`INLA` and `inlabru` advanced features

University of Zurich, March, 2022

Instructor: Sara Martino

Department of Mathematical Science (NTNU)

NTNU
Norwegian University of
Science and Technology

Notes

Notes

Notes

Model choice and model assessment/validation

Notes

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"

Notes

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?"

Notes

Model choice

INLA can compute the following quantities:

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

Notes

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!)

Notes

Marginal likelihood

```
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

Notes

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:

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

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.

Notes

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

Notes

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**

Notes

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 (cpofailure)
- See *Posterior and Cross-validatory Predictive Checks: A Comparison of MCMC and INLA* (2009) by Held, Schr{"o}dle and Rue

Notes

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.

Notes

Pairwise observation 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

Notes

Probability integral transform

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

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

- Given by

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

Notes

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.

Notes

Remote computing

Notes

Remote computing

Very useful for large models. The R-session runs locally, but the computations are done on a remote (Linux/Mac) server.

```
inla(..., inla.call="remote")
```

using `ssh`.

{(Initial set up required, see `inla.remote` and FAQ entry on this issue on `r-inla.org`)}

**Example:

```
data(Seeds)
formula = r ~ x1*x2+f(plate,model="iid")
result = inla(formula,data=Seeds,family="binomial",Ntrials=n, inla.call="remote")
```

Notes

Submit and retrieve jobs

Commands:

```
  inla.qget(id, remove = TRUE)
  inla.qdel(id)
  inla.qstat(id)
  inla.qnuke()
```

Example:

```
data(Seeds)
formula = r ~ x1*x2+f(plate,model="iid")
result = inla(formula,data=Seeds,family="binomial",Ntrials=n, inla.call="submit")
inla.qstat()
result= inla.qget(result, remove=FALSE)
```

Notes

Submit and retrieve jobs

Tips

- Save the temporary result object to a file
- Use this functionality when doing Cross-Validation!
- CV is perfectly parallelizable
- You can work on two projects at once

Notes

Advanced features

Notes

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

Notes

Feature: `replicate`

Notes

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

Notes

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)

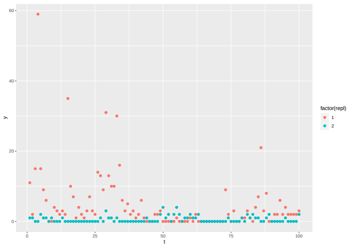
Notes

Example : simulate data

```
# 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))
```

Notes

Example : simulate data



Notes

Model choice and model assessment/validation Remote computing Advanced features Feature: replicate Feature: group Feature: Multiple likelihoods Feature: copy Feature: rgeneric Conclusion

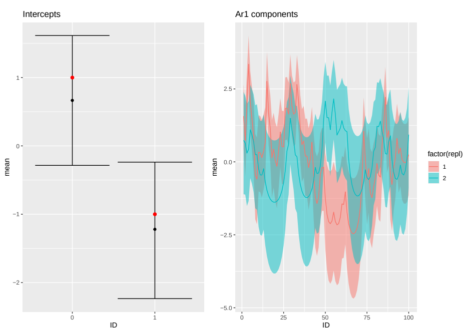
Example: fit the model

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

Notes

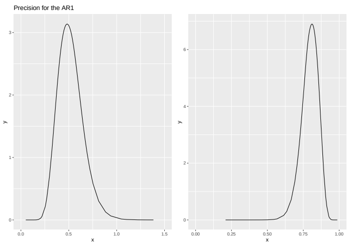
Model choice and model assessment/validation Remote computing Advanced features Feature: replicate Feature: group Feature: Multiple likelihoods Feature: copy Feature: rgeneric Conclusion

Example: Results - Latent field



Notes

Example: Results - Hyperparameters



Notes

Feature: `group`

Notes

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

Notes

Feature: Multiple likelihoods

Notes

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

Notes

Example: Simulate data

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

Notes

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)
```

