Example 7.1

Disease mapping: from foundations to multidimensional modeling

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

This document reproduces the analysis made at Example 7.1 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

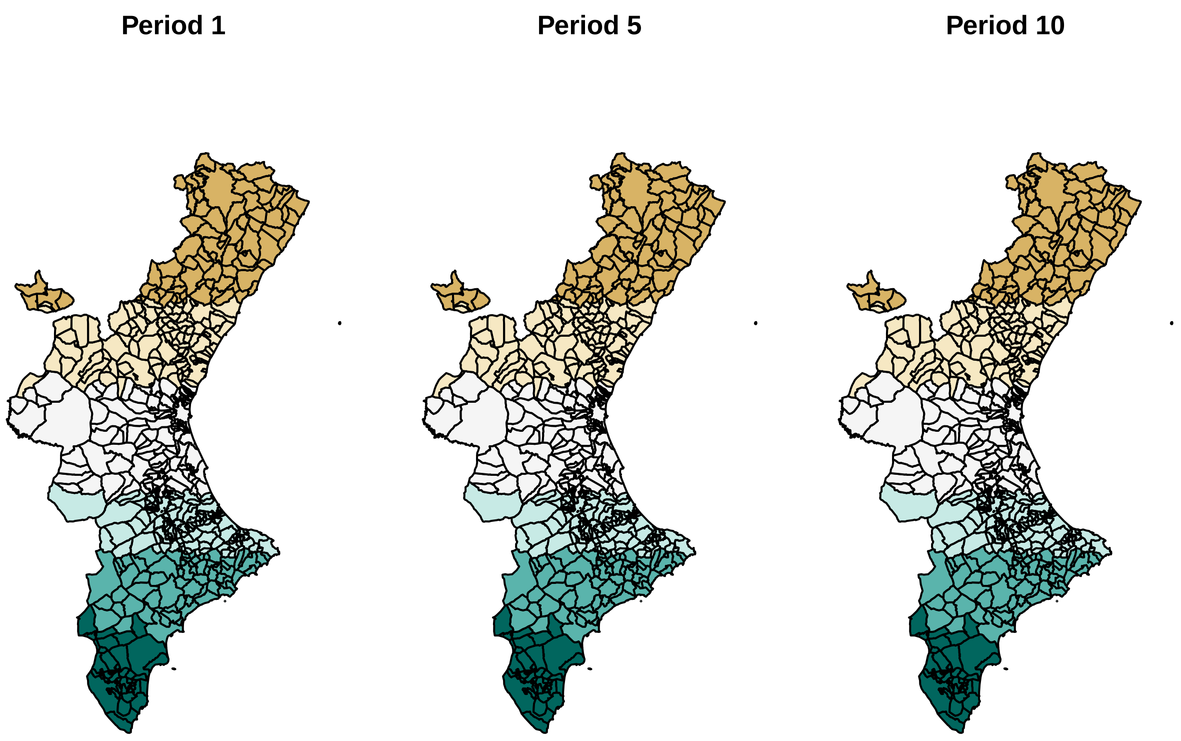
```r
# Libraries loading
#-----------------
if (!require(RColorBrewer)) {
    install.packages("RColorBrewer")
    library(RColorBrewer)
}
if (!require(rgdal)) {
    install.packages("rgdal")
    library(rgdal)
}
if (!require(R2WinBUGS)) {
    install.packages("R2WinBUGS")
    library(R2WinBUGS)
}
if (!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
#------------
load("../Data/VR.Rdata")
load("../Data/Exp.2011.Rdata")
```

