Lecture 10: Applications, extensions, and software

Gaussian Markov random fields

David Bolin
Chalmers University of Technology
March 2, 2015

The plan for the day

Two topics were requested for todays lecture:

1. Excursions and computations of joint probabilities for GMRFs
2. Log-Gaussian Cox processes and GMRFs

The plan for the day is:

1. Introduce the R-INLA package
2. Give a detailed example of a LGCP model
3. Introduce the SPDE methods in INLA
4. Introduce the excursions method
5. Show a spatio-temporal application analysed using SPDEs, INLA, and excursions

We will see how far we actually get...

Latent Gaussian Models

Recall the Latent Gaussian model:

$$\begin{aligned} \boldsymbol{\theta} &\sim \pi(\boldsymbol{\theta}) \\ \boldsymbol{x} \mid \boldsymbol{\theta} &\sim \pi(\boldsymbol{x} \mid \boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\theta})) \\ \boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta} &\sim \prod_i \pi(y_i \mid \eta_i, \boldsymbol{\theta}) \end{aligned}$$

Note that we also assume that the observations are independent given the latent process.

$$\eta_i = g(\mu_i) = \alpha + \boldsymbol{z_i}^T \boldsymbol{\beta} + \sum_\gamma f_\gamma(c_{\gamma,i}) + \boldsymbol{u_i}, \quad i = 1, \ldots, n$$

$$\mathbf{x} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \{f_\gamma(\cdot)\}, \mathbf{u}\}$$

The R-INLA package

We will now look at the basics of how to do modeling and estimation for these LGMs using the R-INLA package.

For easy installation instructions, see

`http://www.r-inla.org/download`

There are essentially three parts to an INLA program:

1. The data organisation
2. The *formula*—notation inherited from R's native `glm` function
3. The call to the INLA program.

`formula`: Specifying the latent field

The latent field is specified using the "standard" `R` method `formula = y ~ 1 + covariate + f(...)`.

- y is the name of your data in the data frame.
- An intercept is fitted *automatically*! Use `-1` in your formula to avoid it.
- The fixed effects (covariates) are taken as i.i.d. normal with a common prior. (This can be changed)
- The `f` function contains the random effect specifications.

Specifying random effects

Random effects are added to the formula through the function

```
f(name, model="...", hyper = ...,
        replicate = ...,  constr =FALSE, cyclic = FALSE)
```

- `name`—the name of the random effect. Also refers to the values in data which are used for various things.
- `model`—the latent model. Eg. "iid", "rw2", "ar1", "bym" etc
- `hyper`—specify the prior on the hyperparameters
- `replicate`—for replicates (what else :p)
- `constr`—Sum to zero constraint?
- `cyclic`—Are you cyclic? (RW1, RW2 and AR1)

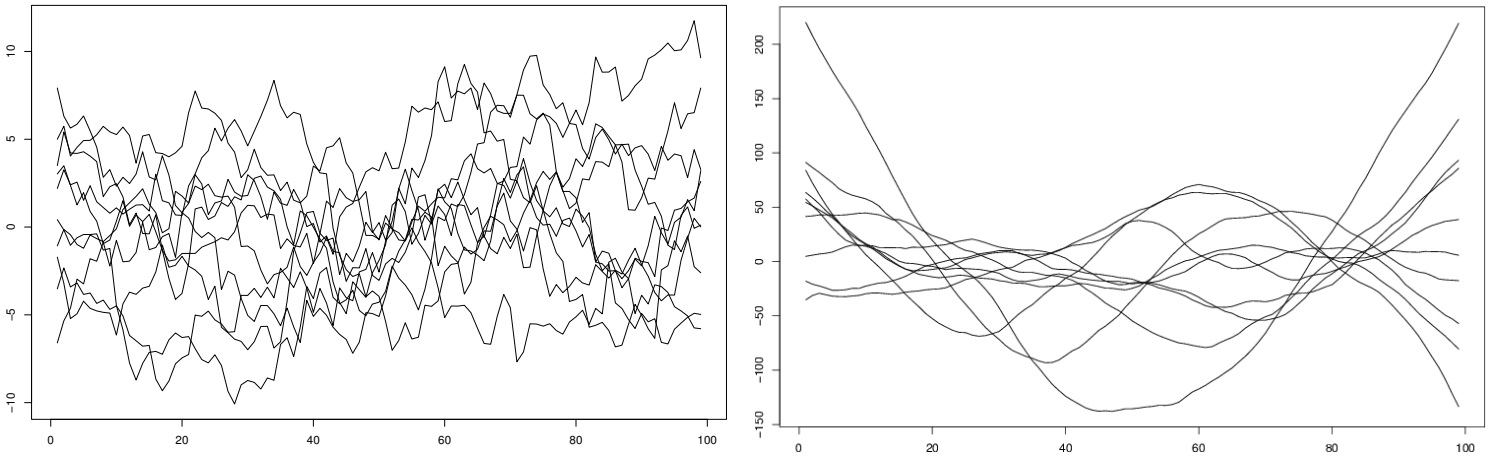
Random walk models

First-order random walk (RW1) models are specified as

```
formula = Y ~  ... + f(covariate, model="rw")
```

Second-order random walk (RW2) are specified as

```
formula = Y ~  ... + f(covariate, model="rw2")
```



The inla function

```
>  result <- inla(
       formula,   #This describes your latent field
       family = "gaussian", #The likelihood distribution.
       data = dat #A list or dataframe
       #This is all that's needed for a basic call

      verbose = TRUE, # I use this a lot!
       keep = FALSE, #Keeps the output

       #Then there are some "control statements"
       #that allow you to customise some things
       control.predictor=list(A = ObservationMatrix)
         )
```

Likelihood functions

For continuous data

- "gaussian"
- "T" (student-t)
- "sn" (Skew Normal)
- "gamma"

For count data

- "binomial"
- "Poisson"
- zero-inflated models (poission and binomial)

For survival data

- "coxph" (cox proportional hazard model)
- "Exponential"
- "weibull"

... And many others: go to `http://r-inla.org/`

Control statements

The `control.xxx` statements control various parts of the INLA program

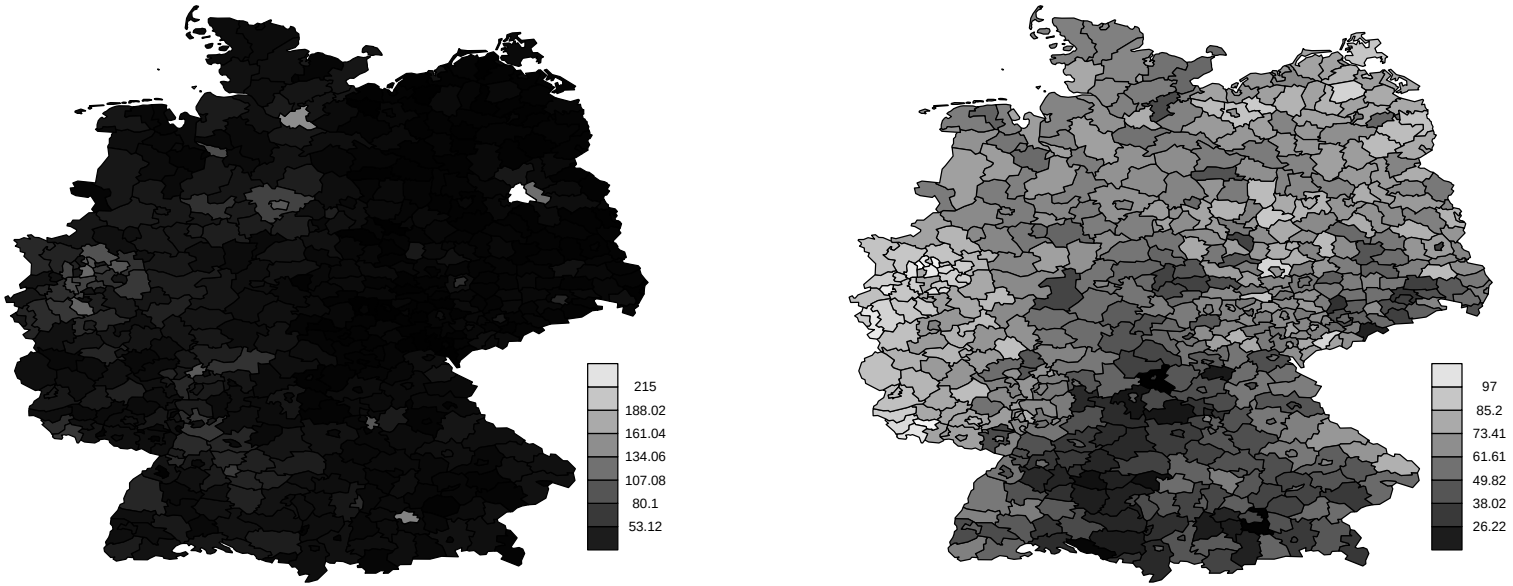
- `control.predictor`
  - `A` — The "A matrix" or "Observation Matrix" linking the latent field to the data.
- `control.mode`
  - `x`,`theta`, `result` — Gives modes to INLA.
  - `restart = TRUE` — Tells INLA to try to improve on the supplied mode
- `control.compute`
  - `dic`, `mlik`, `cpo` — Compute measures of fit.
- `control.inla`
  - `strategy` and `int.strategy` contain useful advanced features.

Various other—see help!

The simplest case: Linear regression

```
x = sort(runif(100))
y = 1 + x + rnorm(n, sd = 0.1)

formula = y ~ 1 + x
result = inla(formula,
        data = data.frame(x,y),
        family = "gaussian")

#prediction
x.pred = 2
xx = c(x, x.pred)
yy = c(y, NA)
formula.pred = yy ~ 1 + xx
result.pred = inla(formula.pred,
        data = data.frame(xx, yy),
        control.predictor = list(compute = T))
```

Example: Larnyx cancer mortality



- Larnyx cancer mortality rates are observed in the 544 districts of Germany from 1986 to 1990.
- Together with the counts, we also have the level of smoking consumption.

Larnyx cancer: model

- We assume the data to be conditionally independent Poisson random variables with mean $E_i \exp(\eta_i)$.
- E_i is fixed and accounts for demographic variation and η_i is the log-relative risk.
- The model for η_i takes the following form

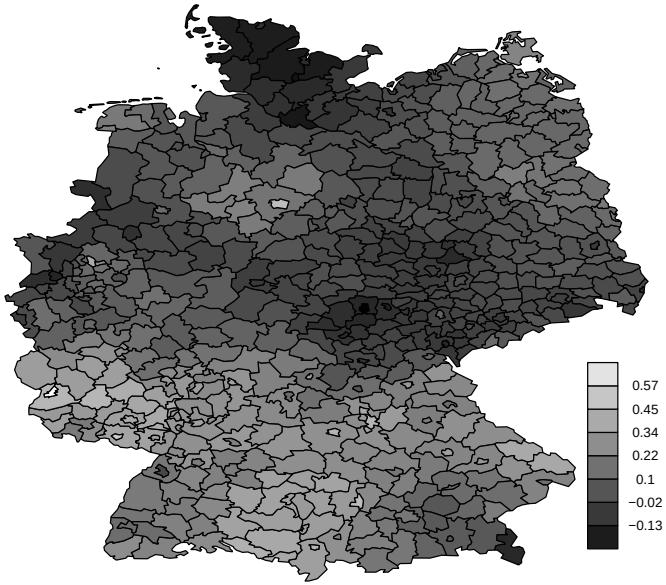
$$\eta_i = \mu + f(s_i) + f(c_i) + u_i$$

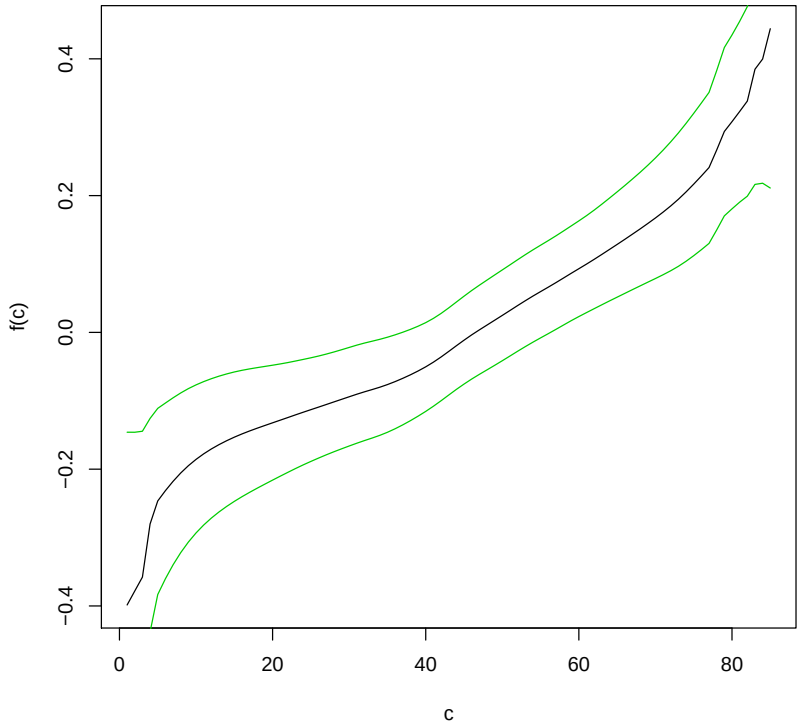
- μ is an intercept and $f(c)$ is a smooth effect of the smoking consumption c.
- $f(s)$ is a spatial effect, which we model as a Besag model, and u_i is an unstructured random effect.

