Spatial modeling with INLA and inlabru

University of Zurich, March, 2022

Instructor: Sara Martino

Department of Mathematical Science (NTNU)

NTNU

Norwegian University of Science and Technology

What have we learned

Spatial models

Gaussian Random Field models

The SPDE approach

Fitting spatial models

Space-time Modeling

Prediction in space

What have we learned

INLA in a nurshell

- many data sets these days are complex, resulting in complex models, e.g. with complex dependence structures (spatial, temporal, etc..)
- usually Markov chain Monte Carlo (MCMC) methods have been used to fit these models
    - (realistically) complex models result in very long running times
    - often impossible (or unrealistic) to fit
- INLA (Integrated nested Laplace approximation) is an alternative to MCMC
    - much, much **faster**
    - suitable for a specific (but very large!) class of models

INLA in a nurshell

Three main ingredients in INLA

- Gaussian Markov random fields
- Latent Gaussian models
- Laplace approximations

which together (with a few other things) give a very nice tool for Bayesian inference

- quick
- accurate

Models amenable to INLA: **Latent Gaussian Models**

$$\mathbf{y}|\mathbf{x},\theta \sim \prod_i \pi(y_i|x_i,\theta) \qquad \text{Likelihood}$$

$$\mathbf{x}|\theta \sim \exp\left(-\frac{1}{2}\mathbf{x}^{\mathbf{T}}\mathbf{Q}(\theta)\mathbf{x}^{\mathbf{T}}\right) \qquad \text{Latent field (GMRF)}$$

$$\theta \sim \pi(\theta) \qquad \text{Hyperparameter}$$

- Each data point depends on only one of the elements in the latent Gaussian field $\mathbf{x}$.

Models amenable to INLA: **Latent Gaussian Models**

$$\mathbf{y}|\mathbf{x},\theta \sim \prod_i \pi(y_i|x_i,\theta) \qquad \text{Likelihood}$$

$$\mathbf{x}|\theta \sim \exp\left(-\frac{1}{2}\mathbf{x}^{\mathbf{T}}\mathbf{Q}(\theta)\mathbf{x}^{\mathbf{T}}\right) \qquad \text{Latent field (GMRF)}$$

$$\theta \sim \pi(\theta) \qquad \text{Hyperparameter}$$

- Each data point depends on only one of the elements in the latent Gaussian field $\mathbf{x}$.
- The size of the hyperparameter vector θ is small (say < 15)

Models amenable to INLA: **Latent Gaussian Models**

$$\mathbf{y}|\mathbf{x},\theta \sim \prod_i \pi(y_i|x_i,\theta) \qquad \text{Likelihood}$$

$$\mathbf{x}|\theta \sim \exp\left(-\frac{1}{2}\mathbf{x^T}\mathbf{Q}(\theta)\mathbf{x^T}\right) \qquad \text{Latent field (GMRF)}$$

$$\theta \sim \pi(\theta) \qquad \text{Hyperparameter}$$

- Each data point depends on only one of the elements in the latent Gaussian field $\mathbf{x}$.
- The size of the hyperparameter vector θ is small (say < 15)
- The latent field $\mathbf{x}$, can be large but it is endowed with some conditional independence (Markov) properties so that the precision matrix $Q(\theta)$ is sparse.

Models amenable to INLA: **Latent Gaussian Models**

$$\mathbf{y}|\mathbf{x},\theta \sim \prod_i \pi(y_i|x_i,\theta) \qquad \text{Likelihood}$$

$$\mathbf{x}|\theta \sim \exp\left(-\frac{1}{2}\mathbf{x}^{\mathbf{T}}\mathbf{Q}(\theta)\mathbf{x}^{\mathbf{T}}\right) \qquad \text{Latent field (GMRF)}$$

$$\theta \sim \pi(\theta) \qquad \text{Hyperparameter}$$

- Each data point depends on only one of the elements in the latent Gaussian field $\mathbf{x}$.
- The size of the hyperparameter vector θ is small (say < 15)
- The latent field $\mathbf{x}$, can be large but it is endowed with some conditional independence (Markov) properties so that the precision matrix $Q(\theta)$ is sparse.
- The linear predictor depends **linearly** on the unknown smooth function of covariates.

