Good news

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

Getting INLA

- The web page www.r-inla.org contains source-code, worked-through examples, reports and instructions for installing the package. An INLA tutorial is in preparation.

Getting INLA

- The web page www.r-inla.org contains source-code, worked-through examples, reports and instructions for installing the package. An INLA tutorial is in preparation.
- The R-package INLA works on Linux, Windows and Mac and can be installed within R by

```
install.packages("INLA",
  repos="https://inla.r-inla-download.org/R/testing")
```

Later, it can be upgraded with

```
update.packages(oldPkgs="INLA",
  repos="https://inla.r-inla-download.org/R/testing")
```

Which INLA version do you have?

```
inla.version()
```

```
##
##
##   INLA version ............: 19.05.19
##   INLA date ...............: Sun 19 May 2019 06:19:09 PM +03
##   INLA hgid ...............: Version_19.05.19
##   INLA-program hgid .......: Version_19.05.19
##   Maintainers .............: Havard Rue <hrue@r-inla.org>
##                            : Finn Lindgren <finn.lindgren@gmail.com>
##                            : Daniel Simpson <dp.simpson@gmail.com>
##                            : Elias Teixeira Krainski <elias.krainski@math.ntn
##                            : Haakon Bakka <bakka@r-inla.org>
##                            : Andrea Riebler <andrea.riebler@math.ntnu.no>
##                            : Geir-Arne Fuglstad <fulgstad@math.ntnu.no>
##   Main web-page ...........: www.r-inla.org
##   Download-page ...........: inla.r-inla-download.org
##   Email support ...........: help@r-inla.org
##                            : r-inla-discussion-group@googlegroups.com
##   Source-code .............: bitbucket.org/hrue/r-inla
```

How to use INLA: Ski flying records

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
   ```

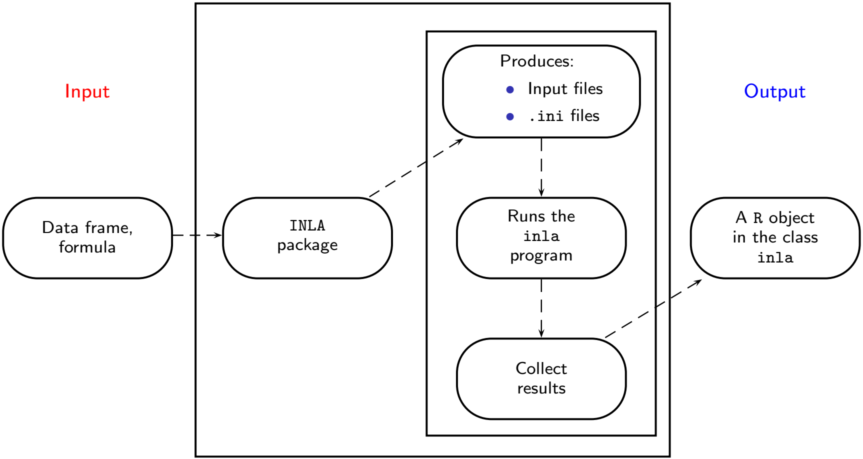
3. Call the inla-program

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

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

   ```
   summary(res)
   ```

The INLA package for R



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.

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.

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

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

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.
  - 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.

What comes out?

```
Call:
"inla(formula = formula, family = \"gaussian\", data = data)"

Time used:
 Pre-processing    Running inla Post-processing           Total
         0.0581          0.0161          0.0181          0.0924

Fixed effects:
                mean     sd 0.025quant 0.5quant 0.975quant     mode kld
(Intercept) 137.0288 1.3929   134.2798 137.0288   139.7741 137.0288   0
x             2.1259 0.0526     2.0221   2.1259     2.2295   2.1259   0
...
```

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 a `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"
- "T"
- "poisson"
- "nbinomial"
- "binomial"
- "exponential"
- "weibull"
- "coxph"
- See list with

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

Posterior inference

Main functions:

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

Example: Simple linear regression

...such as our ski flying example.

Stage 1: Gaussian likelihood

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

Stage 2: Covariates are connected to likelihood by

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

Stage 3: σ_{o}^2: variance of observation noise

Example: Simple linear regression

