Bayesian Statistics with R-INLA

University of Zurich, March, 2022

Instructor: Sara Martino

Department of Mathematical Science (NTNU)

Notes

Notes

Notes

What have we learned in the morning...

- What is a LGM
- Which kind of models are amenable to INLA
- How does INLA work....

Notes

Getting INLA Implementing the INLA algorithm How to use INLA Simple example Add random effects Prediction Smoothing binary time series Disease Mapping Changing the prior Repeated Poisson counts Control statements

What have we learned in the morning...

- What is a LGM
- Which kind of models are amenable to INLA
- How does INLA work....
- ..you have even implemented it yourself! :-)

Notes

Getting INLA Implementing the INLA algorithm How to use INLA Simple example Add random effects Prediction Smoothing binary time series Disease Mapping Changing the prior Repeated Poisson counts Control statements

Good News!

All the theory we have seen is wrapped up in the R-package `INLA` which is easy to use.

Notes

Getting INLA

Notes

Getting INLA

- The web page www.r-inla.org contains source-code, worked-through examples, reports and instructions for installing the package.

Notes

Getting INLA

- The R-package INLA works on Linux, Windows and Mac and can be installed within R by

```
# stable version
install.packages("INLA",
          repos=c(getOption("repos"),
          INLA="https://inla.r-inla-download.org/R/stable"),
          dep=TRUE)

# devel version
install.packages("INLA",
        repos=c(getOption("repos"),
        INLA="https://inla.r-inla-download.org/R/testing"),
        dep=TRUE)
```

and then upgraded in R as:

```
inla.upgrade(testing = TRUE)
```

NB You need R version 4.1 or newer!!

Notes

Which INLA version do I have?

Notes

```
inla.version()
```

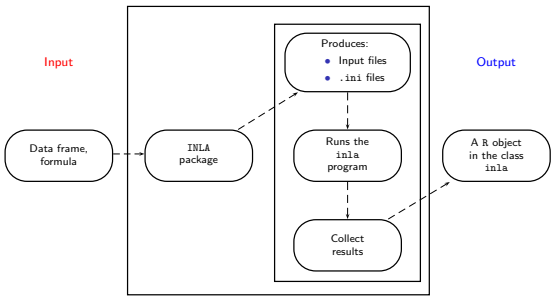
```
##  R-INLA version ..........: 22.02.16-2
##  Date ....................: Wed Feb 16 02:22:38 PM +03 2022 (Version_22.02.16-2)
##  Maintainers .............: Havard Rue <hrue@r-inla.org>
##                           : Finn Lindgren <finn.lindgren@gmail.com>
##                           : Elias Teixeira Krainski <elias@r-inla.org>
##  Main web-page ...........: www.r-inla.org
##  Download-page ...........: inla.r-inla-download.org
##  Repository ..............: github.com/hrue/r-inla
##  Email support ...........: help@r-inla.org
##                           : r-inla-discussion-group@googlegroups.com
```


Implementing the INLA algorithm

Notes

The INLA package for R

Notes



What happens in the black box?

The implementation of the INLA method consists of three parts:

- **GMRFLib-Library:** A library for GMRFs written in `C`
- **`inla`-program:** The implementation of INLA written in `C`
- **`INLA` package for `R`:** An `R`-interface to the `inla`-program

  The first two are *not* particularly user-friendly. They are used in the background by the `INLA` package.

Notes

Implementing INLA

All procedures required to perform INLA need to be carefully implemented to achieve a good speed; easier to implement a slow version of INLA.

Notes

Implementing INLA

All procedures required to perform INLA need to be carefully implemented to achieve a good speed; easier to implement a slow version of INLA.

- **The `GMRFLib`-library**
  - Basic library written in `C`, user friendly for programmers

Notes

Implementing INLA

All procedures required to perform INLA need to be carefully implemented to achieve a good speed; easier to implement a slow version of INLA.

- **The `GMRFLib`-library**
- **The `inla`-program**
  - Define *latent Gaussian models* and interface with the `GMRFLib`-library
  - Avoids the need for `C`-programming
  - Models are defined using `.ini`-files
  - Requires to write input files in a special format
  - `inla`-program write all the results (E/Var/marginals) to files

Notes

Implementing INLA

All procedures required to perform INLA need to be carefully implemented to achieve a good speed; easier to implement a slow version of INLA.

