R-INLA: An R-package for INLA

Håvard Rue

Department of Mathematical Sciences
NTNU, Norway

February 2, 2011

Implementing INLA

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

- The `GMRFLib`-library
- The `inla`-program
- The `INLA` package for `R`

Implementing INLA

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

- The `GMRFLib`-library
  - Basic library written in `C`, user friendly for programmers
- The `inla`-program
- The `INLA` package for `R`

Implementing INLA

All procedures required to perform INLA need to be carefully implemented to achieve a good speed; easy 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
- The `INLA` package for `R`

Implementing INLA

All procedures required to perform INLA need to be carefully implemented to achieve a good speed; easy 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"-file definitions
  - Similar interface as other `R` packages

R-INLA

- Visit the www-site

  `www.r-inla.org`

  and follow the instructions.
- www-site contains source-code, examples, reports +++
- The first time do

  ```
  > source("http://www.math.ntnu.no/inla/givmeINLA.R")
  ```

  Later, you can upgrade the package doing

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

  or if you want the test-version

  ```
  > inla.upgrade(testing=TRUE)
  ```
- Available for Linux, Windows and Mac
- Use OpenMP to do multi-threading

R-INLA

- Visit the www-site

  `www.r-inla.org`

  and follow the instructions.
- www-site contains source-code, examples, reports +++
- The first time do

  ```
  > source("http://www.math.ntnu.no/inla/givmeINLA.R")
  ```

  Later, you can upgrade the package doing

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

  or if you want the test-version

  ```
  > inla.upgrade(testing=TRUE)
  ```
- Available for Linux, Windows and Mac
- Use OpenMP to do multi-threading

R-INLA

- Visit the www-site

  `www.r-inla.org`

  and follow the instructions.
- www-site contains source-code, examples, reports +++
- The first time do

  ```
  > source("http://www.math.ntnu.no/inla/givmeINLA.R")
  ```

  Later, you can upgrade the package doing

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

  or if you want the test-version

  ```
  > inla.upgrade(testing=TRUE)
  ```
- Available for Linux, Windows and Mac
- Use OpenMP to do multi-threading

R-INLA

- Visit the www-site

  `www.r-inla.org`

  and follow the instructions.
- www-site contains source-code, examples, reports +++
- The first time do

  ```
  > source("http://www.math.ntnu.no/inla/givmeINLA.R")
  ```

  Later, you can upgrade the package doing

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

  or if you want the test-version

  ```
  > inla.upgrade(testing=TRUE)
  ```
- Available for Linux, Windows and Mac
- Use OpenMP to do multi-threading

R-INLA

- Visit the www-site

  `www.r-inla.org`

  and follow the instructions.
- www-site contains source-code, examples, reports +++
- The first time do

  ```
  > source("http://www.math.ntnu.no/inla/givmeINLA.R")
  ```

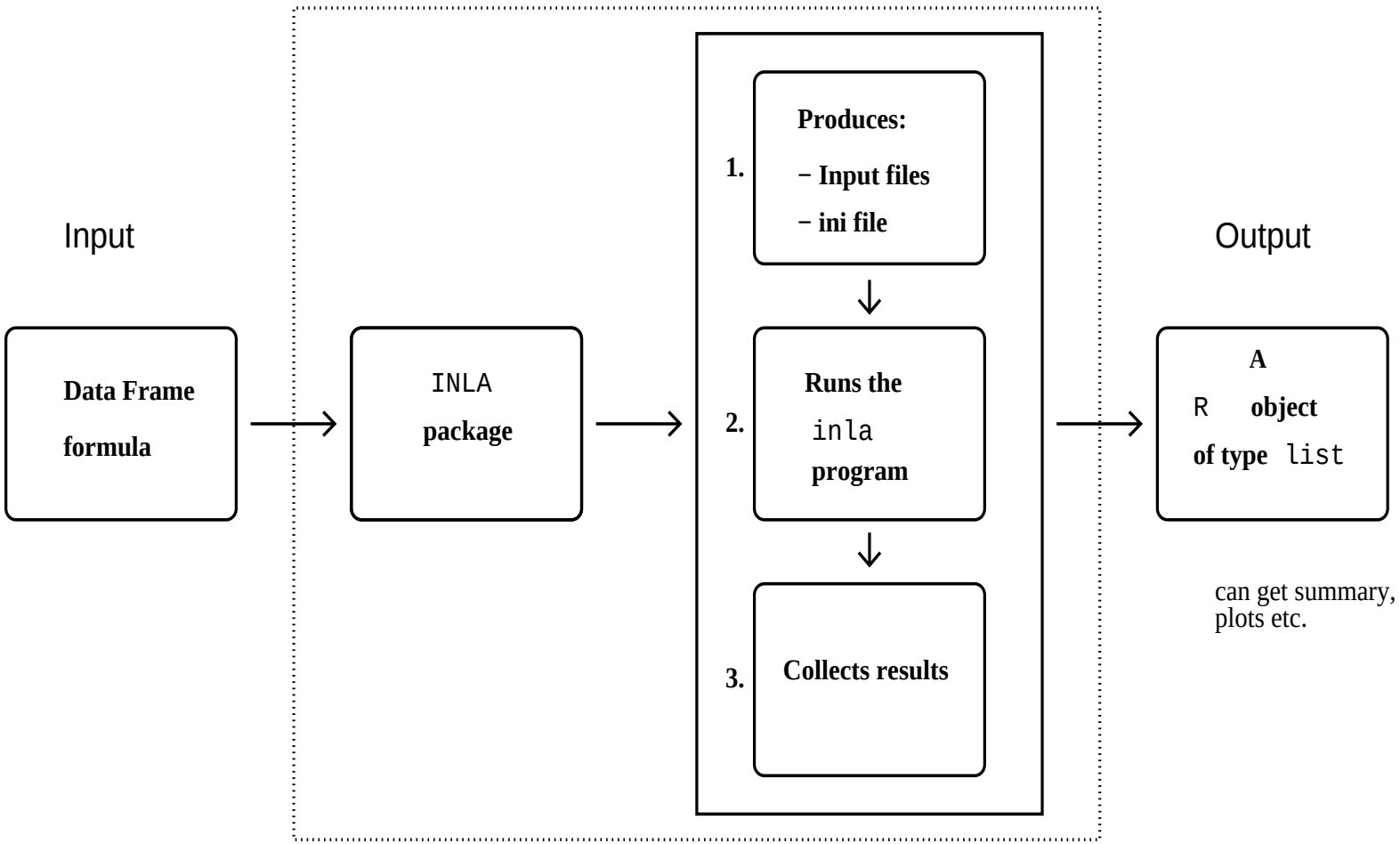
  Later, you can upgrade the package doing

