

# A short introduction on how to fit a **SPDE** model with **INLA**

Elias T. Krainski and Håvard Rue

Created: November, 20 2013 - Last update: August, 9 2017

This document illustrates how to do a geostatistical fully Bayesian analysis through the **Stochastic Partial Differential Equation** approach, [Lindgren et al., 2011], with **Integrated Nested Laplace Approximation**, [Rue et al., 2009], using the **INLA** package, <http://www.r-inla.org>.

## 1 Data simulation

Set  $n$  **locations** and a exponential **covariance** matrix to define the **R**andom **F**ield (RF)

```
n <- 200; coo <- matrix(runif(2*n), n)
s2u <- .5; k <- 10; r <- 2/k ## RF params.
R <- s2u*exp(-k*as.matrix(stats::dist(coo)))
```

**Sample:** one multivariate Normal realization

```
u <- drop(rnorm(n)%*%chol(R))
```

Add a **covariate** effect and a noise (nugget)

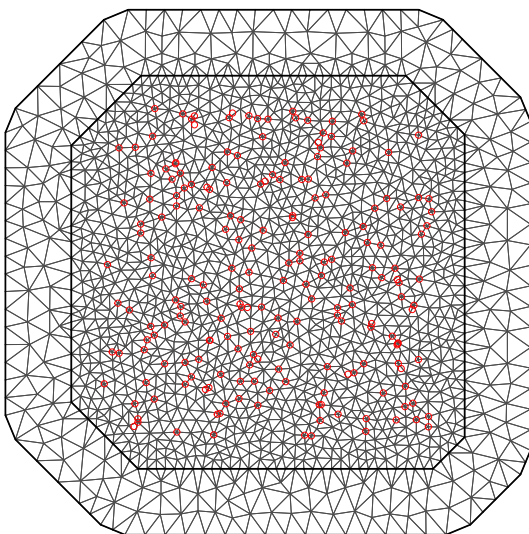
```
x <- runif(n); beta <- 1:2; s2e <- 0.2
lin.pred <- beta[1] + beta[2]*x + u
y <- lin.pred + rnorm(n, 0, sqrt(s2e))
```

## 2 Model fitting

1. **Mesh:** Triangulation around the locations

```
mesh <- inla.mesh.2d(coo, cutoff=r/10,
  max.edge=c(r/4, r/2), offset=c(r/2, r))
plot(mesh, asp=1); points(coo, col='red')
```

Constrained refined Delaunay triangulation



The outer domain additional triangles is to avoid boundary effects. Is good to have approximately isosceles triangles and to avoid tiny triangles, see `cutoff`. We need to have edge length for

the inner mesh triangles less than the range of the process, if there is a spatial effect.

2. The linear predictor at location  $s$ :

$$\eta(s) = \beta_0 + \beta_1 x(s) + A(s, s_0)u(s_0),$$

where  $u(s_0)$  is the RF at the mesh nodes, [Lindgren and Rue, 2015], and  $A(s, s_0)$  is a  $n \times m$  **projection matrix** to project the process from the mesh nodes to the observation locations:

```
A <- inla.spde.make.A(mesh=mesh, loc=coo)
```

3. **Build the SPDE model** on the mesh, where  $\alpha = 3/2$  for the exponential correlation and set the **RF parameters** prior distributions, [Fuglstad et al., 2017]:

```
spde <- inla.spde2.pcmatern(
  mesh=mesh, alpha=1.5,
  prior.range=c(0.2, 0.5), #P(range<0.2)=0.5
  prior.sigma=c(1, 0.5)) ## P(sigma>1)=0.5
```

4. Use `inla.stack()` to deal with effects having different projection matrices:

```
stk.e <- inla.stack(tag='est', ## tag id
  data=list(y=y), ## response
  A=list(1, A), ## two projection matrices
  effects=list(## two elements:
    data.frame(b0=1, x=x), ## covariate
    idx.u=1:spde$n.spde)) ## RF index
```