Notes

Feature: copy

Notes

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`

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

Sometimes this is necessary

Notes

Feature: copy

The linear predictor

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

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

Notes

Feature: copy

It is also possible to include scaled copies

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

```
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).

Notes

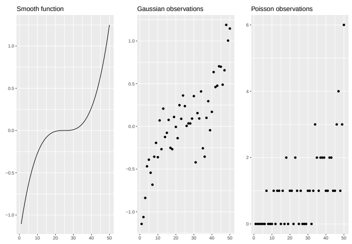
Feature: copy

$$\begin{aligned}
y_i^1 &\sim \mathcal{N}(\mu_i, \tau) \\
\mu_i &= f(i) \\
\\
y_j^2 &\sim \text{Poisson}(\lambda_j), \quad j = 1, \dots, 50 \\
\log(\lambda_j) &= f(i)
\end{aligned}$$

Notes

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))
```



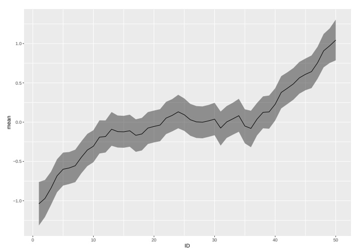
Notes

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)
```

Notes

Example: Results



Notes

Feature: rgeneric

Notes

Model choice and model assessment/validation Remote computing Advanced features Feature: replicate Feature: group Feature: Multiple likelihoods Feature: copy Feature: rgeneric Conclusion

Generic precision matrices

Commands:

- generic0, generic1,generic2,generic3
- rgeneric, cgeneric

For rgeneric the *user* must specify a function, along the lines of

```
ComputePrecisionMatrix = function (theta) {
  # theta is the vector of hyper-parameters
  # your code here
  return(Q)
}
```

together with priors and some additional details.

Examples in

```
vignette("rgeneric", package="INLA")
```

Notes

Model choice and model assessment/validation Remote computing Advanced features Feature: replicate Feature: group Feature: Multiple likelihoods Feature: copy Feature: rgeneric Conclusion

Conclusion

Notes

Summary

- INLA is a fast method to do Bayesian inference with latent Gaussian models and INLA is an R-package that implements this method with a flexible and simple interface
- The INLA approach is *not* a rival/competitor/replacement to/of MCMC, just a better option for the class of LGMs.
- The basic idea behind the INLA procedure is simple and it is very easy to use for simple problems, but for more complex problems it may happen that you have to struggle a bit.
- Ask for help to the INLA team through the `r-inla.org` discussion forum or through e-mail (see `inla.version()`) for a list of e-mails.

Notes

INLA is a widespread tool

```r
my.cran.inla <- function() {
     library("cranly")
     library("tools")
     p = clean_CRAN_db(CRAN_package_db())
     m = length(p$package)
     idx.s = which(unlist(lapply(1:m, function(ii) any(p$suggests[[ii]] %in% "INLA"))))
     idx.d = which(unlist(lapply(1:m, function(ii) any(p$depends[[ii]] %in% "INLA"))))
     return(list(suggests = p$package[idx.s], depends = p$package[idx.d]))
 }

my.cran.inla()
```

```
## $suggests
##  [1] "abn"           "AGPRIS"          "BCEA"          "bigDM"
##  [5] "DClusterm"     "disaggregation" "eCAR"          "excursions"
##  [9] "FRK"           "highriskzone"   "INLABMA"       "inlabru"
## [13] "makemyprior"   "meta4diag"      "nmaINLA"       "phyr"
## [17] "PrevMap"       "rSPDE"          "SimSurvey"     "spatialfusion"
## [21] "SUMMER"        "surveillance"   "survHE"
##
## $depends
## character(0)
```

Notes

INLA is an active project

- A big push into including more general likelihoods and larger problems
- New features for detecting prior sensitivity and specifying priors for parame- ters (PC priors)
- Extensions beyond the basic LGM framework

Notes

Acknowledgement

- Andrea Riebler 
- The whole INLA group, especially Finn and Håvard

- **All of you!!**

Notes

Notes

Notes