Example 7.3

Disease mapping: from foundations to multidimensional modeling

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

This document reproduces the analysis made at Example 7.3 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.

The code used for this Example has been adapted from that used in: "Ugarte, M. D., Adin, A. and Goicoa, T. (2017). One-dimensional, two-dimensional and three dimensional B-splines to specify space-time interactions in Bayesian disease mappping: Model fitting and model identifiability. Spatial statistics, 22: 451-468", which has been kindly shared by its authors. Most technical details of the models implemented, mainly those corresponding to their constraints, can be found at the original paper.

Libraries and data loading

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

# Data loading
#------------
# For reproducing the document, the following line should be changed to
# 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")
```

Some preliminar definitions

```
# Strategy for fitting INLA models
strategy = "gaussian"

# Number of municipalities and periods
nmuni = dim(ObsOral)[1]
nper = dim(ObsOral)[2]

# INLA prior distributions Unif(0,Inf) distribution for standard
# deviations
sdunif = "expression:
  logdens = -log_precision/2;
  return(logdens)"

# Unif(0,1) distribution for the spatial smoothing parameter in LCAR
# random effects
lunif = "expression:
    beta = exp(theta)/(1+exp(theta));
    logdens = 0;
    log_jacobian = log(beta*(1-beta));
    return(logdens+log_jacobian)"
```

Definition of marginal spatial and temporal bases of functions

```
# Construct the temporal B-spline basis
#-------------------------------------
p = 3  ##Cubic B-splines
q = 3  ##3 Number of internal intervals

# Time covariate scaled into the [0,1] interval
xt = 1:nper
xt = (xt - min(xt))/(max(xt) - min(xt))
dist = (max(xt) - min(xt))/q
xtl = min(xt) - dist * 0.05
xtr = max(xt) + dist * 0.05
dxt = (xtr - xtl)/q
knotst = seq(xtl - p * dxt, xtr + p * dxt, by = dxt)

# The temporal B-spline basis
Bt = splineDesign(knotst, xt, p + 1)
# dimension of the basis
kt = ncol(Bt)

# Construct the spatial B-spline basis
#-------------------------------------
qh = 7  ## Number of internal intervals (horizontal)
qv = 14  ## Number of internal intervals (vertical)

# Longitudes scaled into the [0,1] interval
x1 = coordinates(VR.cart)[, 1]  ## Longitude covariate scaled
x1 = (x1 - min(x1))/(max(x1) - min(x1))  ## into the [0,1] interval
```

```
dist1 = (max(x1) - min(x1))/qh
x1l = min(x1) - dist1 * 0.05
x1r = max(x1) + dist1 * 0.05
dx1 = (x1r - x1l)/qh
knots1 = seq(x1l - p * dx1, x1r + p * dx1, by = dx1)

# The horizontal B-spline basis
B1 = splineDesign(knots1, x1, p + 1)
# dimension of the basis
k1 = ncol(B1)

# Latitudes scaled into the [0,1] interval
x2 = coordinates(VR.cart)[, 2]  ## Latitude covariate scaled
x2 = (x2 - min(x2))/(max(x2) - min(x2))  ## into the [0,1] interval
dist2 = (max(x2) - min(x2))/qv
x2l = min(x2) - dist2 * 0.05
x2r = max(x2) + dist2 * 0.05
dx2 = (x2r - x2l)/qv
knots2 = seq(x2l - p * dx2, x2r + p * dx2, by = dx2)

# The vertical B-spline basis
B2 = splineDesign(knots2, x2, p + 1)
# dimension of the basis
k2 = ncol(B2)

## Row-wise Kronecker product ##
Rten = function(X1, X2) {
    one1 = matrix(1, 1, ncol(X1))
    one2 = matrix(1, 1, ncol(X2))
    kronecker(X1, one2) * kronecker(one1, X2)
}

# The spatial B-spline basis
Bs = Rten(B2, B1)
ks = ncol(Bs)
```

Structure matrices for the penalties of the spatial and temporal terms

```
g = inla.read.graph("../Data/VR.graph")

# Structure matrix for the spatial term
#-------------------------------------
# ICAR structure matrix
R.xi = matrix(0, g$n, g$n)
for (i in 1:g$n) {
    R.xi[i, i] = g$nnbs[[i]]
    R.xi[i, g$nbs[[i]]] = -1
}
# auxiliar matrix for setting up a LCAR process
R.Leroux = diag(nmuni) - R.xi

# Structure matrix for the temporal term with a first order (RW1)
```

```r
# penalty
Dt = diff(diag(kt))
Pt = t(Dt) %*% Dt
```

Model without spatio-temporal interaction

```r
# Data for this model
Data.NoInt = list(O = as.vector(ObsOral), E = as.vector(ExpOral), intercept = c(1,
    rep(NA, nmuni + kt)), ID.area = c(NA, 1:nmuni, rep(NA, kt)), ID.year = c(rep(NA,
    1 + nmuni), 1:kt))

inter = rep(1, nmuni * nper)

# Design matrices for random effect terms
Ms = kronecker(matrix(1, nper, 1), diag(nmuni))
B_t = kronecker(Bt, matrix(1, nmuni, 1))

# Formula (we remove the default intercept and add it explicitally in
# order to include this term in the municipal predictions made by INLA.
# Otherwise, the default intercept would not be included in those
# predictions)
f.M1 = O ~ -1 + intercept + f(ID.area, model = "generic1", Cmatrix = R.Leroux,
    constr = TRUE, hyper = list(prec = list(prior = sdunif), beta = list(prior = lunif))) +
    f(ID.year, model = "rw1", constr = TRUE, hyper = list(prec = list(prior = sdunif)))

# INLA fit of the model and predictions at every combination of
# municipality and time interval
result.1 = inla(f.M1, family = "poisson", data = Data.NoInt, E = E, control.predictor = list(compute =
    A = cbind(inter, Ms, B_t), link = 1), control.compute = list(dic = TRUE),
    control.inla = list(strategy = strategy))
summary(result.1)
```