5. Set the **nugget prior** distribution, the linear predictor **formula** and **fit** the posterior marginal distributions (PMD)s for all the model parameters using `inla()`

```
pcprec <- list(hyper=list(theta=list(
  prior='pc.prec', param=c(1, .1)))
mf <- y ~ 0 + b0 + x + f(idx.u, model=spde)
res <- inla(mf, control.family=pcprec,
  data=inla.stack.data(stk.e), ## data
  control.compute=list(return.marginals.predictor=TRUE,
  control.predictor=list(compute=TRUE,
  A=inla.stack.A(stk.e)))# full projector
```

### 3 Posterior marginal distributions - PMDs

The `summary.hyperpar` element of `res` has a summary from each regression coefficient PMD:

```
round(res$summary.fixed, 4)

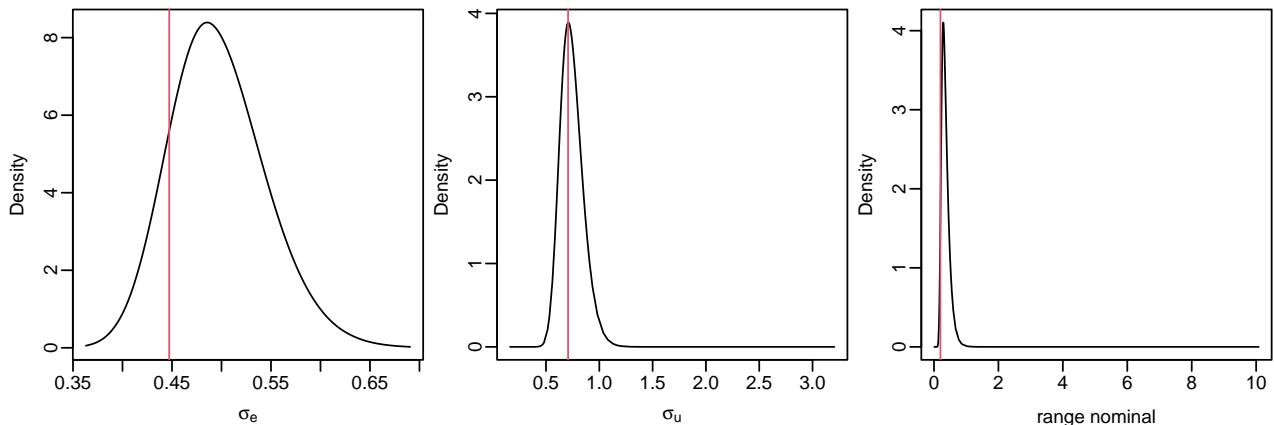
##      mean      sd 0.025quant 0.5quant 0.975quant   mode kld
## b0 0.9917 0.2813    0.3980    0.9984    1.5428 1.0079    0
## x  1.9683 0.1670    1.6402    1.9682    2.2964 1.9681    0
```

Similarly, the `summary.hyperpar` for the hyperparameters. The likelihood hyperparameter is parametrized as precision. We can work with its PMDs transforming it to standard deviation:

```
pmd.s2e <- inla.tmarginal(function(x) sqrt(1/x), ## inverse and square root
  res$marginals.hyperpar$'Precision for the Gaussian observations')
```

