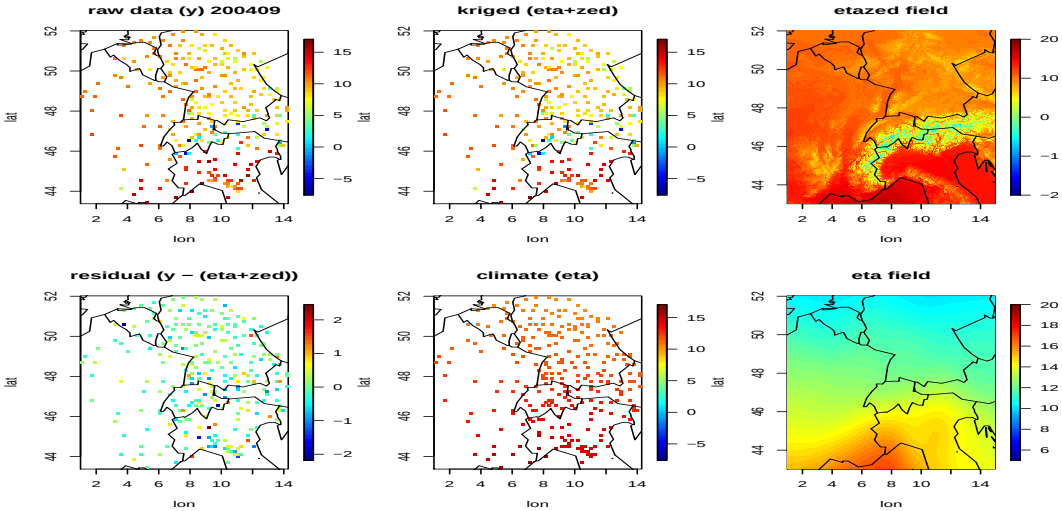
inlabru: Bayesian spatial and spatio-temporal modelling in R

Finn Lindgren
finn.lindgren@ed.ac.uk

4DM Hackathon, Oslo, 2-3 November 2023

Sparse spatial coverage of temperature measurements



Regional observations: $\approx$ 20,000,000 from daily timeseries over 160 years

Spatio-temporal modelling framework

Spatial statistics framework

- Spatial domain D, or space-time domain $D \times \mathbb{T}$, $\mathbb{T} \subset \mathbb{R}$.
- Random field $u(\boldsymbol{s})$, $\boldsymbol{s} \in D$, or $u(\boldsymbol{s}, t)$, $(\boldsymbol{s}, t) \in D \times \mathbb{T}$.
- Observations y_i. In the simplest setting, $y_i = u(\boldsymbol{s}_i) + \epsilon_i$, but more generally $y_i \sim \mathsf{GLMM}$, with $u(\cdot)$ as a structured random effect.
- Needed: models capturing stochastic dependence on multiple scales
- Partial solution: Basis function expansions, with large scale functions and covariates to capture static and slow structures, and small scale functions for more local variability

Two basic model and method components

- Stochastic models for $u(\cdot)$.
- Computationally efficient (i.e. avoid MCMC whenever possible) inference methods for the posterior distribution of $u(\cdot)$ given data $\boldsymbol{y}$.

Covariance functions and stochastic PDEs

The Matérn covariance family on $\mathbb{R}^d$

$$\mathrm{Cov}(u(\mathbf{0}), u(\boldsymbol{s})) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} (\kappa \|\boldsymbol{s}\|)^\nu K_\nu(\kappa \|\boldsymbol{s}\|)$$

Scale $\kappa > 0$, smoothness $\nu > 0$, variance $\sigma^2 > 0$

Whittle (1954, 1963): Matérn as SPDE solution

Matérn fields are the stationary solutions to the SPDE

$$\left(\kappa^2 - \nabla \cdot \nabla\right)^{\alpha/2} u(\boldsymbol{s}) = \mathcal{W}(\boldsymbol{s}), \quad \alpha = \nu + d/2$$

$\mathcal{W}(\cdot)$ white noise, $\nabla \cdot \nabla = \sum_{i=1}^d \frac{\partial^2}{\partial s_i^2}$, $\sigma^2 = \frac{\Gamma(\nu)}{\Gamma(\alpha)\kappa^{2\nu}(4\pi)^{d/2}}$

Gaussian random field (or Gaussian process)

A *Gaussian random field* $u : D \mapsto \mathbb{R}$ is defined via

$$\begin{aligned}
\mathsf{E}(u(\mathbf{s})) &= m(\mathbf{s}), \\
\mathsf{Cov}(u(\mathbf{s}), u(\mathbf{s}')) &= K(\mathbf{s}, \mathbf{s}'), \quad \text{(covariance kernel)} \\
\big[u(\mathbf{s}_i), i = 1, \ldots, n\big] &\sim \mathsf{N}(\boldsymbol{m} = \big[m(\mathbf{s}_i), i = 1, \ldots, n\big], \\
&\qquad \boldsymbol{\Sigma} = \big[K(\mathbf{s}_i, \mathbf{s}_j), i, j = 1, \ldots, n\big])
\end{aligned}$$

for all finite location sets $\{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$, and $K(\cdot, \cdot)$ symmetric positive definite.

Generalised random field

A *generalised Gaussian random field* $u : D \mapsto \mathbb{R}$ is defined via a random measure, $\langle f, u\rangle_D = u^*(f) : H_{\mathcal{R}}(D) \mapsto \mathbb{R}$, $\mathcal{R}$ a covariance operator,

$$\mathsf{E}(\langle f, u\rangle_D) = \langle f, m\rangle_D = \int_D f(\mathbf{s}) m(\mathbf{s})\, \mathrm{d}\mathbf{s},$$

$$\mathsf{Cov}(\langle f, u\rangle_D, \langle g, u\rangle_D) = \langle f, \mathcal{R} g\rangle_D \equiv \iint_{D\times D} f(\mathbf{s}) K(\mathbf{s}, \mathbf{s}') g(\mathbf{s}')\, \mathrm{d}\mathbf{s}\, \mathrm{d}\mathbf{s}',$$

$$\langle f, u\rangle_D \sim \mathsf{N}(\langle f, m\rangle_D, \langle f, \mathcal{R} f\rangle_D)$$

for all $f, g \in H_{\mathcal{R}}(D) \equiv \{f : D \mapsto \mathbb{R};\ \langle f, \mathcal{R} f\rangle_D < \infty\}$.
This allows for singular covariance kernels $K(\cdot, \cdot)$.