- **The `GMRFLib`-library**
- **The `inla`-program**
- **The `INLA` package for `R`**
  - `R`-interface to the `inla`-program. (That's why its not on CRAN.)
  - Convert `formula`-statements into `.ini`-files definitions
  - It also does much more (for example for survival models or when using `inlabru`)

Notes

How to use `INLA`

Notes

How to use INLA

There are essentially four parts to an INLA-program:

1. Data organisation: Make an object to store response, covariates,

```
data = data.fame(y = y, x = x)
```

2. Use the 'formula'-notation to specify the model (similar to lm and glm functions)

```
formula = y~x
```

3. Call the 'inla'-program

```
res = inla(formula, data=data, family="gaussian")
```

4. Extract posterior information, e.g.for a first overview use

```
summary(res)
```

Notes

Data organization

The responses and covariates are collected in a list or data frame. Assume response y, covariates x1 and x2, and time index t. Then they can be organized with:

```
# Option 1

data = list(y = y, x1 = x1, x2 = x2, t = t)

# Option 2

data = data.frame(y = y, x1 = x1, x2 = x2, t = t)
```

Notes

formula: specifying the linear predictor

The model is specified through a 'formula' similar to glm:

```
formula = y ~ x1 + x2 + f(t, ...)
```

- y is the name of the response in the data object
- The fixed effects are given i.i.d. Gaussian priors
- The f() function specifies random effects (e.g. temporal, spatial, smooth effect of covariates and Besag model)
- Use -1 in the formula if you don't want an automatic intercept

Notes

The inla() function

```
result = inla(
  # Description of linear predictor
  formula,
  # Likelihood
  family = "gaussian",
  # List or data frame with response,
  # covariates, etc.
  data = data,
  ## This is all that is needed for a basic call

  ## # check what happens
  verbose = TRUE,
  # ,..., there are also some "control statements"
  # to customize things
  # This you need if you later want to sample from the
  # fitted model
  control.compute=list(config = TRUE)
  )
```

Notes

Likelihood functions

- gaussian
- T
- poisson
- nbinomial
- binomial
- exponential
- weibull
- gev
- coxph

For a complete list type

```
names(inla.models()$likelihood)
```

Notes

Posterior inference

Main functions:

```
# look at a  first summary
summary(result)
# plot the main results
# (does not use ggplot...)
plot(result)
# rerun the model to get better
# estimate of the hyperparemeters
result2 = inla.hyperpar(result)
# sample from the fitted model
# this can be very useful sometimes!
sample = inla.posterior.sample(results)
```

Notes

Simple example

Notes

Example: Simple linear regression

- **Stage 1:** Gaussian likelihood

$$y_i|\eta_i \sim \mathcal{N}(\eta_i, \sigma^2)$$

- **Stage 2:** Covariates are connected to likelihood by

$$\eta_i = \beta_0 + \beta_1 x_i$$

- **Stage 3:** σ^2: variance of observation noise

Notes

Example: Simple linear regression

```
# Generate data
x = runif(10)
y = 1 + 2*x + rnorm(n = 100, sd = 0.1)

# Run inla
formula = y ~ 1 + x
result = inla(formula,
              data = data.frame(x = x, y = y),
              family = "gaussian")
```

Notes

Organization of the inla-object

```
names(result)
```

```
##  [1] "names.fixed"               "summary.fixed"
##  [3] "marginals.fixed"           "summary.lincomb"
##  [5] "marginals.lincomb"         "size.lincomb"
##  [7] "summary.lincomb.derived"   "marginals.lincomb.derived"
##  [9] "size.lincomb.derived"      "mlik"
## [11] "cpo"                       "po"
## [13] "waic"                      "model.random"
## [15] "summary.random"            "marginals.random"
## [17] "size.random"               "summary.linear.predictor"
## [19] "marginals.linear.predictor" "summary.fitted.values"
## [21] "marginals.fitted.values"   "size.linear.predictor"
## [23] "summary.hyperpar"          "marginals.hyperpar"
## [25] "internal.summary.hyperpar" "internal.marginals.hyperpar"
## [27] "offset.linear.predictor"   "model.spde2.blc"
## [29] "summary.spde2.blc"         "marginals.spde2.blc"
## [31] "size.spde2.blc"            "model.spde3.blc"
## [33] "summary.spde3.blc"         "marginals.spde3.blc"
## [35] "size.spde3.blc"            "logfile"
## [37] "misc"                      "dic"
## [39] "mode"                      "joint.hyper"
## [41] "nhyper"                    "version"
## [43] "Q"                         "graph"
## [45] "ok"                        "cpu.used"
## [47] "all.hyper"                 ".args"
## [49] "call"                      "model.matrix"
```

Notes

Organization of the inla-object

You can find summary information in

```
##  [1] "summary.fixed"               "summary.lincomb"
##  [3] "summary.lincomb.derived"     "summary.random"
##  [5] "summary.linear.predictor"    "summary.fitted.values"
##  [7] "summary.hyperpar"            "internal.summary.hyperpar"
##  [9] "summary.spde2.blc"           "summary.spde3.blc"
```

for example

```
result$summary.fixed
```

```
##                    mean         sd 0.025quant  0.5quant 0.975quant      mode
## (Intercept) 0.9886826 0.02279676   0.943834 0.9886819   1.033493 0.9886825
## x           2.0286140 0.04573803   1.938632 2.0286128   2.118518 2.0286143
##                      kld
## (Intercept) 3.088291e-06
## x           3.088224e-06
```