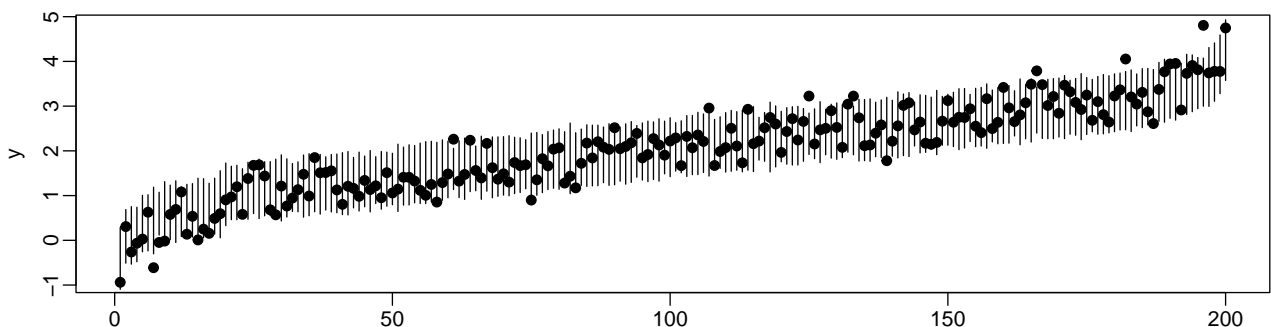
The PMDs for the likelihood SD and RF parameters can be visualized by

```
plot(pmd.s2e, type='l', ylab='Density', xlab=expression(sigma[e]))
abline(v=sqrt(s2e), col=2) ## add the 'true' value
plot(res$marginals.hy[[3]], type='l', xlab=expression(sigma[u]), ylab='Density')
abline(v=sqrt(s2u), col=2) ## add the 'true' value
plot(res$marginals.hy[[2]], type='l', xlab='range nominal', ylab='Density')
abline(v=r, col=2) ## add the 'true' value
```



To look at the PMD for the linear predictor,  $E(Y|X, U)$ , at each data location will be too much. For now, we can show its the 2.5% and 97.5% quantiles ordered by its mean on top of  $y$  values:

```
idx.obs <- inla.stack.index(stk.e, tag='est')$data ## index in the stack
order.eta <- order(res$summary.fitted.values$mean[idx.obs])
plot(y[order.eta], pch=19, ylab='y')
segments(1:n, res$summary.fitted.val$'0.025quant'[idx.obs][order.eta],
  1:n, res$summary.fitted.val$'0.975quant'[idx.obs][order.eta])
```



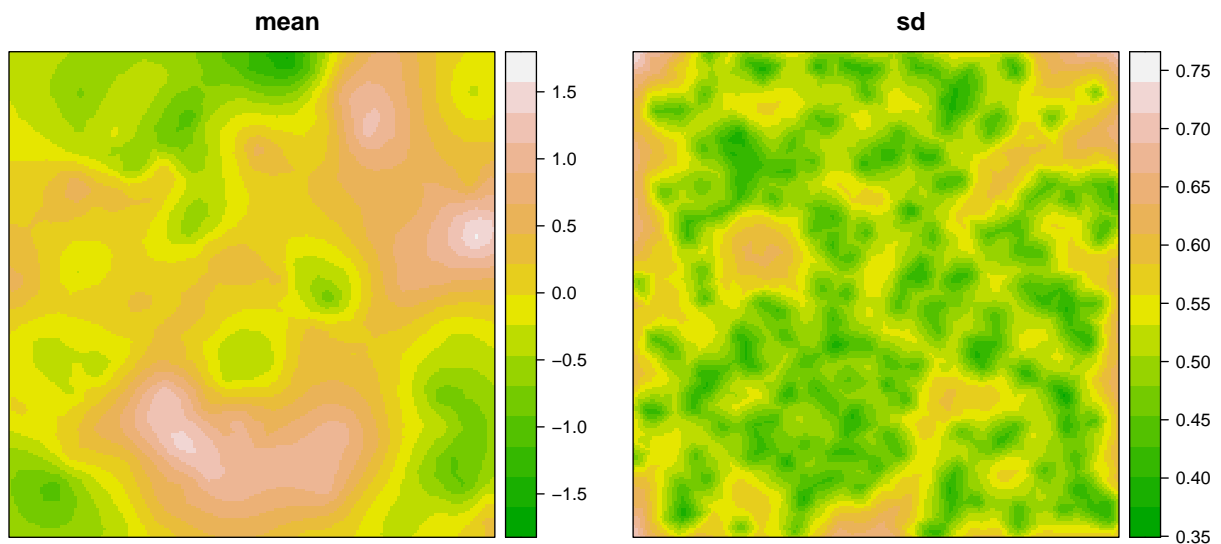
## 4 RF projection on a grid

An interesting result is the map of the RF on a grid. The simplest way to have it is by projection. We just have to define the projector matrix and project, for example, the posterior mean and posterior standard deviation on the grid.

```
nxy <- c(200, 200)
gproj <- inla.mesh.projector(mesh, xlim=0:1, ylim=0:1, dims=nxy)
g.mean <- inla.mesh.project(gproj, res$summary.random$idx.u$mean)
g.sd <- inla.mesh.project(gproj, res$summary.random$idx.u$sd)
```