Simple heat equation

For space-time fields, we write $u(\mathbf{s},t)$, $(\mathbf{s},t) \in \mathbb{R}^d \times \mathbb{R}$, and $S_u(\boldsymbol{k},\omega)$, $(\boldsymbol{k},\omega) \in \mathbb{R}^d \times \mathbb{R}$.
We drive a heat equation with a noise process $\mathcal{E}$ that is white noise in time and Matérn noise in space, with parameters matching the heat operator:

$$\left\{\gamma\frac{\partial}{\partial t} + \kappa^2 - \nabla_s \cdot \nabla_s\right\} u(\mathbf{s}) = \mathcal{E}(\mathbf{s},t),$$
$$(\kappa^2 - \nabla_s \cdot \nabla_s)^{\alpha/2}\mathcal{E}(\mathbf{s},t) = \mathcal{W}(\mathbf{s},t).$$

The Fourier domain version is

$$\left\{i\gamma\omega + \kappa^2 + \|\boldsymbol{k}\|^2\right\}\widehat{u}(\boldsymbol{k},\omega) = \widehat{\mathcal{E}}(\boldsymbol{k},\omega),$$
$$(\kappa^2 + \|\boldsymbol{k}\|^2)^{\alpha/2}\widehat{\mathcal{E}}(\boldsymbol{k},\omega) = \widehat{\mathcal{W}}(\boldsymbol{k},\omega),$$

and

$$S_u(\boldsymbol{k},\omega) = \frac{1}{(2\pi)^{d+1}(\gamma^2\omega^2 + (\kappa^2 + \|\boldsymbol{k}\|^2)^2)(\kappa^2 + \|\boldsymbol{k}\|^2)^{\alpha}}$$

How differentiable are the realisations?

Simple heat equation (cont)

Using that, in the standardised Whittle-Matérn SPDE, the variance is

$$\sigma^2 = \frac{\Gamma(\nu)}{\Gamma(\alpha)\kappa^{2\nu}(4\pi)^{d/2}}, \quad \nu = \alpha - d/2,$$

the marginal spatial spectrum for the heat model is

$$S_u(\boldsymbol{k}) = \int_{\mathbb{R}} S_u(\boldsymbol{k}, \omega)\, \mathrm{d}\omega = \frac{1}{4\pi\gamma} \frac{1}{(2\pi)^d(\kappa^2 + \|\boldsymbol{k}\|^2)^{\alpha+1}},$$

which is a scaled Whittle spectrum for a Matérn covariance with smoothness $\nu = \alpha + 1 - d/2$.

A generalised generalised case

If $\alpha = 0$, $d = 2$, then $\nu = 0$, which is just outside of the allowed range of the Matérn family. However, for every t, $u(\cdot, t)$ is a generalised random field with singular kernel $K(\mathbf{s}, \mathbf{s}') = \frac{1}{4\pi\gamma}\frac{1}{2\pi}K_0(\kappa\|\mathbf{s}' - \mathbf{s}\|)$.

Simple heat equation (cont)

To help understand the temporal properties, take the Fourier transform in only the spatial directions:

$$\left\{\gamma\frac{\partial}{\partial t}+\kappa^2+\|\boldsymbol{k}\|^2\right\}\widetilde{u}(\boldsymbol{k},t)=\frac{\widetilde{\mathcal{W}}(\boldsymbol{k},t)}{(\kappa^2+\|\boldsymbol{k}\|^2)^{\alpha/2}},$$

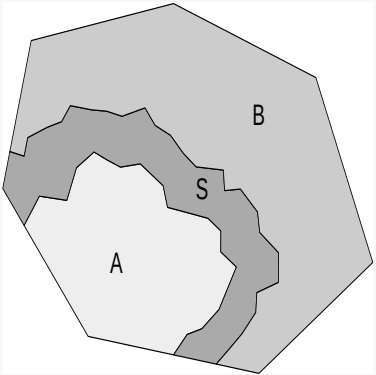
so for each spatial frequency $\boldsymbol{k}$, the temporal evolution of $\widetilde{u}(\boldsymbol{k},t)$ is an Ornstein-Uhlenbeck process with covariance

$$\frac{1}{4\pi\gamma(\kappa^2+\|\boldsymbol{k}\|^2)^{\alpha+1}}\exp\left(-|t|\frac{\kappa^2+\|\boldsymbol{k}\|^2}{\gamma}\right).$$

There is one more property we need to understand: Markov in space

Markov in space

Markov properties



S is a separating set for A and B: $u(A) \perp u(B) \mid u(S)$

Solutions to

$$\left(\kappa^2 - \nabla \cdot \nabla\right)^{\alpha/2} u(\boldsymbol{s}) = \mathcal{W}(\boldsymbol{s})$$

are Markov when α is an integer.

More generally, when the reciprocal of the spectral density is a polynomial, Rozanov, 1977

In graphs with no edges between A and B ($\boldsymbol{Q} = \boldsymbol{\Sigma}^{-1}$):

$$\boldsymbol{Q}_{AB} = \boldsymbol{0}$$
$$\boldsymbol{Q}_{A|S,B} = \boldsymbol{Q}_{AA}$$
$$\boldsymbol{\mu}_{A|S,B} = \boldsymbol{\mu}_A - \boldsymbol{Q}_{AA}^{-1}\boldsymbol{Q}_{AS}(\boldsymbol{u}_S - \boldsymbol{\mu}_S)$$

Generally: Markov iff the precision *operator* $\mathcal{Q} = \mathcal{R}^{-1}$ is local.

Markov in space

The precision matrix block structure $\begin{bmatrix} \boldsymbol{Q}_{AA} & \boldsymbol{Q}_{AS} & \boldsymbol{0} \\ \boldsymbol{Q}_{SA} & \boldsymbol{Q}_{SS} & \boldsymbol{Q}_{SB} \\ \boldsymbol{0} & \boldsymbol{Q}_{BS} & \boldsymbol{Q}_{BB} \end{bmatrix}$ has important implications for practical computation (Cholesky, see later)

A partial history of Markov random fields

Rozanov (1977)

Generally: Markov iff the precision *operator* $\mathcal{Q} = \mathcal{R}^{-1}$ is local.

Stationary case:

$$u(\mathbf{s}) \text{ is stationary Markov} \iff S_u(\boldsymbol{k}) \propto P(\boldsymbol{k})^{-1}$$

where $P(\boldsymbol{k}) \geq 0$ is a symmetric polynomial

Matérn/Whittle is Markov for $\alpha = 1, 2, 3 \ldots$: $S_u(\boldsymbol{k}) \propto (\kappa^2 + \|\boldsymbol{k}\|^2)^{-\alpha}$

GMRF	Covariance on $\mathbb{R}^2$	
SAR(1) CAR(2)	$\propto \kappa\|\boldsymbol{u}\| K_1(\kappa\|\boldsymbol{s}-\boldsymbol{s}'\|)$	Whittle (1954)
CAR(1)	$\frac{1}{2\pi} K_0(\kappa\|\boldsymbol{s}-\boldsymbol{s}'\|)$	Besag (1981)
ICAR(1)	$-\frac{1}{2\pi}\log(\|\boldsymbol{s}-\boldsymbol{s}'\|)$	Besag & Mondal (2005)

On lattices, classical CAR $\rightarrow$ Matérn models (limits of).

