Session 2.4: advanced INLA features

VIBASS, University of Valencia

21 July 2022

Learning Objectives

At the end of this session you should be able to:

- implement a joint model with multiple likelihoods by using `R-ILA` and `inlabru`;
- use the `copy` feature with `R-INLA`;
- Implement with `inlabru` a model with two likelihoods and a shared spatial effect

The topics treated in this lecture can be partially found in **Chapter 8** of the INLA book.

Outline

1. Joint model with multiple likelihoods

2. The `copy` feature

3. Two likelihoods model with shared spatial effect using `inlabru`

Joint model with multiple likelihoods

Multiple likelihoods

- In many situations you need to combine data from different sources and/or differente characteristics. For example:
  - a subset of the data follows a Gaussian distribution and the other follows a Poisson distribution)
  - the data come from the same distribution but with different hyperparameters (e.g. one subset of the data comes from a Gaussian distribution with precision τ_1 and the other from a Gaussian with precision τ_2).

- In this case we need to be able to handle **multiple likelihoods** when implementing the **joint model**.

- In R-INLA this requires to have a particular structure for the data. For example, in the case of two likelihoods with two responses $\boldsymbol{y}_1 = (y_{11}, \ldots, y_{1n})$ and $\boldsymbol{y}_2 = (y_{21}, \ldots, y_{2m})$, the data should be structured in a 2-columns matrix like the following:

$$\begin{bmatrix} y_{11} & NA \\ \ldots & NA \\ y_{1n} & NA \\ NA & y_{21} \\ NA & \ldots \\ NA & y_{2m} \end{bmatrix}$$

Two likelihoods model: example

This example shows how to use information from two data sources (Binomial and Poisson distribution) to estimate the effect of a common covariate x and of two different intercepts.

- **Likelihood 1**:

$$y_{1i} \sim Bin\left(n = 1, p_i = \frac{\eta_{1i}}{1 + \eta_{1i}}\right) \qquad i = 1, \ldots, n$$

where $\eta_{1i} = \alpha + \beta_1 x_{1i}$

- **Likelihood 2**:

$$y_{2i} \sim Poisson\left(\lambda_i = \exp(\eta_{2i})\right) \qquad i = 1, \ldots, m$$

where $\eta_{2i} = \gamma + \beta_1 x_{2i}$

Two likelihoods model: data simulation

- We simulate $n = 100$ data from the **Binomial** distribution:

```
> n = 100
> x1 = runif(n)
> eta1 = 1 + x1 #linear predictor 1
> y1 = rbinom(n, size = 1, prob = exp(eta1)/(1+exp(eta1)))
> df1 = data.frame(x = x1, y = y1)
```

- We simulate $m = 200$ data from the **Poisson** distribution

```
> m = 200
> x2 = runif(m)
> eta2 = 2 + x2 #linear predictor 2
> y2 = rpois(m, exp(eta2))
> df2 = data.frame(x = x2, y = y2)
```

Two likelihoods model: implementation with `inla`

- Define the **block matrix** for the two response variables:

```
> y = matrix(NA, n+m, 2)
> y[1:n, 1] = y1  # first column
> y[(n+1):(n+m), 2] = y2  #second column
> Ntrials = c(rep(1,n), rep(NA, m)) # required only for Binomial data
```

- Define the **block matrix** for the two intercepts:

```
> intercept1 = numeric(n+m) #empty vector
> intercept1[1:n] = 1
> intercept1[(n+1):(n+m)] = NA
>
> intercept2 = numeric(n+m) #empty vector
> intercept2[1:n] = NA
> intercept2[(n+1):(n+m)] = 1
```

- With this approach every row has only a single observed value.
- Covariates and latent effects can be indexed from 1 to $n + m$. When a covariate or effect must only affect observations in a single likelihood, the values of the indices for the other likelihood must be set to `NA`.