```r
# 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")
```

```r
# Get summary
summary(result)
```

```
##
## Call:
##    c("inla(formula = formula, family = \"gaussian\", data =
##    data.frame(x = x, ", " y = y))")
## Time used:
##     Pre = 1.16, Running = 0.0491, Post = 0.346, Total = 1.55
## Fixed effects:
##              mean    sd 0.025quant 0.5quant 0.975quant  mode kld
## (Intercept) 0.966 0.024      0.920    0.966      1.013 0.966   0
## x           2.057 0.034      1.991    2.057      2.124 2.057   0
##
## Model hyperparameters:
##                                           mean    sd 0.025quant 0.5quant
## Precision for the Gaussian observations 131.57 18.61      97.63   130.69
##                                         0.975quant   mode
## Precision for the Gaussian observations     170.51 128.93
##
## Expected number of effective parameters(stdev): 2.08(0.012)
## Number of equivalent replicates : 48.10
##
## Marginal log-Likelihood:  85.30
```

Summary for the fixed effects

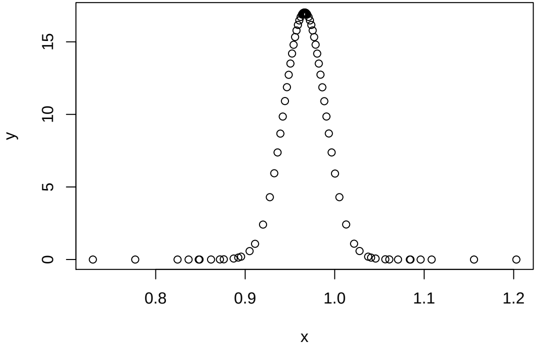
```
result$summary.fixed
```

```
##                  mean         sd 0.025quant 0.5quant 0.975quant      mode
## (Intercept) 0.9663907 0.02361297  0.9199339 0.966390   1.012809 0.9663906
## x           2.0574599 0.03361636  1.9913221 2.057459   2.123542 2.0574600
##                      kld
## (Intercept) 4.070162e-06
## x           4.070159e-06
```

Marginal posterior densities

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

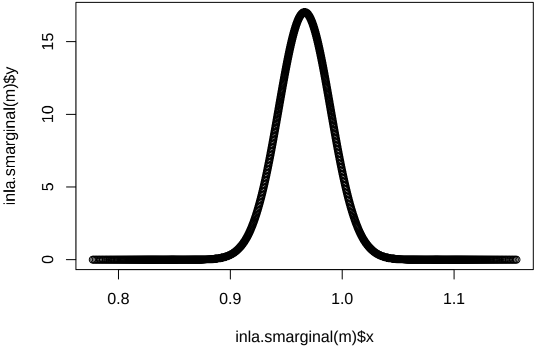
```
m = result$marginals.fixed[[1]]
par(mar=c(5,5,1,1))
plot(m)
```



Marginal posterior densities

The rough shape can be interpolated to higher resolution:

```
par(mar=c(5,5,1,1))
plot(inla.smarginal(m))
```



Marginal posterior densities

```r
# Extract quantiles
inla.qmarginal(0.05, m)
## [1] 0.9274911
# Distribution function
inla.pmarginal(0.975, m)
## [1] 0.6447482
# Density function
inla.dmarginal(1, m)
## [1] 6.05852
# Generate realizations
inla.rmarginal(4, m)
## [1] 0.9783395 0.9733956 1.0191159 0.9803112
```

Marginal posterior densities

```
# Calculate expected value of x and x^2
E = inla.emarginal(function(x) c(x,x^2), m)
E

## [1] 0.9663907 0.9344684

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

## [1] 0.02360998

# Compare to estimate
round(result$summary.fixed[,1:2], 3)

##              mean    sd
## (Intercept) 0.966 0.024
## x           2.057 0.034
```

Organisation of the returned inla-object

You find summary information (mean, sd, quantiles, . . .) 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"
```

For example:

```
result$summary.fixed

##                  mean         sd 0.025quant 0.5quant 0.975quant      mode          kld
## (Intercept) 0.9663907 0.02361297  0.9199339 0.966390   1.012809 0.9663906 4.070162e-06
## x           2.0574599 0.03361636  1.9913221 2.057459   2.123542 2.0574600 4.070159e-06
```

Organisation of the returned inla-object

You find summary information (mean, sd, quantiles, . . .) 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"
```

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

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

```
##              x            y
## [1,] 0.7297841 1.843610e-17
## [2,] 0.7771054 2.980953e-11
## [3,] 0.8244267 1.666505e-06
## [4,] 0.8367475 1.913358e-05
## [5,] 0.8480874 1.563628e-04
## [6,] 0.8489800 1.833776e-04
```

Further general information

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

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

```
# data used
head(result$.args$data)
```

```
##           x        y
## 1 0.9685481 2.968482
## 2 0.1186742 1.088834
## 3 0.6830238 2.526741
## 4 0.5525618 1.952179
## 5 0.9405515 2.893238
## 6 0.8239475 2.611642
```

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

Get estimates for variance not precision

Goal: Posterior mean and standard deviation of $\sigma_0^2 = \frac{1}{\tau_0}$.