Larnyx cancer: INLA call

```
data(Germany)
g = system.file("demodata/germany.graph", package="INLA")

Germany = cbind(Germany,region.struct=Germany$region)

formula = Y ~ f(region.struct,model="besag",graph=g) +
                f(region,model="iid") + f(x, model="rw2")

result =  inla(formula,family="poisson",data=Germany,E=E)

source(system.file("demodata/Bym-map.R", package="INLA"))
Bym.map(result$summary.random$region.struct$mean)
plot(result$summary.random$x$mean,type="l")
lines(result$summary.random$x$"0.025quant")
lines(result$summary.random$x$"0.975quant")
```

Larnyx cancer results





A more surprising example: Point processes

Spatial point processes model

- They focus on the random location at which events happen.
- They make excellent models for 'presence only' data when coupled with an appropriate observation process.
- Realistic models can be quite complicated.

Log-Gaussian Cox processes

The homogeneous Poisson process is often too restrictive. Generalizations include:

- inhomogeneous Poisson process - inhomogeneous intensity
- Markov point process - local interactions among individuals
- Cox process - random intensity

We focus on the Cox process, the random intensity depends on a Gaussian random field $Z(s)$:

$$\Lambda(s) = \exp(Z(s))$$

If Y denotes the set of observed locations, the likelihood is

$$\log(\pi(Y|\eta)) = |\Omega| - \int_\Omega \Lambda(s)\, ds + \sum_{s_i \in Y} \Lambda(s_i),$$

This is very different to the previous examples!

Or is it?

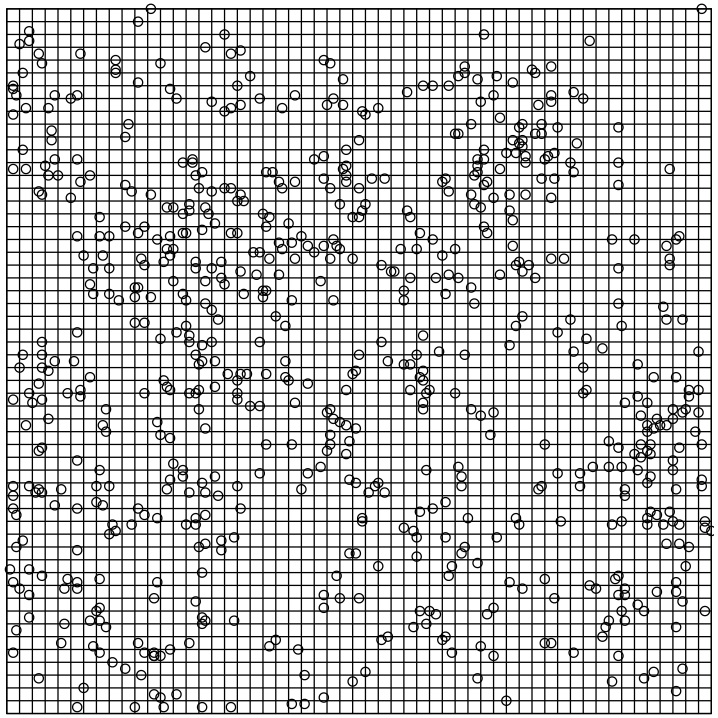
NB: *The number of points in a region R is Poisson distributed with mean $\int_R \Lambda(s)\, ds$.*

- Divide the 'observation window' into rectangles.
- Let y_i be the number of points in rectangle i.

$$y_i | x_i, \boldsymbol{\theta} \sim Po(e^{x_i}),$$

- The log-risk surface is replaced with

$$\mathbf{x} | \boldsymbol{\theta} \sim N(\boldsymbol{\mu}(\boldsymbol{\theta}), \boldsymbol{\Sigma}(\boldsymbol{\theta})).$$



The log-Gaussian Cox processes can be approximated by a LGM!

A LGCP model for Australian plants

- The original data set comprises the locations of 67 different species collected in an area of dimension $22\text{m} \times 22\text{m}$ plot.
- Here we focus on one of the plant species, Andersonia heterophylla, an erect or ascending shrub, $0.1 - 0.5$ m in height, grows in sandy, very nutrient poor soils.
- The model structure of this simple example is the basis of all other more complicated point process models INLA can handle

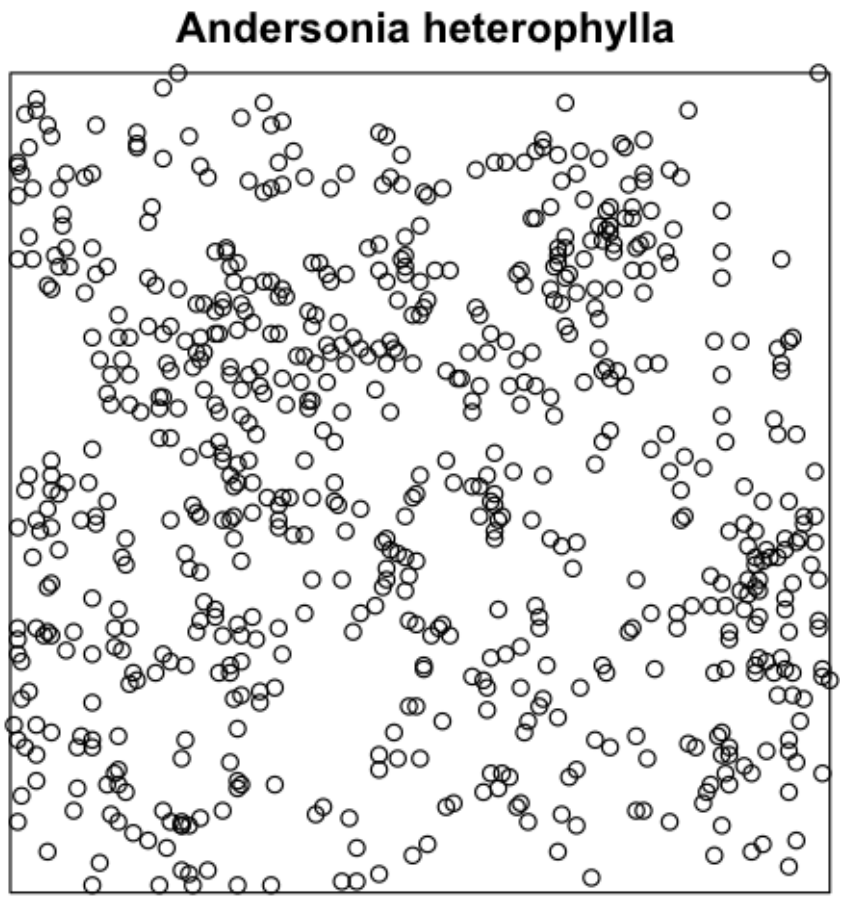
Reading in and gridding the data

```
paul<- read.delim("paul.txt")
# type 5 is Andersonia heterophylla
data<-paul[paul$type=="5",]
x=data$x/10
y=data$y/10
```

We transform the data into a point pattern object (using commands from the library spatstat).

```
x.area=22
x.win=owin(c(0, x.area),c(0, x.area))
data.pp=ppp(x,y,window=x.win)
```

Reading in and gridding the data



Plot the data

```
plot(data.pp, main= " Andersonia heterophylla")
```

Comments

- The plot reveals that the pattern likely is inhomogeneous.
- This might have been caused by varying soil conditions or by how the species is spreading.
- Clearly, if covariate data were available we would want to include these in a model. However, here no covariates are available.
- We therefore fit a model that takes into account the spatial large scale variability of the pattern, assuming that this is the result of an underlying environmental trend even in the absence of data on this.
- The LGCP approach fits a smooth surface to the data that reflects this trend without making any assumptions on the parametric form of the trend.

Gridding the data

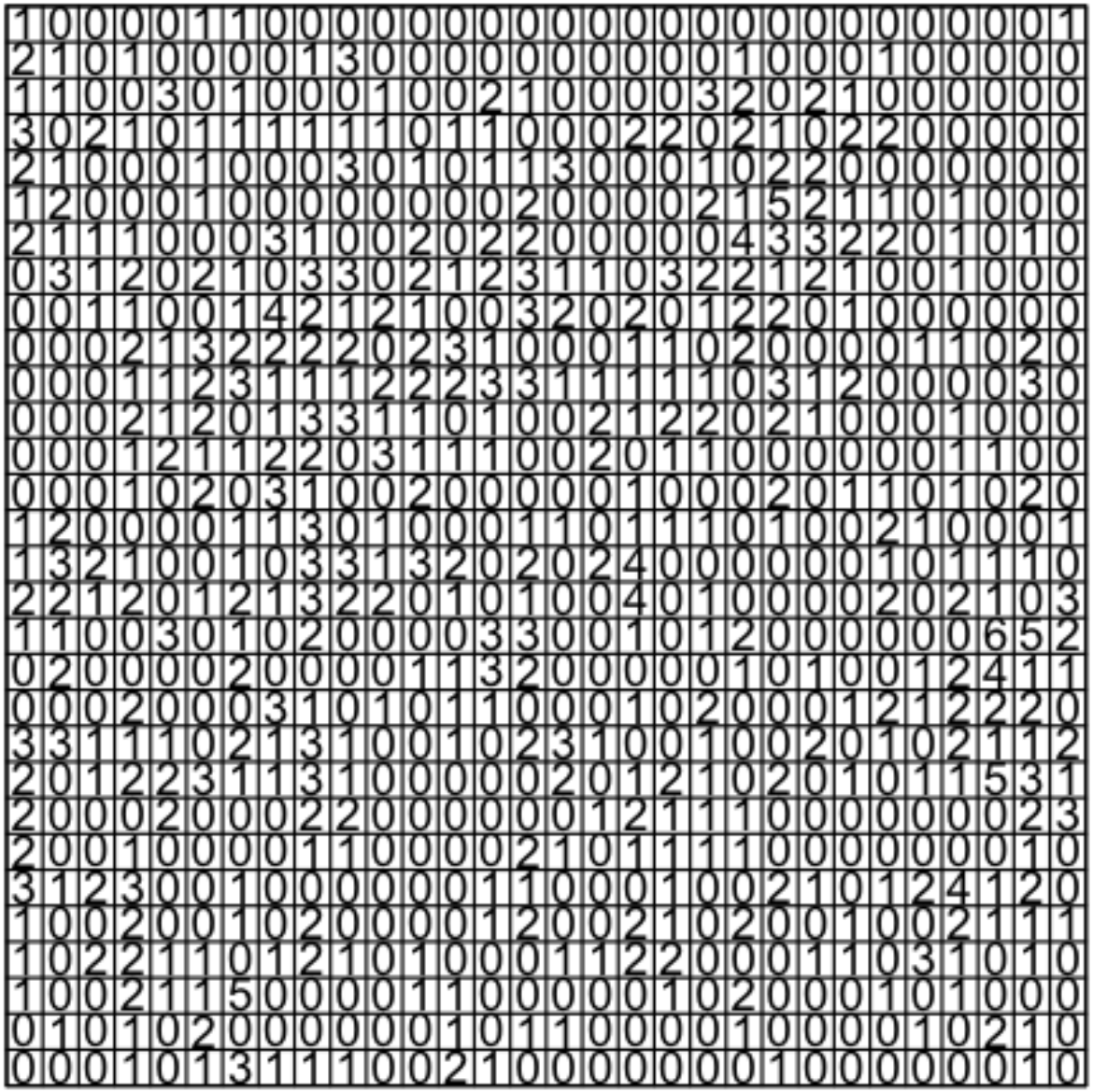
In order to estimate the model, we divide the plot into grid cells and use the fact that the number of points in each grid cell is poisson distributed. Thus, we first transform the data to a grid with 30 x 30 cells

```
nrow=30
ncol=nrow
x.grid=quadrats(x.win,nrow,ncol)
```

and count the number of points in each grid cell; note that this will be our response variable.

```
count.grid=quadratcount(data.pp, tess=x.grid)
plot(count.grid)
Y = as.vector(count.grid)
```

