inlabru: Bayesian spatial and spatio-temporal modelling in R

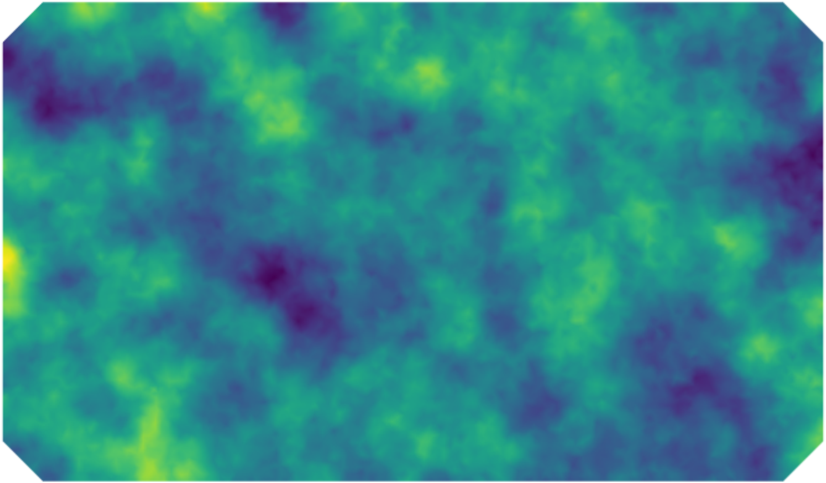
Finn Lindgren
finn.lindgren@ed.ac.uk

Bayes@Lund, 7 March 2024

Highlights

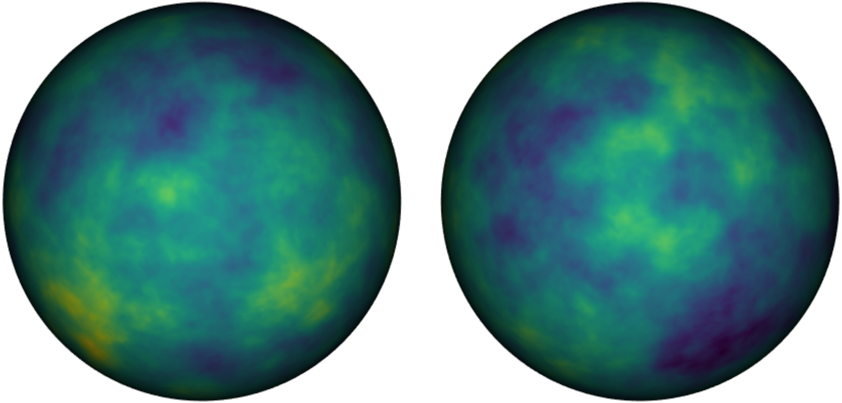
- Spatial dependence modelling with stochastic PDEs;
  precision operators instead of covariance specifications
- Fast Bayesian inference;
  INLA instead of MCMC
- User-friendly model specification;
  `inlabru` instead of `INLA`
- Example: Point process observations;
  non-linear predictor expressions

SPDE/GMRF realisations and non-stationary models



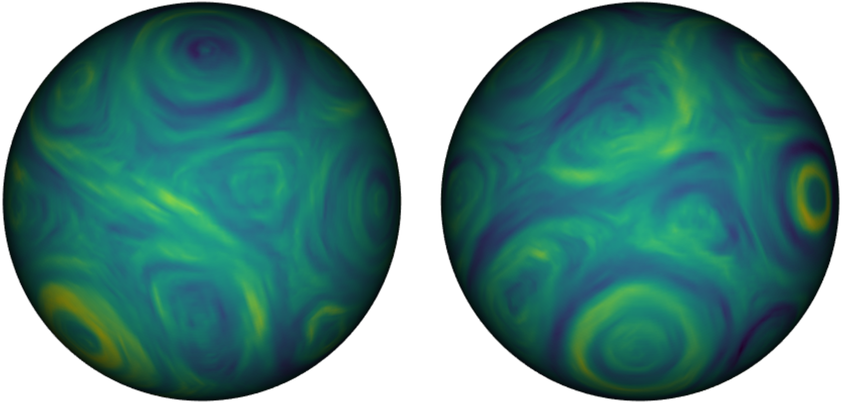
$$(\kappa^2 - \nabla \cdot \nabla) u(\boldsymbol{s}) = \mathcal{W}(\boldsymbol{s}), \quad \boldsymbol{s} \in D$$

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

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

Conditional distribution in a jointly Gaussian model

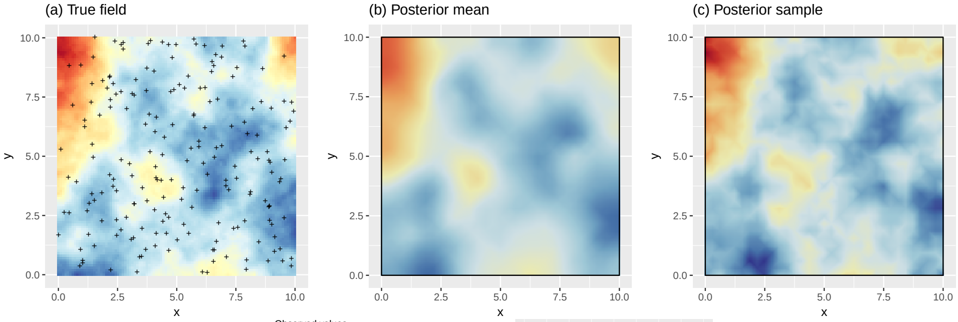
$$\boldsymbol{u} \sim \mathsf{N}(\boldsymbol{\mu}_u, \boldsymbol{Q}_u^{-1}), \quad \boldsymbol{y}|\boldsymbol{u} \sim \mathsf{N}(\boldsymbol{A}\boldsymbol{u}, \boldsymbol{Q}_{y|u}^{-1}) \qquad (A_{ij} = \psi_j(\boldsymbol{s}_i))$$

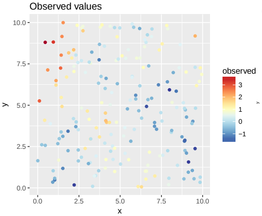
$$\boldsymbol{u}|\boldsymbol{y} \sim \mathsf{N}(\boldsymbol{\mu}_{u|y}, \boldsymbol{Q}_{u|y}^{-1})$$