```
names(result$marginals.hyperpar)

## [1] "Precision for the Gaussian observations"

# get the marginal for the precision
tau0 = result$marginals.hyperpar[[1]]

# 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
## 0.007755933 0.001120030
```

Get also the posterior marginal

```
# transform the postrior marginal
sigma2 = inla.tmarginal(function(x){1/x}, tau0)
head(sigma2)
```

```
##                x         y
## [1,] 0.004959493  2.913510
## [2,] 0.005081670  5.339159
## [3,] 0.005158685  7.540104
## [4,] 0.005216253  9.621419
## [5,] 0.005262674 11.625255
## [6,] 0.005301783 13.570303
```

```
# from the marginal also 'z'ummary information can be derived
inla.zmarginal(sigma2)
```

```
## Mean              0.00775473
## Stdev             0.00111203
## Quantile  0.025 0.00586973
## Quantile  0.25  0.00696248
## Quantile  0.5   0.00764987
## Quantile  0.75  0.00843062
## Quantile  0.975 0.010228
```

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:

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

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

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

Stage 3:

- σ_{o}^2 : variance of observation noise
- ρ : dependence in AR(1) process
- σ^2 : variance of the innovations in AR(1) process

Example: Add random effect

```
# Generate AR(1) sequence
set.seed(580258)
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 = 0.2)

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

```
summary(result)
```

```
##
## Call:
##    c("inla(formula = formula, family = \"gaussian\", data = data.frame(x =
##    x, ", " y = y, t = t))")
## Time used:
##     Pre = 1.41, Running = 0.143, Post = 1.41, Total = 2.96
## Fixed effects:
##              mean    sd 0.025quant 0.5quant 0.975quant  mode kld
## (Intercept) 1.002 0.062      0.879    1.003      1.124 1.003   0
## x           2.007 0.083      1.843    2.007      2.171 2.007   0
##
## Random effects:
##   Name   Model
##     t AR1 model
##
## Model hyperparameters:
##                                           mean     sd 0.025quant 0.5quant
## Precision for the Gaussian observations 24.399  5.476     15.416   23.798
## Precision for t                         67.663 46.210     16.145   56.006
## Rho for t                                0.721  0.167      0.287    0.762
##                                         0.975quant   mode
## Precision for the Gaussian observations     36.829 22.645
## Precision for t                            187.883 37.890
## Rho for t                                    0.929  0.834
##
## Expected number of effective parameters(stdev): 22.25(12.21)
## Number of equivalent replicates : 4.50
##
## Marginal log-Likelihood:  -21.02
```

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`.

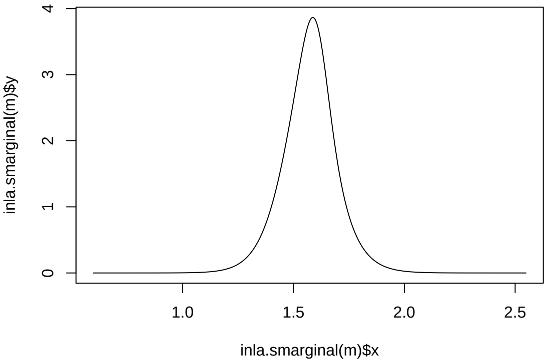
```
# Add new location
x = c(x, 0.3)
t = c(t, 101)
y = c(y, NA)

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

Prediction

```r
m = result.pred$marginals.linear.predictor[[101]]
round(result.pred$summary.linear.predictor[101,], 3)
```

```
##                mean    sd 0.025quant 0.5quant 0.975quant  mode kld
## predictor.101 1.573 0.122      1.321    1.576      1.819 1.587   0
```

```r
par(mar=c(5,5,0.5,0.5))
plot(inla.smarginal(m), type="l")
```



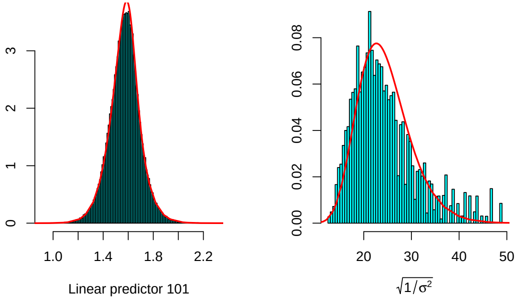
Prediction

Caution: This is not yet the predictive distribution, as the observation noise is missing.
One way to add is by sampling from the posterior distribution.

