Multivariate BYM

Proposal: sparse precision parametrisation for the multivariate BYM model.

In this note, we attempt at providing a parametrisation for the multivariate BYM model resulting in a sparse precision matrix.

Parametrisation for the univariate BYM Before doing so, we start by illustrating the well-established sparse parametrisation of the univariate BYM:

This model, proposed by (Riebler et al. 2016) takes the form:

$$y = \sigma\left(\sqrt{\phi}U + \sqrt{1-\phi}V\right)$$

Where $U \sim N(0, L^+)$, $V \sim N(0, I_n)$, L denotes the graph Laplacian matrix, already scaled in order that $\text{diag}(L)$ has a geometric mean equal to 1; $n = \text{card}(y)$. σ^2 is the scale parameter.

Now, despite $\text{Prec}[y \mid \phi, \sigma] = \sigma^{-2}\left(\phi L + (1-\phi)I_n\right)^{-1}$ which has no reason at all to be a sparse matrix, the joint random vector $\begin{pmatrix} y \\ U \end{pmatrix}$ has indeed a sparse precision. This can be seen considering that $\ln \pi(y, U) = \ln \pi(y|U) + \ln \pi(U)$ and $y|U \sim N(\sigma\sqrt{\phi}U, \sigma^2(1-\phi)I_n)$ thus, with some passages (Riebler et al. 2016),

$$\text{Prec}(y, U|\sigma, \phi) = \begin{pmatrix} \frac{1}{\sigma^2(1-\phi)}I_n & -\frac{\sqrt{\phi}}{\sigma(1-\phi)}I_n \\ -\frac{\sqrt{\phi}}{\sigma(1-\phi)}I_n & \frac{\phi}{1-\phi}I_n + L \end{pmatrix} \tag{1}$$

Which is made of four sparse blocks as long as L is sparse as well.

Hence the univariate BYM model, other than being scalable and allowing for PC priors setting on hyperparameters, can also be fitted in an efficient way leveraging on precision sparsity. And this is indeed the model implemented in `R-INLA`, using the specification `INLA::f(..., model = "bym2", ...)`.

While this parametrisation is essential for computational reasons, only the first half of the random vector, i.e. y, does enter the linear predictor.

Parametrisation for the multivariate BYM A direct extension of the BYM to the case of p variables (p diseases) is the following.

First let us consider the ICAR component. With no loss of generality, define $\mathbf{U} = (U_1, U_2, \dots U_p)$ and $\text{vec}(\mathbf{U}) = (U_1^\top U_2^\top \dots U_p^\top)^\top$. We suppose all its component to have the same prior distribution, i.e.

$$U_j \sim N(0, L^+) \quad \forall j \in [1, p]$$

We further assume independence among them, such that $\text{vec}(\mathbf{U}) \sim N(0, I_p \otimes L^+)$. The same assumption is made on the IID component, i.e. $\text{vec}(\mathbf{V}) \sim N(0, I_p \otimes I_n)$.

By doing so, we can use a unique scale parameter, say Σ, as in the univariate case. We define M as a generic full-rank matrix such that $M^\top M = \Sigma$. Additionally, we define the precision parameter $\Lambda =: \Sigma^{-1}$.

Please notice that M does not have to be the Cholesky factor; in fact, a rather convenient yet not unique definition is $M = D^{\frac{1}{2}}E^\top$, where D is the diagonal matrix of the eigenvalues of Σ and E are the eigenvectors of Σ.

In the more general case of the M-model (Botella-Rocamora, Martinez-Beneito, and Banerjee 2015), consider the matrix-valued mixing parameters $\tilde{\Phi} := \text{diag}(\sqrt{\phi_1}, \sqrt{\phi_2}, \dots \sqrt{\phi_p})$ and $\tilde{\bar{\Phi}} := \text{diag}(\sqrt{1-\phi_1}, \sqrt{1-\phi_2}, \dots \sqrt{1-\phi_p}) = (I_p - \tilde{\Phi})^{-1/2}$.

