SPDE how to

Elias T. Krainski

created: March 31, 2016, last update 2016-04-14

How to fit a simple SPDE model in INLA

This document ilustrates how to do a geostatistical fully Bayesian analysis through the **S**tochastic **P**artial **D**iferential **E**quation approach http://onlinelibrary.wiley.com/doi/10.1111/j.1467-9868.2011.00777.x/full using the **I**ntegrated **N**ested **L**aplace **A**proximation, http://onlinelibrary.wiley.com/doi/10.1111/j.1467-9868.2008.00700.x/full implementation in the package available at http://www.r-inla.org.

Simulating some data

Define some random **Locations** and the Random Field (RF) **covariance** matrix, considering exponential correlation function:

```
n = 200  ## number of location points
coo = matrix(runif(2*n), n) ## location points
k <- 10;   s2s <- 0.7 ## RF parameters
R <- s2s*exp(-k*as.matrix(dist(coo))) ## covariance matrix
```

Draw a **RF** sample: a multivariate Normal realization

```
s <- drop(rnorm(n)%*%chol(R)) ## one RF realization
```

Adding a **covariate** effect and a noise

```
x <- runif(n)  ## covariate
beta <- 1:2  ## regression coefficients
lin.pred <- beta[1] + beta[2]*x + s ## linear predictor
s2e <- 0.3 ## error variance (nugget)
y <- lin.pred + rnorm(n, 0, sqrt(s2e))  ## the outcome
```

Model fitting steps

- **Mesh**: a triangulation to discretize the random field (RF) at 'm' nodes.

```
r0.1 <- sqrt(0.5 * 8)/k ## distance with correlation around 0.139
mesh <- inla.mesh.2d( ## 2D mesh creator
  loc=coo, ## provided locations
  max.edge=c(r0.1/3, r0.1), ## maximum edge length (inner, outer): mandatory
  offset=c(r0.1/3, r0.1*2), ## outer extension
  cutoff=r0.1/10) ## good to have >0
par(mar=c(0,0,1,0))
plot(mesh, asp=1) ## plot the mesh
points(coo, col='red') ## add the points
```

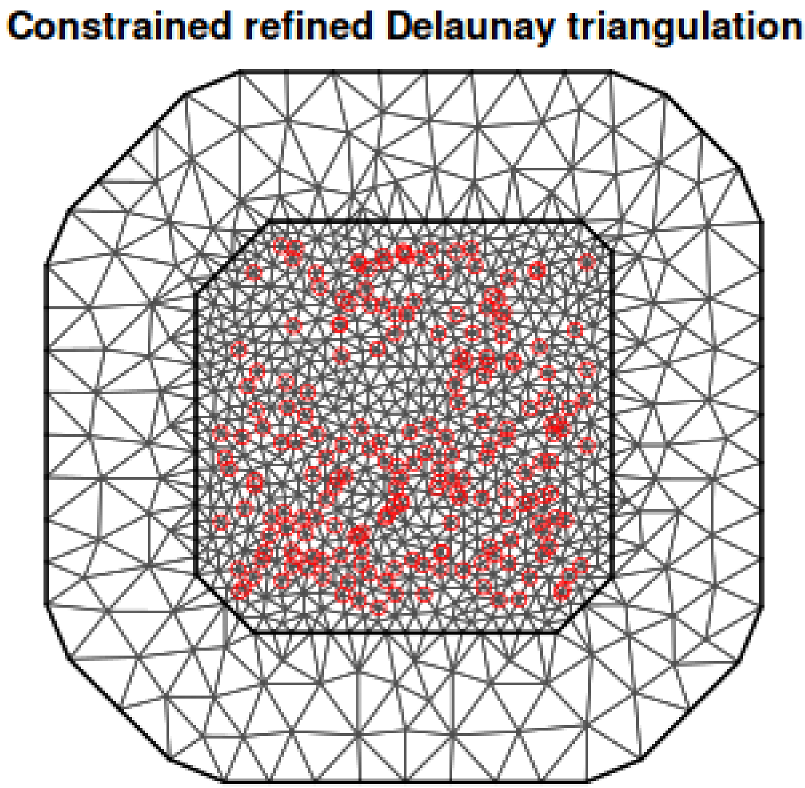

Figure 1: plot of chunk mesh

A little *warning* about the mesh. The additional triangles outer domain is to avoid boundary effects. Is good to have aproximately isosceles triangles. And, to avoid tiny triangles. We need to have edges lengths of the inner mesh triangles less than the range of the process. Of course, if it is too small, there might not be any spatial effect.