```
n = 100000
x = inla.posterior.sample(n, result.pred)

x101 = rep(NA, n)
obs.noise= rep(NA, n)
for(i in 1:n){
   x101[i] = x[[i]]$latent["Predictor.101",]
   obs.noise[i] = x[[i]]$hyperpar[1]
}
```

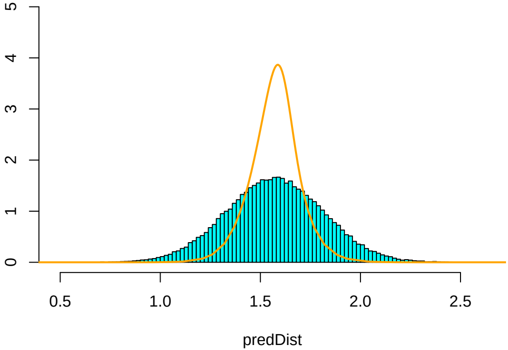
Illustration of samples

```
library(MASS)
par(mfrow=c(1,2), mar=c(5,5,1,1))
truehist(x101, xlab="Linear predictor 101")
lines(result.pred$marginals.linear.predictor[[101]], col=2, lwd=2)
truehist(obs.noise, xlab=expression(sqrt(1/sigma^2)))
lines(result.pred$marginals.hyperpar[[1]], col=2, lwd=2)
```



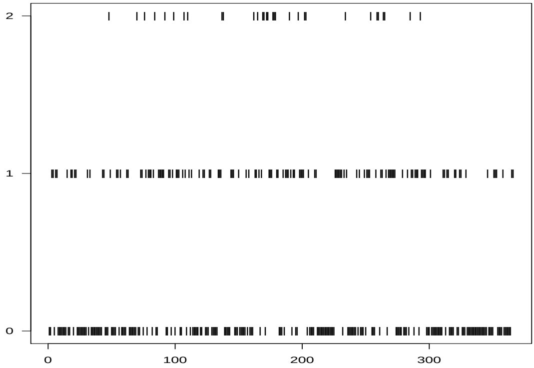
Posterior predictive distribution

```
predDist = rnorm(n, mean=x101, sd=sqrt(1/obs.noise))
par(mar=c(5,5,0.5,0.5))
truehist(predDist, ylim=c(0,5))
lines(result.pred$marginals.linear.predictor[[101]], col="orange", lwd=2)
```



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.



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,4)
```

```
##   y n time
## 1 0 2    1
## 2 0 2    2
## 3 1 2    3
## 4 1 2    4
```

```
Tokyo[60,]
```

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

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

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

Stage 3: τ: Scale parameter in CRW2 with prior

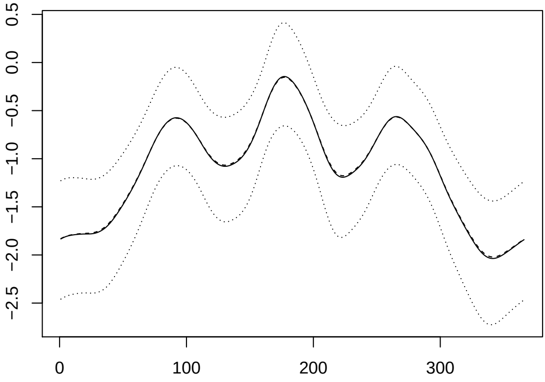
$$\pi(\tau) \sim \text{Gamma}(1, 5 \cdot 10^{-5})$$

Computations

```
# 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)
```

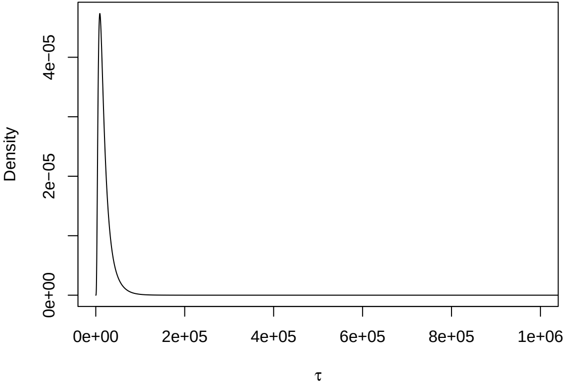
Marginal posterior of CRW2

```
par(mar=c(5,5,1,0.5))
toplot = c("mean", "0.025quant", "0.5quant", "0.975quant")
matplot(result$summary.random$t[, toplot],
        lty =c(1,3,2,3), type="l", col=1, ylab="")
```



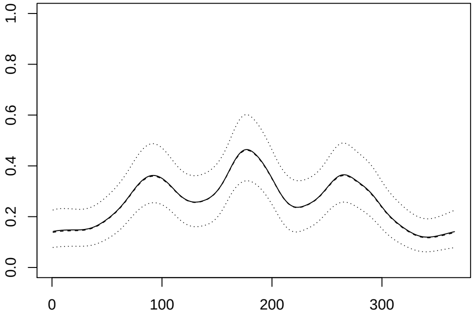
Marginal posterior of scale parameter

```
par(mar=c(5,5,1.5,0.5))
plot(inla.smarginal(result$marginals.hyperpar[[1]]), xlim=c(0, 10^6),
     xlab=expression(tau), ylab="Density", type="l")
```