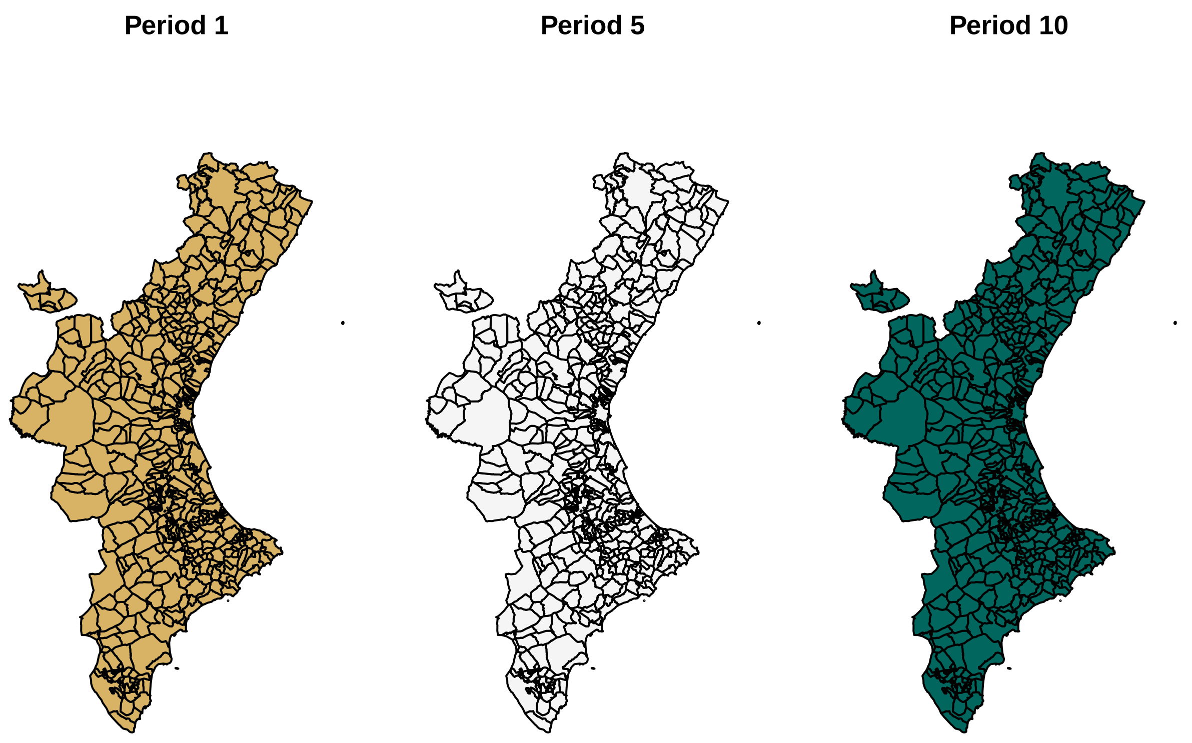
Analysis of simulated data sets

Generation of the simulated data sets

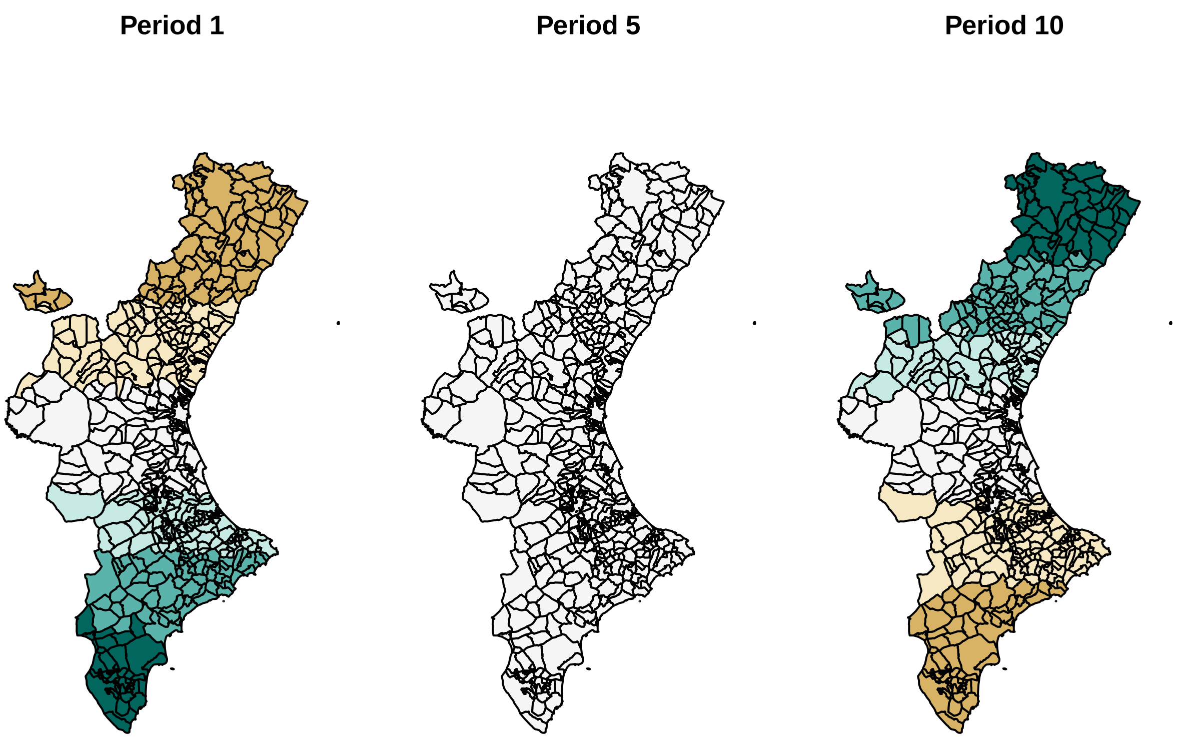
```r
nmuni = length(VR.cart)
centroids = matrix(nrow = nmuni, ncol = 2)
for (i in 1:nmuni) {
    centroids[i, ] = apply(VR.cart@polygons[[i]]@Polygons[[1]]@coords,
        2, mean)
}

miny = min(centroids[, 2])
maxy = max(centroids[, 2])

colors = brewer.pal(7, "BrBG")[7:1]

# Generation of the relative risks for each setting
#-------------------------------------------------
RRisks = list()
# Setting 1
RRisks[[1]] = matrix(nrow = nmuni, ncol = 10)
for (i in 1:10) {
    RRisks[[1]][, i] = 0.5 + ((centroids[, 2] - miny)/(maxy - miny))
}

par(mfrow = c(1, 3))
par(mar = c(1, 1, 2, 1) + 0.1)
per = 1
RRisks.cut = as.numeric(cut(RRisks[[1]][, per], c(-0.1, 1/1.5, 1/1.25,
    1/1.1, 1.1, 1.25, 1.5, 100)))
plot(VR.cart, col = colors[RRisks.cut], main = "Period 1")
per = 5
RRisks.cut = as.numeric(cut(RRisks[[1]][, per], c(-0.1, 1/1.5, 1/1.25,
    1/1.1, 1.1, 1.25, 1.5, 100)))
plot(VR.cart, col = colors[RRisks.cut], main = "Period 5")
per = 10
RRisks.cut = as.numeric(cut(RRisks[[1]][, per], c(-0.1, 1/1.5, 1/1.25,
    1/1.1, 1.1, 1.25, 1.5, 100)))
plot(VR.cart, col = colors[RRisks.cut], main = "Period 10")
```



```
# Setting 2
RRisks[[2]] = matrix(nrow = nmuni, ncol = 10)
r.per = seq(1.5, 0.5, length = 10)
for (i in 1:10) {
    RRisks[[2]][, i] = r.per[i]
}

par(mfrow = c(1, 3))
par(mar = c(1, 1, 2, 1) + 0.1)
per = 1
RRisks.cut = as.numeric(cut(RRisks[[2]][, per], c(-0.1, 1/1.5, 1/1.25,
    1/1.1, 1.1, 1.25, 1.5, 100)))
plot(VR.cart, col = colors[RRisks.cut], main = "Period 1")
per = 5
RRisks.cut = as.numeric(cut(RRisks[[2]][, per], c(-0.1, 1/1.5, 1/1.25,
    1/1.1, 1.1, 1.25, 1.5, 100)))
plot(VR.cart, col = colors[RRisks.cut], main = "Period 5")
per = 10
RRisks.cut = as.numeric(cut(RRisks[[2]][, per], c(-0.1, 1/1.5, 1/1.25,
    1/1.1, 1.1, 1.25, 1.5, 100)))
plot(VR.cart, col = colors[RRisks.cut], main = "Period 10")
```



```r
# Setting 3
RRisks[[3]] = matrix(nrow = nmuni, ncol = 10)
r.perN = seq(1.5, 0.5, length = 10)
r.perS = seq(0.5, 1.5, length = 10)

for (i in 1:10) {
    RRisks[[3]][, i] = r.perS[i] + ((centroids[, 2] - miny)/(maxy - miny)) *
        (r.perN[i] - r.perS[i])
}

par(mfrow = c(1, 3))
par(mar = c(1, 1, 2, 1) + 0.1)
per = 1
RRisks.cut = as.numeric(cut(RRisks[[3]][, per], c(-0.1, 1/1.5, 1/1.25,
    1/1.1, 1.1, 1.25, 1.5, 100)))
plot(VR.cart, col = colors[RRisks.cut], main = "Period 1")
per = 5
RRisks.cut = as.numeric(cut(RRisks[[3]][, per], c(-0.1, 1/1.5, 1/1.25,
    1/1.1, 1.1, 1.25, 1.5, 100)))
plot(VR.cart, col = colors[RRisks.cut], main = "Period 5")
per = 10
RRisks.cut = as.numeric(cut(RRisks[[3]][, per], c(-0.1, 1/1.5, 1/1.25,
    1/1.1, 1.1, 1.25, 1.5, 100)))
plot(VR.cart, col = colors[RRisks.cut], main = "Period 10")
```



```
# Generation of the observed cases for each setting
Obs = list()
Obs10 = list()
for (setting in 1:3) {
    Obs[[setting]] = matrix(nrow = nmuni, ncol = 10)
    Obs10[[setting]] = matrix(nrow = nmuni, ncol = 10)
    for (i in 1:10) {
        Obs[[setting]][, i] = rpois(nmuni, Exp.2011 * RRisks[[setting]][,
            i])
        Obs10[[setting]][, i] = rpois(nmuni, 10 * Exp.2011 * RRisks[[setting]][,
            i])
    }
}
```

