NTNU
Norwegian University of
Science and Technology

# R-INLA: An R-package for INLA

Andrea Riebler <andrea.riebler@math.ntnu.no>





## Outline







## 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

— The inla-program
- Define *latent Gaussian models* and interface with C
- Avoids the need for C-programming
- Requires to write input files in a special format
- inla-program write all the results (E/Var/marginals) to files

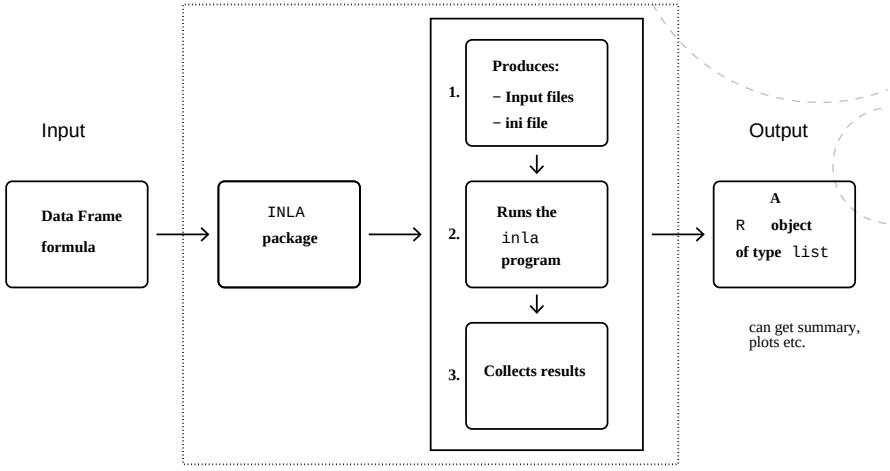
— The INLA package for R
- R-interface to the inla-program.
- Convert "formula"-statements into ".ini"-file definitions

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





## The INLA package for R



# Getting R-INLA

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

— The package can be installed by
```
> source("http://www.math.ntnu.no/inla/givemeINLA.R")
```

— Later, it can be upgraded with
```
> inla.upgrade(testing=TRUE)
```

— Works on Linux, Windows and Mac





# Documentation

See r-inla.org for documentation and worked through examples.

Tutorials:

— Basic INLA (in preparation)





# Structure of an R-program using R-INLA

There are essentially four parts to an R-INLA-program:

1. The data organization
2. The formula-notation (similar to lm and glm functions)
3. The call to the inla-program
4. The extraction of posterior information





# 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)

# Both R-structures can be extended by a further variable z,
    say, using
data["z"] = z
```

# `formula`: specifying the linear predictor

The model is specified through `formula` similar to `glm`:

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

- *y* is the name of the response in the `data`
- 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` if you don't want an automatic intercept





# 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,
    # keep working files
    keep = TRUE,

    # there are also some "control statements"
    # to customize things)
```





# Likelihood functions

- `"gaussian"`
- `"poisson"`
- `"nbinomial"`
- `"binomial"`
- `"exponential"`
- See list at `http://www.r-inla.org/models/likelihoods` or

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





# Posterior inference

Main functions:

- `summary(result)`
- `plot(result)`
- `result2 = inla.hyperpar(result)`

# Example: Simple linear regression

Stage 1: Gaussian likelihood

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

Stage 2: Covariates are connected to likelihood by

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

Stage 3: $\sigma_o^2$: variance of observation noise





# Example: Simple linear regression

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

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

# Get summary
summary(result)
```





# summary(result)

```
Call:
"inla(formula = y ~ x, data = data)"

Time used:
 Pre-processing    Running inla Post-processing          Total
         0.3198          0.0396          0.0352          0.3946

Fixed effects:
             mean     sd 0.025quant 0.5quant 0.975quant   mode kld
(Intercept) 1.0140 0.0195     0.9757   1.0140     1.0523 1.0140   0
x           2.0025 0.0316     1.9405   2.0025     2.0645 2.0025   0

The model has no random effects

Model hyperparameters:
                                       mean    sd     0.025quant 0.5quant
Precision for the Gaussian observations 114.87  16.28  85.66     113.98
                                       0.975quant mode
Precision for the Gaussian observations 149.37     112.42

Expected number of effective parameters(std dev): 2.189(0.0213)
Number of equivalent replicates : 45.69