Notes

Organization of the inla-object

You can find estimated posterior marginals in

```
##  [1] "marginals.fixed"               "marginals.lincomb"
##  [3] "marginals.lincomb.derived"     "marginals.random"
##  [5] "marginals.linear.predictor"    "marginals.fitted.values"
##  [7] "marginals.hyperpar"            "internal.marginals.hyperpar"
##  [9] "marginals.spde2.blc"           "marginals.spde3.blc"
```

Each object is thereby a list. Get the marginal for intercept:

```
  head(result$marginals.fixed[[1]])
```

```
##              x           y
## [1,] 0.7603140 1.865103e-17
## [2,] 0.8059877 2.998142e-11
## [3,] 0.8516614 1.689587e-06
## [4,] 0.8636612 1.990772e-05
## [5,] 0.8744983 1.599353e-04
## [6,] 0.8754425 1.905200e-04
```

Notes

Organization of the inla-object

Further general information

```
# formula used
result$.args$formula
```

```
## y ~ 1 + x
## NULL
```

```
# data used
result$.args$data[1:3,]
```

```
##           x        y
## 1 0.3457213 1.648559
## 2 0.4242941 1.910131
## 3 0.8127039 2.389109
```

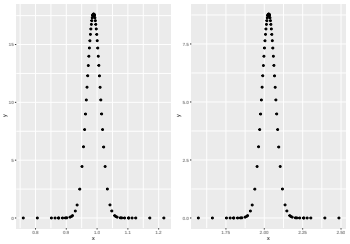
```
# log-file including information of INLA approximations
result$logfile
```

Notes

Marginal posterior densities

The marginal posterior densities are stored as a matrices with x- and y-values

```
intercept = data.frame(result$marginals.fixed$`(Intercept)`)
x = data.frame(result$marginals.fixed$x)
p1 = ggplot(data = intercept) + geom_point(aes(x,y))
p2 = ggplot(data = x) + geom_point(aes(x,y))
p1+p2
```

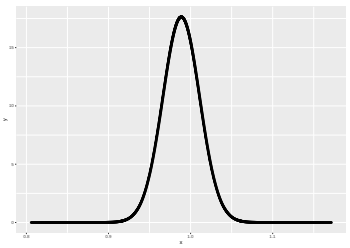


Notes

Marginal posterior densities

The rough shape can be interpolated to higher resolution using the `inla.smarginal()` function:

```
smoother_dens = data.frame(inla.smarginal(intercept))
ggplot(data = smoother_dens) + geom_point(aes(x,y))
```



Notes

Marginal posterior densities

Manipulation of the computed posterior marginals is possible through the `inla.*marginal()` functions:

```
# compute the 0.05 quantile
inla.qmarginal(0.05, intercept)
```

```
## [1] 0.9511269
```

```
# Distribution function
inla.pmarginal(0.975, intercept)
```

```
## [1] 0.2742024
```

```
# Density function
inla.dmarginal(1, intercept)
```

```
## [1] 15.53033
```

```
# Generate realizations
inla.rmarginal(4, intercept)
```

```
## [1] 0.9730758 0.9857803 0.9878979 0.9635838
```

Notes

Other `inla.*marginal()` functions.

Function Name	Usage
`inla.dmarginal(x, marginal, ...)`	Density at a vector of evaluation points x
`inla.pmarginal(q, marginal, ...)`	Distribution function at a vector of quantiles q
`inla.qmarginal(p, marginal, ...)`	Quantile function at a vector of probabilities p.
`inla.rmarginal(n, marginal)`	Generate n random deviates
`inla.hpdmarginal(p, marginal, ...)`	Compute the highest posterior density interval at level p
`inla.emarginal(fun, marginal, ...)`	Compute the expected value of the marginal assuming the transformation given by fun
`inla.mmarginal(marginal)`	Compute the mode
`inla.smarginal(marginal, ...)`	Smoothed density in form of a list of length two. The first entry contains the x-values, the second entry includes the interpolated y-values
`inla.tmarginal(fun, marginal, ...)`	Transform the marginal using the function fun.
`inla.zmarginal(marginal)`	Summary statistics for the marginal

Notes

Add random effects

Notes

Add random effects

```
f(name, model="...", hyper=...,
                  constr=FALSE, cyclic=FALSE, ...)
```

- `name` – the index of the effect (each f-function needs its own!)
- `model` – the type of latent model. E.g.`iid`, `rw2`, `ar1`, `besag`, and so on
- `hyper` – specify the prior on the hyperparameters
- `constr` – sum-to-zero constraint?
- `cyclic` – are you cyclic?
- `...`

Notes

Example: Add random effect

Add an AR(1) random effect to the linear predictor.