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

  or if you want the test-version

  ```
  > inla.upgrade(testing=TRUE)
  ```
- Available for Linux, Windows and Mac
- Use OpenMP to do multi-threading

The INLA *package for* R



Model specification the INLA *package (I)*

Assume the following model:

$$\begin{aligned} y &\sim \pi(y|\eta) \\ \eta &= g(\lambda) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + f(x_3) \end{aligned}$$

where

$$\begin{aligned} x_1, x_2 &\quad \text{are covariates, linear effect} \\ \beta_i &\sim \mathcal{N}(0, \tau_1^{-1}) \\ x_3 &\quad \text{can be the index for spatial effect, random effect, } +++ \\ \{f_1, f_2, \ldots\} &\sim \mathcal{N}(0, Q_f^{-1}(\tau_2)) \end{aligned}$$

Model specification the INLA *package II*

The model is specified in R through a formula, similar to glm/gam++:

```
> formula = y ~ x1 + x2 + f(x3, ...)
```

The f() function is used to specify various "random"-effects in the model.

Some models

- iid, iid1d, ii2d, iid3d: random effects
- rw1, rw2, ar1: smooth effect of covariates or time effect
- seasonal: seasonal effect
- besag: spatial effect (CAR model)
- generic: user defined precision matrix

Model specification the INLA *package II*

The model is specified in R through a formula, similar to glm/gam++:

```
> formula = y ~ x1 + x2 + f(x3, ...)
```

The f() function is used to specify various "random"-effects in the model.
Some models

- iid, iid1d, ii2d, iid3d: random effects
- rw1, rw2, ar1: smooth effect of covariates or time effect
- seasonal: seasonal effect
- besag: spatial effect (CAR model)
- generic: user defined precision matrix

Main functions of the INLA *package*

- `f()` Define your model as a formula
- `inla()` Run the analysis
- `summary()`
- `plot()`
- `inla.hyperpar()`
- `inla.cpo()`

Documentation is available at `www.r-inla.org` and has help-pages in `R`

```
n = 100
x = sort(runif(n))
y = 1 + x + rnorm(n, sd = 0.1)
plot(x,y)

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

result.cpo = inla(formula,
        data = data.frame(x,y),
        family = "gaussian",
        control.compute = list(cpo=T))

x.pred = 2
xx = c(x, x.pred)
yy = c(y, NA)
```

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
....							
59	1	4	3	2	1	12	37

1. Mixed model with repeated Poisson counts II

The model

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

$$
\begin{aligned}
\alpha_i &\sim \mathcal{N}(0, \tau_\alpha) && \tau_\alpha \ \text{known} \\
\text{Ind}_j &\sim \mathcal{N}(0, \tau_{\text{Ind}}) && \tau_{\text{Ind}} \sim \text{Gamma}(a_1, b_1) \\
\beta_j k &\sim \mathcal{N}(0, \tau_\beta) && \tau_\beta \sim \text{Gamma}(a_2, b_2)
\end{aligned}
$$

EPIL

The `Epil` data frame:

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

Specifying the model:

```
formula = y ∼ log(Base/4) + Trt + I(Trt *
log(Base/4)) + log(Age) + V4 + f(Ind, model = "iid")
+ f(rand, model="iid")
```

Running `inla`

```
> result = inla(formula, family="poisson", data =
Epil)
```

EPIL

The `Epil` data frame:

