An introduction to INLA with a comparison to JAGS

Gianluca Baio

University College London
Department of Statistical Science

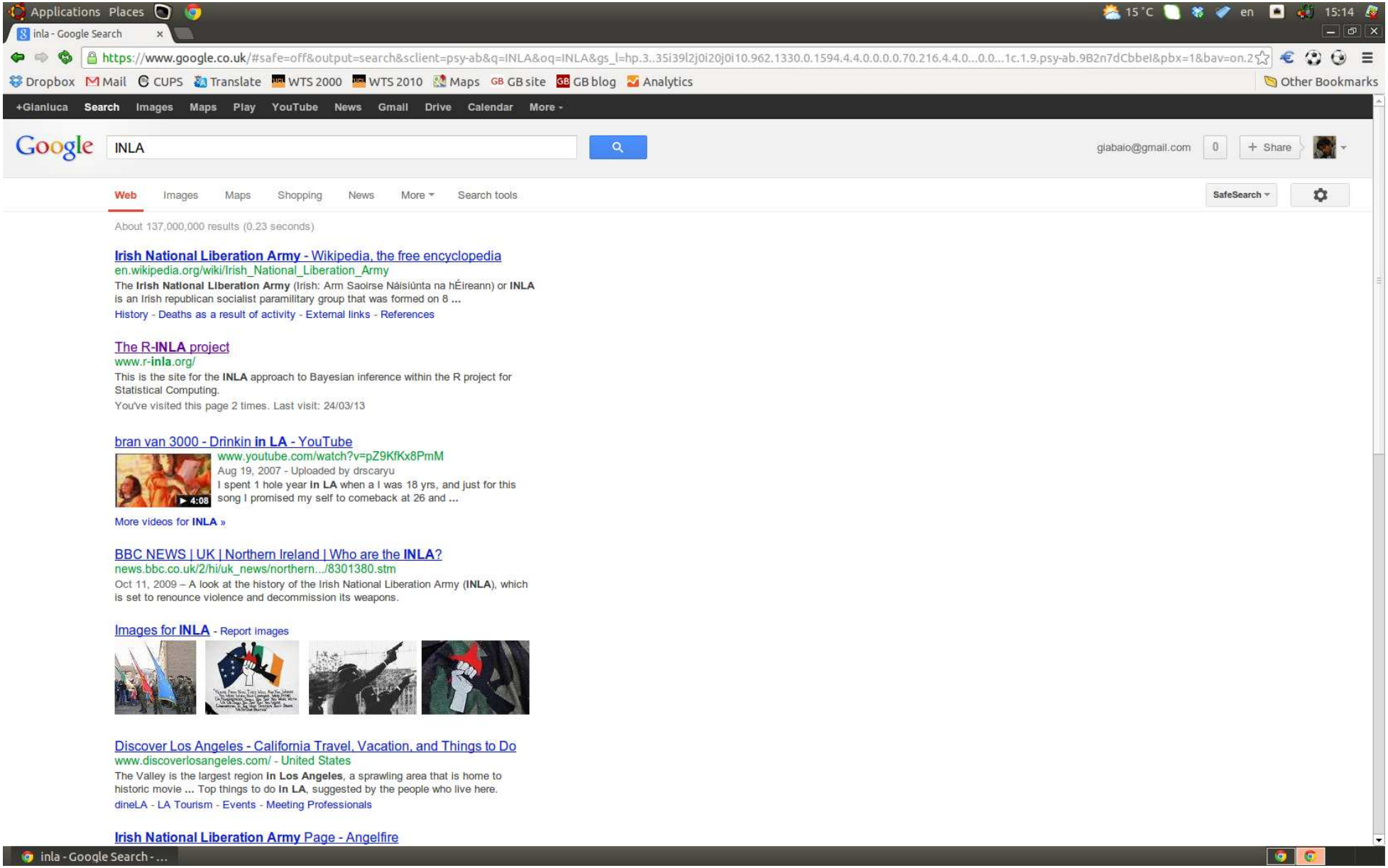
gianluca@stats.ucl.ac.uk

(Thanks to Håvard Rue, University of Science and Technology Trondheim, Norway)

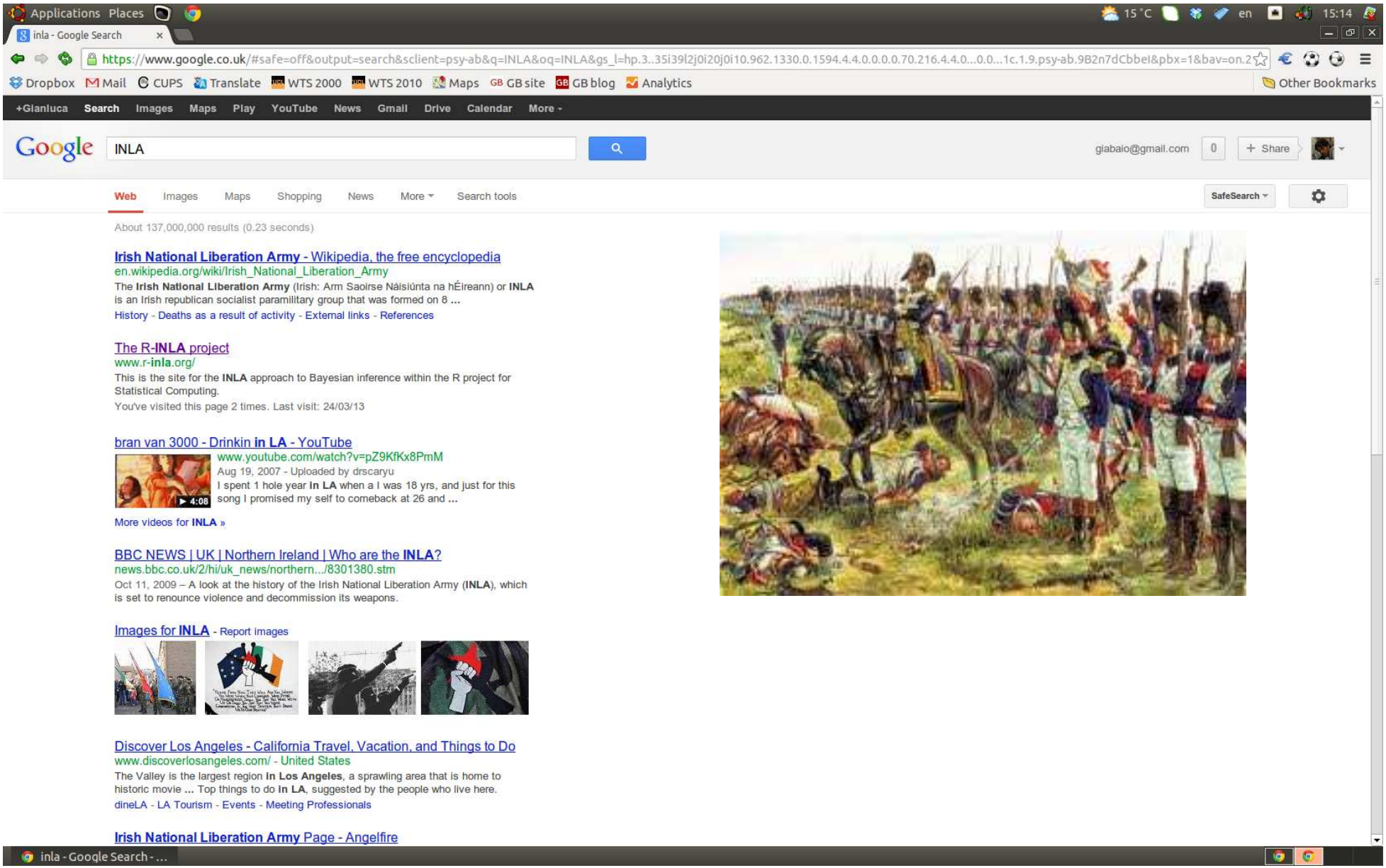
Universitat de Girona

Thursday 24 April 2014

Laplace's liberation army (?)



Laplace's liberation army (?)



Outline of presentation

1. **9.30 – 10.15: (Quick & moderately clean) introduction to Bayesian computation**
   - MCMC
   - Latent Gaussian models
   - Gaussian Markov Random Fields
2. 10.15 – 10.30: Coffee break
3. 10.30 – 11.15: Introduction to INLA
   - Basic ideas
   - Some details
   - A simple example
4. 11.15 – 11.30: Coffee break
5. 11.30 – 12.30: Using the package `R-INLA`
   - How does it work?
   - Some simple examples
   - (Slightly) more complex examples

Outline of presentation

1. 9.30 – 10.15: (Quick & moderately clean) introduction to Bayesian computation
   - MCMC
   - Latent Gaussian models
   - Gaussian Markov Random Fields
2. **10.15 – 10.30: Coffee break**
3. 10.30 – 11.15: Introduction to INLA
   - Basic ideas
   - Some details
   - A simple example
4. 11.15 – 11.30: Coffee break
5. 11.30 – 12.30: Using the package `R-INLA`
   - How does it work?
   - Some simple examples
   - (Slightly) more complex examples

Outline of presentation

1. 9.30 – 10.15: (Quick & moderately clean) introduction to Bayesian computation
   - MCMC
   - Latent Gaussian models
   - Gaussian Markov Random Fields
2. 10.15 – 10.30: Coffee break
3. **10.30 – 11.15: Introduction to INLA**
   - Basic ideas
   - Some details
   - A simple example
4. 11.15 – 11.30: Coffee break
5. 11.30 – 12.30: Using the package `R-INLA`
   - How does it work?
   - Some simple examples
   - (Slightly) more complex examples

Outline of presentation

1. 9.30 – 10.15: (Quick & moderately clean) introduction to Bayesian computation
   - MCMC
   - Latent Gaussian models
   - Gaussian Markov Random Fields
2. 10.15 – 10.30: Coffee break
3. 10.30 – 11.15: Introduction to INLA
   - Basic ideas
   - Some details
   - A simple example
4. **11.15 – 11.30: Coffee break**
5. 11.30 – 12.30: Using the package `R-INLA`
   - How does it work?
   - Some simple examples
   - (Slightly) more complex examples

Outline of presentation

1. 9.30 – 10.15: (Quick & moderately clean) introduction to Bayesian computation
   - MCMC
   - Latent Gaussian models
   - Gaussian Markov Random Fields
2. 10.15 – 10.30: Coffee break
3. 10.30 – 11.15: Introduction to INLA
   - Basic ideas
   - Some details
   - A simple example
4. 11.15 – 11.30: Coffee break
5. **11.30 – 12.30: Using the package `R-INLA`**
   - How does it work?
   - Some simple examples
   - (Slightly) more complex examples