We can visualize it by

```
library(lattice); library(gridExtra)
trellis.par.set(regions=list(col=terrain.colors(16)))
grid.arrange(levelplot(g.mean, scales=list(draw=F), xlab='', ylab='', main='mean'),
              levelplot(g.sd, scal=list(draw=F), xla='', yla='', main='sd'), nrow=1)
```



## 5 Prediction

The prediction is usually needed when one wants to know the distribution the expected value for the outcome given the data. It consider each model component, not only the random field, which is only one component in the model, showed in the previous section.

First, one has to define the scenario for the prediction, that is the locationas and value for the covariates. We show an example with only three locations, predictions over a fine grid can also be considered, and covariate values set in its the mean value

```
tcoo <- rbind(c(0.3,0.9), c(0.5,0.5), c(0.7,0.3))
dim(Ap <- inla.spde.make.A(mesh=mesh, loc=tcoo))

## [1] 3 1689

x0 <- c(0.5, 0.5, 0.5)
```

There is more than one ways to compute the posterior marginals for the linear predictor. When predictions over a fine grid is needed, it will be preferable a computationally cheaper way.

## 5.1 Expensive way: NA's in the response vector

An usual way is to build a scenario, for fixed and random effects, and assign NA for the outcome. In this case, the linear predictor for such missing observations is also part of the model graph, [Rue et al., 2017], and is treated in the entire model fitting process.

Defining a prediction stack, join and use the full stack in `inla()`

```
stk.pred <- inla.stack(tag='pred', A=list(Ap, 1), data=list(y=NA), ## response as NA
  effects=list(idx.u=1:spde$n.spde, data.frame(x=x0, b0=1))) ## all idx.u
stk.full <- inla.stack(stk.e, stk.pred) ## join the data and prediction scenario
p.res <- inla(mf, data=inla.stack.data(stk.full), ## supply the full data
  control.compute=list(return.marginals.predictor=TRUE),
  control.predictor=list(compute=TRUE, A=inla.stack.A(stk.full)), ## full
  control.mode=list(theta=res$mode$theta, restart=FALSE)) ## use mode already found
```

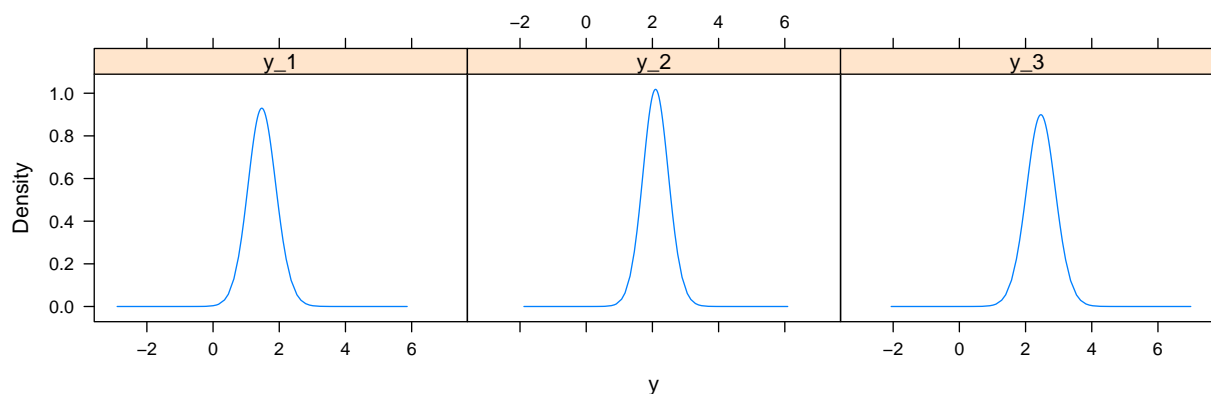
Get the prediction data index and have a look into the summary

```
pred.ind <- inla.stack.index(stk.full, tag='pred')$data
round(p.res$summary.fitted.val[pred.ind,], 4)

##               mean      sd 0.025quant 0.5quant 0.975quant  mode
## fitted.APredictor.201 1.4834 0.4381    0.6292   1.4797    2.3576  1.4720
## fitted.APredictor.202 2.1032 0.3985    1.3251   2.1006    2.8956  2.0950
## fitted.APredictor.203 2.4654 0.4526    1.5755   2.4646    3.3607  2.4629
```

