Spatial modeling with INLA and inlabru

University of Zurich, March, 2022

Instructor: Sara Martino

Department of Mathematical Science (NTNU)

NTNU
Norwegian University of Science and Technology

Model choice and model assessment/validation

Introduction

We have now seen some fancy modelling approaches

Introduction

We have now seen some fancy modelling approaches

How can we assess the models and choose between them?

- Rather underdeveloped in statistical literature; Many suggestions; no clear “yes, this is how it should be done”

Model choice and assessment

- **Model assessment** is the art and science of evaluating how well a model and/or estimate agrees with observed reality, and of how useful it for specific purposes
  - Simple models -summary characteristics
  - Complex models - assessing variability in space
  - All models - prediction ability; calibrated uncertainty
- **Model choice** - which covariate and random effects to include
- **Model comparison** - which model is “better?”

Model choice

INLA can compute the following quantities:

- Marginal likelihood $\Rightarrow$ Bayes factors
- Deviance information criterion (DIC)
- Widely applicable information criterion (WAIC)

General advice

- We have little experience with practical usage of them for complex spatial models
- It is not clear what they actually mean in the context of the models we look at here
- Advice: use them cautiously
- Less adventurous if you are comparing models with only different numbers of covariates - and “the rest” is the same:
  - Use the same mesh in the models you compare (do not treat the mesh resolution as a model choice!)

Marginal likelihood

```r
result = inla(...,
              control.compute=list(mlik=TRUE))

result = bru(...,options = list(control.compute =
                                  list(mlik = TRUE)))
```

- Calculates $\log(\pi(\boldsymbol{y}))$
- Can calculate Bayes factors through differences in value
- **NB:** Problematic for intrinsic models

Deviance information criterion

```
result = inla(...,
              control.compute=list(dic=TRUE))

result = bru(...,options = list(control.compute =
                                    list(dic = TRUE)))
```

DIC is a measure of complexity and fit. It is used to compare complex hierarchical models and is defined as:

$$\text{DIC} = \overline{D} + p_D$$

where $\overline{D}$ is the posterior mean of the deviance and p_D is the effective number of parameters. Smaller values of the DIC indicate a better trade-off between complexity and fit of the model.

Widely applicable information criterion (WAIC)

```
result = inla(...,
              control.compute=list(waic=TRUE))

result = bru(...,options = list(control.compute =
                                 list(waic = TRUE)))
```

- WAIC is like DIC just newer, and perhaps better
- See *"Understanding predictive information criteria for Bayesian models"* (2013) by Andrew Gelman, Jessica Hwang, and Aki Vehtari

Model assessment with cross-validated scores

Posterior predictive distributions can be used for model assessment and model selection.

Full cross-validation or out-of-sample validation is expensive.

`R-INLA` provides two leave-one-out crossvalidation quantities:

- **Conditional predictive ordinate**
- **Probability integral transform**

Conditional predictive ordinate

```
result = inla(...,
              control.compute=list(cpo=TRUE))

result = bru(...,options = list(control.compute =
                                list(cpo = TRUE)))
```

- Measures fit through the predictive density $\pi(y_i^{obs} \mid \boldsymbol{y}_{-i})$
- Basically, Bayesian hold-one out
- Easy to compute in the INLA-approach
- Possible failure ($cpo$failure)
- See *Posterior and Cross-validatory Predictive Checks: A Comparison of MCMC and INLA* (2009) by Held, Schr{"o}dle and Rue

Proper scoring rule based on CPO

A predictive score is proper if its expected value is minimised under the true distribution.

- The **log-CPO-score**

$$\text{logCPO} = -\sum_{i=1}^{n} \log(\text{CPO}_i) = -\sum_{i=1}^{n} \log[p(y_i^{\text{obs}} | y_j^{\text{obs}}, j \neq i)]$$

is a strictly proper scoring rule.

- The logCPO score encourages appropriate prediction uncertainty; bias, overconfidence, and underconfidence all increase the score.
- 2logCPO is similar in scale to DIC and WAIC but has a clear cross validation prediction interpretation.

Pairwise observasion CPO scores

- The aggregated logCPO score hides information
- Model comparison for predictions is a pairwise comparison problem for each individual observation!
- Compute the collection of pairwise logCPO differences for two models
- Inspect the empirical score difference distribution; is it consitently positive/negative?
- Inspect the spatial pattern of the score differences

Probability integral transform

```
result = inla(...,
              control.compute=list(pit=TRUE))

result = bru(...,options = list(control.compute =
                                  list(pit = TRUE)))
```

- Given by

$$\text{Prob}(Y_i \leq y_i^{obs} \mid \boldsymbol{y}_{-i})$$

PIT: assessing prediction bias, scale and shape

- A direct consequence of the PIT definition is that under the true model, each PIT_i value is a sample of a uniform distribution on [0, 1].
- The usual plotting method for PIT is a histogram.
- For models with too small predictive variance, the histogram tends to increase toward 0 and 1.
- For models with too large predictive variance, the histogram tends to have peak in the middle.
- For incorrectly skewed predictions, the PIT histogram will tend to be skewed.
- Unfortunately, that doesn't necessarily imply that overfitting and oversmoothing can be detected and/or correctly diagnosed.