- **Stage 1:**
$$y_i|\eta_i \sim \mathcal{N}(\eta_i, \sigma^2)$$
- **Stage 2:** Covariates and AR(1) component connected to likelihood by
$$\eta_i = \beta_0 + \beta_1 x_i + a_i$$
- **Stage 3:**
  - σ^2: variance of observation noise
  - ρ: dependence in AR(1) process
  - σ^2: variance of the innovations in AR(1) process

Notes

Example: Add random effect

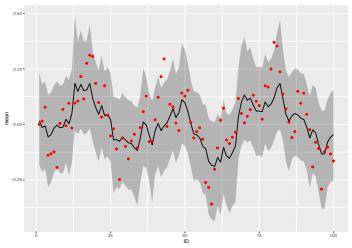
```
# Generate AR(1) sequence
set.seed(580258)
t = 1:100
rho = 0.8
sd_ar1 = 0.1
ar = rep(0,100)
for(i in 2:100)
  ar[i] = rho * ar[i-1] + rnorm(n = 1, sd = sd_ar1)
# Generate data with AR(1) component
x = runif(100)
y = 1 + 2*x + ar + rnorm(n = 100, sd = 0.2)

# Run inla
formula = y ~ 1 + x + f(t, model="ar1")

result = inla(formula,
     data = data.frame(x = x, y = y, t = t),
     family = "gaussian")
```

Notes

Example

Estimates of the random effect

```
result$summary.random$t %>% ggplot() +
  geom_line(aes(ID, mean)) +
  geom_ribbon(aes(ID, ymin = `0.025quant`, ymax = `0.975quant`), alpha = 0.3) +
  geom_point(data = data.frame(t =t , ar = ar), aes(t,ar), color = "red")
```

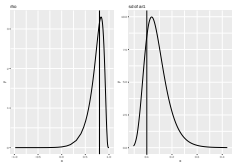


Notes

Example

Estimates of the hyperparameters

```
# rho
p1 = ggplot() + geom_line(data = data.frame(result$marginals.hyperpar$`Rho for t`),
                          aes(x,y)) +
  geom_vline(xintercept = rho) + ggtitle("rho")
# sd of the ar1 effect
prec = result$marginals.hyperpar$`Precision for t`
sd = inla.tmarginal(function(x) 1/sqrt(x), prec)
p2 = ggplot() + geom_line(data = data.frame(sd), aes(x,y)) +
  geom_vline(xintercept =sd_ar1 ) + ggtitle("sd of ar1")
p1+p2
```



Notes

Prediction

Notes

The interpretation of NA

R-INLA uses NA differently than other packages

- NA in the **response** means no likelihood contribution, i.e.response is unobserved
- NA in a **fixed effect** means no contribution to the linear predictor, i.e. the covariate is set equal to zero
- NA in a **random effect** f(...) means no contribution to the linear predictor

Notes

Prediction

The distribution of the linear predictor at an unobserved location can be computed by specifying the value of the covariate x and the desired time index t and set y to NA.

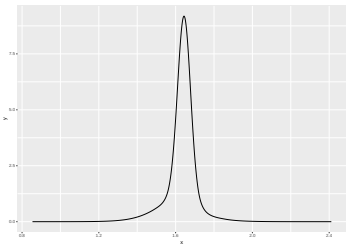
```
# Add one new location
n = 1
x = c(x, runif(n))
t = c(t, 101:(100+n))
y = c(y, rep(NA,n))

# Re-compute
result.pred = inla(formula,
    data = data.frame(x = x, t = t, y = y),
    family="gaussian",
    control.inla = list(int.strategy = "grid"),
    control.compute = list(config = TRUE,
                           return.marginals.predictor=TRUE),
    # tell inla to return the marginals for eta!
    control.predictor = list(compute = TRUE))
```

Notes

Prediction

Predicted marginal of the linear predictor η_{101}

```
pred = result.pred$marginals.linear.predictor[[100+n]]
pred = inla.smarginal(pred)
ggplot() +
  geom_line(data = data.frame(pred), aes(x, y))
```



Notes

Prediction

Caution: This is **not** yet the predictive distribution, as the observation noise is missing.