```
##
## Call:
## c("inla(formula = f.M1, family = \"poisson\", data = Data.NoInt, E = E, ",  "    control.compute = li
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          1.3409         15.7723          0.5369         17.6500
##
## Fixed effects:
##               mean    sd 0.025quant 0.5quant 0.975quant    mode kld
## intercept -0.1669 0.078    -0.3483  -0.1609    -0.0272 -0.1537   0
##
## Random effects:
## Name   Model
##  ID.area   Generic1 model
## ID.year   RW1 model
##
## Model hyperparameters:
##                           mean      sd 0.025quant 0.5quant 0.975quant   mode
## Precision for ID.area  5.063  1.3644     2.9594   4.8660     8.2848 4.4935
```

```
## Beta for ID.area       0.851  0.1092     0.5682   0.8787     0.9803 0.9414
## Precision for ID.year 13.431 17.6586     0.4881   7.8146    59.9016 1.0903
##
## Expected number of effective parameters(std dev): 88.07(13.03)
## Number of equivalent replicates : 73.58
##
## Deviance Information Criterion (DIC) ...............: 7338.55
## Deviance Information Criterion (DIC, saturated) ....: 3837.81
## Effective number of parameters .....................: 90.59
##
## Marginal log-Likelihood:  -3697.17
## Posterior marginals for linear predictor and fitted values computed
```

1-dimensional P-spline models

```
# Data for these models
Data.1d = list(O = as.vector(ObsOral), E = as.vector(ExpOral), intercept = c(1,
    rep(NA, nmuni + kt + nmuni * kt)), ID.area = c(NA, 1:nmuni, rep(NA,
    kt + nmuni * kt)), ID.year = c(rep(NA, 1 + nmuni), 1:kt, rep(NA, nmuni *
    kt)), ID.area.year = c(rep(NA, 1 + nmuni + kt), 1:(nmuni * kt)))

# Design matrix for the spatio-temporal term
B_st = kronecker(Bt, diag(nmuni))
```

iid (type I) penalty for the coefficients of the spatio-temporal term

```
# Linear constraint for the spatio-temporal term in this model
A.constr = kronecker(matrix(1, 1, nper) %*% Bt, matrix(1, 1, nmuni))

# Formula
f.M2.1 = O ~ -1 + intercept + f(ID.area, model = "generic1", Cmatrix = R.Leroux,
    constr = TRUE, hyper = list(prec = list(prior = sdunif), beta = list(prior = lunif))) +
    f(ID.year, model = "rw1", constr = TRUE, hyper = list(prec = list(prior = sdunif))) +
    f(ID.area.year, model = "iid", constr = FALSE, hyper = list(prec = list(prior = sdunif)),
        extraconstr = list(A = A.constr, e = 0))

# INLA fit of the model and predictions at every combination of
# municipality and time interval
result.2.1 = inla(f.M2.1, family = "poisson", data = Data.1d, E = E, control.predictor = list(compute =
    A = cbind(inter, Ms, B_t, B_st), link = 1), control.compute = list(dic = TRUE),
    control.inla = list(strategy = strategy))
summary(result.2.1)
```

```
##
## Call:
## c("inla(formula = f.M2.1, family = \"poisson\", data = Data.1d, E = E, ",  "    control.compute = lis
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          1.9035         57.3436          0.9259         60.1731
##
```

```
## Fixed effects:
##              mean    sd 0.025quant 0.5quant 0.975quant    mode kld
## intercept -0.1717 0.076    -0.3437  -0.1666    -0.0341 -0.1599   0
##
## Random effects:
## Name   Model
##  ID.area   Generic1 model
## ID.year   RW1 model
## ID.area.year   IID model
##
## Model hyperparameters:
##                                 mean       sd 0.025quant 0.5quant
## Precision for ID.area         5.3222   1.5121     3.0013   5.1019
## Beta for ID.area              0.8568   0.1064     0.5777   0.8845
## Precision for ID.year        13.6759  17.7152     0.6420   8.1609
## Precision for ID.area.year 170.2941 326.8536    15.5317  82.3512
##                            0.975quant    mode
## Precision for ID.area          8.8895  4.6908
## Beta for ID.area               0.9808  0.9437
## Precision for ID.year         59.5536  1.6091
## Precision for ID.area.year   873.9304 33.9332
##
## Expected number of effective parameters(std dev): 104.90(19.37)
## Number of equivalent replicates : 61.77
##
## Deviance Information Criterion (DIC) ...............: 7338.44
## Deviance Information Criterion (DIC, saturated) ....: 3837.71
## Effective number of parameters .....................: 106.95
##
## Marginal log-Likelihood:  -3698.29
## Posterior marginals for linear predictor and fitted values computed
```

Temporal (type II) penalty for the coefficients of the spatio-temporal term

```
# Temporal (type II) penalty for the coefficients of the
# spatio-temporal term
R = kronecker(Pt, diag(nmuni))

# nmuni linear constraints for the spatio-temporal term in this model.
# The splines coefficients for each municipality should sum 0.
A.constr = kronecker(matrix(1, 1, kt), diag(nmuni))

# Formula
f.M2.2 = O ~ -1 + intercept + f(ID.area, model = "generic1", Cmatrix = R.Leroux,
    constr = TRUE, hyper = list(prec = list(prior = sdunif), beta = list(prior = lunif))) +
    f(ID.year, model = "rw1", constr = TRUE, hyper = list(prec = list(prior = sdunif))) +
    f(ID.area.year, model = "generic0", Cmatrix = R, rankdef = nmuni, constr = TRUE,
        hyper = list(prec = list(prior = sdunif)), extraconstr = list(A = A.constr,
            e = rep(0, nmuni)))

# INLA fit of the model and predictions at every combination of
# municipality and time interval
```

```
result.2.2 = inla(f.M2.2, family = "poisson", data = Data.1d, E = E, control.predictor = list(compute =
    A = cbind(inter, Ms, B_t, B_st), link = 1), control.compute = list(dic = TRUE),
    control.inla = list(strategy = strategy))
summary(result.2.2)
```