count.grid


1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
2 1 0 1 1 0 0 0 0 1 3 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0
1 1 0 0 3 0 1 1 0 0 0 1 1 0 0 2 1 1 0 0 0 0 3 2 0 2 1 0 0 0 0 0
3 0 2 1 1 0 1 1 1 1 1 1 1 0 1 1 0 0 0 2 2 0 2 1 1 0 2 2 0 0 0 0
2 1 0 0 0 1 1 0 0 0 3 0 1 1 0 1 1 3 0 0 0 1 0 2 2 0 0 0 0 0 0 0
1 2 0 0 0 1 1 0 0 0 0 0 0 0 0 0 2 0 0 0 0 2 1 5 2 1 1 0 1 0 0 0
2 1 1 1 1 0 0 0 3 1 1 0 0 2 0 2 2 0 0 0 0 0 0 4 3 3 2 2 0 1 1 0 1 0
0 3 1 1 2 0 2 1 1 0 3 3 0 2 1 1 2 3 1 1 0 3 2 2 1 1 2 1 1 0 0 1 1 0 0 0
0 0 1 1 1 0 0 1 1 4 2 1 1 2 1 1 0 0 3 2 1 0 2 0 1 1 2 2 0 1 1 0 0 0 0 0 0
0 0 0 0 2 1 1 3 2 2 2 2 2 0 2 2 3 1 1 0 0 0 1 1 1 0 2 1 0 0 0 0 1 1 1 0 2 1 0
0 0 0 1 1 1 2 3 1 1 1 1 2 2 2 2 3 3 1 1 1 1 1 1 0 3 1 1 2 1 0 0 0 0 3 1 0
0 0 0 0 2 1 1 2 0 1 1 3 3 1 1 1 0 1 1 0 0 2 1 1 2 2 2 0 2 1 1 0 0 0 1 1 0 0 0
0 0 0 0 1 1 2 1 1 1 2 2 2 0 3 1 1 1 1 0 0 2 1 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0
0 0 0 0 1 1 0 2 1 0 3 1 1 0 0 2 1 1 0 0 0 0 0 0 1 1 0 0 0 2 1 1 1 0 1 1 0 2 0
1 1 2 1 0 0 0 0 1 1 1 3 1 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 0 1 1 0 0 2 1 1 0 0 0 1
1 1 3 2 1 1 0 0 1 1 0 3 3 1 1 3 2 1 0 2 0 2 4 1 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0
2 2 2 1 2 1 0 1 2 1 1 3 2 2 2 1 1 0 1 1 0 0 4 1 1 0 0 0 0 2 1 0 2 1 1 0 3
1 1 1 0 0 3 1 0 1 0 2 1 1 0 0 0 0 3 3 1 1 0 0 1 1 0 1 2 1 0 0 0 0 0 0 0 0 6 5 2
0 1 2 1 0 0 0 0 0 2 1 1 0 0 0 0 1 1 1 3 2 1 1 0 0 0 0 0 1 1 0 1 1 1 0 0 1 2 4 1 1
0 0 0 0 2 1 0 0 0 3 1 1 0 1 1 0 1 1 1 1 0 0 0 1 1 0 2 1 0 0 0 1 2 1 1 2 2 2 0
3 3 1 1 1 1 0 2 1 1 3 1 1 0 0 1 1 0 2 3 1 1 0 0 1 1 0 0 2 1 1 0 2 1 1 1 2
2 1 0 1 2 2 3 1 1 1 3 1 1 1 0 0 0 0 0 2 1 1 2 1 1 0 2 1 1 1 0 1 1 1 5 3 1
2 1 0 0 0 2 1 0 0 0 2 2 1 1 0 0 0 0 0 0 1 1 2 1 1 1 1 1 0 0 0 0 0 0 0 0 2 3
2 1 0 0 1 1 0 0 0 0 1 1 1 1 0 0 0 0 2 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0
3 1 1 2 3 1 0 0 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 2 1 1 0 1 1 2 4 1 1 2 1 0
1 1 0 0 2 1 0 0 1 1 0 2 1 1 0 0 0 0 1 1 2 1 1 0 0 2 1 1 1 0 2 1 1 0 0 1 1 1 0 0 2 1 1 1
1 1 0 2 2 1 1 1 0 1 1 2 1 1 1 0 1 1 0 0 0 1 1 1 2 2 1 1 1 0 0 1 1 1 0 3 1 1 1 0 1 1 0
1 1 0 0 2 1 1 1 5 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 0 2 1 1 1 0 0 0 1 1 1 0 1 1 0 0 0
0 1 1 0 1 1 0 2 1 1 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 2 1 1 0
0 0 0 1 1 0 1 1 3 1 1 1 1 1 0 0 2 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0

Gridding the data (II)

The mean of the poisson distribution at a cell s_{ij} is given by the integral of the intensity over that cell,

$$\Lambda_{ij} = \int_{s_{ij}} \exp(Z(s))ds$$

It is difficult to calculate these integrals exactly, so we approximate

$$\Lambda_{ij} \approx E_{ij} \exp(Z_{ij}),$$

where E_{ij} is the area of the cell and Z_{ij} is a "representative value" of the latent field $Z(s)$ within the cell.
Thus, we need to calculate the area of each grid cell.

```
n = ncol*nrow
cell.area<-x.area^2/n
E<-rep(cell.area, n)
```

Specifying the latent field

We can now specify the latent field model,

$$Z(s) = \mu + f_1(s)$$

where μ is an intercept, $f_1(s)$ is a Gaussian random field.

This is specified using a formula as

```
I = 1:n
formula = Y ~ 1+f(I, model="rw2d", nrow=nrow, ncol=ncol)
```

here, rw2d is the ICAR(2) model (see Lecture 5 and section 3.4.2 in the course book).

Estimation and results

Run the model (this should take only a few seconds at most)

```
result=inla(formula,data=data.frame(Y,I),
            family="poisson",E=E)
```

Look at a summary of the results

```
summary(result)
```

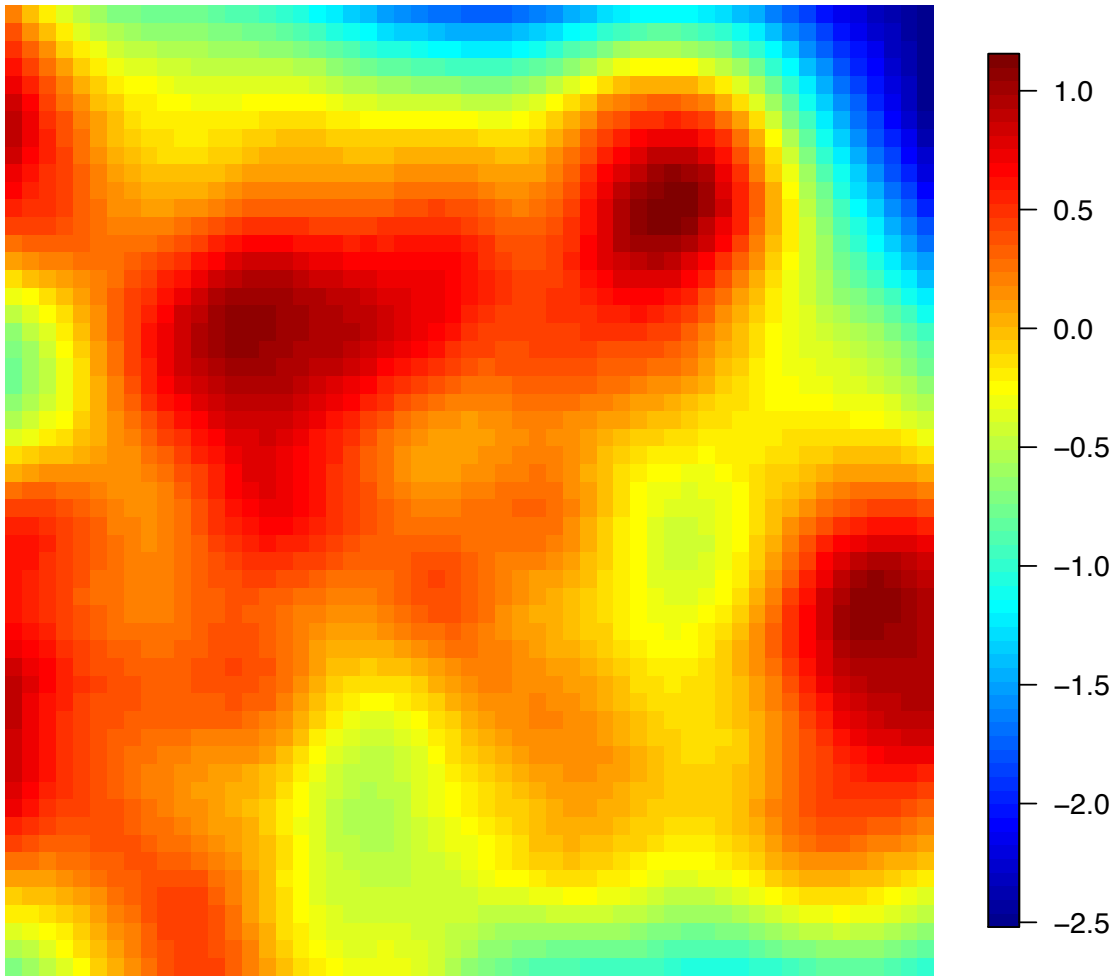
the (posterior mean of the) spatial effect

```
f.spat=result$summary.random$I$mean
```

plot it

```
inla.display.matrix(matrix(f.spat, nrow, ncol))
```

Posterior mean of the spatial effect



Constructed covariates

For mechanistic reasons, it is often interesting to model inter-point correlations directly in point process models:

- Clustering
- Hierarchical clustering
- Repulsion

There are a number of classical models with these structures (Thomas processes, Matérn Hard Core proceses, Strauss processes), but they are *really hard to fit*.

Can we model inter-point correlations in LGCPs?

No.

- but we can add covariates!
- Idea: Construct some 'fake' covariates from the data and include them in the model.
- Example: Distance (on a lattice)

$$d(s_{ij}) = \min_{p \in Y \setminus s_{ij}} \|c_{ij} - p\|$$

  This is the distance to the closest point to the centre of cell s_{ij} that is not *in* s_{ij}.
- You fit a random effects of these. See if it makes a difference.

Example: Effect of a second species

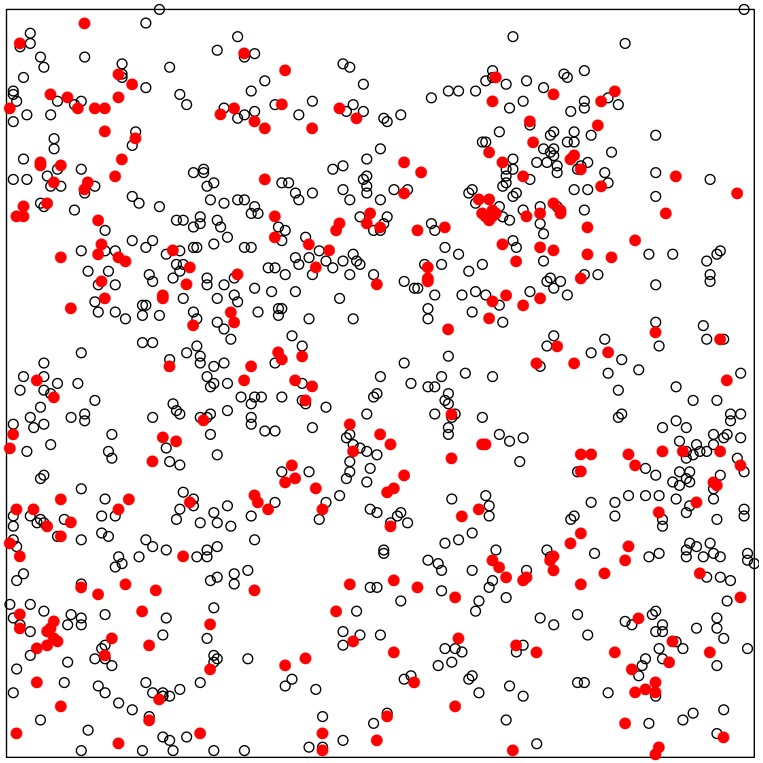
We want to know if the nearness of a second species of tree affects the probability of a tree existing at a location.

- Poor man's multivariate modelling!
- Construct the covariate

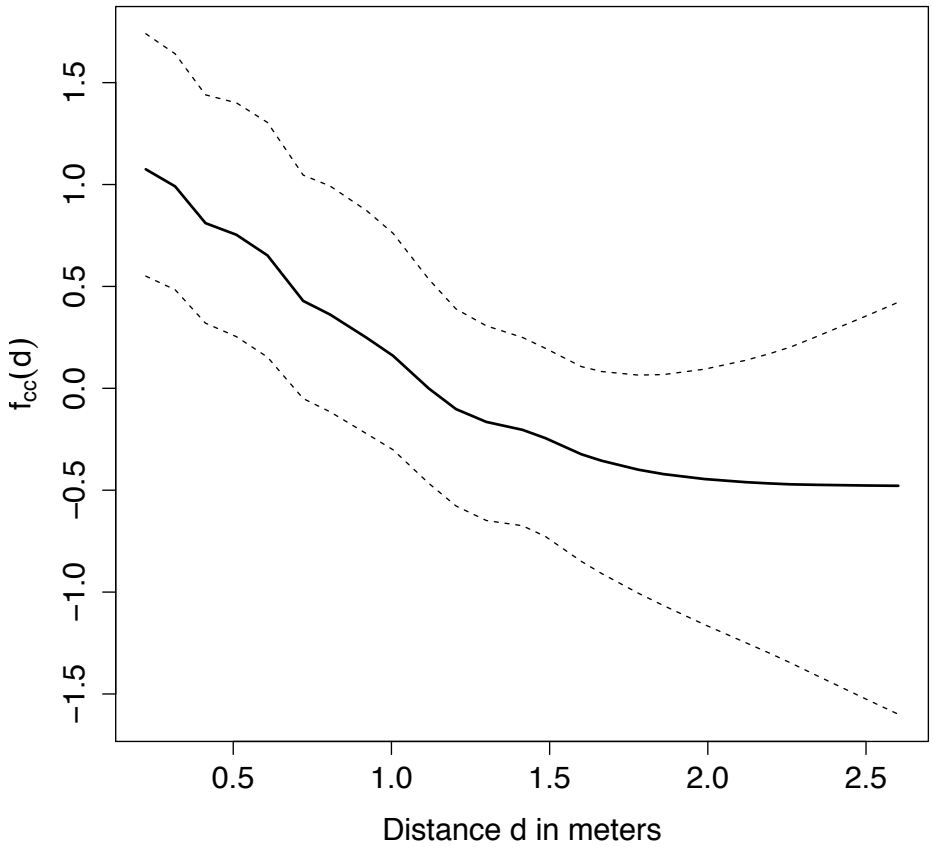
$$d_{12}(s_{ij}) = \min_{p \in Y_2 \setminus s_{ij}} \|c_{ij} - p\|,$$

  where Y_2 is the *other* species.
- Fit the model
  $\eta_{ij}^{(1)} = \mu + f_1(d_1(s_{ij})) + f_2(d_{12}(s_{ij})) + f_s(s_{ij}) + u_{ij}$ where $f_s(\cdot)$ is a spatial effect and u_{ij} is white noise.