The predictive distribution is

$$\pi(y_{101}|\mathbf{y})$$

what we got is

$$\pi(\eta_{101}|\mathbf{y})$$

Notes

Prediction

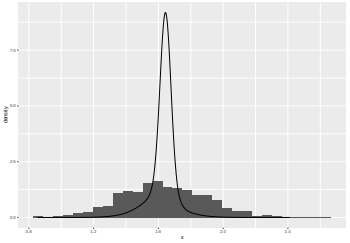
One way to add the observation noise to the linear predictor is by sampling from the posterior distribution.

```
n = 1000
x = inla.posterior.sample(n, result.pred)

func = function(...)
{
  eta = Predictor
  eta = eta[101]
  sd = 1/sqrt(theta[1])
  out = rnorm(1, mean = eta, sd =sd)
  return(out)
}

samples = inla.posterior.sample.eval(func, x)[1,]
```

Notes

Prediction

Comparing $\pi(y_{101}|\mathbf{y})$ and $\pi(\eta_{101}|\mathbf{y})$



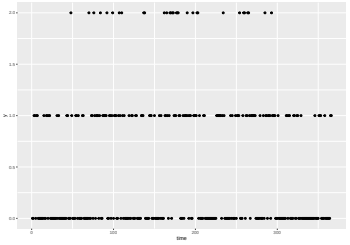
Notes

Smoothing binary time series

Notes

Example: Smoothing binary time series

The data set `Tokyo` is available in the `INLA` package and consists of the number of days in Tokyo with rainfall above 1 mm in 1983–1984.



Notes

Observations

Each observation consists of

- t: Day of year; $t \in \{1, 2, \ldots, 366\}$
- n_t: Number of observations for day t in 1983–1984; $n_t \in \{1, 2\}$
- y_t: Number of days with rain out of n_t days for day t; $y_t \in \{0, 1, 2\}$

```
 data(Tokyo)
head(Tokyo,3)
```

```
##   y n time
## 1 0 2    1
## 2 0 2    2
## 3 1 2    3
```

```
Tokyo[60,]
```

```
##    y n time
## 60 0 1   60
```

Notes

Hierarchical model

- **Stage 1:** We have binomial responses with known n_t, but unknown probabilities

$$y_t \sim \text{Binomial}(n_t, p_t)$$

- **Stage 2:** A cyclic second order random walk (CRW2) is connected to the likelihood by

$$p_t = \frac{\exp(\eta_t)}{1+\exp(\eta_t)} \text{ with linear predictor } \eta_t = \text{CRW2}_t$$

- **Stage 3:**
  - τ: Scale parameter in CRW2 with prior

$$\pi(\tau) \sim \text{Gamma}(1, 5 \cdot 10^{-5})$$

Notes

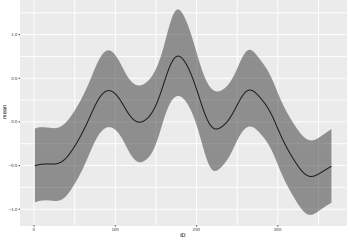
INLA implementation

```
# Read data
data(Tokyo)
# Specify linear predictor
formula = y ~ -1 + f(time, model="rw2", cyclic=TRUE)
# Run model
result = inla(formula,
              family = "binomial",
              Ntrials = n,
              data = Tokyo)
```

Notes

Marginal posterior of CRW2

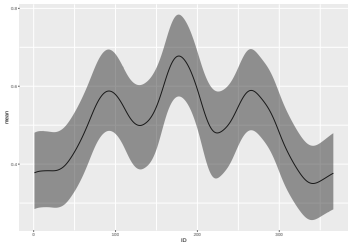
```
ggplot(data = result$summary.random$t) +
  geom_line(aes(ID, mean)) +
  geom_ribbon(aes(ID, ymin = `0.025quant`, ymax = `0.975quant` ),
              alpha = 0.5)
```



Notes

Transform to probability

```
pred = result$summary.fitted.values
pred$ID = 1:dim(Tokyo)[1]
ggplot(data = pred) +
  geom_line(aes(ID, mean)) +
  geom_ribbon(aes(ID, ymin = `0.025quant`, ymax = `0.975quant` ),
              alpha = 0.5)
```



Notes

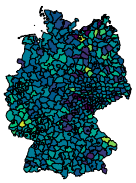
Disease Mapping

Notes

Example: disease mapping

We observed larynx cancer mortality counts for males in 544 district of Germany from 1986 to 1990 and want to make a model.

- y_i: The count at location i.
- E_i: An offset; expected number of cases in district i.
- c_i: A covariate (level of smoking consumption) at i
- $\boldsymbol{s}_i$: spatial location i .



Notes

Disease mapping

Assume

$$Y_i \mid \eta_i \sim \text{Poisson}(E_i \exp(\eta_i))$$

where the log relative risk is decomposed into

$$\eta_i = \mu + u_i + v_i$$

- μ is the overall level (intercept).
- $v_i \sim \mathcal{N}(0, \tau_v^{-1})$ represents non-spatial overdispersion.
- u_i are random effects with spatial structure.