- **Define the** n x m **projector matrix** to project the process at the mesh nodes to locations

```
image(A <- inla.spde.make.A( ## projector creator
    mesh=mesh, ## provide the mesh
    loc=coo) ### locations where to project the field
    ) ## an 'n' by 'm' projector matrix
```

- **Build the SPDE model** on the mesh. Set $\alpha = 3/2$ to build the precision structure for an Exponential correlation function

```
spde <- inla.spde2.matern( ## precision components creator
    mesh=mesh, ## mesh supplied
    alpha=1.5) ## smoothness parameter
```

- **Create a data stack** to organize the data. This is a way to allow models with complex linear predictors. In our case, we have a SPDE model defined on m nodes. It must be combined with the covariate (and the intercept) effect at n locations. We do it using different projector matrices.

```
stk.e <- inla.stack( ## stack creator
  data=list(y=y),  ## response
```

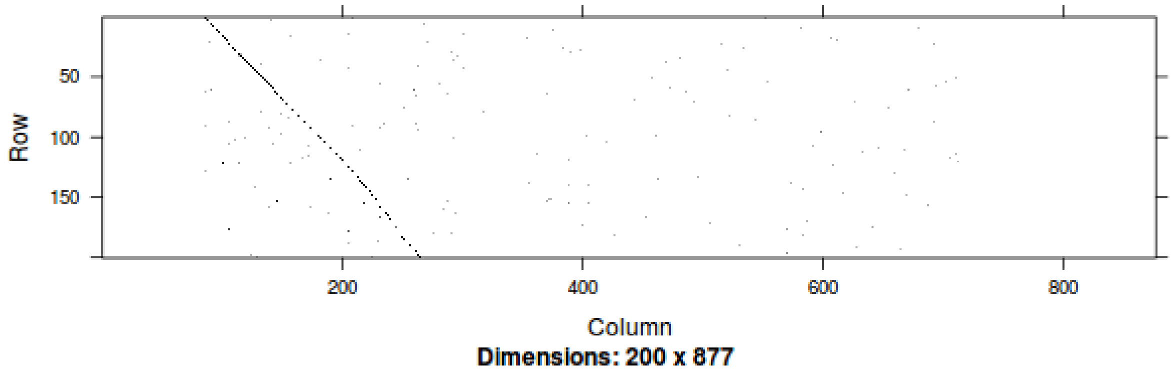

Figure 2: plot of chunk projector

```
  effects=list(## two elements:
    data.frame(b0=1, x=x), ## regressor part
    s=1:spde$n.spde),  ## RF index
  A=list(## projector list of each effect
    1, ## for the covariates
    A), ## for the RF
  tag='est') ## tag
```

- **Fit** the posterior marginal distributions for all model parameters

```
formula <- y ~ 0 + b0 + x + ## fixed part
  f(s, model=spde) ## RF term
res <- inla( ## main function in INLA package
  formula, ## model formula
  data=inla.stack.data(stk.e), ## dataset
  control.predictor=list( ## inform projector needed in SPDE models
    A = inla.stack.A(stk.e))) ## projector from the stack data
```

Posterior marginal distributions - PMDs

Summary of the regression coefficients PMDs

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

```
##      mean     sd 0.025quant 0.5quant 0.975quant   mode kld
## b0 1.0182 0.2921     0.3963   1.0291     1.5753 1.0466   0
## x  1.9122 0.1919     1.5331   1.9128     2.2872 1.9142   0
```

We have to transform the precision PMD to have the variance PMD. It can be done and visialized by

```
m.prec <- res$marginals.hyperpar$'Precision for the Gaussian observations' ## the marginal
post.s2e <- inla.tmarginal(## function to compute a tranformation
  function(x) 1/x, ## inverse transformation
  m.prec) ## marginal to be applied
### visualize it
plot(post.s2e, type='l', ylab='Density',
     xlab=expression(sigma[e]^2))
abline(v=s2e, col=2) ## add value used to generate the data
```

