Statistical Modelling of Environmental Extremes
Lab sheet Practical Session 4

Daniela Castro-Camilo · Institute Henri Poincaré, Paris – March 2022

To get started, open R/RStudio and install and load the necessary libraries:

```
library(INLA)
```

Modelling spatial exceedances using latent Gaussian models

Modelling framework

As we learned in Session 1, INLA is a tool for Bayesian inference of latent Gaussian models (LGMs). Loosely speaking, this class of models assume that there is a (unobservable) latent process that drives the trends and dependence that we observe in the data. Here we focus on LGMs using the generalised Pareto likelihood for data observed over space.

Consider the data `gpsim` which contains GP observations simulated over $n = 200$ locations. At each location, we observe $m = 40$ exceedances and the value of a covariate x. This could represent, for instance, precipitation exceedances and scaled wind speed. We want to fit a GP model taking into account the spatial variability and the influence of the covariate x. To this end, we propose the following model

$$\begin{aligned} Y(s) &\sim \text{GPD}(q_{0.5}(s), \xi), \quad \text{where } q_{0.5} \text{ is the median} \\ \log(q_{0.5}(s)) = \eta(s) &= \beta_0 + \beta_1 x(s) + f(s), \end{aligned}$$

where β_0 is an intercept. Note that we assume that the covariate $x(s)$ has a linear effect over the log-median. We also assume a non-linear spatial effect, which corresponds to a Gaussian process (GP) with Matérn covariance structure. This means that the spatial covariance between two locations separated at a distance h is

$$C(h) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2}\frac{h}{\rho}\right)^{\nu} K_v \left(\sqrt{2}\frac{h}{\rho}\right),$$

where ν is a fixed smooth parameter, σ^2 is called marginal variance, ρ is the range of dependence and K_ν is the modified Bessel function of the second kind. Both σ^2 and ρ should be estimated. To estimate the spatial effect (which is, in principle, continuous), we discretise the study area using a triangular mesh and we define the covariance C over the mesh modes. INLA uses a very fast numerical technique to obtain estimates of σ^2 and ρ called the SPDE approach. Unfortunately, we can't go into details about this approach, but it is not fundamental at this point. The only thing you need to know is that it provides an efficient way to estimate the spatial effect.

For simplicity, we assume that the 40 replicates are independent over time. As we learned from Session 3, this assumption implies that we will most likely underestimate the uncertainty of estimates. A more sensible approach would assume both temporal and spatial patterns in the data. We can fit spatio-temporal models in INLA, but this is out of the scope of this session.

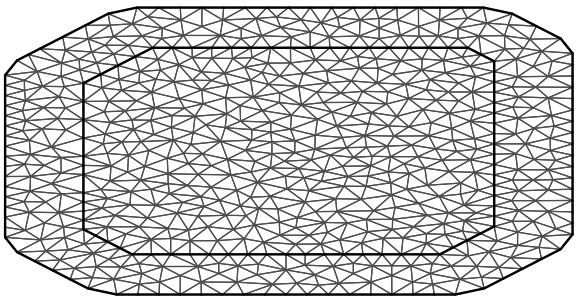
INLA fit

We start by reading the data and the additional file `gplocs.txt` which contains the unique locations:

```
gpsim = read.table('gpsim.txt', header = T)
loc.data = read.table('gplocs.txt', header = T)
```

The first thing we need to do is to define $f(s)$ and for that we need to discretise the study area using a triangular mesh:

```
mesh = inla.mesh.2d(loc = loc.data, max.edge = c(0.05, 0.1), cutoff = 0.05)
plot(mesh)
```



Before we go into the detail of `inla.mesh.2d`, note that the mesh has an inner domain and an outer extension. The inner domain contains the actual locations and it usually has a finer partition than the outer extension. This is to obtain a more accurate estimation of the spatial field. The outer extension. is there to avoid boundary problems when fitting the model. The function `inla.mesh.2d` takes information about the study region (here, the locations) and at least two additional arguments. The first one, `max.edge`, specifies the maximum allowed triangle edge length in the inner domain and the outer extension. So it can be a single numeric value or a vector of length two. The `cutoff` argument controls the shape of the triangles and is used to set the minimum allowed distance between points (to avoid too small triangles).

Next, we define the Matérn structure C. As we know, the smooth parameter is fixed ($\nu = 1$) while the marginal variance and range of dependence need to be estimated. Since we are in a Bayesian framework, we need to define prior distributions over σ and ρ (INLA does not have default priors in this case). We proceed as follows

```
spde = inla.spde2.pcmatern(mesh,
  prior.range = c(0.7, 0.5), # P(range < 0.7) = 0.5
  prior.sigma = c(1, 0.5)) # (P(sd > 1) = 0.5)
```

The priors are defined such that $P(\rho < 0.7) = 0.5$ and $P(\sigma > 1) = 0.5$. Now, additional data preparation is required to build the spatial model. First, a list of named index vectors need to be created. This index set contains information about the number of mesh points (`n.spde`) and the number of temporal replicates (`n.group`):

```
mesh.index = inla.spde.make.index(name = "field", n.spde = spde$n.spde, n.group = 40)
```

Second, a projection matrix needs to be defined using the coordinates of the observed data. Loosely speaking, this matrix is used to link the mesh nodes (`mesh`) to the observations' locations (`loc`), taking into account the temporal replicates (`group`):

```
A = inla.spde.make.A(mesh = mesh, loc = as.matrix(gpsim[,2:3]), group = gpsim$time)
```