The convolution model is generalised to:

$$\mathbf{Y} = \mathbf{U}M\tilde{\Phi} + \mathbf{V}M\tilde{\bar{\Phi}} \tag{2}$$

We then have

$$\mathbb{E}[\text{vec}(\mathbf{Y}) \mid U, \tilde{\Phi}, \Sigma] = \text{vec}(\mathbf{U}M\tilde{\Phi}) = [(\tilde{\Phi}M^\top) \otimes I_n]\text{vec}(\mathbf{U})$$

and similarly

$$\text{VAR}[\text{vec}(\mathbf{Y}) \mid \mathbf{U}, \tilde{\Phi}, \Sigma] = \left[(\tilde{\bar{\Phi}}M^\top) \otimes I_n\right] \mathbb{E}\left[\text{vec}(\mathbf{V})\text{vec}(\mathbf{V})^\top\right] \left[(\tilde{\Phi}M^\top) \otimes I_n\right] = \left(\tilde{\bar{\Phi}}\Sigma\tilde{\bar{\Phi}}\right) \otimes I_n$$

The distribution of $\mathbf{Y} \mid \mathbf{U}, \Sigma, \tilde{\Phi}$ then reads:

$$\begin{aligned}
&-2\ln\pi\left(\text{vec}(\mathbf{Y}) \mid \mathbf{U}, \Sigma, \phi\right) = \\
&= \left\{\text{vec}(\mathbf{Y}) - [(\tilde{\Phi}M^\top) \otimes I_n]\text{vec}(\mathbf{U})\right\}^\top \left[\left(\tilde{\bar{\Phi}}^{-1}\Lambda\tilde{\bar{\Phi}}^{-1}\right) \otimes I_n\right] \left\{\text{vec}(\mathbf{Y}) - [(\tilde{\Phi}M^\top) \otimes I_n]\text{vec}(\mathbf{U})\right\} = \\
&= \text{vec}(\mathbf{Y})^\top \left[\left(\tilde{\bar{\Phi}}^{-1}\Lambda\tilde{\bar{\Phi}}^{-1}\right) \otimes I_n\right] \text{vec}(\mathbf{Y}) + \\
&-2\text{vec}(\mathbf{Y})^\top \left[\left(\tilde{\bar{\Phi}}^{-1}\Lambda\tilde{\bar{\Phi}}^{-1}\right) \otimes I_n\right] \left[(\tilde{\Phi}M^\top) \otimes I_n\right] \text{vec}(\mathbf{U}) + \\
&+\text{vec}(\mathbf{U})^\top \left[(M\tilde{\Phi}) \otimes I_n\right] \left[\left(\tilde{\bar{\Phi}}^{-1}\Lambda\tilde{\bar{\Phi}}^{-1}\right) \otimes I_n\right] \left[(\tilde{\Phi}M^\top) \otimes I_n\right] \text{vec}(\mathbf{U}) = \\
&\text{vec}(\mathbf{Y})^\top \left[\left(\tilde{\bar{\Phi}}^{-1}\Lambda\tilde{\bar{\Phi}}^{-1}\right) \otimes I_n\right] \text{vec}(\mathbf{Y}) + \\
&-2\text{vec}(\mathbf{Y})^\top \left[\left(\tilde{\bar{\Phi}}^{-1}\Lambda\tilde{\bar{\Phi}}^{-1}\tilde{\Phi}M^\top\right) \otimes I_n\right] \text{vec}(\mathbf{U}) + \\
&+\text{vec}(\mathbf{U})^\top \left[\left(M\tilde{\Phi}\tilde{\bar{\Phi}}^{-1}\Lambda\tilde{\bar{\Phi}}^{-1}\tilde{\Phi}M^\top\right) \otimes I_n\right] \text{vec}(\mathbf{U})
\end{aligned}$$

Now, for brevity let us define the following $p \times p$ matrices:

$$q_{11} := \tilde{\bar{\Phi}}^{-1}\Lambda\tilde{\bar{\Phi}}^{-1}; \quad q_{12} := q_{11}\tilde{\Phi}M^\top; \quad q_{22} := M\tilde{\Phi}q_{12}$$

Hence

$$\begin{aligned}
&-2\ln\pi\left(\text{vec}(\mathbf{Y}) \mid \mathbf{U}, \Sigma, \phi\right) = \\
&= \text{vec}(\mathbf{Y})^\top \left(q_{11} \otimes I_n\right) \text{vec}(\mathbf{Y}) - 2\text{vec}(\mathbf{Y})^\top \left(q_{12} \otimes I_n\right) \text{vec}(\mathbf{U}) + \text{vec}(\mathbf{U})^\top \left(q_{12} \otimes I_n\right) \text{vec}(\mathbf{U})
\end{aligned}$$

