# R-INLA: An R-package for INLA

February 18, 2019

1

[1] Slides are based on lecture notes kindly provided by Andrea Riebler.

# Getting R-INLA

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

# Getting R-INLA

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

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

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

Later, it can be upgraded with

```
update.packages("INLA", dep=TRUE)
```

# 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)
```

# `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"
- To see the list of available likelihood models

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

# Example: Simple linear regression

$$y_i = \underbrace{\beta_0 + \beta_1 x_i}_{\eta_i} + \epsilon_i \text{ with } \epsilon_i \sim \mathcal{N}(0, \sigma_0^2)$$

Stage 1: Gaussian likelihood

$$y_i \mid \eta_i \sim \mathcal{N}(\eta_i, \sigma_{\mathrm{o}}^2)$$

Stage 2: Covariates are connected to likelihood by

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

Stage 3: $\sigma_{\mathrm{o}}^2$: variance of observation noise

# Example: Simple linear regression

```
>   library(INLA)
>   # 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)

Call:
c("inla(formula = formula, family = \"gaussian\", data = list(x = x, ",  "    y

Time used:
 Pre-processing    Running inla Post-processing           Total
        1.0205          0.1164          0.0688          1.2057

Fixed effects:
              mean     sd 0.025quant 0.5quant 0.975quant   mode kld
(Intercept) 0.9847 0.0205     0.9444   0.9847     1.0250 0.9847   0
x           2.0392 0.0362     1.9679   2.0392     2.1104 2.0392   0

The model has no random effects

Model hyperparameters:
                                         mean    sd 0.025quant 0.5quant
Precision for the Gaussian observations 98.73 13.97      73.27    98.07
                                        0.975quant  mode
Precision for the Gaussian observations     127.94 96.75

Expected number of effective parameters(std dev): 2.059(0.0087)
Number of equivalent replicates : 48.56

Marginal log-Likelihood:  70.88
```

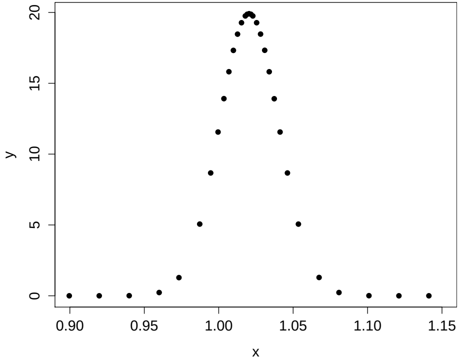
# Fixed Effects

```
>   result$summary.fixed
                 mean         sd 0.025quant  0.5quant 0.975quant      mode
(Intercept) 0.9847003 0.02048812  0.9443914 0.9846997   1.024975 0.9847002
x           2.0391946 0.03621315  1.9679478 2.0391935   2.110382 2.0391946
                     kld
(Intercept) 4.078523e-06
x           4.078518e-06
```

# Marginal posterior densities

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

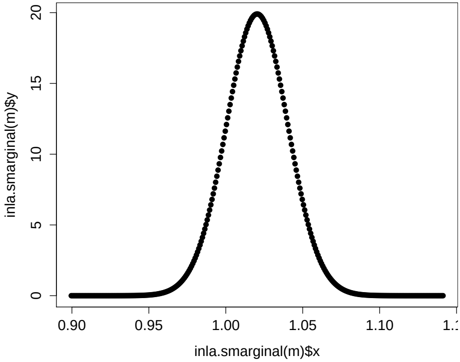
```
m = result$marginals.fixed[[1]]
plot(m)
```



# 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
```

# Organisation of the returned 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"
...
```

# 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?
- ...

# Example: Add random effect

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

Stage 1:

$$y_i | \eta_i \sim \mathcal{N}(\eta_i, \sigma_{\text{o}}^2)$$

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

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

Stage 3:

- $\sigma_{\text{o}}^2$ : 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
```

# Other choices for `f`-terms

For example:

- `rw1`, `rw2`
- `besag`
- `iid`

For a complete list see:|| `names(inla.models()$latent)`

# Changing the prior: Internal scale

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

# Changing the prior: Code

```
hyper = list(prec = list(prior = "loggamma",
                         param = c(1, 0.1))

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

# EPIL example

Seizure counts in a randomised trial of anti-convulsant 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

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

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

# Prepare the dataset