Hilbert space approximation ("The SPDE approach" from Lindgren et al, 2011)

Can extend to (non-)stationary SPDE models on irregular triangulations.

From continuous to discrete

We want to construct finite dimensional approximations to the distribution of $u(\cdot)$, where

$$\left[\left\langle f_i, (\kappa^2 - \nabla \cdot \nabla)^{\alpha/2} u(\cdot)\right\rangle_D, \, i = 1, \ldots, m\right] \stackrel{d}{=} \left[\left\langle f_i, \mathcal{W}(\cdot)\right\rangle_D, \, i = 1, \ldots, m\right]$$

for all finite collections of test functions $f_i \in H_{\mathcal{R}_\mathcal{W}}(D)$.

A finite basis expansion

$$u(\mathbf{s}) = \sum_{j=1}^{n} \psi_j(\mathbf{s}) u_j$$

can only hope to achieve this for a subspace of size n.

Two main approaches:

- Galerkin: $\{f_i = \psi_i, \, i = 1, \ldots, n\}$
- Least squares: $\{f_i = (\kappa^2 - \nabla \cdot \nabla)^{\alpha/2} \psi_i, \, i = 1, \ldots, n\}$

We use least squares for $\alpha = 1$, Galerkin for $\alpha = 2$, and a recursion for $\alpha \geq 3$.

Stochastic Green's first identity

On any sufficiently smooth manifold domain D,

$$\langle f, -\nabla \cdot \nabla g \rangle_D = \langle \nabla f, \nabla g \rangle_D - \langle f, \partial_n g \rangle_{\partial D}$$

holds, even if either ∇f or $-\nabla \cdot \nabla g$ are as generalised as white noise.

For now, we'll impose deterministic Neumann boundary conditions, informally $\partial_n u(\mathbf{s}) = 0$ for all $\mathbf{s} \in \partial D$. For $\alpha = 2$ and Galerkin,

$$\begin{aligned}\left\langle \psi_i, (\kappa^2 - \nabla \cdot \nabla) \sum_j \psi_j u_j \right\rangle_D &= \sum_j \left\{ \kappa^2 \langle \psi_i, \psi_j \rangle_D + \langle \nabla \psi_i, \nabla \psi_j \rangle_D \right\} u_j \\ &= (\kappa^2 \boldsymbol{C} + \boldsymbol{G}) \boldsymbol{u}\end{aligned}$$

The covariance for the RHS of the SPDE is

$$\left[\mathsf{Cov}(\langle \psi_i, \mathcal{W} \rangle_D, \langle \psi_j, \mathcal{W} \rangle_D \right] = \left[\langle \psi_i, \psi_j \rangle_D \right] = \boldsymbol{C}$$

by the definition of $\mathcal{W}$.

We seek $\boldsymbol{u} \sim \mathsf{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ such that $\mathsf{Var}\{(\kappa^2\boldsymbol{C} + \boldsymbol{G})\boldsymbol{u}\} = \boldsymbol{C}$:

$$(\kappa^2\boldsymbol{C} + \boldsymbol{G})\boldsymbol{\Sigma}(\kappa^2\boldsymbol{C} + \boldsymbol{G}) = \boldsymbol{C}$$
$$\boldsymbol{\Sigma} = (\kappa^2\boldsymbol{C} + \boldsymbol{G})^{-1}\boldsymbol{C}(\kappa^2\boldsymbol{C} + \boldsymbol{G})^{-1}$$

If ψ_i are piecewise linear on a triangulation of D, then $\boldsymbol{C}$ and $\boldsymbol{G}$ are both very sparse, and in addition, $\boldsymbol{C} = \mathrm{diag}(\langle \psi_i, 1 \rangle_D)$ is a valid approximation. Then, the *precision* matrix is also sparse,

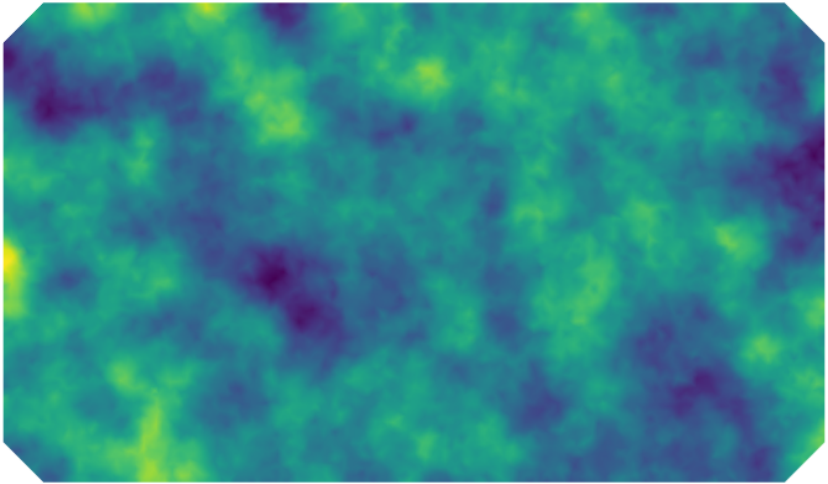
$$\boldsymbol{Q} = (\kappa^2\boldsymbol{C} + \boldsymbol{G})\boldsymbol{C}^{-1}(\kappa^2\boldsymbol{C} + \boldsymbol{G})$$

and $\boldsymbol{u}$ is Markov on the adjacency graph given by the non-zero structure of $\boldsymbol{Q}$.