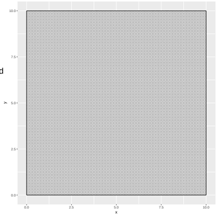
$$\boldsymbol{Q}_{u|y} = \boldsymbol{Q}_u + \boldsymbol{A}^T \boldsymbol{Q}_{y|u} \boldsymbol{A}$$ (∼"Sparse iff ψ_k have compact support")

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

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. Major method/memory/speed updates in ca 2022 and more expected in 2024.
- INLA (R package, late 2000s) R interface to inla.
- SPDE models added to INLA (2010/11)
  Proper fractional SPDEs in rSPDE (2021–23)
  Non-separable SPDEs in INLAspacetime (2022/23)
- excursions (2012/14) Joint credibility regions for contours and excursion sets
- inlabru (R package, ca 2015-2017) Improved and extended model specification interface
  Major internal code refactoring 2018-2022, including sf/terra support.
- 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
- MetricGraph (2023) Whittle-Matérn fields on metric graphs
- fdmr (R package, 2023) Interface and tools on top of inlabru/R-INLA

Software history

- GMRFLib (C library, early 2000s) Efficient GMRF computation
- inla (C program, mid-late 2000s) Integrated Nested Laplace Approximation, built on GMRFLib. Major method/memory/speed updates in ca 2022 and more expected in 2024.
- INLA (R package, late 2000s) R interface to inla.
- SPDE models added to INLA (2010/11)
  Proper fractional SPDEs in rSPDE (2021–23)
  Non-separable SPDEs in INLAspacetime (2022/23)
- excursions (2012/14) Joint credibility regions for contours and excursion sets
- inlabru (R package, ca 2015-2017) Improved and extended model specification interface
  Major internal code refactoring 2018-2022, including sf/terra support.
- 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
- MetricGraph (2023) Whittle-Matérn fields on metric graphs
- fdmr (R package, 2023) Interface and tools on top of inlabru/R-INLA

Software history

- GMRFLib (C library, early 2000s) Efficient GMRF computation
- inla (C program, mid-late 2000s) Integrated Nested Laplace Approximation, built on GMRFLib. Major method/memory/speed updates in ca 2022 and more expected in 2024.
- INLA (R package, late 2000s) R interface to inla.
- SPDE models added to INLA (2010/11)
  Proper fractional SPDEs in rSPDE (2021–23)
  Non-separable SPDEs in INLAspacetime (2022/23)
- excursions (2012/14) Joint credibility regions for contours and excursion sets
- inlabru (R package, ca 2015-2017) Improved and extended model specification interface
  Major internal code refactoring 2018-2022, including sf/terra support.
- 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
- MetricGraph (2023) Whittle-Matérn fields on metric graphs
- fdmr (R package, 2023) Interface and tools on top of inlabru/R-INLA

Software history

- GMRFLib (C library, early 2000s) Efficient GMRF computation
- inla (C program, mid-late 2000s) Integrated Nested Laplace Approximation, built on GMRFLib. Major method/memory/speed updates in ca 2022 and more expected in 2024.
- INLA (R package, late 2000s) R interface to inla.
- SPDE models added to INLA (2010/11)
  Proper fractional SPDEs in rSPDE (2021–23)
  Non-separable SPDEs in INLAspacetime (2022/23)
- excursions (2012/14) Joint credibility regions for contours and excursion sets
- inlabru (R package, ca 2015-2017) Improved and extended model specification interface
  Major internal code refactoring 2018-2022, including sf/terra support.
- 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
- MetricGraph (2023) Whittle-Matérn fields on metric graphs
- fdmr (R package, 2023) Interface and tools on top of inlabru/R-INLA

Software history

- GMRFLib (C library, early 2000s) Efficient GMRF computation
- inla (C program, mid-late 2000s) Integrated Nested Laplace Approximation, built on GMRFLib.
  Major method/memory/speed updates in ca 2022 and more expected in 2024.
- INLA (R package, late 2000s) R interface to inla.
- SPDE models added to INLA (2010/11)
  Proper fractional SPDEs in rSPDE (2021–23)
  Non-separable SPDEs in INLAspacetime (2022/23)
- excursions (2012/14) Joint credibility regions for contours and excursion sets
- inlabru (R package, ca 2015-2017) Improved and extended model specification interface
  Major internal code refactoring 2018-2022, including sf/terra support.
- 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
- MetricGraph (2023) Whittle-Matérn fields on metric graphs
- fdmr (R package, 2023) Interface and tools on top of inlabru/R-INLA

Software history

- `GMRFLib` (C library, early 2000s) Efficient GMRF computation
- `inla` (C program, mid-late 2000s) Integrated Nested Laplace Approximation, built on GMRFLib.
  Major method/memory/speed updates in ca 2022 and more expected in 2024.
- `INLA` (R package, late 2000s) R interface to `inla`.
- SPDE models added to INLA (2010/11)
  Proper fractional SPDEs in `rSPDE` (2021–23)
  Non-separable SPDEs in `INLAspacetime` (2022/23)
- `excursions` (2012/14) Joint credibility regions for contours and excursion sets
- `inlabru` (R package, ca 2015-2017) Improved and extended model specification interface
  Major internal code refactoring 2018-2022, including `sf/terra` support.
- `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
- `MetricGraph` (2023) Whittle-Matérn fields on metric graphs
- `fdmr` (R package, 2023) Interface and tools on top of `inlabru/R-INLA`

Software history

- GMRFLib (C library, early 2000s) Efficient GMRF computation
- inla (C program, mid-late 2000s) Integrated Nested Laplace Approximation, built on GMRFLib. Major method/memory/speed updates in ca 2022 and more expected in 2024.
- INLA (R package, late 2000s) R interface to inla.
- SPDE models added to INLA (2010/11)
  Proper fractional SPDEs in rSPDE (2021–23)
  Non-separable SPDEs in INLAspacetime (2022/23)
- excursions (2012/14) Joint credibility regions for contours and excursion sets
- inlabru (R package, ca 2015-2017) Improved and extended model specification interface
  Major internal code refactoring 2018-2022, including sf/terra support.
- 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
- MetricGraph (2023) Whittle-Matérn fields on metric graphs
- fdmr (R package, 2023) Interface and tools on top of inlabru/R-INLA