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

Specifying the model:

```
formula = y ~ log(Base/4) + Trt + I(Trt *
log(Base/4)) + log(Age) + V4 + f(Ind, model = "iid")
                + f(rand, model="iid")
```

Running `inla`

```
> result = inla(formula, family="poisson", data =
    Epil)
```

EPIL

The `Epil` data frame:

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

Specifying the model:

```
formula = y ~ log(Base/4) + Trt + I(Trt *
log(Base/4)) + log(Age) + V4 + f(Ind, model = "iid")
                + f(rand, model="iid")
```

Running `inla`

```
> result = inla(formula, family="poisson", data =
                                        Epil)
```

EPIL

Some option of the `inla()` function:

- `verbose=TRUE` shows the output from the inla-program
- `keep=TRUE` keeps the `ini` file and all input files.

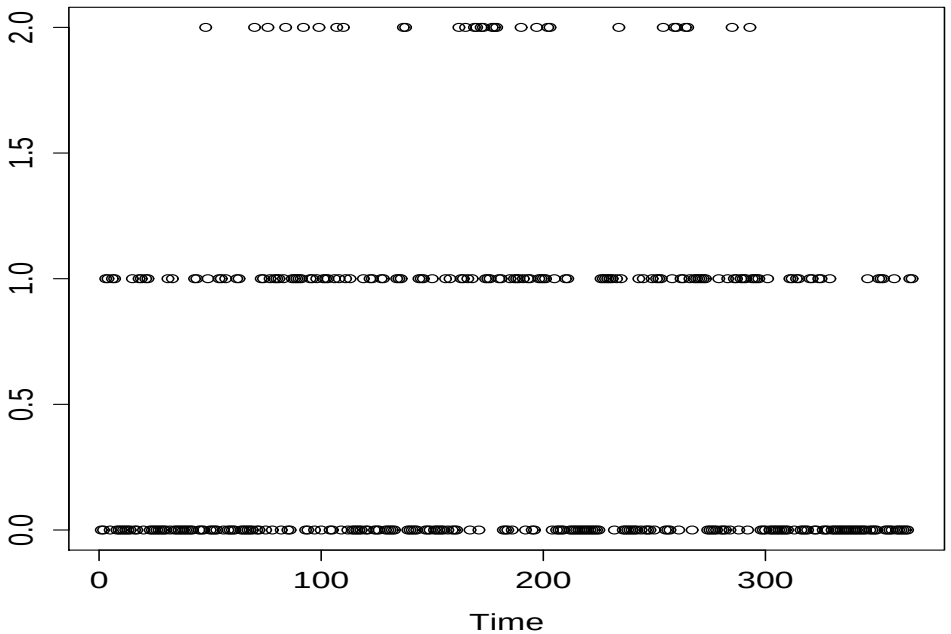
```
data(Epil)
my.center = function(x) (x - mean(x))

Epil$CTrt    = my.center(Epil$Trt)
Epil$ClBase4 = my.center(log(Epil$Base/4))
Epil$CV4     = my.center(Epil$V4)
Epil$ClAge   = my.center(log(Epil$Age))

formula = y ~ ClBase4*CTrt + ClAge + CV4 +
    f(Ind, model="iid") +
    f(rand, model="iid")

result = inla(formula,family="poisson", data = Epil,
        verbose=TRUE, keep=TRUE)
```

Smoothing binary times series



Number of days in Tokyo with rainfall above 1 mm in 1983-84. We want to estimate the probability of rain p_t for calendar day $t = 1, \ldots, 366$

2. A model with time series component II

The model

$$
\begin{aligned}
y_t &\sim \text{Binomial}(n_t, p_t);\ t = 1, \ldots, 365 \\
p_t &= \frac{\exp(\eta_t)}{1+\exp(\eta_t)} \\
\eta_t &= f(t) \\
\mathbf{f} &= \{f_1, \ldots, f_{366}\} \sim \text{cyclic RW2}(\tau) \\
\tau &\sim \text{Gamma}(1, 0.0001)
\end{aligned}
$$

Smoothing binary time series

The Tokyo data frame:

y	n	time
0	2	1
0	2	2
1	2	3
⋮		

Smoothing binary time series

The Tokyo data frame:

y	n	time
0	2	1
0	2	2
1	2	3
⋮		

Specifying the model:

```
formula = y ~ f(time, model="rw2", cyclic=TRUE,
param=c(1,0.0001))-1
```

Smoothing binary time series

The Tokyo data frame:

y	n	time
0	2	1
0	2	2
1	2	3
⋮		

Specifying the model:

```
formula = y ~ f(time, model="rw2", cyclic=TRUE,
param=c(1,0.0001))-1
```

