Spatial Statistics using R-INLA and Gaussian Markov random fields

David Bolin and Johan Lindström

1 Introduction

In this lab we will look at an example of how to use the SPDE models in the INLA package. We will analyse one month of data from the popular data set of precipitation measurements from the Paraná region in Brazil. The aims of this practical are to get a general understanding of how to:

- create meshes for continuously indexes spatial effects.
- implement SPDE models in R-INLA.
- include fixed and random effects in the model.
- obtain predictions at unobserved locations.

For a much more thorough description of R-INLA and the details underlying the SPDE-models see:
`www.math.ntnu.no/inla/r-inla.org/papers/jss/lindgren.pdf`
For more details on the example we study here, see:
`www.math.ntnu.no/inla/r-inla.org/tutorials/spde/spde-tutorial.pdf`

2 A model for precipitation

Precipitation data is always positive, and our statistical model therefore needs to take this into account. A popular modelling assumption for precipitation is that it is Gamma distributed. INLA uses the following parameterisation of the Gamma distribution, $\mathsf{\Gamma}(\mu, c\phi)$

$$\pi(y) = \frac{1}{\Gamma(c\phi)} \left(\frac{c\phi}{\mu}\right)^{c\phi} y^{c\phi-1} \exp\left(-(c\phi)\frac{y}{\mu}\right).$$

The distribution has expected value $\mathsf{E}(x) = \mu$ and variance $\mathsf{V}(x) = \mu^2/(c\phi)$, where c is a fixed (known) scaling parameter and $1/\phi$ is a dispersion parameter.

Here μ will be modelled using a stochastic model that includes both covariates and spatial structure, resulting in the latent Gaussian model for the precipitation measurements

$$\begin{aligned} y_i|\mu(s_i), \theta &\sim \mathsf{\Gamma}(\mu(s_i), c\phi) \\ \log(\mu(s)) = \eta(s) &= \sum_k f_k(c_k(s)) + x(s) \\ \theta &\sim \pi(\theta) \end{aligned} \tag{1}$$

where y_i denotes the measurement taken at location s_i, $c_k(s)$ are covariates, $x(s)$ is a mean-zero Gaussian Matérn field, and θ is a vector containing all parameters of the model.

3 Examining the data

3.1 Load R-packages

First we load some libraries we need.

```
> library(gridExtra); library(ggplot2); library(lattice)
> library(INLA); library(splancs); library(fields)
```

You might need to install INLA (and its dependencies)

```
> source("http://www.math.ntnu.no/inla/givemeINLA.R")
```

3.2 Load and examine data

Load the data and the border of the region

```
> data(PRprec); data(PRborder)
```

The data frame contains daily measurements at 616 stations for the year 2011, as well as coordinates and altitude information for the measurement stations. We will not analyse the full spatio-temporal data set, but instead look at the total precipitation in January, which we calculates as

```
> Y <- rowMeans(PRprec[,3+1:31])
```

Let's also extract the coordinates and the altitude and remove the locations with missing values

```
> ind <- !is.na(Y)
> Y <- Y[ind]
> coords <- as.matrix(PRprec[ind,1:2])
> alt <- PRprec$Altitude[ind]
```

Plot the precipitation using the `ggplot` function (messy code to get a nice plot)

```
> ggplot() +
    geom_point(aes(x=coords[,1],y=coords[,2],colour=Y), size=2,
               alpha=1) +
    scale_colour_gradientn(colours=tim.colors(100)) +
    geom_path(aes(x=PRborder[,1],y=PRborder[,2])) +
    geom_path(aes(x=PRborder[1034:1078,1],
                  y=PRborder[1034:1078,2]), colour="red")
```

The red line in the figure shows the coast line, and we believe that the distance to the coast might be a good covariate for precipitation. This covariate is not available, so we have to calculate it for each observation location

```
> seaDist <- apply(spDists(coords, PRborder[1034:1078,],
                           longlat=TRUE), 1, min)
```

Plot precipitation as a function of the possible covariates

```
> par(mfrow=c(2,2))
> plot(coords[,1], Y, cex=.5, xlab="Longitude")
> plot(coords[,2], Y, cex=.5, xlab="Latitude")
> plot(seaDist, Y, cex=.5, xlab="Distance to sea")
> plot(alt, Y, cex=.5, xlab="Altitude")
```

Unfortunately we only have the altitude information at the measurement locations, so if we want to do kriging using a model with altitude as a covariate we would have to obtain the altitude information also at the prediction locations. This is indeed possible but to simplify things, we discard the altitude covariate from the analysis.