Software history

- GMRFLib (C library, early 2000s) Efficient GMRF computation
- inla (C program, mid-late 2000s) Integrated Nested Laplace Approximation, built on GMRFLib. Major method/memory/speed updates in ca 2022 and more expected in 2024.
- INLA (R package, late 2000s) R interface to inla.
- SPDE models added to INLA (2010/11)
  Proper fractional SPDEs in rSPDE (2021–23)
  Non-separable SPDEs in INLAspacetime (2022/23)
- excursions (2012/14) Joint credibility regions for contours and excursion sets
- inlabru (R package, ca 2015-2017) Improved and extended model specification interface
  Major internal code refactoring 2018-2022, including sf/terra support.
- 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
- MetricGraph (2023) Whittle-Matérn fields on metric graphs
- fdmr (R package, 2023) Interface and tools on top of inlabru/R-INLA

Software history

- GMRFLib (C library, early 2000s) Efficient GMRF computation
- inla (C program, mid-late 2000s) Integrated Nested Laplace Approximation, built on GMRFLib. Major method/memory/speed updates in ca 2022 and more expected in 2024.
- INLA (R package, late 2000s) R interface to inla.
- SPDE models added to INLA (2010/11)
  Proper fractional SPDEs in rSPDE (2021–23)
  Non-separable SPDEs in INLAspacetime (2022/23)
- excursions (2012/14) Joint credibility regions for contours and excursion sets
- inlabru (R package, ca 2015-2017) Improved and extended model specification interface
  Major internal code refactoring 2018-2022, including sf/terra support.
- 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
- MetricGraph (2023) Whittle-Matérn fields on metric graphs
- fdmr (R package, 2023) Interface and tools on top of inlabru/R-INLA

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

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 $\boldsymbol{\theta}$:

$$p_G(\boldsymbol{u} \mid \boldsymbol{y}, \boldsymbol{\theta}) \sim \mathsf{N}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{Q}}^{-1})$$

$$\boldsymbol{0} = \nabla_{\boldsymbol{u}} \left\{ \ln p(\boldsymbol{u} \mid \boldsymbol{\theta}) + \ln p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{\theta}) \right\} \Big|_{\boldsymbol{u}=\widehat{\boldsymbol{\mu}}(\boldsymbol{\theta})}$$

$$\widehat{\boldsymbol{Q}} = \boldsymbol{Q}_u - \nabla_{\boldsymbol{u}}^2 \ln p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{\theta}) \Big|_{\boldsymbol{u}=\widehat{\boldsymbol{\mu}}_{\boldsymbol{\theta}}}$$

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 $\boldsymbol{\theta}$:

$$p_G(\boldsymbol{u} \mid \boldsymbol{y}, \boldsymbol{\theta}) \sim \mathsf{N}(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{Q}}^{-1})$$

$$\boldsymbol{0} = \nabla_{\boldsymbol{u}} \left\{\ln p(\boldsymbol{u} \mid \boldsymbol{\theta}) + \ln p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{\theta})\right\}\Big|_{\boldsymbol{u}=\widehat{\boldsymbol{\mu}}(\boldsymbol{\theta})}$$

$$\widehat{\boldsymbol{Q}} = \boldsymbol{Q}_u - \nabla_{\boldsymbol{u}}^2 \ln p(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{\theta})\Big|_{\boldsymbol{u}=\widehat{\boldsymbol{\mu}}_{\boldsymbol{\theta}}}$$

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:

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

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, $\boldsymbol{\eta} = h(\boldsymbol{u}_1, \dots, \boldsymbol{u}_{\boldsymbol{K}}, \boldsymbol{A}_1\boldsymbol{u}_1, \dots, \boldsymbol{A}_K\boldsymbol{u}_{\boldsymbol{K}})$, 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 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

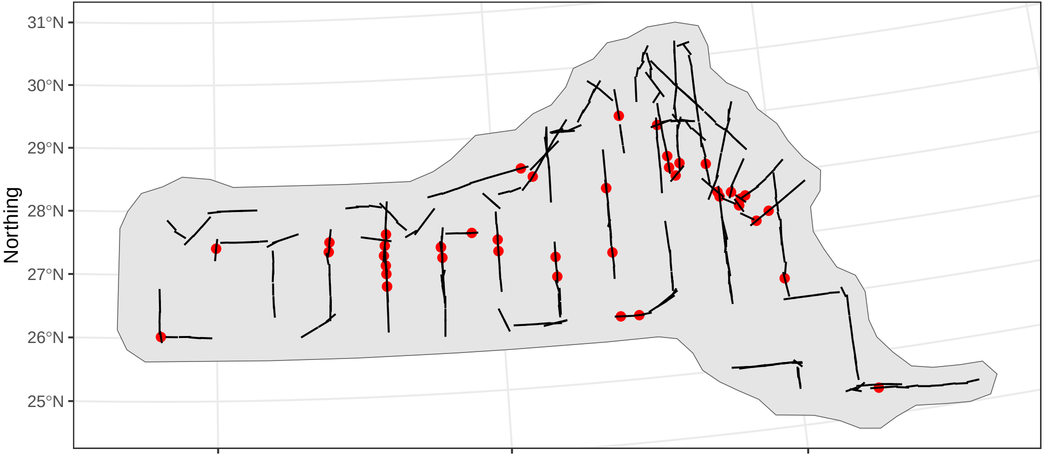
```
matern <- inla.spde2.pcmatern(mesh, ...)

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"))
```

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

The log-Gaussian Cox Process (LGCP) model

The LGCP model is defined by an intensity function $\lambda(\mathbf{s})$, defining the count of points Y in any region $\Omega \subseteq \mathbb{R}^2$ as $N(\Omega) \sim \text{Po}\left(\int_\Omega \lambda(\mathbf{s})\,\mathrm{d}\mathbf{s}\right)$, and the log-likelihood is

$$l(Y|\boldsymbol{\lambda}) = \sum_{i=1}^{N(\Omega)} \log \lambda(\boldsymbol{y}_i) - \int_\Omega \lambda(\mathbf{s})\,\mathrm{d}\mathbf{s}$$

For practical implementations, we use a numerical integration scheme for the integral, adapted to the resolution of the computational mesh for any Gaussian random field model included in the linear predictor, e.g. $\log \lambda(\mathbf{s}) = \text{Intercept} + \text{field}(\mathbf{s})$.