Models amenable to INLA: **Latent Gaussian Models**

$$\mathbf{y}|\mathbf{x}, \theta \sim \prod_i \pi(y_i|x_i, \theta) \qquad \text{Likelihood}$$

$$\mathbf{x}|\theta \sim \exp\left(-\frac{1}{2}\mathbf{x}^{\mathbf{T}}\mathbf{Q}(\theta)\mathbf{x}^{\mathbf{T}}\right) \qquad \text{Latent field (GMRF)}$$

$$\theta \sim \pi(\theta) \qquad \text{Hyperparameter}$$

- Each data point depends on only one of the elements in the latent Gaussian field $\mathbf{x}$.
- The size of the hyperparameter vector θ is small (say < 15)
- The latent field $\mathbf{x}$, can be large but it is endowed with some conditional independence (Markov) properties so that the precision matrix $Q(\theta)$ is sparse.
- The linear predictor depends **linearly** on the unknown smooth function of covariates.

INLA allows us to compute in a **fast** and **efficient** way an **accurate** approximation to the posterior marginals for the hyperparameters

$$\tilde{\pi}(\theta_j|\mathbf{y})$$

and for the latent field

$$\tilde{\pi}(x_i|\mathbf{y})$$

INLA allows us to compute in a **fast** and **efficient** way an **accurate** approximation to the posterior marginals for the hyperparameters

$$\tilde{\pi}(\theta_j|\mathbf{y})$$

and for the latent field

$$\tilde{\pi}(x_i|\mathbf{y})$$

All of this is implemented in the `INLA` library in `R`.

Yesterday we saw how to implement simple models with `INLA`

Today we are going to look at how to implement continuous spatial models.

Such model are **a lot** easier to implement in `inlabru`!!!!

inlabru in a nutshell

`inlabru` is a friendlier version of `R-INLA`

- it makes `INLA` more accessible to the user
- makes complex features and predictions (especially for spatial data) a lot easier
- is a softer-wrapper around `INLA`
- in some cases it allows to release some of the contraints of `R-INLA` (the linearity of the predictor)

inlabru

- Installation:
  - There is a CRAN version

```
install.packages("inlabru")
```

- You can also install the development version of inlabru from GitHub (recommended)

```
# install.packages("remotes")
remotes::install_github("inlabru-org/inlabru",
         ref="devel")
```

- Documentation
  - Web site: https://sites.google.com/inlabru.org/inlabru
  - Github: https://github.com/inlabru-org/inlabru

Spatial models

Types of Spatial Data

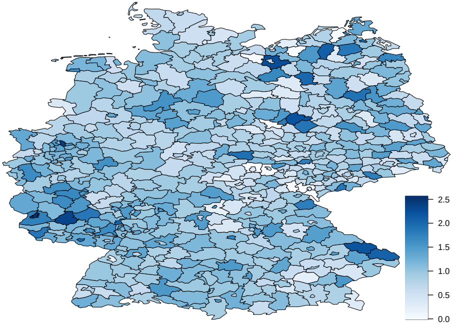
We can distinguish three types of spatial data

- Discrete space
  - data on a spatial grid
- Continuous space:
  - geostatistical data
  - spatial point data

Discrete Counts:

Data on a spatial grid

- examples: number of individuals in a region, average rainfall in a province
- (originally geostatistical or point data; gridded for practical reasons)

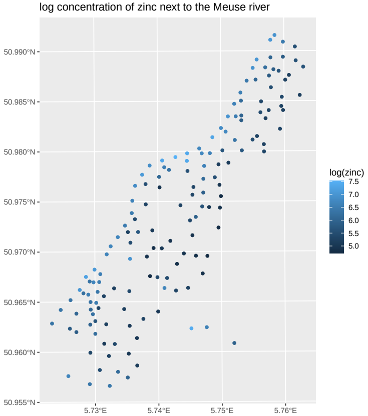


Observed response(s):

- Measurement over each grid cell (e.g. number of individuals in cell; rainfall in province)

Continuous Space: Geostatistics

- phenomenon that is continuous in space
- examples: nutrient levels in soil, salinity in the sea
- measurements at a given set of locations that are determined by surveyor

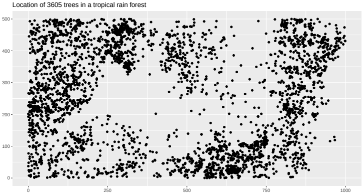


Observed response(s):

- measurement(s) taken at given locations

Continuous Space: Point Process

- locations of objects (individuals) in space (typically 2D)
- examples: locations of trees in a forest, groups of animals



Observed response(s):

- x, y coordinates of points (individuals/groups)
- maybe also properties of individuals/groups ("marks")

Gaussian Random Field models

Gaussian Random fields

Definition:A random function $u(x) : R^d \to R$ is a Gaussian random field if for any finite collection of locations, $(x_1, \ldots, x_n)$, $x_i \in R^d$, the joint distribution of $\mathbf{u} = (u(x_1), \ldots, u(x_n))$ is $\mathbf{u} \sim N(0, \Sigma)$, and

$$
\begin{gathered}
E(u(x)) = 0, \\
Cov(u(x), u(x')) = R(x, x'), \quad \Sigma_{ij} = R(x_i, x_j)
\end{gathered}
$$

for some expectation function $\mu(\cdot)$ and positive definite covariance function $R(\cdot, \cdot)$. Σ is the covariance matrix for the specific location collection.