Notes

A spatially structured effect

To incorporate a spatial structure into a model, the so called Besag model is often used.

$$\begin{aligned} p(\mathbf{u} \mid \kappa_\mathbf{u}) &\propto \kappa_u^{(n-1)/2} \exp\left(-\frac{\kappa_u}{2} \sum_{i \sim j} (u_i - u_j)^2\right) \\ &= \kappa_u^{(n-1)/2} \exp\left(-\frac{\kappa_u}{2} \mathbf{u}^\mathbf{T} \mathbf{R} \mathbf{u}\right). \end{aligned}$$

where R is called structure matrix and defined as

$$R_{ij} = \begin{cases} n_i & i = j \\ -1 & i \sim j \\ 0 & \text{otherwise.} \end{cases}$$

Here, $i \sim j$ denotes that i and j are neighbouring regions.

Notes

What does this mean?

Example: Five counties of the US state Rhode Island

The structure matrix $\mathbf{R}$ defines the neighborhood structure.

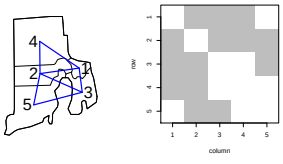


Figure 1: Adjacency matrix

3	-1	-1	- 1	0
-1	4	-1	-1	-1
-1	-1	3	0	-1
-1	-1	0	2	0
0	-1	-1	0	2

Table 1: Structure matrix $\mathbf{R}$

With increasing number of regions $\mathbf{R}$ will be sparse, which allows to do many computations very efficient.

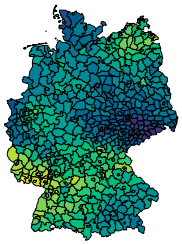
Notes

INLA code

```
library(spam)
# load the dataset
data(Oral)
# load the file including neighbourhood information
g = system.file("demodata/germany.graph", package="INLA")
# add one column
Oral = cbind(Oral, region = 1:544, region.unstruc= 1:544)
# define formula
formula = Y ~ f(region, model="besag", graph=g) +
                          f(region.unstruc, model="iid")
# run the model
result = inla(formula, family="poisson", E=E, data=Oral)
```

Notes

Median of u on exp-scale



Notes

Other choices for `f`-terms

```
##  [1] "linear"       "iid"          "mec"          "meb"          "rgeneric"
##  [6] "cgeneric"     "rw1"          "rw2"          "crw2"         "seasonal"
## [11] "besag"        "besag2"       "bym"          "bym2"         "besagproper"
## [16] "besagproper2" "fgn"          "fgn2"         "ar1"          "ar1c"
## [21] "ar"           "ou"           "intslope"     "generic"      "generic0"
## [26] "generic1"     "generic2"     "generic3"     "spde"         "spde2"
## [31] "spde3"        "iid1d"        "iid2d"        "iid3d"        "iid4d"
## [36] "iid5d"        "iidkd"        "2diid"        "z"            "rw2d"
## [41] "rw2diid"      "slm"          "matern2d"     "dmatern"      "copy"
## [46] "clinear"      "sigm"         "revsigm"      "log1exp"      "logdist"
```

Notes

Changing the prior

Notes

Changing the prior: Internal scale

- Hyperparameters are represented internally with more well-behaved transformations, e.g. correlation ρ and precision τ are internally

$$\theta_1 = \log(\tau)$$
$$\theta_2 = \log\left(\frac{1+\rho}{1-\rho}\right)$$

- The prior must be set on the parameter in **internal scale**
- Initial values for the mode-search must be set in **internal scale**
- The functions `to.theta()` and `from.theta()` can be used to map back and forth.

Notes

Changing the prior: Code

```
hyper = list(prec = list(prior = "loggamma",
                         param = c(1, 0.1),
                         initial = 4,
                         fixed = FALSE))

formula = y ~ f(idx, model = "iid", hyper = hyper) + ...
```

```
# For the iid model, default options can be seen with
inla.doc("iid")
```

Notes

Repeated Poisson counts

Notes

EPIL example

Seizure counts in a randomised trial of anti-convulsant therapy in epilepsy. From `WinBUGS` manual.

```
## # A tibble: 6 x 8
##     Ind Repl1 Repl2 Repl3 Repl4   Trt  Base   Age
##   <int> <dbl> <dbl> <dbl> <dbl> <int> <int> <int>
## 1     1     5     3     3     3     0    11    31
## 2     2     3     5     3     3     0    11    30
## 3     3     2     4     0     5     0     6    25
## 4     4     4     4     1     4     0     8    36
## 5     5     7    18     9    21     0    66    22
## 6     6     5     2     8     7     0    27    29
```

Covariates are treatment (0,1), 8-week baseline seizure counts, and age in years.