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 from the ship, e.g. via

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

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

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

inlabru knows how to construct the Poisson process likelihood along lines and on polygons, and kronecker spaces (line $\times$ distance)
We can define σ and ξ as transformed $\text{N}(0, 1)$ variables and iteratively linearise. The non-linearity is mild, and the iterative INLA method converges.

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 from the ship, e.g. via

$$\mathsf{P}(\text{detection}) = 1 - \exp\left[-\left(\frac{\sigma}{\text{distance}}\right)^{\xi}\right] \quad \text{(hazard rate model)}$$

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

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

inlabru knows how to construct the Poisson process likelihood along lines and on polygons, and kronecker spaces (line $\times$ distance)
We can define σ and ξ as transformed $\mathsf{N}(0,1)$ variables and iteratively linearise. The non-linearity is mild, and the iterative INLA method converges.

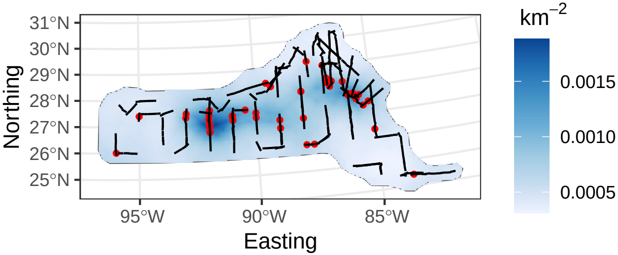
```
## Rcpp_interface.cc(144) 'mesh_loc' points imported.
## Rcpp_interface.cc(146) 'mesh_tv' points imported.
```

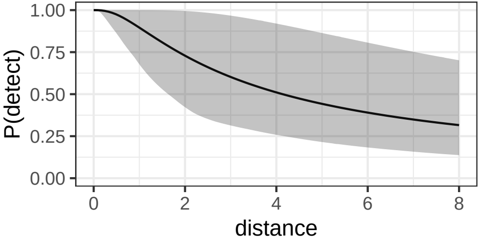
```
log_det_prob <- function(distance, sigma, xi) {
  log1p(-exp(-(sigma / distance)^xi))
}

comp <- ~ field(geometry, model = matern) +
  sigmainv(1, prec.linear = 1, marginal = bru_mapper_marginal(qexp, rate = 1)) +
  xi(1, prec.linear = 1, marginal = bru_mapper_marginal(qgamma, shape = 20, rate = 20)) +
  Intercept(1)
form <- geometry + distance ~
  Intercept + field + log_det_prob(distance, 1/sigmainv, xi) + log(2)

fit <- bru(
  components = comp,
  like(
    family = "cp", formula = form,
    data = points, # sfc_POINT
    samplers = transects, # sfc_LINESTRING
```

Posterior prediction method

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

```r
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)))
```

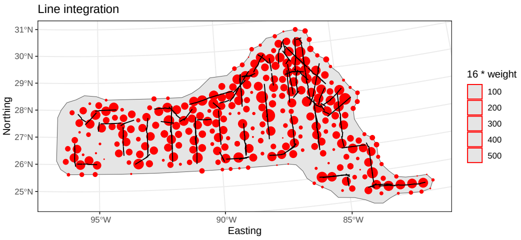
mean	sd	q0.025	q0.5	q0.975	median	mean.mc_std_err	sd.mc_std_err
91.15876	26.46627	51.37564	85.567	145.231	85.567	2.646627	1.7456

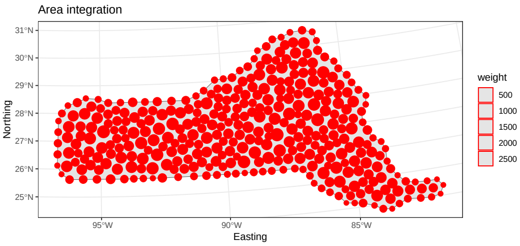
How many would be seen under perfect detection across the whole study area?

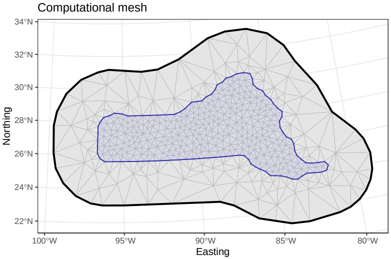
```
predpts <- fm_int(mesh, samplers = region_of_interest)
Lambda <- predict(fit, predpts, ~ sum(weight * exp(field + Intercept)))
```

mean	sd	q0.025	q0.5	q0.975	median	mean.mc_std_err	sd.mc_std_err
319.58	92.41475	177.3311	309.5392	508.2446	309.5392	9.241475	8.561235