Execution of the autoregressive model in WinBUGS for the generated data sets

```
# WinBUGS code for the autoregressive Spatio-temporal model
Autoregressive = function() {
    for (i in 1:nmuni) {
        for (j in 1:nperiods) {
            Obs[i, j] ~ dpois(lambda[i, j])
            # Modelling of the mean for every municipality and period
            log(lambda[i, j]) <- log(Exp[i, j]) + log.theta[i, j]
            # SMR for every municipality and period
            log.theta[i, j] <- mediainter + sd.inter * inter[j] + ST[i,
                j]
        }
    }

    # Spatio-temporal effect for the first period
    for (i in 1:nmuni) {
        ST[i, 1] <- pow(1 - ro * ro, -0.5) * BYM[i, 1]
        BYM[i, 1] <- sd.het * psi[i, 1] + sd.spat * phi[1, i]
        psi[i, 1] ~ dnorm(0, 1)
    }
    phi[1, 1:nmuni] ~ car.normal(map[], w[], nvec[], 1)
    # Spatio-temporal effect for the subsequent periods
    for (j in 2:nperiods) {
        for (i in 1:nmuni) {
            ST[i, j] <- ro * ST[i, j - 1] + BYM[i, j]
            BYM[i, j] <- sd.het * psi[i, j] + sd.spat * phi[j, i]
            psi[i, j] ~ dnorm(0, 1)
        }
        phi[j, 1:nmuni] ~ car.normal(map[], w[], nvec[], 1)
    }

    # Prior distribution for the mean risk for every municipality and
    # period
    mediainter ~ dflat()
    # Prior distribution for the global time trend
    inter[1:nperiods] ~ car.normal(mapT[], wT[], nvecT[], 1)
    # Prior distribution for the precision parameters in the model
    sd.inter ~ dunif(0, 5)
    sd.het ~ dunif(0, 5)
    sd.spat ~ dunif(0, 5)
    # Prior distribution for the temporal dependence parameter
    ro ~ dunif(-1, 1)
}

nperiods = 10

adjT = c(2, c(1, 3), c(2, 4), c(3, 5), c(4, 6), c(5, 7), c(6, 8), c(7,
    9), c(8, 10), 9)
numT = c(1, rep(2, 8), 1)
indexT = c(1, cumsum(numT))

Result = list()
Result10 = list()

# WinBUGS calls for executing the model above on the simulated data
# sets
for (setting in 1:3) {

    Exp = matrix(nrow = nmuni, ncol = 10)
    for (j in 1:10) {
        Exp[, j] = Exp.2011
    }

    data = list(Obs = Obs[[setting]], Exp = Exp, nmuni = nmuni, nperiods = nperiods,
        w = rep(1, length(VR.wb$adj)), nvec = VR.wb$num, map = VR.wb$adj,
        wT = rep(1, length(adjT)), nvecT = numT, mapT = adjT)
    inits = function() {
        list(ro = runif(1, -1, 1), mediainter = rnorm(1, 0, 1), sd.inter = runif(1,
            0, 0.5), sd.het = runif(1, 0, 0.5), sd.spat = runif(1, 0, 0.5),
            psi = matrix(rnorm(nperiods * nmuni, 0, 1), ncol = nperiods,
                nrow = nmuni), phi = matrix(rnorm(nperiods * nmuni, 0,
                1), nrow = nperiods, ncol = nmuni))
    }
    param = c("log.theta", "mediainter", "sd.inter", "sd.het", "sd.spat",
        "ro")

    Result[[setting]] = pbugs(data = data, inits = inits, parameters = param,
        model.file = Autoregressive, n.iter = 5000, n.burnin = 1000, DIC = F,
        n.chains = 3, bugs.seed = 1)

    data = list(Obs = Obs10[[setting]], Exp = 10 * Exp, nmuni = nmuni,
        nperiods = nperiods, w = rep(1, length(VR.wb$adj)), nvec = VR.wb$num,
        map = VR.wb$adj, wT = rep(1, length(adjT)), nvecT = numT, mapT = adjT)
    Result10[[setting]] = pbugs(data = data, inits = inits, parameters = param,
        model.file = Autoregressive, n.iter = 5000, n.burnin = 1000, DIC = F,
        n.chains = 3, bugs.seed = 1)
}
```