Least squares and Galerkin recursion gives precisions for all $\alpha = 1, 2, \ldots$:

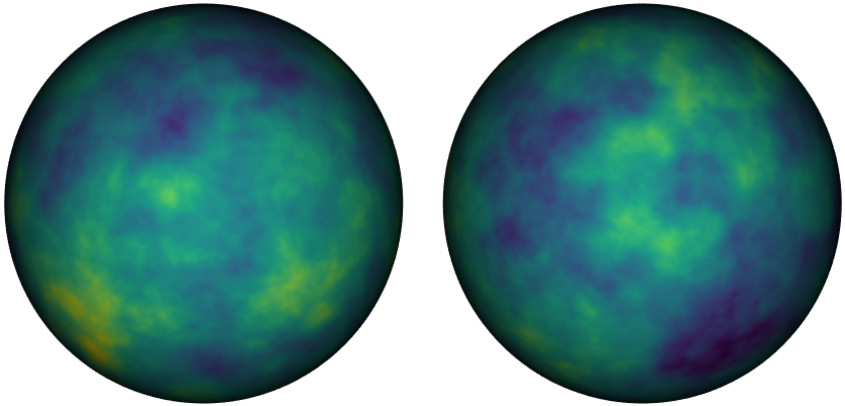
- $\boldsymbol{Q}_1 = (\kappa^2\boldsymbol{C} + \boldsymbol{G})$
- $\boldsymbol{Q}_2 = (\kappa^2\boldsymbol{C} + \boldsymbol{G})\boldsymbol{C}^{-1}(\kappa^2\boldsymbol{C} + \boldsymbol{G}) = \kappa^4\boldsymbol{C} + 2\kappa^2\boldsymbol{G} + \boldsymbol{G}\boldsymbol{C}^{-1}\boldsymbol{G}$
- $\boldsymbol{Q}_\alpha = (\kappa^2\boldsymbol{C} + \boldsymbol{G})\boldsymbol{C}^{-1}\boldsymbol{Q}_{\alpha-2}\boldsymbol{C}^{-1}(\kappa^2\boldsymbol{C} + \boldsymbol{G})$
- Any $\alpha \geq 0$: $\boldsymbol{Q}_\alpha = \boldsymbol{C}^{1/2}\left\{\boldsymbol{C}^{-1/2}(\kappa^2\boldsymbol{C} + \boldsymbol{G})\boldsymbol{C}^{-1/2}\right\}^\alpha \boldsymbol{C}^{1/2}$
  (non-sparse for non-integer α)

SPDE/GMRF realisations and non-stationary models



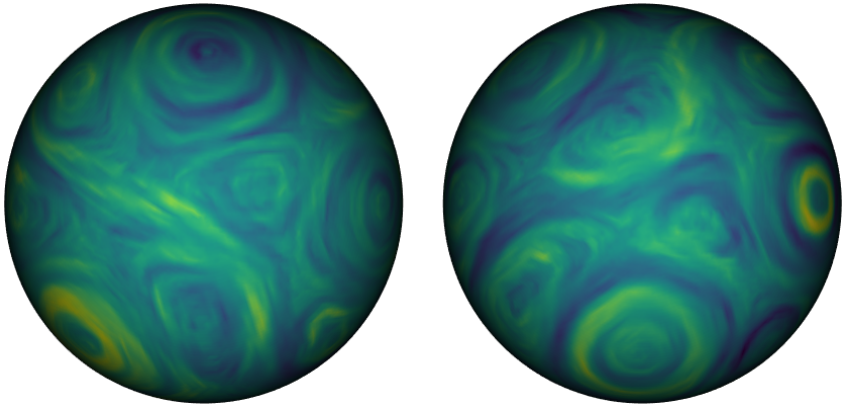
$$(\kappa^2 - \nabla \cdot \nabla) u(\boldsymbol{s}) = \mathcal{W}(\boldsymbol{s}), \quad \boldsymbol{s} \in D$$

SPDE/GMRF realisations and non-stationary models



$$(\kappa^2 - \nabla \cdot \nabla) u(\boldsymbol{s}) = \mathcal{W}(\boldsymbol{s}), \quad \boldsymbol{s} \in D = \mathbb{S}^2$$

Markov does *not* mean that dependence is only local



$$(\kappa(\boldsymbol{s})^2 - \nabla \cdot \boldsymbol{H}(\boldsymbol{s})\nabla)u(\boldsymbol{s}) = \kappa(\boldsymbol{s})\mathcal{W}(\boldsymbol{s}), \quad \boldsymbol{s} \in \Omega$$

Hierarchical models

Continuous Markovian spatial models (Lindgren et al, 2011)

Local basis: $u(\boldsymbol{s}) = \sum_k \psi_k(\boldsymbol{s}) u_k$, (compact, piecewise linear)

Basis weights: $\boldsymbol{u} \sim \mathsf{N}(\boldsymbol{0}, \boldsymbol{Q}^{-1})$, sparse $\boldsymbol{Q}$ based on an SPDE

Special case: $(\kappa^2 - \nabla \cdot \nabla) u(\boldsymbol{s}) = \mathcal{W}(\boldsymbol{s}), \quad \boldsymbol{s} \in \Omega$

Precision: $\boldsymbol{Q} = \kappa^4 \boldsymbol{C} + 2\kappa^2 \boldsymbol{G} + \boldsymbol{G}_2$ $\quad (\kappa^4 + 2\kappa^2 |\boldsymbol{\omega}|^2 + |\boldsymbol{\omega}|^4)$

Conditional distribution in a jointly Gaussian model

$$\boldsymbol{u} \sim \mathsf{N}(\boldsymbol{\mu}_u, \boldsymbol{Q}_u^{-1}), \quad \boldsymbol{y}|\boldsymbol{u} \sim \mathsf{N}(\boldsymbol{A}\boldsymbol{u}, \boldsymbol{Q}_{y|u}^{-1}) \qquad (A_{ij} = \psi_j(\boldsymbol{s}_i))$$

$$\boldsymbol{u}|\boldsymbol{y} \sim \mathsf{N}(\boldsymbol{\mu}_{u|y}, \boldsymbol{Q}_{u|y}^{-1})$$

$$\boldsymbol{Q}_{u|y} = \boldsymbol{Q}_u + \boldsymbol{A}^T \boldsymbol{Q}_{y|u} \boldsymbol{A} \quad (\sim\text{"Sparse iff } \psi_k \text{ have compact support"})$$

$$\boldsymbol{\mu}_{u|y} = \boldsymbol{\mu}_u + \boldsymbol{Q}_{u|y}^{-1} \boldsymbol{A}^T \boldsymbol{Q}_{y|u} (\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\mu}_u)$$

The computational GMRF work-horse

Cholesky decomposition (Cholesky, 1924)

$$\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^\top, \quad \boldsymbol{L} \text{ lower triangular } (\sim \mathcal{O}(n^{(d+1)/2}) \text{ for } d = 1, 2, 3)$$