Integration points





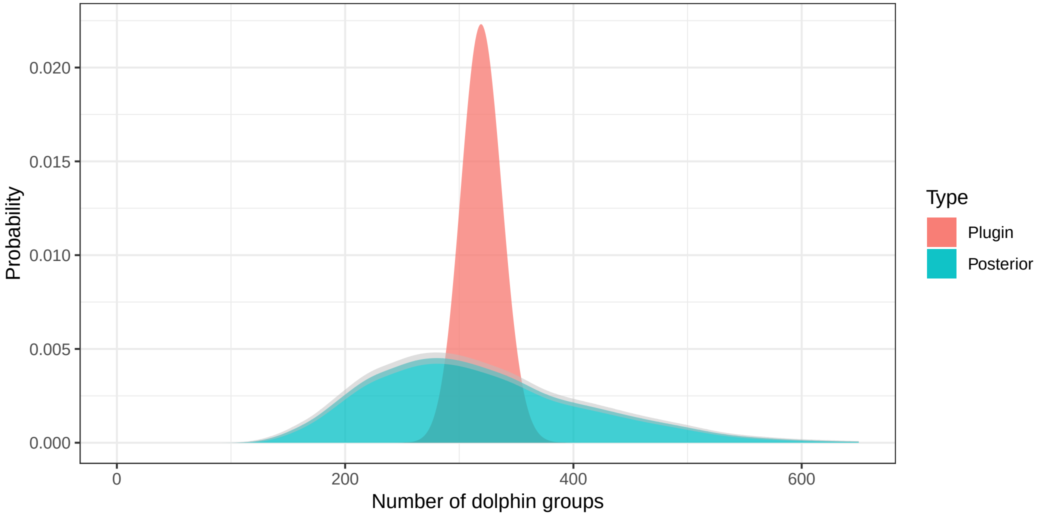


Complex prediction expressions

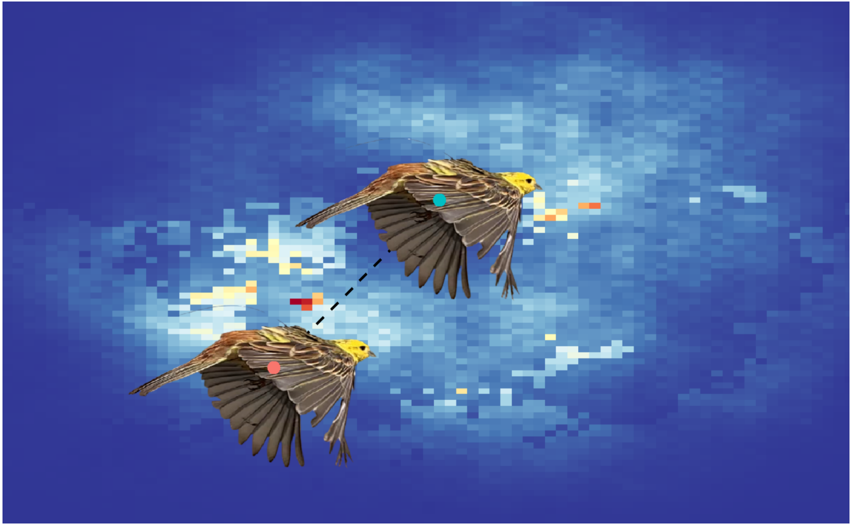
What's the predictive distribution of group counts?

```
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



Animal movement



Step selection analysis with telemetry data

Goal: Understand sequential movement decisions

- The general movement capacity of an animal. Expressed by a movement kernel:

$$K(\mathbf{y}_t|\mathbf{y}_{t-1},\mathbf{y}_{t-2},\boldsymbol{\theta}) = K_{\text{length}}(\mathbf{y}_t|\mathbf{y}_{t-1},\boldsymbol{\theta})K_{\text{angle}}(\mathbf{y}_t|\mathbf{y}_{t-1},\mathbf{y}_{t-2},\boldsymbol{\theta}), \quad \mathbf{y}_\cdot \in \mathcal{D} \subset \mathbb{R}^2$$

- Selection behaviour of the animal. Modelled by a resource selection function (RSF):

$$\xi(\mathbf{s}) = \exp[\eta(\mathbf{s})] = \exp[\beta_1 X_1(\mathbf{s}) + \ldots + \beta_p X_p(\mathbf{s}) + u(\mathbf{s})], \quad \mathbf{s} \in \mathcal{D}$$

  Spatially (or spatio-temporally) varying covariates $X_\cdot$ and a residual random field $u(\mathbf{s})$.

- Combined normalised conditional observation density function:

$$f_{t|<t}(\mathbf{y}_t|\boldsymbol{\theta},\eta) = \frac{K(\mathbf{y}_t|\mathbf{y}_{<t},\boldsymbol{\theta})\exp[\eta(\mathbf{y}_t)]}{\int_{\mathcal{D}} K(\mathbf{s}|\mathbf{y}_{<t},\boldsymbol{\theta})\exp[\eta(\mathbf{s})]\,\mathrm{d}\mathbf{s}}$$

Step selection analysis with telemetry data

Goal: Understand sequential movement decisions

- The general movement capacity of an animal. Expressed by a movement kernel:

$$K(\mathbf{y}_t|\mathbf{y}_{t-1},\mathbf{y}_{t-2},\boldsymbol{\theta}) = K_{\text{length}}(\mathbf{y}_t|\mathbf{y}_{t-1},\boldsymbol{\theta})K_{\text{angle}}(\mathbf{y}_t|\mathbf{y}_{t-1},\mathbf{y}_{t-2},\boldsymbol{\theta}), \quad \mathbf{y}_\cdot \in \mathcal{D} \subset \mathbb{R}^2$$

- Selection behaviour of the animal. Modelled by a resource selection function (RSF):

$$\xi(\mathbf{s}) = \exp[\eta(\mathbf{s})] = \exp[\beta_1 X_1(\mathbf{s}) + \ldots + \beta_p X_p(\mathbf{s}) + u(\mathbf{s})], \quad \mathbf{s} \in \mathcal{D}$$

  Spatially (or spatio-temporally) varying covariates $X_\cdot$ and a residual random field $u(\mathbf{s})$.

- Combined normalised conditional observation density function:

$$f_{t|<t}(\mathbf{y}_t|\boldsymbol{\theta},\eta) = \frac{K(\mathbf{y}_t|\mathbf{y}_{<t},\boldsymbol{\theta})\exp[\eta(\mathbf{y}_t)]}{\int_{\mathcal{D}} K(\mathbf{s}|\mathbf{y}_{<t},\boldsymbol{\theta})\exp[\eta(\mathbf{s})]\,\mathrm{d}\mathbf{s}}$$