Convergence checking for the models executed

```
# Computing time
Result[[1]]$exec_time
```

```
## Time difference of 36.3299 mins
```

```
# Convergence checking
summary(Result[[1]]$summary[, "Rhat"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9995  1.0015  1.0037  1.0050  1.0070  1.0580
```

```
summary(Result[[1]]$summary[, "n.eff"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    48.0   310.0   600.0   624.8  1000.0  1000.0
```

```
# Computing time
Result[[2]]$exec_time
```

```
## Time difference of 37.20109 mins
```

```
# Convergence checking
summary(Result[[2]]$summary[, "Rhat"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9995  1.0018  1.0040  1.0061  1.0078  1.1037
```

```r
summary(Result[[2]]$summary[, "n.eff"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    40.0   330.0   670.0   646.1  1000.0  1000.0
```

```r
# Computing time
Result[[3]]$exec_time
```

```
## Time difference of 38.03519 mins
```

```r
# Convergence checking
summary(Result[[3]]$summary[, "Rhat"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.9995  1.0022  1.0048  1.0065  1.0088  1.0621
```

```r
summary(Result[[3]]$summary[, "n.eff"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    37.0   270.0   540.0   594.4  1000.0  1000.0
```

```r
# Computing time
Result10[[1]]$exec_time
```

```
## Time difference of 36.0586 mins
```

```r
# Convergence checking
summary(Result10[[1]]$summary[, "Rhat"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.9995  1.0009  1.0024  1.0037  1.0051  1.0590
```

```r
summary(Result10[[1]]$summary[, "n.eff"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    47.0   430.0   890.0   723.7  1000.0  1000.0
```

```r
# Computing time
Result10[[2]]$exec_time
```

```
## Time difference of 37.97588 mins
```

```r
# Convergence checking
summary(Result10[[2]]$summary[, "Rhat"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.9995  1.0014  1.0038  1.0056  1.0076  1.1005
```

```r
summary(Result10[[2]]$summary[, "n.eff"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    29.0   320.0   650.0   640.1  1000.0  1000.0
```

```r
# Computing time
Result10[[3]]$exec_time
```

```
## Time difference of 38.04845 mins
```

```r
# Convergence checking
summary(Result10[[3]]$summary[, "Rhat"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.9995  1.0011  1.0030  1.0047  1.0062  1.2360
```

```
summary(Result10[[3]]$summary[, "n.eff"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    13.0   330.0   680.0   653.6  1000.0  1000.0
```