```
##
## Call:
## c("inla(formula = f.M2.2, family = \"poisson\", data = Data.1d, E = E, ",  "    control.compute = lis
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          1.7545       2168.8964          3.1287       2173.7796
##
## Fixed effects:
##               mean     sd 0.025quant 0.5quant 0.975quant    mode   kld
## intercept -0.1705 0.0784     -0.345  -0.1655    -0.0264 -0.1573 1e-04
##
## Random effects:
## Name   Model
##  ID.area   Generic1 model
## ID.year   RW1 model
## ID.area.year   Generic0 model
##
## Model hyperparameters:
##                                 mean       sd 0.025quant 0.5quant
## Precision for ID.area         5.1214   1.3966     2.9738   4.9177
## Beta for ID.area              0.8537   0.1081     0.5712   0.8816
## Precision for ID.year        18.3426  24.5562     1.2837  10.9271
## Precision for ID.area.year 234.0653 460.3383    15.9089 109.8550
##                            0.975quant   mode
## Precision for ID.area          8.4246  4.534
## Beta for ID.area               0.9807  0.943
## Precision for ID.year         80.6732  3.420
## Precision for ID.area.year  1227.9284 38.571
##
## Expected number of effective parameters(std dev): 101.18(17.84)
## Number of equivalent replicates : 64.04
##
## Deviance Information Criterion (DIC) ...............: 7339.99
## Deviance Information Criterion (DIC, saturated) ....: 3839.25
## Effective number of parameters .....................: 104.08
##
## Marginal log-Likelihood:  -4182.21
## Posterior marginals for linear predictor and fitted values computed
```

Spatial (type III) penalty for the coefficients of the spatio-temporal term

```
# Spatial (type III) penalty for the coefficients of the
# spatio-temporal term
R = kronecker(diag(kt), R.xi)
# kt linear constraints for the spatio-temporal term in this model. The
# splines coefficients for each element in the basis should sum 0.
```

```r
A.constr = kronecker(diag(kt), matrix(1, 1, nmuni))

# Formula
f.M2.3 = O ~ -1 + intercept + f(ID.area, model = "generic1", Cmatrix = R.Leroux,
    constr = TRUE, hyper = list(prec = list(prior = sdunif), beta = list(prior = lunif))) +
    f(ID.year, model = "rw1", constr = TRUE, hyper = list(prec = list(prior = sdunif))) +
    f(ID.area.year, model = "generic0", Cmatrix = R, rankdef = kt, constr = TRUE,
        hyper = list(prec = list(prior = sdunif)), extraconstr = list(A = A.constr,
            e = rep(0, kt)))

# INLA fit of the model and predictions at every combination of
# municipality and time interval
result.2.3 = inla(f.M2.3, family = "poisson", data = Data.1d, E = E, control.predictor = list(compute =
    A = cbind(inter, Ms, B_t, B_st), link = 1), control.compute = list(dic = TRUE),
    control.inla = list(strategy = strategy))
summary(result.2.3)
```

```
##
## Call:
## c("inla(formula = f.M2.3, family = \"poisson\", data = Data.1d, E = E, ",  "    control.compute = lis
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          2.3276         66.4796          0.9824         69.7895
##
## Fixed effects:
##              mean     sd 0.025quant 0.5quant 0.975quant   mode   kld
## intercept -0.1731 0.0791    -0.3556  -0.1669    -0.0317 -0.159 1e-04
##
## Random effects:
## Name   Model
##  ID.area   Generic1 model
## ID.year   RW1 model
## ID.area.year   Generic0 model
##
## Model hyperparameters:
##                               mean      sd 0.025quant 0.5quant 0.975quant
## Precision for ID.area       5.9300  1.8931     3.1429   5.6159    10.5053
## Beta for ID.area            0.7522  0.1532     0.3843   0.7827     0.9596
## Precision for ID.year      13.2237 18.2003     0.4326   7.4295    60.6878
## Precision for ID.area.year 24.8616 28.1332     3.7657  16.4222    97.1053
##                               mode
## Precision for ID.area       5.0454
## Beta for ID.area            0.8727
## Precision for ID.year       0.9265
## Precision for ID.area.year 8.6323
##
## Expected number of effective parameters(std dev): 106.22(17.91)
## Number of equivalent replicates : 61.00
##
## Deviance Information Criterion (DIC) ...............: 7338.79
## Deviance Information Criterion (DIC, saturated) ....: 3838.05
## Effective number of parameters .....................: 108.89
##
```

```
## Marginal log-Likelihood:  -6058.88
## Posterior marginals for linear predictor and fitted values computed
```

Spatio-temporally structured (type IV) penalty for the coefficients of the spatio-temporal term

```
# Spatio-temporally structured (type IV) penalty for the coefficients
# of the spatio-temporal term
R = kronecker(Pt, R.xi)

# nmuni+kt linear constraints for the spatio-temporal term in this
# model. The splines coefficients for each element in the basis and
# municipality should sum 0.
A1 = kronecker(matrix(1, 1, kt), diag(nmuni))
A2 = kronecker(diag(kt), matrix(1, 1, nmuni))
A.constr = rbind(A1, A2)

# Formula
f.M2.4 = O ~ -1 + intercept + f(ID.area, model = "generic1", Cmatrix = R.Leroux,
    constr = TRUE, hyper = list(prec = list(prior = sdunif), beta = list(prior = lunif))) +
    f(ID.year, model = "rw1", constr = TRUE, hyper = list(prec = list(prior = sdunif))) +
    f(ID.area.year, model = "generic0", Cmatrix = R, rankdef = nmuni +
        kt - 1, constr = TRUE, hyper = list(prec = list(prior = sdunif)),
        extraconstr = list(A = A.constr, e = rep(0, nmuni + kt)))

# INLA fit of the model and predictions at every combination of
# municipality and time interval
result.2.4 = inla(f.M2.4, family = "poisson", data = Data.1d, E = E, control.predictor = list(compute =
    A = cbind(inter, Ms, B_t, B_st), link = 1), control.compute = list(dic = TRUE),
    control.inla = list(strategy = strategy))
summary(result.2.4)
```

```
##
## Call:
## c("inla(formula = f.M2.4, family = \"poisson\", data = Data.1d, E = E, ",  "    control.compute = lis
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          1.9653       2163.0221          1.2989       2166.2864
##
## Fixed effects:
##              mean     sd 0.025quant 0.5quant 0.975quant    mode   kld
## intercept -0.1659 0.0766    -0.3421  -0.1603    -0.0285 -0.1534 1e-04
##
## Random effects:
## Name   Model
##  ID.area   Generic1 model
## ID.year   RW1 model
## ID.area.year   Generic0 model
##
## Model hyperparameters:
##                                 mean        sd 0.025quant 0.5quant
## Precision for ID.area         5.1587    1.3975     2.9937   4.9611
```