Collect the linear predictor PMDs to work with, and isualize with commands bellow

```
ypost <- p.res$marginals.fitted.values[pred.ind]
names(ypost) <- paste('y', seq_along(ypost), sep='_');
xyplot(y~x | .id, ldply(ypost), panel='llines', xlab='y', ylab='Density')
```



In **INLA** we have some functions to work with marginals distributions

```
apropos('marginal')

[1] "inla.1djmarginal"
[2] "inla.dmarginal"
[3] "inla.emarginal"
[4] "inla.hpdmarginal"
[5] "inla.mmarginal"
[6] "inla.pmarginal"
[7] "inla.qmarginal"
[8] "inla.rjmarginal"
4 [9] "inla.rjmarginal.eval"
[10] "inla.rmarginal"
[11] "inla.smarginal"
```

```
[16] "marginalSUNdistr"

## inla.mmarginal(ypost[[1]]) ## mode
inla.qmarginal(c(0.15, 0.7),
               ypost[[1]]) ## quantiles

## [1] 1.034313 1.705013

inla.pmarginal(inla.qmarginal(
               0.3, ypost[[1]]), ypost[[1]])

## [1] 0.3
```

## 5.2 Cheaper way: Monte Carlo samples

The way we show here is cheaper if the number of locations to be predicted is not small. The idea is to draw samples from the joint posterior distribution. As any functional of interest can be computed we can compute the linear predictor at a set of target locations.

We can ask `inla()` to store the precision matrix for each hyperparameter configuration and use it to draw Monte Carlo samples, [Rue et al., 2017]. The `inla.posterior.sample()` function drawn samples from each hyperparameter configuration using its relative posterior as weights and then sample from the latent field for each hyperparameter sampled configuration.

We can set `control.compute=list(cofig=TRUE)` before the `inla()` to have such configurations. Or, we can rerun the model after asking it

```
res$.args$control.compute$config <- TRUE
res <- inla.rerun(res)
```

Drawn Monte Carlo samples

```
samples <- inla.posterior.sample(n=2e3, result=res, add.names=FALSE)
```

We have to find the index set for the elements we need samples, the fixed effects ('b0' and 'x') and the random effect ('s'). The names were stored for the first sample as rownames of the latent field

```
xnames <- rownames(samples[[1]]$latent) ### collect the names
idx <- lapply(c('b0', 'x', 'idx.u'), function(nam) ## for each effect
             which(substr(xnames, 1, nchar(nam))==nam)) ## find the index
```

These indexes are used to collect the desired part of the latent field and organize it into a matrix