Step selection analysis with telemetry data

Goal: Understand sequential movement decisions

- The general movement capacity of an animal. Expressed by a movement kernel:

$$K(\mathbf{y}_t|\mathbf{y}_{t-1},\mathbf{y}_{t-2},\boldsymbol{\theta}) = K_{\text{length}}(\mathbf{y}_t|\mathbf{y}_{t-1},\boldsymbol{\theta})K_{\text{angle}}(\mathbf{y}_t|\mathbf{y}_{t-1},\mathbf{y}_{t-2},\boldsymbol{\theta}), \quad \mathbf{y}_\cdot \in \mathcal{D} \subset \mathbb{R}^2$$

- Selection behaviour of the animal. Modelled by a resource selection function (RSF):

$$\xi(\mathbf{s}) = \exp[\eta(\mathbf{s})] = \exp[\beta_1 X_1(\mathbf{s}) + \ldots + \beta_p X_p(\mathbf{s}) + u(\mathbf{s})], \quad \mathbf{s} \in \mathcal{D}$$

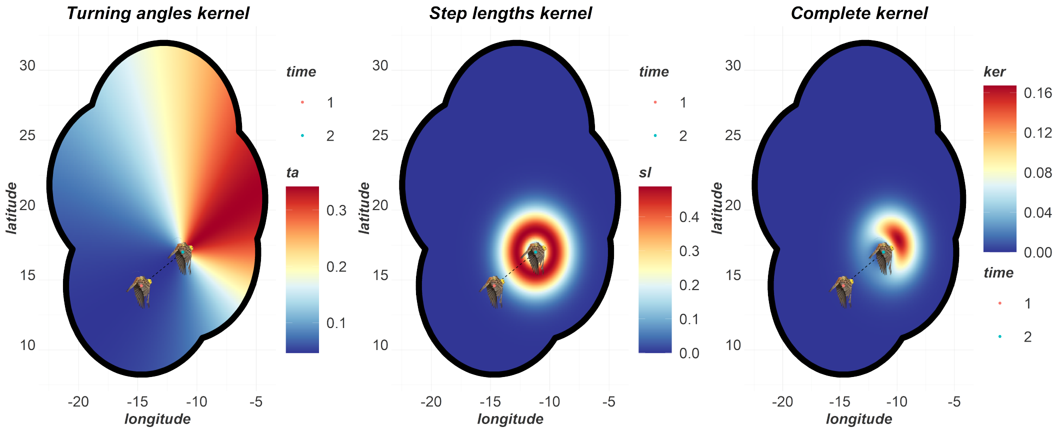
  Spatially (or spatio-temporally) varying covariates $X_\cdot$ and a residual random field $u(\mathbf{s})$.

- Combined normalised conditional observation density function:

$$f_{t|<t}(\mathbf{y}_t|\boldsymbol{\theta},\eta) = \frac{K(\mathbf{y}_t|\mathbf{y}_{<t},\boldsymbol{\theta})\exp[\eta(\mathbf{y}_t)]}{\int_{\mathcal{D}} K(\mathbf{s}|\mathbf{y}_{<t},\boldsymbol{\theta})\exp[\eta(\mathbf{s})]\,\mathrm{d}\mathbf{s}}$$

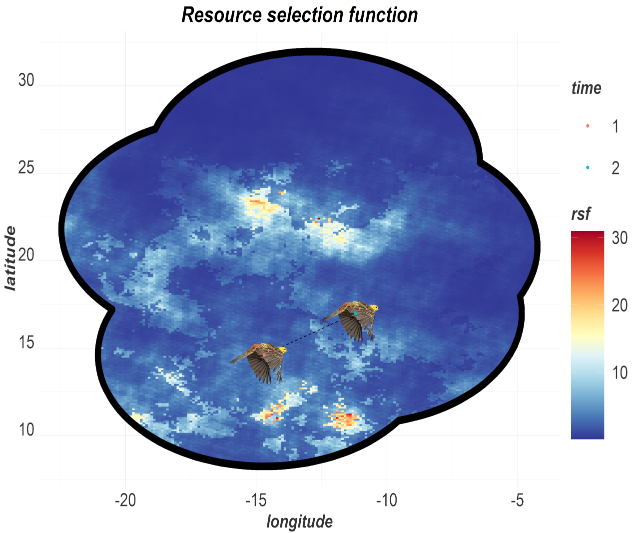
Movement kernel

Movement capacity of an animal:



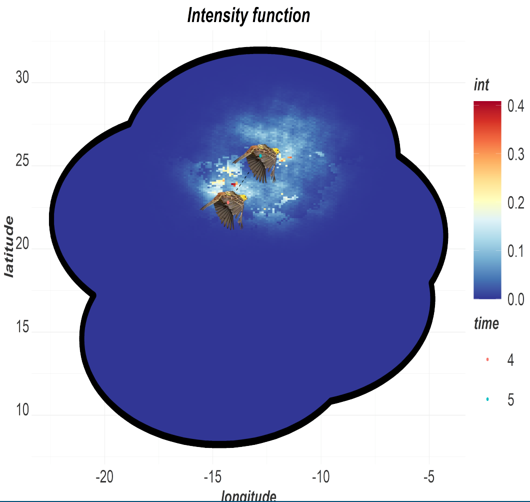
Resource selection function

Spatial features in the study area:

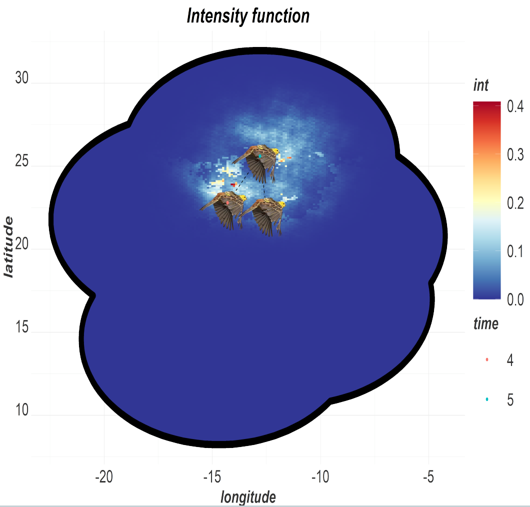


Combined effect

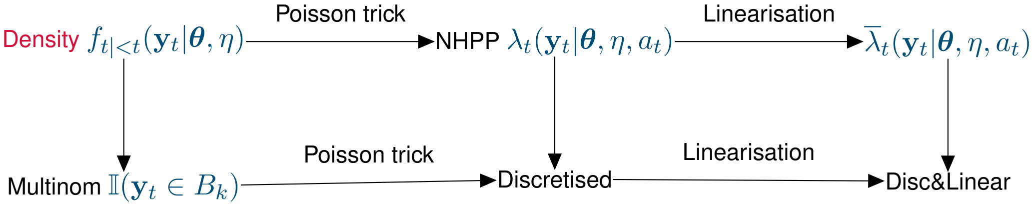
Intensity function



Movement decision!



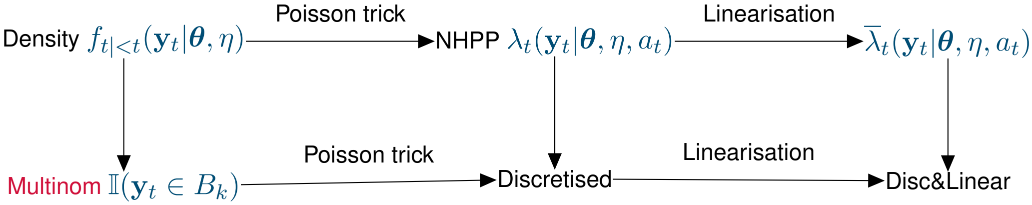
From movement kernel to discretised point process likelihood



$$f_{t|<t}(\mathbf{y}_t|\boldsymbol{\theta},\eta) = \frac{K(\mathbf{y}_t|\mathbf{y}_{<t},\boldsymbol{\theta})\exp[\eta(\mathbf{y}_t)]}{\int_{\mathcal{D}} K(\mathbf{s}|\mathbf{y}_{<t},\boldsymbol{\theta})\exp[\eta(\mathbf{s})]\,\mathrm{d}\mathbf{s}}$$

Problem: Inconvenient normalisation integral.

From movement kernel to discretised point process likelihood



$$f_{t|<t}(\mathbf{y}_t|\boldsymbol{\theta},\eta) = \frac{K(\mathbf{y}_t|\mathbf{y}_{<t},\boldsymbol{\theta})\exp[\eta(\mathbf{y}_t)]}{\int_{\mathcal{D}} K(\mathbf{s}|\mathbf{y}_{<t},\boldsymbol{\theta})\exp[\eta(\mathbf{s})]\,\mathrm{d}\mathbf{s}}$$

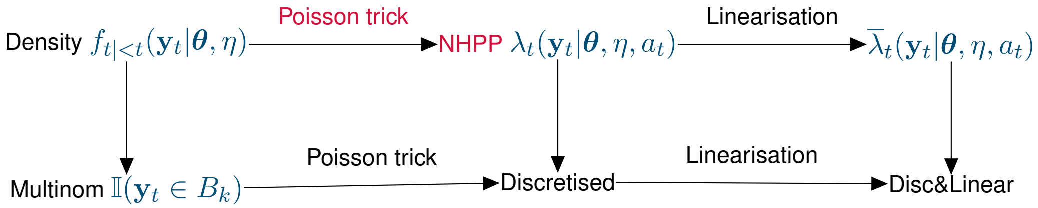