(Quick & moderately clean) introduction to Bayesian computation

Bayesian computation

- In a (**very** small!) nutshell, Bayesian inference boils down to the computation of **posterior** and/or **predictive** distributions

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\boldsymbol{y} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} \qquad p(y^* \mid \boldsymbol{y}) = \int p(y^* \mid \boldsymbol{\theta})p(\boldsymbol{\theta} \mid \boldsymbol{y})d\boldsymbol{\theta}$$

Bayesian computation

- In a (**very** small!) nutshell, Bayesian inference boils down to the computation of **posterior** and/or **predictive** distributions

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\boldsymbol{y} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} \qquad p(y^* \mid \boldsymbol{y}) = \int p(y^* \mid \boldsymbol{\theta})p(\boldsymbol{\theta} \mid \boldsymbol{y})d\boldsymbol{\theta}$$

- Since the advent of simulation-based techniques (notably MCMC), Bayesian computation has enjoyed incredible development
- This has certainly been helped by dedicated software (eg `BUGS` and then `WinBUGS`, `OpenBUGS`, `JAGS`)
- MCMC methods are very general and can effectively be applied to "any" model

Bayesian computation

- In a (**very** small!) nutshell, Bayesian inference boils down to the computation of **posterior** and/or **predictive** distributions

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\boldsymbol{y} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} \qquad p(y^* \mid \boldsymbol{y}) = \int p(y^* \mid \boldsymbol{\theta})p(\boldsymbol{\theta} \mid \boldsymbol{y})d\boldsymbol{\theta}$$

- Since the advent of simulation-based techniques (notably MCMC), Bayesian computation has enjoyed incredible development
- This has certainly been helped by dedicated software (eg `BUGS` and then `WinBUGS`, `OpenBUGS`, `JAGS`)
- MCMC methods are very general and can effectively be applied to "any" model
- However:
  - Even if **in theory**, MCMC can provide (nearly) exact inference, given perfect convergence and MC error $\to 0$, in practice, this has to be balanced with model complexity and running time
  - This is particularly an issue for problems characterised by large data or very complex structure (eg hierarchical models)

MCMC — Gibbs sampling

- The objective is to build a Markov Chain (MC) that converges to the desired target distribution p (eg the unknown posterior distribution of some parameter of interest)
- Usually easy to create a MC, under mild "regularity conditions"

MCMC — Gibbs sampling

- The objective is to build a Markov Chain (MC) that converges to the desired target distribution p (eg the unknown posterior distribution of some parameter of interest)
- Usually easy to create a MC, under mild "regularity conditions"
- The **Gibbs sampling** (GS) is one of the most popular schemes for MCMC

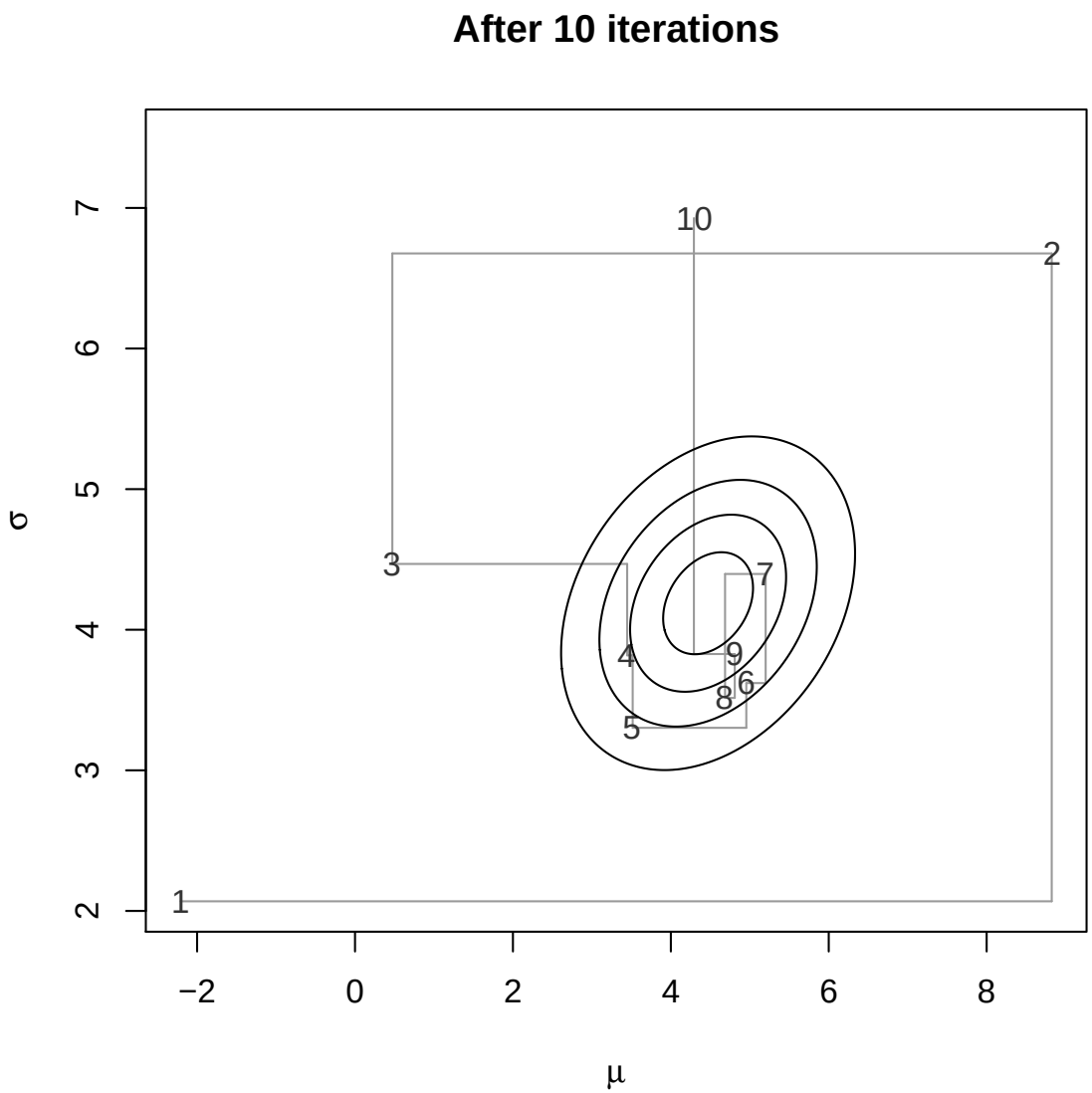
MCMC — Gibbs sampling

- The objective is to build a Markov Chain (MC) that converges to the desired target distribution p (eg the unknown posterior distribution of some parameter of interest)
- Usually easy to create a MC, under mild "regularity conditions"
- The **Gibbs sampling** (GS) is one of the most popular schemes for MCMC
  1. Select a set of initial values $(\theta_1^{(0)}, \theta_2^{(0)}, \ldots, \theta_J^{(0)})$
  2. Sample $\theta_1^{(1)}$ from the conditional distribution $p(\theta_1 \mid \theta_2^{(0)}, \theta_3^{(0)}, \ldots, \theta_J^{(0)}, y)$
     Sample $\theta_2^{(1)}$ from the conditional distribution $p(\theta_2 \mid \theta_1^{(1)}, \theta_3^{(0)}, \ldots, \theta_J^{(0)}, y)$
     ...
     Sample $\theta_J^{(1)}$ from the conditional distribution $p(\theta_J \mid \theta_1^{(1)}, \theta_2^{(1)}, \ldots, \theta_{J-1}^{(1)}, y)$