Then, we have

$$\begin{aligned}
&-2\ln\pi\left(\text{vec}(\mathbf{Y}), \text{vec}(\mathbf{U}) \mid \Sigma, \phi\right) = \\
&= \text{vec}(\mathbf{Y})^\top \left(q_{11} \otimes I_n\right) \text{vec}(\mathbf{Y}) - 2\text{vec}(\mathbf{Y})^\top \left(q_{12} \otimes I_n\right) \text{vec}(\mathbf{U}) + \text{vec}(\mathbf{U})^\top \left(q_{12} \otimes I_n + I_p \otimes L\right) \text{vec}(\mathbf{U})
\end{aligned}$$

Hence, with some straightforward algebra, it can be concluded that:

$$\begin{pmatrix}\text{vec}(\mathbf{Y}) \\ \text{vec}(\mathbf{U})\end{pmatrix} \sim N\left(0, \begin{pmatrix} q_{11} \otimes I_n & -q_{12} \otimes I_n \\ -q_{12}^\top \otimes I_n & q_{22} \otimes I_n + I_p \otimes L \end{pmatrix}^{-1}\right) \tag{3}$$

Which generalises to the multivariate case equation 1. The sparse parametrisation is thus also possible for the multivariate BYM.

Application to SIDS data

Here we attempt to an application to the well-known SIDS dataset. The model has the following structure:

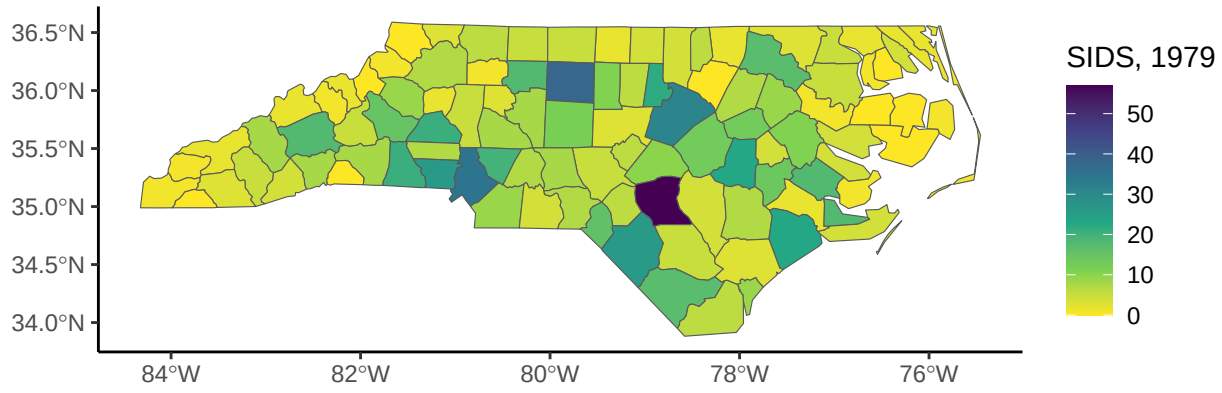
$$y_{i,t} \sim \text{Poisson}(e^{E_{i,t} + \eta_{i,t}})$$

Where $y_{i,t}$ are the sudden infant death cases for $i = 1 \dots 100$ and $t = 1974, 1979$; $E_{i,t}$ are the expected SIDS cases, i.e. the global fatality rate times the number of births, and $\eta_{i,t} := \beta_0 + \beta X_{i,t} + z_{i,t}$ is the linear predictor. Specifically, β_0 is the intercept, $X_{i,t}$ is the non-white birth proportion (NWBIR), β is the effect of NWBIR, and $z_{i,t}$ is a spatially structure latent effect.

```
library(spdep)
nc <- st_read(system.file("shapes/sids.gpkg", package="spData")[1], quiet=TRUE)

# neighbouring/adjacency matrix
W<- spdep::nb2mat(spdep::poly2nb(nc), style = "B")
# Laplacian matrix and nullspace of the ICAR field
Lapl <- diag(rowSums(W)) - W
A_constr <- kronecker(diag(1,2), t(pracma::nullspace(Lapl)))

ggplot2::ggplot() +
  ggplot2::geom_sf(data = nc,
                   ggplot2::aes(fill = .data$SID79))+
  ggplot2::labs(fill = "SIDS, 1979")+
  ggplot2::scale_fill_viridis_c(na.value = "white", direction = -1)+
  ggplot2::theme_classic()
```