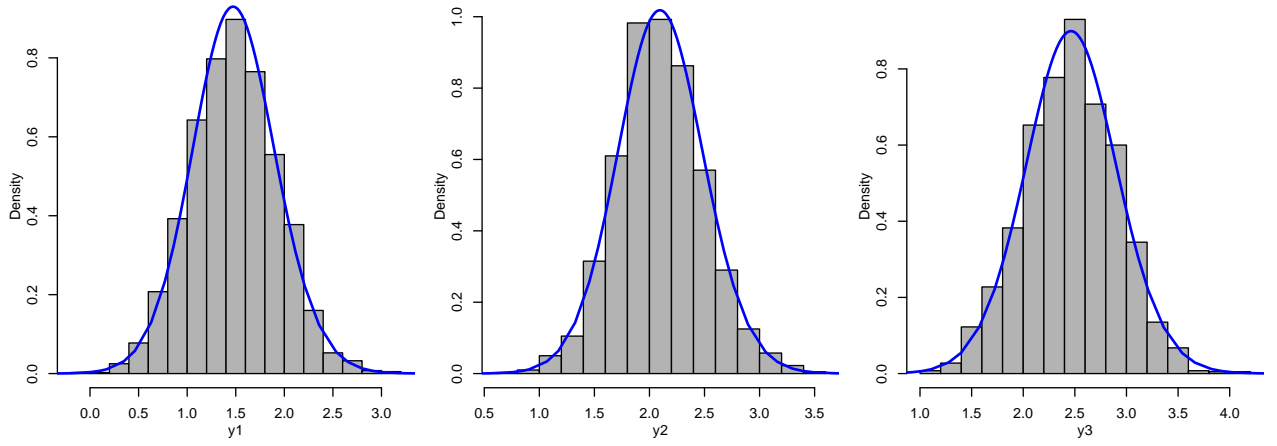
```
mat.samples <- sapply(samples, function(spl)
  c(b0=spl$latent[idx[[1]]], x=spl$latent[idx[[2]]], u=spl$latent[idx[[3]]]))
```

The next step is to compute the linear predictor for the scenario needed for each sample.

```
eta3.sample <- as.matrix(cbind(b0=1, x=0.5, u=Ap)%*%mat.samples)
```

We can visualize it comparing with the previous result with

```
par(mfrow=c(1,3), mar=c(3,3,1,1), mgp=c(2,1,0))
for (j in 1:3) {
  hist(eta3.sample[j,], freq=FALSE, xlab=paste0('y',j), main='', col=gray(0.7))
  lines(ypost[[j]], lwd=2, col='blue')
}
```



### 5.3 Prediction on a grid, cheaper way

We can consider the previously defined grid projection matrix and samples

```
eta.g.samp <- as.matrix(cbind(b0=1, x=0.5,
                             s=gproj$proj$A)%*%mat.samples)
```

Any summary statistics can be computed from that. Computing the mean, standard deviation,  $P(E(Y) > 3)$  and the 95% quantile for  $E(Y)$ :

```
library(matrixStats)
ss <- list(mean=rowMeans(eta.g.samp),
           sd=rowSds(eta.g.samp),
           p3=rowMeans(eta.g.samp>3),
           q95=rowQuantiles(eta.g.samp, probs=0.95))
```

It is visualized with the commands below (locations added on top of the sd map):

```
library(fields); for (s in c(1,3,4,2))
  image.plot(list(x=gproj$x, y=gproj$y,
                 z=matrix(ss[[s]], nxy[1])), asp=1,
             legend.mar=3, main=names(ss)[s])
points(coo, pch=4, cex=0.5)
```

## References

[Fuglstad et al., 2017] Fuglstad, G., Simpson, D., Lindgren, F., and Rue, H. (2017). Constructing priors that penalize the complexity of gaussian random fields. *submitted*.

[Lindgren and Rue, 2015] Lindgren, F. and Rue, H. (2015). Bayesian spatial and spatio-temporal modelling with R-INLA. *Journal of Statistical Software*, 63(19).

[Lindgren et al., 2011] Lindgren, F., Rue, H., and Lindström, J. (2011). An explicit link between gaussian fields and gaussian markov random fields: the stochastic partial differential equation approach (with discussion). *J. R. Statist. Soc. B*, 73(4):423–498.

[Rue et al., 2009] Rue, H., Martino, S., and Chopin, N. (2009). Approximate bayesian inference for latent gaussian models using integrated nested laplace approximations (with discussion). *Journal of the Royal Statistical Society, Series B*, 71(2):319–392.

[Rue et al., 2017] Rue, H., Riebler, A. I., Sørbye, S. H., Illian, J. B., Simpson, D. P., and Lindgren, F. K. (2017). Bayesian computing with inla: A review. *Annual Review of Statistics and Its Application*, 4:395–421.

