

# The dMatern model

## Parametrisation

This model is the Gaussian field with a Matérn correlation function, directly, meaning **dense matrices**. This model is intended for a low-dimension only. The correlation function is

$$\text{Corr}(d) = \frac{1}{2^{\nu-1}\Gamma(\nu)} (\kappa d)^\nu K_\nu(\kappa d), \quad \alpha = \nu + d/2,$$

where  $K_\nu$  is the modified Bessel function and  $\Gamma(\cdot)$  is the Gamma-function. The range is *defined* to be

$$r = \sqrt{8\nu}/\kappa$$

which about the distance where the covariance function becomes about 0.1.

## Hyperparameters

The hyperparameters are the precision parameter  $\tau$ , the range  $r$  and the smoothness  $\nu$ , where the internal representation are

$$\theta = (\log(\tau), \log(r), \log(\nu))$$

The latent field has marginal variance  $1/\tau$  and range (as defined above)  $r$ .

We do **not recommend** to treat  $\nu$  as random, and for this reason it is default fixed. You can change its value by changing the initial value.

## Specification

The `dmatern` model is specified inside the `f()` function as:

```
f(idx, model="dmatern", locations = L, hyper = <hyper>)
```

where  $L$  is a matrix of the locations for which the Gaussian field is defined; row  $L[i,]$  are the coordinates for the  $i$ 'th location. `idx` represent the location indexing the corresponding row in  $L$ , so `idx = 3` means location  $L[idx,]$ . `idx` must be integers 1, 2, ..., `nrow(L)`, or `NA`.

## Hyperparameter specification and default values

**doc** Dense Matern field

**hyper**

**theta1**

**hyperid** 35101

**name** log precision

**short.name** prec

**initial** 3

**fixed** FALSE

**prior** pc.prec

**param** 1 0.01

**to.theta** function(x) log(x)

**from.theta** function(x) exp(x)

**theta2**

**hyperid** 35102

```

    name log range
    short.name range
    initial 0
    fixed FALSE
    prior pc.range
    param 1 0.5
    to.theta function(x) log(x)
    from.theta function(x) exp(x)
  theta3
    hyperid 35103
    name log nu
    short.name nu
    initial -0.693147180559945
    fixed TRUE
    prior loggamma
    param 0.5 1
    to.theta function(x) log(x)
    from.theta function(x) exp(x)

  constr FALSE

  nrow.ncol FALSE

  augmented FALSE

  aug.factor 1

  aug.constr

  n.div.by

  n.required TRUE

  set.default.values TRUE

  status experimental

  pdf dmatern

```

## Example

```

library(INLA)
library(mvtnorm)

# 1D example. locations are 1, 2, ..., 'n', with 'nr' replications
range = 10
n = 50
nr = 20
loc = 1:n
var = 1.0
nu = 0.5

S = matrix(0, n, n)
for(i in 1:n) {

```

```

    for(j in i:n) {
      d = sqrt((loc[i] - loc[j])^2)
      S[i, j] = var * INLA:::inla.matern.cf(d, range = range, nu = nu);
      S[j, i] = S[i, j]
    }
  }
y = c(t(rmvnorm(nr, sigma = S)))
r1 = inla(y ~ -1 + f(idx, model="dmatern", locations = loc, replicate = re,
  ## placing the prior at the correct value, just for
  ## demonstration
  hyper = list(range = list(initial = log(range),
    param = c(range, 0.5)))),
  data = data.frame(y, idx = rep(1:n, nr), re = rep(1:nr, each = n)),
  family = "gaussian",
  ## just this this at some high value
  control.family = list(hyper = list(
    prec = list(initial = 12, fixed=TRUE))))

if (FALSE)
  plot(r1, plot.random.effect = FALSE)

# 2D example. Simulate 'n' data in a [0, 1]^2 box, with 'nr' replications
range = 0.2
n = 50
nr = 20
loc = matrix(runif(2*n), ncol = 2, nrow = n)
var = 1.0
nu = 0.5

S = matrix(0, n, n)
for(i in 1:n) {
  for(j in i:n) {
    dif = loc[i, ] - loc[j, ]
    d = sqrt(sum(dif^2))
    S[i, j] = var * INLA:::inla.matern.cf(d, range = range, nu = nu);
    S[j, i] = S[i, j]
  }
}
y = c(t(rmvnorm(nr, sigma = S)))
r2 = inla(y ~ -1 + f(idx, model="dmatern", locations = loc, replicate = re,
  ## placing the prior at the correct value, just for
  ## demonstration
  hyper = list(range = list(initial = log(range),
    param = c(range, 0.5)))),
  data = data.frame(y, idx = rep(1:n, nr), re = rep(1:nr, each = n)),
  family = "gaussian",
  ## just this this at some high value
  control.family = list(hyper = list(
    prec = list(initial = 12, fixed=TRUE))))

if (FALSE)
  plot(r2, plot.random.effect = FALSE)

```

## Notes

Note that the above definition of range, might differ from the definition in other packages. It is the same used for the SPDE-models.