Advanced features

Useful features

There are several features that can be used to extend the standard models in `R-INLA` (and `inlabru`)

- Replicate
- Group
- Copy
- Multiple likelihoods
- Generic precision matrices (`rgeneric`)

Main goals

- know about the features
- be exposed to the ideas

Feature:Replicate

Feature: `replicate`

`replicate` generates iid replicates from the same f()-model with the same hyperparameters.

If $\mathbf{x} \mid \theta \sim \text{AR}(\mathbf{1})$, then nrep=3, makes

$$\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3})$$

with mutually independent $\mathbf{x_i}$'s from AR(1) with the same θ

```
    f(..., replicate = r [, nrep = nr ])
```

where replicate are integers $1, 2, \ldots,$ etc

Example

$$y_i^1 \sim \text{Poisson}(\lambda_i^1), \quad i = 1, \ldots, n_1$$
$$y_i^2 \sim \text{Poisson}(\lambda_i^2), \quad i = 1, \ldots, n_2$$

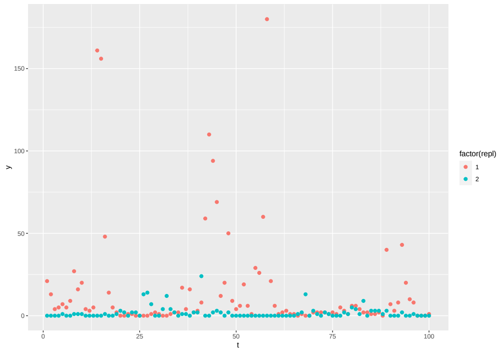
$$\log(\lambda_i^1) = \mu_1 + u_i^1$$
$$\log(\lambda_i^2) = \mu_2 + u_i^2$$

and $\mathbf{u}^1$ and $\mathbf{u}^2$ are two replicates of the same AR1 model (they share the same parameters)

Example : simulate data

```r
# Simulate data - 2 groups with same AR1 param
n = 100
rho <- 0.8
mu = c(1,-1)
x1 = arima.sim(n=n, model=list(ar=c(rho))) + mu[1]
x2 = arima.sim(n=n, model=list(ar=c(rho))) + mu[2]
# generate Poisson observations
y1 = rpois(n, lambda = exp(x1))
y2 = rpois(n, lambda = exp(x2))

df_groups <- data.frame(y = c(y1, y2),
                        t = rep(1:n, 2),
                        repl = rep(1:2, each = n),
                        int = rep(0:1, each = n))
```

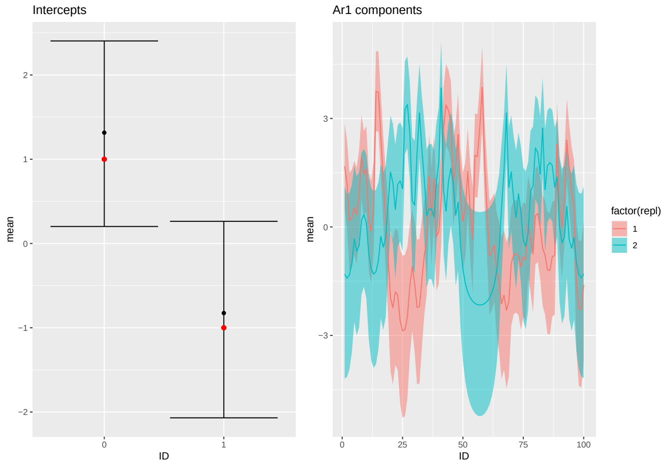
Example : simulate data



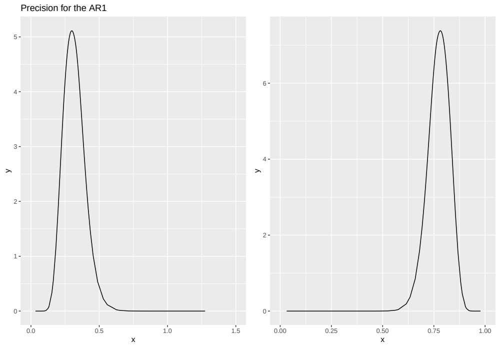
Example: fit the model

```
cmp <- y ~  -1 + int(int,  model = "factor_full") +
  myar1(t, model = "ar1", replicate = repl)
fit <- bru(cmp, family = "poisson", data = df_groups)
```

Example: Results - Latent field



Example: Results - Hyperparameters



Feature:Group

Feature: group

- Similar concept as replicate, but with a dependence structure on the replicates. E.g.~rw1, rw2, ar1, exchangeable
- Implemented as a Kronecker product (often space and time)
- It's possible to use both replicate and group! This will be replications of the grouped model
- Usage

```
f(..., group = g [, ngroup = ng])
```

where replicate are integers $1, 2, \ldots,$ etc

Feature:Multiple likelihoods

Feature:Multiple likelihood

There is no constraint in INLA that the type of likelihood must be the same for all observations. In fact, every observation could have its own likelihood.