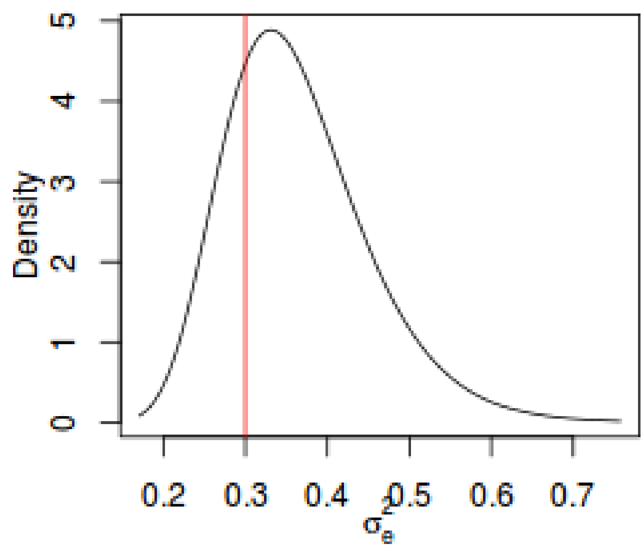

Figure 3: plot of chunk nugget

The SPDE approach uses a local variance, τ^2, such that $\sigma_s^2 = 1/(2\pi\kappa^2\tau^2)$. On **INLA** we work $\log(\tau^2)$ and $\log(\kappa)$. So, especially for σ_s^2, we have to do an additional computation. The PMDs for all RF parameters on user scale are computed by

```
rf <- inla.spde.result( ## function to compute the 'interpretable' parameters
    inla=res, ## the inla() output
    name='s', ## name of RF index set
    spde=spde, ## SPDE model object
    do.transf=TRUE) ## to user scale
```

It can be visualized by

```
par(mfrow=c(1,3), mar=c(3,3,0.3,0.3), mgp=c(2,0.5,0))
plot(rf$marginals.var[[1]], ty='l',
     xlab=expression(sigma[s]^2), yla='Density')
abline(v=s2s, col=2) ## add the true value
plot(rf$marginals.kap[[1]], type='l',
     xlab=expression(kappa), ylab='Density')
abline(v=k, col=2) ## add the true value
plot(rf$marginals.range[[1]], type='l',
     xlab='range nominal', ylab='Density')
abline(v=sqrt(0.5 * 8)/k, col=2) ## add the 'true' value
```

Projection on a grid / visualization

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.

```
gproj <- inla.mesh.projector( ## projector builder
  mesh, ## mesh used to define the model
```

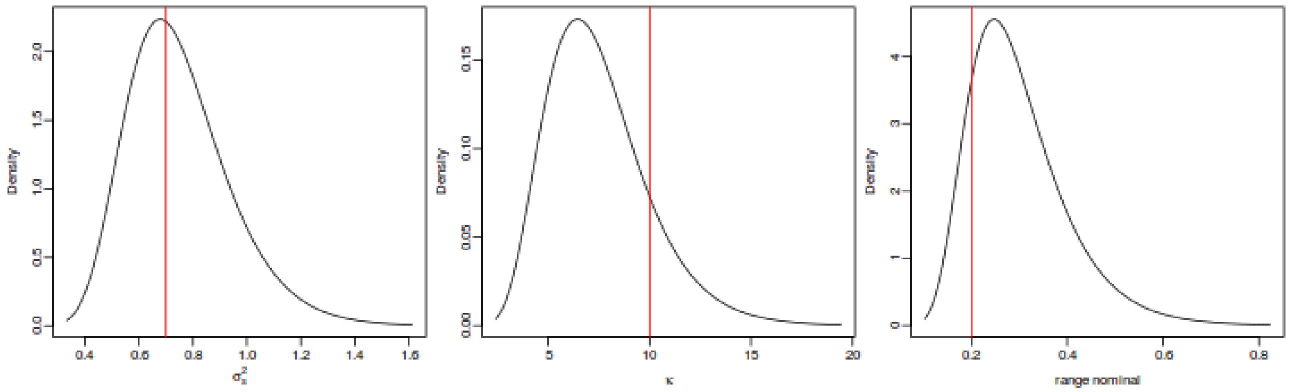

Figure 4: plot of chunk parameters

```
  xlim=0:1, ylim=0:1, ## limits where to create the grid
  dims=c(300,300)) ## grid dimension
## project the mean and the SD
g.mean <- inla.mesh.project(gproj, res$summary.random$s$mean)
g.sd <- inla.mesh.project(gproj, res$summary.random$s$sd)
```