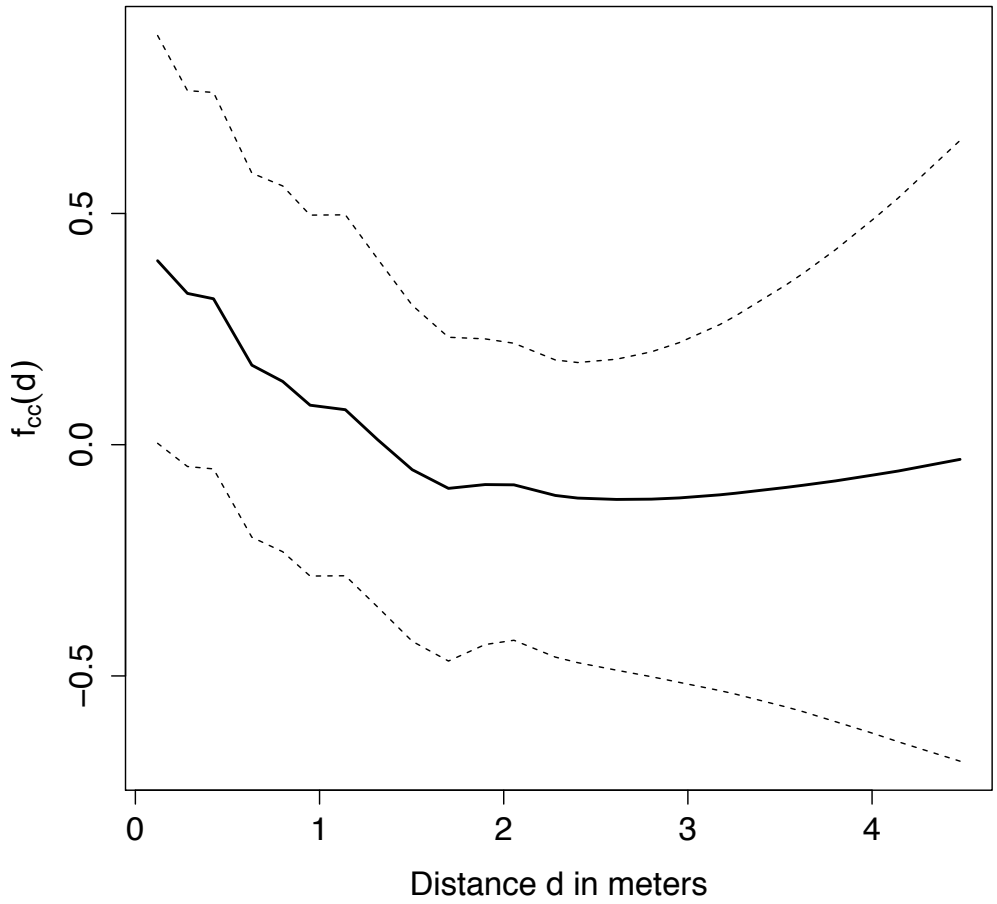
Andersonia heterophylla (689 plants) and Conospermum crassinervium (266 plants)



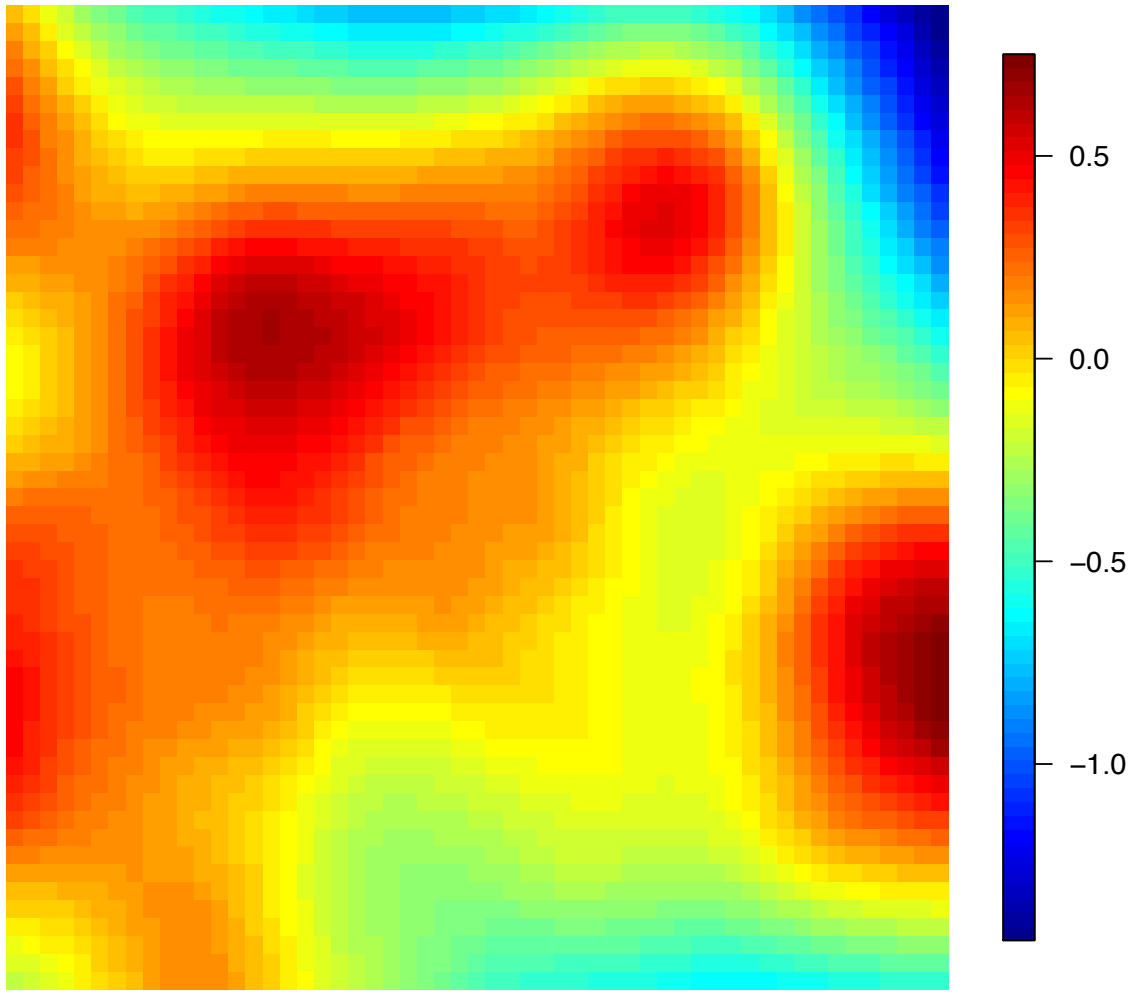
Constructed covariate 1: Intra-species interaction



Constructed covariate 2: Inter-species interaction



Spatial effect with inter-species term



SPDE models in INLA

The SPDE models have been incorporated into the INLA package.

These are specified in INLA as any other random effect in the formula

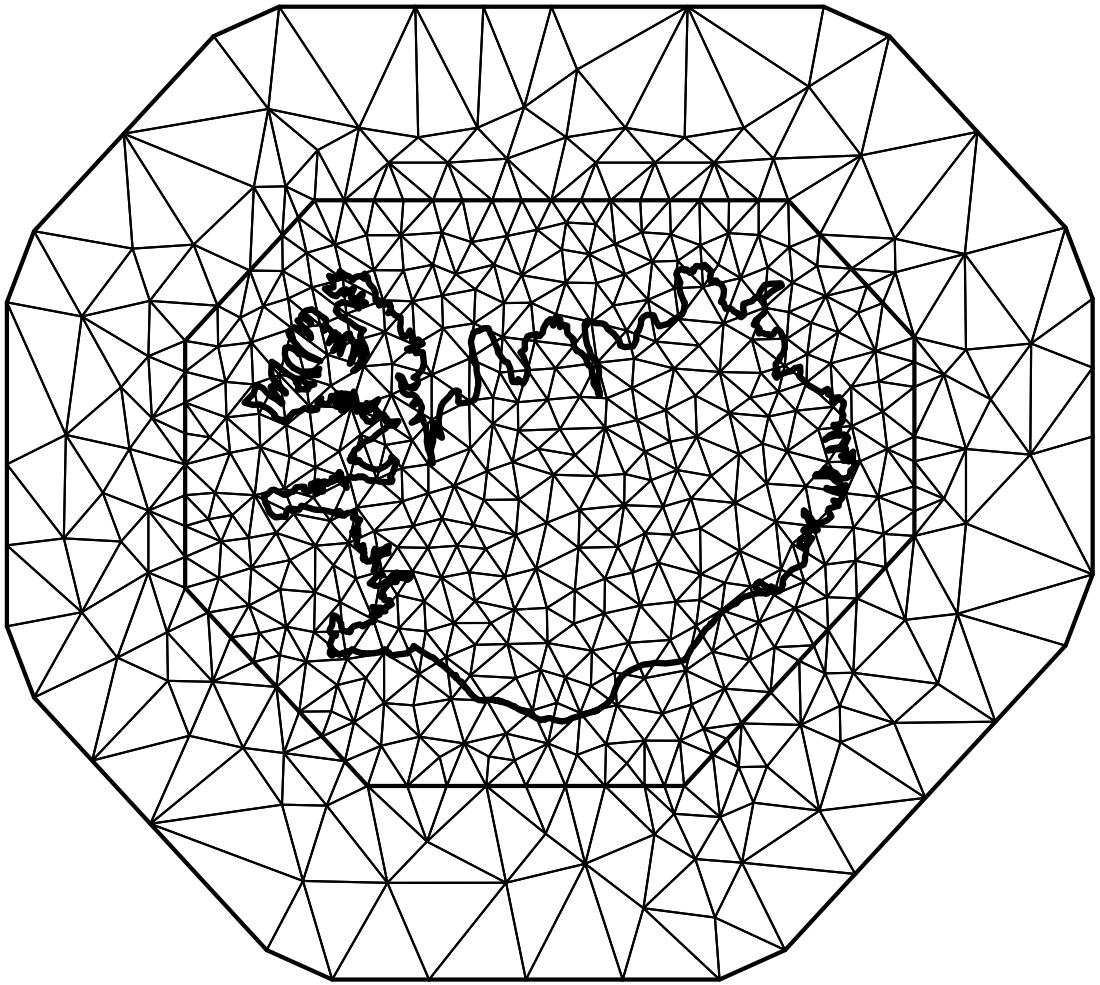
However, we also need to specify the mesh and observation matrix for the piecewise linear basis functions when defining the model

The simplest way to create a mesh in INLA is to use the function `inla.mesh.2d`.

- Creates a mesh with two regions: the interior mesh, which is where the action happens; and the exterior mesh, which is designed to alleviate the boundary effects.
- See the help text for details!

Typical use

```
mesh <- inla.mesh.2d(points.domain=iceland_ISN,
                     max.edge=c(40,800),
                     offset=c(50,150),
                     min.angle=25)
```



Making observation matrices

When the observations don't occur at mesh points, we need some way to map between the latent field and the observation process.

`inla.spde.make.A(mesh,loc)` constructs the matrix $A_{ij} = \varphi_j(s_i)$ that maps a field defined on the mesh to the observation locations s_i.

- See help text for the full list of (optional) arguments.
- The function will also automatically deal with space-time models and replicates.
- A related function (inla.mesh.projector) builds an A-matrix for projecting onto a lattice. This is useful for plotting.