Matern covariance function

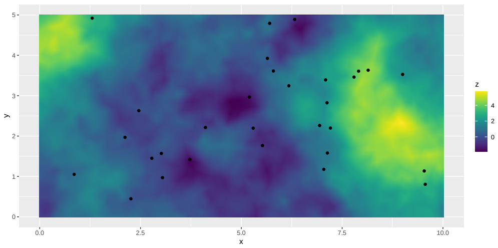
The Matern covariance function is a popular model in spatial problems:

$$c_\nu(d;\sigma,\rho) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{8\nu}\frac{d}{\rho}\right)^\nu K_\nu \left(\sqrt{8\nu}\frac{d}{\rho}\right)$$

* ρ and σ are the range and the marginal standard deviation

- K_ν modified Bessel function of the second kind, order ν.

Example



Gaussian Random field

GRF are a very popular model

- Flexible and easy to use
- Can be part of the latent Gaussian field in a LGM

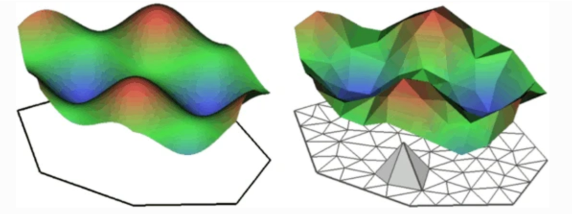
However:

- **computationally inefficient** (the precision matrix is dense)
- not flexible enough (complicated boundary, barrier,...)

The SPDE approach

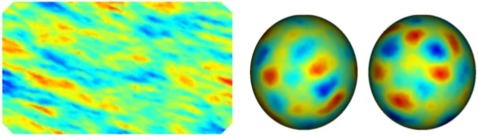
The SPDE approach

- Matern fields can be seen as solution to a PDE
- Using finite element methods such solution can be represented using a GRMF



Advantages of the SPDE approach

- Computationally fast
- Allows for flexible modeling
  - non-stationary models (anisotropy)
  - models on a sphere
  - non separable models



All these models (and my more) can be fitted with `R-INLA` and `inlabru`

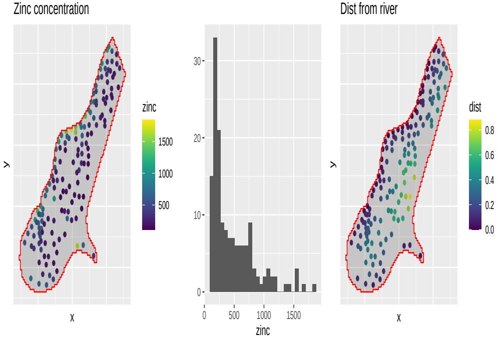
Learning about the SPDE approach

- F. Lindgren, H. Rue, and J. Lindström. *An explicit link between Gaussian fields and Gaussian Markov random fields: The SPDE approach (with discussion)*. In: Journal of the Royal Statistical Society, Series B 73.4 (2011), pp. 423–498.
- H. Bakka, H. Rue, G. A. Fuglstad, A. Riebler, D. Bolin, J. Illian, E. Krainski, D. Simpson, and F. Lindgren. *Spatial modelling with R-INLA: A review*. In: WIREs Computational Statistics 10:e1443.6 (2018). (Invited extended review). DOI: 10.1002/wics.1443.
- E. T. Krainski, V. Gómez-Rubio, H. Bakka, A. Lenzi, D. Castro-Camilio, D. Simpson, F. Lindgren, and H. Rue. *Advanced Spatial Modeling with Stochastic Partial Differential Equations using R and INLA*. Github version www.r-inla.org/spde-book. CRC press, Dec. 20

Fitting spatial models

Example: Meuse Data

Measures of zinc concentration.