4 Creating the SPDE model

4.1 The mesh

Creating a good mesh for the SPDE method is important. A naive approach would be to only triangulate the observations points

```
> m.bad <- inla.mesh.create(coords)
> plot(m.bad, asp=1, main=""); lines(PRborder, col=3)
```

The resulting triangulation is "bad", having too sharp angles and triangels of many different sizes.

Instead we use a triangulation with "regular" triangles, and extend the mesh a bit outside the region of interest to reduce boundary effects

```
> m1 <- inla.mesh.2d(coords, max.edge=c(.45,1), cutoff=0.2)
> plot(m1, asp=1, main=""); lines(PRborder, col=3)
> points(coords[,1], coords[,2], pch=19, cex=.5, col="red")
```

The function takes the convex hull of the region and extends that to create the region for the "inner" mesh, with small triangles. Try changing the parameters of `inla.mesh.2d` to create different meshes, you can:

1. decide how much the region should be extended by changing the parameter `offset`
2. increase/decrease the density of the mesh by changing the `max.edge` and `cutoff`

We might want to use a mesh which is based on a non-convex hull to avoid adding many small triangles outside the domain of interest (more triangles = larger computation times), which can be done as follows

```
> prdomain <- inla.nonconvex.hull(coords, -0.03, -0.05,
                                  resolution=c(100,100))
> prmesh <- inla.mesh.2d(boundary=prdomain, max.edge=c(.45,1),
                         cutoff=0.2)
> plot(prmesh, asp=1, main=""); lines(PRborder, col=3)
> points(coords[,1], coords[,2], pch=19, cex=.5, col="red")
```

the rest of the code is based on this mesh, but use your own mesh if you want to. If you want to, you can also try the rest of the analysis using a couple of different meshes to see how sensitive the results are to the choice of the mesh.

4.2 The observation matrix (A)

We now create the A matrix, that connects the mesh to the observation locations and then create the SPDE model

```
> A <- inla.spde.make.A(prmesh, loc=coords)
> spde <- inla.spde2.matern(prmesh, alpha=2)
```

4.3 The `inla.stack`

Since the covariates already are evaluated at the observation locations, we only want to apply the A matrix to the spatial effect and not the fixed effects. It is difficult to do this manually, but we can use the `inla.stack` function

```
    > mesh.index <- inla.spde.make.index(name="field",
                                         n.spde=spde$n.spde)
    > stk.dat <- inla.stack(data=list(y=Y), A=list(A,1), tag="est",
                            effects=list(c(mesh.index,list(Intercept=1)),
                              list(long=inla.group(coords[,1]),
                                   lat=inla.group(coords[,2]),
                                   seaDist=inla.group(seaDist))))
```

Here the observation matrix A is applied to the spatial effect and the intercept while an identity observation matrix, denoted by "1", is applied to the covariates. This means the covariates are unaffected by the observation matrix.

Understanding the `inla.stack` is key to building more complex models. The observation matrices in `A=list(A,1)` are used to link the corresponding elements in the `effects`-list to the observations. Thus in our model the latent spatial-field (`mesh.index`) and the intercept are linked to the log-expectation of the observations, i.e. $\eta(s)$ in (1), through the A-matrix. The covariates, on the other hand, are linked directly to $\eta(s)$. The `stk.dat` object defined above implies the following principal linkage between model components and observations

$$\eta(s) \sim A \cdot x(s) + A \cdot \text{Intercept} + \text{long} + \text{lat} + \text{seaDist}$$

$\eta(s)$ will then be used in the observation-likelihood,

$$y_i|\eta(s_i), \theta \sim \Gamma\left(\exp(\eta(s_i)), c\phi\right)$$

4.4 Model fitting

We can now fit a model using, for example, distance to the sea as a covariate through a random walk 1 model.

```
    > f.s <- y ~ -1 + Intercept + f(seaDist, model="rw1") +
        f(field, model=spde)
```

Here `-1` is added to remove R's implicit intercept, which is replaced by the explicit `+Intercept` from when we created the stack.