Here, a long version of the dataframe is employed. Expected cases are computed as in (Palmí-Perales, Gómez-Rubio, and Martinez-Beneito 2021)

```
r74 <- sum(nc.sids$SID74) / sum(nc.sids$BIR74)
r79 <- sum(nc.sids$SID79) / sum(nc.sids$BIR79)

nc.long <- data.frame(NAME = c(nc$NAME, nc$NAME),
                      SID = c(nc$SID74, nc$SID79),
                      BIR = c(nc$BIR74, nc$BIR79),
                      NWBIR = c(nc$NWBIR74, nc$NWBIR79),
                      EXP = c(r74 * nc.sids$BIR74, r79 * nc.sids$BIR79),
                      ID = c(1:2*nrow(nc))) %>%
  dplyr::mutate(NWPROP = as.vector(scale(.data$NWBIR/.data$BIR)))
```

As a baseline model, also for comparisons, we can fit a PCAR model for Z, namely:

$$\text{vec}(Z) \sim N(0, \Lambda \otimes (D - \alpha W)^{-1})$$

Where Λ is the marginal precision parameter, D is the graph degree matrix, W is the neighbourhood matrix, and α is a time-invariant autoregressive parameter. This model can be conveniently fitted using the `INLAMSM` R package; a Wishart prior is assigned to the precision and a Uniform prior on $[0, 1]$ is assigned to α.

```
pcar.inla <- inla(
  SID ~ 1 + NWPROP +
    f(ID, model = INLAMSM::inla.MCAR.model(k=2, W = W, alpha.min = 0, alpha.max = 1)),
  data = nc.long, E = EXP,
  family = "poisson", num.threads = 1,
  control.compute =list(waic = T, internal.opt =F),
  verbose = T)
```

A more flexible yet more complex class of models is that of M-models, which allow to employ disease-specific hyperparameters. For the PCAR, we would thus have an autocorrelation parameter specific to each year. The prior distribution for Z becomes thus, for $t = 1974, 1979$ and $n = 100$:

$$\text{vec}(Z) \sim N\left(0, (M^\top \otimes I_n)\, \text{Bdiag}(D - \alpha_t W)\, (M \otimes I_n)\right)$$

Where BDiag denotes a block-diagonal matrix.

In the following, we use the `bigDM` function from the `bigDM` R package (Vicente et al. 2023) to fit the M-Model extension of the PCAR. This package has the advantage of automatically implementing models in which the Bartlett decomposition (see further) is used on the marginal scale parameter, which allows to specify a smaller number of parameters than direct parametrisation of M.

```
inla.PMMCAR.model <- function(...){
  INLA::inla.rgeneric.define(bigDM::Mmodel_pcar, ...)
}

pcar.inla.mmod <- inla(
  SID ~ 1 + NWPROP +
    f(ID, model = inla.PMMCAR.model(J=2, W = W, alpha.min = 0, alpha.max = 1,
                                    initial.values = c(0,0,0))),
  data = nc.long, E = EXP,
  family = "poisson", num.threads = 1,
  control.compute =list(waic = T, internal.opt =F),
  verbose = T)
```