Two likelihoods model: implementation with inla

- Fit the model with inla:

```
> # Covariate vector
> x = c(x1, x2)
>
> formula = y  ~ -1 + intercept1 + intercept2 + x #all the terms
>
> library(INLA)
> output = inla(formula,
+               data = list(y = y, intercept1 = intercept1,
+                           intercept2 = intercept2, x = x),
+               family = c("binomial", "Poisson"),
+               Ntrials = Ntrials)
>
> output$summary.fixed[,c("mean","0.025quant","0.975quant")]
```

```
               mean 0.025quant 0.975quant
intercept1 0.9557212  0.4789041   1.472898
intercept2 2.0236658  1.9330591   2.112957
x          0.9511763  0.8114298   1.091260
```

Two likelihoods model: implementation with inlabru

- Define the model components and the likelihoods:

```
> cmp = y ~ Intercept1(1) + Intercept2(1) + x
>
> library(inlabru)
> like1 = like(formula = y ~ Intercept1 + x,
+              family = "binomial",
+              data = df1,
+              Ntrials = rep(1, nrow(df1)))
>
> like2 = like(formula = y ~ Intercept2 + x,
+              family = "Poisson",
+              data = df2)
```

- Fit the model with inlabru:

```
> outputbru = bru(cmp, like1, like2)
> outputbru$summary.fixed[,c("mean","0.025quant","0.975quant")]
```

```
               mean 0.025quant 0.975quant
Intercept1 0.9557213  0.4789015   1.472903
Intercept2 2.0236658  1.9330383   2.112979
x          0.9511766  0.8114168   1.091274
```

The copy feature

The copy feature

- In R-INLA when we have a model defined with the following formula

```
> formula = y ~ f(idx1, model1, ...) + f(idx2, model2, ...)
```

it means that only one element from each sub-model contributes to the linear predictor for each observation. This means that the linear predictor is connected to only one element of the random effect idx1 and to only one element of the random effect idx2.

- This is not sufficient when an element of a model is needed more than once for each observation or when the same effect is shared among two or more linear predictors (and will be estimated by using two or more parts of the dataset).

- The solution for this is the copy feature. Formally, it defines a copy of a generic effect θ as

$$\theta^\star = \beta\theta + \epsilon$$

  where $\epsilon \sim \text{Normal}(0, b)$, with b being a large and fixed value (ϵ is a tiny error), and β is a hyperparameter. By fixing $\beta = 1$, it means that $\theta^\star$ is an exact copy of θ. According to the specification of the fixed option (TRUE or FALSE), the hyperparameter β is kept fixed or estimated.

- Several copies of the same effect can be created and all will share the same hyperparameters.

The copy feature: example

The following example considers a Gaussian likelihood whose linear predictor includes two correlated random effects - a and b - coming from a bivariate Normal distribution:

$$y_i \sim Normal(\eta_i, \tau^{-1}), \qquad i = 1, \ldots, n$$

where the linear predictor is given by

$$\eta_i = a_i + b_i z_i$$

and

$$(a_i, b_i) \overset{iid}{\sim} Normal(\mathbf{0}, \boldsymbol{Q}^{-1})$$

The term z_i refers to a known covariate.

In `R-INLA` the bivariate Gaussian model is implemented with `f(index, model = "iid2d", ...)` but this does not allow to have each η_i connected to two elements of the same (bi-variate Gaussian) model. We will use `copy`!

The copy feature: data simulation

We simulate $n = 1000$ values from the previous model with $\tau = \tau_a = \tau_b = 1$, $\rho_{ab} = 0.8$.

```
> set.seed(3)
> n = 1000
> z = rnorm(n) #covariate
>
> Sigma = matrix(c(1, 0.8, 0.8, 1), 2, 2) # var-cov matrix
> library(mvtnorm)
> ab = rmvnorm(n, sigma = Sigma) #bivariate normal
> a = ab[, 1]
> b = ab[, 2]
>
> eta = a + b * z #linear predictor
> y = eta + rnorm(n, sd = 1) #response variable
```