Variance decompositions

```
decomp = array(dim = c(3, 3, 3), dimnames = list(c("Setting 1", "Setting2",
    "Setting3"), c("2.5%", "50%", "97.5%"), c("S", "T", "ST")))
decomp.mean = matrix(nrow = 3, ncol = 3, dimnames = list(c("Setting1",
    "Setting2", "Setting3"), c("S", "T", "ST")))
for (setting in 1:3) {
    nIter = dim(Result[[setting]]$sims.list$log.theta)[1]
    components = matrix(nrow = nIter, ncol = 4)
    for (i in 1:nIter) {
        m = mean(Result[[setting]]$sims.list$log.theta[i, , ])
        S = apply(Result[[setting]]$sims.list$log.theta[i, , ], 1, mean) -
            m
        T = apply(Result[[setting]]$sims.list$log.theta[i, , ], 2, mean) -
            m
        ST = Result[[setting]]$sims.list$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)
    decomp[setting, , ] = apply(aux, 2, quantile, c(0.025, 0.5, 0.975))
    decomp.mean[setting, ] = apply(aux, 2, mean)
}
# quantiles of spatial, temporal and spatio-temporal components for
# each setting
decomp[1, , ] * 100
```

```
##              S            T        ST
## 2.5%  62.19386  0.008006354  1.645084
## 50%   84.71792  2.853927061 10.723018
## 97.5% 97.10326 12.822562934 33.575082
```

```
decomp[2, , ] * 100
```

```
##               S        T         ST
## 2.5%  0.0178301 78.44491  0.2440039
## 50%   1.1470402 93.30116  5.0748725
## 97.5% 6.8010804 99.69503 17.7010569
```

```
decomp[3, , ] * 100
```

```
##               S           T       ST
## 2.5%   9.705283  0.01759927 45.55651
## 50%   26.267752  1.20939752 70.94716
## 97.5% 51.479693 12.43938503 88.26735
```

```
# posterior mean of the percentage of variance per component for each
# setting
decomp.mean * 100
```

```
##                   S         T        ST
## Setting1 83.417556  3.931460 12.650984
## Setting2  1.759185 92.194022  6.046793
## Setting3 27.576305  2.536917 69.886778
```

Variance decompositions for the data sets with expected cases 10 times larger

```
decomp10 = array(dim = c(3, 3, 3), dimnames = list(c("Setting1", "Setting2",
    "Setting3"), c("2.5%", "50%", "97.5%"), c("S", "T", "ST")))
decomp10.mean = matrix(nrow = 3, ncol = 3, dimnames = list(c("Setting1",
    "Setting2", "Setting3"), c("S", "T", "ST")))
for (setting in 1:3) {
    nIter = dim(Result10[[setting]]$sims.list$log.theta)[1]
    components = matrix(nrow = nIter, ncol = 4)
    for (i in 1:nIter) {
        m = mean(Result10[[setting]]$sims.list$log.theta[i, , ])
        S = apply(Result10[[setting]]$sims.list$log.theta[i, , ], 1, mean) -
            m
        T = apply(Result10[[setting]]$sims.list$log.theta[i, , ], 2, mean) -
            m
        ST = Result10[[setting]]$sims.list$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)
    decomp10[setting, , ] = apply(aux, 2, quantile, c(0.025, 0.5, 0.975))
    decomp10.mean[setting, ] = apply(aux, 2, mean)
}
# quantiles of spatial, temporal and spatio-temporal components for
# each setting
decomp10[1, , ] * 100
```

```
##              S            T        ST
## 2.5%  95.28471 0.0001983652 0.2396576
## 50%   98.52353 0.0635202674 1.3078809
## 97.5% 99.68244 0.7641258705 4.6572998
```

```
decomp10[2, , ] * 100
```

```
##                S        T        ST
## 2.5%  0.01286651 95.10054 0.1583825
## 50%   0.13816422 98.15504 1.6473041
## 97.5% 0.67942809 99.80615 4.4985835
```

```
decomp10[3, , ] * 100
```

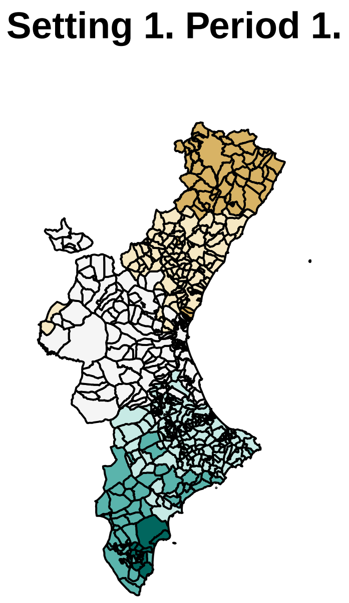
```
##              S           T        ST
## 2.5%   5.81779 0.009682832 76.90955
## 50%   10.88098 1.542052375 87.11946
## 97.5% 19.82124 7.121782334 93.00312
```