Marginal Likelihood:  78.36
```





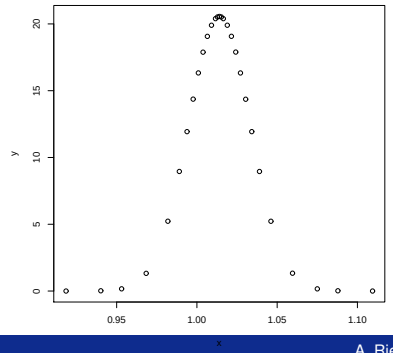
# result$summary.fixed

```
                mean         sd 0.025quant 0.5quant 0.975quant     mode
(Intercept) 1.014012 0.01949959  0.9756744 1.014011   1.052312 1.014012
x           2.002502 0.03155688  1.9404595 2.002501   2.064484 2.002502
                  kld
(Intercept) 1.167936e-12
x           4.459456e-13
```

# Marginal posterior densities

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

```
m = result$marginals.fixed[[1]]
plot(m)
```



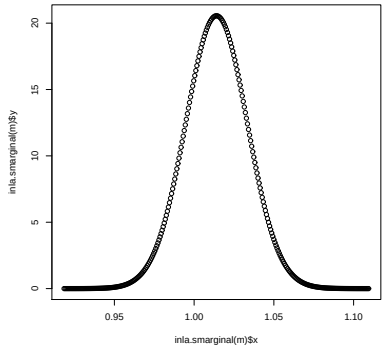





# Marginal posterior densities

The rough shape can be interpolated to higher resolution

```
plot(inla.smarginal(m))
```








# Marginal posterior densities

```
# Extract quantiles
> inla.qmarginal(0.05, m)
[1] 0.9818604

# Distribution function
> inla.pmarginal(0.975, m)
[1] 0.02314047

# Density function
> inla.dmarginal(1, m)
[1] 15.80794

# Generate realizations
> inla.rmarginal(4, m)
[1] 1.009122 1.013116 1.032004 1.007458
```





# Marginal posterior densities

```
# Calculate expected value of x and x^2
> E = inla.emarginal(function(x) c(x,x^2), m)
> E
[1] 1.014012 1.028600

# Calculate sd
> sqrt(E[2]-E[1]^2)
[1] 0.0194985

# Compare to estimate
> round(result$summary.fixed[,1:2], 3)
             mean    sd
(Intercept) 1.014 0.019
x           2.003 0.032
```

# Get estimates for variance not precision

Assume that we are interested in posterior mean and standard deviation of $\sigma_0^2 = \frac{1}{\tau_0}$. This can be easily done by selecting the appropriate posterior marginal from the output of the `inla()` function:

```
# get the marginal for the precision
> tau0 = result$marginals.hyperpar$"Precision for the Gaussian
    observations"

# Calculate expected value of 1/x and 1/x^2
> E = inla.emarginal(function(x) c(1/x,(1/x)^2), tau0)

# Calculate sd
> mysd = sqrt(E[2] - E[1]^2)

> print(c(mean=E[1], sd=mysd))
      mean         sd
0.01055980 0.00151231
```





# Add random effects

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

- `name` – the index of the effect
- `model` – the type of latent model. E.g. `"iid"`, `"rw2"`, `"ar1"`, `"besag"`, and so on
- `hyper` – specify the prior on the hyperparameters
- `constr` – do we want/need a sum-to-zero constraint?
- `cyclic` – are you cyclic?
- ...





# Example: Add random effect

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

Stage 1:

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

Stage 2: Covariates and AR(1) component connected to likelihood by

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

Stage 3:
- $\sigma_o^2$ : variance of observation noise
- $\rho$ : dependence in AR(1) process
- $\sigma^2$ : variance of the innovations in AR(1) process





# Example: Add random effect

```
# Generate AR(1) sequence
t = 1:100
ar = rep(0,100)
for(i in 2:100)
    ar[i] = 0.8*ar[i-1]+rnorm(n = 1, sd = 0.1)

# Generate data with AR(1) component
x = runif(100)
y = 1 + 2*x + ar + rnorm(n = 100, sd = 0.1)

# Run inla
formula = y ~ 1 + x + f(t, model="ar1")
result = inla(formula,
              data = list(x = x, y = y, t = t),
              family = "gaussian")

# Get summary
summary(result)
```

## summary(result)

```
Fixed effects:
             mean     sd 0.025quant 0.5quant 0.975quant   mode kld
(Intercept) 1.0354 0.0624      0.913   1.0344     1.1635 1.0328   0
x           2.0173 0.0459      1.927   2.0173     2.1077 2.0173   0