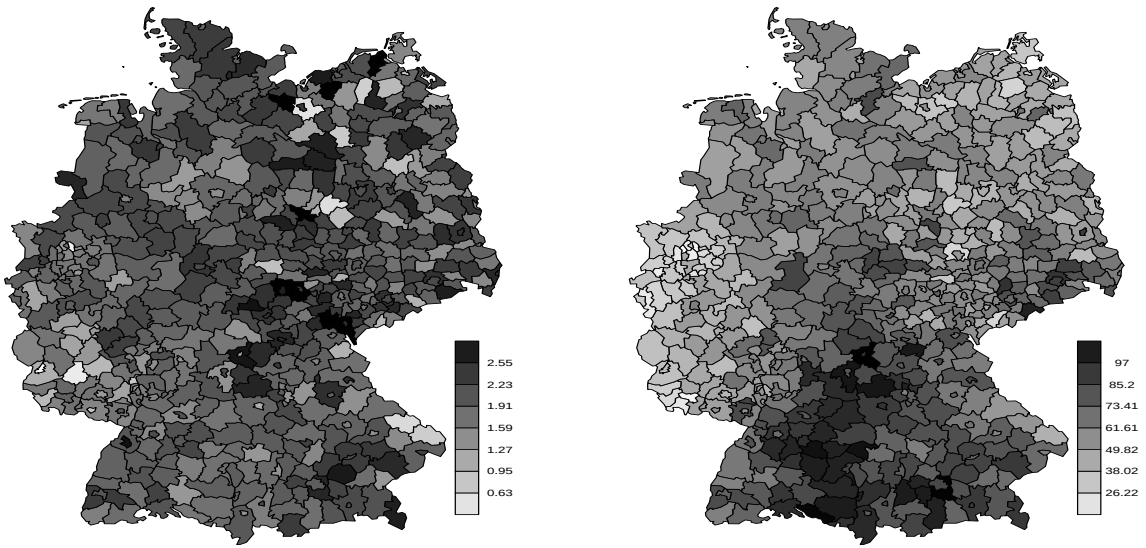
Running inla

```
result = inla(formula,family="binomial", Ntrials=n,
data=Tokyo)
```

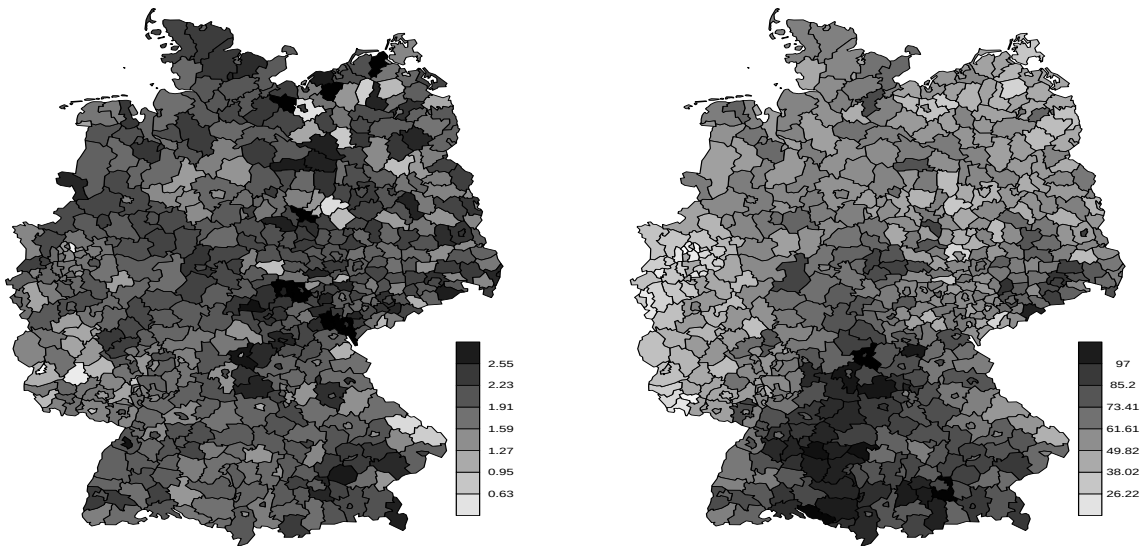
```
data(Tokyo)
n = dim(Tokyo)[1]
formula = y ~ f(time, model="rw2",
                  cyclic=TRUE, param=c(1,0.0001)) - 1
result = inla(formula, family="binomial", Ntrials=n,
              data=Tokyo)
```

Disease mapping in Germany

Larynx cancer mortality counts are observed in the 544 district of Germany from 1986 to 1990 and level of smoking consumption (100 possible values).



y_i, $i = 1, \ldots, 544$ counts of cancer mortality in Region i
E_i, $i = 1, \ldots, 544$ known variable accounting for demographic variation in Region i
c_i, $i = 1, \ldots, 544$ level of smoking consumption registered in Region i



The model

$$\begin{aligned} y_i &\sim \text{Poisson}\{E_i \exp(\eta_i)\}; \; i = 1, \ldots, 544 \\ \eta_i &= \mu + f(c_i) + f_s(s_i) + f_u(s_i) \end{aligned}$$

where:

- $f(c_i)$ is a smooth effect of the covariate

$$\mathbf{f} = \{f_1, \ldots, f_{100}\} \sim \text{RW2}(\tau_f)$$

- $f_s(s_i)$ is a spatial effect modelled as an intrinsic GMRF

$$f_s(s)|f_s(s'), s \neq s', \lambda_s \sim \mathcal{N}(\frac{1}{n_s} \sum_{s \sim s'} f_s(s'), \frac{\tau_{f_s}}{n_s})$$