Transform to probability

```
result = inla(formula, family = "binomial",
              Ntrials = n, data = Tokyo,
              control.predictor = list(compute=TRUE))
par(mar=c(5,5,1.5,0.5))
matplot(result$summary.fitted.values[, toplot],
        lty =c(1,3,2,3), type="l", col=1, ylim=c(0,1), ylab="")
```

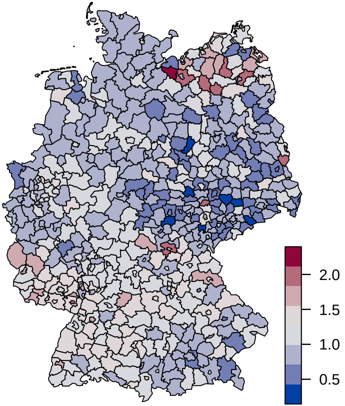


Example: Disease mapping

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

Information available:

- y_i: The count at location i.
- E_i: An offset; expected number of cases in district i.
- $\boldsymbol{s}_i$: spatial location i (here, district).



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.

A spatially structured effect

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

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

where R is called structure matrix and defined as

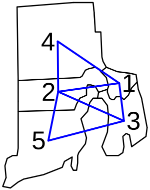
$$R_{ij} = \begin{cases} n_i & i = j \\ -1 & i \sim j \\ 0 & \text{otherwise.} \end{cases}$$

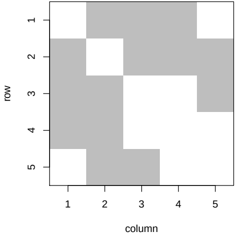
Here, $i \sim j$ denotes that i and j are neighbouring regions.

What does this mean?

Example: Five counties of the US state Rhode Island

The structure matrix **R** defines the neighborhood structure.





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

Structure matrix

With increasing number of regions **R** will be sparse, which allows to do many computations very efficient.

INLA code

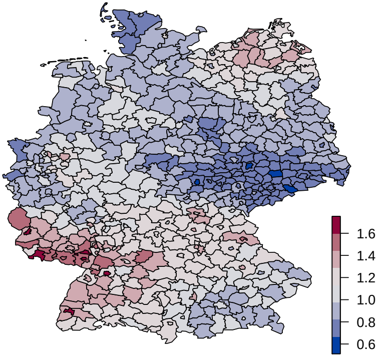
```
library(spam)
data(Oral)
# load the file including neighbourhood information
g = system.file("demodata/germany.graph", package="INLA")

head(Oral, 4)
```

```
##    Y        E       SMR
## 1 18 16.35051 1.1008834
## 2 62 45.90600 1.3505861
## 3 44 44.66248 0.9851669
## 4 12 16.32308 0.7351552
```

```
# we need two region indices
Oral = cbind(Oral, region = 1:544, region.unstruc= 1:544)

formula = Y ~ f(region, model="besag", graph=g) +
                            f(region.unstruc, model="iid")
result = inla(formula, family="poisson", E=E, data=Oral)
```

Median of u on exp-scale

```
library(fields)
library(colorspace)
col <- diverge_hcl(8)
par(mar=c(1,1,.5,1))
map.landkreis(exp(result$summary.random$region$"0.5quant"),col=col)
```



Other choices for f-terms

```
names(inla.models()$latent)
##  [1] "linear"       "iid"          "mec"
##  [4] "meb"          "rgeneric"     "rw1"
##  [7] "rw2"          "crw2"         "seasonal"
## [10] "besag"        "besag2"       "bym"
## [13] "bym2"         "besagproper"  "besagproper2"
## [16] "fgn"          "fgn2"         "ar1"
## [19] "ar1c"         "ar"           "ou"
## [22] "intslope"     "generic"      "generic0"
## [25] "generic1"     "generic2"     "generic3"
## [28] "spde"         "spde2"        "spde3"
## [31] "iid1d"        "iid2d"        "iid3d"
## [34] "iid4d"        "iid5d"        "2diid"
## [37] "z"            "rw2d"         "rw2diid"
## [40] "slm"          "matern2d"     "dmatern"
## [43] "copy"         "clinear"      "sigm"
## [46] "revsigm"      "log1exp"      "logdist"
```

Add weight to components of a random effect

```
formula = y ~ ... + f(idx , weight, model = <MODEL>, ...)
```

extends the usual

$$\eta_i = \ldots + f_{\mathrm{idx}_i}$$

to

$$\eta_i = \ldots + \mathrm{weight}_{\mathrm{idx}_i} f_{\mathrm{idx}_i}$$

Changing the prior: Internal scale

- Hyperparameters are represented internally with more well-behaved transformations, e.g. correlation ρ and precision τ 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 the mode-search 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) + ...
```

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

We come back to priors in the last lecture block.

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.

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)))
```