Model results are quite similar; in both cases, computational time is relatively short (less than 10 seconds).

```
summary(pcar.inla)
```

```
## Time used:
##     Pre = 0.918, Running = 4.16, Post = 0.124, Total = 5.2
## Fixed effects:
##               mean    sd 0.025quant 0.5quant 0.975quant   mode   kld
## (Intercept) -0.032 0.494     -1.045   -0.032      0.980 -0.032 0.001
## NWPROP       0.229 0.032      0.167    0.229      0.291  0.229 0.000
##
## Random effects:
##   Name     Model
##     ID RGeneric2
##
## Model hyperparameters:
##                  mean    sd 0.025quant 0.5quant 0.975quant   mode
## Theta1 for ID -0.131 1.360      -2.83   -0.125       2.53 -0.099
## Theta2 for ID  0.199 1.080      -2.02    0.231       2.23  0.374
## Theta3 for ID  0.199 1.080      -2.02    0.231       2.23  0.374
## Theta4 for ID  0.124 0.855      -1.59    0.133       1.78  0.174
##
## Watanabe-Akaike information criterion (WAIC) ...: 959.05
## Effective number of parameters .................: 6.09
##
## Marginal log-Likelihood:  -488.84
##  is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```
summary(pcar.inla.mmod)
```

```
## Time used:
##     Pre = 0.618, Running = 6.04, Post = 0.328, Total = 6.98
## Fixed effects:
##               mean    sd 0.025quant 0.5quant 0.975quant   mode kld
## (Intercept) -0.031 0.673     -1.428   -0.031      1.368 -0.031   0
## NWPROP       0.229 0.032      0.167    0.229      0.291  0.229   0
##
## Random effects:
##   Name     Model
##     ID RGeneric2
##
## Model hyperparameters:
##                  mean    sd 0.025quant 0.5quant 0.975quant   mode
## Theta1 for ID -0.082 1.382     -2.812   -0.079       2.63 -0.065
## Theta2 for ID -0.060 1.388     -2.800   -0.058       2.66 -0.048
## Theta3 for ID  0.531 0.375     -0.249    0.545       1.22  0.611
## Theta4 for ID  0.363 0.429     -0.533    0.380       1.15  0.462
## Theta5 for ID  0.000 1.132     -2.228    0.000       2.23  0.001
##
## Watanabe-Akaike information criterion (WAIC) ...: 959.06
## Effective number of parameters .................: 6.10
##
## Marginal log-Likelihood:  -488.09
##  is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

The PCAR is flexible and computationally efficient due to its sparse precision. However, it is not scalable. The BYM, on the contrary, is scalable, but *as it is*, it does not have sparse precision, which implies computational inefficiency.

Simplified BYM

Here we provide a rather simple implementation of the BYM, loosely based on `bigDM` and `INLAMSM` codes:

```
inla.rgeneric.MMBYM.dense <-
  function (cmd = c("graph", "Q", "mu", "initial", "log.norm.const",
                    "log.prior", "quit"), theta = NULL) {
    envir <- parent.env(environment())
    #' Scaling the Laplacian matrix may be time-consuming,
    #' so it is better to do it just once.
    if(!exists("cache.done", envir=envir)){
    #' Unscaled Laplacian matrix (marginal precision of u_1, u_2 ... u_k)
      L_unscaled <- Matrix::Diagonal(nrow(W), rowSums(W)) -  W
      L_unscaled_block <- kronecker(diag(1,k), L_unscaled)
      A_constr <- t(pracma::nullspace(as.matrix(L_unscaled_block)))
      scaleQ <- INLA:::inla.scale.model.internal(
        L_unscaled_block, constr = list(A = A_constr, e = rep(0, nrow(A_constr))))
        #' Block Laplacian, i.e. precision of U = I_k \otimes L
      n <- nrow(W)
      L <- scaleQ$Q[c(1:n), c(1:n)]
      Sigma.u <- MASS::ginv(as.matrix(L))
      endtime.scale <- Sys.time()
      assign("Sigma.u", Sigma.u, envir = envir)
      assign("cache.done", TRUE, envir = envir)
    }
    interpret.theta <- function() {
      alpha <- 1/(1 + exp(-theta[as.integer(1:k)]))
      #' Bartlett decomposition ==> First define Sigma,
      #' then use its eigendecomposition to define M ==>
      #' ==> the function employs k(k+1)/2 parameters,
      #' i.e. lower-triangular factor in the Bartlett decomposition indeed.
      diag.N <- sapply(theta[as.integer(k + 1:k)], function(x) {
        exp(x)
      })
      no.diag.N <- theta[as.integer(2 * k + 1:(k * (k - 1)/2))]
      N <- diag(diag.N, k)
      N[lower.tri(N, diag = FALSE)] <- no.diag.N
      Sigma <- N %*% t(N)
      e <- eigen(Sigma)
      M <- t(e$vectors %*% diag(sqrt(e$values)))
      return(list(alpha = alpha, M = M))
    }
    graph <- function() {
      MI <- kronecker(Matrix::Matrix(1, ncol = k, nrow = k),
                      Matrix::Diagonal(nrow(W), 1))
      IW <- Matrix::Diagonal(nrow(W), 1) + W
      BlockIW <- Matrix::bdiag(replicate(k, IW, simplify = FALSE))
      G <- (MI %*% BlockIW) %*% MI
      return(G)
    }
    Q <- function() {
```

