Introduction to the **INLA** package

Elias T. Krainski

Jul–2017, Lavras/MG
62^a RBras & 17^o SEAGRO

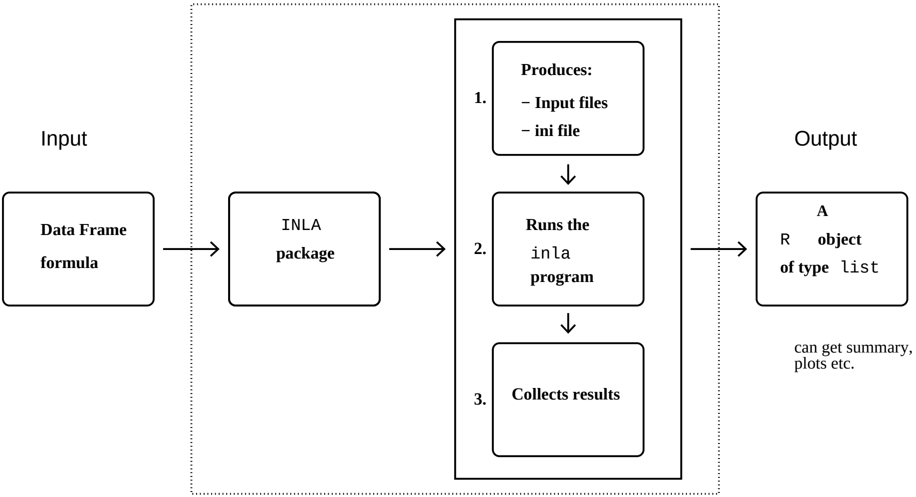
Outline

1. Overview
2. Tokyo example
3. GAG urine data
4. Close look at GAG urine example
5. Details

What is **INLA**?

- The algorithm name:
  Integrated Nested Laplace Aproximations - **INLA**
- **INLA**:
  - The **R**-package that implements **INLA**
  - **INLA** is a collection of R-code and C-code
    - C-code: fast computations for **GMRF**
    - R-code: to have it available for R-users
  - available at `www.r-inla.org`
- all the source code is public available at
  `https://bitbucket.org/hrue/r-inla`

INLA sheme



Input

Data Frame

formula

`INLA` package

1. Produces:
   - Input files
   - ini file

2. Runs the `inla` program

3. Collects results

Output

A `R` object of type `list`

can get summary, plots etc.

How to install **INLA**?

new way (from 3rd January 2015)

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

or

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

INLA is in continuous development:

```
> inla.upgrade()
```

or

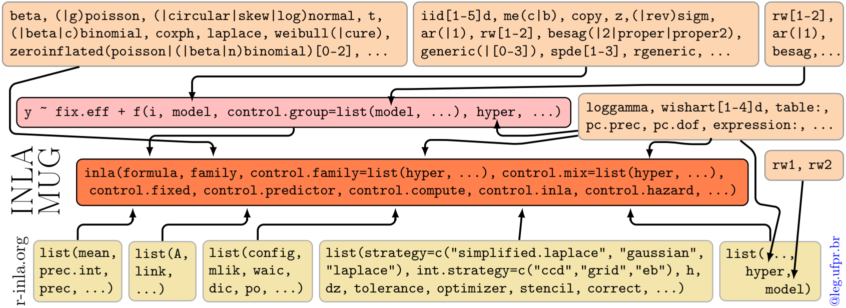
```
> inla.upgrade(testing=TRUE)
```

Data analisys steps using **INLA**

Five steps to fit an model

1. data organization
   response, covariates and index sets (for random effects)
2. Define the priors
3. define the `formula`
   - notation (similar to `lm` and `glm` functions
   - `f(.)` term to define random effects
4. call to the `inla`-program
5. extraction of posterior information