Constructing SPDE models

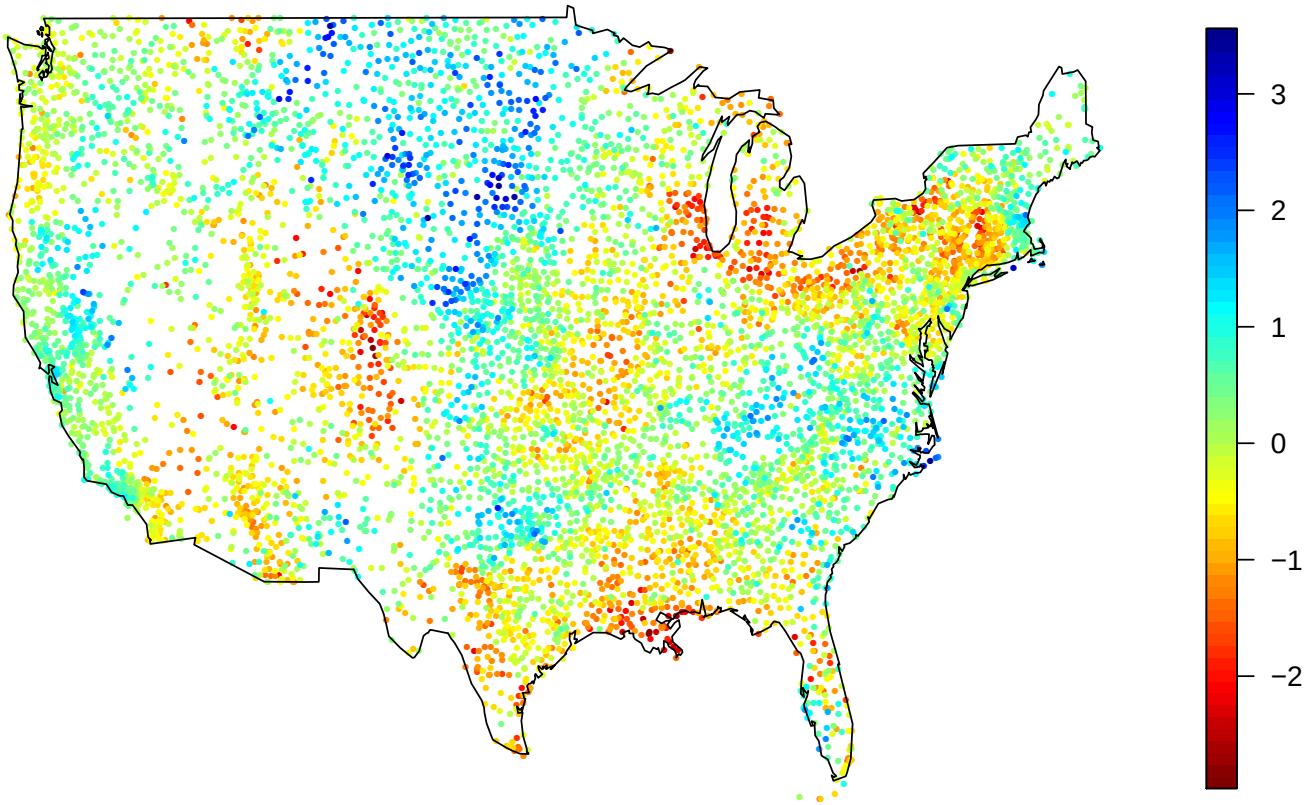
For historical reasons there are two different SPDE classes (`spde1` and `spde2`)

- `spde1` is the "classic" SPDE model!
- The `spde2` class is more flexible and defines non-stationarity in a more natural way.
- The primary difference between the two models is in the prior specification.
- For "stationary" models, these are fairly much the same (up to prior specification)

The standard `spde2`-call:

```
inla.spde2.matern(mesh, alpha)
```

Kriging of precipitation anomalies



The data are yearly precipitation anomalies from 1962 at US weather stations. This means they are standardized with respect to the long run mean and standard deviation of each station.

`http://www.image.ucar.edu/Data/precip_tapering/`

Code: load the data and crate the mesh

```
require(INLA)
require(maps)
require(fields)

load("anom1962.RData")

quilt.plot(loc,z,ny=200, nx=250)
world( add=TRUE, lwd=2)

mesh <- inla.mesh.2d(loc = loc, max.edge=c(1,10),
                     cutoff=0.7)
plot(mesh)
world( add=TRUE, lwd=2)
```

Code: construct and estimate the model

```
#construct A matrix
A <- inla.spde.make.A(mesh, loc=loc)

#construct SPDE model
spde <- inla.spde2.matern(mesh, alpha=2)

#Organise the data
data = list(z=z, loc = 1:spde$n.spde)

#make a formula
formula = z ~ -1 + f(loc, model =spde)

#do the inference
result = inla(formula,family = "gaussian", data = data,
         control.predictor = list(A = A), verbose= TRUE)
```

Code: plot the results

```
proj <- inla.mesh.projector(mesh,dims=c(200,200))
mean.plot = inla.mesh.project(proj,
          result$summary.random$loc[,"mean"])
sd.plot = inla.mesh.project(proj,
          result$summary.random$loc[,"sd"])

levelplotmap = function(..., mm) {
    panel.levelplot(...)
    panel.lines(mm$x, mm$y, col="black")
}
levelplot(row.values=proj$x, column.values=proj$y,
          x=mean.plot, mm=map("usa"), panel=levelplotmap,
          col.regions=tim.colors(100),
          xlab="Easting",ylab="Northing",main="mean")
```

A non-stationary model with $\kappa = 2$, $\tau(x, y) = 0.1 + 3x$

Create a mesh:

```
mesh=inla.mesh.2d(loc.domain=cbind(c(0,0,1,1),c(0,1,0,1)),
                  max.edge=0.05)
```

Construct basis functions:

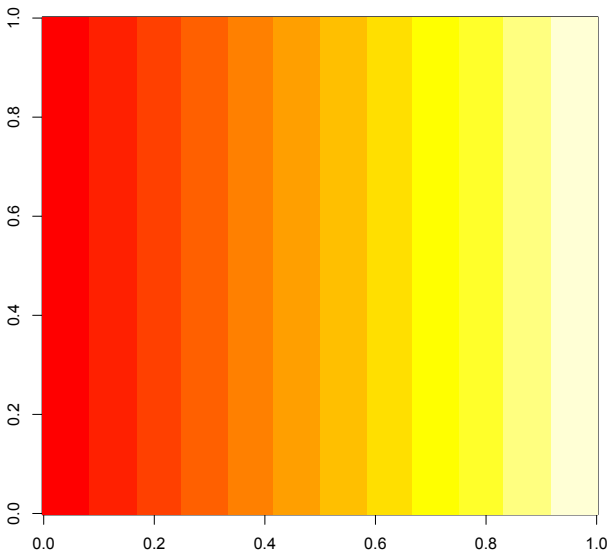
```
one =rep(1,mesh$n)
basis.fn = 3*mesh$loc[,1]
B.kappa=matrix(c(0*one,2*one),nrow=mesh$n,ncol=2)
B.tau = matrix(c(0.1*one,basis.fn),nrow=mesh$n,ncol=2)
```

Create model and sample:

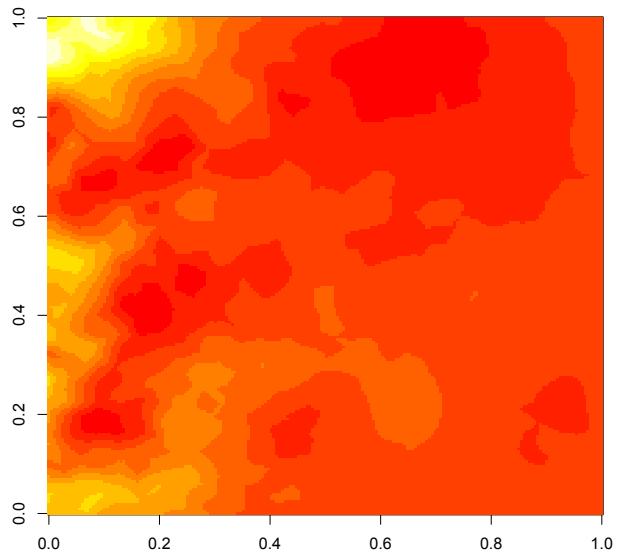
```
spde=inla.spde2.matern(mesh,B.tau=B.tau,B.kappa=B.kappa)
Q= inla.spde2.precision(spde,theta=1)
sample = as.vector(inla.qsample(1,Q))
```

Plot the results:

```
proj = inla.mesh.projector(mesh, xlim=c(0,1),
                           ylim=c(0,1), dims=c(200,200))

image(inla.mesh.project(proj,basis.fn))
image(inla.mesh.project(proj,sample))
```



(a) Basis function



(b) Sample

Organising data

Real life is hard!

- In complicated models, we may have multiple sources of data occurring in different places with different likelihoods.
- The latent field may also be composed of sections defined at different resolutions (grid for a spatial covariate, mesh for random field, etc).
- So we need a function that takes these components and chains them together in a way that makes sense.

We are rescued by `inla.stack`!

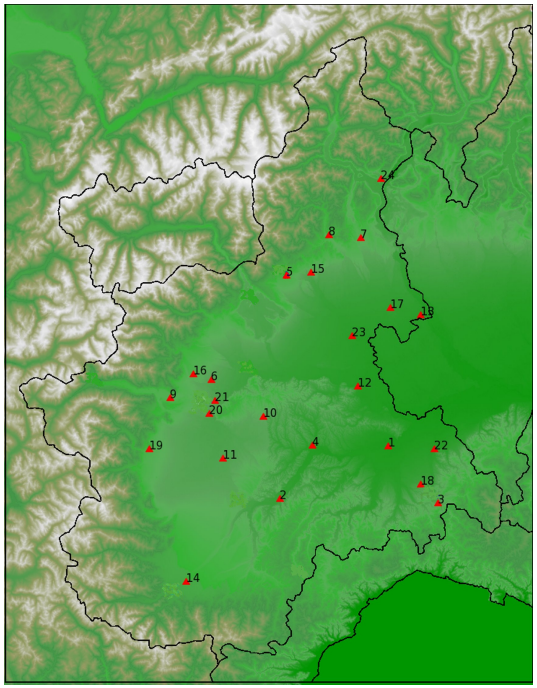
The inla.stack call

```
stack = inla.stack(data = list(...),
                   A = list(...),
                   effects = list(...),
                   tag = NULL, ...)
```

- The trick here is lists!
- The first element of the `effects` list is mapped to the first element of the `data` list by the first element for the `A` list.
- The functions `inla.stack.data(stack)` and `inla.stack.A(stack)` are used to extract the data.frame and the A-matrix for use in the `inla` call.

PM-10 concentration in Piemonte, Italy

As a final example, we look at a more complicated space-time model. This example is described in Cameletti *et al.* (2012).



PM10 concentration:

- 24 monitoring stations
- Daily data from 10/05 to 03/06

Covariates

- Daily mean wind speed (WS, m/s)
- Daily maximum mixing height (HMIX, m)
- Daily precipitation (P, mm)
- Daily mean temperature (TEMP, K°)
- Daily emissions (EMI, g/s)
- Altitude (A, m) Coordinates (UTMX and UTMY, km).

Model

- The following measurement equation is assumed,

$$y(\mathbf{s}_i, t) = x(\mathbf{s}_i, t) + \mathcal{E}(\mathbf{s}_i, t),$$

  where $\mathcal{E}(\mathbf{s}_i, t) \sim \mathsf{N}(0, \sigma^2_{\mathcal{E}})$ is Gaussian measurement noise, both spatially and temporally uncorrelated.
- $x(\mathbf{s}_i, t)$ is the latent field assumed to be on the form

$$x(\mathbf{s}_i, t) = \sum_{k=1}^{p} z_k(\mathbf{s}_i, t)\beta_k + \xi(\mathbf{s}_i, t),$$

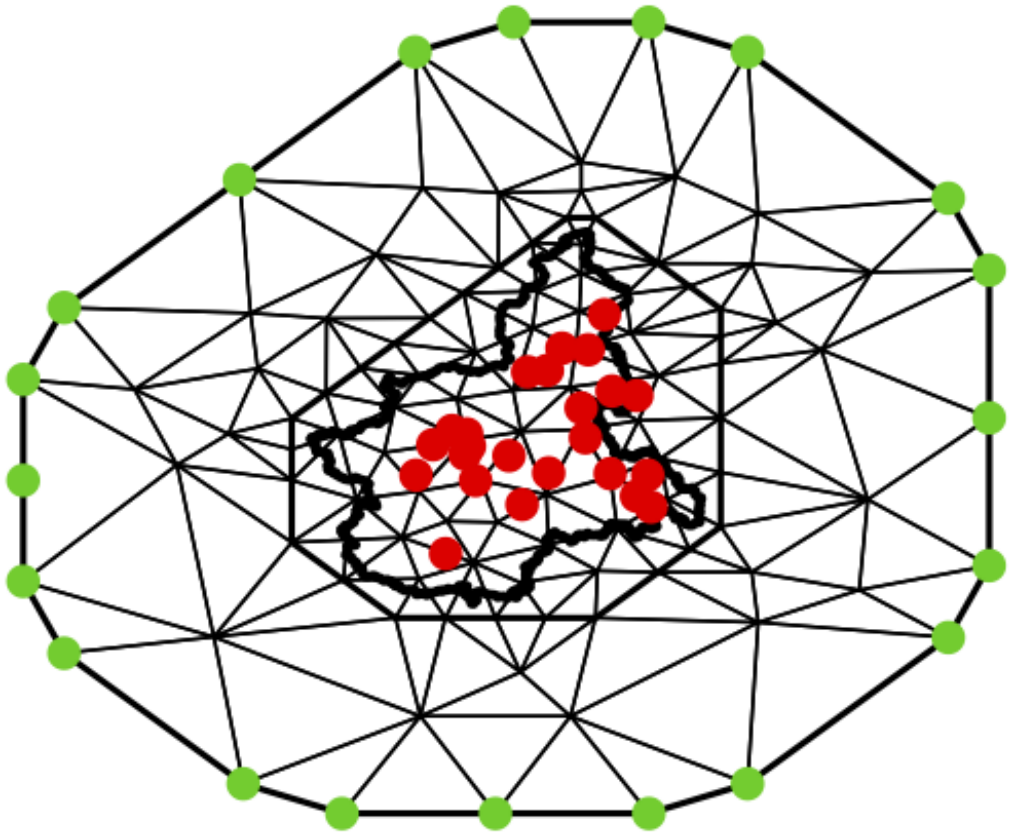
  where the $p = 9$ covariates z_k are used.
- ξ is assumed to follow first order AR-dynamics in time

$$\xi(\mathbf{s}_i, t) = a\xi(\mathbf{s}_i, t-1) + \omega(\mathbf{s}_i, t),$$

  where $|a| < 1$ and $\omega(\mathbf{s}_i, t)$ is a zero-mean temporally independent Gaussian process with spatial Matérn covariances.