```
    param <- interpret.theta()
    M.inv <- solve(param$M)
    MI <- kronecker(M.inv, Matrix::Diagonal(nrow(W), 1))
    D <- as.vector(apply(W, 1, sum))
    BlockIW <- Matrix::bdiag(lapply(1:k, function(i) {
      solve(param$alpha[i]*Sigma.u +
              (1-param$alpha[i])*Matrix::Diagonal(nrow(W), 1))
    }))
    Q <- (MI %*% BlockIW) %*% kronecker(t(M.inv), Matrix::Diagonal(nrow(W),  1))
    return(Q)
  }
  mu <- function() {
    return(numeric(0))
  }
  log.norm.const <- function() {
    val <- numeric(0)
    return(val)
  }
  log.prior <- function() {
    param <- interpret.theta()
    val <- sum(-theta[as.integer(1:k)] - 2 * log(1 + exp(-theta[as.integer(1:k)])))
    #' Diagonal entries of the lower-triangular
    #' factor of Sigma: Chi-squared prior
    val <- val + k * log(2) + 2 * sum(theta[k + 1:k]) +
      sum(dchisq(exp(2 * theta[k + 1:k]),
                 df = (k + 2) - 1:k + 1, log = TRUE))
      #' Off-diagonal entries of the factor:
      #' Normal prior
      val <- val + sum(dnorm(theta[as.integer((2 * k) + 1:(k *  (k - 1)/2))],
                             mean = 0, sd = 1, log = TRUE))
    return(val)
  }
  initial <- function() {
    return(c(rep(0, k * (k+3)/2)) )
  }
  quit <- function() {
    return(invisible())
  }
  if (as.integer(R.version$major) > 3) {
    if (!length(theta))
      theta = initial()
  }
  else {
    if (is.null(theta)) {
      theta <- initial()
    }
  }
  val <- do.call(match.arg(cmd), args = list())
  return(val)
}

inla.MMBYM.dense <- function (...)  INLA::inla.rgeneric.define(inla.rgeneric.MMBYM.dense, ...)
```

Specifically, we use a Uniform prior on the mixing parameters and we model the M matrix through the

Bartlett decomposition, namely we define $\Sigma = AA^\top$, where A is a lower-triangular matrix whose squared diagonal elements are assigned a Chi-squared distribution and whose off-diagonal entries are assigned a Normal distribution. For more details on the Bartlett decomposition, see (Vicente et al. 2023)

```
bym.dense.inla.mmod <- inla(
    SID ~ 1 + NWPROP +
    f(ID, model = inla.MMBYM.dense(k=2, W=W),
      extraconstr = list(A = A_constr, e = c(0,0))),
  data = nc.long, E = EXP,
  family = "poisson", num.threads = 1,
  control.compute =list(waic = T, internal.opt =F),
  verbose = T)
```

```
summary(bym.dense.inla.mmod)
```

```
## Time used:
##     Pre = 1.1, Running = 10.9, Post = 0.095, Total = 12.1
## Fixed effects:
##               mean    sd 0.025quant 0.5quant 0.975quant   mode kld
## (Intercept) -0.028 1.227     -2.548   -0.029      2.505 -0.028   0
## NWPROP       0.229 0.032      0.167    0.229      0.291  0.229   0
##
## Random effects:
##   Name     Model
##     ID RGeneric2
##
## Model hyperparameters:
##                mean    sd 0.025quant 0.5quant 0.975quant  mode
## Theta1 for ID 0.021 1.422     -2.772    0.018       2.83 0.009
## Theta2 for ID 0.093 1.450     -2.731    0.082       2.98 0.037
## Theta3 for ID 0.531 0.374     -0.245    0.544       1.22 0.607
## Theta4 for ID 0.350 0.440     -0.564    0.367       1.16 0.442
## Theta5 for ID 0.270 1.010     -1.730    0.274       2.25 0.291
##
## Watanabe-Akaike information criterion (WAIC) ...: 959.07
## Effective number of parameters .................: 6.10
##
## Marginal log-Likelihood:  -488.55
##  is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