The copy feature: implementation with inla

The iid2d model is represented internally as one vector of length $2n$. We set the indexes correspondingly and define the formula with the copy feature:

```
> i = 1 : n # use only the first n elements (a_1, ..., a_n)
> j = (n+1) : (2*n) # use only the last n elements (b_1, ..., b_n)
>
> formula = y ~ -1 + f(i, model = "iid2d", n = 2*n) + f(j, z, copy = "i",fixed=TRUE)
```

And finally run inla:

```
> result = inla(formula,
+               data = data.frame(y = y, z = z, i = i, j = j),
+               family = "gaussian")
> result$summary.hyperpar[,c("mean", "0.025quant", "0.975quant")]
```

```
                                              mean 0.025quant 0.975quant
Precision for the Gaussian observations 1.0757964  0.8536628  1.3287137
Precision for i (component 1)           0.9783850  0.7767089  1.2274618
Precision for i (component 2)           1.4771635  1.1474891  1.8748224
Rho1:2 for i                            0.7565963  0.6793477  0.8226678
```

The copy feature: implementation with inlabru

```
> # Model components
> comp = y ~ eff(i, model = "iid2d", n = 2*n) + eff2(j, z, copy = "eff", fixed=T)
```

Note that in inlabru when using copy we refer to the effect name and not to the index name.

```
> # Likelihood
> lik = like("gaussian",
+            formula = y ~ + eff + eff2,
+            data = data.frame(y = y, z = z, i = i, j = j))
>
> # Run inlabru
> resultinlabru = bru(comp, lik)
> resultinlabru$summary.hyperpar[,c("mean", "0.025quant", "0.975quant")]
```

```
                                          mean 0.025quant 0.975quant
Precision for the Gaussian observations 1.1338876  0.7552099  1.6477099
Precision for eff (component 1)         0.9466587  0.5844661  1.4266440
Precision for eff (component 2)         1.5311295  1.0203687  2.2336774
Rho1:2 for eff                          0.7430300  0.6308005  0.8315172
```

Two likelihoods model with shared spatial effect using `inlabru`

Two likelihoods and shared spatial effect: example

Model | Data

- We consider two sets of observations in space (with n_1 and n_2 locations):