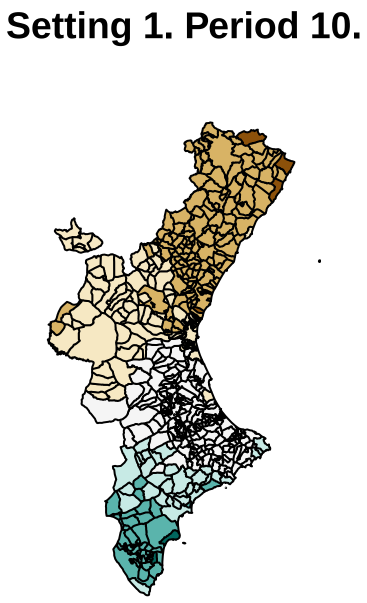
```r
# posterior mean of the percentage of variance per component for each
# setting
decomp10.mean * 100
```

```
##                   S          T        ST
## Setting1 98.2443394  0.1526936  1.602967
## Setting2  0.1984589 97.9973339  1.804207
## Setting3 11.5152170  2.0410384 86.443745
```

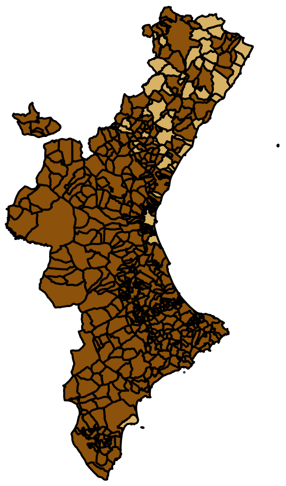
Choropleth maps for period 1 and 10 for each setting

```r
# choropleth maps of some patterns fitted
for (setting in 1:3) {
    par(mfrow = c(1, 2))
    per = 1
    aux = cut(apply(exp(Result[[setting]]$sims.list$log.theta[, , per]),
        2, mean), c(0, 0.66, 0.8, 0.9, 1.1, 1.25, 1.5, 10))
    plot(VR.cart, col = brewer.pal(7, "BrBG")[7:1][aux], main = paste0("Setting ",
        setting, ". Period 1."))
    per = 10
    aux = cut(apply(exp(Result[[setting]]$sims.list$log.theta[, , per]),
        2, mean), c(0, 0.66, 0.8, 0.9, 1.1, 1.25, 1.5, 10))
    plot(VR.cart, col = brewer.pal(7, "BrBG")[7:1][aux], main = paste0("Setting ",
        setting, ". Period 10."))
}
```

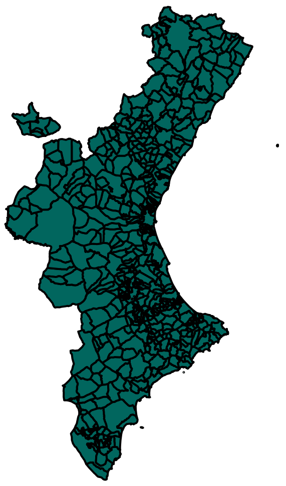


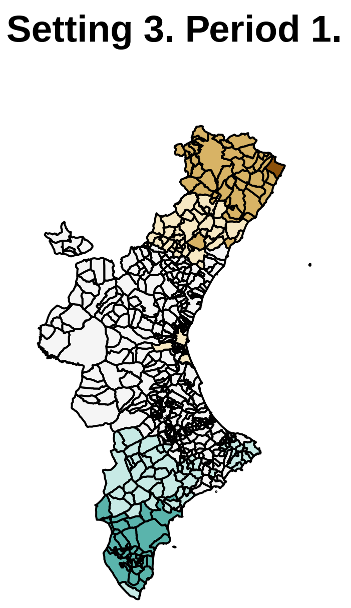


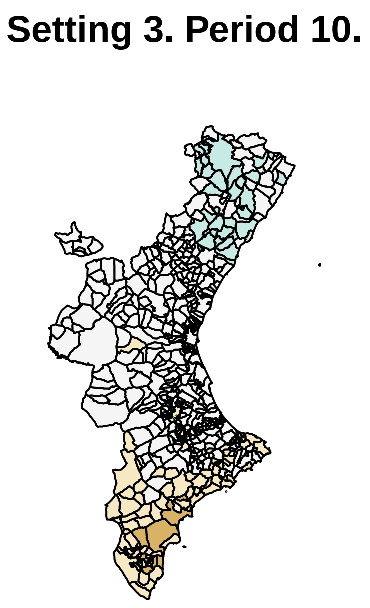
Setting 2. Period 1.



Setting 2. Period 10.