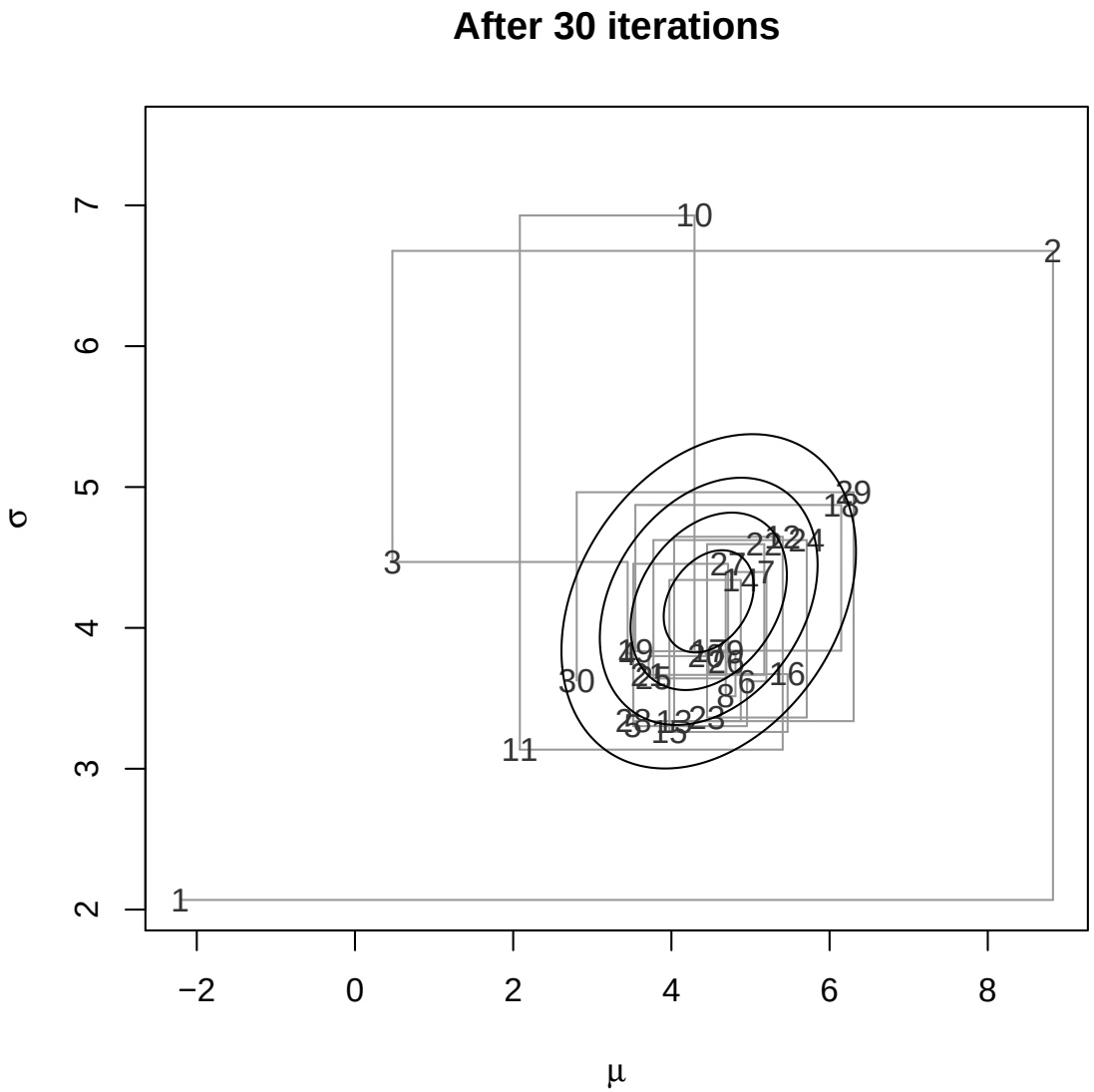
MCMC — Gibbs sampling

- The objective is to build a Markov Chain (MC) that converges to the desired target distribution p (eg the unknown posterior distribution of some parameter of interest)
- Usually easy to create a MC, under mild "regularity conditions"
- The **Gibbs sampling** (GS) is one of the most popular schemes for MCMC
  1. Select a set of initial values $(\theta_1^{(0)}, \theta_2^{(0)}, \ldots, \theta_J^{(0)})$
  2. Sample $\theta_1^{(1)}$ from the conditional distribution $p(\theta_1 \mid \theta_2^{(0)}, \theta_3^{(0)}, \ldots, \theta_J^{(0)}, y)$
     Sample $\theta_2^{(1)}$ from the conditional distribution $p(\theta_2 \mid \theta_1^{(1)}, \theta_3^{(0)}, \ldots, \theta_J^{(0)}, y)$
     ...
     Sample $\theta_J^{(1)}$ from the conditional distribution $p(\theta_J \mid \theta_1^{(1)}, \theta_2^{(1)}, \ldots, \theta_{J-1}^{(1)}, y)$
  3. Repeat step 2. for S times until convergence is reached to the target distribution $p(\boldsymbol{\theta} \mid y)$
  4. Use the sample from the target distribution to compute all relevant statistics: (posterior) mean, variance, credibility intervals, etc.

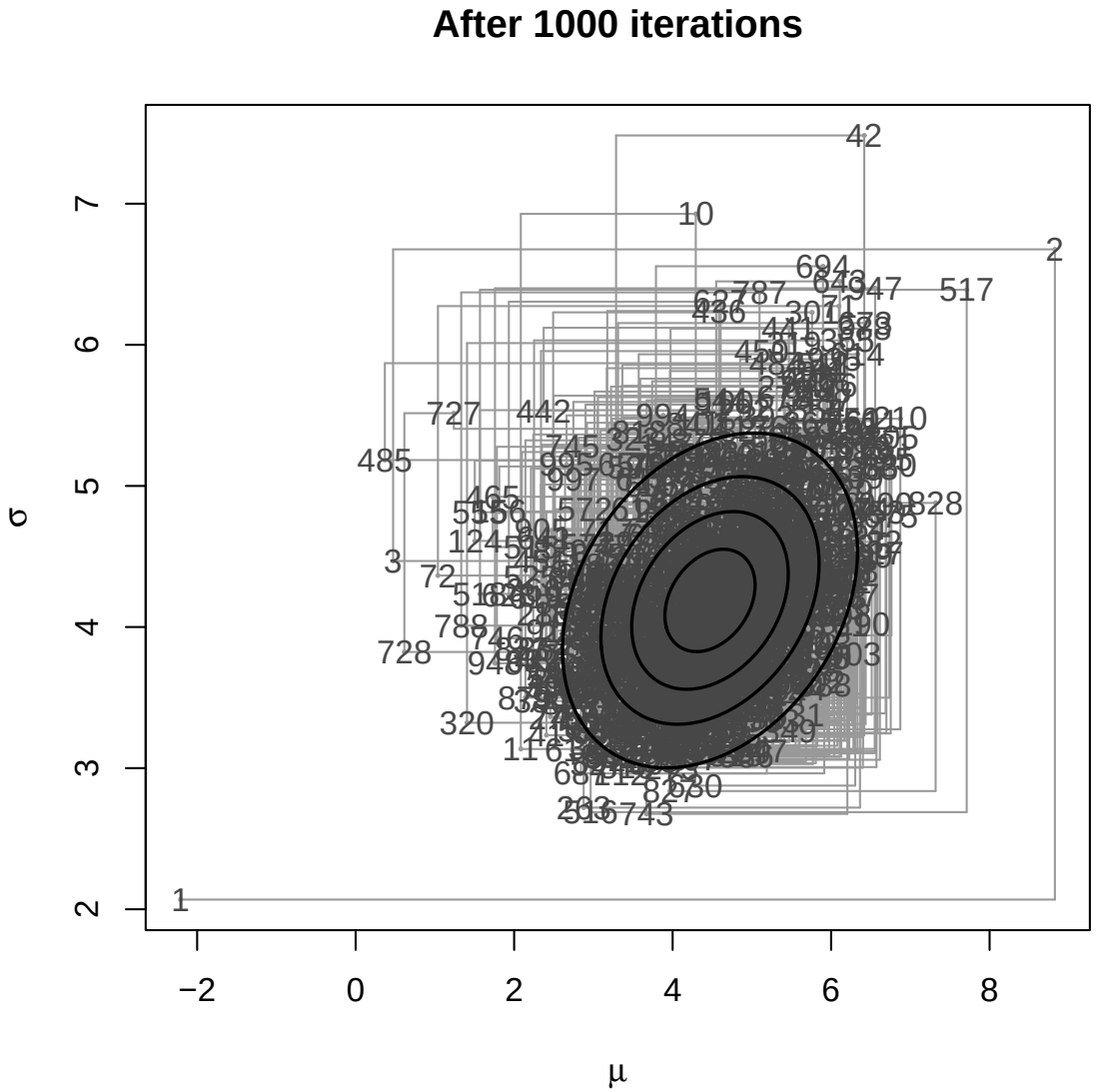
- If the *full conditionals* are not readily available, they need to be estimated (eg via Metropolis-Hastings or slice sampling) before applying the GS