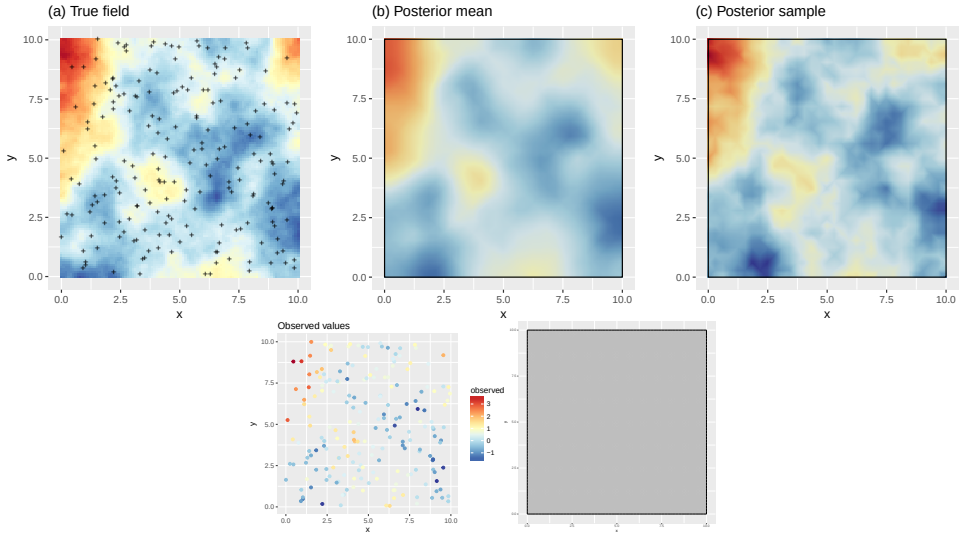
$$\boldsymbol{Q}^{-1}\boldsymbol{x} = \boldsymbol{L}^{-\top}\boldsymbol{L}^{-1}\boldsymbol{x}, \quad \text{via forward/backward substitution}$$

$$\log\det \boldsymbol{Q} = 2\log\det \boldsymbol{L} = 2\sum_i \log L_{ii}$$

André-Louis Cholesky (1875–1918)

"He invented, for the solution of the condition equations in the method of least squares, a very ingenious computational procedure which immediately proved extremely useful, and which most assuredly would have great benefits for all geodesists, if it were published some day." (Euology by Commandant Benoit, 1922)

Example: 2D georeferenced data



Software history

- `GMRFLib` (C library, early 2000s) Efficient GMRF computation
- `inla` (C program, mid-late 2000s) Integrated Nested Laplace Approximation, built on GMRFLib. Initially meant to provide good MCMC proposals, but bias $<$ MC error!
- `INLA` (R package, late 2000s) R interface to `inla`
- SPDE models added to INLA (2010/11)
- `inlabru` (R package, ca 2015-2017) Improved and extended model specification interface to `inla`, built on `INLA`
- `fmesher` (C++ library 2010, R package 2023) Extracting and unifying the internal support functions from `INLA` for mesh building, geographical coordinate system transformation, function space handling, and finite element calculations
- `fdmr` (R package, 2023) Interface and tools on top of `inlabru`/`R-INLA`

All of these libraries/programs/packages are in active development, with the inla method getting a major method update ca 2022, and the core computing libraries being updated for more efficient parallel computing.

A major `inlabru` internal code refactoring took place ca 2018-2022, allowing new features to be more easily added, e.g. transparent handling of geographical coordinates via `sf` and `terra`.

Models in theory and practice

- The class of generalised additive models (GLM/GLMM/GAM/etc) is large
- The R-INLA package implements fast MCMC-free Bayesian inference for latent Gaussian models of GAM type
- R-INLA handles construction of GMRF approximations to SPDE models of Matérn type as well as graph and lattice based models, but more general spatial models can be defined in R code by the user (via `inla.rgeneric.define` and `inla.cgeneric.define`)
- `inlabru` has greatly simplified the specification of complex spatial and spatio-temporal models:
  - Simplify specification of complex latent model components
  - Simplify specification of linked multi-observation models
  - Extend the model class to include mild but non-trivial predictor non-linearities
  - Do this without re-implementing R-INLA from scratch
  - Make the simple things easy, and the complex things possible
  - Goal: make every building block interoperable with every other building block

Latent Gaussian models

Hierarchical model with latent jointly Gaussian variables

$$\boldsymbol{\theta} \sim p(\boldsymbol{\theta}) \quad \text{(covariance parameters)}$$
$$(\boldsymbol{u} \mid \boldsymbol{\theta}) \sim \mathsf{N}(\boldsymbol{\mu}_u, \boldsymbol{Q}_u^{-1}) \quad \text{(latent Gaussian variables)}$$
$$(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{\theta}) \sim p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{\theta}) \quad \text{(observation model)}$$

We are interested in the posterior densities $p(\boldsymbol{\theta} \mid \boldsymbol{y})$, $p(\boldsymbol{u} \mid \boldsymbol{y})$ and $p(u_i \mid \boldsymbol{y})$.

Approximate conditional posterior distribution

Let $\widehat{\boldsymbol{u}}(\boldsymbol{\theta})$ be the mode of the posterior density $p(\boldsymbol{u} \mid \boldsymbol{y}, \boldsymbol{\theta}) \propto p(\boldsymbol{u} \mid \boldsymbol{\theta}) p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{\theta})$. Construct an approximate conditional posterior distribution, via Newton optimisation for $\boldsymbol{u}$ given $\boldsymbol{\theta}$:

$$p_G(\boldsymbol{u} \mid \boldsymbol{y}, \boldsymbol{\theta}) \sim \mathsf{N}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{Q}}^{-1})$$
$$\mathbf{0} = \nabla_{\boldsymbol{u}} \left\{ \ln p(\boldsymbol{u} \mid \boldsymbol{\theta}) + \ln p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{\theta}) \right\} \big|_{\boldsymbol{u} = \widehat{\boldsymbol{\mu}}(\boldsymbol{\theta})}$$
$$\widehat{\boldsymbol{Q}} = \boldsymbol{Q}_u - \nabla_{\boldsymbol{u}}^2 \ln p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{\theta}) \big|_{\boldsymbol{u} = \widehat{\boldsymbol{\mu}}_{\boldsymbol{\theta}}}$$

Integrated Nested Laplace Approximation (INLA; Rue, Martino, Chopin, 2009)

1. Estimate the posterior mode for $p(\boldsymbol{\theta}|\boldsymbol{y})$ by optimisation of the approximation

$$\widehat{p}(\boldsymbol{\theta} \mid \boldsymbol{y}) \propto \left. \frac{p(\boldsymbol{\theta})p(\boldsymbol{u} \mid \boldsymbol{\theta})p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{\theta})}{p_G(\boldsymbol{u} \mid \boldsymbol{y}, \boldsymbol{\theta})} \right|_{\boldsymbol{u}=\widehat{\boldsymbol{\mu}}(\boldsymbol{\theta})}$$

   where $p_G(\boldsymbol{u} \mid \boldsymbol{y}, \boldsymbol{\theta})$ is a Gaussian approximation matching the low order derivatives at the mode $\widehat{\boldsymbol{\mu}}(\boldsymbol{\theta})$ of the exact conditional log-posterior for $\boldsymbol{u}$. (In a fully Gaussian model this is exact.) This is a Laplace approximation of $p(\boldsymbol{\theta} \mid \boldsymbol{y})$.