Notes

Repeated Poisson counts

$$
\begin{aligned}
y_{jk} &\sim \text{Poisson}(\mu_{jk});\ j = 1, \ldots, 59;\ k = 1, \ldots, 4 \\
\log(\mu_{jk}) &= \alpha_0 + \alpha_1 \log(\text{Base}_j/4) + \alpha_2 \text{Trt}_j \\
&\quad + \alpha_3 \text{Trt}_j \log(\text{Base}_j/4) + \alpha_4 \log(\text{Age}_j) \\
&\quad + \alpha_5 V4 + \text{Ind}_j + \beta_{jk} \\
\alpha_i &\sim \mathcal{N}(0, \tau_\alpha) \qquad \tau_\alpha \ \text{ known } (0.001) \\
\text{Ind}_j &\sim \mathcal{N}(0, \tau_{\text{Ind}}) \quad \tau_{\text{Ind}} \sim \text{Gamma}(1, 0.01) \\
\beta_{jk} &\sim \mathcal{N}(0, \tau_\beta) \qquad \tau_\beta \sim \text{Gamma}(1, 0.01)
\end{aligned}
$$

Here, `V4` is an indicator variable for the 4th visit.

Notes

Model specification in INLA

The data:

```
##   y Trt Base Age V4 rand Ind
## 1 5   0   11  31  0    1   1
## 2 3   0   11  31  0    2   1
## 3 3   0   11  31  0    3   1
## 4 3   0   11  31  1    4   1
## 5 3   0   11  30  0    5   2
## 6 5   0   11  30  0    6   2
```

The formula:

```
formula = y ~ ClBase4*CTrt + ClAge + CV4 +
              f(Ind, model="iid",
                 hyper = list(prec = list(prior = "loggamma",
                 param = c(1,0.01)))) +
              f(rand, model="iid",
                hyper = list(prec = list(prior = "loggamma",
                                         param = c(1,0.01))))
```

Run the model:

```
result = inla(formula, family="poisson", data = Epil,
              control.fixed = list(prec.intercept = 0.001,
                                   prec = 0.001))
```

Notes

Comparing results with MCMC

- When comparing the results of `R-INLA` with MCMC, it is important to use the same model.

That means, same data, same priors, same constraints on parameters, intercept included or not, ...

Notes

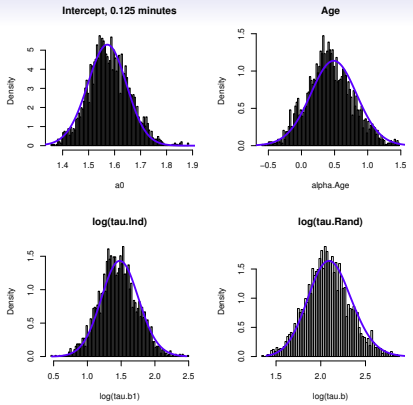
Comparing results with MCMC

- When comparing the results of `R-INLA` with MCMC, it is important to use the same model.

That means, same data, same priors, same constraints on parameters, intercept included or not, ...

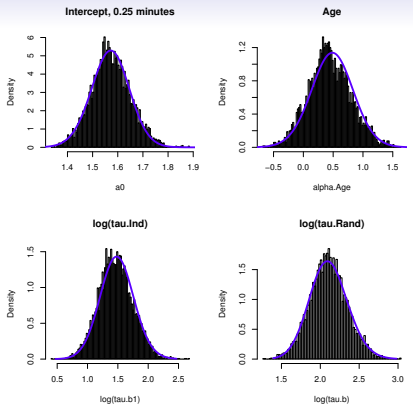
- Here we have compared the results with those obtained using '`JAGS` via the `rjags` package