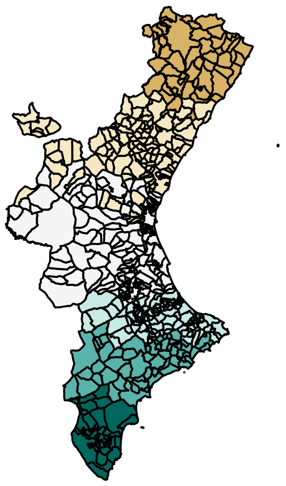




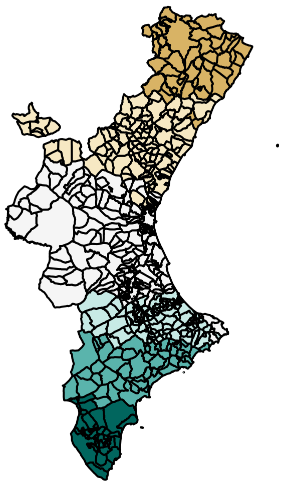


```r
# choropleth maps of some patterns fitted (expected and observed cases
# 10 times larger)
for (setting in 1:3) {
    par(mfrow = c(1, 2))
    per = 1
    aux = cut(apply(exp(Result10[[setting]]$sims.list$log.theta[, , per]),
        2, mean), c(0, 0.66, 0.8, 0.9, 1.1, 1.25, 1.5, 10))
    plot(VR.cart, col = brewer.pal(7, "BrBG")[7:1][aux], main = paste0("Setting ",
        setting, ". Period 1."))
    per = 10
    aux = cut(apply(exp(Result10[[setting]]$sims.list$log.theta[, , per]),
        2, mean), c(0, 0.66, 0.8, 0.9, 1.1, 1.25, 1.5, 10))
    plot(VR.cart, col = brewer.pal(7, "BrBG")[7:1][aux], main = paste0("Setting ",
        setting, ". Period 10."))
}
```

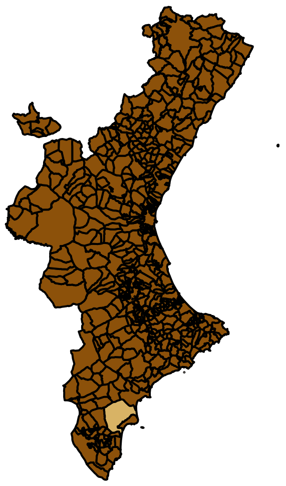
Setting 1. Period 1.



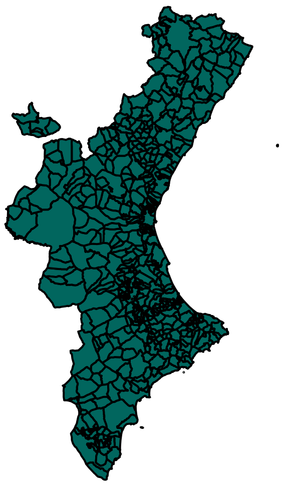
Setting 1. Period 10.

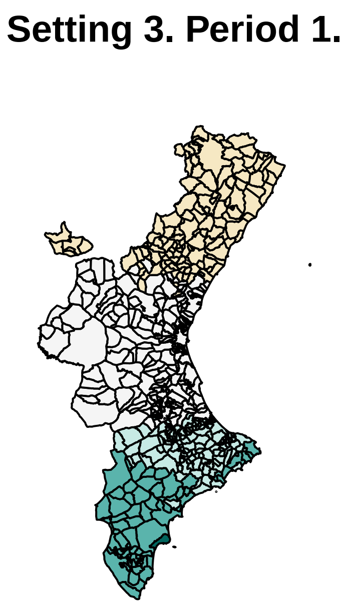


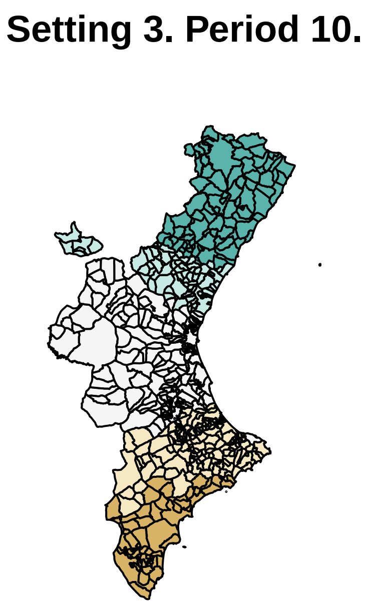
Setting 2. Period 1.



Setting 2. Period 10.







Analysis of real data from the Valencian Region

Data not provided in order to preserve confidentiality. The model and variance decomposition carried out for the real data set is exactly the same than that used for the simulated data sets.

We reproduce here only those combination of genders and causes of death that are shown in Table 7.2.

Execution of WinBUGS models for the real data sets

```
load("../Data/ET-causes.Rdata")

causes = c("Men. Oral", "Men. Stomach", "Women. Stomach", "Men. Lung",
    "Women. Lung", "Men. Bladder", "Men. Cereb.", "Women. Cereb.", "Men. Pneumo.",
    "Women. Pneumo.")

nperiods.12 <- 12
adjT.12 = c(rbind(2:nperiods.12, 1:(nperiods.12 - 1)))
numT.12 = c(1, rep(2, nperiods.12 - 2), 1)
indexT.12 = c(1, cumsum(numT.12))

Result.causes = list()
for (i in 1:10) {
    data = list(Obs = Obs.Causes[[i]], Exp = Exp.Causes[[i]], nmuni = nmuni,
        nperiods = nperiods.12, w = rep(1, length(VR.wb$adj)), nvec = VR.wb$num,
        map = VR.wb$adj, wT = rep(1, length(adjT.12)), nvecT = numT.12,
        mapT = adjT.12)
    inits = function() {
        list(ro = runif(1, -1, 1), mediainter = rnorm(1, 0, 1), sd.inter = runif(1,
            0, 0.5), sd.het = runif(1, 0, 0.5), sd.spat = runif(1, 0, 0.5),
            psi = matrix(rnorm(nperiods.12 * nmuni, 0, 1), ncol = nperiods.12,
                nrow = nmuni), phi = matrix(rnorm(nperiods.12 * nmuni,
                0, 1), nrow = nperiods.12, ncol = nmuni))
    }
    param = c("log.theta", "mediainter", "sd.inter", "sd.het", "sd.spat",
        "ro")

    Result.causes[[i]] = pbugs(data = data, inits = inits, parameters = param,
        model.file = Autoregressive, n.iter = 5000, n.burnin = 1000, DIC = F,
        n.chains = 3, bugs.seed = 1)
}
```