2. Numerical integration for the marginal latent variables
   - Construct a numerical integration grid/scheme $(\boldsymbol{\theta}_k, w_k)$ for $\boldsymbol{\theta}$, where w_k are integration weights;
   - Construct $p_{GG}(u_i \mid \boldsymbol{y}, \boldsymbol{\theta}_k)$ as Variational Bayes approximations of the marginal conditional posterior densities, integrating out $\boldsymbol{u}_{-i} = \{u_j, j \neq i\}$.
   - Combine to form marginal posterior density approximations:

$$\widehat{p}(u_i \mid \boldsymbol{y}) = \sum_k p_{GG}(u_i \mid \boldsymbol{y}, \boldsymbol{\theta}_k)\, \widehat{p}(\boldsymbol{\theta}_k \mid \boldsymbol{y})\, w_k$$

inlabru software interface concepts

- Model components are declared similarly to R-INLA:

```
# INLA:
~ covar + f(name, model = ...)
# inlabru
~ covar + name(input, model = ...)
~ covar # is translated into...
~ covar(covar, model = "linear")
~ name(1) # Used for intercept-like components
```

- In R-INLA, $\boldsymbol{\eta} = \boldsymbol{A}\boldsymbol{u} = \boldsymbol{A}_0 \sum_{k=1}^{K} \boldsymbol{A}_k \boldsymbol{u}_k$, where the rows of $\boldsymbol{A}_k$ only extract individual elements from each $\boldsymbol{u}_k$, and the overall $\boldsymbol{A}_0$ is user defined (via `inla.stack()`).
- In inlabru, $\boldsymbol{\eta} = h(\boldsymbol{u}_1, \ldots, \boldsymbol{u_K}, \boldsymbol{A}_1\boldsymbol{u}_1, \ldots, \boldsymbol{A}_K\boldsymbol{u_K})$, where $h(\cdot)$ is a general R expression of named latent components $\boldsymbol{u}_k$ and intermediate "effects" $\boldsymbol{A}_k\boldsymbol{u}_k$
- $\boldsymbol{A}_k$ by default acts either as in R-INLA, or is determined by a *mapper* method. Predefined default mappers include e.g. spatial evaluation of SPDE/GRMF models that map between coordinates and meshes, and mappers that combine other mappers (used to combine main/group/replicate for all components)

Input mappers

- Each named component has main/group/replicate *inputs*, that are given to the mappers to evaluate $\boldsymbol{A}_k$. For a given latent *state*, the resulting *effect* values are made available to the predictor expression.

```
bru_get_mapper() # Obtain default mapper for a model object
bru_mapper_index(n) # Basic index mapping
bru_mapper_linear() # Basic linear mapping
bru_mapper_matrix(labels) # Basic linear multivariable mapping
bru_mapper_factor(values, factor_mapping) # Factor variable mapping
bru_mapper_multi(mappers) # Kronecker product components
bru_mapper_collect(mappers, hidden) # For concatenated components, like bym

bru_mapper_const() # Constants
bru_mapper_scale() # Fixed scaling
bru_mapper_marginal() # Marginal distribution transformation
bru_mapper_aggregate()/logsumexp() # Weighted block-wise sum or log-sum-exp
bru_mapper_pipe() # Composition of mappers

bru_mapper.fm_mesh_2d(mesh) # 2D and spherical mesh mappings
bru_mapper.fm_mesh_1d(mesh) # Interval and cyclic interval mappings
```

- Model component definition examples:

```
comp <- ~ -1 + field(cbind(easting, northing), model = spde) + param(1) # Raw data
comp <- ~ -1 + field(geometry, model = spde) + param(1) # sf data
```

- Predictor formula examples, including naming of the response variable:

```
form1 <- my_counts ~ param + field
form2 <- response ~ exp(param) + exp(field)
```

- Main method call structure:

```
bru(components = comp,
    like(formula = form1, family = "poisson", data = data1),
    like(formula = form2, family = "normal", data = data2))
```

- Simplified notations for common special cases;

  `formula = response ~ .`

  gives the full additive model of all the available components, or

  `components = response ~ Intercept(1) + field(...`

Plain INLA code for a separable space-time model

```
matern <- inla.spde2.pcmatern(mesh, ...)

field_A <- inla.spde.make.A(mesh,
                            st_coordinates(data),
                            group = data$year - min(data$year) + 1,
                            n.group = 10)
stk <- inla.stack(data = list(response = data$response),
                  A = list(field_A, 1),
                  effects = list(field_index, list(covar = data$covar)))

formula <- response ~ 1 + covar +
  f(field, model = matern, group = field_group, control.group = ...)

fit <- inla(formula = formula,
            data = inla.stack.data(stk, matern = matern),
            family = "normal",
            control.predictor = list(A = inla.stack.A(stk)))
```

inlabru code for a separable space-time model

```
matern <- inla.spde2.pcmatern(mesh, ...)

year_mapper <- bru_mapper(fm_mesh_1d(sort(unique(data$year))), indexed = TRUE)

comp <- ~ Intercept(1) + covar +
  field(geometry, model = matern, group = year, group_mapper = year_mapper,
        control.group = ...)

fit <- bru(components = comp, like(response ~ ., data = data, family = "normal"))
```

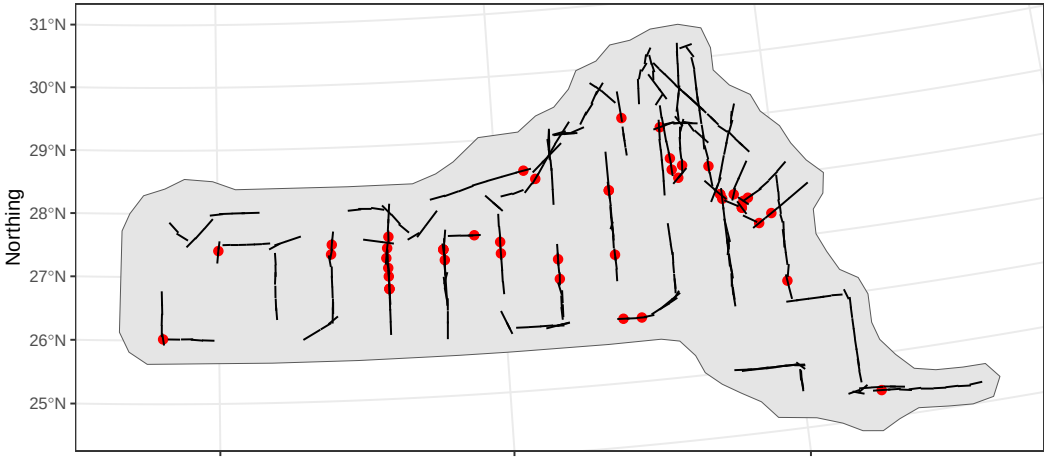
inlabru code for a non-separable space-time model

```
model <- INLAspacetime::stModel.define(...)

comp <- ~ Intercept(1) + covar +
  field(list(space = geometry, time = year), model = model)

fit <- bru(components = comp, like(response ~ ., data = data, family = "normal"))
```

`fdmr` is built on top of `inlabru`/`fmesher`/`INLA`.

