "91-95", "87-91", "<87"), cex = 0.65,
    inset = 0.03, title = "sSMR")
per = 12
STwithoutT.12.cut = cut(ST[, 12], c(0, 0.87, 0.91, 0.95, 1.05, 1.1, 1.15,
    10))
plot(VR.cart, col = brewer.pal(7, "BrBG")[7:1][STwithoutT.12.cut])
title("sSMR without spatial and time comp.\n 12th season", cex = 0.75)
legend(x = "bottomright", fill = brewer.pal(7, "BrBG"), legend = c(">115",
    "110-115", "115-110", "95-110", "91-95", "87-91", "<87"), cex = 0.65,
    inset = 0.03, title = "sSMR")
```









INLA implementation of the previous models

```r
# Uniform prior distribution on the standard deviation
sdunif = "expression:
  logdens = -log_precision/2;
  return(logdens)"

# Covariates
muni = 1:540
interval.centered = (1:12) - mean(1:12)

# INLA fit of the linear model in time
data.lin = data.frame(ObsOral = as.vector(ObsOral), ExpOral = as.vector(ExpOral),
    muni = rep(muni, 12), muni.dup = rep(muni, 12), interval.centered = rep(interval.centered,
        each = 540))
formula.lin = ObsOral ~ 1 + f(muni, model = "bym", graph = "../Data/VR.graph",
    hyper = list(prec.spatial = list(prior = sdunif), prec.unstruct = list(prior = sdunif))) +
    interval.centered + f(muni.dup, interval.centered, model = "bym", graph = "../Data/VR.graph",
    hyper = list(prec.spatial = list(prior = sdunif), prec.unstruct = list(prior = sdunif)))
res.lin = inla(formula.lin, family = "poisson", data = data.lin, E = ExpOral,
    control.compute = list(dic = TRUE))

summary(res.lin)
```

```
##
## Call:
## c("inla(formula = formula.lin, family = \"poisson\", data = data.lin, ",  "    E = ExpOral, control.c
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          2.3831         79.3553          0.4541         82.1926
##
## Fixed effects:
##                      mean     sd 0.025quant 0.5quant 0.975quant    mode
## (Intercept)       -0.1384 0.0336    -0.2068  -0.1376    -0.0747 -0.1357
## interval.centered -0.0342 0.0057    -0.0453  -0.0343    -0.0227 -0.0344
##                     kld
## (Intercept)       1e-04
## interval.centered 0e+00
##
## Random effects:
## Name   Model
##  muni   BYM model
## muni.dup   BYM model
##
## Model hyperparameters:
##                                                 mean        sd 0.025quant
## Precision for muni (iid component)           248.968 4.806e+02     29.715
## Precision for muni (spatial component)         6.846 2.374e+00      3.119
## Precision for muni.dup (iid component)     33047.169 1.282e+05   1369.394
## Precision for muni.dup (spatial component) 77695.569 7.783e+05    826.121
##                                             0.5quant 0.975quant     mode
## Precision for muni (iid component)           121.319    1259.11   54.187
```

```
## Precision for muni (spatial component)         6.557      12.35    5.961
## Precision for muni.dup (iid component)      9711.998  205554.71 2893.021
## Precision for muni.dup (spatial component) 10473.527  504347.62 1767.725
##
## Expected number of effective parameters(std dev): 98.52(14.14)
## Number of equivalent replicates : 65.77
##
## Deviance Information Criterion (DIC) ...............: 7344.26
## Deviance Information Criterion (DIC, saturated) ....: 3843.52
## Effective number of parameters .....................: 99.19
##
## Marginal log-Likelihood:  -3502.55
## Posterior marginals for linear predictor and fitted values computed
```

```r
ResulLinear$summary[c("mu.alpha", "mu.beta"), ]
```

```
##                 mean          sd       2.5%        25%       50%
## mu.alpha -0.13659920 0.032843867 -0.2035025 -0.1583000 -0.136100
## mu.beta  -0.03472355 0.005478601 -0.0449635 -0.0384525 -0.034925
##                 75%      97.5%     Rhat n.eff
## mu.alpha -0.1137000 -0.0737565 1.000713  3000
## mu.beta  -0.0310875 -0.0233700 1.001179  2900
```