Previous approach: Subdivide space into disjoint sets B_k, with $\mathcal{D} = \cup_{k=1}^N B_k$.

$$\mathbf{z}_t = [\mathbb{I}(\mathbf{y}_t \in B_1), \dots, \mathbb{I}(\mathbf{y}_t \in B_N)] \sim \text{Multinomial}\left(1, \{p_k, k = 1, \dots, N\}\right)$$

$$p_k = \mathsf{P}(\mathbf{y}_t \in B_k|\mathbf{y}_{<t},\boldsymbol{\theta},\eta) = \int_{B_k} f_{t|<t}(\mathbf{s}|\boldsymbol{\theta},\eta)\,\mathrm{d}\mathbf{s}$$

(No improvement: multiple integrals)

From movement kernel to discretised point process likelihood



$$\lambda_t(\mathbf{y}_t|\boldsymbol{\theta},\eta,a_t) = K(\mathbf{y}_t|\mathbf{y}_{<t},\boldsymbol{\theta})\exp[\eta(\mathbf{y}_t)+a_t], \quad a_t \sim \text{Unif}(\mathbb{R})$$

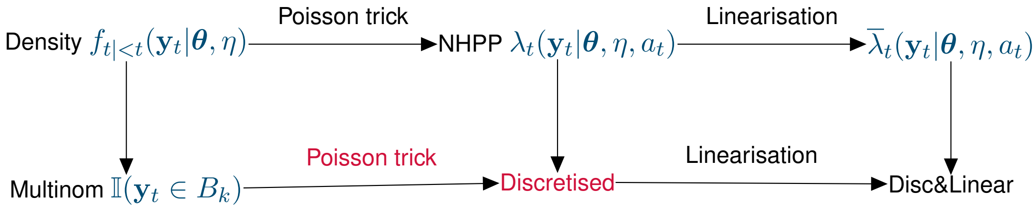
$$l(\{\mathbf{y}_t\}|\boldsymbol{\theta},\eta,\{a_t\}) = -\sum_t \int_{\mathcal{D}} \lambda_t(\mathbf{s}|\boldsymbol{\theta},\eta,a_t)\,\mathrm{d}\mathbf{s} + \sum_t \log\lambda_t(\mathbf{y}_t|\boldsymbol{\theta},\eta,a_t)$$

Non-homogeneous Poisson point process with a single point observation for each t.

a_t replaces the explicit density normalisation by *estimating* it.

The posterior distribution for $\boldsymbol{\theta}$, $\beta_\cdot$, and $u(\cdot)$ is unchanged!

From movement kernel to discretised point process likelihood

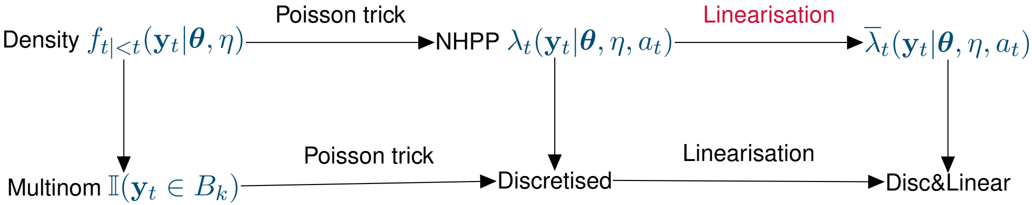


$$\lambda_t(\mathbf{y}_t|\boldsymbol{\theta},\eta,a_t) = K(\mathbf{y}_t|\mathbf{y}_{<t},\boldsymbol{\theta})\exp[\eta(\mathbf{y}_t)+a_t], \quad a_t \sim \text{Unif}(\mathbb{R})$$

$$l(\{\mathbf{y}_t\}|\boldsymbol{\theta},\eta,\{a_t\}) \approx -\sum_t\sum_k \lambda_t(\mathbf{s}_k|\boldsymbol{\theta},\eta,a_t)w_k + \sum_t \log\lambda_t(\mathbf{y}_t|\boldsymbol{\theta},\eta,a_t)$$

Integration points and weigths $(\mathbf{s}_k, w_k)$, adapted to the spatial model resolution.

From movement kernel to discretised point process likelihood



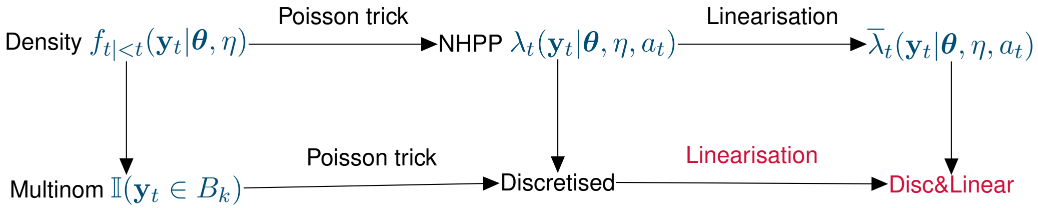
$$\log \overline{\lambda}(\mathbf{y}_t|\boldsymbol{\theta},\eta,a_t) = \log K(\mathbf{y}_t|\mathbf{y}_{<t},\boldsymbol{\theta}_0) + \frac{\mathrm{d}\log K(\mathbf{y}_t|\mathbf{y}_{<t},\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}}(\boldsymbol{\theta}-\boldsymbol{\theta}_0) + \eta(\mathbf{y}_t) + a_t$$

$$l(\{\mathbf{y}_t\}|\boldsymbol{\theta},\eta,\{a_t\}) = -\sum_t \int_{\mathcal{D}} \overline{\lambda}_t(\mathbf{s}|\boldsymbol{\theta},\eta,a_t)\,\mathrm{d}\mathbf{s} + \sum_t \log \overline{\lambda}_t(\mathbf{y}_t|\boldsymbol{\theta},\eta,a_t)$$