- Coregionalization model
- Marked point process
- Joint models of various kinds

Example: Simulate data

We fit a simple model where we imagine that some data come from a Gaussian and some from a Poisson likelihood:

Example: Fit the model

```
cmp = ~ Intercept_1(1) + Intercept_2(1) +
  x1(x1, model = "linear") + x2(x2, model = "linear")

lik1 = like(formula = y1~Intercept_1 + x1,
            family = "gaussian",
            exclude = c("Intercept_2","x2"),
            data = d1)

lik2 = like(formula = y2~Intercept_2 + x2,
            family = "poisson",
            exclude = c("Intercept_1","x1"),
            data = d2)

fit = bru(cmp, lik1,lik2)
```

Feature: `copy`

Feature: `copy`

`Allows different elements of the same `f(...)` to be in the the same linear predictor.`

`Without copy we can not (directly) specify the model`

$$\eta_i = u_i + u_{i+1} + \ldots$$

Sometimes this is necessary

Feature: `copy`

`The linear predictor`

$$\eta_i = u_i + u_{i+1} + \ldots$$

can be coded as

```
   formula = y ~ f(i, model = "iid")
                 + f(i.plus, copy="i") + ...
```

- The copy-feature, creates internally an additional sub-model which is ϵ-close to the target
- Many copies allowed, and copies of copies

Feature: copy

It is also possible to include scaled copies

$$\eta_i = u_i + \beta u_{i+1} + \ldots$$

```
   formula = y ~ f(i, model="iid") +
                 f(i.plus, copy="i",
                   hyper = list(beta=list(fixed=FALSE)))
                 + ...
```

This introduces another hyperparameter in the model (which is fixed to 1 by default).

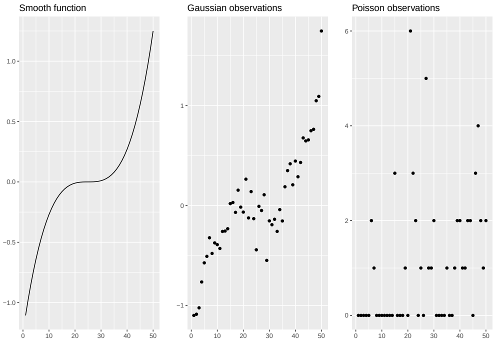
Feature: copy

$$
\begin{aligned}
y_i^1 &\sim \mathcal{N}(\mu_i, \tau) \\
\mu_i &= f(i)
\end{aligned}
$$

$$
\begin{aligned}
y_j^2 &\sim \text{Poisson}(\lambda_j), \quad j = 1, \ldots, 50 \\
\log(\lambda_j) &= f(i)
\end{aligned}
$$

Example : simulate data

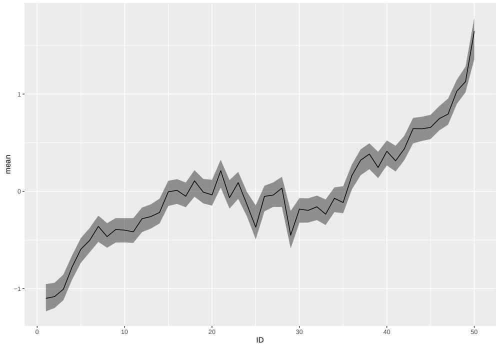
```
n = 50
idx = 1:n
x = idx
func = 10 * ((idx-n/2)/n)^3

y1 = rnorm(50, mean = func, sd = 0.2)
y2 = rpois(50, lambda  = exp(func))
```



Example : fit the model

```
df1 = data.frame(y1 = y1, idx1 =  1:n)
df2 = data.frame(y2 = y2, idx2 =  1:n)

cmp = ~ -1 +
  field(idx1, model = "rw1") +
  field_copy(idx2, copy = "field")

lik1 = like(formula  = y1~ field,
            family = "gaussian",
            exclude = c("field_copy"),
            data = df1)

lik2 = like(formula  = y2~ field_copy,
            family = "poisson",
            exclude = c("field"),
            data = df2)

fit = bru(cmp,
          lik1,
          lik2)
```

Example: Results


mean
ID