# R documentation

of 'inla.spde2.pcmatern.Rd'

May 4, 2021

---

`inla.spde2.pcmatern` *Matern SPDE model object with PC prior for INLA*

---

## Description

Create an `inla.spde2` model object for a Matern model, using a PC prior for the parameters.

## Usage

```
inla.spde2.pcmatern(
  mesh,
  alpha = 2,
  param = NULL,
  constr = FALSE,
  extraconstr.int = NULL,
  extraconstr = NULL,
  fractional.method = c("parsimonious", "null"),
  n.iid.group = 1,
  prior.range = NULL,
  prior.sigma = NULL
)
```

## Arguments

| | |
|---|---|
| `mesh` | The mesh to build the model on, as an `inla.mesh()` or `inla.mesh.1d()` object. |
| `alpha` | Fractional operator order, $0 < \alpha \leq 2$ supported, for $\nu = \alpha - d/2 > 0$. |
| `param` | Further model parameters. Not currently used. |
| `constr` | If `TRUE`, apply an integrate-to-zero constraint. Default `FALSE`. |
| `extraconstr.int` | Field integral constraints. |
| `extraconstr` | Direct linear combination constraints on the basis weights. |
| `fractional.method` | Specifies the approximation method to use for fractional (non-integer) `alpha` values. `'parsimonious'` gives an overall approximate minimal covariance error, `'null'` uses approximates low-order properties. |

`n.iid.group` If greater than 1, build an explicitly iid replicated model, to support constraints applied to the combined replicates, for example in a time-replicated spatial model. Constraints can either be specified for a single mesh, in which case it's applied to the average of the replicates (`ncol(A)` should be `mesh$n` for 2D meshes, `mesh$m` for 1D), or as general constraints on the collection of replicates (`ncol(A)` should be `mesh$n * n.iid.group` for 2D meshes, `mesh$m * n.iid.group` for 1D).

`prior.range` A length 2 vector, with (range0,Prange) specifying that $P(\rho < \rho_0) = p_\rho$, where $\rho$ is the spatial range of the random field. If `Prange` is `NA`, then `range0` is used as a fixed range value.

`prior.sigma` A length 2 vector, with (sigma0,Psigma) specifying that $P(\sigma > \sigma_0) = p_\sigma$, where $\sigma$ is the marginal standard deviation of the field. If `Psigma` is `NA`, then `sigma0` is used as a fixed range value.

## Details

This method constructs a Matern SPDE model, with spatial range $\rho$ and standard deviation parameter $\sigma$. In the parameterisation

$$(\kappa^2 - \Delta)^{\alpha/2}(\tau$$

$$x(u)) = W(u)$$

the spatial scale parameter $\kappa = \sqrt{8\nu}/\rho$, where $\nu = \alpha - d/2$, and $\tau$ is proportional to $1/\sigma$.

Stationary models are supported for $0 < \alpha \leq 2$, with spectral approximation methods used for non-integer $\alpha$, with approximation method determined by `fractional.method`.

Integration and other general linear constraints are supported via the `constr`, `extraconstr.int`, and `extraconstr` parameters, which also interact with `n.iid.group`.

The joint PC prior density for the spatial range, $\rho$, and the marginal standard deviation, $\sigma$, and is

$$\pi(\rho, \sigma) =$$

$$\frac{d\lambda_\rho}{2}\rho^{-1-d/2}\exp(-\lambda_\rho\rho^{-d/2})$$

$$\lambda_\sigma \exp(-\lambda_\sigma\sigma)$$

where $\lambda_\rho$ and $\lambda_\sigma$ are hyperparameters that must be determined by the analyst. The practical approach for this in INLA is to require the user to indirectly specify these hyperparameters through

$$P(\rho < \rho_0) = p_\rho$$

and

$$P(\sigma > \sigma_0) = p_\sigma$$

where the user specifies the lower tail quantile and probability for the range ($\rho_0$ and $p_\rho$) and the upper tail quantile and probability for the standard deviation ($\sigma_0$ and $\alpha_\sigma$).

This allows the user to control the priors of the parameters by supplying knowledge of the scale of the problem. What is a reasonable upper magnitude for the spatial effect and what is a reasonable lower scale at which the spatial effect can operate? The shape of the prior was derived through a construction that shrinks the spatial effect towards a base model of no spatial effect in the sense of distance measured by Kullback-Leibler divergence.