After 10 iterations
σ
μ
2
3
4
5
6
7
−2
0
2
4
6
8
1
2
3
4
5
6
7
8
9
10

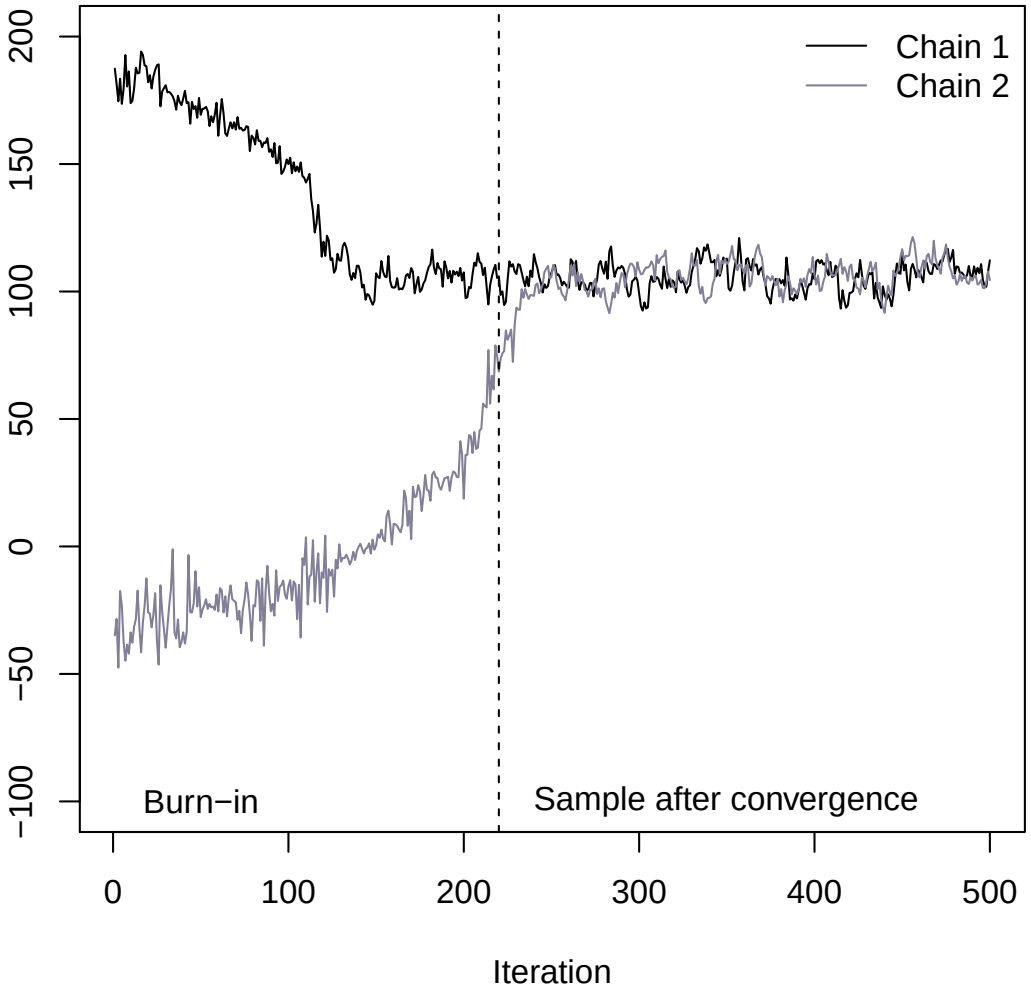
MCMC — convergence



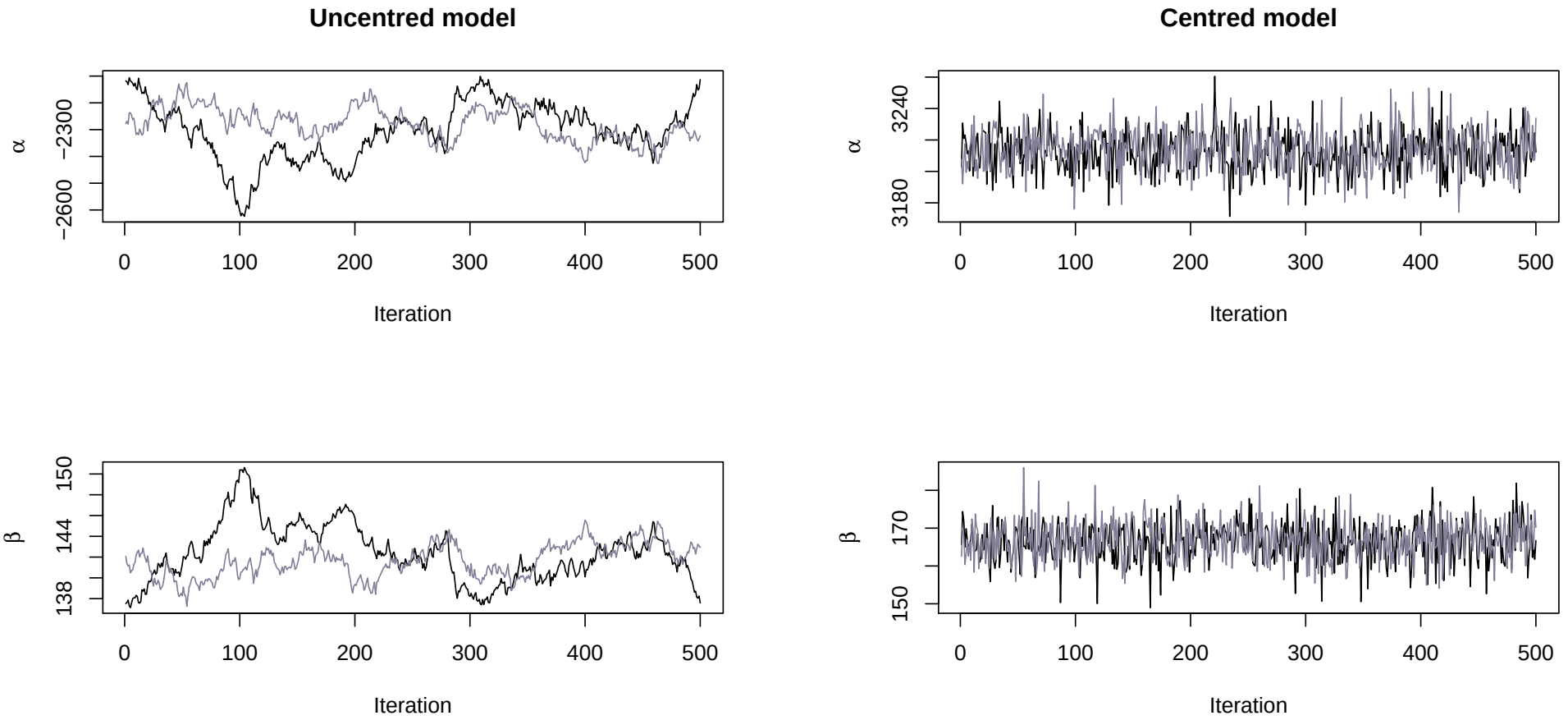
MCMC — convergence



MCMC — convergence



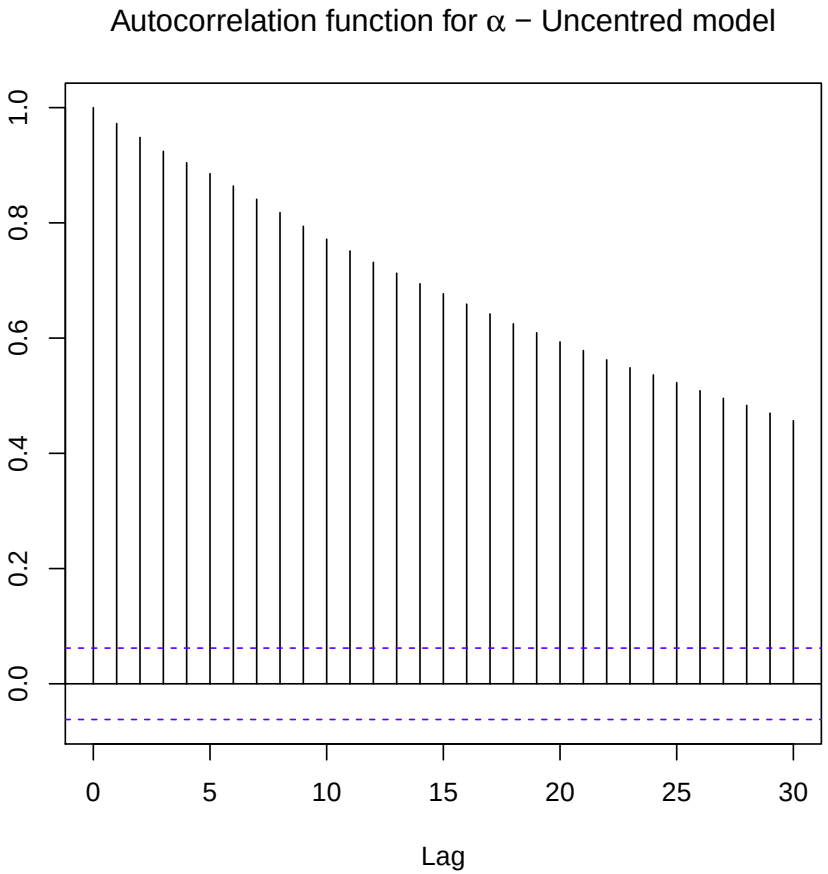
MCMC — convergence

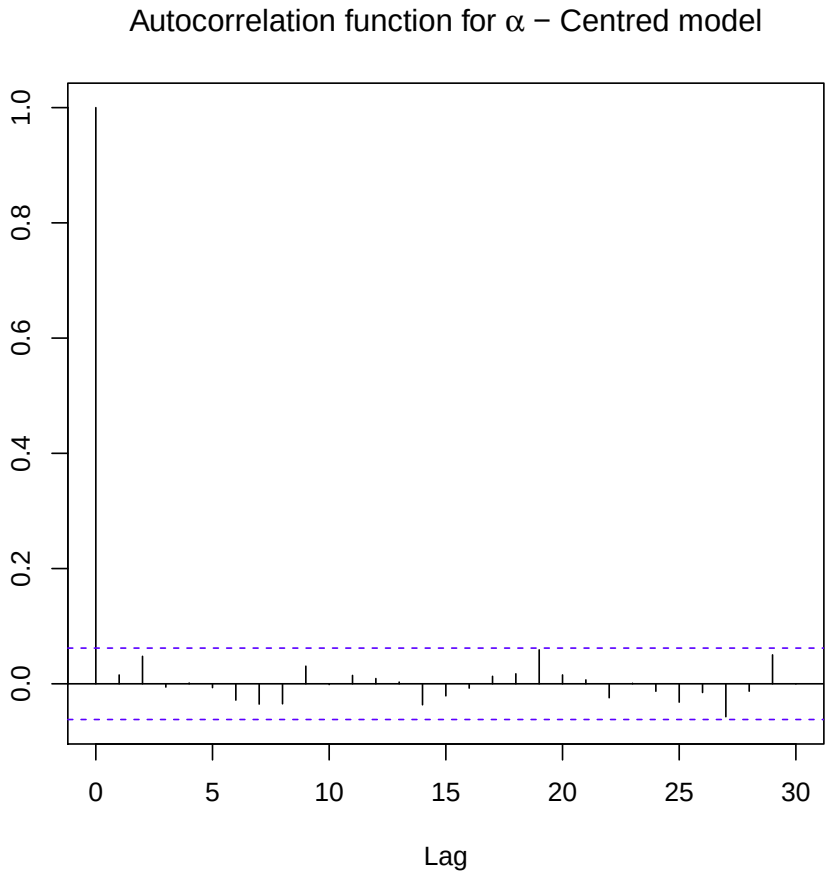


- Formal assessment of convergence: potential scale reduction

$$\hat{R} = \sqrt{\frac{\widehat{\mathrm{Var}}(\theta_k \mid \boldsymbol{y})}{W(\theta_k)}}$$