We use the two functions `inla.stack.data` and `inla.stack.A` to extract the data and the A matrix from the stack when we call to the `inla` function to estimate the model.

```
    > r.s <- inla(f.s, family="Gamma",
                  data=inla.stack.data(stk.dat), verbose=TRUE,
                  control.predictor=list(A=inla.stack.A(stk.dat),
                    compute=TRUE))
```

4.5 INLA results

We can look at some summaries of the posterior distributions for the parameters, for example the fixed effects (i.e. the intercept) and the hyper-parameters (i.e. dispersion in the gamma likelihood, the precision of the RW1, and the parameters of the spatial-field)

```
    > r.s$summary.fixed
    > r.s$summary.hyperpar
```

The SPDE model is parametrised using an internal parameterisation that might be difficult to interpret; however, we can extract the posterior distributions for the range and variance parameters through

```
    > r.f <- inla.spde2.result(r.s, "field", spde, do.transf=TRUE)
```

and based on the posterior distribution, we can calculate the posterior mean using the function `inla.emarginal`, for example for the variance

```
    > inla.emarginal(function(x) x,
                     r.f$marginals.variance.nominal[[1]])
```

The posterior distributions for the hyper-parameters can be plotted using

```
    > par(mfrow=c(2,3))
    > plot(r.s$marginals.fix[[1]], type="l", xlab="Intercept",
           ylab="Density")
    > plot(r.s$marginals.hy[[1]], type="l", ylab="Density",
           xlab=expression(phi))
    > plot.default(r.f$marginals.variance.nominal[[1]], type="l",
                   xlab=expression(sigma[x]^2), ylab="Density")
    > plot.default(r.f$marginals.range.nominal[[1]], type="l",
                   xlab="Practical range", ylab="Density")
```

For this model, we used a random effect for the distance to the sea covariate, we can plot that as well as a pointwise confidence band for it,

```
    > plot(r.s$summary.random$seaDist[,1:2], type="l",
           xlab="Distance to sea (Km)", ylab="random effect")
    > abline(h=0, lty=3)
    > for (i in c(4,6))
          lines(r.s$summary.random$seaDist[,c(1,i)], lty=2)
```

as well as the posterior precision of the random effect.

```
    > plot(r.s$marginals.hy[[2]], type="l", ylab="Density",
           xlab=names(r.s$marginals.hy)[2])
```

5 Calculating kriging predictions

Finally, we would like to calculate a prediction (i.e. do kriging) of the expected precipitation on a dense grid in the region. In order to do this, we first need to create the grid to do the prediction to

```
    > nxy <- c(150,100)
    > projgrid <- inla.mesh.projector(prmesh,
                                       xlim=range(PRborder[,1]),
                                      ylim=range(PRborder[,2]),
                                       dims=nxy)
```

This lattice contains 150×100 locations, and you can change the resolution of the kriging prediction by changing `nxy`. Find the cells that are outside the region of interest, we will not plot the estimates there.

```
    > xy.in <- inout(projgrid$lattice$loc,
                     cbind(PRborder[,1], PRborder[,2]))
```

The locations we do prediction to is plotted by

```
    > coord.prd <- projgrid$lattice$loc[xy.in,]
    > plot(coord.prd, type="p", cex=.1); lines(PRborder)
    > points(coords[,1], coords[,2], pch=19, cex=.5, col="red")
```

Now, there are a few ways we could calculate the kriging prediction. The simplest way is to evaluate the mean of all individual random effects in the linear predictor and then to calculate the exponential of their sum (since $\mu(s) = \exp(\eta(s))$). A more accurate way is to calculate the prediction jointly with the estimation, which unfortunately is quite computationally expensive if we do prediction on a fine grid. However, because we would like to see how this is done, we proceed with this option. First, link the prediction coordinates to the mesh nodes through an A matrix

```
    > A.prd <- projgrid$proj$A[xy.in, ]
```

Since we are using distance to the sea as a covariate, we also have to calculate this covariate for the prediction locations

```
> seaDist.prd <- apply(spDists(coord.prd, PRborder[1034:1078,],
                       longlat=TRUE), 1, min)
```

We now make a stack for the prediction locations. We have no data at the prediction locations, so we set `y=NA`. We then join this stack with the estimation stack.

```
>  ef.prd = list(c(mesh.index,list(Intercept=1)),
                 list(long=inla.group(coord.prd[,1]),
                 lat=inla.group(coord.prd[,2]),
                 seaDist=inla.group(seaDist.prd)))
>  stk.prd <- inla.stack(data=list(y=NA), A=list(A.prd,1),
                         tag="prd", effects=ef.prd)
>  stk.all <- inla.stack(stk.dat, stk.prd)
```