```
## Beta for ID.area             0.8427    0.1105     0.5629   0.8683
## Precision for ID.year       14.0459   18.9542     0.4681   7.9983
## Precision for ID.area.year 462.2446 3204.9603    12.1942  89.5134
##                            0.975quant    mode
## Precision for ID.area          8.4475  4.5888
## Beta for ID.area               0.9794  0.9364
## Precision for ID.year         63.6668  1.0136
## Precision for ID.area.year  3000.6926 23.3748
##
## Expected number of effective parameters(std dev): 100.29(15.21)
## Number of equivalent replicates : 64.61
##
## Deviance Information Criterion (DIC) ...............: 7340.88
## Deviance Information Criterion (DIC, saturated) ....: 3840.14
## Effective number of parameters .....................: 103.65
##
## Marginal log-Likelihood:  -6148.97
## Posterior marginals for linear predictor and fitted values computed
```

2-dimensional P-spline models

```
# Additional design matrices for random effects:
B_s = kronecker(matrix(1, nper, 1), Bs)
Mt = kronecker(diag(nper), matrix(1, nmuni, 1))
# Redefinition of the design matrix of the spatio-temporal term as a
# function of the spatial basis B_s.
B_st = kronecker(diag(nper), Bs)

# RW1 Penalty function for the coefficients of the spatial spline
# (Longitudes)
D1 = diff(diag(k1))
P1 = t(D1) %*% D1
R1 = kronecker(diag(k2), P1)

# RW1 Penalty function for the coefficients of the spatial spline
# (Latitudes)
D2 = diff(diag(k2))
P2 = t(D2) %*% D2
R2 = kronecker(P2, diag(k1))

# Set of penalties for the spatial spline
Cmat.s = list(inla.as.sparse(R1), inla.as.sparse(R2))

# RW1 Penalty function for the coefficients of the spatial spline
# (temporal)
Dt = diff(diag(nper))
Pt = t(Dt) %*% Dt

# Data for the 2-dimensional spline models
Data.2d = list(O = as.vector(ObsOral), E = as.vector(ExpOral), intercept = c(1,
    rep(NA, ks + nper + ks * nper)), ID.area = c(NA, 1:ks, rep(NA, nper +
    ks * nper)), ID.year = c(rep(NA, 1 + ks), 1:nper, rep(NA, ks * nper)),
```

```
    ID.area.year = c(rep(NA, 1 + ks + nper), 1:(ks * nper)))
```

iid (type I) penalty for the coefficients of the spatio-temporal term

```
# linear constraint for the spatio-temporal term in this model.
A.constr = kronecker(matrix(1, 1, nper), matrix(1, 1, nmuni) %*% Bs)

# Formula
f.M3.1 = O ~ -1 + intercept + f(ID.area, model = "generic3", Cmatrix = Cmat.s,
    constr = TRUE, diagonal = 1e-06, hyper = list(prec1 = list(prior = sdunif),
        prec2 = list(prior = sdunif))) + f(ID.year, model = "rw1", constr = TRUE,
    hyper = list(prec = list(prior = sdunif))) + f(ID.area.year, model = "iid",
    constr = FALSE, hyper = list(prec = list(prior = sdunif)), extraconstr = list(A = rbind(A.constr),
        e = 0))

# INLA fit of the model and predictions at every combination of
# municipality and time interval
result.3.1 = inla(f.M3.1, family = "poisson", data = Data.2d, E = E, control.predictor = list(compute =
    A = cbind(inter, B_s, Mt, B_st), link = 1), control.compute = list(dic = TRUE),
    control.inla = list(strategy = strategy))
summary(result.3.1)
```

```
##
## Call:
## c("inla(formula = f.M3.1, family = \"poisson\", data = Data.2d, E = E, ",  "    control.compute = lis
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          2.4056         77.0482          0.7750         80.2288
##
## Fixed effects:
##              mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## intercept -0.1912 0.1213    -0.4334  -0.1908     0.0486 -0.1898   0
##
## Random effects:
## Name   Model
##  ID.area   Generic3 model
## ID.year   RW1 model
## ID.area.year   IID model
##
## Model hyperparameters:
##                                          mean      sd 0.025quant 0.5quant
## Precision for Cmatrix[[1]] for ID.area   1.593   1.359     0.2721    1.213
## Precision for Cmatrix[[2]] for ID.area   2.402   2.165     0.4046    1.783
## Precision for ID.year                  150.228 117.324    30.0545  118.486
## Precision for ID.area.year             268.777 808.792    18.4428   96.865
##                                        0.975quant    mode
## Precision for Cmatrix[[1]] for ID.area      5.200  0.6899
## Precision for Cmatrix[[2]] for ID.area      8.107  1.0007
## Precision for ID.year                     460.793 73.2976
## Precision for ID.area.year               1574.255 35.1819
##
```

```
## Expected number of effective parameters(std dev): 46.24(10.29)
## Number of equivalent replicates : 140.14
##
## Deviance Information Criterion (DIC) ...............: 7342.98
## Deviance Information Criterion (DIC, saturated) ....: 3842.24
## Effective number of parameters .....................: 49.18
##
## Marginal log-Likelihood:  -3697.28
## Posterior marginals for linear predictor and fitted values computed
```

Temporal (type II) penalty for the coefficients of the spatio-temporal term

```
# Temporal (type II) penalty for the coefficients of the
# spatio-temporal term
R = kronecker(Pt, diag(ks))

# ks linear constraints for the spatio-temporal term in this model. The
# splines coefficients for each element in the spatial basis should sum
# 0.
A.constr = -kronecker(matrix(1, 1, nper), diag(ks))

# Formula
f.M3.2 = O ~ -1 + intercept + f(ID.area, model = "generic3", Cmatrix = Cmat.s,
    constr = TRUE, diagonal = 1e-06, hyper = list(prec1 = list(prior = sdunif),
        prec2 = list(prior = sdunif))) + f(ID.year, model = "rw1", constr = TRUE,
    hyper = list(prec = list(prior = sdunif))) + f(ID.area.year, model = "generic0",
    Cmatrix = R, rankdef = ks, constr = TRUE, hyper = list(prec = list(prior = sdunif)),
    extraconstr = list(A = A.constr, e = rep(0, ks)))

# INLA fit of the model and predictions at every combination of
# municipality and time interval
result.3.2 = inla(f.M3.2, family = "poisson", data = Data.2d, E = E, control.compute = list(dic = TRUE)
    control.inla = list(strategy = strategy), control.predictor = list(compute = TRUE,
        A = cbind(inter, B_s, Mt, B_st), link = 1))
summary(result.3.2)
```