MCMC — autocorrelation

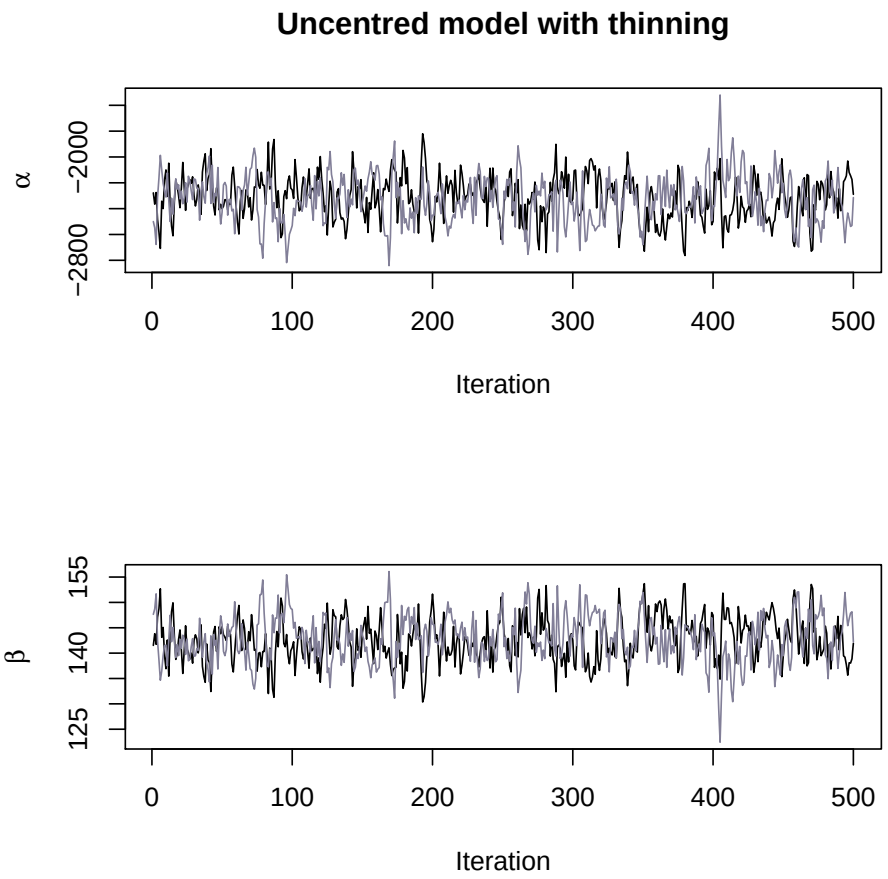


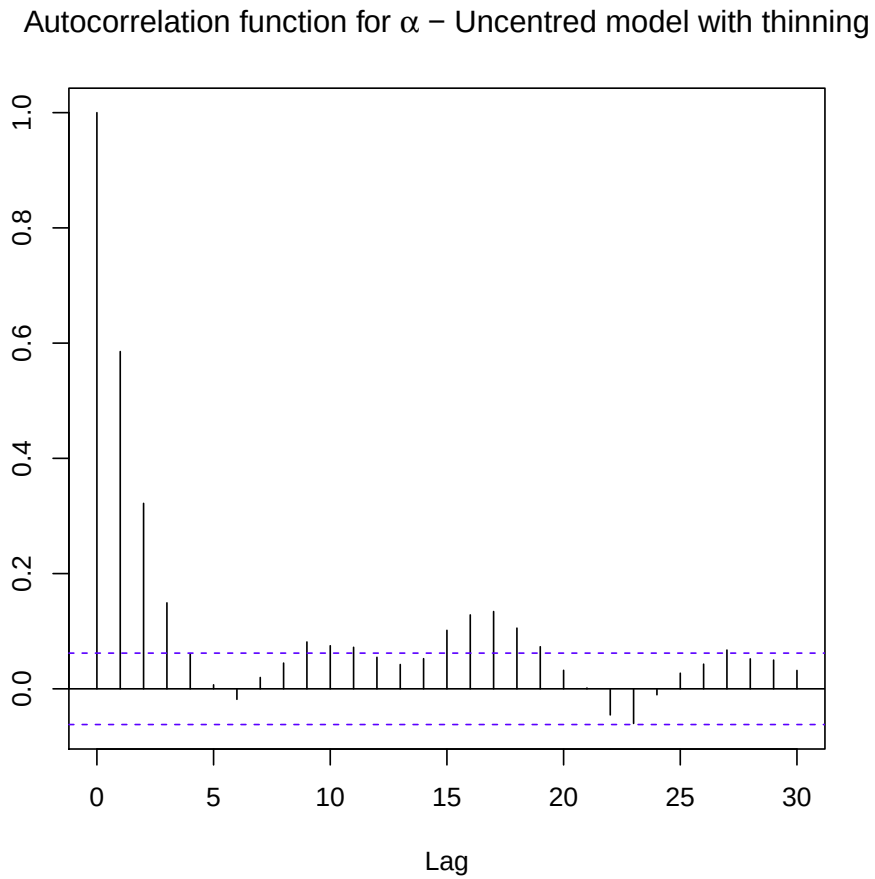


- Formal assessment of autocorrelation: effective sample size

$$n_{\text{eff}} = \frac{S}{1 + 2\sum_{t=1}^{\infty} \rho_t}$$

MCMC — brute force





MCMC — pros & cons

- "Standard" MCMC sampler are generally easy-ish to program and are in fact implemented in readily available software
- However, depending on the complexity of the problem, their efficiency might be limited

MCMC — pros & cons

- "Standard" MCMC sampler are generally easy-ish to program and are in fact implemented in readily available software
- However, depending on the complexity of the problem, their efficiency might be limited

- Possible solutions
  1. More complex **model specification**
     - Blocking
     - Overparameterisation

MCMC — pros & cons

- "Standard" MCMC sampler are generally easy-ish to program and are in fact implemented in readily available software
- However, depending on the complexity of the problem, their efficiency might be limited

- Possible solutions
  1. More complex **model specification**
     - Blocking
     - Overparameterisation
  2. More complex **sampling schemes**
     - Hamiltonian Monte Carlo
     - No U-turn sampling (eg `stan`)

MCMC — pros & cons

- "Standard" MCMC sampler are generally easy-ish to program and are in fact implemented in readily available software
- However, depending on the complexity of the problem, their efficiency might be limited

- Possible solutions
  1. More complex **model specification**
     - Blocking
     - Overparameterisation
  2. More complex **sampling schemes**
     - Hamiltonian Monte Carlo
     - No U-turn sampling (eg `stan`)
  3. Alternative methods of inference
     - Approximate Bayesian Computation (ABC)
     - INLA

MCMC — pros & cons

- "Standard" MCMC sampler are generally easy-ish to program and are in fact implemented in readily available software
- However, depending on the complexity of the problem, their efficiency might be limited

- Possible solutions
  1. More complex **model specification**
     - Blocking
     - Overparameterisation
  2. More complex **sampling schemes**
     - Hamiltonian Monte Carlo
     - No U-turn sampling (eg `stan`)
  3. Alternative methods of inference
     - Approximate Bayesian Computation (ABC)
     - INLA

The basic ideas revolve around

- Formulating the model using a specific characterisation
  - All models that can be formulated in this way have certain features in common, which facilitate the computational aspects
  - The characterisation is still quite general and covers a wide range of possible models (more on that later!)
  - **NB**: This implies less flexibility with respect to MCMC — but in many cases this is not a huge limitation!

Basics of INLA

The basic ideas revolve around

- Formulating the model using a specific characterisation
  - All models that can be formulated in this way have certain features in common, which facilitate the computational aspects
  - The characterisation is still quite general and covers a wide range of possible models (more on that later!)
  - **NB**: This implies less flexibility with respect to MCMC — but in many cases this is not a huge limitation!

- Use some basic probability conditions to approximate the relevant distributions

Basics of INLA

The basic ideas revolve around

- Formulating the model using a specific characterisation
  - All models that can be formulated in this way have certain features in common, which facilitate the computational aspects
  - The characterisation is still quite general and covers a wide range of possible models (more on that later!)
  - **NB**: This implies less flexibility with respect to MCMC — but in many cases this is not a huge limitation!

- Use some basic probability conditions to approximate the relevant distributions

- Compute the relevant quantities typically using numerical methods

Latent Gaussian models (LGMs)

- The general problem of (parametric) inference is posited by assuming a probability model for the observed data, as a function of some relevant parameters

$$\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{\psi} \sim p(\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{\psi}) = \prod_{i=1}^{n} p(y_i \mid \boldsymbol{\theta}, \boldsymbol{\psi})$$

Latent Gaussian models (LGMs)

- The general problem of (parametric) inference is posited by assuming a probability model for the observed data, as a function of some relevant parameters

$$\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{\psi} \sim p(\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{\psi}) = \prod_{i=1}^{n} p(y_i \mid \boldsymbol{\theta}, \boldsymbol{\psi})$$

- Often (in fact for a surprisingly large range of models!), we can assume that the parameters are described by a **Gaussian Markov Random Field** (GMRF)

$$\boldsymbol{\theta} \mid \boldsymbol{\psi} \sim \text{Normal}(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\psi}))$$
$$\theta_l \perp\!\!\!\perp \theta_m \mid \boldsymbol{\theta}_{-lm}$$

  where
  - The notation "$-lm$" indicates all the other elements of the parameters vector, excluding elements l and m
  - The covariance matrix $\boldsymbol{\Sigma}$ depends on some hyper-parameters $\boldsymbol{\psi}$

Latent Gaussian models (LGMs)

- The general problem of (parametric) inference is posited by assuming a probability model for the observed data, as a function of some relevant parameters

$$\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{\psi} \sim p(\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{\psi}) = \prod_{i=1}^{n} p(y_i \mid \boldsymbol{\theta}, \boldsymbol{\psi})$$

- Often (in fact for a surprisingly large range of models!), we can assume that the parameters are described by a **Gaussian Markov Random Field** (GMRF)

$$\boldsymbol{\theta} \mid \boldsymbol{\psi} \sim \text{Normal}(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\psi}))$$
$$\theta_l \perp\!\!\!\perp \theta_m \mid \boldsymbol{\theta}_{-lm}$$

  where
  - The notation "$-lm$" indicates all the other elements of the parameters vector, excluding elements l and m
  - The covariance matrix $\boldsymbol{\Sigma}$ depends on some hyper-parameters $\boldsymbol{\psi}$

- This kind of models is often referred to as **Latent Gaussian models**

LGMs as a general framework

- In general, we can partition $\boldsymbol{\psi} = (\boldsymbol{\psi}_1, \boldsymbol{\psi}_2)$ and re-express a LGM as

$$
\begin{aligned}
\boldsymbol{\psi} &\sim p(\boldsymbol{\psi}) && \text{(“hyperprior”)} \\
\boldsymbol{\theta} \mid \boldsymbol{\psi} &\sim p(\boldsymbol{\theta} \mid \boldsymbol{\psi}) = \text{Normal}(0, \boldsymbol{\Sigma}(\boldsymbol{\psi}_1)) && \text{(“GMRF prior”)} \\
\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{\psi} &\sim \prod_i p(y_i \mid \boldsymbol{\theta}, \boldsymbol{\psi}_2) && \text{(“data model”)}
\end{aligned}
$$

ie $\boldsymbol{\psi}_1$ are the **hyper-parameters** and $\boldsymbol{\psi}_2$ are **nuisance parameters**

LGMs as a general framework

- In general, we can partition $\boldsymbol{\psi} = (\boldsymbol{\psi}_1, \boldsymbol{\psi}_2)$ and re-express a LGM as

$$
\begin{aligned}
\boldsymbol{\psi} &\sim p(\boldsymbol{\psi}) && \text{("hyperprior")} \\
\boldsymbol{\theta} \mid \boldsymbol{\psi} &\sim p(\boldsymbol{\theta} \mid \boldsymbol{\psi}) = \text{Normal}(0, \boldsymbol{\Sigma}(\boldsymbol{\psi}_1)) && \text{("GMRF prior")} \\
\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{\psi} &\sim \prod_i p(y_i \mid \boldsymbol{\theta}, \boldsymbol{\psi}_2) && \text{("data model")}
\end{aligned}
$$

  ie $\boldsymbol{\psi}_1$ are the **hyper-parameters** and $\boldsymbol{\psi}_2$ are **nuisance parameters**

- The dimension of $\boldsymbol{\theta}$ can be very large (eg 10^2-10^5)
- Conversely, because of the conditional independence properties, the dimension of $\boldsymbol{\psi}$ is generally small (eg 1-5)

LGMs as a general framework

- A very general way of specifying the problem is by modelling the mean for the i-th unit by means of an additive linear predictor, defined on a suitable scale (e.g. logistic for binomial data)

$$\eta_i = \alpha + \sum_{m=1}^{M} \beta_m x_{mi} + \sum_{l=1}^{L} f_l(z_{li})$$

where

  - α is the intercept;
  - $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_M)$ quantify the effect of $\boldsymbol{x} = (x_1, \ldots, x_M)$ on the response;
  - $\boldsymbol{f} = \{f_1(\cdot), \ldots, f_L(\cdot)\}$ is a set of functions defined in terms of some covariates $\boldsymbol{z} = (z_1, \ldots, z_L)$

and then assume

$$\boldsymbol{\theta} = (\alpha, \boldsymbol{\beta}, \boldsymbol{f}) \sim \text{GMRF}(\boldsymbol{\psi})$$