We can visualize it by

```
par(mfrow=c(1,2), mar=c(0,0,1,0))
require(fields)
image.plot(g.mean, asp=1, main='RF posterior mean', axes=FALSE, horizontal=TRUE)
image.plot(g.sd, asp=1, main='RF posterior SD', axes=FALSE, horizontal=TRUE)
```

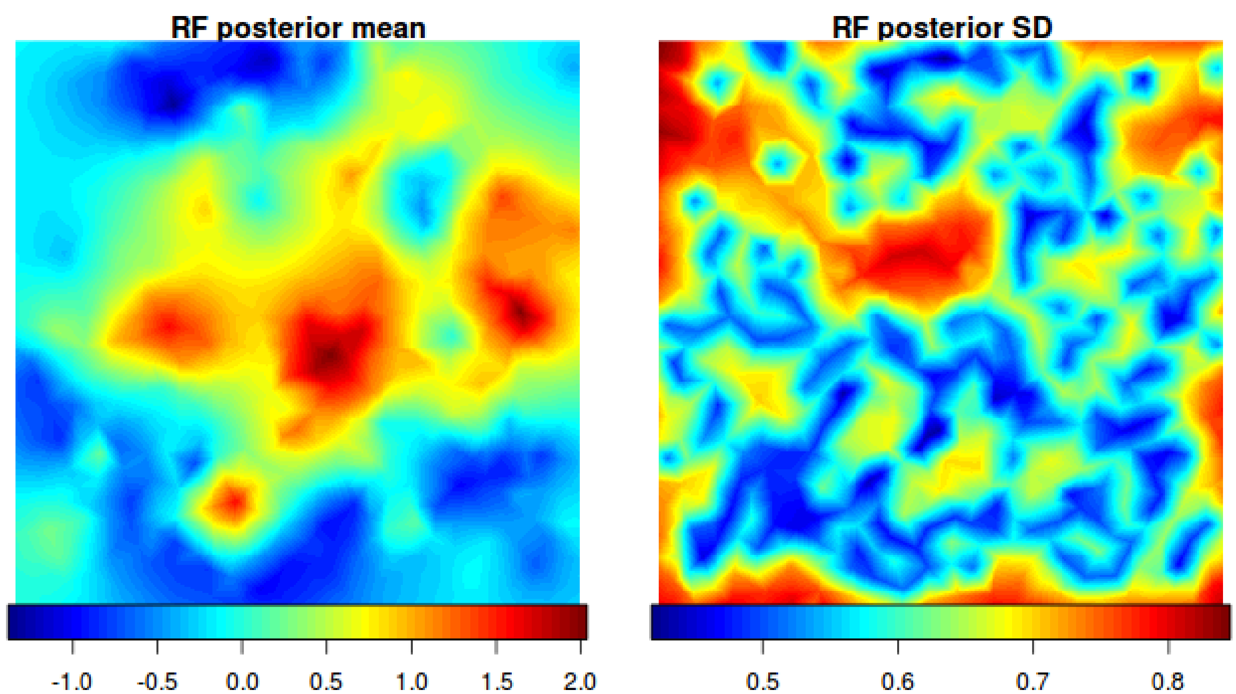

Figure 5: plot of chunk visualize

Prediction

Define the set of target locations, the corresponding projector matrix and covariate values at target locations

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

```
## [1]   3 877
```

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

To do a fully Bayesian analysis, we have to include the target locations on the estimation process by assigning NA for the response at these locations. Defining the prediction stack

```
stk.pred <- inla.stack(
  tag='pred', ## will be used to collect the posterior marginals
  data=list(y=NA), ## response set as NA
  effects=list(
    data.frame(x=x0, b0=1), ## covariate scenario
    s=1:spde$n.spde), ## same as before
  A=list(1, Ap)) ## covariate and target locations field projectors
```