```
##
## Call:
## c("inla(formula = f.M3.2, family = \"poisson\", data = Data.2d, E = E, ",  "    control.compute = lis
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          2.4111        381.2732          0.5814        384.2657
##
## Fixed effects:
##              mean     sd 0.025quant 0.5quant 0.975quant  mode kld
## intercept -0.1914 0.1217    -0.4342   -0.191     0.0492 -0.19   0
##
## Random effects:
## Name   Model
##  ID.area   Generic3 model
## ID.year   RW1 model
## ID.area.year   Generic0 model
```

```
##
## Model hyperparameters:
##                                           mean       sd 0.025quant
## Precision for Cmatrix[[1]] for ID.area   1.574    1.334     0.2711
## Precision for Cmatrix[[2]] for ID.area   2.380    2.122     0.4043
## Precision for ID.year                  163.220  130.451    31.5449
## Precision for ID.area.year             473.346 1175.606    36.0200
##                                        0.5quant 0.975quant    mode
## Precision for Cmatrix[[1]] for ID.area    1.203      5.112  0.6873
## Precision for Cmatrix[[2]] for ID.area    1.775      7.981  1.0021
## Precision for ID.year                   127.585    507.944 77.6058
## Precision for ID.area.year              193.611   2632.901 73.6699
##
## Expected number of effective parameters(std dev): 42.04(7.611)
## Number of equivalent replicates : 154.15
##
## Deviance Information Criterion (DIC) ...............: 7342.82
## Deviance Information Criterion (DIC, saturated) ....: 3842.08
## Effective number of parameters .....................: 44.95
##
## Marginal log-Likelihood:  -3908.85
## Posterior marginals for linear predictor and fitted values computed
```

Spatial (type III) penalty for the coefficients of the spatio-temporal term

```
# Spatial (type III) penalty for the coefficients of the
# spatio-temporal term
RR1 = kronecker(diag(nper), R1)
RR2 = kronecker(diag(nper), R2)
Cmat.st = list(inla.as.sparse(RR1), inla.as.sparse(RR2))

# nper linear constraints for the spatio-temporal term in this model.
# The splines coefficients for each period should sum 0.
A.constr = kronecker(diag(nper), matrix(1, 1, ks))

# Formula
f.M3.3 = O ~ -1 + intercept + f(ID.area, model = "generic3", Cmatrix = Cmat.s,
    constr = TRUE, diagonal = 1e-06, hyper = list(prec1 = list(prior = sdunif),
        prec2 = list(prior = sdunif))) + f(ID.year, model = "rw1", constr = TRUE,
    hyper = list(prec = list(prior = sdunif))) + f(ID.area.year, model = "generic3",
    Cmatrix = Cmat.st, constr = TRUE, diagonal = 1e-06, extraconstr = list(A = A.constr,
        e = rep(0, nper)), hyper = list(prec1 = list(prior = sdunif), prec2 = list(prior = sdunif)))

# INLA fit of the model and predictions at every combination of
# municipality and time interval
result.3.3 = inla(f.M3.3, family = "poisson", data = Data.2d, E = E, control.predictor = list(compute =
    A = cbind(inter, B_s, Mt, B_st), link = 1), control.compute = list(dic = TRUE),
    control.inla = list(strategy = strategy))
summary(result.3.3)
```

```
##
## Call:
## c("inla(formula = f.M3.3, family = \"poisson\", data = Data.2d, E = E, ",  "    control.compute = lis
```

```
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          2.8032        126.9212          0.5406        130.2651
##
## Fixed effects:
##              mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## intercept -0.1911 0.1219    -0.4342  -0.1908       0.05 -0.1897   0
##
## Random effects:
## Name   Model
##  ID.area   Generic3 model
## ID.year   RW1 model
## ID.area.year   Generic3 model
##
## Model hyperparameters:
##                                                mean      sd 0.025quant
## Precision for Cmatrix[[1]] for ID.area        1.579   1.355     0.2699
## Precision for Cmatrix[[2]] for ID.area        2.382   2.131     0.4039
## Precision for ID.year                       169.862 135.298    32.2561
## Precision for Cmatrix[[1]] for ID.area.year  63.578 142.686     3.3293
## Precision for Cmatrix[[2]] for ID.area.year 246.456 401.071    14.6326
##                                             0.5quant 0.975quant    mode
## Precision for Cmatrix[[1]] for ID.area         1.200      5.175  0.6826
## Precision for Cmatrix[[2]] for ID.area         1.774      8.002  0.9999
## Precision for ID.year                        133.123    527.803 80.3974
## Precision for Cmatrix[[1]] for ID.area.year   27.028    350.944  8.0799
## Precision for Cmatrix[[2]] for ID.area.year  129.398   1204.794 37.8881
##
## Expected number of effective parameters(std dev): 43.37(8.334)
## Number of equivalent replicates : 149.40
##
## Deviance Information Criterion (DIC) ...............: 7343.97
## Deviance Information Criterion (DIC, saturated) ....: 3843.23
## Effective number of parameters .....................: 45.69
##
## Marginal log-Likelihood:  -3698.28
## Posterior marginals for linear predictor and fitted values computed
```

Spatio-temporally structured (type IV) penalty for the coefficients of the spatio-temporal term

```
## Spatio-temporally structured (type IV) penalty for the coefficients
## of the spatio-temporal term
RR1 = kronecker(Pt, R1)
RR2 = kronecker(Pt, R2)
Cmat.st = list(inla.as.sparse(RR1), inla.as.sparse(RR2))

# nper+ks linear constraints for the spatio-temporal term in this
# model. The splines coefficients for each period and element in the
# basis should sum 0.
A1 = kronecker(diag(nper), matrix(1, 1, ks))
```

```r
A2 = kronecker(matrix(1, 1, nper), diag(ks))
A.constr = rbind(A1, A2)

# Formula
f.M3.4 = O ~ -1 + intercept + f(ID.area, model = "generic3", Cmatrix = Cmat.s,
    constr = TRUE, diagonal = 1e-06, hyper = list(prec1 = list(prior = sdunif),
        prec2 = list(prior = sdunif))) + f(ID.year, model = "rw1", constr = TRUE,
    hyper = list(prec = list(prior = sdunif))) + f(ID.area.year, model = "generic3",
    Cmatrix = Cmat.st, constr = TRUE, diagonal = 1e-06, extraconstr = list(A = A.constr,
        e = rep(0, ks + nper)), hyper = list(prec1 = list(prior = sdunif),
        prec2 = list(prior = sdunif)))