We compare the posteriors of α for the PCAR and BYM model. Starting from the PCAR

```
# \alpha_{1974}
inla.zmarginal(inla.tmarginal(fun = function(X) exp(X)/(1 + exp(X)),
               marginal = pcar.inla.mmod$marginals.hyperpar[[1]]))
```

```
## Mean            0.485252
## Stdev           0.257423
## Quantile  0.025 0.0577171
## Quantile  0.25  0.266514
## Quantile  0.5   0.480312
## Quantile  0.75  0.700483
## Quantile  0.975 0.931046
```

```
# \alpha_{1979}
inla.zmarginal(inla.tmarginal(fun = function(X) exp(X)/(1 + exp(X)),
               marginal = pcar.inla.mmod$marginals.hyperpar[[2]]))
```

```
## Mean            0.489134
## Stdev           0.258192
## Quantile  0.025 0.058376
## Quantile  0.25  0.269832
## Quantile  0.5   0.485394
## Quantile  0.75  0.705724
## Quantile  0.975 0.933306
```

Then for the BYM:

```
# \alpha_{1974}
inla.zmarginal(inla.tmarginal(fun = function(X) exp(X)/(1 + exp(X)),
               marginal = bym.dense.inla.mmod$marginals.hyperpar[[1]]))
```

```
## Mean            0.503559
## Stdev           0.262013
## Quantile  0.025 0.0599415
## Quantile  0.25  0.281032
## Quantile  0.5   0.504251
## Quantile  0.75  0.726358
## Quantile  0.975 0.942563
```

```
# \alpha_{1979}
inla.zmarginal(inla.tmarginal(fun = function(X) exp(X)/(1 + exp(X)),
               marginal = bym.dense.inla.mmod$marginals.hyperpar[[2]]))
```

```
## Mean            0.515696
## Stdev           0.264819
## Quantile  0.025 0.0622565
## Quantile  0.25  0.290955
## Quantile  0.5   0.519624
## Quantile  0.75  0.743267
## Quantile  0.975 0.950154
```

Results are similar.