LGMs as a general framework

- A very general way of specifying the problem is by modelling the mean for the i-th unit by means of an additive linear predictor, defined on a suitable scale (e.g. logistic for binomial data)

$$\eta_i = \alpha + \sum_{m=1}^{M} \beta_m x_{mi} + \sum_{l=1}^{L} f_l(z_{li})$$

where

  - α is the intercept;
  - $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_M)$ quantify the effect of $\boldsymbol{x} = (x_1, \ldots, x_M)$ on the response;
  - $\boldsymbol{f} = \{f_1(\cdot), \ldots, f_L(\cdot)\}$ is a set of functions defined in terms of some covariates $\boldsymbol{z} = (z_1, \ldots, z_L)$

and then assume

$$\boldsymbol{\theta} = (\alpha, \boldsymbol{\beta}, \boldsymbol{f}) \sim \text{GMRF}(\boldsymbol{\psi})$$

- **NB**: This of course implies some form of Normally-distributed marginals for $\alpha, \boldsymbol{\beta}$ and $\boldsymbol{f}$

LGMs as a general framework — examples

Upon varying the form of the functions $f_l(\cdot)$, this formulation can accommodate a wide range of models

LGMs as a general framework — examples

Upon varying the form of the functions $f_l(\cdot)$, this formulation can accommodate a wide range of models

- Standard regression
  - $f_l(\cdot) = \text{NULL}$

LGMs as a general framework — examples

Upon varying the form of the functions $f_l(\cdot)$, this formulation can accommodate a wide range of models

- Standard regression
  - $f_l(\cdot) = \text{NULL}$
- Hierarchical models
  - $f_l(\cdot) \sim \text{Normal}(0, \sigma_f^2)$ (Exchangeable)
    $\sigma_f^2 \mid \boldsymbol{\psi} \sim$ some common distribution

LGMs as a general framework — examples

Upon varying the form of the functions $f_l(\cdot)$, this formulation can accommodate a wide range of models

- Standard regression
  - $f_l(\cdot) = \text{NULL}$
- Hierarchical models
  - $f_l(\cdot) \sim \text{Normal}(0, \sigma_f^2)$ (Exchangeable)
    $\sigma_f^2 \mid \boldsymbol{\psi} \sim$ some common distribution
- Spatial and spatio-temporal models
  - Two components: $f_1(\cdot) \sim \text{CAR}$ (Spatially structured effects)
    $f_2(\cdot) \sim \text{Normal}(0, \sigma_{f_2}^2)$ (Unstructured residual)

LGMs as a general framework — examples

Upon varying the form of the functions $f_l(\cdot)$, this formulation can accommodate a wide range of models

- Standard regression
  - $f_l(\cdot) = \text{NULL}$
- Hierarchical models
  - $f_l(\cdot) \sim \text{Normal}(0, \sigma_f^2)$ (Exchangeable)
    $\sigma_f^2 \mid \boldsymbol{\psi} \sim$ some common distribution
- Spatial and spatio-temporal models
  - Two components: $f_1(\cdot) \sim \text{CAR}$ (Spatially structured effects)
    $f_2(\cdot) \sim \text{Normal}(0, \sigma_{f_2}^2)$ (Unstructured residual)
- Spline smoothing
  - $f_l(\cdot) \sim \text{AR}(\phi, \sigma_\varepsilon^2)$

LGMs as a general framework — examples

Upon varying the form of the functions $f_l(\cdot)$, this formulation can accommodate a wide range of models

- Standard regression
  - $f_l(\cdot) = \text{NULL}$
- Hierarchical models
  - $f_l(\cdot) \sim \text{Normal}(0, \sigma_f^2)$ (Exchangeable)
    $\sigma_f^2 \mid \boldsymbol{\psi} \sim$ some common distribution
- Spatial and spatio-temporal models
  - Two components: $f_1(\cdot) \sim \text{CAR}$ (Spatially structured effects)
    $f_2(\cdot) \sim \text{Normal}(0, \sigma_{f_2}^2)$ (Unstructured residual)
- Spline smoothing
  - $f_l(\cdot) \sim \text{AR}(\phi, \sigma_\varepsilon^2)$
- Survival models / logGaussian Cox Processes
  - More complex specification in theory, but relatively easy to fit within the INLA framework!

Gaussian Markov Random Field

In order to preserve the underlying conditional independence structure in a GMRF, it is necessary to constrain the parameterisation

Gaussian Markov Random Field

In order to preserve the underlying conditional independence structure in a GMRF, it is necessary to constrain the parameterisation

- Generally, it is complicated to do it in terms of the covariance matrix $\boldsymbol{\Sigma}$
  - Typically, $\boldsymbol{\Sigma}$ is dense (ie many of the entries are non-zero)
  - If it happens to be sparse, this implies (marginal) independence among the relevant elements of $\boldsymbol{\theta}$ — this is generally too stringent a requirement!

Gaussian Markov Random Field

In order to preserve the underlying conditional independence structure in a GMRF, it is necessary to constrain the parameterisation

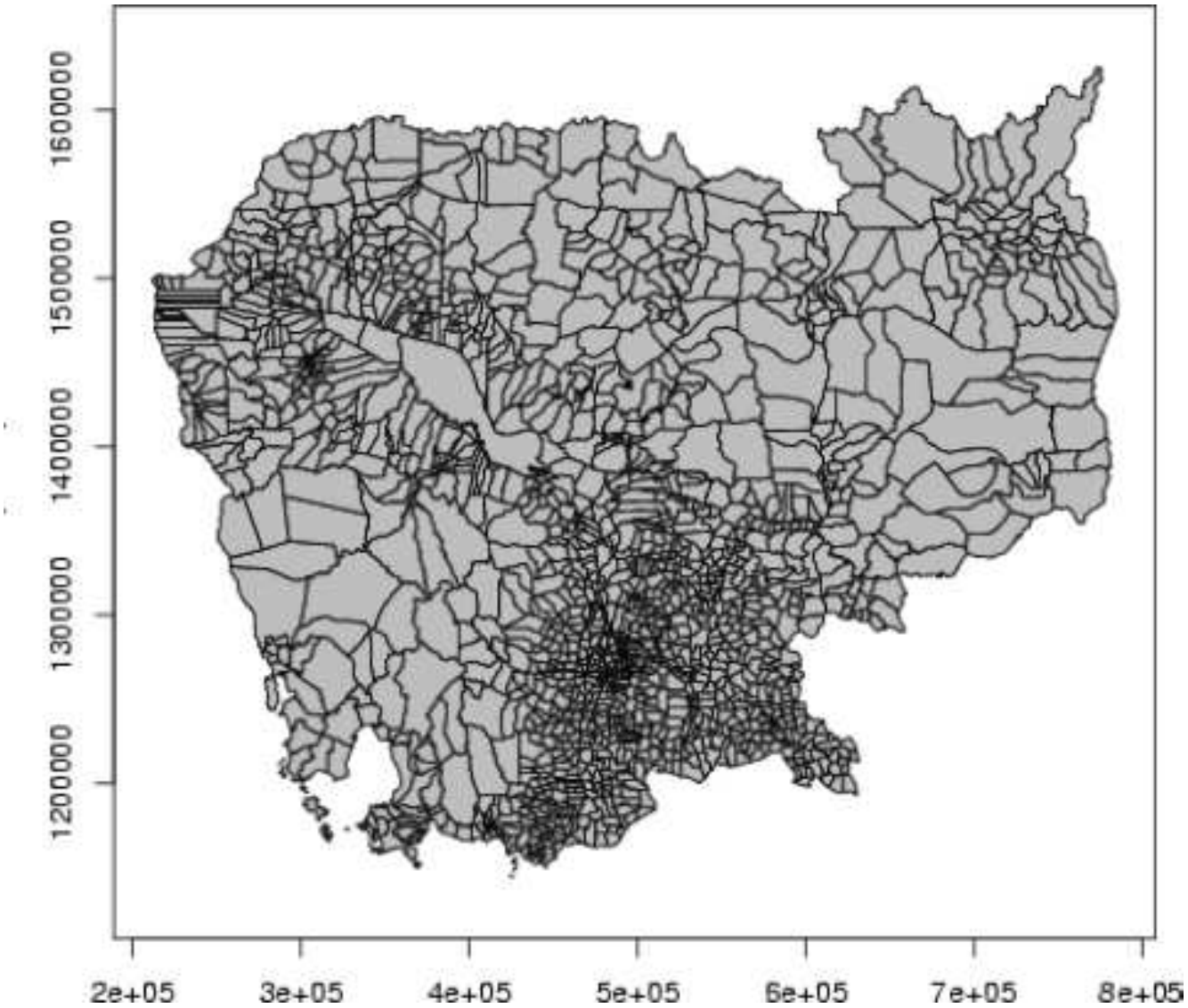
- Generally, it is complicated to do it in terms of the covariance matrix $\boldsymbol{\Sigma}$
  - Typically, $\boldsymbol{\Sigma}$ is dense (ie many of the entries are non-zero)
  - If it happens to be sparse, this implies (marginal) independence among the relevant elements of $\boldsymbol{\theta}$ — this is generally too stringent a requirement!