Step 1: Make the mesh

```
mesh = inla.mesh.2d(loc=cbind(coordinates$UTMX,
                              coordinates$UTMY),
                    loc.domain=borders, offset=c(10, 140),
                    max.edge=c(50, 1000), min.angle=c(26, 21),
                    cutoff=0)
```



Step 2: Make the latent model

In order to construct the space-time model in INLA, we use the `group` feature

```
spde = inla.spde2.matern(mesh=mesh, alpha=2)

formula <- logPM10 ~ -1 + Intercept + A + UTMX + UTMY
                     + WS + TEMP + HMIX + PREC + EMI
                     + f(field, model=spde,
                         group=field.group,
                         control.group=list(model="ar1"))
```

- This tells INLA that the observations are grouped in a certain way.
- `control.group` contains the grouping model as well as their prior specifications.

Step 3: Make an A matrix

```
loc = as.matrix(coordinates[Piemonte_data$Station.ID,
                            c("UTMX","UTMY")])
A.est = inla.spde.make.A(mesh,
                         loc= loc,
                         group=Piemonte_data$time,
                         n.group=n_days)
```

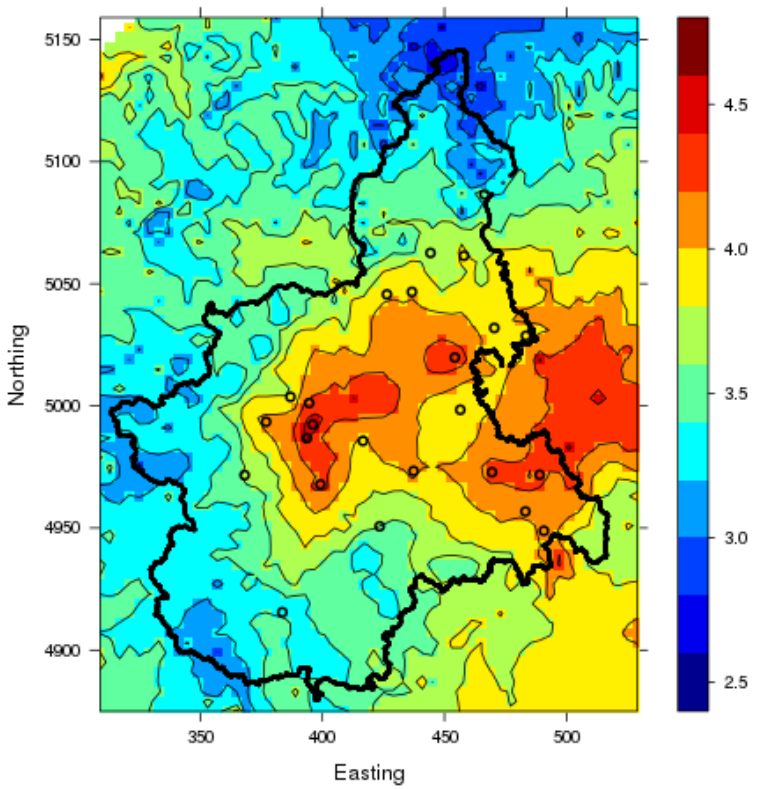
This locates the data points in each `group` level and stacks the corresponding local A matrices in an appropriate way.

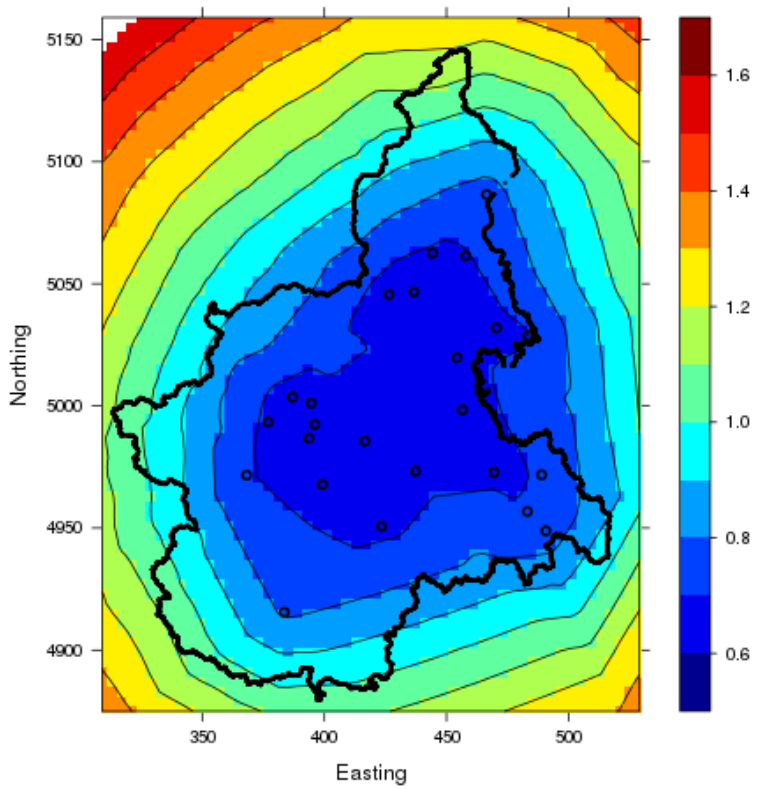
Step 4: Organising the data and estimating the model

We can now put everything together and make sure that $\boldsymbol{A}$ only applies to the random effect by using the `inla.stack` function.

```
stack = inla.stack(data=list(
                        logPM10=Piemonte_data$logPM10),
                        A=list(A.est, 1),
                        effects= list(
                                  c(field.indices,
                                   list(Intercept=1)),
                                   list(Piemonte_data[,3:10]))
                      tag="est")

result = inla(formula, family = "gaussian",
       data=inla.stack.data(stack),
       control.predictor = list(A=inla.stack.A(stack)))
```

Posterior mean PM10 concentration for 30/01/2006 (log scale)





What are we actually interested in analysing here?

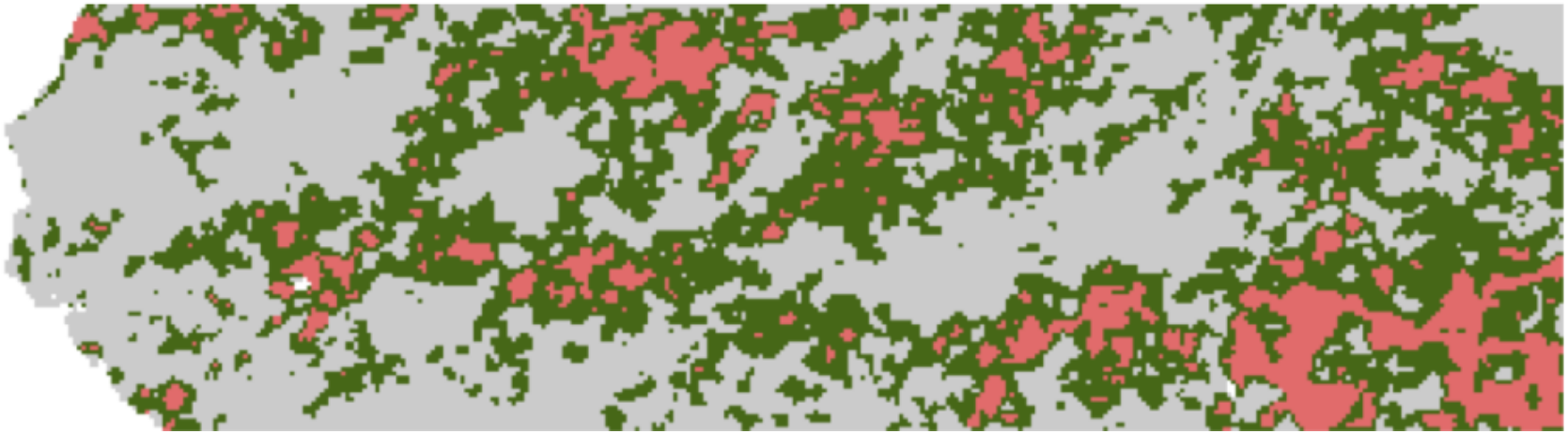
- The limit value fixed by the European directive 2008/50/EC for PM_{10} is $50\mu g/m^3$. The daily mean concentration cannot exceed this value more than 35 days in a year.
- A region where this value is periodically exceeded is the Piemonte region in northern Italy.
- The goal is to analyse exceedance probabilities of the limit value.

Thus, we are here not interested only in the marginal posteriors for the latent field!

This scenario is quite common!

Proof by a few examples.

Vegetation in Sahel: Where has it increased?



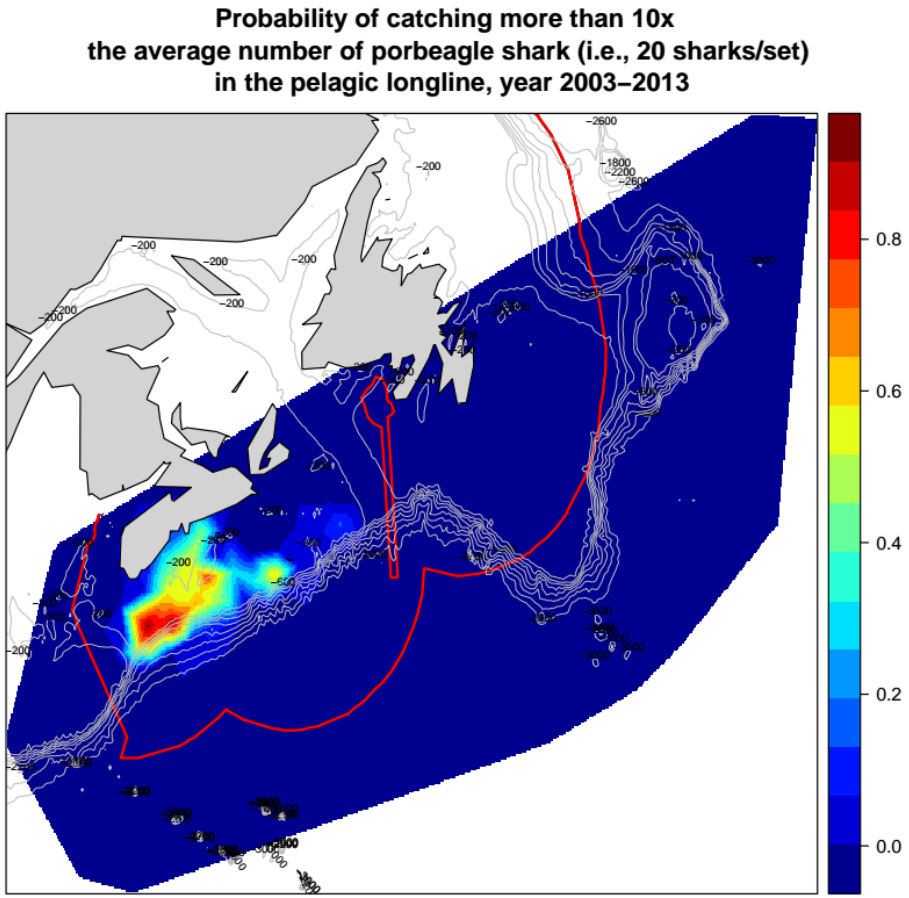
- Has the vegetation increased in the Sahel region since the severe droughts in the early 80s?
- If so, where and how much?
- Estimates of significant trends in vegetation in the western Sahel for the period 1983 - 1999.

Joint work with Lindström (Lund), Eklundh (Lund), Lindgren (Bath). CSDA (2009) and JRSSB (2015)

fMRI studies: which regions in the brain are active?

Joint work with Yue (CUNY), Lindquist (Johns Hopkins), Lindgren (Bath), Simpson (Warwick), and Rue (NTNU).

Bycatch hotspots: where do we accidentally catch sharks?



Joint work with Godin (Dalhousie), Krainski (NTNU), Worm (Dalhousie), Flemming (Dalhousie), and Campana (Bedford Inst of Oceanography).

What are we interested in? Excursion sets!