The model

$$\log(Y(s)) \sim \mathcal{N}(\eta(s), \sigma_y^2)$$
$$\eta(s) = \alpha + \beta\ x(s) + u(s)$$

where

- $Y(s)$ is the measure of zinc in location s
- α a common intercept
- β a model parameter
- $x(s)$ distance from the river at location s
- $u(s)$ the Matern Gaussian field

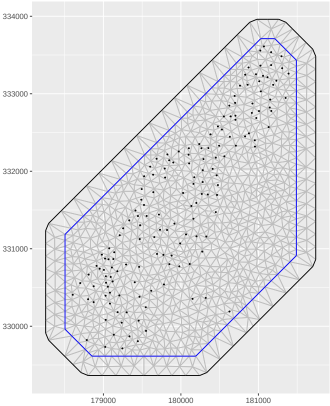
Step 1: Define the SPDE representation

- The mesh
- The SPDE model

Define the SPDE representation: The mesh

1. Define the mesh

```
mesh <- inla.mesh.2d(loc.domain = cbind(meuse$x, meuse$y),
                     max.edge = c(150, 500),
                     offset = c(100, 250) )
```



Define the SPDE representation: The mesh

- All random field models need to be discretised for practical calculations.
- The SPDE models were developed to provide a consistent model definition across a range of discretisations.
- We use finite element methods with local, piecewise linear basis functions defined on a triangulation of a region of space containing the domain of interest.
- Deviation from stationarity is generated near the boundary of the region.
- The choice of region and choice of triangulation affects the numerical accuracy.

Define the SPDE representation: The mesh

- All random field models need to be discretised for practical calculations.
- The SPDE models were developed to provide a consistent model definition across a range of discretisations.
- We use finite element methods with local, piecewise linear basis functions defined on a triangulation of a region of space containing the domain of interest.
- Deviation from stationarity is generated near the boundary of the region.
- The choice of region and choice of triangulation affects the numerical accuracy.

Two separate issues:

- Continuous space with bounded domain: Boundary effect
- Discretised model: Numerical accuracy

Sometimes the boundary effect may be desireable.

Define the SPDE representation: The mesh

- Too fine meshes → heavy computation
- Too coarse mesh → not accurate enought

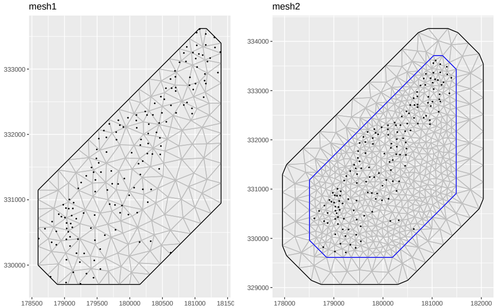
Some guidelines

- Create triangulation meshes with `inla.mesh.2d()`
- Move undesired boundary effects away from the domain of interest by extending to a smooth external boundary (`inla.nonconvex.hull(loc, convex)`, `convex` $\geq$ correlation range)
- Use a coarser resolution in the extension to reduce computational cost (`max.edge=c(inner, outer)`)
- Use a fine resolution (subject to available computational resources) for the domain of interest (inner correlation range) and filter out small input point clusters (`0 < cutoff < inner`)
- Coastlines and similar can be added to the domain specification in `inla.mesh.2d()`

Define the SPDE representation: The mesh

```
mesh1 = inla.mesh.2d(loc.domain = cbind(meuse$x, meuse$y),
                     max.edge = 350,
                     offset = 10)

mesh2 = inla.mesh.2d(loc.domain = cbind(meuse$x, meuse$y),
                     max.edge = c(150, 500),
                     cutoff = 100,
                     offset = c(100, 550) )
```



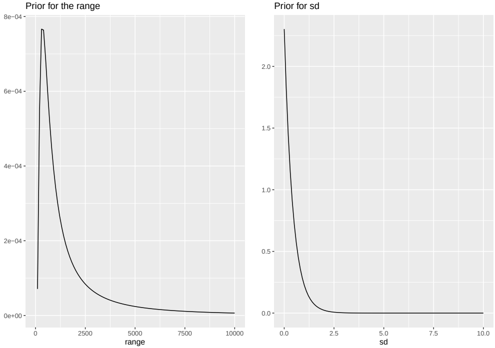
Define the SPDE representation: The SPDE model

```
meuse.spde <- inla.spde2.pcmatern(mesh = mesh,
                                  prior.sigma = c(1, 0.1),
                                  prior.range = c(1000, 0.5))
```

PC-priors for the range ρ and the standard deviarion σ

- Define the prior for the range `prior.range = (range0,Prange)` $\text{Prob}(\rho < \rho_0) = p_\rho$
- Define the prior for the range `prior.sigma = (sigma0,Psigma)` $\text{Prob}(\sigma > \sigma_0) = p_\sigma$