Example: Thinned Poisson point processes

We want to model the presence of groups of dolphins using a Log-Gaussian Cox Process (LGCP)
However, when surveying dolphins (`points`) from a ship travelling along lines (`transects`), the probability of detecting a group of animals depends their distance distance from the ship.



Example: Thinned Poisson point processes

We want to model the presence of groups of dolphins using a Log-Gaussian Cox Process (LGCP)
However, when surveying dolphins from a ship travelling along lines (*transects*), the probability of detecting a group of animals depends their distance distance from the ship, e.g. via

$$\mathrm{P}(\text{detection}) = 1 - \exp\left(-\frac{\sigma}{\text{distance}}\right) \quad \text{(hazard rate model)}$$

This results in a *thinned* Poisson process model on $(\text{space}, \text{distance})$ along the transects:

$$\log(\lambda(\boldsymbol{s}, \text{distance})) = \text{Intercept} + \text{field}(\boldsymbol{s}) + \log\left[\mathrm{P}(\text{detection at } \boldsymbol{s} \mid \text{distance}, \sigma)\right] + \log(2)$$

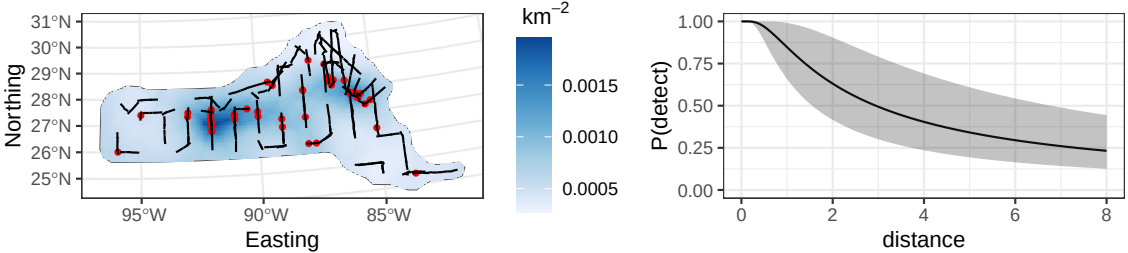
inlabru knows how to construct the Poisson process likelihood along lines and on polygons, and kronecker spaces (line $\times$ distance)
We can define $\log(\sigma)$ as a latent Gaussian variable and iteratively linearise. The non-linearity is mild, and the iterative INLA method converges.

```r
log_det_prob <- function(distance, sigma) {
  log1p(-exp(-sigma / distance))
}

comp <- ~ field(geometry, model = matern) + log_sig(1, prec.linear = 2) + Intercept(1)
form <- geometry + distance ~
  Intercept + field + log_det_prob(distance, exp(log_sig)) + log(2)

fit <- bru(
  components = comp,
  like(
    family = "cp", formula = form,
    data = points, # sfc_POINT
    samplers = transects, # sfc_LINESTRING
    domain = list(
      geometry = mesh,
      distance = fm_mesh_1d(seq(0, 8, length.out = 30))
    )
  ),
  options = list(bru_verbose = 0)
)
```

Posterior prediction method

```
pred_points <- fm_pixels(mesh, dims = c(200, 100), mask = region_of_interest)
pred <- predict(fit, pred_points, ~ exp(field + Intercept))

det_prob <- function(distance, sigma) { 1 - exp(-sigma / distance) }
pred_dist <- data.frame(distance = seq(0, 8, length.out = 100))
det_prob <- predict(fit, pred_dist, ~ det_prob(distance, exp(log_sig)))
```

```
ggplot() + gg(pred, geom = "tile") + gg(transects) + gg(region_of_interest) + ...
```



Data level prediction

47 groups were seen. How many would be seen along the transects under perfect detection?

```
predpts_transect <- fm_int(mesh, transects)
Lambda_transect <- predict(fit, predpts_transect,
                           ~ 16 * sum(weight * exp(field + Intercept)))
```

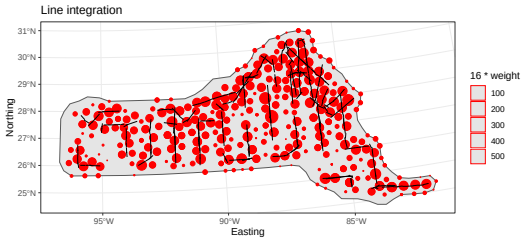
mean	sd	q0.025	q0.5	q0.975	median	mean.mc_std_err	sd.mc_std_err
100.7835	27.22059	64.734	95.42889	160.0679	95.42889	2.722059	2.076135

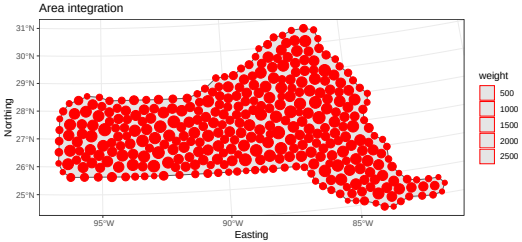
How many would be seen under perfect detection across the whole study area?