```r
# INLA fit of the quadratic model in time
data.quad = data.frame(ObsOral = as.vector(ObsOral), ExpOral = as.vector(ExpOral),
    muni = rep(muni, 12), muni.dup = rep(muni, 12), muni.dup2 = rep(muni,
        12), interval.centered = rep(interval.centered, each = 540), interval.centered2 = rep(interval.c
        each = 540))
formula.quad = ObsOral ~ 1 + f(muni, model = "bym", graph = "../Data/VR.graph",
    hyper = list(prec.spatial = list(prior = sdunif), prec.unstruct = list(prior = sdunif))) +
    interval.centered + f(muni.dup, interval.centered, model = "bym", graph = "../Data/VR.graph",
    hyper = list(prec.spatial = list(prior = sdunif), prec.unstruct = list(prior = sdunif))) +
    interval.centered2 + f(muni.dup2, interval.centered2, model = "bym",
    graph = "../Data/VR.graph", hyper = list(prec.spatial = list(prior = sdunif),
        prec.unstruct = list(prior = sdunif)))
res.quad = inla(formula.quad, family = "poisson", data = data.quad, E = ExpOral,
    control.compute = list(dic = TRUE))

summary(res.quad)
```

```
##
## Call:
## c("inla(formula = formula.quad, family = \"poisson\", data = data.quad, ",  "    E = ExpOral, control
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          2.3122        243.0968          0.9116        246.3207
##
## Fixed effects:
##                       mean     sd 0.025quant 0.5quant 0.975quant    mode
## (Intercept)        -0.0567 0.0411    -0.1393  -0.0560     0.0221 -0.0546
## interval.centered  -0.0331 0.0060    -0.0446  -0.0332    -0.0210 -0.0334
## interval.centered2 -0.0076 0.0023    -0.0122  -0.0075    -0.0033 -0.0073
##                    kld
## (Intercept)          0
```

```
## interval.centered    0
## interval.centered2   0
##
## Random effects:
## Name    Model
##  muni    BYM model
## muni.dup    BYM model
## muni.dup2    BYM model
##
## Model hyperparameters:
##                                                 mean        sd 0.025quant
## Precision for muni (iid component)          2.197e+02 3.432e+02     26.216
## Precision for muni (spatial component)      7.279e+00 2.791e+00      3.514
## Precision for muni.dup (iid component)      1.657e+04 4.110e+04   1023.629
## Precision for muni.dup (spatial component)  9.199e+04 1.092e+06    912.697
## Precision for muni.dup2 (iid component)     1.953e+05 3.841e+05  12754.835
## Precision for muni.dup2 (spatial component) 2.690e+04 3.876e+04   3287.842
##                                              0.5quant 0.975quant      mode
## Precision for muni (iid component)            121.035  1.020e+03    55.940
## Precision for muni (spatial component)          6.709  1.428e+01     5.747
## Precision for muni.dup (iid component)       6690.603  9.314e+04  2320.492
## Precision for muni.dup (spatial component)  11064.803  5.885e+05  1969.688
## Precision for muni.dup2 (iid component)     91476.847  1.021e+06 31019.870
## Precision for muni.dup2 (spatial component) 15531.316  1.205e+05  7311.112
##
## Expected number of effective parameters(std dev): 116.02(16.83)
## Number of equivalent replicates : 55.85
##
## Deviance Information Criterion (DIC) ...............: 7336.13
## Deviance Information Criterion (DIC, saturated) ....: 3835.39
## Effective number of parameters .....................: 117.07
##
## Marginal log-Likelihood:  -3413.89
## Posterior marginals for linear predictor and fitted values computed
```

```
ResulQuad$summary[c("mu.alpha", "mu.beta", "mu.delta"), ]
```

```
##                 mean          sd       2.5%         25%        50%
## mu.alpha -0.054445112 0.042831318 -0.1363025 -0.08406500 -0.0543700
## mu.beta  -0.033712155 0.005611317 -0.0444800 -0.03761250 -0.0337200
## mu.delta -0.007359049 0.002455554 -0.0124600 -0.00891525 -0.0072915
##                75%        97.5%     Rhat n.eff
## mu.alpha -0.0251075  0.028321500 1.011731   180
## mu.beta  -0.0299175 -0.022608250 1.002175  1400
## mu.delta -0.0056280 -0.003039775 1.019854   110
```