INLA picture



```
beta, (|g)poisson, (|circular|skew|log)normal, t,
(|beta|c)binomial, coxph, laplace, weibull(|cure),
zeroinflated(poisson|(|beta|n)binomial)[0-2], ...
```

```
iid[1-5]d, me(c|b), copy, z,(|rev)sigm,
ar(|1), rw[1-2], besag(|2|proper|proper2),
generic(|[0-3]), spde[1-3], rgeneric, ...
```

```
rw[1-2],
ar(|1),
besag,...
```

```
y ~ fix.eff + f(i, model, control.group=list(model, ...), hyper, ...)
```

```
loggamma, wishart[1-4]d, table:,
pc.prec, pc.dof, expression:, ...
```

INLA
MUG

```
inla(formula, family, control.family=list(hyper, ...), control.mix=list(hyper, ...),
 control.fixed, control.predictor, control.compute, control.inla, control.hazard, ...)
```

```
rw1, rw2
```

r-inla.org

```
list(mean,
prec.int,
prec, ...)
```

```
list(A,
link,
...)
```

```
list(config,
mlik, waic,
dic, po, ...)
```

```
list(strategy=c("simplified.laplace", "gaussian",
"laplace"), int.strategy=c("ccd","grid","eb"), h,
dz, tolerance, optimizer, stencil, correct, ...)
```

```
list(...,
  hyper,
    model)
```

@leg.ufpr.br

Outline

1. Overview
2. Tokyo example
3. GAG urine data
4. Close look at GAG urine example
5. Details

Tokyo example

- Observation model: $p(\boldsymbol{y}|\boldsymbol{x})$ likelihood

$$y_i \sim \text{Binomial}(n_i, p_i)$$

$$p_i = \frac{1}{1 + \exp(-x_i)}$$

$$\pi(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{366} \pi(y_i|x_i)$$

- $\pi(\boldsymbol{x}|\theta)$
  1. first order random walk

$$x_{a_i} - x_{a_{i-\mathbf{1}}} \sim N(0, \sigma_1^2)$$

  2. second order random walk (smoother)

$$x_{a_i} - (2x_{a_{i-\mathbf{1}}} - x_{a_{i-\mathbf{2}}}) \sim N(0, \sigma_2^2)$$

- $\pi(\sigma^2) = \pi(\theta)$: Penalized Complexity (PC) prior

PC prior idea

- define a base/simplest model
- $p(x|\psi)$: ψ is a flexibility parameter
- $\psi = 0$ is the base model case

PC prior idea

- define a base/simplest model
- $p(x|\psi)$: ψ is a flexibility parameter
- $\psi = 0$ is the base model case
- **Idea**: a prior that has mode at the base model
- posterior concentrates on $\psi > 0$ only if the data requires a more complex model

R-code for Tokyo example rw1

The dataset

```
data(Tokyo)
Tokyo[1:3,]

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

Define the formula (include hyperprior)

```
pcprec <- list(theta=list(
  prior='pc.prec', param=c(2, 0.05)))
formula1 <- y ~ 0 +
 f(time, model='rw1',
  cyclic=TRUE, constr=FALSE,
   scale.model=TRUE, hyper=pcprec)
```

```
tokyo1 <- inla(formula1, data=Tokyo, Ntrials=n,
 family='binomial', control.predictor=list(compute=TRUE),
  control.compute=list(dic=TRUE, cpo=TRUE))
```

R-code: Tokyo example rw2

Define the formula

```
formula2 = y ~ 0 +
 f(time, model='rw2', cyclic=TRUE, constr=FALSE,
   scale.model=TRUE, hyper=pcprec)
```

Fit the model

```
tokyo2 <- inla(formula2, data=Tokyo, Ntrials=n,
 family='binomial', control.predictor=list(compute=TRUE),
  control.compute=list(dic=TRUE, cpo=TRUE))
```

Tokyo `rw1` summary