(Iterative) linearisation to a log-linear point process intensity allows more general movement kernel parameterisation.

(Preliminary theory: posterior approximation related to Fisher scoring)

From movement kernel to discretised point process likelihood



$$\log \overline{\lambda}(\mathbf{y}_t|\boldsymbol{\theta},\eta,a_t) = \log K(\mathbf{y}_t|\mathbf{y}_{<t},\boldsymbol{\theta}_0) + \frac{\mathrm{d}\log K(\mathbf{y}_t|\mathbf{y}_{<t},\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}}(\boldsymbol{\theta}-\boldsymbol{\theta}_0) + \eta(\mathbf{y}_t) + a_t$$

$$l(\{\mathbf{y}_t\}|\boldsymbol{\theta},\eta,\{a_t\}) \approx -\sum_t\sum_k \overline{\lambda}_t(\mathbf{s}_k|\boldsymbol{\theta},\eta,a_t)w_k + \sum_t \log\overline{\lambda}_t(\mathbf{y}_t|\boldsymbol{\theta},\eta,a_t)$$

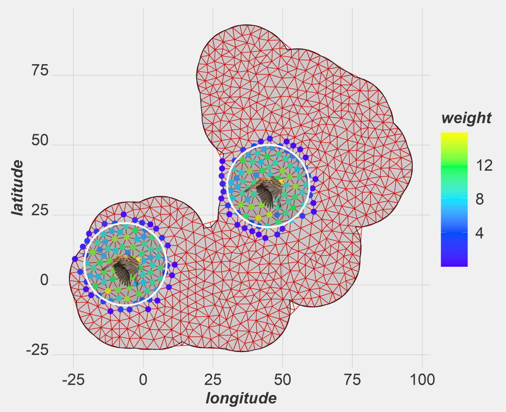
This is *almost* a log-linear Poisson count log-likelihood;

In $-E\lambda + y\log(E\lambda)$, R-INLA allows us to specify the two terms separately, without pairing them up with equal E and λ values.

Mesh, integration points and weights

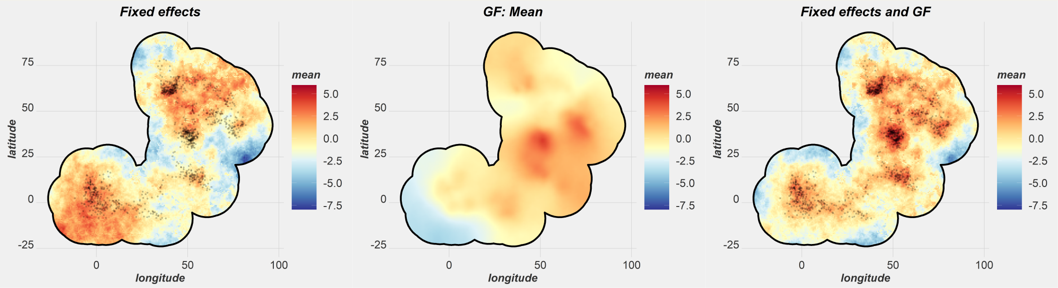
- Restricted domain of availability at each time point: Disk with radius (at least) equal to the maximum observed step length
- Integration points: At mesh nodes to ensure stability
- Deterministic integration: Previous Monte Carlo strategies are inefficient and unstable

Domains of availability, integration points and weights



Estimated log-intensity function

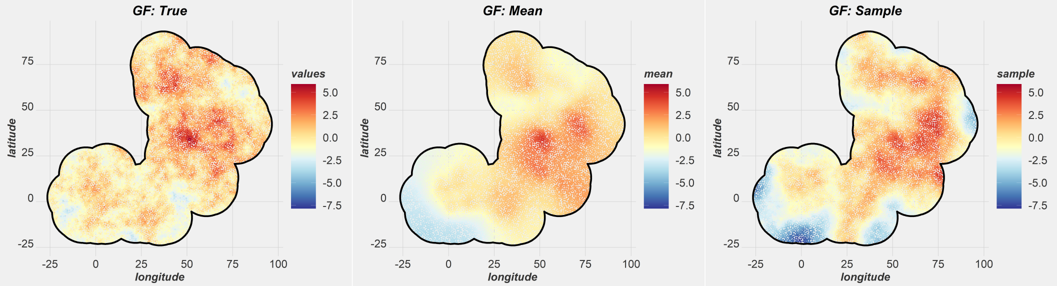
Contributions to the linear predictor



The Gaussian random field (GF) contribution improves the estimated animal density.

Estimated Gaussian random field (GF)

Comparison of the true GF, the estimated mean and a sample GF



Posterior samples can be used to quantify uncertainty of the fields and linear/nonlinear functionals of the fields.

Note: Recall that conditional means are fundamentally smoother than conditional realisations!

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”:

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

- 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+ supports 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 for more general transformation models (likely supported from 2.11.0):

```
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:

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

  - 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