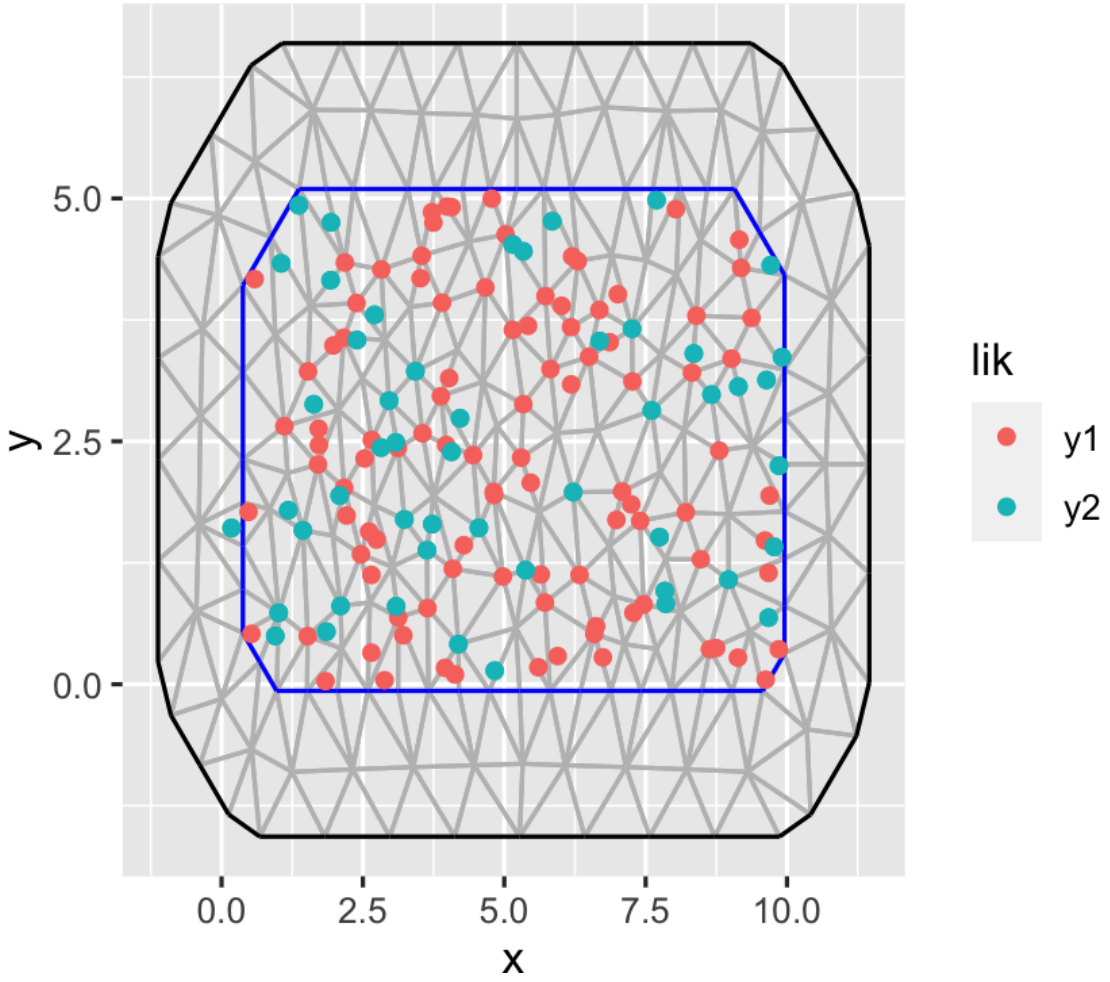
$$y_{1i} \sim \text{Normal}(\beta_1 + \gamma_1 x_i + \xi_i, \sigma_1^2) \qquad i = 1, \dots, n$$
$$y_{2i} \sim \text{Normal}(\beta_2 + \gamma_2 z_i + \xi_i, \sigma_2^2) \qquad i = 1, \dots, m$$

- Each set of data has its own intercept (β_1 and β_2) and covariate (with γ_1 and γ_2 as linear effect).
- The random effect ξ is a shared GF with range r and variance σ^2. It enters both in the linear predictor of y_1 and y_2.
- For the simulation we use: $n = 100$, $m = 50$, $\beta_1 = 3$, $\gamma_1 = 2$, $\sigma_1^2 = 0.3$, $\beta_2 = 10$, $\gamma_2 = 0.5$, $\sigma_2^2 = 0.2$, $r = 4$, $\sigma^2 = \sqrt{0.5}$.

Two likelihoods and shared spatial effect: example

Model | Data

```
> loc_all %>%
+   ggplot() +
+   gg(mesh) +
+   geom_point(aes(s1,s2,col=lik))
```



Implementation in inlabru

1. Define the SPDE model:

```
> spde <- inla.spde2.pcmatern(
+   mesh = mesh,
+   prior.range = c(1, 0.01), # P(range < 1) = 0.01
+   prior.sigma = c(1, 0.01)) # P(sigma > 1) = 0.01
```

2. We write down all the model components of the joint model:

```
> jcmp <- ~ Intercept1(1) + Intercept2(1) +
+   beta1(x1, model="linear") + beta2(x2, model="linear") +
+   field(coordinates, model = spde)
```

3. We specify the two likelihoods:

```
> lik1 <- like("gaussian",
+              formula = y ~ Intercept1 + beta1 + field,
+              data = df1)
> lik2 <- like("gaussian",
+              formula = y ~ Intercept2 + beta2 + field,
+              data = df2)
```

