Supplementary Material for "Graph-structured variable selection with Gaussian Markov random field horseshoe prior"

Marie Denis[1] and Mahlet G. Tadesse[2]

[1] *CIRAD, UMR AGAP Institut, Montpellier, France*
[2] *Department of Mathematics and Statistics, Georgetown University, Washington,DC,USA*

This supplement provides the R code for fitting the HS-GMRF model along with a detailed simulation example. These can also be found at https://github.com/DenisMarie/HS-GMRF/blob/main/README.md.

1. **R code for HS-GMRF**

```
# Arguments to input:
# - Y = vector of observations (n x 1)
# - X = matrix of predictors (n x p)
# - C = contrast matrix of dimension q x p
# - niter = number of iterations for the MCMC algorithm
# - a, b = hyperparameters of the inverse Gamma prior of the residual variance
# Output of function:
# a list containing the niter samples of beta, tau, lambda, se2, mu
HS_GMRF <- function(Y = Y, X = X, C = C,
                        a = 1, b = 1, niter = 1000){
  p <- ncol(C)
  q <- nrow(C)
  n <- length(Y)
  XprimeX <- crossprod(X)
  # to store samples
  mu_store <- rep(0, niter)
  tau_store <- matrix(0, ncol = q, nrow = niter)
  nu_store <- matrix(0, ncol = q, nrow = niter)
  beta_store <- matrix(0, ncol = p, nrow = niter)
  lambda_store <- rep(0,niter)
  se2_store <- rep(0,niter)
  omega_store<- rep(0,niter)
  # intial values
  lambda_store[1] <- 1
  se2_store[1] <- 1
  beta_store[1,] <- rep(1,p)
  nu_store[1,] <- rep(1,q)
  tau_store[1,] <- rep(1,q)
  omega_store[1] <- 1
  mu_store[1] <- mean(Y)
  phi <- rep(1,q)
  for (i in 2:niter){
    # update of tau and nu
    for (j in 1:q){
      tau_store[i,j] <- max(1E-5,rinvgamma(1,
                                           shape = 1,
                                           rate = phi[j]^2/(2*lambda_store[i-1])+1/nu_store[i-1,j]))
      nu_store[i,j] <- rinvgamma(1,
                                 shape = 1,
                                 rate = 1+1/tau_store[i,j])

    }
    Q_tmp <- crossprod(C,diag(1/tau_store[i,]))%*%C
    # update of beta
    Sigma_tmp <- solve(XprimeX/se2_store[i-1]+
                         Q_tmp/lambda_store[i-1])
    beta_store[i,] <- rmvn(1,
                           mu = (1/se2_store[i-1])*Sigma_tmp%*%crossprod(X,Y-mu_store[i-1]),
                           sigma = Sigma_tmp)

    phi <- C%*%c(beta_store[i,])
    # update of mu
```

```
    mu_store[i] <- rnorm(1, (t(rep(1,n))/n)%*%(Y-X%*%beta_store[i,]), sd = sqrt(se2_store[i-1]/n))
    # update of lambda and omega
    lambda_store[i] <- max(1E-5,rinvgamma(1,
                                          shape =(p+1)/2,
                                          rate = 1/omega_store[i-1]+
                                          crossprod(beta_store[i,],Q_tmp)%*%beta_store[i,]/2))
    omega_store[i] <- rinvgamma(1,
                                shape =1,
                                rate = 1/se2_store[i-1]+1/lambda_store[i])
    #update of se2
    se2_store[i] <- rinvgamma(1,
                              shape = (n+1)/2+a,
                              rate = 0.5*crossprod(Y-X%*%beta_store[i,]-mu_store[i]) +
                                1/omega_store[i]+b )
  }
  return(list(beta_store = beta_store,lambda_store = lambda_store,
              tau_store = tau_store, se2_store =se2_store , mu_store=mu_store))
}
```