# INLA fit of the model and predictions at every combination of
# municipality and time interval
result.3.4 = inla(f.M3.4, family = "poisson", data = Data.2d, E = E, control.predictor = list(compute =
    A = cbind(inter, B_s, Mt, B_st), link = 1), control.compute = list(dic = TRUE),
    control.inla = list(strategy = strategy))
summary(result.3.4)
```

```
##
## Call:
## c("inla(formula = f.M3.4, family = \"poisson\", data = Data.2d, E = E, ",  "    control.compute = lis
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          2.9562        741.8858          1.0040        745.8460
##
## Fixed effects:
##              mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## intercept -0.1922 0.1212    -0.4339  -0.1918      0.047 -0.1906   0
##
## Random effects:
## Name   Model
##  ID.area   Generic3 model
## ID.year   RW1 model
## ID.area.year   Generic3 model
##
## Model hyperparameters:
##                                                mean       sd 0.025quant
## Precision for Cmatrix[[1]] for ID.area        1.579    1.342     0.2714
## Precision for Cmatrix[[2]] for ID.area        2.392    2.131     0.4054
## Precision for ID.year                       191.350  175.804    33.6202
## Precision for Cmatrix[[1]] for ID.area.year  36.940   43.795     2.9453
## Precision for Cmatrix[[2]] for ID.area.year 549.474 1264.896    27.5394
##                                             0.5quant 0.975quant    mode
## Precision for Cmatrix[[1]] for ID.area         1.205      5.139  0.6877
## Precision for Cmatrix[[2]] for ID.area         1.785      8.023  1.0065
## Precision for ID.year                        140.359    656.817 80.4561
## Precision for Cmatrix[[1]] for ID.area.year   23.717    151.740  8.1129
## Precision for Cmatrix[[2]] for ID.area.year  230.137   3115.965 69.7871
##
## Expected number of effective parameters(std dev): 44.41(7.747)
## Number of equivalent replicates : 145.92
##
```

```
## Deviance Information Criterion (DIC) ...............: 7343.66
## Deviance Information Criterion (DIC, saturated) ....: 3842.92
## Effective number of parameters .....................: 47.51
##
## Marginal log-Likelihood:  -3698.28
## Posterior marginals for linear predictor and fitted values computed
```

3-dimensional P-spline models

```
# Design matrices for random effects
B = kronecker(Bt, Bs)
k = dim(B)[2]

# Data for these models
Data.3d = list(O = as.vector(ObsOral), E = as.vector(ExpOral), intercept = c(1,
    rep(NA, k)), ID.spline = c(NA, 1:k))

# RW1 Penalty function for the coefficients of the spatial spline
# (temporal)
Dt = diff(diag(kt))
Pt = t(Dt) %*% Dt

# Penalty function for the three-dimensional splines
R1 = kronecker(diag(kt), kronecker(diag(k2), P1))
R2 = kronecker(diag(kt), kronecker(P2, diag(k1)))
R3 = kronecker(Pt, kronecker(diag(k2), diag(k1)))
```

Spatio-temporally structured (type IV) penalty for the coefficients of the spatio-temporal term

```
Cmat = list(inla.as.sparse(R1), inla.as.sparse(R2), inla.as.sparse(R3))

f.M4.4 = O ~ -1 + intercept + f(ID.spline, model = "generic3", Cmatrix = Cmat,
    constr = TRUE, hyper = list(prec1 = list(prior = sdunif), prec2 = list(prior = sdunif),
        prec3 = list(prior = sdunif)))

result.4.4 = inla(f.M4.4, family = "poisson", data = Data.3d, E = E, control.predictor = list(compute =
    A = cbind(inter, B), link = 1), control.compute = list(dic = TRUE),
    control.inla = list(strategy = strategy))

summary(result.4.4)
```

```
##
## Call:
## c("inla(formula = f.M4.4, family = \"poisson\", data = Data.3d, E = E, ",  "    control.compute = lis
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          2.2341        162.0625          0.8048        165.1014
##
```

```
## Fixed effects:
##              mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## intercept -0.2121 0.1062    -0.4251  -0.2114    -0.0024 -0.2098   0
##
## Random effects:
## Name   Model
##  ID.spline   Generic3 model
##
## Model hyperparameters:
##                                           mean     sd 0.025quant 0.5quant
## Precision for Cmatrix[[1]] for ID.spline 0.4486 0.4632     0.0551   0.3123
## Precision for Cmatrix[[2]] for ID.spline 0.6248 0.6876     0.0830   0.4197
## Precision for Cmatrix[[3]] for ID.spline 2.5112 2.0809     0.4461   1.9377
##                                          0.975quant   mode
## Precision for Cmatrix[[1]] for ID.spline      1.675 0.1463
## Precision for Cmatrix[[2]] for ID.spline      2.420 0.2069
## Precision for Cmatrix[[3]] for ID.spline      8.035 1.1278
##
## Expected number of effective parameters(std dev): 45.50(5.799)
## Number of equivalent replicates : 142.41
##
## Deviance Information Criterion (DIC) ...............: 7353.24
## Deviance Information Criterion (DIC, saturated) ....: 3852.51
## Effective number of parameters .....................: 47.88
##
## Marginal log-Likelihood:  -3696.79
## Posterior marginals for linear predictor and fitted values computed
```

3-dimensional ANOVA type P-spline models

```
# Data for these models
Data.3dANOVA = list(O = as.vector(ObsOral), E = as.vector(ExpOral), intercept = c(1,
    rep(NA, ks + kt + ks * kt)), ID.area = c(NA, 1:ks, rep(NA, kt + ks *
    kt)), ID.year = c(rep(NA, 1 + ks), 1:kt, rep(NA, ks * kt)), ID.area.year = c(rep(NA,
    1 + ks + kt), 1:(ks * kt)))