Fit the model again with the full stack

```
stk.full <- inla.stack(stk.e, stk.pred) ## join both data
p.res <- inla(
  formula, data=inla.stack.data(stk.full), ## using the full data
  control.predictor=list(compute=TRUE, ## compute the predictor
                         A=inla.stack.A(stk.full)), ## from full data
  control.mode=list(theta=res$mode$theta)) ## use the mode previously found
```

Get the prediction data index and collect the linear predictor PMDs to work with

```
pred.ind <- inla.stack.index( ## stack index extractor function
  stk.full, ## the data stack to be considered
  tag='pred' ## which part of the data to look at
  )$data ## which elements to collect
ypost <- p.res$marginals.fitted.values[pred.ind]
```

Visualize with commands bellow

```
xyl <- apply(Reduce('rbind', ypost), 2, range)
par(mfrow=c(1,3), mar=c(3,3,2,1), mgp=c(2,1,0))
for (j in 1:3)
  plot(ypost[[j]], type='l', xlim=xyl[,1], ylim=xyl[,2],
       xlab='y', ylab='density', main=paste0('y', j))
```

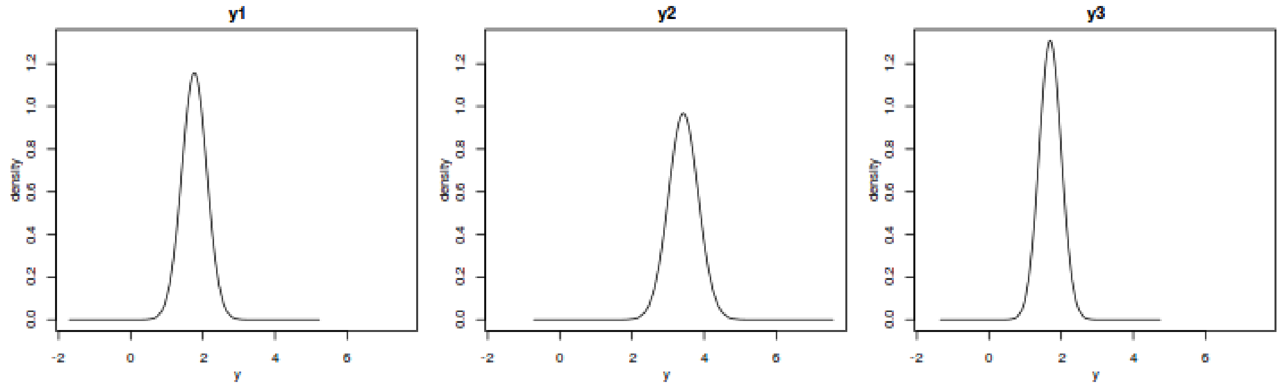

Figure 6: plot of chunk predicted

Manipulating marginals

We have already used the *inla.tmarginal()* function. There are some other functions to work with marginal distributions which may be usefull as well:

```
apropos('marginal')
```

```
##  [1] "inla.dmarginal"   "inla.emarginal"   "inla.hpdmarginal"
##  [4] "inla.mmarginal"   "inla.pmarginal"   "inla.qmarginal"
##  [7] "inla.rmarginal"   "inla.smarginal"   "inla.tmarginal"
## [10] "inla.zmarginal"
```

Playing with the posterior marginal for the first target location

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

```
## [1] 1.402537 1.941658
```

```
inla.emarginal(function(x) x^2, ypost[[1]]) - ## E(y^2) -
  inla.emarginal(function(x) x, ypost[[1]])^2 ## E(y)^2 to compute the variance
```

```
## [1] 0.07995596
```

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

```
## [1] 0.3
```

```
inla.zmarginal(ypost[[1]]) ## posterior summary
```

```
## Mean            1.76299
## Stdev           0.282765
## Quantile  0.025 1.07944
## Quantile  0.25  1.5281
## Quantile  0.5   1.76103
## Quantile  0.75  1.99335
## Quantile  0.975 2.43761
```