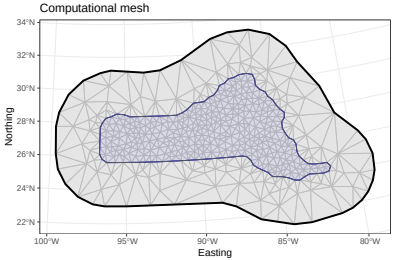
```
predpts <- fm_int(mesh, samplers = region_of_interest)
Lambda <- predict(fit, predpts, ~ sum(weight * exp(field + Intercept)))
```

mean	sd	q0.025	q0.5	q0.975	median	mean.mc_std_err	sd.mc_std_err
365.5898	116.2963	203.7757	347.1875	585.2433	347.1875	11.62963	15.24792

Integration points





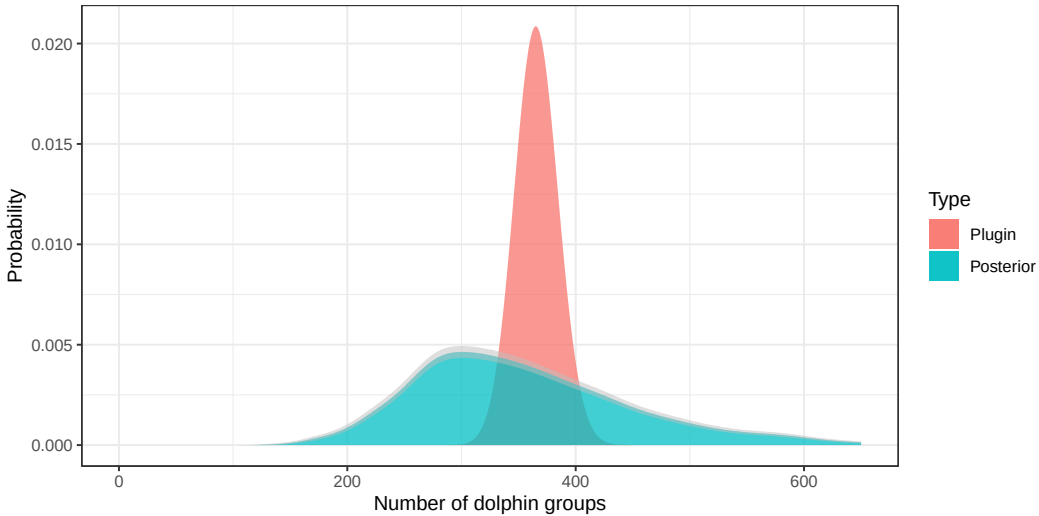


Complex prediction expressions

What's the predictive distribution of group counts?

```
Ns <- seq(50, 650, by = 1)
Nest <- predict(
  fit,
  predpts,
  ~ data.frame(
    N = Ns,
    density = dpois(Ns, lambda = sum(weight * exp(field + Intercept)))
  ),
  n.samples = 2500
)

Nest$plugin_estimate <- dpois(Nest$N, lambda = Lambda$mean)
```

Full posterior prediction uncertainty vs plugin prediction



General aggregation modelling

- Misaligned and aggregated data can be handled by modelling on a continuous domain and linking each observation model into that (with Luisa Parkinson, Man Ho Suen, Andy Seaton, Elias Krainski)
- Related work (with Christopher Merchant and Xue Wang):
  Multi-band satellite data with nadir and oblique views, with non-rectangular "pixels":

$$\mathsf{E}(\mathsf{measured}(\mathsf{pixel},\mathsf{band})) = \left(\frac{1}{|D_{\mathsf{pixel}}|}\int_{D_{\mathsf{pixel}}} \mathsf{conversion}\big[\mathsf{SST}(\boldsymbol{s}),\mathsf{TCWV}(\boldsymbol{s}),\mathsf{band}\big]^b \,\mathrm{d}\boldsymbol{s}\right)^{1/b}$$

- Both SST and TCWV are unknown spatial fields and b is an unknown parameter
- `conversion` is a function evaluated on a grid of SST and TCWV for each frequency band

```
components <- ~ SST(geometry, ...) + TCWV(geometry, ...) + b(...)
integ <- fm_int(domain = list(geometry = mesh, band = seq_len(n_bands)),
                samplers = observed_polygons_and_bands)
agg <- bru_mapper_aggregate(rescale = TRUE)
formula <- measured ~ ibm_eval(agg, list(block = .block, weights = weight),
                               conversion(SST, TCWV, band)^b)^(1/b)
```

Extensions and projects in progress

- (w Francesco Serafini and Mark Naylor) `ETAS.inlabru` for temporal Hawkes processes for earthquake forecasting; self-exciting Poisson processes with $\lambda(\boldsymbol{s},t) = \mu(\boldsymbol{s},t,\boldsymbol{u}) + \sum_{i;t_i<t} h(\boldsymbol{s}-\boldsymbol{s}_i, t-t_i, \boldsymbol{u})$ which is not log-linear.
- (w Elias Krainski) Extending the supported set of R-INLA models (survival models, etc)
- Copulas and transformation models; Version 2.9.0+ will support inbuilt marginal transformation of $N(0,1)$ components into fixed non-Gaussians: `effect` $= F^{-1}[\Phi(u);\theta]$

```
comp <- ~ field(geometry, model = matern) + Intercept(1) +
  sigma(1, prec.linear = 1, marginal = bru_mapper_marginal(qexp, rate = 1/8))
form <- geometry + distance ~
  Intercept + field + log_det_prob(distance, sigma) + log(2)
```

Experimental example:

```
comp <- ~ Intercept(1) + field(geometry, model = matern) +
  field2(geometry, model = "iid", hyper=list(prec=list(initial = 0, fixed = TRUE)))
marg <- bru_mapper_marginal(qexp)
form <- ... ~ ... +
  ibm_eval(marg, input = list(rate = exp(Intercept + field)), state = field2)
```

Summary

- INLA and inlabru allows a wide variety of generalisations of GAMs to be specified
- Whether the the model and data form a well-posed problem and/or has any relation to reality is the user's responsibility.
- The software may help diagnose some issues;
  - Posterior prediction and model assessment
  - How accurate are the linearised posteriors? Future diagnostic metric:

$$\mathrm{E}_{\boldsymbol{u}\sim\overline{p}(\boldsymbol{u}|\boldsymbol{y})}\left(\log\left(\frac{\overline{p}(\boldsymbol{u}|\boldsymbol{y},\boldsymbol{\theta})}{\widetilde{p}(\boldsymbol{u}|\boldsymbol{y},\boldsymbol{\theta})}\right)\right)$$

  - Optimization convergence plots (`bru_convergence_plot()`) and log output (`bru_log()`)
  - Detection of unintended incorrect user input

References

- Fabian E. Bachl, Finn Lindgren, David L. Borchers, and Janine B. Illian (2019) *inlabru: an R package for Bayesian spatial modelling from ecological survey data*, Methods in Ecology and Evolution, 10(6):760–766. https://doi.org/10.1111/2041-210X.13168
- The INLA package; https://www.r-inla.org
- CRAN packages: inlabru, fmesher, INLAspacetime, rSPDE, excursions
- Online documentation: https://inlabru-org.github.io/inlabru/ https://inlabru-org.github.io/fmesher/
- Package development, bug fixes, specific problem discussion pages: https://github.com/inlabru-org/inlabru/ https://github.com/inlabru-org/fmesher/
- inlabru: The Scottish INLA interface