Doing the joint estimation takes a while, and we therefore turn of the computation of certain things that we are not interested in, such as the marginals for the random effect. We also use a simplified integration strategy (actually only using the posterior mode of the hyper-parameters) through the command `control.inla = list(int.strategy = "eb")`, i.e. empirical Bayes.

```
>  r2.s <- inla(f.s, family="Gamma",
                 data=inla.stack.data(stk.all),
                control.predictor=list(A=inla.stack.A(stk.all),
                                       compute=TRUE,link = 1),
                quantiles=NULL,
                control.results=list(return.marginals.random=F,
                                     return.marginals.predictor=F),
                verbose=TRUE,
                control.inla = list(int.strategy = "eb"))
```

Finally, we extract the indices to the prediction nodes and then extract the mean and the standard deviation of the response

```
> id.prd <- inla.stack.index(stk.all, "prd")$data
> sd.prd <- m.prd <- matrix(NA, nxy[1], nxy[2])
> m.prd[xy.in] <- r2.s$summary.fitted.values$mean[id.prd]
> sd.prd[xy.in] <- r2.s$summary.fitted.values$sd[id.prd]
```

Plot the results

```
> grid.arrange(levelplot(m.prd, col.regions=tim.colors(99),
                         xlab="", ylab="", main="mean",
                         scales=list(draw=FALSE)),
               levelplot(sd.prd, col.regions=topo.colors(99),
                         xlab="", ylab="",
                         scales=list(draw=FALSE),
                        main="standard deviation"))
```

6 Other things to try

If you have completed the practical, the following contains a few additional things that might be interesting to investigate.

6.1 Additional covariates and fixed effects

Investigate how different covariates affect the model by changing the model formulation in `f.s`. You can also try including the covariates as fixed or random effects (using random walk 1 or 2 models).

6.2 Using Gaussian observations

Given a latent model, it is straight forward to alter the observational model in INLA. To estimate the model above with Gaussian-likelihood instead of Gamma-likelihood simply change `family` to “gaussian” in the inla-call. How does this change the predictions?

A complete list of observational likelihoods is given by

```
    > str(inla.models()$likelihood,1)
```

or from `www.r-inla.org/models/likelihoods`.

6.3 Comparing the correlation to a Matérn

We could also compare dependence in the SPDE model to those of the Matérn covariance we're approximating.

First we compute the precision matrix given the posterior mean of the estimated parameters

```
    > Q <- inla.spde2.precision(spde,
                                  r.s$summary.hyperpar[3:4,"mean"])
```

and study the sparsity

```
    > image(Q)
```

Inverting the precision matrix gives the covariance and correlation matrices

```
    > S <- as.matrix(solve(Q))
    > S <- diag(1/sqrt(diag(S))) %*% S %*% diag(1/sqrt(diag(S)))
```

We also need the distance matrix

```
    > D <-  as.matrix(dist(prmesh$loc[,1:2]))
```

We now need the covariance function and variograms for the SPDE model. We obtain the approximate posterior means for the variance and range

```
    > range <- exp( r.f$summary.log.range.nominal$mean )
    > sigma <- exp( r.f$summary.log.variance.nominal$mean )
```

and compute the theoretical Matérn correlation function

```
    > d <- seq(0, max(D), len=1e3)
    > kappa <- sqrt(8*1)/range
    > S.theory <- (d*kappa) * besselK(d*kappa,1)
    > S.theory[1] <- 1
```

We can now compare the SPDE and Matérn correlation for 10 points at the centre of the mesh

```
    > par(mfrow=c(2,1))
    > I <- order(apply(D,2,max))[1:10]
    > plot(D[,I], as.matrix(S[,I]), pch=19, cex=.1,
          xlab="Distance", ylab="Covariance")
    > lines(d, S.theory, col=2)
```

or for the points on the boundary of the mesh

```
    > I <- unique(c(prmesh$segm$bnd$idx))
    > plot(D[,I], as.matrix(S[,I]), pch=19, cex=.1,
          xlab="Distance", ylab="Covariance")
    > lines(d, S.theory, col=2)
```

The rather larger spread around the theoretical function in the second plot is due to edge effects and the mesh size being reasonable large compared to the range. The first plot gives the correlation between 10 central points and the rest of the field, reducing the edge effects.