Excursion sets

Let $x(s)$, $s \in \Omega$ be a random process. The positive and negative level u excursion sets with probability $1-\alpha$ are

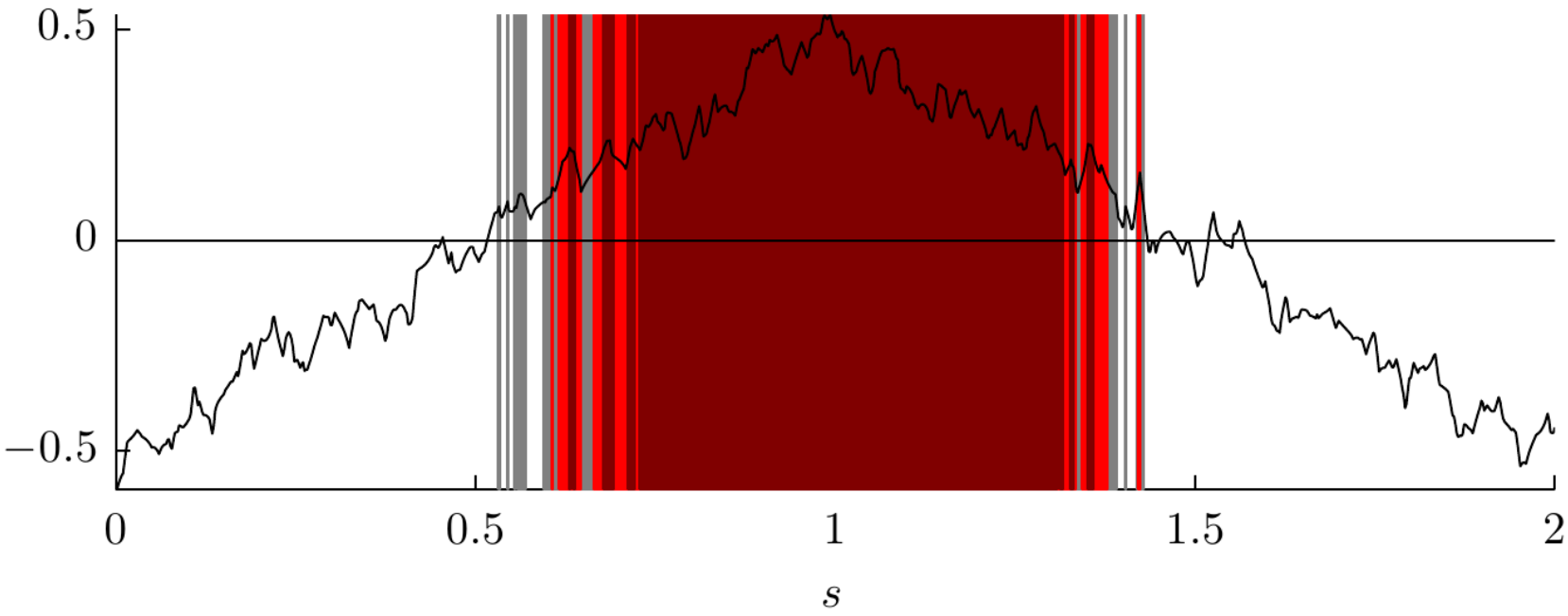
$$E^+_{u,\alpha}(x) = \arg\max_D \{|D| : \mathsf{P}(D \subseteq A^+_u(x)) \geq 1-\alpha\}.$$

$$E^-_{u,\alpha}(x) = \arg\max_D \{|D| : \mathsf{P}(D \subseteq A^-_u(x)) \geq 1-\alpha\}.$$

where $A^+_u(f) = \{s : f(s) > u\}$ and $A^-_u(f) = \{s : f(s) < u\}$.

- $E^+_{u,\alpha}(x)$ is the largest set so that, with probability $1-\alpha$, the level u is exceeded *at all locations* in the set.
- $E^-_{u,\alpha}(x)^c$ is the set that with probability $1-\alpha$ contains *all excursions*.

Example: Gaussian process with exponential covariance



- Gaussian process with exponential covariance function.
- $E^{+}_{0,0.05}(x)$ is shown in red.
- The grey area contains $\{s : \mathrm{P}(x(s) > 0) > 0.95\}$.
- The dark red set is the Bonferroni lower bound.
- The black curve is the kriging estimate of $x(s)$.

Calculating excursion sets in practise

Surprisingly little research has been devoted to this topic

Reason: The problem is extremely difficult computationally!

There are, in principle, two main problems that have to be solved:

1. Probability calculation: e.g. calculate the probability $\mathrm{P}(D \subseteq A_u^+(x))$ for a given set D.
2. Shape optimization: find the largest region D satisfying the required probability constraint.

The main problem is the probability calculation: Evaluating high-dimensional Gaussian integrals is difficult!

However, we can do this for GMRF-based models!

Gaussian integrals

- For a Gaussian vector $\mathbf{x}$, the probabilities $\mathsf{P}(D \subseteq A_u^+(x))$, $\mathsf{P}(D \subseteq A_u^-(x))$, and $\mathsf{P}(D^+ \subseteq A_u^+(x), D^- \subseteq A_u^-(x))$ can all be written on the form

$$I(\mathbf{a}, \mathbf{b}, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \int_{\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}} \exp(-\frac{1}{2}\mathbf{x}^\top \Sigma^{-1}\mathbf{x}) \, \mathrm{d}\mathbf{x},$$

- $\mathbf{a}$ and $\mathbf{b}$ are vectors depending on the mean value of $\mathbf{x}$, the domain D, and on u.
- There have been considerable research efforts devoted to approximating integrals of this form in recent years[1].
- For GMRFs, we want to use the sparsity of $\mathbf{Q}$.
- We use a method based on sequential importance sampling.

[1] A good introduction given in Genz and Bretz (2009), Computation of Multivariate Normal and t Probabilities, Lecture Notes in Statistics, Springer

A sequential Monte-Carlo algorithm

- Recall that a GMRF can be viewed as a non-homogeneous AR-process defined backwards in the indices of $\mathbf{x}$.
- Let L be the Cholesky factor of $\mathbf{Q}$, then

$$x_i|x_{i+1},\ldots,x_n \sim \mathsf{N}\left(\mu_i - \frac{1}{L_{ii}}\sum_{j=i+1}^{n} L_{ji}(x_j-\mu_j), L_{ii}^{-2}\right),$$

- Let I_i be the integral of the last $d-i$ components,

$$I_i = \int_{a_d}^{b_d} \pi(x_d) \int_{a_{d-1}}^{b_{d-1}} \pi(x_{d-1}|x_d) \cdots \int_{a_i}^{b_i} \pi(x_i|x_{i+1:d})\,\mathrm{d}x,$$

  which is the normalizing constant to the truncated density

$$f(\mathbf{x}_{i:d}) \propto 1(\mathbf{a}_{i:d} < \mathbf{x}_{i:d} < \mathbf{b}_{i:d})\pi(\mathbf{x}_{i:d})$$

- Let $\tilde{\mu}_i = \frac{1}{L_{ii}}\sum_{j=i+1}^{n} L_{ji}(x_j-\mu_j)$ and let $\pi_0(x_i)$ denote density function for the Gaussian distribution $\mathsf{N}(0, L_{ii})$.

A sequential Monte-Carlo algorithm

- The integral can then be written as

$$I_i = \int_{a_d}^{b_d} \pi_0(x_d) \int_{a_{d-1}+\tilde{\mu}_{d-1}}^{b_{d-1}+\tilde{\mu}_{d-1}} \pi_0(x_{d-1}) \cdots \int_{a_i+\tilde{\mu}_i}^{b_i+\tilde{\mu}_i} \pi_0(x_i)\, \mathrm{d}\mathbf{x}$$

- where, because of the Markov structure, $\tilde{\mu}_i$ only depends on the elements in $x_{\mathcal{N}_i \cap \{i+1:d\}}$, and $\mathcal{N}_i$ is the neighbourhood of i in the graph of the GMRF.
- In each step x_j is sampled from the truncated Gaussian distribution $1(a_j < x_j < b_j)\pi(x_j|x_{j+1:d})$ and update the importance weights recursively.
- This is a particle filter closely related to the Geweke-Hajivassiliou-Keane simulator

The algorithm

A sequential importance sampler

Calculate $I_d = \Phi(L_{ii}b_d) - \Phi(L_{ii}a_d)$, simulate N samples $\{x_d^j\}_{j=1}^N$ from the truncated normal distribution $h_d(x_d) \propto 1(a_d < x_d < b_d)\pi_0(x_d)$, and set $w_d^j = I_d$.
Loop over $i = d-1, \ldots, 1$:

- Simulate x_i^j from the truncated normal distribution $h_i(x_i|x_i^j) = 1(a_i + \tilde{\mu}_i < x_i < b_i + \tilde{\mu}_i)\pi_0(x_i)$
- Set $\mathbf{x}_{i:d}^j = \{x_i^j, x_{i+1:d}^j\}$.
- Calculate $I_i \approx \frac{1}{N}\sum_{j=1}^N w_i^j$ where w_i^j are the importance weights which are updated recursively through $w_i^j = \left[\Phi\left(L_{ii}(b_i + \tilde{\mu}_i)\right) - \Phi\left(L_{ii}(a_i + \tilde{\mu}_i)\right)\right] w_{i+1}^j$.

Parametric families for excursion sets

- We can simplify the optimization problem by assuming a parametric family for the excursion sets
- We based the parametric families on the marginal quantiles of $x(s)$, $\mathsf{P}(x(s) \leq q_\rho(s)) = \rho$, which are easy to calculate.

One-parameter family

Let $q_\rho(s)$ be the marginal quantiles for $x(s)$, then a one-parameter family for the positive and negative u excursion sets is given by

$$D_1^+(\rho) = \{s; \mathsf{P}(x(s) > u) \geq 1 - \rho\} = A_u^+(q_\rho),$$
$$D_1^-(\rho) = \{s; \mathsf{P}(x(s) < u) \geq 1 - \rho\} = A_u^-(q_{1-\rho}).$$

- Using this parametric family reduces the complexity of the shape optimization to finding the correct value of ρ.
- Important: $D_1^\star(\rho_1) \subseteq D_1^\star(\rho_2)$ if $\rho_1 < \rho_2$.

Putting the pieces together

Calculating excursion sets using a one-parameter family

Assume that $\pi(\mathbf{x})$ is Gaussian and that $D(\rho)$ is a parametric family, such that $D(\rho_1) \subseteq D(\rho_2)$ if $\rho_1 < \rho_2$. The following strategy is then used to calculate $E^+_{u,\alpha}$.

- Choose a suitable (sequential) integration method.
- Reorder the nodes to the order they will be added to the excursion set when the parameter ρ is increased.
- sequentially add nodes to the set D and in each step update the probability $\mathsf{P}(D \subseteq A^+_u(x))$. Stop as soon as this probability falls below $1 - \alpha$.
- $E^+_{u,\alpha}$ is given by the last set D for which $\mathsf{P}(D \subseteq A^+_u(x)) \geq 1 - \alpha$.

Extension to a latent Gaussian setting

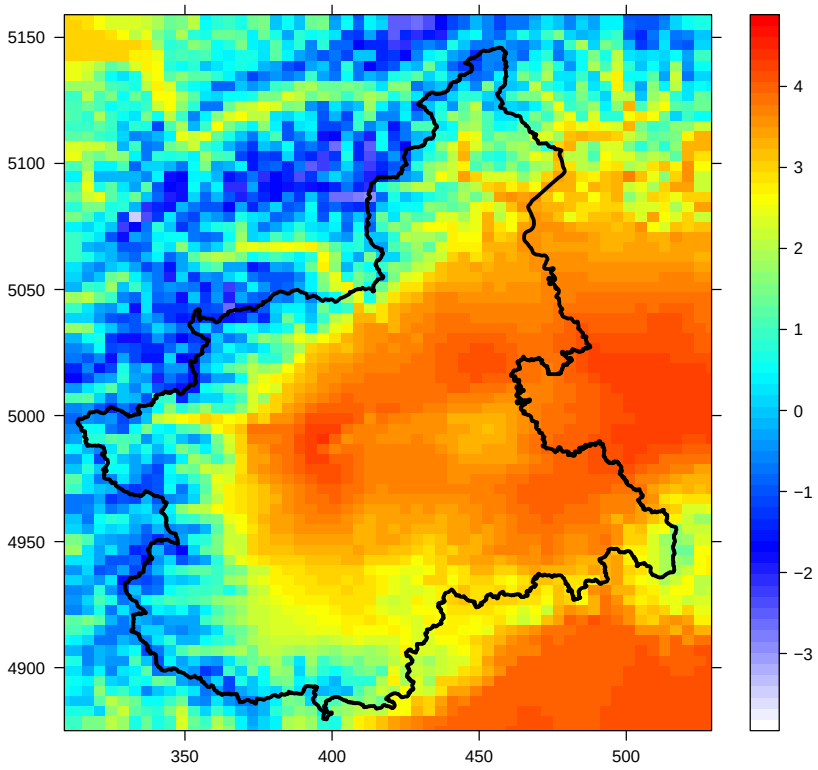
- The previous method can only be used in a purely Gaussian setting with known parameters.
- For the more general latent Gaussian setting, the posterior distribution can be written as

$$\pi(\mathbf{x}|\mathbf{y}) = \int \pi(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta})\pi(\boldsymbol{\theta}|\mathbf{y})\,\mathrm{d}\boldsymbol{\theta},$$