- Conversely, it is much simpler when using the precision matrix $\boldsymbol{Q} =: \boldsymbol{\Sigma}^{-1}$
  - As it turns out, it can be shown that

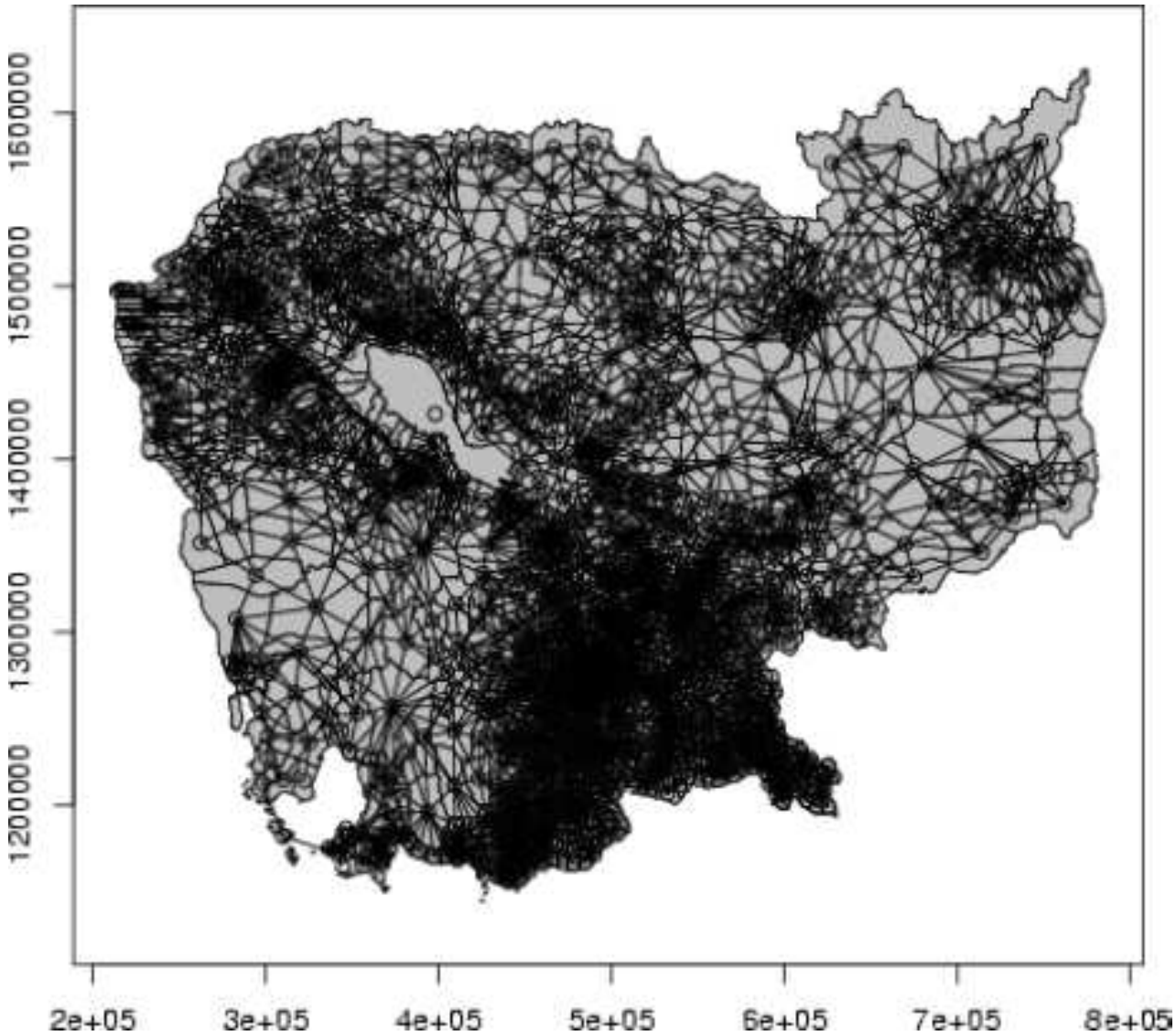
$$\theta_l \perp\!\!\!\perp \theta_m \mid \boldsymbol{\theta}_{-lm} \Leftrightarrow \boldsymbol{Q}_{lm} = 0$$

  - Thus, under **conditional** independence (which is a less restrictive constraint), the precision matrix is typically **sparse**
  - We can use existing numerical methods to deal with sparse matrices (eg the `R` package `Matrix`)
  - Most computations in GMRFs are performed in terms of computing Cholesky's factorisations

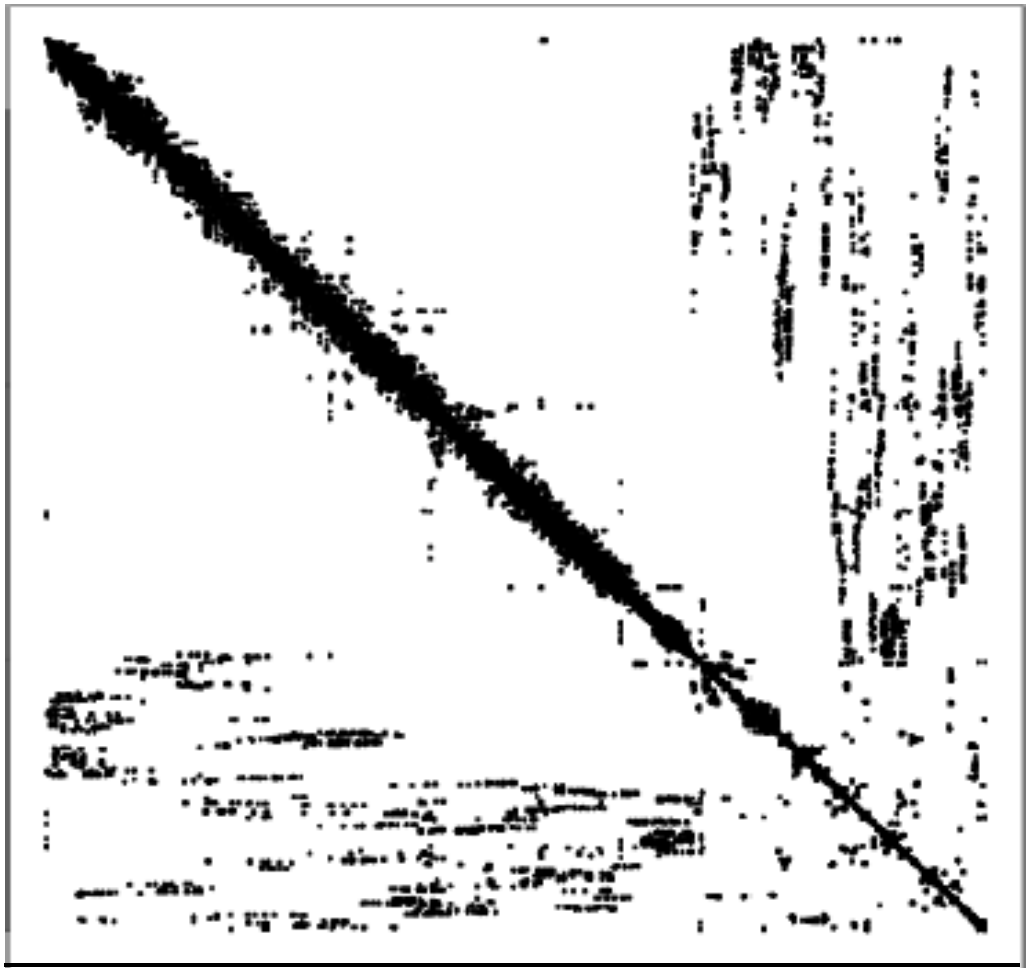
Precision matrix and conditional independence



Precision matrix and conditional independence

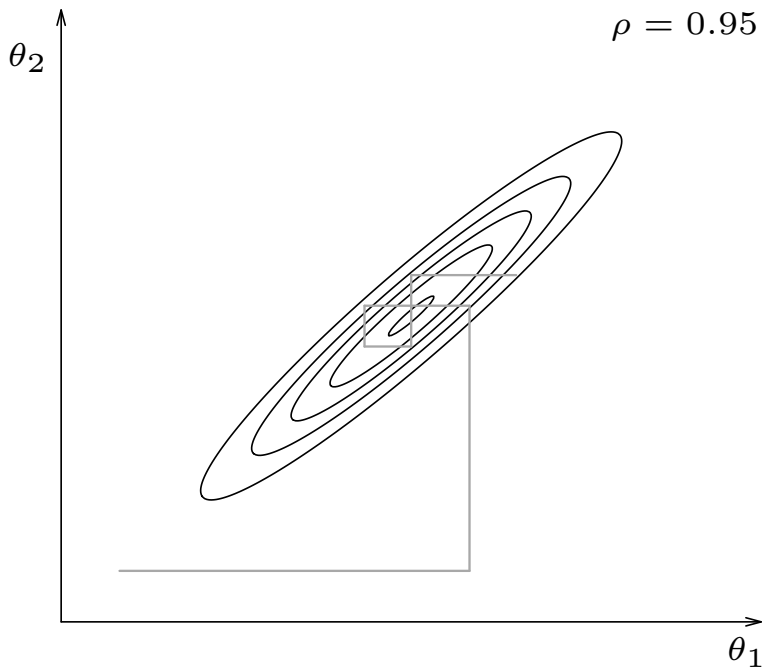


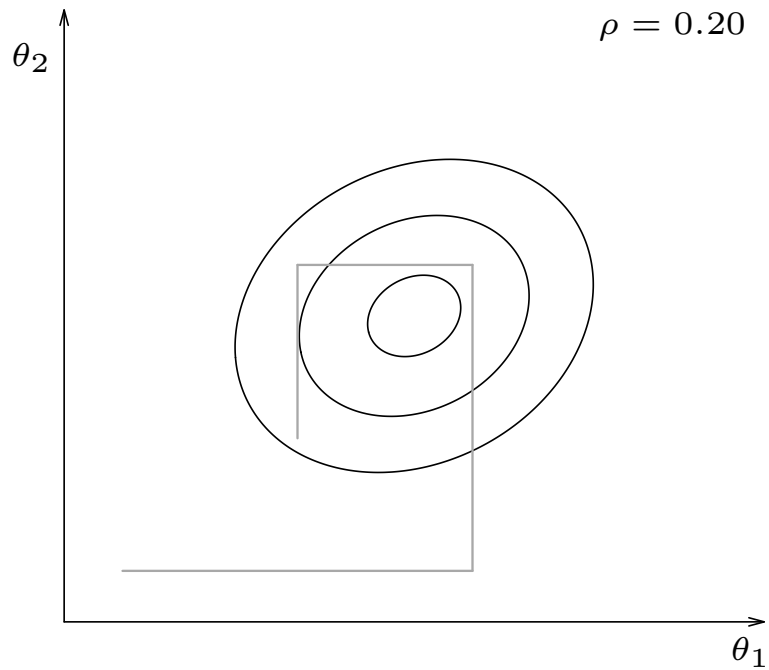
Precision matrix and conditional independence



MCMC and LGMs

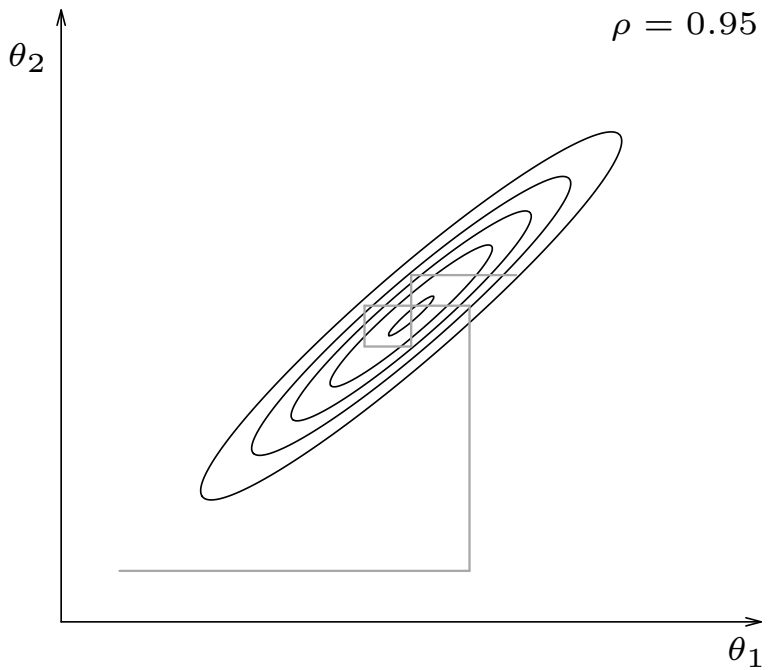
- (Standard) MCMC methods tend to perform poorly when applied to (non-trivial) LGMs. This is due to several factors
  - The components of the latent Gaussian field $\boldsymbol{\theta}$ tend to be highly correlated, thus impacting on convergence and autocorrelation
  - Especially when the number of observations is large, $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ also tend to be highly correlated