Implementation in inlabru and output summary

4. We run inlabru:

```
> jfit <- bru(jcmp, lik1, lik2)
```

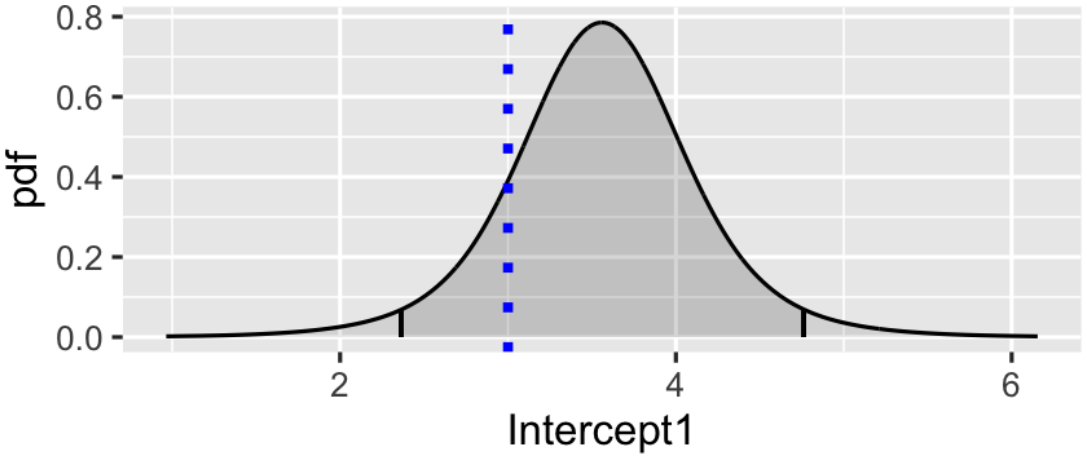
```
> jfit$summary.fixed[,c("mean","0.025quant","0.975quant")]
```

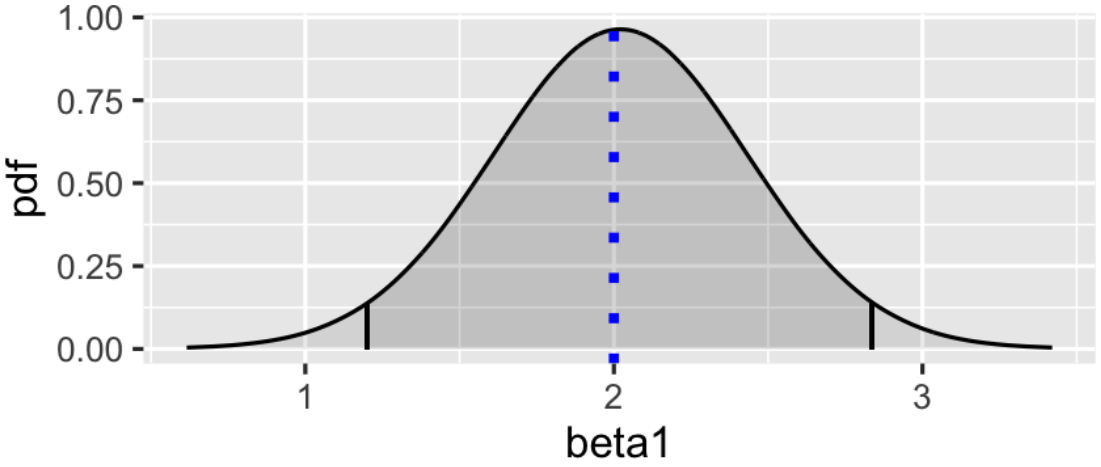
```
               mean 0.025quant 0.975quant
Intercept1  3.562606  2.3544165   4.773242
Intercept2 10.138851  8.9264961  11.362755
beta1       2.020479  1.1985237   2.841611
beta2       1.120432  0.3123589   1.931600
```