The prior is constructed in two steps, under the idea that having a spatial field is an extension of not having a spatial field. First, a spatially constant random effect ($\rho = \infty$) with finite variance is more complex than not having a random effect ($\sigma = 0$). Second, a spatial field with spatial variation ($\rho < \infty$) is more complex than the random effect with no spatial variation. Each of these extensions are shrunk towards the simpler model and, as a result, we shrink the spatial field towards the base model of no spatial variation and zero variance ($\rho = \infty$ and $\sigma = 0$).

The details behind the construction of the prior is presented in Fuglstad, et al. (2016) and is based on the PC prior framework (Simpson, et al., 2015).

## Value

An `inla.spde2` object.

## Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

## References

Fuglstad, G.-A., Simpson, D., Lindgren, F., and Rue, H. (2016) Constructing Priors that Penalize the Complexity of Gaussian Random Fields. arXiv:1503.00256

Simpson, D., Rue, H., Martins, T., Riebler, A., and Sørbye, S. (2015) Penalising model component complexity: A principled, practical approach to constructing priors. arXiv:1403.4630

## See Also

`inla.mesh.2d()`, `inla.mesh.create()`, `inla.mesh.1d()`, `inla.mesh.basis()`, `inla.spde2.matern()`, `inla.spde2.generic()`

## Examples

```
## Spatial interpolation
  n = 100
  field.fcn = function(loc) (10*cos(2*pi*2*(loc[,1]+loc[,2])))
  loc = matrix(runif(n*2),n,2)
  ## One field, 2 observations per location
  idx.y = rep(1:n,2)
  y = field.fcn(loc[idx.y,]) + rnorm(length(idx.y))

  mesh = inla.mesh.2d(loc, max.edge=0.05, cutoff=0.01)
  spde = inla.spde2.pcmatern(mesh,
           prior.range=c(0.01,0.1), prior.sigma=c(100,0.1))
  data = list(y=y, field=mesh$idx$loc[idx.y])
  formula = y ~ -1 + f(field, model=spde)
  result = inla(formula, data=data, family="normal")

  ## Plot the mesh structure:
  plot(mesh)

  if (require(rgl)) {
    col.pal = colorRampPalette(c("blue","cyan","green","yellow","red"))
    ## Plot the posterior mean:
    plot(mesh, rgl=TRUE,
         result$summary.random$field[,"mean"],
         color.palette = col.pal)
```

```
  ## Plot residual field:
  plot(mesh, rgl=TRUE,
       result$summary.random$field[,"mean"]-field.fcn(mesh$loc),
       color.palette = col.pal)
}


result.field = inla.spde.result(result, "field", spde)
par(mfrow=c(2,1))
plot(result.field$marginals.range.nominal[[1]],
     type="l", main="Posterior density for range")
plot(inla.tmarginal(sqrt, result.field$marginals.variance.nominal[[1]]),
     type="l", main="Posterior density for std.dev.")
par(mfrow=c(1,1))

## Spatial model
set.seed(1234234)

## Generate spatial locations
nObs = 200
loc = matrix(runif(nObs*2), nrow = nObs, ncol = 2)

## Generate observation of spatial field
nu = 1.0
rhoT = 0.2
kappaT = sqrt(8*nu)/rhoT
sigT = 1.0
Sig = sigT^2*inla.matern.cov(nu = nu,
                             kappa = kappaT,
                             x = as.matrix(dist(loc)),
                             d = 2,
                             corr = TRUE)
L = t(chol(Sig))
u = L %*% rnorm(nObs)

## Construct observation with nugget
sigN = 0.1
y = u + sigN*rnorm(nObs)

## Create the mesh and spde object
mesh = inla.mesh.2d(loc,
                    max.edge = 0.05,
                    cutoff = 0.01)
spde = inla.spde2.pcmatern(mesh,
                           prior.range = c(0.01, 0.05),
                           prior.sigma = c(10, 0.05))

## Create projection matrix for observations
A = inla.spde.make.A(mesh = mesh,
                     loc = loc)

## Run model without any covariates
idx = 1:spde$n.spde
res = inla(y ~ f(idx, model = spde) - 1,
           data = list(y = y, idx = idx, spde = spde),
           control.predictor = list(A = A))
```

```
## Re-run model with fixed range
  spde.fixed = inla.spde2.pcmatern(mesh,
                                   prior.range = c(0.2, NA),
                                   prior.sigma = c(10, 0.05))

  res.fixed = inla(y ~ f(idx, model = spde) - 1,
                   data = list(y = y, idx = idx, spde = spde.fixed),
                   control.predictor = list(A = A))
```

# Index