Random effects:
Name      Model
 t   AR1 model

Model hyperparameters:
                                       mean     sd       0.025quant 0.5quant
Precision for the Gaussian observations 129.8753  49.6529  60.8214   120.5645
Precision for t                          38.3033  13.9965  16.8866    36.4192
Rho for t                                 0.8031   0.0817   0.6028     0.8181
                                       0.975quant mode
Precision for the Gaussian observations 251.9389   104.1904
Precision for t                          70.9695    32.7097
Rho for t                                 0.9185     0.8463
```





## 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





## 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.

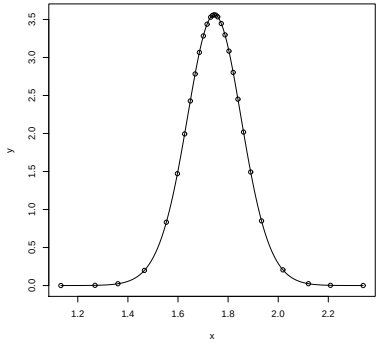
```
# Add new location
x = c(x, 0.3)
t = c(t, 101)
y = c(y, NA)

# Re-compute
result.pred = inla(formula,
                   data = list(x = x, t = t, y = y),
                   family="gaussian",
                   control.predictor = list(compute = TRUE))
```





## Prediction

```
> m = result.pred$marginals.linear.predictor[[101]]
> round(result.pred$summary.linear.predictor[101,], 3)
                mean    sd 0.025quant 0.5quant 0.975quant  mode
predictor.101 1.744 0.116      1.514    1.744      1.972 1.745
> plot(m)
> lines(inla.smarginal(m))
```



# Other choices for `f`-terms

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

```
 [1] "linear"       "iid"          "mec"          "meb"          "rgeneric"
 [6] "rw1"          "rw2"          "crw2"         "seasonal"     "besag"
[11] "besag2"       "bym"          "bym2"         "besagproper"  "besagproper2"
[16] "ar1"          "ar"           "ou"           "generic"      "generic0"
[21] "generic1"     "generic2"     "spde"         "spde2"        "iid1d"
[26] "iid2d"        "iid3d"        "iid4d"        "iid5d"        "2diid"
[31] "z"            "rw2d"         "slm"          "matern2d"     "copy"
[36] "clinear"
```





# Changing the prior: Internal scale

— Hyperparameters are represented internally with more well-behaved transformations, e.g. correlation $\rho$ and precision $\tau$ are internally represented as

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

— The prior must be set on the parameter in **internal scale**

— Initial values for must be set in **internal scale**

— The functions `to.theta` and `from.theta` can be used to map back and forth.





# 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, ...) + ...
```





# Changing the prior: Default options

```
# For the iid model, default options can be seen with
inla.doc("iid")
# or
inla.models()$latent$iid$hyper
```

```
theta
      name "log precision"
short.name "prec"
     prior "loggamma"
     param c(1e+00, 5e-05)
   initial 4
     fixed FALSE
  to.theta function(x){log(x)}
from.theta function(x){exp(x)}
```

# Changing the prior: Available models

Some of the available choices:

- `"gaussian"`
- `"loggamma"`
- `"flat"`
- `"logtgaussian"`

You can get information about the paramterisation of the `loggamma` prior, say, using

```
1 inla.doc("loggamma")
```





# EPIL example

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

| Patient | y1 | y2 | y3 | y4 | Trt | Base | Age |
|---|---|---|---|---|---|---|---|
| 1 | 5 | 3 | 3 | 3 | 0 | 11 | 31 |
| 2 | 3 | 5 | 3 | 3 | 0 | 11 | 30 |
| 3 | 2 | 4 | 0 | 5 | 0 | 6 | 25 |
| .... | | | | | | | |
| 59 | 1 | 4 | 3 | 2 | 1 | 12 | 37 |