```
summary(tokyo1)
```

```
##
## Call:
## c("inla(formula = formula1, family = \"binomial\", data = Tokyo, Ntrials = n, ",  "    control.compute = lis
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          0.3272          0.8145          0.0702          1.2119
##
## The model has no fixed effects
##
## Random effects:
## Name   Model
##  time   RW1 model
##
## Model hyperparameters:
##                      mean     sd 0.025quant 0.5quant 0.975quant   mode
## Precision for time 1.049 0.7564     0.2807   0.8384      3.113 0.5629
##
## Expected number of effective parameters(std dev): 22.08(6.605)
## Number of equivalent replicates : 16.57
##
## Deviance Information Criterion (DIC) ...: 616.70
## Effective number of parameters .........: 23.27
##
## Marginal log-Likelihood:  -944.22
## CPO and PIT are computed
##
## Posterior marginals for linear predictor and fitted values computed
```

Some Tokyo results `rw1`

Random effects

```
round(tokyo1$summary.random$time[1:3,], 4)
```

```
##   ID    mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## 1  1 -1.6366 0.4452    -2.5249  -1.6345    -0.7625 -1.6317   0
## 2  2 -1.6144 0.4450    -2.4992  -1.6134    -0.7381 -1.6127   0
## 3  3 -1.5759 0.4468    -2.4573  -1.5774    -0.6880 -1.5814   0
```

Fitted values

```
round(tokyo1$summary.fitted.values[1:3,], 4)
```

```
##                          mean     sd 0.025quant 0.5quant 0.975quant   mode
## fitted.Predictor.001 0.1717 0.0626     0.0742   0.1632     0.3182 0.1473
## fitted.Predictor.002 0.1748 0.0635     0.0760   0.1661     0.3235 0.1499
## fitted.Predictor.003 0.1804 0.0655     0.0790   0.1712     0.3344 0.1541
```

Compare

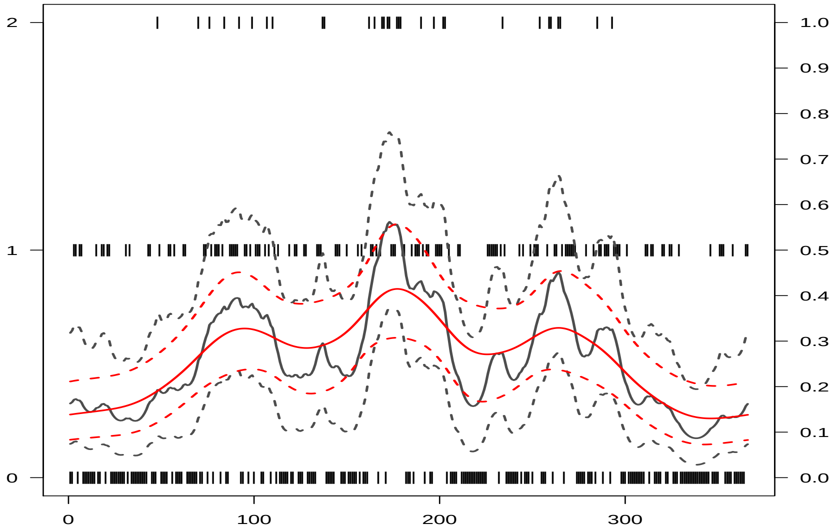
Precision

```
round(rbind(rw1=tokyo1$summary.hy[, 1:6],
            rw2=tokyo2$summary.hy[, 1:6]), 4)

##       mean     sd 0.025quant 0.5quant 0.975quant   mode
## rw1 1.0492 0.7564     0.2807   0.8384     3.1128 0.5629
## rw2 1.8688 3.7269     0.1095   0.6424    11.8073 0.2035
```

DIC, -LCPO

```
round(rbind(DIC=c(rw1=tokyo1$dic$dic, rw2=tokyo2$dic$dic),
 LCPO=c(sum(-log(tokyo1$cpo$cpo)), sum(-log(tokyo2$cpo$cpo)))),