```
> library(tidyverse)
> data(Epil)
> head(Epil, n = 2)

  y Trt Base Age V4 rand Ind
1 5   0   11  31  0    1   1
2 3   0   11  31  0    2   1

> my.center = function(x) (x - mean(x))
> Epil = Epil %>% mutate(
+     CTrt    = my.center(Trt),
+     ClBase4 = my.center(log(Base/4)),
+     CV4     = my.center(V4),
+     ClAge   = my.center(log(Age)))
> Epil %>% round(2) %>% head(n=2)

  y Trt Base Age V4 rand Ind  CTrt ClBase4   CV4 ClAge
1 5   0   11  31  0    1   1 -0.53   -0.76 -0.25  0.11
2 3   0   11  31  0    2   1 -0.53   -0.76 -0.25  0.11
```

# 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
```

# 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)))
```

# 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)))
```

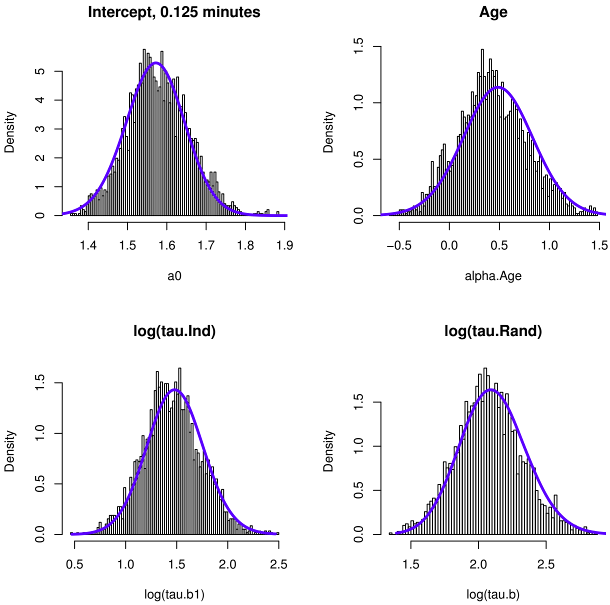
```
> result = inla(formula, family="poisson", data = Epil,
     control.fixed = list(prec.intercept = 0.001,
                          prec = 0.001))
```

# Comparing results with MCMC

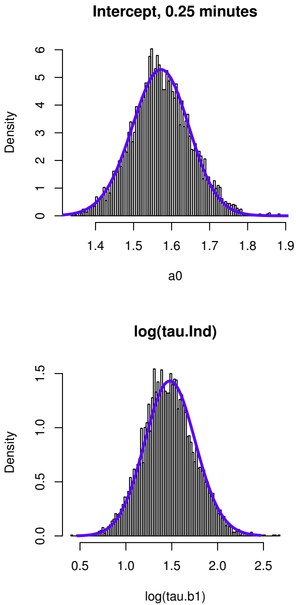
- When comparing the results of `R-INLA` with MCMC, it is important to use the same model. That means, same data, same priors, same constraints on parameters, intercept included or not, . . . .

# Comparing results with MCMC

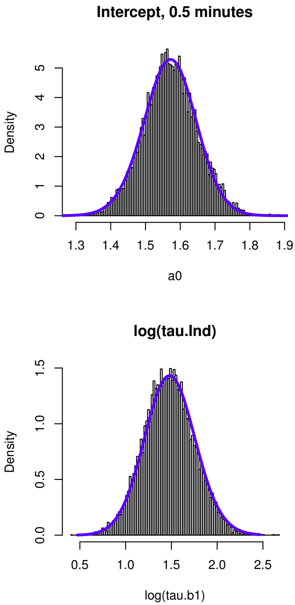
- When comparing the results of `R-INLA` with MCMC, it is important to use the same model. That means, same data, same priors, same constraints on parameters, intercept included or not, . . . .
- Here we have compared the results with those obtained using `JAGS` via the `rjags` package