Prior for range ρ and sd σ



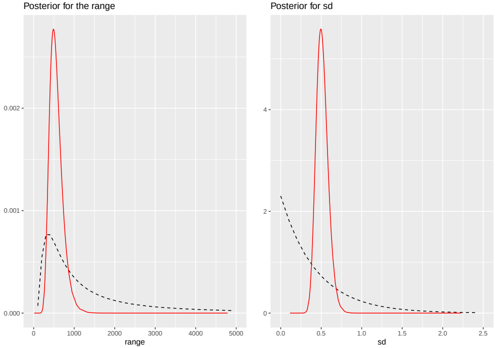
Run the model inlabru

```
# create a spatial object
coordinates(meuse) = c("x","y")
#  covariate values
dist_SPDE = SpatialPixelsDataFrame(data$dist_raster[,c(1,2)],
                                   data = data.frame(dist = data$dist_raster[,3]))
# model components
cmp =  ~ Intercept(1) + dist(dist_SPDE, model = "linear") +
  spde(coordinates, model = meuse.spde)
# define likelihood
lik = like(formula = Y ~ Intercept + dist +   spde,
           family = "gaussian",
           data  = meuse)
#fit the model
fit <- bru(cmp, lik)
# define prediction area
pix <- pixels(mesh, nx = 200, ny = 200, mask = boundary)
# generate predictions
pred = predict(fit, pix, ~ data.frame(
                                       spde = spde,
                                       logscale = Intercept + dist + spde,
                                       naturalscale = exp(Intercept + dist + spde)))
```

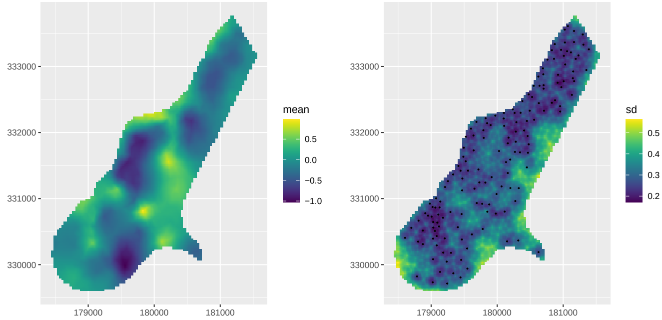
Notes!

- The data are a spatial object!
- For prediction, the covariates are stored in a `SpatialPixelsDataFrame` and need to cover all the mesh nodes

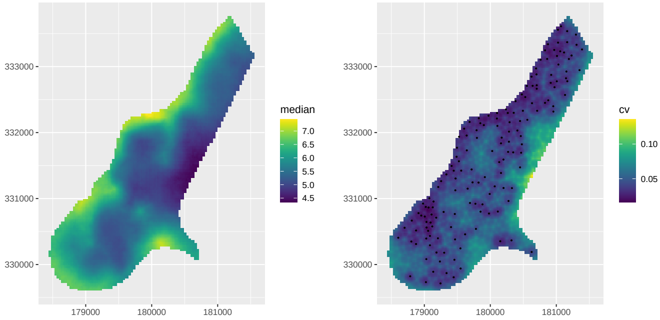
Estimation: Posterior for the hyperparameters



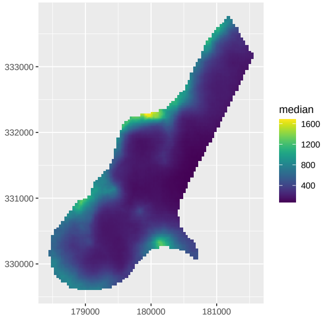
Predictions: The SPDE field

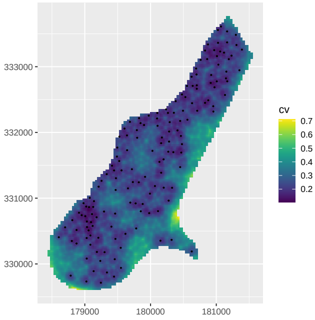


Predictions: The log concentrations



Predictions: The concentrations





Same in plain INLA (1)

```
A.meuse <- inla.spde.make.A(mesh = mesh, loc = coordinates(meuse))
s.index <- inla.spde.make.index(name = "spatial.field",
  n.spde = meuse.spde$n.spde)

#Create data structure
meuse.stack <- inla.stack(data  = list(zinc = meuse$zinc),
  A = list(A.meuse, 1),
  effects = list(c(s.index, list(Intercept = 1)),
    list(dist = meuse$dist)),
  tag = "meuse.data")

data(meuse.grid)
coordinates(meuse.grid) = ~x+y
gridded(meuse.grid) = TRUE

#Create data structure for prediction
A.pred <- inla.spde.make.A(mesh = mesh, loc = coordinates(meuse.grid))
meuse.stack.pred <- inla.stack(data = list(zinc = NA),
  A = list(A.pred, 1),
  effects = list(c(s.index, list (Intercept = 1)),
    list(dist = meuse.grid$dist)),
  tag = "meuse.pred")

#Join stack
join.stack <- inla.stack(meuse.stack, meuse.stack.pred)
```

Same in plain INLA (2)

```
#Fit model
form <- log(zinc) ~ -1 + Intercept + dist + f(spatial.field, model = spde)

m1 <- inla(form, data = inla.stack.data(join.stack, spde = meuse.spde),
  family = "gaussian",
  control.predictor = list(A = inla.stack.A(join.stack), compute = TRUE))
```

Note: We still have not compute predictions. . . and this is not too easy in plain INLA!!