- $f_u(s_i)$ is a random effect

$$\mathbf{f}_u = \{f_u(s_1), \ldots, f_u(s_{544})\} \sim \mathbf{N}(0, \tau_{f_u} \mathbf{I})$$

- μ is an intercept term $\mu \sim \mathcal{N}(0, 0.0001)$

The model

$$\begin{aligned} y_i &\sim \text{Poisson}\{E_i \exp(\eta_i)\}; \; i = 1, \ldots, 544 \\ \eta_i &= \mu + f(c_i) + f_s(s_i) + f_u(s_i) \end{aligned}$$

where:

- $f(c_i)$ is a smooth effect of the covariate

$$\mathbf{f} = \{f_1, \ldots, f_{100}\} \sim \text{RW2}(\tau_f)$$

- $f_s(s_i)$ is a spatial effect modelled as an intrinsic GMRF

$$f_s(s)|f_s(s'), s \neq s', \lambda_s \sim \mathcal{N}(\frac{1}{n_s} \sum_{s \sim s'} f_s(s'), \frac{\tau_{f_s}}{n_s})$$

- $f_u(s_i)$ is a random effect

$$\mathbf{f}_u = \{f_u(s_1), \ldots, f_u(s_{544})\} \sim \mathbf{N}(0, \tau_{f_u} \mathbf{I})$$

- μ is an intercept term $\mu \sim \mathcal{N}(0, 0.0001)$

The model

$$\begin{aligned} y_i &\sim \text{Poisson}\{E_i \exp(\eta_i)\}; \; i = 1, \ldots, 544 \\ \eta_i &= \mu + f(c_i) + f_s(s_i) + f_u(s_i) \end{aligned}$$

where:

- $f(c_i)$ is a smooth effect of the covariate

$$\mathbf{f} = \{f_1, \ldots, f_{100}\} \sim \text{RW2}(\tau_f)$$

- $f_s(s_i)$ is a spatial effect modelled as an intrinsic GMRF

$$f_s(s)|f_s(s'), s \neq s', \lambda_s \sim \mathcal{N}(\frac{1}{n_s}\sum_{s\sim s'} f_s(s'), \frac{\tau_{f_s}}{n_s})$$

- $f_u(s_i)$ is a random effect

$$\mathbf{f}_u = \{f_u(s_1), \ldots, f_u(s_{544})\} \sim \mathbf{N}(0, \tau_{f_u}\mathbf{I})$$

- μ is an intercept term $\mu \sim \mathcal{N}(0, 0.0001)$

The model

$$\begin{aligned} y_i &\sim \text{Poisson}\{E_i \exp(\eta_i)\}; \; i = 1, \ldots, 544 \\ \eta_i &= \mu + f(c_i) + f_s(s_i) + f_u(s_i) \end{aligned}$$

where:

- $f(c_i)$ is a smooth effect of the covariate

$$\mathbf{f} = \{f_1, \ldots, f_{100}\} \sim \text{RW2}(\tau_f)$$

- $f_s(s_i)$ is a spatial effect modelled as an intrinsic GMRF

$$f_s(s)|f_s(s'), s \neq s', \lambda_s \sim \mathcal{N}(\frac{1}{n_s}\sum_{s\sim s'} f_s(s'), \frac{\tau_{f_s}}{n_s})$$

- $f_u(s_i)$ is a random effect

$$\mathbf{f}_u = \{f_u(s_1), \ldots, f_u(s_{544})\} \sim \mathbf{N}(0, \tau_{f_u}\mathbf{I})$$

- μ is an intercept term $\mu \sim \mathcal{N}(0, 0.0001)$

The model

$$\begin{aligned} y_i &\sim \text{Poisson}\{E_i \exp(\eta_i)\};\ i = 1, \ldots, 544 \\ \eta_i &= \mu + f(c_i) + f_s(s_i) + f_u(s_i) \end{aligned}$$

where:

- $f(c_i)$ is a smooth effect of the covariate

$$\mathbf{f} = \{f_1, \ldots, f_{100}\} \sim \text{RW2}(\tau_f)$$

- $f_s(s_i)$ is a spatial effect modelled as an intrinsic GMRF

$$f_s(s)|f_s(s'), s \neq s', \lambda_s \sim \mathcal{N}(\frac{1}{n_s}\sum_{s\sim s'} f_s(s'), \frac{\tau_{f_s}}{n_s})$$

- $f_u(s_i)$ is a random effect

$$\mathbf{f}_u = \{f_u(s_1), \ldots, f_u(s_{544})\} \sim \mathbf{N}(0, \tau_{f_u}\mathbf{I})$$

