Some topics in inference with R-INLA

Thomas Opitz, BioSP, INRA Avignon

Workshop: Theory and practice of INLA and SPDE

November 8, 2018

Some points addressed in this talk

- Good prior choices for hyperparameters : **penalizing model complexity**
- Transforming posterior estimations
- **Troubleshooting** `R-INLA` : What can go wrong ? Why does `inla(...)` crash ?

1. Constructing prior models
2. Transforming posterior distributions returned by `R-INLA`
3. Troubleshooting

Choosing prior distributions

We distinguish two types of parameters in INLA models :

- **latent Gaussian variables** can be numerous and describe the linear predictor
  $\Rightarrow$ Gaussian process priors with various precision/dependence structures
- few parameters (or none) not arising linearly in the predictor : **hyperparameters**
  $\Rightarrow$ need to fix hyperprior distributions (or just fix to deterministic value)

Specifying prior distributions in `R-INLA` (when not using the default) :

```
hyper=list(theta=list(initial=initval,fixed=TRUE/FALSE,prior=priorname,...)
```

- `theta` is the standard name for hyperparameters
  (often, we also use `prec` for precision hyperparameters)
- `priorname` is the name of the prior distribution
- `initval` is a numeric starting value for the INLA optimization routines
- `fixed=TRUE/FALSE` $\Rightarrow$ keep `initval` fixed, or estimate the hyperparameter ?

Example : penalized complexity prior for the precision of the Gaussian likelihood (`theta`= log-precision)

```
control.family=list(hyper=list(theta=list(initial=log(1),fixed=FALSE,
                    prior="pc.prior",param=c(10,0.01))
```

Penalized Complexity priors

Simpson et al. (2017)

A principled and intuitive way of defining prior distributions for hyperparameters :

- shrink model towards a **simpler reference model** at **constant rate penalty**
  (we measure distance to reference model through Kullback-Leibler divergence)
- allows defining moderately informative priors, relatively "stable" in practice
- Examples :
  - Gaussian precision/variance : reference model is variance$= 0$
    $\Rightarrow$ exponential prior distribution on standard deviation
  - Matérn correlation : reference model is constant field 0 everywhere
    $\Rightarrow$ range of Matérn correlation : reference model is infinite range
    $\Rightarrow$ exponential prior distribution on $1/\text{range}$

In `R-INLA`, can often specify PC priors by **two parameters u and α** satisfying

$$\text{Probability}(\text{hyperparameter} > u) = \alpha$$
$$(\text{or} < u)$$

Gaussian dependence : IGMRF or SPDE ?

In **Intrinsic Gauss–Markov Random Field models**,
what happens at a specific site/instant is conditionally specified as

weighted average of direct neighbors + independent noise

- need to define a finite number of neighbors, need to discretize space/time/...
- dependence and precision are interwoven, only 1 parameter (**precision**) to estimate

1D : `rw1`, `rw2`,...
spatial : `besag`, `bym`,...

In the **SPDE models**, the partial differential equation generates dependence over **continuous space** :

$$(\kappa - \Delta)^{\alpha/2} x(s) = \text{white noise}, \quad \alpha = d/2 + \text{Matern shape}, \quad s \in K \subset \mathbb{R}^d$$

(+ finite element discretization over mesh $\Rightarrow$ computationally convenient GMRF)

1D : spline models with dependent spline coefficients.
spatial : classical Matérn covariance model.

If raw data are not aggregated over areas/classes, SPDE approaches seem preferable.
BUT : IGRMF models may be faster and simpler to estimate.

Posterior estimates and transformations

In Bayesian statistics, the estimation itself is a distribution :
it tells us which values of the parameter space arise with which probability.

To summarize this distribution, we often use the **posterior mean as a point estimate** :

$$\hat{\alpha} = \mathbb{E}(\alpha \mid \boldsymbol{y}).$$

for a parameter α of interest.

⚠ **Careful with transformations :**

$$\widehat{T(\alpha)} = \mathbb{E}(T(\alpha) \mid \boldsymbol{y}) \neq T(\mathbb{E}(\alpha \mid \boldsymbol{y})) = T(\hat{\alpha})$$

If the estimation uncertainty (e.g., the variance $\mathbb{V}(\alpha \mid \boldsymbol{y})$) is very low,
the right-hand side can give a practically useful approximation.

Otherwise, we have to "recalculate"

$$\mathbb{E}(T(\alpha) \mid \boldsymbol{y}) = \int T(\alpha)\pi(\alpha \mid \boldsymbol{y}), \mathrm{d}\alpha$$

based on the posterior density $\pi(\alpha \mid \boldsymbol{y})$.

⚠ If we use the **posterior median** $\hat{\alpha}_{\text{med}}$ (and not the posterior mean) as a point estimate, then we can directly transform the estimator without "transformation error".