##        rw1    rw2
## DIC  616.7 635.29
## LCPO 307.9 317.68
```

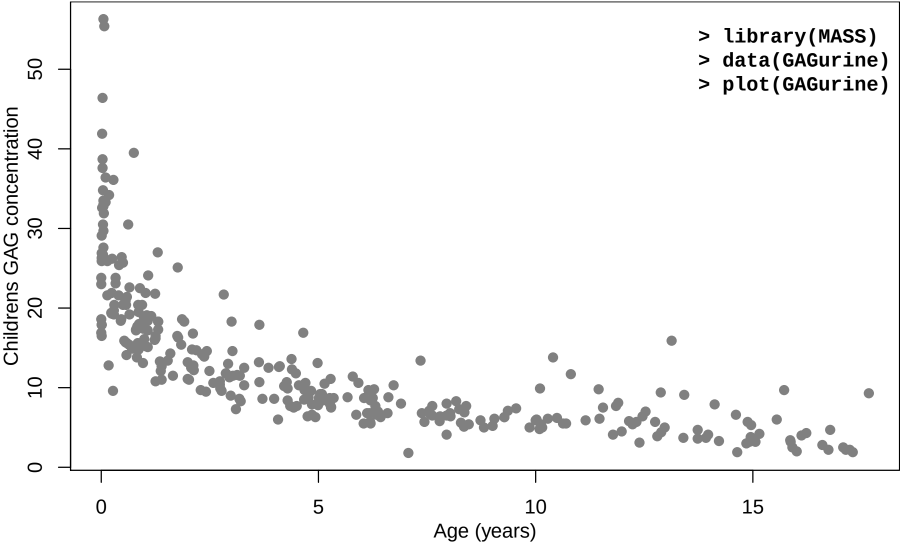
Tokyo results



Outline

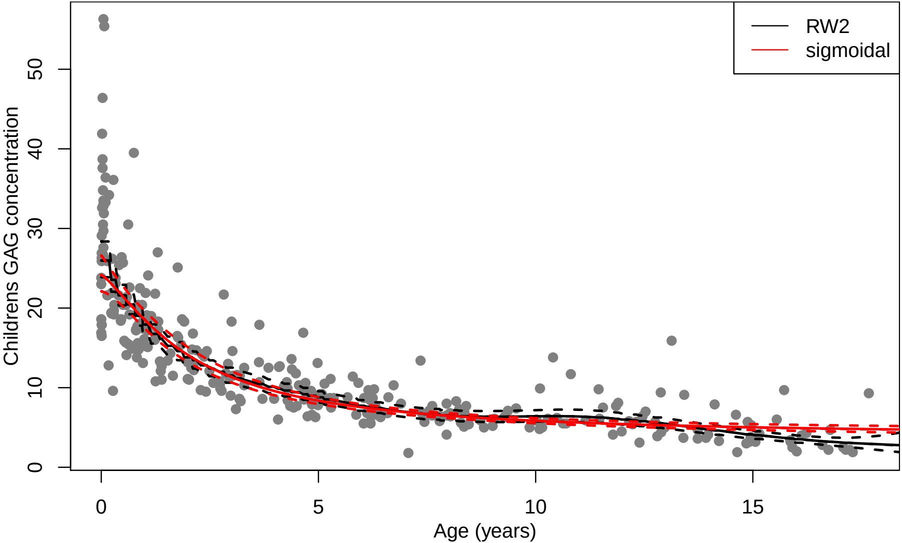
1. Overview
2. Tokyo example
3. GAG urine data
4. Close look at GAG urine example
5. Details

GAG urine



