inlabru: Bayesian spatial and spatio-temporal modelling in R

Finn Lindgren
finn.lindgren@ed.ac.uk

RSS conference, Harrogate, 6 September 2023

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 eparable 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

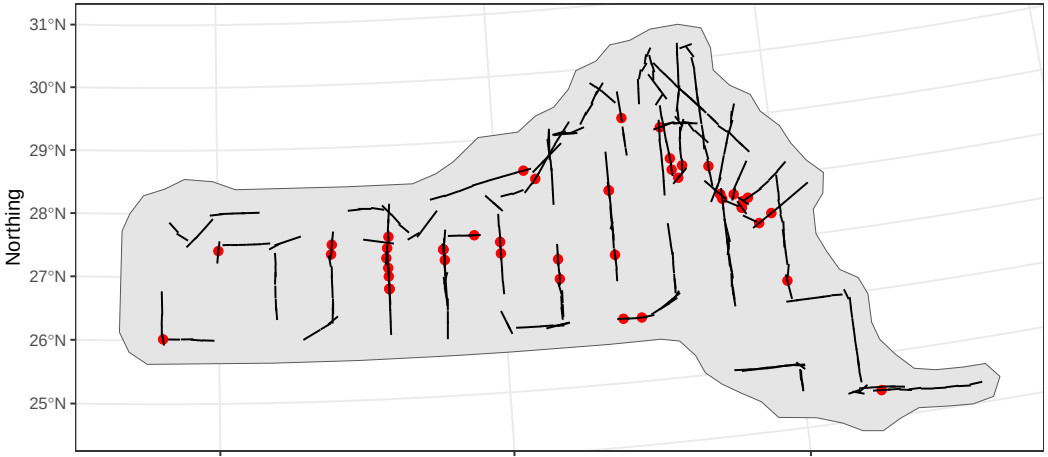
```
matern <- inla.spde2.pcmatern(mesh, ...)

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

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:

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

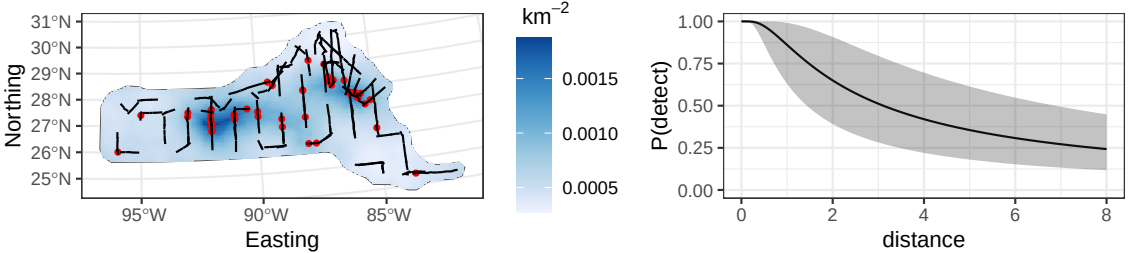
inlabru knows how to construct the Poisson process likelihood along lines and on polygons, and kronecker spaces (line $\times$ distance)
We can define $\log(\sigma)$ as a latent Gaussian variable and iteratively linearise. The non-linearity is mild, and the iterative INLA method converges.

```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)))
```

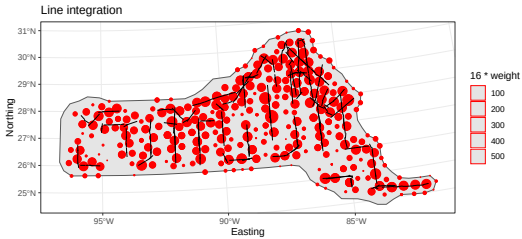
mean	sd	q0.025	q0.5	q0.975	median	mean.mc_std_err	sd.mc_std_err
104.0787	28.30949	58.5873	101.0749	158.4985	101.0749	2.830949	2.917886

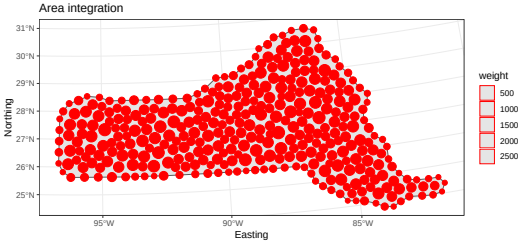
How many would be seen under perfect detection across the whole study area?

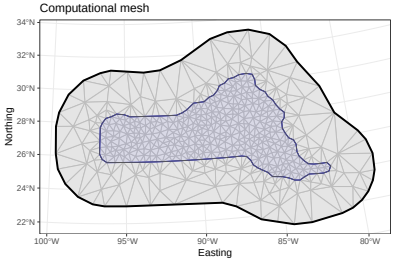
```
predpts <- fm_int(mesh, samplers = region_of_interest)
Lambda <- predict(fit, predpts, ~ sum(weight * exp(field + Intercept)))
```

mean	sd	q0.025	q0.5	q0.975	median	mean.mc_std_err	sd.mc_std_err
357.3103	113.744	196.039	334.1039	608.0631	334.1039	11.3744	14.97826

Integration points





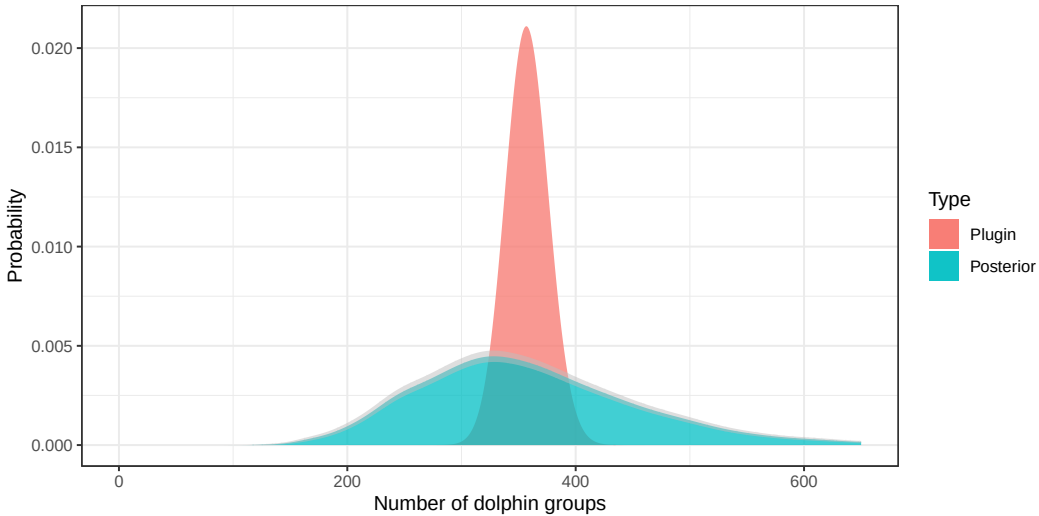


Complex prediction expressions

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