Example : precision to standard deviation

Remember the example of monthly `rw1` for monthly Nottingham temperatures.

```
mean       sd 0.025quant 0.5quant 0.975quant    mode
Precision for the Gaussian observations 0.1884 0.0176     0.1558   0.1878     0.2249 0.1868
Precision for month                     0.0544 0.0216     0.0224   0.0511     0.1061 0.0446
```

"Naive" transformation of posterior mean of month precision parameter gives

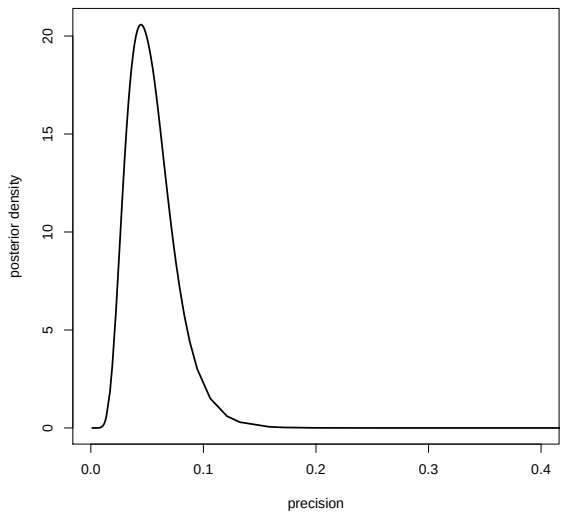
```
> sqrt(1/fit$summary.hyperpar$mean[2])
[1] 4.28788
```

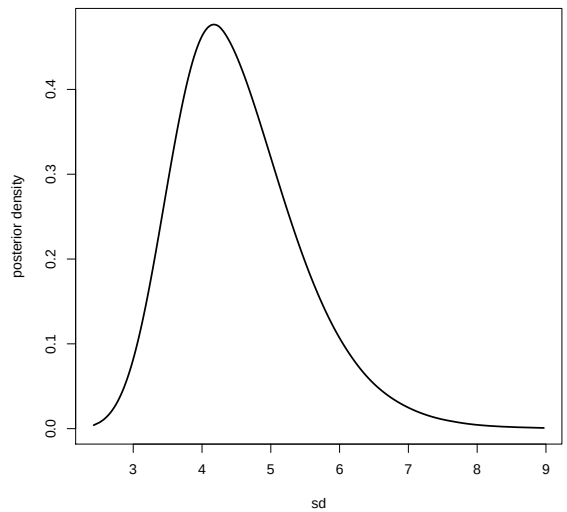
"Correct" transformation taking into account posterior uncertainty :

```
> prec2sd=function(prec){sqrt(1/prec)}
> inla.emarginal(prec2sd,fit$marginals.hyperpar$‘Precision for month‘)
[1] 4.544564
```

Plot posterior densities for precision and standard deviation :

```
plot(fit$marginals.hyperpar$‘Precision for month‘,...)
plot(inla.tmarginal(prec2sd,fit$marginals.hyperpar$‘Precision for month‘),...)
```





Sampling from the posterior distribution

To calculate the posterior estimates of some transformed quantity

$$\mathbb{E}(T(\alpha) \mid \boldsymbol{y}) = \int T(\alpha)\pi(\alpha \mid \boldsymbol{y})\, \mathrm{d}\alpha,$$

we can also adopt a Monte-Carlo approach by sampling from the posterior distribution :

```
sample.post = inla.posterior.sample(n = 1000, result = inlafit)
```

$\Rightarrow$ `sample.post[[i]]` contains the ith simulation from n simulations :

- `sample.post[[i]]$latent` is a vector containing a sample of the linear predictor $\boldsymbol{\eta}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ and the fixed and random effects included in $\boldsymbol{x}$
- `sample.post[[i]]$latent` contains the sampled hyperparameter values (if any)

$\Rightarrow$ construct $\widehat{T(\alpha)} = \frac{1}{n}\sum_{i=1}^{n} T(\alpha_i)$

1. Constructing prior models
2. Transforming posterior distributions returned by `R-INLA`
3. Troubleshooting

Things often do not work immediately...

The `inla(...)` run **may fail/crash for various reasons**.
In rare cases, even if `inla(...)` runs through, results may be nonsensical.
E.g., posterior effect $\approx$ prior effect in (overly) complex models for very high-dimensional data.

Typical issues :

- numerical instabilities in matrix calculations (very high-dimensional, near singular...)
- joint likelihood of all data is too "flat" for the numerical Laplace approximation to converge easily (e.g., many latent Gaussian "iid" variables)
- no more RAM memory available $\Rightarrow$ `inla(...)` crashes, or becomes extremely slow
- sometimes, with very high dimension of data, numerical instabilities if data or SPDE coordinates are "far from $O(1)$"

Troubleshooting

Always check fitted models if they "make sense" based on what you know about the data.

Troubleshooting for **very complex problems** and/or **very high-dimensional data**

$\Rightarrow$ use **trial and error** to see what works and where the problem is :

- start with simple models, add effects one at a time towards the "full" model
- to start, try a numerically convenient Gaussian response for simplicity :
  Laplace approximation is exact for Gaussian response
  $\Rightarrow$ no need for calculating it iteratively
  (BUT : don't do this for highly nongaussian data, e.g. 0/1-data)
- fit your model on a random subsample of your data to see if it works correctly
- use a less accurate but faster approximation strategy :
  `inla(...,control.inla=list(int.strategy="eb", strategy="gaussian"))`
- put more informative priors on fixed effects, e.g.
  `inla(...,control.fixed=list(prec=.1,prec.intercept=.01))`
- if responses y_i, covariates or SPDE coordinates are "far from $O(1)$",
  rescale them to have more moderate values