```
> library(MASS)
> data(GAGurine)
> plot(GAGurine)
```

GAG urine aim



GAG urine problem

- Problem: build a chart for GAG urine

$$E(GAG|age) = \mu(age)$$

- $\mu(age)$ must be positive
  → Easy: model for the log of GAG
- How to relat *GAG* and age?
  → People usually uses parametric non-linear models
- but we have INLA...
  → Lets define a semi-parametric one

GAG urine model

- linear model for y = log (GAG)

$$y = \beta_0 + f(\text{age}) + \text{error}$$

  error $\sim N(0, \sigma_e^2)$

- non linear effect from age

$$f(\text{age}) = x_{\text{age}}$$

  where x for some age is similar to x for neighbour ages

- define some grid of ages: $a_1, a_2, ..., a_k$
- first order random walk

$$x_{a_i} - x_{a_{i-1}} \sim N(0, \sigma^2)$$

- second order random walk (smoother)

$$x_{a_i} - (2x_{a_{i-1}} - x_{a_{i-2}}) \sim N(0, \sigma_x^2)$$

GAG urine analysis

GAG urine for 314 children aged from 0 to 17 years

```
require(MASS)
data(GAGurine)
GAGurine[c(1, 314), ]

##        Age  GAG
## 1     0.00 23.0
## 314 17.67  9.3
```

Prepare the data

```
GAGurine$y =
    log(GAGurine$GAG)
```

Define the formula

```
gag.form = y ~
 f(inla.group(Age), model='rw2',
   scale.model=TRUE, hyper=pcprec)
```

Fit the model

```
gag.rw2 <- inla(gag.form,
                data=GAGurine)
```

Outline

1. Overview
2. Tokyo example
3. GAG urine data
4. Close look at GAG urine example
   - GAG reference limits
5. Details

GAG urine detailed

```
## prepare the data to predict new observations
gag.dat <- list(lgag=c(log(GAGurine$GAG), NA, NA),
                age=c(GAGurine$Age, 18, 19)) ## extrapolate !!
## two models for the latent effect
gag.rw2.form <- lgag ~ f(inla.group(age, 100), model='rw2',
  scale.model=TRUE, hyper=list(theta=list(prior='pc.prec', para
gag.sigm.form <- lgag ~ f(age, model='sigm')
## ask for more results
gag.rw2 <- inla(gag.rw2.form, data=gag.dat,
 control.predictor=list(compute=TRUE), ### to predict
 control.compute=list(config=TRUE, ## store configurations
   dic=TRUE, cpo=TRUE, po=TRUE), ## to compare
 control.inla=list(strategy='laplace')) ### best strategy
```

GAG model results

```
c(dic1=gag.rw2$dic$dic, lcpo1=-sum(log(gag.rw2$cpo$cpo), na.rm=TRUE),
  dic2=gag.sigm$dic$dic, lcpo2=-sum(log(gag.sigm$cpo$cpo), na.rm=TRUE))
```

```
##      dic1     lcpo1      dic2     lcpo2
## 136.83245  71.59637 175.26973  88.44898
```

```
round(gag.rw2$summary.hy[, 1:2], 4)
```

```
##                                            mean      sd
## Precision for the Gaussian observations 11.4984  0.9310
## Precision for inla.group(age, 100)      15.2572 12.4783
```

```
round(gag.sigm$summary.hy[, 1:2], 4)
```

```
##                                            mean     sd
## Precision for the Gaussian observations 10.0989 0.8266
## SIGM beta for age                       -1.8636 0.1119
## SIGM halflife for age                    4.0201 0.4520
## SIGM shape for age                       1.2921 0.1229
```

Prediction-extrapolation

```
cbind(true=c(GAGurine$GAG[314], NA, NA), # with 314
      rw2=exp(gag.rw2$summary.fitted.val[314:316, 1]),
      sigm=exp(gag.sigm$summary.fitted.val[314:316, 1]))

##      true      rw2     sigm
## [1,]  9.3 2.954101 4.803916
## [2,]   NA 2.877849 4.780320
## [3,]   NA 2.659282 4.714983
```

Error in the linear predictor scale

```
rbind(rw2=gag.rw2$summary.fitted.val[314:316, 2],
      sigm=gag.sigm$summary.fitted.val[314:316, 2])

##            [,1]       [,2]       [,3]
## rw2  0.13067358 0.16259568 0.28892200
## sigm 0.04275026 0.04332279 0.04499502
```

The variance

- compute the distribution for $\sigma|y$ (std.dev.)