When is `inlabru` easier to use

- spatial modeling.
- point processes.
- multiple likelihoods
- when interested in spatial predictions

When is `inlabru` easier to use

- spatial modeling.
- point processes.
- multiple likelihoods
- when interested in spatial predictions
- `inlabru` is also useful if one has non-linearities in the predictor η
  - born for ecological models (for example transect sampling) but used also in other fields

When is `inlabru` easier to use

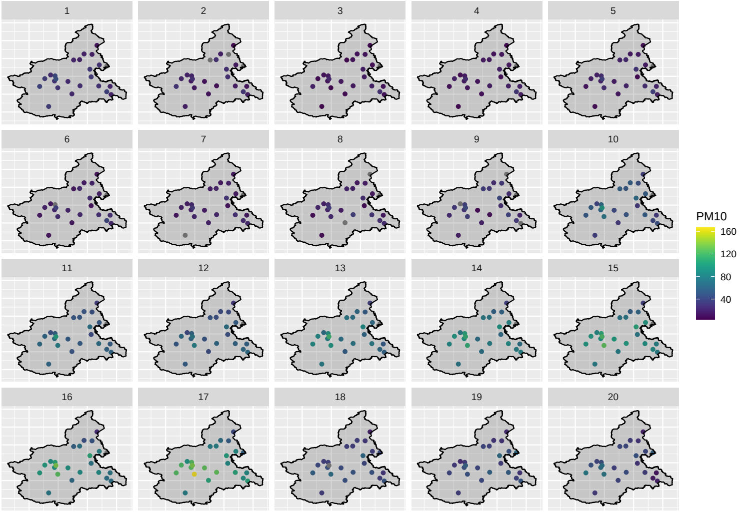
- spatial modeling.
- point processes.
- multiple likelihoods
- when interested in spatial predictions
- `inlabru` is also useful if one has non-linearities in the predictor η
  - born for ecological models (for example transect sampling) but used also in other fields

What cannot (at the moment) be done with `inlabru`

- Survival models

Space-time Modeling

Modeling PM10 concentration in Piemonte (Italy)



Modeling PM10 concentration in Piemonte (Italy)

Example from the Blangiardo & Cameletti book but simplified!

```
##   Station.ID       Date   dem      x       y   WS   temp   HMIX PREC   EMI PM10
## 1          1 2005-10-01  95.2 469.45 4972.85 0.90 288.81 1294.6    0 26.05   28
## 2          2 2005-10-01 164.1 423.48 4950.69 0.82 288.67 1139.8    0 18.74   22
## 3          3 2005-10-01 242.9 490.71 4948.86 0.96 287.44 1404.0    0  6.28   17
## 4          4 2005-10-01 149.9 437.36 4973.34 1.17 288.63 1042.4    0 29.35   25
## 5          5 2005-10-01 405.0 426.44 5045.66 0.60 287.63 1038.7    0 32.19   20
##   time  logPM10
## 1    1 3.332205
## 2    1 3.091042
## 3    1 2.833213
## 4    1 3.218876
## 5    1 2.995732
```

The model

We model the log PM10 concentration:

$$
\begin{aligned}
y_{it} &\sim \mathcal{N}(\eta_i t, \sigma_e^2) \\
\eta_{it} &= \alpha + \beta_1 \text{dem}_i + \beta_2 \text{temp}_{it} + \omega_{it}
\end{aligned}
$$

- y_{it} is the log-concentration at location i in time t
- α is an intercept
- β_1 and β_2 parameters of altitude and temperature
- ω_{it} is the space-time residual