Notes



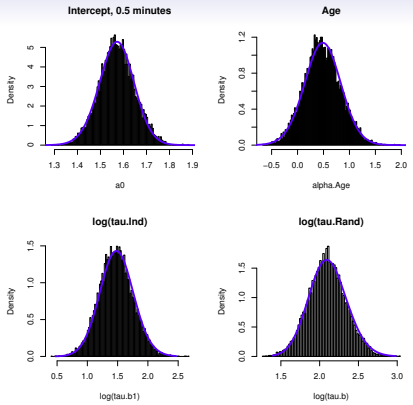
Running time of INLA < 0.5 seconds

Notes



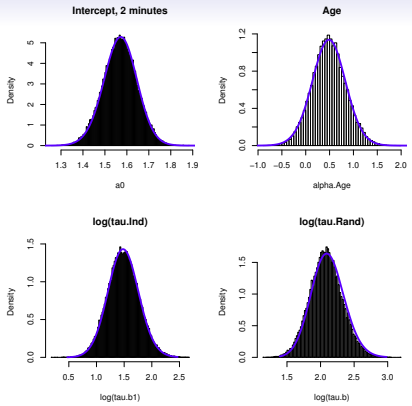
Running time of INLA < 0.5 seconds

Notes



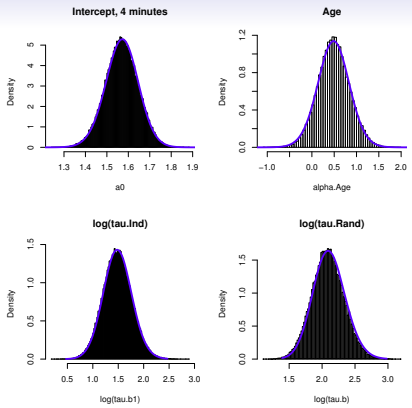
Running time of INLA < 0.5 seconds

Notes



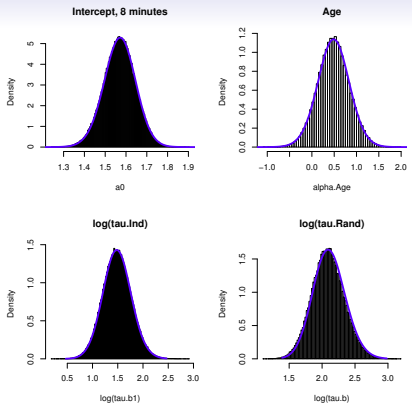
Running time of INLA < 0.5 seconds

Notes



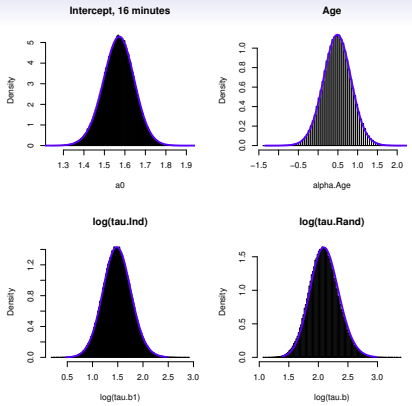
Running time of INLA < 0.5 seconds

Notes



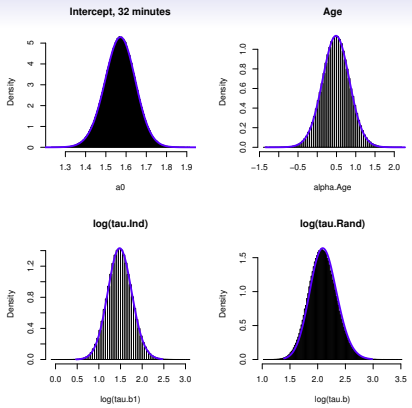
Running time of INLA < 0.5 seconds

Notes



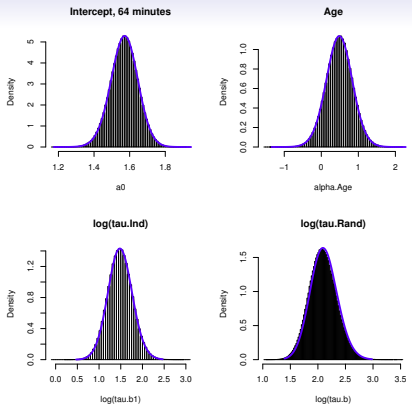
Running time of INLA < 0.5 seconds

Notes



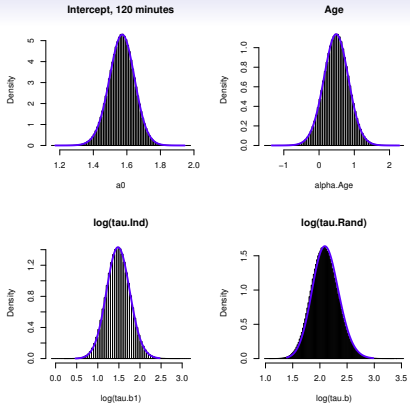
Running time of INLA < 0.5 seconds

Notes



Running time of INLA < 0.5 seconds

Notes



Running time of INLA < 0.5 seconds

Notes

Control statements

Notes

Control statements

`control.xxx` statements control computations

- `control.fixed`
  - `prec`: Default precision for all fixed effects except the intercept.
  - `prec.intercept`: Precision for intercept (Default: 0.0)
- 'control.predictor
  - `compute`: Compute posterior marginals of linear predictors
- `control.compute`
  - `dic`, `mlik`, `cpo`: Compute measures of fit?
    - `config`: Save internal GMRF approximations? (needed to use `inla.posterior.sample()`)
- `control.inla`

  `strategy` and `int.strategy` contain useful advanced features
- There are various others as well; see help.

Notes

Thank you for your attention!

If you have any doubts or questions, please write :
sara.martino@math.ntnu.no

Notes

Notes

Notes