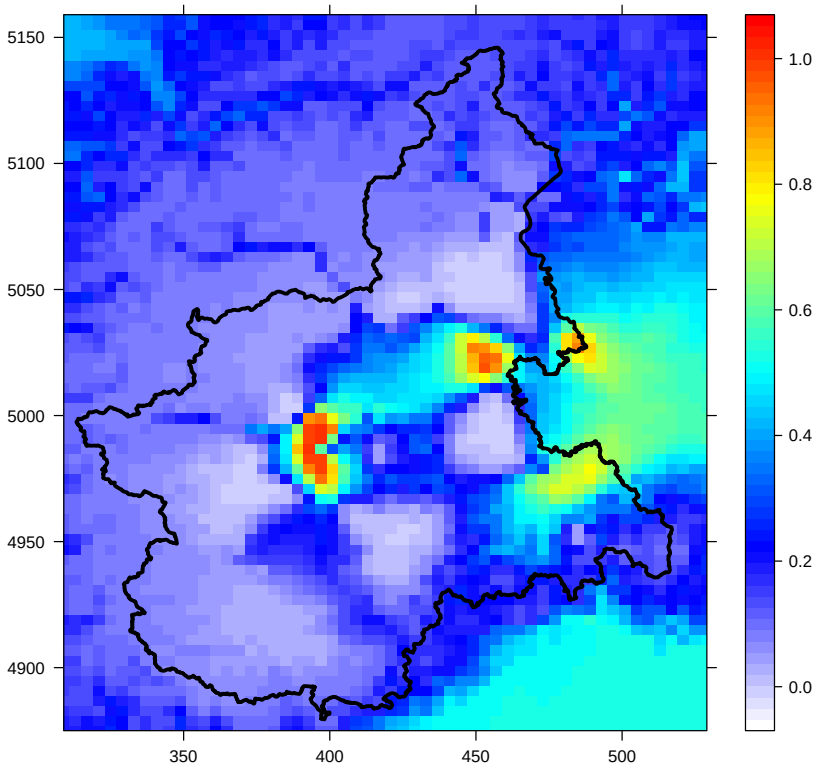
  where $\mathbf{y}$ is data and $\boldsymbol{\theta}$ the parameter vector.
- For Gaussian likelihoods, $\pi(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta})$ is Gaussian.
- There are a number of, more or less complex, ways we can extend the method to the latent Gaussian setting.
- The simplest is to use an empirical Bayes estimator where $\pi(\mathbf{x}|\mathbf{y})$ is replaced with a GMRF approximation $\pi_G(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta}_0)$
- More accurate methods can be constructed using INLA approximations and quantile corrections

Back to the Air pollution example

Spatial reconstruction

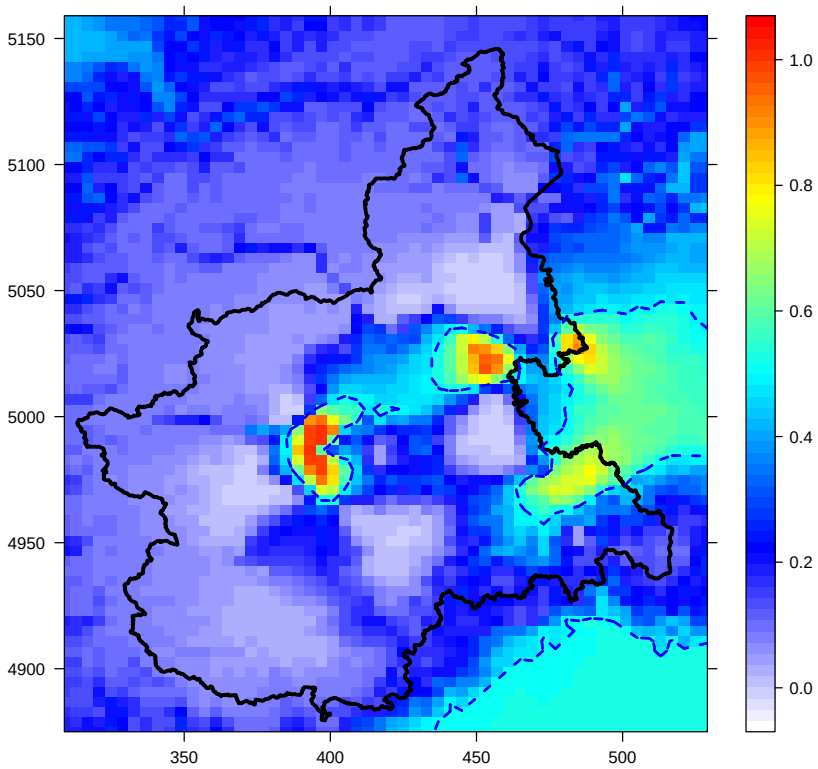


Marginal probabilities

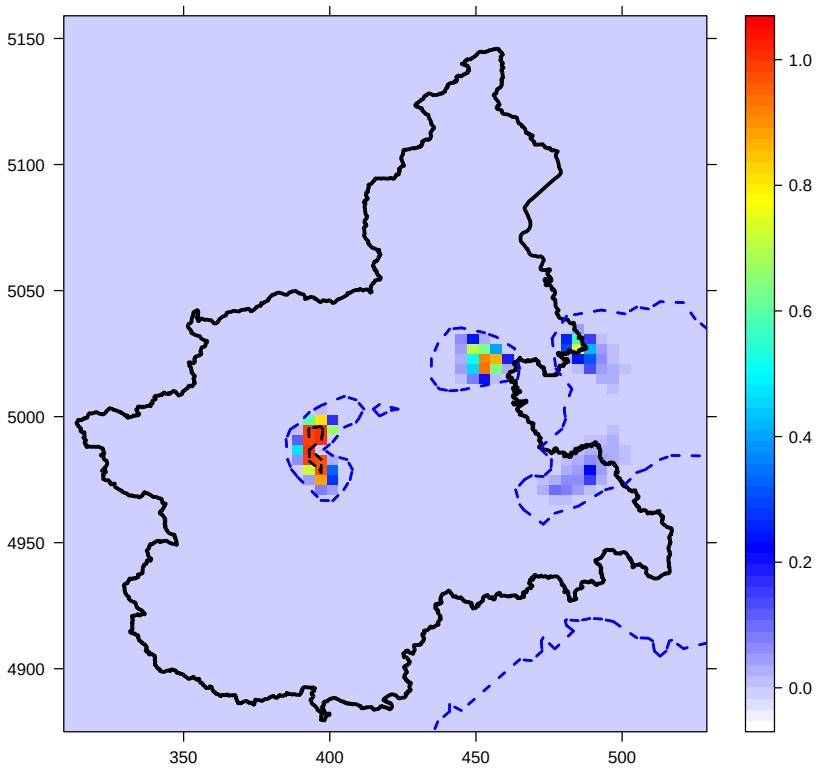


Results for January 30, 2006

Marginal probabilities

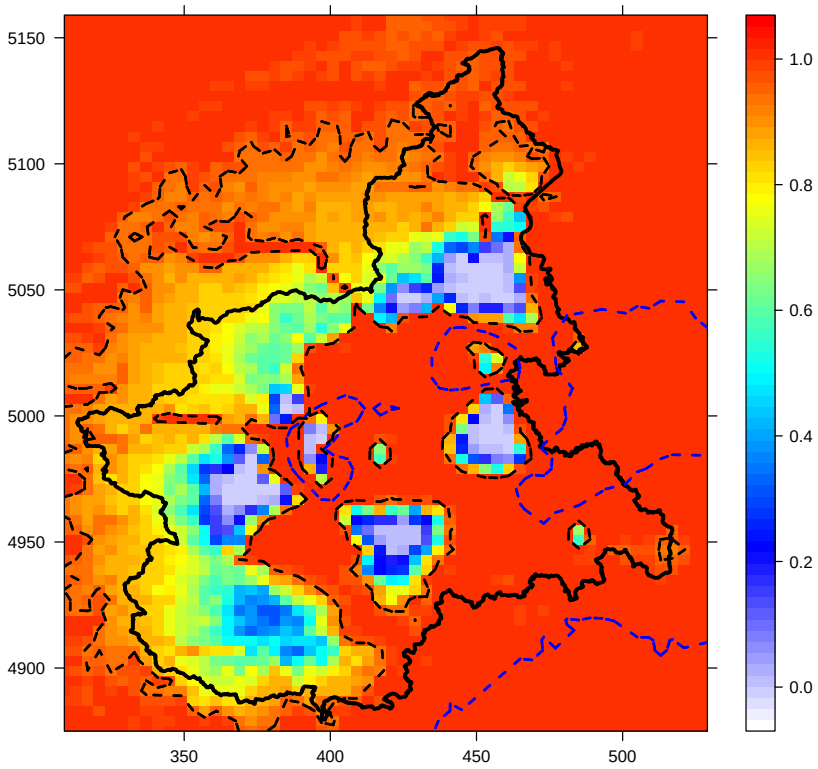


$F^{+}_{50}(s)$

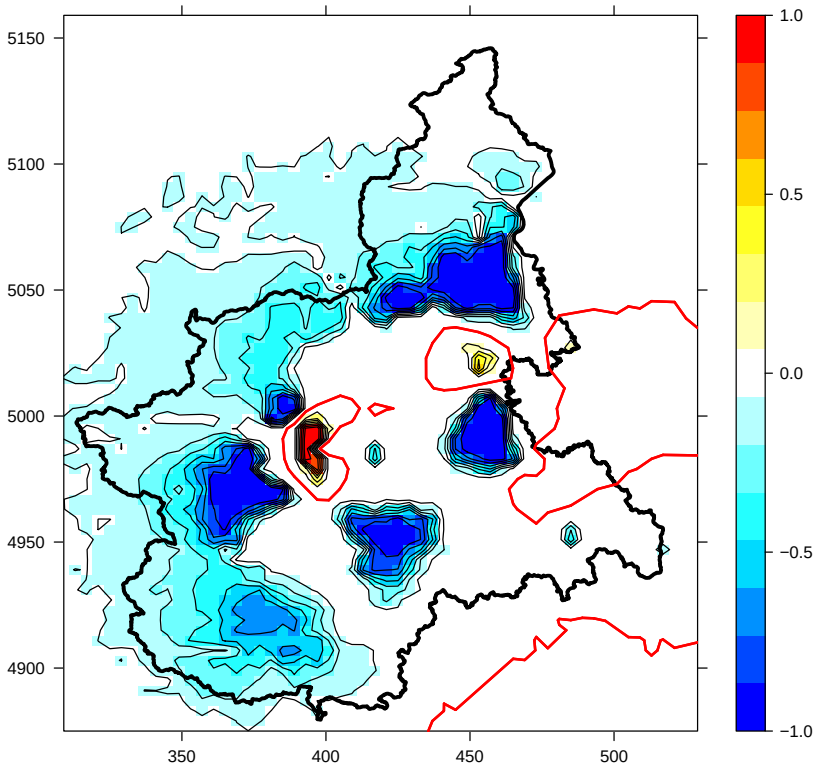


Results for January 30, 2006

Contour function $F^c_{50}(s)$



Signed avoidance $\pm F_{50}(s)$



Reference and software

There is a lot more to say about these methods, the purpose here was mostly to show another reason for why GMRFs are so useful

For more details, see Bolin and Lindgren, Excursion and contour uncertainty regions for latent Gaussian models, JRSSB (2015)

As with the INLA method: For these methods to be of any use, we need good software with a user-friendly interface.

The `excursions` package, available on CRAN, does this and has an interface to INLA. Currently it can be used to calculate

- excursion sets and excursion functions,
- Uncertainty regions for contour curves,
- Quality measures for contour maps,
- Simultaneous credible regions,

for latent Gaussian models as well as calculating Gaussian integrals.

Closing thoughts

We have covered a substantial amount of material.
Key points:

- With the increasing availability of "big data", GMRFs are increasingly important
- Modern model-based methods have many advantages compared with the standard methods in spatial statistics
- Latent Gaussian models cover a huge number of useful models
- If we're careful, we can do fast inference on big problems
- SPDE-based models have several advantages compared with covariance-based models
- R-INLA is a powerful tool that does not require specialist knowledge to use.
- Much of what we have covered are very modern methods
- There is a lot of future scope!

The final moments in the course

1. Tomorrow:
   - The final computer lab
   - I will give some more details on the INLA package
   - The plan is also to return the first project

2. Thursday 12/3:
   Daniel Simpson (Warwick) will give a seminar

   With low power comes great responsibility: challenges in modern spatial data analysis

   Don't miss this!

3. Oral exams

Regarding the oral exam

- You can choose a time at one of the following two days:
  - Friday 20/3
  - Wednesday 1/4
- I will add a Doodle signup link is on the homepage
- The exams will be approximately 45 minutes long
- You cannot take the exam unless you have passed both projects
- The exam will consist of two parts
  1. Discussion of the projects
  2. Brief presentation of a random topic from the course
- I will add a list with the possible topics to the homepage
- Jonas Wallin will also be present at the exams