# Design matrices
B_t = kronecker(Bt, matrix(1, nmuni, 1))
B_st = kronecker(Bt, Bs)

# Set of penalties for the temporal term
Cmat.t = list(inla.as.sparse(Pt))

# Set of penalties for the spatio-temporal term
RR1 = kronecker(diag(kt), kronecker(diag(k2), P1))
RR2 = kronecker(diag(kt), kronecker(P2, diag(k1)))
RR3 = kronecker(Pt, kronecker(diag(k2), diag(k1)))
```

iid (type IV) penalty for the coefficients of the spatio-temporal term

```
Cmat.st = list(inla.as.sparse(RR1), inla.as.sparse(RR2), inla.as.sparse(RR3))

# Formula
f.M5.4 = O ~ -1 + intercept + f(ID.area, model = "generic3", Cmatrix = Cmat.s,
    constr = TRUE, hyper = list(prec1 = list(prior = sdunif), prec2 = list(prior = sdunif))) +
    f(ID.year, model = "generic3", Cmatrix = Cmat.t, constr = TRUE, hyper = list(prec1 = list(prior = s
    f(ID.area.year, model = "generic3", Cmatrix = Cmat.st, constr = TRUE,
        hyper = list(prec1 = list(prior = sdunif), prec2 = list(prior = sdunif),
            prec3 = list(prior = sdunif)))

# INLA fit of the model and predictions at every combination of
# municipality and time interval
result.5.4 = inla(f.M5.4, family = "poisson", data = Data.3dANOVA, E = E,
    control.predictor = list(compute = TRUE, A = cbind(inter, B_s, B_t,
        B_st), link = 1), control.compute = list(dic = TRUE), control.inla = list(strategy = strategy))

summary(result.5.4)
```

```
##
## Call:
## c("inla(formula = f.M5.4, family = \"poisson\", data = Data.3dANOVA, ",  "    E = E, control.compute
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          3.8314        635.6393          0.8018        640.2724
##
## Fixed effects:
##               mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## intercept -0.2573 0.1499    -0.5606  -0.2557      0.036 -0.2528   0
##
## Random effects:
## Name   Model
##  ID.area   Generic3 model
## ID.year   Generic3 model
## ID.area.year   Generic3 model
##
## Model hyperparameters:
##                                                mean     sd 0.025quant
## Precision for Cmatrix[[1]] for ID.area        3.403  5.495     0.1508
## Precision for Cmatrix[[2]] for ID.area        5.098  8.459     0.2011
## Precision for Cmatrix[[1]] for ID.year       19.266 31.146     0.7677
## Precision for Cmatrix[[1]] for ID.area.year  1.232  3.075     0.0455
## Precision for Cmatrix[[2]] for ID.area.year  1.441  3.403     0.0610
## Precision for Cmatrix[[3]] for ID.area.year 35.849 66.243     1.4776
##                                             0.5quant 0.975quant   mode
## Precision for Cmatrix[[1]] for ID.area        1.7667     16.863 0.3739
## Precision for Cmatrix[[2]] for ID.area        2.5832     25.698 0.4888
## Precision for Cmatrix[[1]] for ID.year        9.9321     95.631 1.8501
## Precision for Cmatrix[[1]] for ID.area.year   0.4766      7.134 0.1101
## Precision for Cmatrix[[2]] for ID.area.year   0.5842      8.186 0.1496
## Precision for Cmatrix[[3]] for ID.area.year  16.9959    187.109 3.6450
##
```

```
## Expected number of effective parameters(std dev): 36.89(5.484)
## Number of equivalent replicates : 175.66
##
## Deviance Information Criterion (DIC) ...............: 7342.29
## Deviance Information Criterion (DIC, saturated) ....: 3841.56
## Effective number of parameters .....................: 38.58
##
## Marginal log-Likelihood:  -3690.47
## Posterior marginals for linear predictor and fitted values computed
```

DIC comparison of the models fitted

```
result.1$dic$dic
```

```
## [1] 7338.55
```

```
c(result.2.1$dic$dic, result.2.2$dic$dic, result.2.3$dic$dic, result.2.4$dic$dic)
```

```
## [1] 7338.442 7339.985 7338.791 7340.875
```

```
c(result.3.1$dic$dic, result.3.2$dic$dic, result.3.3$dic$dic, result.3.4$dic$dic)
```

```
## [1] 7342.981 7342.818 7343.969 7343.660
```

```
result.4.4$dic$dic
```

```
## [1] 7353.243
```

```
result.5.4$dic$dic
```

```
## [1] 7342.293
```

Computing times

```
result.1$cpu.used[4]/60
```

```
##     Total
## 0.2941675
```

```
c(result.2.1$cpu.used[4], result.2.2$cpu.used[4], result.2.3$cpu.used[4],
    result.2.4$cpu.used[4])/60
```

```
##     Total     Total     Total     Total
##  1.002885 36.229660  1.163159 36.104773
```

```
c(result.3.1$cpu.used[4], result.3.2$cpu.used[4], result.3.3$cpu.used[4],
    result.3.4$cpu.used[4])/60
```

```
##     Total     Total     Total     Total
##  1.337147  6.404428  2.171084 12.430767
```

```
result.4.4$cpu.used[4]/60
```

```
##    Total
## 2.751691
```

```
result.5.4$cpu.used[4]/60
```

```
##    Total
## 10.67121
```

Variance decomposition

```
VarDecomp = function(sSMRs, nmuni, nper) {
    log.sSMRs = log(sSMRs)
    grand.mean = matrix(mean(log.sSMRs), nrow = nmuni, ncol = nper)
    rows.mean = matrix(rep(apply(log.sSMRs, 1, mean), nper), nrow = nmuni,
        ncol = nper, byrow = FALSE)
    columns.mean = matrix(rep(apply(log.sSMRs, 2, mean), nmuni), nrow = nmuni,
        ncol = nper, byrow = TRUE)

    var.spat = var(apply(log.sSMRs, 1, mean))
    var.temp = var(apply(log.sSMRs, 2, mean))
    var.spattemp = var(c(log.sSMRs - rows.mean - columns.mean + grand.mean))
    var.total = var.spat + var.temp + var.spattemp
    dev = round(100 * c(var.spat, var.temp, var.spattemp)/var.total, 2)
    names(dev) = c("var.sp", "var.t", "var.sp-t")
    dev
}

VarDecomp(matrix(result.1$summary.fitted.values[1:(nmuni * nper), 1], nrow = nmuni,
    ncol = nper), nmuni, nper)
```