```
tau.marginal <- gag.rw2$marginals.hyperpar[[1]]
s.marginal <- inla.tmarginal(function(x) sqrt(1/x), tau.marginal)
```

- summarize a marginal

```
inla.zmarginal(s.marginal)

## Mean                0.295627
## Stdev               0.0119119
## Quantile  0.025 0.273161
## Quantile  0.25  0.287379
## Quantile  0.5   0.295262
## Quantile  0.75  0.303493
## Quantile  0.975 0.31992
```

- other functions to work with marginals

```
apropos('marginal')

##  [1] "inla.dmarginal"   "inla.emarginal"   "inla.hpdmarginal"
##  [4] "inla.mmarginal"   "inla.pmarginal"   "inla.qmarginal"
##  [7] "inla.rmarginal"   "inla.smarginal"   "inla.tmarginal"
## [10] "inla.zmarginal"   "s.marginal"       "tau.marginal"
```

GAG limits: The problem

- Define limits (2.5%, 95.5%) for GAG for each age
- The statistical model is for the mean: $E(y)$
- Additionally we have the *error*: GAG = E(GAG) + error

GAG limits: The problem

- Define limits (2.5%, 95.5%) for GAG for each age
- The statistical model is for the mean: $E(y)$
- Additionally we have the *error*: GAG = E(GAG) + error
- Easy way solution:
  - sample from $E(y)$ and the error
  - compute the sum
  - define the interval

GAG limits: code

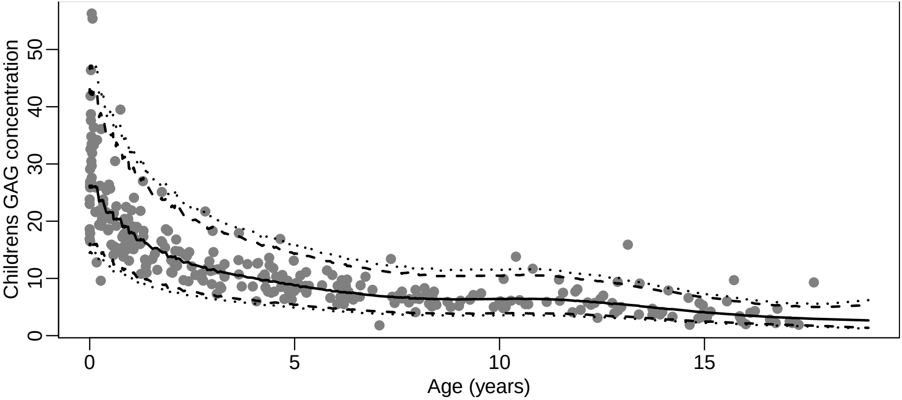
get samples from the posterior distribution

```
 samples <- inla.posterior.sample(n=10000, res=gag.rw2)
```

for each sample, sample the error and compute the sum

```
 psam <- sapply(samples, function(x)  {
     s.d <- sqrt(1/x$hyperpar[1])
     mu <- x$latent[1:316,1]
     err <- rnorm(316, 0, s.d)
     return(exp(mu+err)) ## in the GAG original scale
 })
q.sam <- apply(psam, 1, quantile,
    c(.025, 0.05, 0.5, 0.95, .975))
```

