# Generic1 model

### Parametrization

The Type 1 generic model implements the following precision matrix

$$\mathbf{Q} = \tau (\mathbf{I} - \frac{\beta}{\lambda_{max}} \mathbf{C})$$

where **C** is the structure matrix. The parameter  $\lambda_{max}$  is the maximum eigenvalue of **C**, which allows  $\beta$  to be in the range  $\beta \in [0, 1)$ 

# Hyperparameters

The two parameters of the generic model are represented as

$$\theta_1 = \log(\tau)$$
 $\theta_2 = \log it(\beta)$ 

and priors are assigned to  $(\theta_1, \theta_2)$ 

## **Specification**

The generic1 model is specified inside the f() function as

```
f(<whatever>,model="generic1",Cmatrix = <Cmat>, hyper = <hyper>)
```

where <Cmat> can be given in two different ways:

- a dense matrix or a sparse-matrix defined be Matrix::sparseMatrix().
- the name of a file giving the structure matrix. The file should have the following format

$$i \quad j \quad \mathbf{C}_{ij}$$

where i and j are the row and column index and  $C_{ij}$  is the corresponding element of the precision matrix. Only the non-zero elements of the precision matrix need to be stored in the file.

### Hyperparameter spesification and default values

```
doc A generic model (type 1)
```

hyper

#### theta1

hyperid 19001
name log precision
short.name prec
prior loggamma
param 1 5e-05
initial 4
fixed FALSE
to.theta function(x) log(x)
from.theta function(x) exp(x)

```
theta2
         hyperid 19002
         name beta
         short.name beta
         initial 2
         fixed FALSE
         prior gaussian
         param 0 0.1
         to.theta function(x) log(x / (1 - x))
         from.theta function(x) exp(x) / (1 + exp(x))
constr FALSE
nrow.ncol FALSE
augmented FALSE
aug.factor 1
aug.constr
n.div.by
n.required TRUE
set.default.values TRUE
\operatorname{pdf} generic1
Example
n = 100
## build a structure matrix
Cm = matrix(runif(n^2,min=-1,max=1),n,n)
diag(Cm) = 0
Cm = 0.5*(Cm + t(Cm))
lambda.max = max(eigen(Cm)$values)
## define the precision matrix
beta = 0.9
Q = diag(rep(1,n)) - beta/lambda.max * Cm
Sigma = solve(Q)
#simulate data
require(mvtnorm)
sd = 0.001
z = rnorm(n)
eta = rmvnorm(n=1,sigma = Sigma)
y = c(eta) + sd*rnorm(n) + z
idx = 1:n
d = list(y=y,idx=idx,z=z)
## Alternative 1
```

```
## print the file containing the C matrix
file = "Cmatrix.dat"
cat("",file=file, append = FALSE)
for(i in 1:n)
   cat(i, j, Cm[i,j], "\n", sep = " ", file=file, append=TRUE)
   if (i < n)
       for(j in (i+1):n)
          cat(i, j, Cm[i,j], "\n", sep = " ", file=file, append=TRUE)
formula = y ~ f(idx, model = "generic1", Cmatrix = file) + z
## Alternative 2
## formula = y ~ f(idx, model = "generic1", Cmatrix = Cm) + z
## Alternative 3
## formula = y ~ f(idx, model = "generic1", Cmatrix = as(Cm, "dgTMatrix"))+z
result = inla(formula, data=d,family="gaussian",
            control.family = list(initial = log(1/sd^2), fixed=TRUE),
            verbose=T, keep=T)
```

## Notes

None