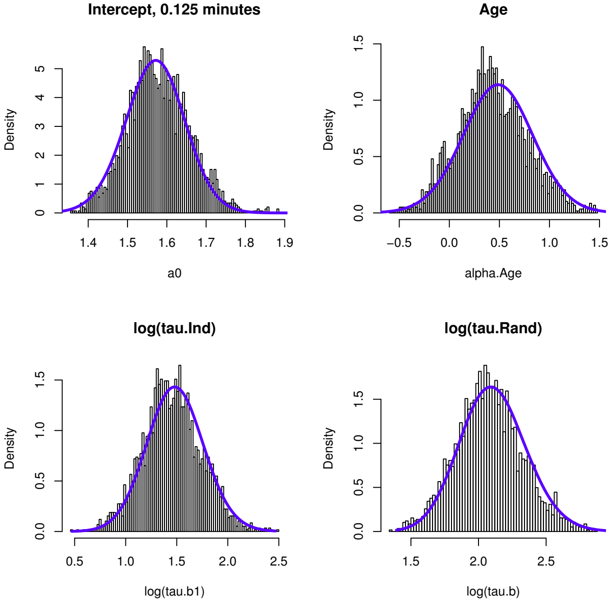
```
> result = inla(formula, family="poisson", data = Epil,
     control.fixed = list(prec.intercept = 0.001,
                          prec = 0.001))
```

Comparing results with MCMC

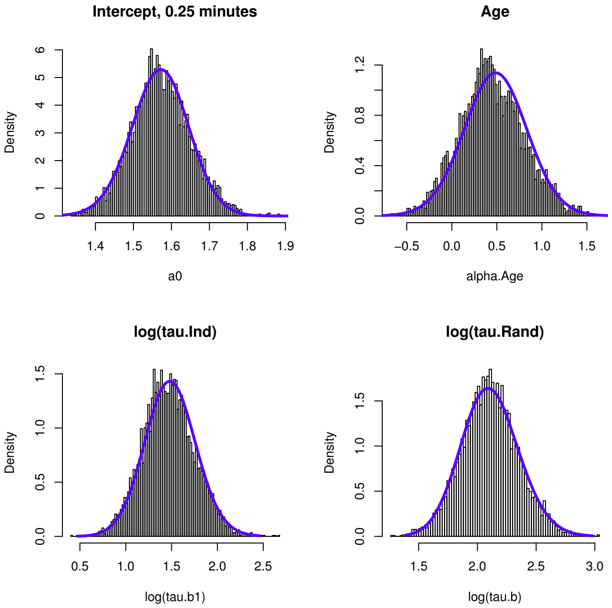
- When comparing the results of `R-INLA` with MCMC, it is important to use the same model. That means, same data, same priors, same constraints on parameters, intercept included or not, ...

Comparing results with MCMC

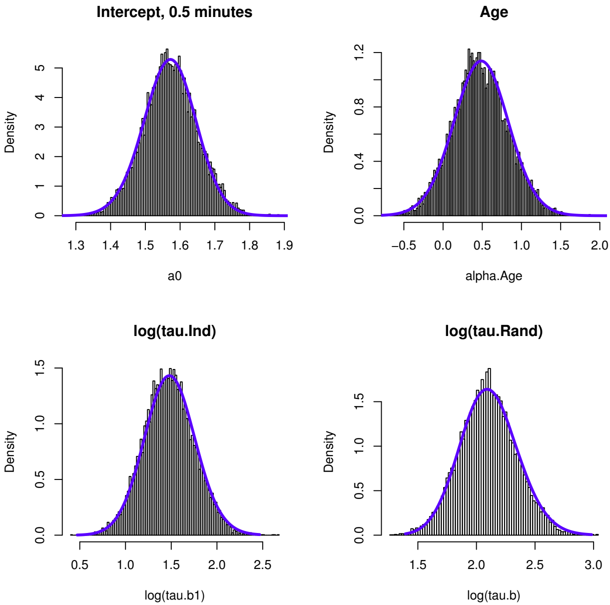
- When comparing the results of `R-INLA` with MCMC, it is important to use the same model. That means, same data, same priors, same constraints on parameters, intercept included or not, ...
- Here we have compared the results with those obtained using `JAGS` via the `rjags` package