Space time residual model

A first order autoregressive process with spatially colored innovations

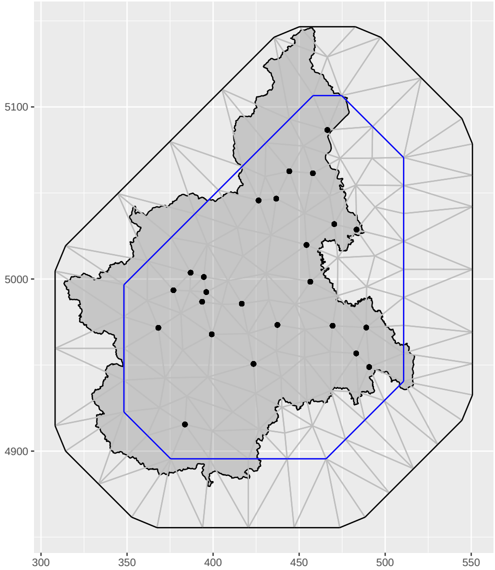
$$\omega_{it} = a\omega_{i(t-1)} + \xi_{it}$$

- $|a| < 1$ parameter of the AR1 process
- ξ_{it} is a zero mean, temporally independent, Gaussian field with

$$\mathrm{Cov}(\xi_{it}, \xi_{ju}) = \begin{cases} 0, & \text{if } t \neq u \\ \mathcal{C}(h), & \text{if } t = u \end{cases}$$

- h distance between locations i and j - $\mathcal{C}(h)$ is a Matern correlation function

Building the model: mesh



Building the model: Prior for the spde model

```
spde = inla.spde2.pcmatern(mesh = mesh,
                           prior.range=c(100,0.5),
                           prior.sigma=c(1,0.1))
```

- $\text{Prob}(\rho < 100 \text{ Km}) = 0.5$
- $\text{Prob}(\sigma > 1) = 0.1$

Implement the model

```r
# make the data a spatial object
coordinates(df) = c("x","y")

# model component
cmp  = ~ Intercept(1) +
  SPDE(coordinates, model = spde,
       group = time, control.group = list(model = "ar1")) +
  dem(dem, model = "linear") +
  temp(temp, model = "linear")

# likelihood
lik = like(formula = logPM10 ~ Intercept + SPDE + dem + temp ,
           family = "gaussian",
           data = df)

# fit the model
fit = bru(cmp, lik,
          options = list(verbose = F,
                         bru_max_iter = 1,
                         inla.mode  = "experimental"))
```

Model Parameters

Fixed effects:

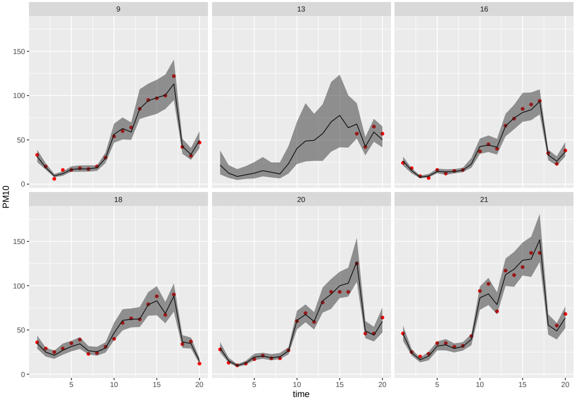
```
##                   mean           sd    0.025quant   0.975quant
## Intercept -1.438103e+01 9.2183351517 -32.139029373 4.0476156436
## dem       -7.283307e-04 0.0008432533  -0.002398673 0.0009354501
## temp       6.250902e-02 0.0320760645  -0.001679595 0.1243716481
```

Hyperparameters of the random field:

```
##                                             mean          sd  0.025quant
## Precision for the Gaussian observations  41.8579575  7.23518895  29.3185249
## Range for SPDE                          129.1799687 16.96397904 101.0510932
## Stdev for SPDE                            0.7511733  0.09963733   0.5913784
## GroupRho for SPDE                         0.9105294  0.02594638   0.8555876
##                                         0.975quant
## Precision for the Gaussian observations  57.6625010
## Range for SPDE                          167.2757328
## Stdev for SPDE                            0.9796968
## GroupRho for SPDE                         0.9557794
```

Prediction at station locations

```
pred_at_station = predict(fit, df,
                          ~exp(Intercept + SPDE + dem + temp ))
```



Prediction is space

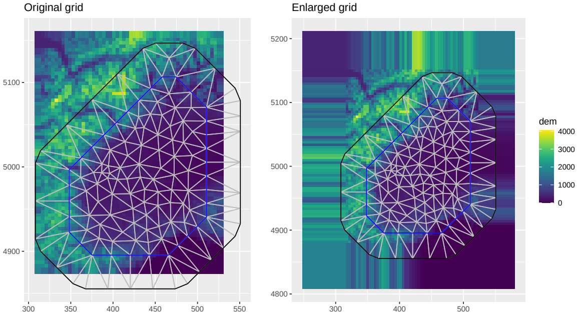
- To do this we need to have the covariates over the space of interest.
- The should cover the whole mesh