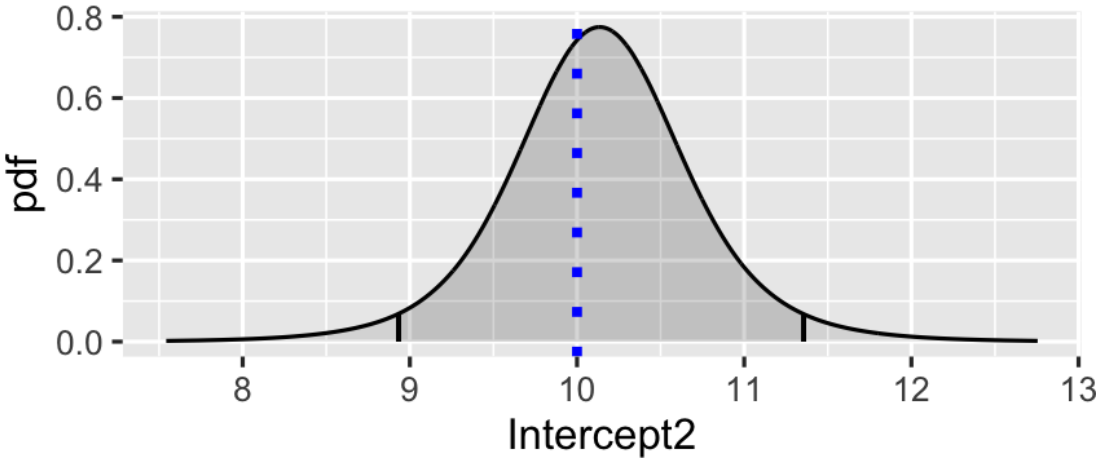
```
> jfit$summary.hyperpar[,c("mean","0.025quant","0.975quant")]
```

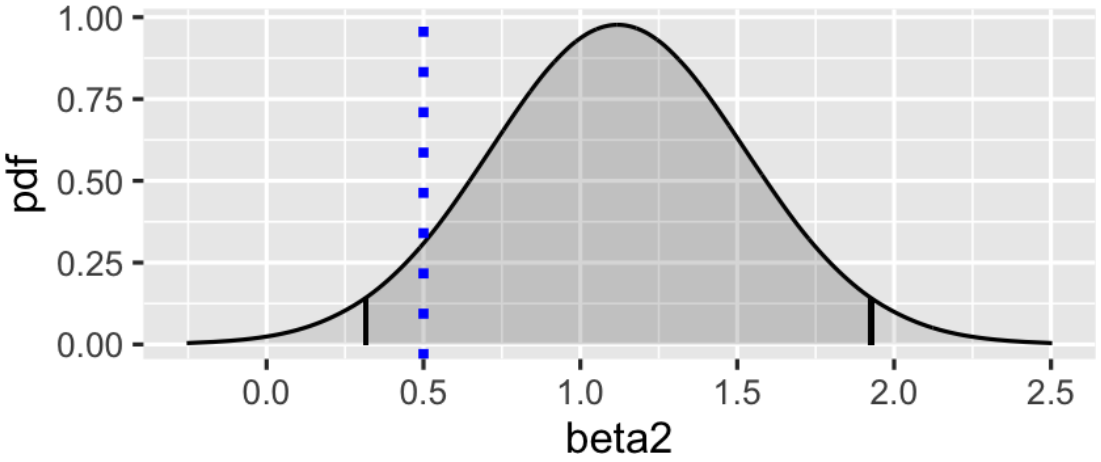
```
                                                mean 0.025quant 0.975quant
Precision for the Gaussian observations    0.8223147  0.5876694   1.114021
Precision for the Gaussian observations[2] 0.9598422  0.6834580   1.303628
Range for field                            4.3916432  2.2068024   8.376719
Stdev for field                            0.8570362  0.5722699   1.231279
```

Fixed effects









Spatial parameters

```
> spde.range <- spde.posterior(jfit,
+                              "field",
+                              what = "range")
> spde.var <- spde.posterior(jfit,
+                            "field",
+                            what = "variance")
```