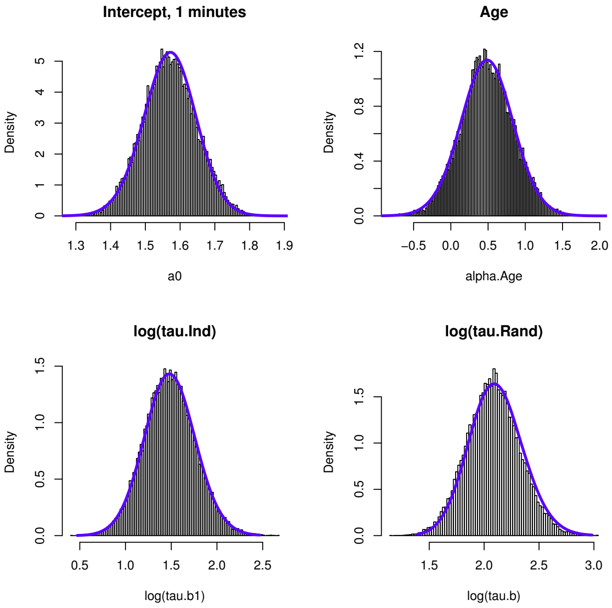
Running time of INLA < 0.5 seconds



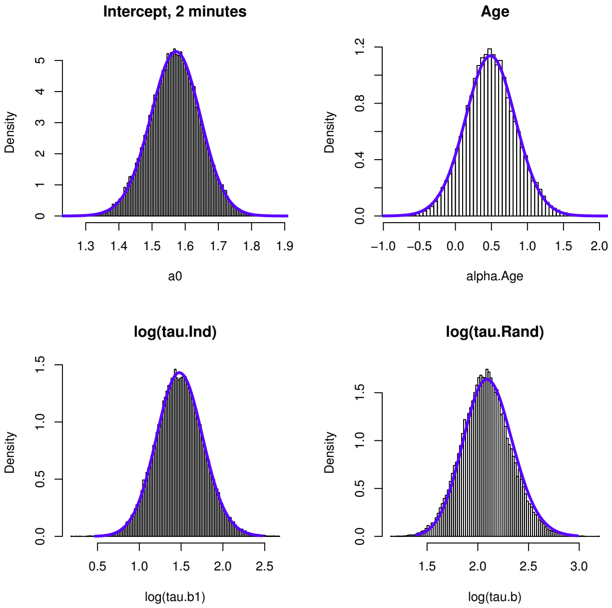
Running time of INLA < 0.5 seconds



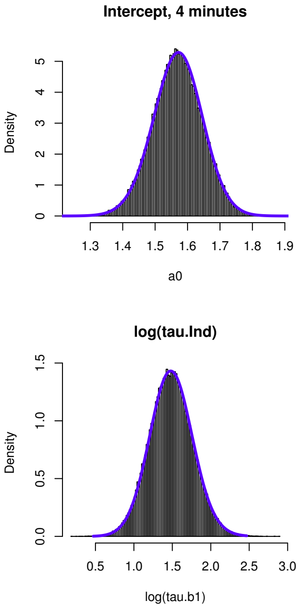
Running time of INLA < 0.5 seconds



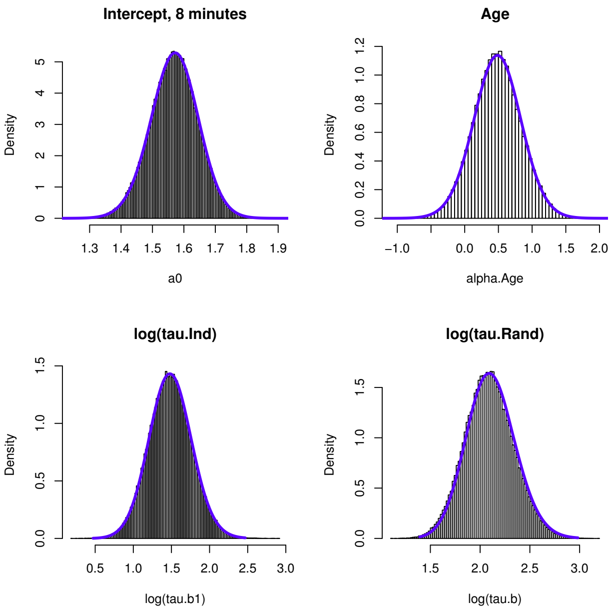
Running time of INLA < 0.5 seconds



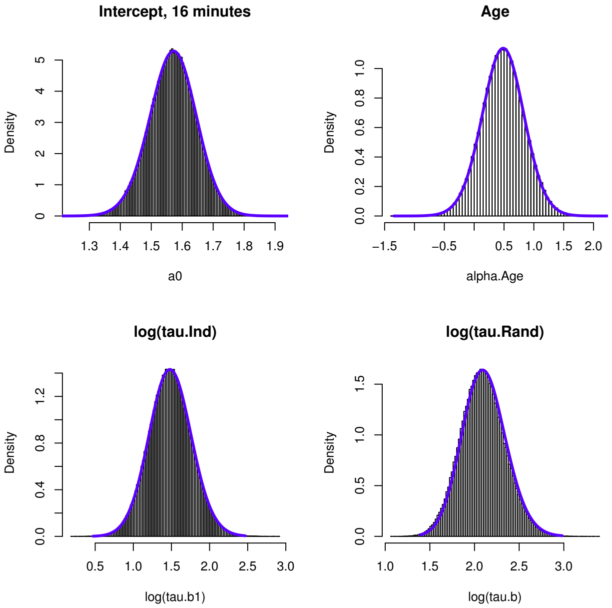
Running time of INLA < 0.5 seconds



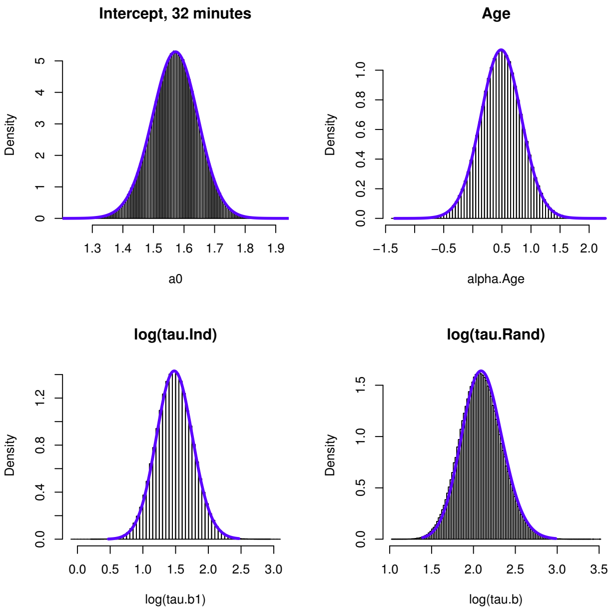
Running time of INLA < 0.5 seconds



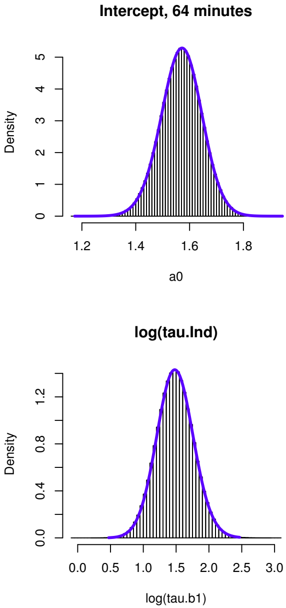
Running time of INLA < 0.5 seconds



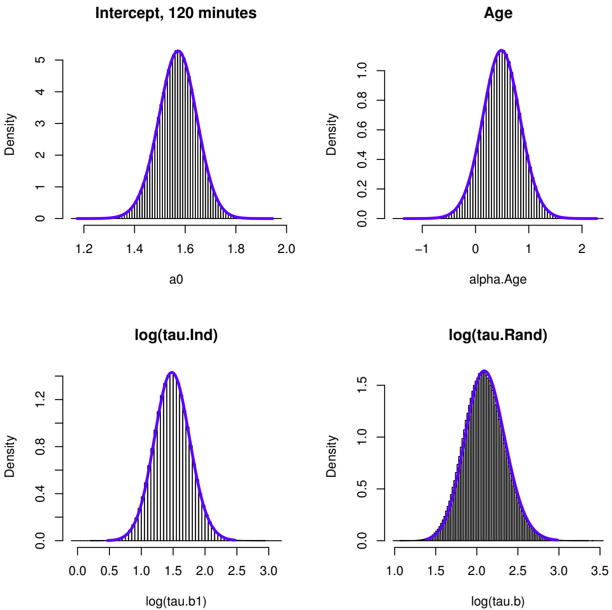
Running time of INLA < 0.5 seconds



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`, `mlik`, `cpo`: Compute measures of fit?