- μ is an intercept term $\mu \sim \mathcal{N}(0, 0.0001)$

For identifiably we define a sum-to-zero constraint for all intrinsic models, so

$$\begin{aligned} \sum_s f_s(s) &= 0 \\ \sum_i f_i &= 0 \end{aligned}$$

Prior for the precision parameters:

$$\begin{aligned} \tau_f &\sim \text{Gamma}(1, 0.00005) \\ \tau_{f_s} &\sim \text{Gamma}(1, 0.05) \\ \tau_{f_u} &\sim \text{Gamma}(1, 0.001) \end{aligned}$$

The `Germany` data frame:

region	E	Y	x
0	7.965008	8	56
1	22.836219	22	65

The model is:

$$\eta_i = \mu + f(c_i) + f_s(s_i) + f_u(s_i)$$

- The data set has to contain *one separate column for each term specified through* `f()` so in this case we have to add one column.

```
> Germany = cbind(Germany, region.struct=Germany$region)
```

- We also need the graph file where the neighbourhood structure is specified `germany.graph`

The `Germany` data frame:

region	E	Y	x
0	7.965008	8	56
1	22.836219	22	65

The model is:

$$\eta_i = \mu + f(c_i) + f_s(s_i) + f_u(s_i)$$

- The data set has to contain *one separate column for each term specified through* `f()` so in this case we have to add one column.

  ```
  > Germany = cbind(Germany, region.struct=Germany$region)
  ```

- We also need the graph file where the neighbourhood structure is specified `germany.graph`

The `Germany` data frame:

region	E	Y	x
0	7.965008	8	56
1	22.836219	22	65

The model is:

$$\eta_i = \mu + f(c_i) + f_s(s_i) + f_u(s_i)$$

- The data set has to contain *one separate column for each term specified through* `f()` so in this case we have to add one column.

  ```
  > Germany = cbind(Germany, region.struct=Germany$region)
  ```

- We also need the graph file where the neighbourhood structure is specified `germany.graph`

The new data set is:

region	E	Y	x	region.struct
0	7.965008	8	56	0
1	22.836219	22	65	1

Then the formula is

```
formula <- Y ~
f(region.struct,model="besag",graph.file="germany.graph",
param=c(1,0.00005))+f(x,model="rw2",param=c(1,0.05))+f(region)
```

The new data set is:

region	E	Y	x	region.struct
0	7.965008	8	56	0
1	22.836219	22	65	1

Then the formula is

```
formula <- Y ~
f(region.struct,model="besag",graph.file="germany.graph",
param=c(1,0.00005))+f(x,model="rw2",param=c(1,0.05))+f(region)
```

The sum-to-zero constraint is *default* in the `inla` function for all *intrinsic models*.

The new data set is:

region	E	Y	x	region.struct
0	7.965008	8	56	0
1	22.836219	22	65	1

Then the formula is

```
formula <- Y ~
f(region.struct,model="besag",graph.file="germany.graph",
param=c(1,0.00005))+f(x,model="rw2",param=c(1,0.05))+f(region)
```

The sum-to-zero constraint is *default* in the `inla` function for all *intrinsic models*.

The new data set is:

region	E	Y	x	region.struct
0	7.965008	8	56	0
1	22.836219	22	65	1

Then the formula is

```
formula <- Y ~
f(region.struct,model="besag",graph.file="germany.graph",
param=c(1,0.00005))+f(x,model="rw2",param=c(1,0.05))+f(region)
```

The new data set is:

region	E	Y	x	region.struct
0	7.965008	8	56	0
1	22.836219	22	65	1

Then the formula is

```
formula <- Y ~
f(region.struct,model="besag",graph.file="germany.graph",
param=c(1,0.00005))+f(x,model="rw2",param=c(1,0.05))+f(region)
```

The location of the graph file has to be provided here (the graph file cannot be loaded in R)

The graph file

The `germany.graph` file:

```
544
1   2   12
2   3   10   11
3   5   6    8    15   387
⋮
```

- Total number of nodes in the graph
- Identifier for the node
- Number of neighbours
- Identifiers for the neighbours

The graph file

The `germany.graph` file:

```
544
1   2   12
2   3   10   11
3   5   6    8    15   387
⋮
```

- Total number of nodes in the graph
- Identifier for the node
- Number of neighbours
- Identifiers for the neighbours

The graph file

The `germany.graph` file:

```
544
1    2    12
2    3    10    11
3    5    6     8     15    387
⋮
```

- Total number of nodes in the graph
- Identifier for the node
- Number of neighbours
- Identifiers for the neighbours

The graph file

The `germany.graph` file:

```
544
1    2    12
2    3    10    11
3    5    6     8     15    387
⋮
```

- Total number of nodes in the graph
- Identifier for the node
- Number of neighbours
- Identifiers for the neighbours

The graph file

The `germany.graph` file:

```
544
1    2    12
2    3    10    11
3    5    6     8     15    387
⋮
```