Variance decompositions

```
decomp.causes = array(dim = c(10, 3, 3), dimnames = list(causes, c("2.5%",
    "50%", "97.5%"), c("S", "T", "ST")))
decomp.causes.mean = matrix(nrow = 10, ncol = 3, dimnames = list(causes,
    c("S", "T", "ST")))
for (setting in 1:10) {
    nIter = dim(Result.causes[[setting]]$sims.list$log.theta)[1]
    components = matrix(nrow = nIter, ncol = 4)
    for (i in 1:nIter) {
        m = mean(Result.causes[[setting]]$sims.list$log.theta[i, , ])
        S = apply(Result.causes[[setting]]$sims.list$log.theta[i, , ],
            1, mean) - m
        T = apply(Result.causes[[setting]]$sims.list$log.theta[i, , ],
            2, mean) - m
        ST = Result.causes[[setting]]$sims.list$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)
    decomp.causes[setting, , ] = apply(aux, 2, quantile, c(0.025, 0.5,
        0.975))
    decomp.causes.mean[setting, ] = apply(aux, 2, mean)
}
# posterior mean of the percentage of variance per component for each
# setting
decomp.causes.mean * 100
```

```
##                        S          T        ST
## Men. Oral       70.29817 16.4235441 13.278288
## Men. Stomach    39.58204 52.8081411  7.609819
## Women. Stomach 27.41250 65.7820986  6.805401
## Men. Lung       91.72446  2.2222099  6.053328
## Women. Lung    54.39131 32.5021517 13.106541
## Men. Bladder   83.86089  0.4950175 15.644088
## Men. Cereb.    22.34811 69.8476201  7.804267
## Women. Cereb.  20.98026 67.7251794 11.294559
## Men. Pneumo.   56.59353 17.0026602 26.403812
## Women. Pneumo. 46.37459 17.2301926 36.395218
```

```
# posterior 2.5%, 50% and 97.5% quantiles of spatial, temporal and
# spatio-temporal components for each setting
decomp.causes[1, , ]
```

```
##               S          T         ST
## 2.5%  0.5367309 0.07519768 0.02472723
## 50%   0.7083701 0.16025465 0.12432767
## 97.5% 0.8312270 0.28256256 0.29572792
```

```
decomp.causes[2, , ]
```

```
##               S         T         ST
## 2.5%  0.2945958 0.4191649 0.01175617
## 50%   0.3958668 0.5295295 0.07194011
## 97.5% 0.5030321 0.6278358 0.16014217
```

```
decomp.causes[3, , ]
```

```
##               S         T          ST
## 2.5%  0.1352251 0.5100348 0.009193517
## 50%   0.2733616 0.6570479 0.060666571
## 97.5% 0.4120795 0.8030045 0.166534374
```

```
decomp.causes[4, , ]
```

```
##               S           T         ST
## 2.5%  0.8779576 0.009757925 0.02819362
## 50%   0.9180803 0.021941795 0.05893249
## 97.5% 0.9528366 0.037385920 0.10122212
```

```
decomp.causes[5, , ]
```

```
##               S         T         ST
## 2.5%  0.4036846 0.1986284 0.03251101
## 50%   0.5443945 0.3218129 0.12561180
## 97.5% 0.6773613 0.4778121 0.25029857
```

```
decomp.causes[6, , ]
```

```
##               S            T         ST
## 2.5%  0.7348290 1.973574e-05 0.06795961
## 50%   0.8415926 2.287830e-03 0.15342608
## 97.5% 0.9305368 2.538436e-02 0.26135294
```

```
decomp.causes[7, , ]
```

```
##               S         T         ST
## 2.5%  0.1905488 0.6598916 0.05713331
## 50%   0.2234699 0.6978485 0.07810108
## 97.5% 0.2597054 0.7360914 0.10022997
```

```r
decomp.causes[8, , ]
```

```
##                S         T         ST
## 2.5%  0.1805839 0.6448316 0.09302632
## 50%   0.2098064 0.6767585 0.11252124
## 97.5% 0.2380508 0.7105449 0.13538117
```

```r
decomp.causes[9, , ]
```

```
##                S         T        ST
## 2.5%  0.4624372 0.1120453 0.1706953
## 50%   0.5666150 0.1676155 0.2642729
## 97.5% 0.6647495 0.2410154 0.3595841
```

```r
decomp.causes[10, , ]
```

```
##                S         T        ST
## 2.5%  0.3655442 0.1050157 0.2440360
## 50%   0.4604431 0.1698454 0.3653376
## 97.5% 0.5727941 0.2525612 0.4702528
```