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





# Repeated Poisson counts

$$
\begin{aligned}
y_{jk} &\sim \text{Poisson}(\mu_{jk});\ j = 1,\ldots,59;\ k = 1,\ldots,4 \\
\log(\mu_{jk}) &= \alpha_0 + \alpha_1 \log(\text{Base}_j/4) + \alpha_2 \text{Trt}_j \\
&\quad + \alpha_3 \text{Trt}_j \log(\text{Base}_j/4) + \alpha_4 \text{Age}_j \\
&\quad + \alpha_5 V4 + \text{Ind}_j + \beta_{jk}
\end{aligned}
$$

$$
\begin{array}{lll}
\alpha_i \sim \mathcal{N}(0, \tau_\alpha) & \tau_\alpha \text{ known } (0.001) \\
\text{Ind}_j \sim \mathcal{N}(0, \tau_{\text{Ind}}) & \tau_{\text{Ind}} \sim \text{Gamma}(1, 0.01) \\
\beta_{jk} \sim \mathcal{N}(0, \tau_\beta) & \tau_\beta \sim \text{Gamma}(1, 0.01)
\end{array}
$$

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





# Model specification in INLA

```
> data(Epil)
> head(Epil,n=3)
  y Trt Base Age V4 rand Ind       CTrt      ClBase4   CV4       ClAge
1 5   0   11  31  0    1   1 -0.5254237 -0.75635379 -0.25  0.11420370
2 3   0   11  31  0    2   1 -0.5254237 -0.75635379 -0.25  0.11420370
3 3   0   11  31  0    3   1 -0.5254237 -0.75635379 -0.25  0.11420370
4 3   0   11  31  1    4   1 -0.5254237 -0.75635379  0.75  0.11420370
```

```
> 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)))
```

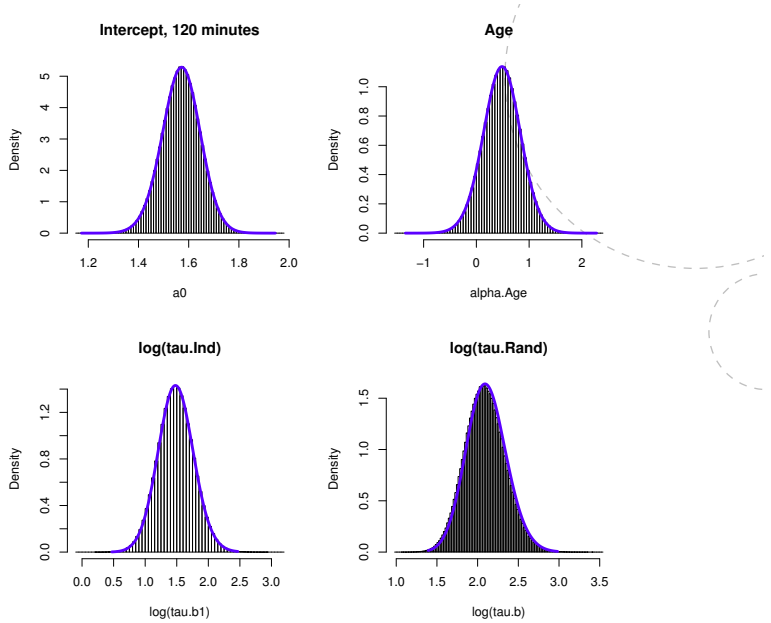
```
> result = inla(formula, family="poisson", data = Epil,
    control.fixed = list(prec.intercept = 0.001,
                         prec = 0.001))
```

# Comparing results with MCMC

— When comparing the results of R-INLA with MCMC, it is important to use same data and the same model.
That means, 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






Running time of INLA $< 0.5$ seconds





# 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: Compute measures of fit, here DIC, to do model comparison?

— There are various others as well; see help.





# Model choice

There is a need to compare and choose between various models, i.e. with covariates versus without, smoothed effects versus linear, etc.

One option to this in R-INLA is the deviance information criterion (DIC):

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

# See result
result$dic$dic
```

# Deviance information criterion

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 (measures model fit) and $p_D$ is the effective number of parameters (measures model complexity).

$\Rightarrow$ Smaller values of the DIC indicate a better trade-off between complexity and fit of the model to the data.





# Useful features

There are several features that can be used to extend the standard models in R-INLA:

- Replicate to say that two random effects should come from the same distribution.
- Correlate two random effects.
- Multiple likelihoods (parts of the data are Gaussian and parts are Poisson, for example).
- Linear combinations (difference of two random effects)
- Remote computing

However, we do not have time to cover those in this course.





# Getting help

Visit `r-inla.org` and look for similar cases

Check help pages: Online on `r-inla.org` or using `inla.doc()`.

Public help: Public discussion forum at
`r-inla-discussion-group@googlegroups.com`