```r
par(mar=c(2.5, 2.5, 0, 0), mgp=c(1.5, 0.5, 0))
plot(gag.dat$age, c(GAGurine$GAG,NA,NA), pch=19,
     col=gray(.5), xlab='Age (years)',
     ylab='Childrens GAG concentration')
for (j in 1:5)
 lines(gag.dat$age, q.sam[j, ], lty=c(3,2,1,2,3)[j], lwd=2)
```



Visualize GAG limits

Outline

1. Overview
2. Tokyo example
3. GAG urine data
4. Close look at GAG urine example
5. **Details**

Results, post-processing and sampling

- Accessing results:
  - `summary(result)`
  - `plot(result)`
- post-processing
  - `result2 = inla.hyperpar(result)`
  - `result2 = inla.cpo(result)`
  - `result2 = inla.rerun(result)`
- sampling from posterior joint distribution
  - `samples = inla.hyperpar.sample(n=1000, result)`
  - `samples = inla.posterior.sample(n=1000, result)`

The interpretation of `NA`

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

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

```r
## define the prior
hyper = list(prec = list(prior = "loggamma",
                         param = c(1, 0.1),
                         initial = 4,
                         fixed = FALSE))
## insert it into the formula
formula = y ~ f(idx, model = "iid", hyper = hyper) + ...
```

Changing the prior: Default options

```r
## Default options can be seen with
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)}
```

Available priors

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

```
##  [1] "normal"                  "gaussian"
##  [3] "wishart1d"               "wishart2d"
##  [5] "wishart3d"               "wishart4d"
##  [7] "wishart5d"               "loggamma"
##  [9] "minuslogsqrtruncnormal" "logtnormal"
## [11] "logtgaussian"            "flat"
## [13] "logflat"                 "logiflat"
## [15] "mvnorm"                  "pc.ar"
## [17] "none"                    "invalid"
## [19] "betacorrelation"         "logitbeta"
## [21] "pc.prec"                 "pc.dof"
## [23] "pc.cor0"                 "pc.cor1"
## [25] "pc.fgnh"                 "pc.spde.GA"
## [27] "pc.matern"               "pc.range"
## [29] "pc.gamma"                "pc.mgamma"
## [31] "pc.gammacount"           "pc"
## [33] "ref.ar"                  "jeffreystdf"
## [35] "expression:"             "table:"
```

Available likelihoods

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


##  [1] "poisson"                         "qpoisson"
##  [3] "cenpoisson"                      "gpoisson"
##  [5] "binomial"                        "testbinomial1"
##  [7] "gamma"                           "gammacount"
##  [9] "qkumar"                          "qloglogistic"
## [11] "beta"                            "betabinomial"
## [13] "cbinomial"                       "nbinomial"
## [15] "simplex"                         "gaussian"
## [17] "normal"                          "circularnormal"
## [19] "wrappedcauchy"                   "iidgamma"
## [21] "iidlogitbeta"                    "loggammafrailty"
## [23] "logistic"                        "skewnormal"
## [25] "sn"                              "sn2"
## [27] "gev"                             "lognormal"
## [29] "lognormalsurv"                   "exponential"
## [31] "exponentialsurv"                 "coxph"
## [33] "weibull"                         "weibullsurv"
## [35] "loglogistic"                     "weibullcure"
## [37] "stochvol"                        "stochvolt"
## [39] "stochvolnig"                     "zeroinflatedpoisson0"
## [41] "zeroinflatedpoisson1"            "zeroinflatedpoisson2"
## [43] "zeroinflatedbetabinomial0"       "zeroinflatedb
```

Available latent models

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

```
##  [1] "linear"       "iid"          "mec"          "meb"
##  [5] "rgeneric"     "rw1"          "rw2"          "crw2"
##  [9] "seasonal"     "besag"        "besag2"       "bym"
## [13] "bym2"         "besagproper"  "besagproper2" "fgn"
## [17] "ar1"          "ar"           "ou"           "generic"
## [21] "generic0"     "generic1"     "generic2"     "generic3"
## [25] "spde"         "spde2"        "spde3"        "iid1d"
## [29] "iid2d"        "iid3d"        "iid4d"        "iid5d"
## [33] "2diid"        "z"            "rw2d"         "rw2diid"
## [37] "slm"          "matern2d"     "copy"         "clinear"
## [41] "sigm"         "revsigm"      "log1exp"      "logdist"
```