- Total number of nodes in the graph
- Identifier for the node
- Number of neighbours
- Identifiers for the neighbours

```
data(Germany)
g = system.file("demodata/germany.graph", package="INLA")
source(system.file("demodata/Bym-map.R", package="INLA"))
Germany = cbind(Germany, region.struct=Germany$region)

# standard BYM model
formula1 = Y ~ f(region.struct,model="besag",graph.file=g) +
               f(region,model="iid")

# with linear covariate
formula2 = Y ~ f(region.struct,model="besag",graph.file=g) +
               f(region,model="iid") + x

# with smooth covariate
formula3 = Y ~ f(region.struct,model="besag",graph.file=g) +
               f(region,model="iid") + f(x, model="rw2")

result1 = inla(formula1,family="poisson",data=Germany,E=E,
               control.compute=list(dic=TRUE))
```

```
result1 = inla(formula1,family="poisson",data=Germany,E=E,
               control.compute=list(dic=TRUE))

result2 = inla(formula2,family="poisson",data=Germany,E=E,
               control.compute=list(dic=TRUE))

result3 = inla(formula3,family="poisson",data=Germany,E=E,
               control.compute=list(dic=TRUE))
```

Space-varying regression

Number of (insurance-type) losses N_{kt} in 431 municipalities/regions of Norway in relation to one weather covariate W_{kt}.
The likelihood is

$$N_{kt} \sim \text{Poisson}(A_{kt}\ p_{kt});\ k = 1, \ldots, 431\ t = 1, \ldots, 10$$

The model for $\log p_{kt}$ is:

$$\log p_{kt} = \beta_0 + \beta_k\ W_{kt}$$

where β_k is the regression coefficients for each municipality.

Borrow strength..

Few losses is in each region; high variability in the estimates.

Borrow strength, by letting $\{\beta_1, \ldots, \beta_{431}\}$ to be smooth in space:

$$\{\beta_1, \ldots, \beta_{431}\} \sim \text{CAR}(\tau_\beta)$$

Borrow strength..

Few losses is in each region; high variability in the estimates.

Borrow strength, by letting $\{\beta_1, \ldots, \beta_{431}\}$ to be smooth in space:

$$\{\beta_1, \ldots, \beta_{431}\} \sim \mathsf{CAR}(\tau_\beta)$$

The data set:

	y	region	W
1	0	1	0.4
2	0	1	0.4
10	0	1	0.4
11	1	2	0.2
12	0	2	0.2
20	0	2	0.2

Second argument in `f()` is the weight which defaults to 1

$$\eta_i = ... + w_i f_i + ...$$

is represented as

```
f(i, w, ...)
```

No need for sum-to-zero constraint!

```
norway = read.table("norway.dat", header=TRUE)
formula = y ~ 1 +
    f(region, W, model="besag",
      graph.file="norway.graph",
      constr=FALSE, param = c(1,0.01))
result = inla(formula, family="poisson", data=norway)
```

Survival models

patient	time	event	age	sex
1	8,16	1,1	28,28	0
2	23,13	1,0	48,48	1
3	22,18	1,1	32,32	0

- Times of infection from the time of insertion of catheter on 38 kidney patients using portable dialysis equipment.
- 2 observation for each patient (38 patients).
- Each time can be an *event* (infection) or a *censoring* (no infection)

Hazard rate and survival function

Density function:

$$y \sim f(y)$$

Survival function:

$$S(y) = 1 - F(y) = \int_y^{\infty} f(u)\ du$$

Hazard function:

$$\begin{aligned} h(y)\ dy &= \text{Prob}(y \leq Y < y + \ dy | Y > y) \\ h(y) &= \frac{f(t)}{S(t)} \end{aligned}$$

Cox proportinal hazards model

Write the hazard function for each patient as:

$$h(y_{ij}|w_i, \mathbf{x}_{ij}) = h_0(y_{ij})\ w_i\ \exp(\mathbf{x}_{ij}^T \boldsymbol{\beta});\ i = 1, \ldots, 38;\ j = 1, 2$$

where

$h_0(\cdot)$	is the baseline hazard function
w_i	is the log-Normal frailty effect associated with patient i
$\mathbf{x}_{ij}$	is the vector of observed covariates for patient i at observation j
$\boldsymbol{\beta}$	is a vector of unknown parameters

Cox proportinal hazard model

Can rewrite this a Poisson regression, augmenting data for each part of the picewise-constant baseline hazard.

The Kidney data

The `Kidney` data frame

time	event	age	sex	ID
8	1	28	0	1
16	1	28	0	1
23	1	48	1	2
13	0	48	1	2
22	1	32	0	3
28	1	32	0	3

```
data(Kidney)
formula = inla.surv(time,event) ~ age + sex + f(ID,model="iid")
result1 = inla(formula, family="coxph", data=Kidney)
result2 = inla(formula, family="weibull", data=Kidney)
result3 = inla(formula, family="exponential", data=Kidney)
```

Some more advanced features

- replicate
- more than one "family"
- copy
- linear combinations
- remote computing

Feature: replicate

"replicate" generates iid replicates from the same model with the same hyperparameters.