Lastly here is the proposed code for the sparse BYM, still has to be tested:

```
inla.rgeneric.MMBYM.sparse <-
  function (cmd = c("graph", "Q", "mu", "initial", "log.norm.const",
                    "log.prior", "quit"), theta = NULL) {
    envir <- parent.env(environment())
    #' Scaling the Laplacian matrix may be time-consuming,
    #' so it is better to do it just once.
    if(!exists("cache.done", envir=envir)){
      #' Unscaled Laplacian matrix (marginal precision of u_1, u_2 ... u_k)
      L_unscaled <- Matrix::Diagonal(nrow(W), rowSums(W)) -  W
      L_unscaled_block <- kronecker(diag(1,k), L_unscaled)
      A_constr <- t(pracma::nullspace(as.matrix(L_unscaled_block)))
      scaleQ <- INLA:::inla.scale.model.internal(
        L_unscaled_block, constr = list(A = A_constr, e = rep(0, nrow(A_constr))))
      #' Block Laplacian, i.e. precision of U = I_k \otimes L
      n <- nrow(W)
```

```
  L <- scaleQ$Q[c(1:n), c(1:n)]
  assign("L", L, envir = envir)
  assign("cache.done", TRUE, envir = envir)
}
interpret.theta <- function() {
  phi.vector <- 1/(1 + exp(-theta[as.integer(1:k)]))
  #' Bartlett decomposition ==> First define Sigma,
  #' then use its eigendecomposition to define M ==>
  #' ==> the function employs k(k+1)/2 parameters,
  #' i.e. lower-triangular factor in the Bartlett decomposition indeed.
  diag.N <- sapply(theta[as.integer(k + 1:k)], function(x) {
    exp(x)
  })
  no.diag.N <- theta[as.integer(2 * k + 1:(k * (k - 1)/2))]
  N <- diag(diag.N, k)
  N[lower.tri(N, diag = FALSE)] <- no.diag.N
  Sigma <- N %*% t(N)
  e <- eigen(Sigma)
  M <- t(e$vectors %*% diag(sqrt(e$values)))
  return(list(phi.vector = phi.vector, M = M))
}
graph <- function() {
  MI <- kronecker(Matrix::Matrix(1, ncol = k, nrow = k),
                  Matrix::Diagonal(nrow(W), 1))
  IW <- Matrix::Diagonal(nrow(W), 1) + W
  BlockIW <- Matrix::bdiag(replicate(k, IW, simplify = FALSE))
  G <- (MI %*% BlockIW) %*% MI
  return(G)
}
Q <- function() {
  param <- interpret.theta()
  M.inv <- solve(param$M)
  Phi <- Matrix::Diagonal(sqrt(param$phi.vector), n=k)
  invPhihat <- Matrix::Diagonal(1/sqrt(1-param$phi.vector), n=k)
  q11 <- invPhihat %*% M.inv %*% t(M.inv) %*% invPhihat
  q12 <- q11 %*% Phi %*% t(param$M)
  q22 <- param$M %*% Phi %*% q12
  Q.11 <- kronecker(q11, Matrix::Diagonal(n=nrow(W), x = 1))
  Q.12 <- kronecker(-q12, Matrix::Diagonal(n=nrow(W), x = 1))
  Q.22 <- kronecker(q11, Matrix::Diagonal(n=nrow(W), x = 1)) +
    kronecker(Matrix::Diagonal(n=k, x=1), L)
  Q <- cbind(rbind(Q.11, Q.12),
             rbind(t(Q.12), Q.22))
  return(Q)
}
mu <- function() {
  return(numeric(0))
}
log.norm.const <- function() {
  val <- numeric(0)
  return(val)
}
log.prior <- function() {
```

```
      param <- interpret.theta()
      val <- sum(-theta[as.integer(1:k)] - 2 * log(1 + exp(-theta[as.integer(1:k)])))
      #' Diagonal entries of the lower-triangular
      #' factor of Sigma: Chi-squared prior
      val <- val + k * log(2) + 2 * sum(theta[k + 1:k]) +
        sum(dchisq(exp(2 * theta[k + 1:k]),
                   df = (k + 2) - 1:k + 1, log = TRUE))
      #' Off-diagonal entries of the factor:
      #' Normal prior
      val <- val + sum(dnorm(theta[as.integer((2 * k) + 1:(k *  (k - 1)/2))],
                             mean = 0, sd = 1, log = TRUE))
      return(val)
    }
    initial <- function() {
      return(c(rep(0, k * (k+3)/2)) )
    }
    quit <- function() {
      return(invisible())
    }
    if (as.integer(R.version$major) > 3) {
      if (!length(theta))
        theta = initial()
    }
    else {
      if (is.null(theta)) {
        theta <- initial()
      }
    }
    val <- do.call(match.arg(cmd), args = list())
    return(val)
  }

inla.MMBYM.sparse<- function(...){
  INLA::inla.rgeneric.define(inla.rgeneric.MMBYM.sparse, ...)}
```


Botella-Rocamora, Paloma, Miguel A Martinez-Beneito, and Sudipto Banerjee. 2015. "A Unifying Modeling Framework for Highly Multivariate Disease Mapping." *Statistics in Medicine* 34 (9): 1548–59.

Palmí-Perales, Francisco, Virgilio Gómez-Rubio, and Miguel A. Martinez-Beneito. 2021. "Bayesian Multivariate Spatial Models for Lattice Data with INLA." *Journal of Statistical Software* 98 (2): 1–29. https://doi.org/10.18637/jss.v098.i02.

Riebler, Andrea, Sigrunn H Sørbye, Daniel Simpson, and Håvard Rue. 2016. "An Intuitive Bayesian Spatial Model for Disease Mapping That Accounts for Scaling." *Statistical Methods in Medical Research* 25 (4): 1145–65.

Vicente, Gonzalo, Aritz Adin, Tomás Goicoa, and María Dolores Ugarte. 2023. "High-Dimensional Order-Free Multivariate Spatial Disease Mapping." *Statistics and Computing* 33 (5): 104.