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 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`

Control statements

`control.xxx` statements control computations

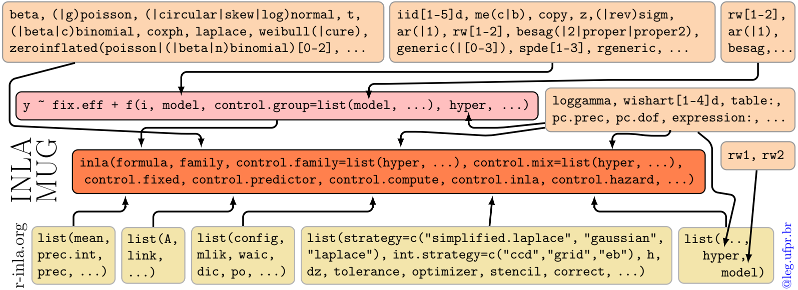
- `control.fixed`
  - `prec`: Default precision for all fixed effects except the intercept. `prec.intercept`: Precision for intercept (Default: 0.0)
- `control.predictor`
  - `compute`: Compute posterior marginals of linear predictors
- `control.compute`
  - `dic`, `mlik`, `cpo`: Compute measures of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

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.

Main function in R-INLA



Thank you for your attention!

If you have any doubts or questions, please write us:
elias@r-inla.org
help@r-inla.org