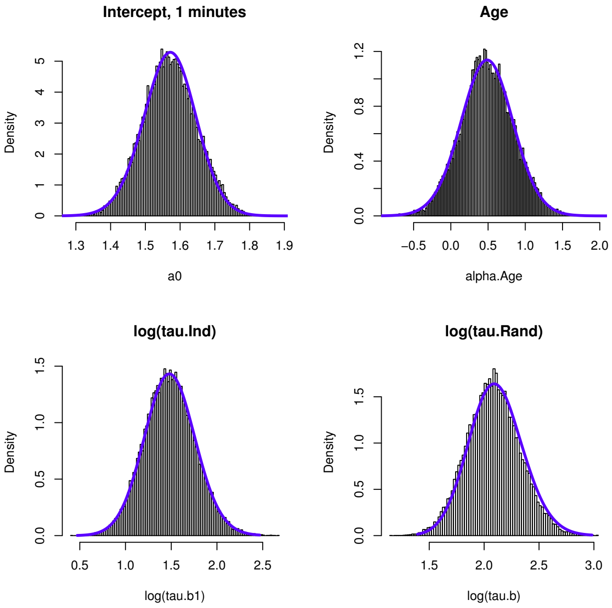

Running time of INLA $< 0.5$ seconds



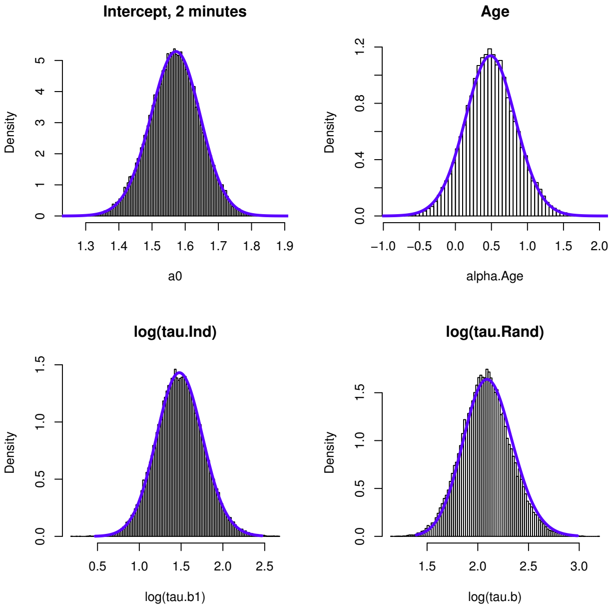

Running time of INLA $< 0.5$ seconds



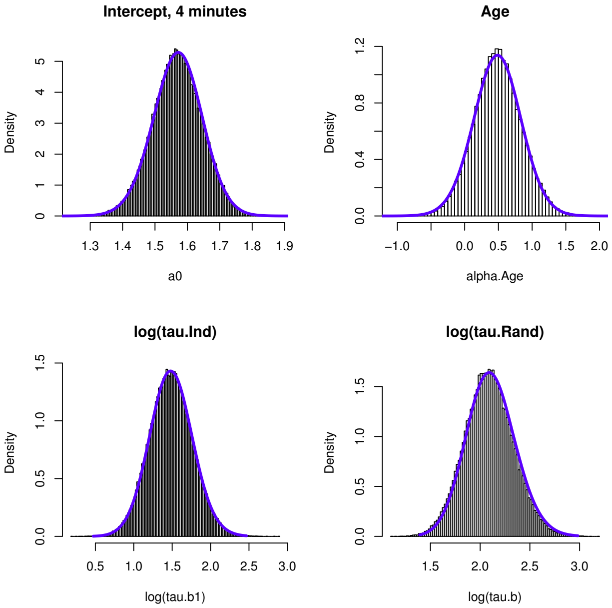

Running time of INLA $< 0.5$ seconds



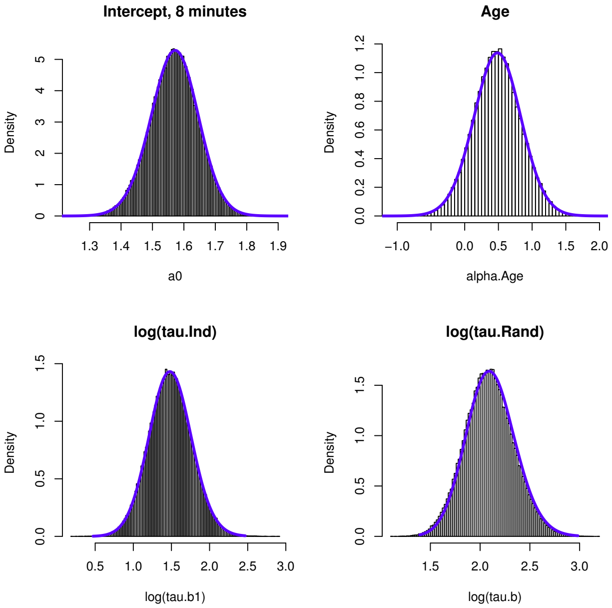

Running time of INLA $< 0.5$ seconds



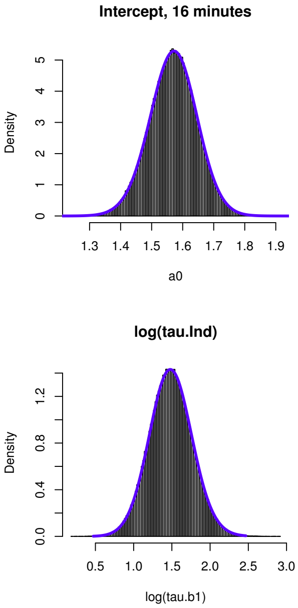

Running time of INLA $< 0.5$ seconds



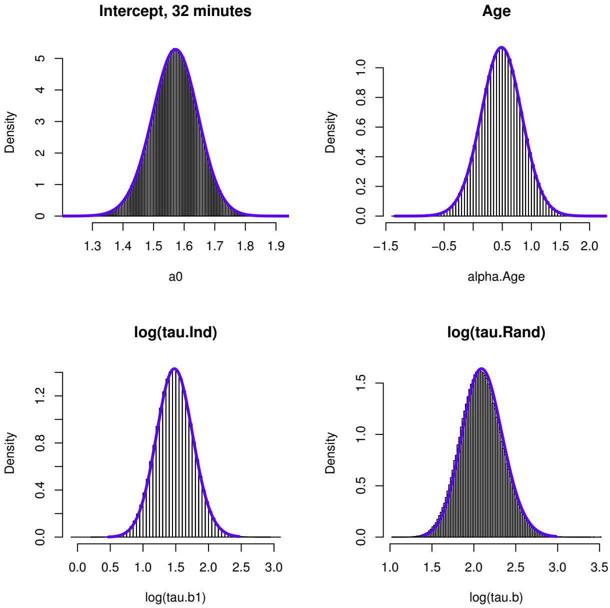

Running time of INLA $< 0.5$ seconds



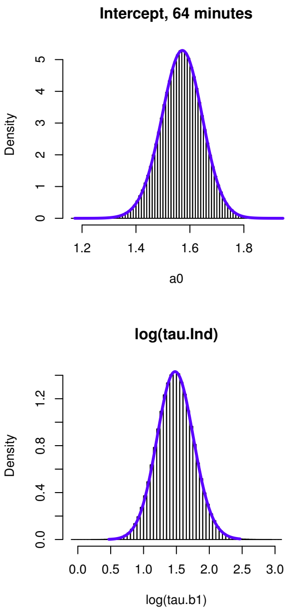

Running time of INLA < 0.5 seconds



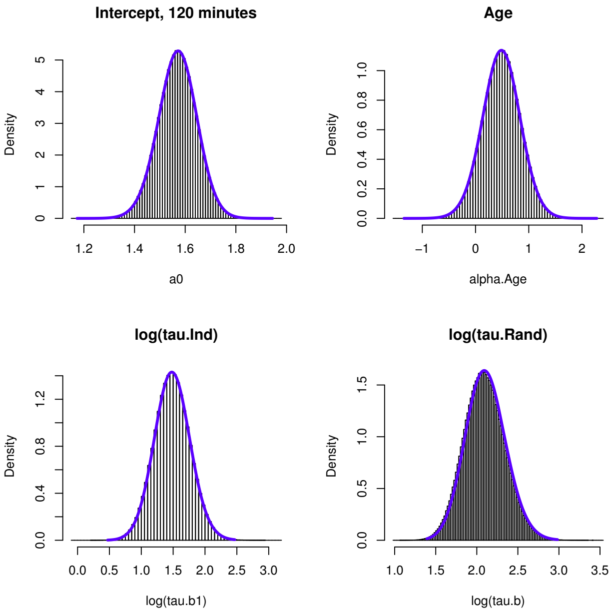

Running time of INLA $< 0.5$ seconds




Running time of INLA $< 0.5$ seconds




Running time of INLA < 0.5 seconds




Running time of INLA $< 0.5$ seconds

# Control statements

`control.xxx` statements control computations

- `control.fixed`

# 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 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`

# 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 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`

# 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?

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

# Discussion

INLA is a promising alternative to MCMC for the class of latent Gaussian models. It avoids time-conusming sampling and approximates the quantities of interest directly.