2. **R code for constructing the contrast and adjacency matrices and for computing the Matthews correlation coefficient and mean squared error**

```
# R code to create the graph/adjacency matrix
# Arguments to input:
# - k: the size of groups
# - nb.gp: the number of groups
# - rho.seq: vector of dimension nb.gp containing the correlation associated to each group

# Output of function:
# - G: matrix of dimension p x p representing the graph

adjacency <- function( k, nb.gp, rho.seq){
  adj <- matrix(0, ncol = k, nrow = k)
  adj[1,] <- 1; adj[,1] <- 1
  diag(adj) <- 1
  G <- kronecker(rho.seq*diag(nb.gp),adj)
  G[G!=0] <- 1
  diag(G) <- 1
  return(G)
}
```

```
# R code to create the contrast matrix
# Arguments to input:
# - G = matix representing the graph
# - signX = matrix containing the sample correlation between covariates

# Output of function:
# - C = contrast matrix of dimension q x p

CQ_const <- function(G = G, signX = FALSE){
  p <- nrow(G)
  q <- sum(G[upper.tri(G)]!=0) # number of connected parameters
  ind <- apply(G, 1, function(x){which(x!=0)})
  C <- NULL
  for(i in 1:p)
  {
    ind_tmp <- ind[[i]]
    if (sum(ind_tmp!=i & ind_tmp>i) != 0){
      ind_tmp <- ind_tmp[ind_tmp!=i & ind_tmp>i]
      tmp <- matrix(0, ncol=p, nrow=length(ind_tmp))
      tmp[, i] <- 1
      for (j in 1:length(ind_tmp))
      {
        tmp[j, ind_tmp[j]] <- -1*signX[i, ind_tmp[j]]
      }
      C <- rbind(C, tmp)
    }else{
      if (length(ind_tmp) == 1){
        tmp <- rep(0,p)
        tmp[i] <- 1
        C <- rbind(C, tmp)
      }
    }
  }
```

```
  sub.graph <- clusters(graph.adjacency(G))$membership
  ind.gp <- as.numeric(names(table(sub.graph))[table(sub.graph)>1])
  ind.plus <-  match( ind.gp,sub.graph)
  zero <- matrix(0, ncol = p, nrow = length(ind.plus))
  if (length(ind.plus) == 0) zero <- c(1, rep(p-1)) else{
    for (i in 1:length(ind.plus)) zero[i,ind.plus[i]] <- 1
  }
  C <- rbind(C, zero)
  return(C)
}
```

```
# R code to compute Matthews correlation coefficient
# Arguments to input:
# - beta = vector of true coefficients
# - ind_sel = indices of the selected variables

# Output of function:
# - MCC: Matthews correlation coefficient

output <- function(beta, ind_sel){
  aux <- rep(0,length(beta))
  aux[ind_sel] <- 1
  TP <- sum(aux[beta != 0] > 0.5)
  FN <- sum(aux[beta != 0] < 0.5)
  FP <- sum(aux[beta == 0] > 0.5)
  TN <- sum(aux[beta == 0] < 0.5)
  MCC <- (TP*TN -FP*FN) / sqrt((TP+FP)* (TP+FN)*(TN +FP)*(TN +FN))
  return(MCC)
}
```

```
# R code to compute the mean squared error of the regression coefficients
# Arguments to input:
# - beta_true = vector of true coefficients
# - beta_hat = vector of estimated coefficients

# Output of function:
# - the mean squared error of the regression coefficients
MSE <- function(beta_true, beta_hat){
   return(mean((beta_true-beta_hat)^2))
}
```

3. **R script replicating one of the simulated examples presented in the paper**

Data are simulated according to scenario with the following settings: $\Sigma_g = \Sigma_{g,half}$, $\rho = 0.9$, and $\beta_g = (5, 5/\sqrt(10), ..., 5/\sqrt(10), 0, \ldots, 0)$.

```
library(mvtnorm)
library(coda)
library(invgamma)
library(bayesreg)
library(Matrix)
library(bayesreg)
library(igraph)
library(mvnfast)

source("UtilFunctions.R")
source("MainFunction.R")

n <- 100 # the number of observations
p <- 140 # the number of predictors
rho <- 0.9 # the correlation
k <- 10 # the size of groups
nb.gp <- p/k # the number of groups
rho.seq <- c(rep(rho, nb.gp/2), rep(0, nb.gp/2)) # vector containing
the correlation associated to each group (here the same rho)

# the associated graph
G <- adjacency(k = k, nb.gp = nb.gp, rho.seq = rho.seq)

# the associated contrast matrix
C <- CQ_const(G = G, signX = matrix(1, ncol = p, nrow = p))

# the matrix of predictors: matrix of dimension n x p
X <- matrix(0, nrow = n, ncol = p)
for (g in 0:(nb.gp-1)){
  X[,g*k+1] <- rnorm(n, 0, 1)
  for (i in 2:k)
    X[,g*k+(i)] <- rnorm(n, mean = rho.seq[g+1]*X[,g*k+(1)],
                         sd = sqrt(1-rho.seq[g+1]^2))
}

# the true coefficients
beta_tmp <- rep(c(5, rep(5/sqrt(10), k-1)), times = nb.gp)
beta_true <- rep(0,p)
ind_gp <- c(1,3,5,8,10)
ind_sel <- rep(1:nb.gp, each = k)%in%ind_gp
beta_true[ind_sel] <- beta_tmp[ind_sel]

# the residual variance
se2 <- sum(beta_true[ind_sel]^2)/length(ind_gp)

# the response variable: vector of dimension n
Y <- as.vector(X %*% beta_true + rnorm(n, 0, sqrt(se2)))


# to run HS-GMRF model on a simulated datasete
```

```
niter <- 1000
nburn <- 500
res.gmrf <- HS_GMRF(Y = Y, X = X, C = C, a = 1, b = 1,
                    niter = niter)

# to analyze the results
beta_hat <- colMeans(res.gmrf$beta_store[-(1:nburn),])
ci_gmrf <- t(apply(res.gmrf$beta_store[-(1:nburn),], 2,
                   function(c) HPDinterval(as.mcmc(c), probs = 0.95)))
sel_gmrf <- which(apply(sign(ci_gmrf), 1, prod) == 1)
output(beta_true, sel_gmrf)
MSE(beta_true, beta_hat)
```