Expanding the covariates: the bru_fill_missing

```
# read the covariate values
alt = read.table("Data/Piemonte_data/Altitude.dat")
## object with altitude, this does not cover the whole mesh!
dem = SpatialPixelsDataFrame(points = cbind(alt[,1],
                                            alt[,2]),
                             data = data.frame(dem = alt[,3]))
## create another object which  does cover the whole mesh
large_grid = expand_grid(x = seq(250, 580,4),
                         y = seq(4810, 5210,4))
# expand dem ------------------------------------------------------------
dem_large = SpatialPixelsDataFrame(points = large_grid,
                                   data = data.frame(dem =
                                                    rep(NA,length(large_grid$x))))

# Use bru_fill_missing to fill the new object
dem_large$dem <- bru_fill_missing(data = dem,
                       where = dem_large,
                       values = dem_large$dem)
```

Expanding the covariates: the `bru_fill_missing`



Note: You can do this with your favourite method!

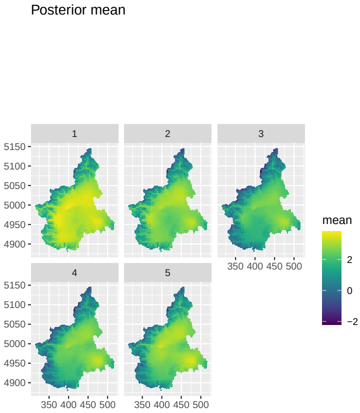
Prediction in space

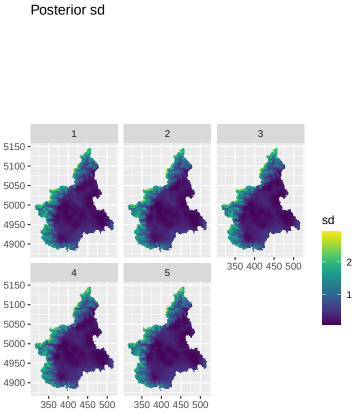
Prediction in space

```
# Create a space time grid
pxl = pixels(mesh, nx = 200, ny = 200, mask = shape)
ips2 <- ipoints(domain = c(1:5), name = "time")
pxl_time <- cbind(cprod(pxl,ips2), data.frame(temp = 0))

pred_space = predict(fit, pxl_time,
                      ~ Intercept + SPDE +
                       dem_eval(inlabru:::eval_SpatialDF(dem_large,
                                                          .data.)) +
                       temp_eval(inlabru:::eval_SpatialDF(temp_large,
                                                           .data.,
                                                           selector = "time")))
```

Prediction in space





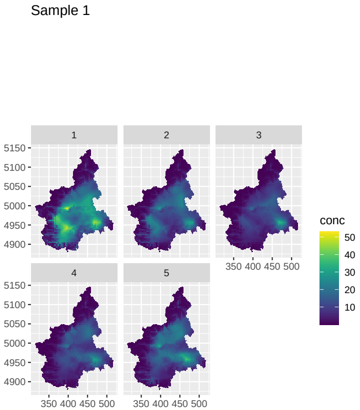
Samples form the fitted model

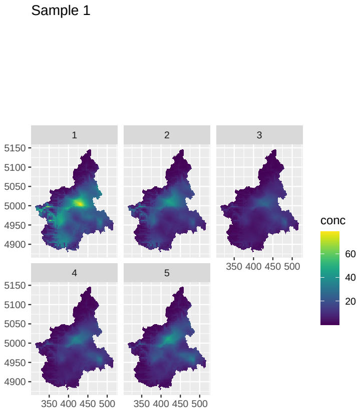
It is also possible to generate from the fitted model

```
sim_space = generate(fit, pxl_time,
                     ~ exp(Intercept + SPDE +
                      dem_eval(inlabru:::eval_SpatialDF(dem_large,
                                                        .data.)) +
                      temp_eval(inlabru:::eval_SpatialDF(temp_large,
                                                         .data.,
                                                         selector = "time"))),
                    n.samples = 2)
```

This can be usefull as the posterior means are always smoother than the “real” field

Samples form the fitted model





Some concluding remark

- The functions `predict()` and `generate()` in `inlabru` use the `INLA` function `inla.posterior.sample()` internally.
- `inlabru` is a wrapper around `INLA` so all the internal computations are identical!
- `inlabru` makes handling of spatial object a lot easier
- Transition from `sp` to `sf`/`terra` some changing can be expected

Thank you for your attention!

If you have any doubts or questions, please write :

sara.martino@math.ntnu.no