If $\mathbf{x} \mid \boldsymbol{\theta} \sim$ AR(1), then nrep=3, makes

$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$$

with mutually independent $\mathbf{x}_i$'s from AR(1) with the same $\boldsymbol{\theta}$

Most `f()`-models can be replicated

Example: replicate

```
n=100
x1 = arima.sim(n, model=list(ar=0.9)) + 1
x2 = arima.sim(n, model=list(ar=0.9)) - 1
y1 = rpois(n,exp(x1))
y2 = rpois(n,exp(x2))
y = c(y1,y2)
i = rep(1:n,2)
r = rep(1:2,each=n)
intercept = as.factor(r)
formula = y ~ f(i, model="ar1", replicate=r) + intercept -1
result = inla(formula, family = "poisson",
              data = data.frame(y=y,i=i,r=r))
```

More than one family

Every observation could have its own likelihood!

- Response is a matrix or list
- Each "column" defines a separate "family"
- Each "family" has its own hyperparameters

```
n=100
x1 = arima.sim(n, model=list(ar=0.9))
x2 = arima.sim(n, model=list(ar=0.9))
y1 = rbinom(n,size=1, prob=exp(x1)/(1+exp(x1)))
y2 = rpois(n,exp(x2))
y = matrix(NA, 2*n, 2)
y[  1:n, 1] = y1
y[n+1:n, 2] = y2
i = rep(1:n,2)
r = rep(1:2,each=n)
intercept = as.factor(r)
Ntrials = c(rep(1,n), rep(NA,n))

formula = y ~ f(i, model="rw1", replicate=r) + intercept -1
result = inla(formula, family = c("binomial", "poisson"),
                  Ntrials = Ntrials, data = data.frame(y=y,i=i,r=r),
          verbose=T)
```

More examples

Some rather advanced examples on `www.r-inla.org` using this feature

- Preferential sampling, geostatistics (marked point process)
- Weibull-survival data and "longitudinal" data

Feature: copy

The model

```
formula = y ~ f(i, ...) + ...
```

Only allow ONE element from each sub-model, to contribute to the linear predictor for each observation.

Sometimes this is not sufficient.

Feature: copy

Suppose

$$\eta_i = u_i + u_{i+1} + ...$$

Then we can code this as

```
formula = f(i, model="iid") + f(i.plus, copy="i")
```

- The copy-feature, creates an additional sub-model which is ϵ-close to the target.
- Many copies allowed
- Copy with unknown scaling (default scaling is fixed to 1).

Toy-example using these new tools

State-space model

$$y_t = x_t + v_t$$

$$x_t = 2x_{t-1} - x_{t-2} + w_t$$

Rewrite this as

$$y_t = x_t + v_t$$

$$0 = x_t - 2x_{t-1} + x_{t-2} + w_t$$

and implement this as two families

1. Observations y_t with precision $\text{Prec}(v_t)$
2. Observations 0 with precision $\text{Prec}(w_t)$, or Prec=HIGH.

Toy-example using these new tools

State-space model

$$y_t = x_t + v_t$$

$$x_t = 2x_{t-1} - x_{t-2} + w_t$$

Rewrite this as

$$y_t = x_t + v_t$$

$$0 = x_t - 2x_{t-1} + x_{t-2} + w_t$$

and implement this as two families

1. Observations y_t with precision Prec(v_t)
2. Observations 0 with precision Prec(w_t), or Prec=HIGH.

```
n = 100
m = n-2
y = sin((1:n)*0.2) + rnorm(n, sd=0.1)
formula = Y ~ f(i, model="iid", initial=-10, fixed=TRUE) +
              f(j, w, copy="i") + f(k, copy="i") +
              f(l, model ="iid") -1
Y = matrix(NA, n+m, 2)
Y[1:n,     1] = y
Y[1:m + n, 2] = 0
i = c(1:n, 3:n)                # x_t
j = c(rep(NA,n), 3:n -1)       # x_t-1
w = c(rep(NA,n), rep(-2,m)) # weights for j
k = c(rep(NA,n), 3:n -2)       # x_t-2
l = c(rep(NA,n), 1:m)          # v_t
r = inla(formula, data = data.frame(i,j,w,k,l,Y),
         family = c("gaussian", "gaussian"),
         control.data = list(list(), list(initial=10, fixed=TRUE
```

Linear combinations

Possible to extract extra information from the model through linear combinations of the latent field.

```
data(Epil)
my.center = function(x) (x - mean(x))

Epil$CTrt    = my.center(Epil$Trt)
Epil$ClBase4 = my.center(log(Epil$Base/4))
Epil$CV4     = my.center(Epil$V4)
Epil$ClAge   = my.center(log(Epil$Age))

formula = y ~ ClBase4*CTrt + ClAge + CV4 +
    f(Ind, model="iid") + f(rand, model="iid")

## Now I want the posterior for
##
##   1)    2*CTrt - CV4
##   2)    Ind[2] - rand[2]
##
lc1 = inla.make.lincomb( CTrt = 2, CV4 = -1)
names(lc1) = "lc1"
lc2 = inla.make.lincomb( Ind = c(NA,1), rand = c(NA,-1))
```

Feature: remote computing

For large/huge models, its more convenient to run the computations on the remote (Linux/Mac) computational server

```
inla(....,  inla.call="remote")
```

using `ssh` (and `Cygwin` on windows).