What follows is justs a very complicated way to pass all this information to INLA. Instead of a data frame or a list, INLA uses stacks as data structures. This is because data frames or list aren't generic enough for the multiple options INLA offers. The stack is created as follows

```
stack = inla.stack(data = list(y = gpsim$y),
                   A = list(A, 1, 1),
                   effects = list(mesh.index,
```

```
                                  intercept = rep(1, nrow(gpsim)),
                                  covar = gpsim$x),
                    tag = "est")
```

With this we finalise one of the most tricky parts with INLA: getting the data in the right shape! Next, we define the model formula

```
formula = y ~ -1 + intercept + covar +
  f(field, model = spde, group = field.group, control.group = list(model = "iid"))
```

The spatial model is defined through the function f. The first argument is the name of the field (which we defined in `mesh.index`). Then, the type of non-linear model we want to fit (`spde`). The are different possible non-linear models in INLA, and you can investigate them doing `inla.list.models("latent")`. The argument `group` is needed because we have temporal replicates. Finally, we specify `model = "iid"` because we are assuming that these replicates are independent.

```
#Initial values of the hyperparameters (optional)
init2 = c(-1.3, -0.42, 0.62)
# Model fitting
# res.gp = inla(formula,
#   data = inla.stack.data(stack, spde = spde),
#   family ="gp",
#   control.mode = list(restart = TRUE, theta = init2),
#   control.family = list(control.link = list(quantile = 0.5)),
#   control.predictor = list(A = inla.stack.A(stack),compute = TRUE),
#   verbose = TRUE) # app 2mins
load('inlafit.Rdata')
```

An overall summary of the fit can be obtains using `summary(res.gp)`, but output is a bit messy, so we will see each output separately.

A summary of the fitted values (i.e., fitted GP median) can be obtained as follows

```
fitted = res.gp$summary.fitted.values
head(round(fitted,3))
```

```
##                          mean    sd 0.025quant 0.5quant 0.975quant  mode
## fitted.APredictor.0001 0.220 0.131      0.064    0.188      0.558 0.139
## fitted.APredictor.0002 1.655 0.788      0.616    1.494      3.626 1.218
## fitted.APredictor.0003 0.421 0.290      0.102    0.346      1.177 0.235
## fitted.APredictor.0004 1.156 0.663      0.353    1.003      2.853 0.755
## fitted.APredictor.0005 2.832 1.435      0.990    2.526      6.449 2.010
## fitted.APredictor.0006 0.064 0.034      0.022    0.057      0.149 0.044
```

```
tmp    = startsWith(rownames(res.gp$summary.fitted.values), "fitted.APredictor")
fitted = res.gp$summary.fitted.values[tmp,] # only at the 8000 space-time locations
head(round(fitted,3))
```

```
##                          mean    sd 0.025quant 0.5quant 0.975quant  mode
## fitted.APredictor.0001 0.220 0.131      0.064    0.188      0.558 0.139
## fitted.APredictor.0002 1.655 0.788      0.616    1.494      3.626 1.218
## fitted.APredictor.0003 0.421 0.290      0.102    0.346      1.177 0.235
## fitted.APredictor.0004 1.156 0.663      0.353    1.003      2.853 0.755
## fitted.APredictor.0005 2.832 1.435      0.990    2.526      6.449 2.010
## fitted.APredictor.0006 0.064 0.034      0.022    0.057      0.149 0.044
```

A summary of the fitted fixed effects can be obtained as follows

```
res.gp$summary.fixed
```

```
##               mean         sd 0.025quant 0.5quant 0.975quant     mode
```

```
## intercept 1.361859 0.17547573   1.017335 1.361853   1.706091 1.361855
## covar     1.997752 0.01461421   1.969059 1.997752   2.026419 1.997753
##                    kld
## intercept 6.072550e-07
## covar     4.794584e-07
```

Posterior estimates for the range and standard deviation of the spatial field, as well as for the tail parameter can be obtained doing

```
round(res.gp$summary.hyperpar, 3)
```

```
##                                          mean    sd 0.025quant 0.5quant
## Tail parameter for the gp observations 0.109 0.001      0.107    0.108
## Range for field                        0.658 0.002      0.656    0.658
## Stdev for field                        1.886 0.030      1.843    1.881
##                                        0.975quant  mode
## Tail parameter for the gp observations      0.111 0.108
## Range for field                             0.662 0.657
## Stdev for field                             1.955 1.861
```

The following lines produce a map of the posterior mean of the spatial effect for a single time point ($t = 1$):

```
library(ggplot2)
field = res.gp$summary.random[['field']][['mean']]
df    = data.frame(Longitude = mesh$loc[,1],
                   Latitude = mesh$loc[,2],
                   P.mean = field[1:length(mesh$loc[,2])])
ggplot(data = df, aes(x = Longitude, y = Latitude)) +
  geom_point(aes(colour = P.mean)) +
  scale_colour_gradientn(colours = viridis::viridis(10)) +
  theme_minimal() +
  ggtitle('Spatial effect') +
  theme(axis.text = element_text(size=12),
        axis.title=element_text(size=14),
        plot.title = element_text(size=22, hjust = 0.5))
```