```
##   var.sp    var.t var.sp-t
##    60.75    39.25     0.00
```

```
VarDecomp(matrix(result.2.1$summary.fitted.values[1:(nmuni * nper), 1],
    nrow = nmuni, ncol = nper), nmuni, nper)
```

```
##   var.sp    var.t var.sp-t
##    61.93    37.95     0.12
```

```
VarDecomp(matrix(result.2.2$summary.fitted.values[1:(nmuni * nper), 1],
    nrow = nmuni, ncol = nper), nmuni, nper)
```

```
##   var.sp    var.t var.sp-t
##    62.70    37.16     0.14
```

```
VarDecomp(matrix(result.2.3$summary.fitted.values[1:(nmuni * nper), 1],
    nrow = nmuni, ncol = nper), nmuni, nper)
```

```
##   var.sp    var.t var.sp-t
##    63.24    34.73     2.02
```

```
VarDecomp(matrix(result.2.4$summary.fitted.values[1:(nmuni * nper), 1],
    nrow = nmuni, ncol = nper), nmuni, nper)
```

```
##   var.sp    var.t var.sp-t
##    63.17    35.87     0.96
```

```
VarDecomp(matrix(result.3.1$summary.fitted.values[1:(nmuni * nper), 1],
    nrow = nmuni, ncol = nper), nmuni, nper)
```

```
##   var.sp    var.t var.sp-t
```

```
##    74.63    25.01     0.36
```

```
VarDecomp(matrix(result.3.2$summary.fitted.values[1:(nmuni * nper), 1],
    nrow = nmuni, ncol = nper), nmuni, nper)
```

```
##   var.sp    var.t var.sp-t
##    77.09    22.27     0.64
```

```
VarDecomp(matrix(result.3.3$summary.fitted.values[1:(nmuni * nper), 1],
    nrow = nmuni, ncol = nper), nmuni, nper)
```

```
##   var.sp    var.t var.sp-t
##    74.45    24.97     0.59
```

```
VarDecomp(matrix(result.3.4$summary.fitted.values[1:(nmuni * nper), 1],
    nrow = nmuni, ncol = nper), nmuni, nper)
```

```
##   var.sp    var.t var.sp-t
##    76.29    22.39     1.32
```

```
VarDecomp(matrix(result.4.4$summary.fitted.values[1:(nmuni * nper), 1],
    nrow = nmuni, ncol = nper), nmuni, nper)
```

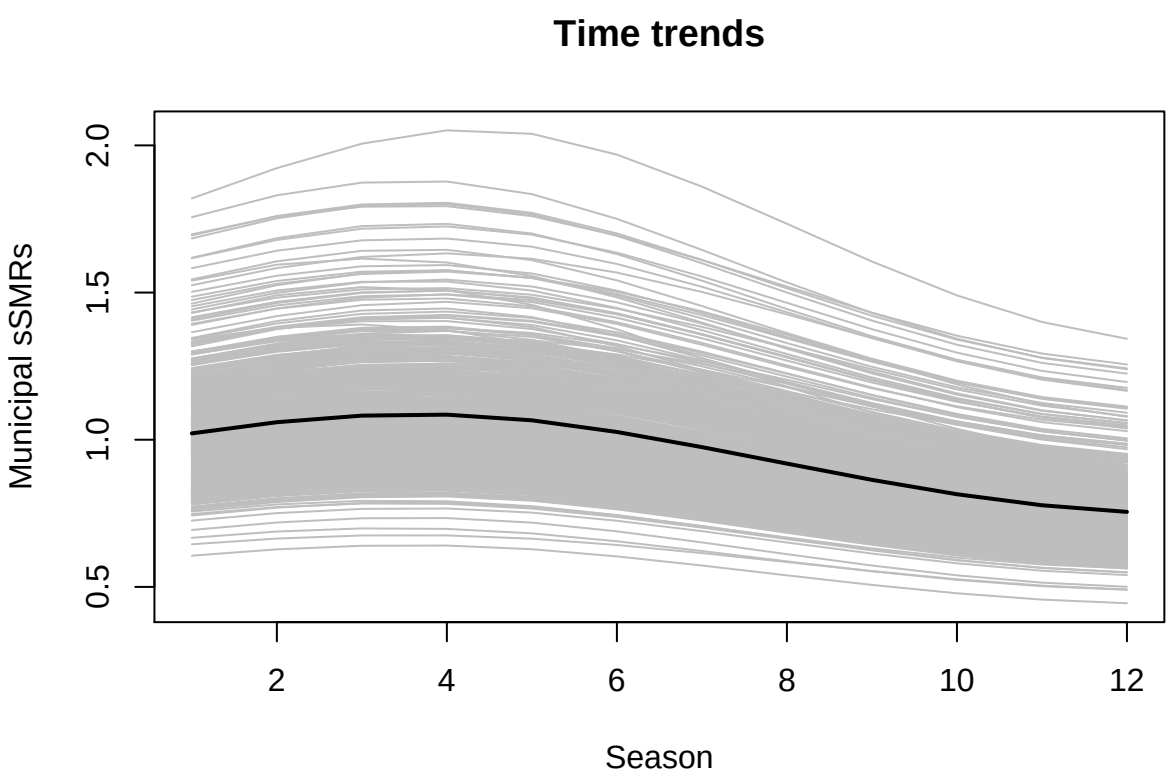
```
##   var.sp    var.t var.sp-t
##    81.97    12.28     5.74
```

```
VarDecomp(matrix(result.5.4$summary.fitted.values[1:(nmuni * nper), 1],
    nrow = nmuni, ncol = nper), nmuni, nper)
```

```
##   var.sp    var.t var.sp-t
##    78.48    20.82     0.70
```

sSMRs time trends plot for the one-dimensional model with unstructured penalty

```
rmes1 = matrix(result.2.1$summary.fitted.values[1:(nmuni * nper), 1], nrow = nmuni,
    ncol = nper)
rme.overall = apply(rmes1, 2, mean)
plot(rmes1[1, ], type = "n", ylim = c(min(c(rmes1)), max(c(rmes1))), xlab = "Season",
    ylab = "Municipal sSMRs", main = "Time trends")
for (i in 1:540) {
    lines(rmes1[i, ], type = "l", col = "grey")
}
lines(rme.overall, type = "l", main = "Common temporal trend", lwd = 2)
```


Time trends
Municipal sSMRs
0.5
1.0
1.5
2.0
2
4
6
8
10
12
Season