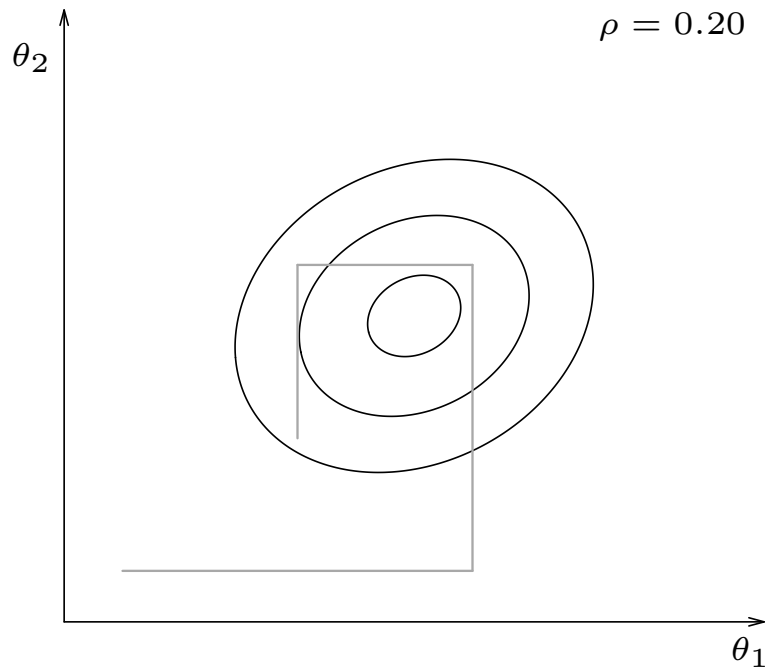




MCMC and LGMs

- (Standard) MCMC methods tend to perform poorly when applied to (non-trivial) LGMs. This is due to several factors
  - The components of the latent Gaussian field $\boldsymbol{\theta}$ tend to be highly correlated, thus impacting on convergence and autocorrelation
  - Especially when the number of observations is large, $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ also tend to be highly correlated





- Again, blocking and overparameterisation can **alleviate**, but rarely eliminate the problem

Summary so far

- Bayesian computation (especially for LGMs) is in general difficult
- MCMC can be efficiently used in many simple cases, but becomes a bit trickier for complex models
  - Issues with convergence
  - Time to run can be very long

Summary so far

- Bayesian computation (especially for LGMs) is in general difficult
- MCMC can be efficiently used in many simple cases, but becomes a bit trickier for complex models
  - Issues with convergence
  - Time to run can be very long

- A wide class of statistical models can be represented in terms of LGM
- This allows us to take advantage of nice computational properties
  - GMRFs
  - Sparse precision matrices

- This is at the heart of the INLA approach

Introduction to INLA

- The starting point to understand the INLA approach is the definition of conditional probability, which holds for any pair of variables (x, z) — and, technically, provided $p(z) > 0$

$$p(x \mid z) =: \frac{p(x, z)}{p(z)}$$

which can be re-written as

$$p(z) = \frac{p(x, z)}{p(x \mid z)}$$

- The starting point to understand the INLA approach is the definition of conditional probability, which holds for any pair of variables (x, z) — and, technically, provided $p(z) > 0$

$$p(x \mid z) =: \frac{p(x, z)}{p(z)}$$

which can be re-written as

$$p(z) = \frac{p(x, z)}{p(x \mid z)}$$

- In particular, a conditional version can be obtained further considering a third variable w as

$$p(z \mid w) = \frac{p(x, z \mid w)}{p(x \mid z, w)}$$

which is particularly relevant to the Bayesian case

Integrated Nested Laplace Approximation (INLA)

- The second "ingredient" is **Laplace approximation**

- The second "ingredient" is **Laplace approximation**
- Main idea: approximate $\log g(x)$ using a quadratic function by means of a Taylor's series expansion around the mode $\hat{x}$

$$\begin{aligned}\log g(x) &\approx \log g(\hat{x}) + \frac{\partial \log g(\hat{x})}{\partial x}(x-\hat{x}) + \frac{1}{2}\frac{\partial^2 \log g(\hat{x})}{\partial x^2}(x-\hat{x})^2 \\ &= \log g(\hat{x}) + \frac{1}{2}\frac{\partial^2 \log g(\hat{x})}{\partial x^2}(x-\hat{x})^2 \qquad \left(\text{since } \frac{\partial \log g(\hat{x})}{\partial x} = 0\right)\end{aligned}$$

- The second "ingredient" is **Laplace approximation**
- Main idea: approximate $\log g(x)$ using a quadratic function by means of a Taylor's series expansion around the mode $\hat{x}$

$$\begin{aligned}\log g(x) &\approx \log g(\hat{x}) + \frac{\partial \log g(\hat{x})}{\partial x}(x-\hat{x}) + \frac{1}{2}\frac{\partial^2 \log g(\hat{x})}{\partial x^2}(x-\hat{x})^2 \\ &= \log g(\hat{x}) + \frac{1}{2}\frac{\partial^2 \log g(\hat{x})}{\partial x^2}(x-\hat{x})^2 \qquad \left(\text{since } \frac{\partial \log g(\hat{x})}{\partial x} = 0\right)\end{aligned}$$

- Setting $\hat{\sigma}^2 = -1 \Big/ \frac{\partial^2 \log g(\hat{x})}{\partial x^2}$ we can re-write

$$\log g(x) \approx \log g(\hat{x}) - \frac{1}{2\hat{\sigma}^2}(x-\hat{x})^2$$

or equivalently

$$\int g(x)dx = \int e^{\log g(x)}dx \approx \text{const} \int \exp\left[-\frac{(x-\hat{x})^2}{2\hat{\sigma}^2}\right]dx$$

- The second "ingredient" is **Laplace approximation**
- Main idea: approximate $\log g(x)$ using a quadratic function by means of a Taylor's series expansion around the mode $\hat{x}$

$$\begin{aligned}\log g(x) &\approx \log g(\hat{x}) + \frac{\partial \log g(\hat{x})}{\partial x}(x-\hat{x}) + \frac{1}{2}\frac{\partial^2 \log g(\hat{x})}{\partial x^2}(x-\hat{x})^2 \\ &= \log g(\hat{x}) + \frac{1}{2}\frac{\partial^2 \log g(\hat{x})}{\partial x^2}(x-\hat{x})^2 \qquad \left(\text{since } \frac{\partial \log g(\hat{x})}{\partial x} = 0\right)\end{aligned}$$

- Setting $\hat{\sigma}^2 = -1 \Big/ \frac{\partial^2 \log g(\hat{x})}{\partial x^2}$ we can re-write

$$\log g(x) \approx \log g(\hat{x}) - \frac{1}{2\hat{\sigma}^2}(x-\hat{x})^2$$

or equivalently

$$\int g(x)dx = \int e^{\log g(x)}dx \approx \text{const} \int \exp\left[-\frac{(x-\hat{x})^2}{2\hat{\sigma}^2}\right]dx$$

- Thus, under LA, $g(x) \approx \text{Normal}(\hat{x}, \hat{\sigma}^2)$

Laplace approximation — example

- Consider a χ^2 distribution: $p(x) = \frac{g(x)}{c} = \frac{x^{\frac{k}{2}-1} e^{\frac{-x}{2}}}{c}$

Laplace approximation — example

- Consider a χ^2 distribution: $p(x) = \frac{g(x)}{c} = \frac{x^{\frac{k}{2}-1} e^{\frac{-x}{2}}}{c}$
  1. $l(x) = \log g(x) = \left(\frac{k}{2} - 1\right) \log x - \frac{x}{2}$

Laplace approximation — example

- Consider a χ^2 distribution: $p(x) = \frac{g(x)}{c} = \frac{x^{\frac{k}{2}-1} e^{\frac{-x}{2}}}{c}$
  1. $l(x) = \log g(x) = \left(\frac{k}{2} - 1\right) \log x - \frac{x}{2}$
  2. $l'(x) = \frac{\partial \log g(x)}{\partial x} = \left(\frac{k}{2} - 1\right) x^{-1} - \frac{1}{2}$

Laplace approximation — example

- Consider a χ^2 distribution: $p(x) = \frac{g(x)}{c} = \frac{x^{\frac{k}{2}-1} e^{\frac{-x}{2}}}{c}$
  1. $l(x) = \log g(x) = \left(\frac{k}{2} - 1\right) \log x - \frac{x}{2}$
  2. $l'(x) = \frac{\partial \log g(x)}{\partial x} = \left(\frac{k}{2} - 1\right) x^{-1} - \frac{1}{2}$
  3. $l''(x) = \frac{\partial^2 \log g(x)}{\partial x^2} = -\left(\frac{k}{2} - 1\right) x^{-2}$

Laplace approximation — example

- Consider a χ^2 distribution: $p(x) = \frac{g(x)}{c} = \frac{x^{\frac{k}{2}-1} e^{\frac{-x}{2}}}{c}$
  1. $l(x) = \log g(x) = \left(\frac{k}{2} - 1\right) \log x - \frac{x}{2}$
  2. $l'(x) = \frac{\partial \log g(x)}{\partial x} = \left(\frac{k}{2} - 1\right) x^{-1} - \frac{1}{2}$
  3. $l''(x) = \frac{\partial^2 \log g(x)}{\partial x^2} = -\left(\frac{k}{2} - 1\right) x^{-2}$

- Then
  - Solving $l'(x) = 0$ we find the mode: $\hat{x} = k - 2$
  - Evaluating $-\frac{1}{l''(x)}$ at the mode gives $\hat{\sigma}^2 = 2(k-2)$

Laplace approximation — example

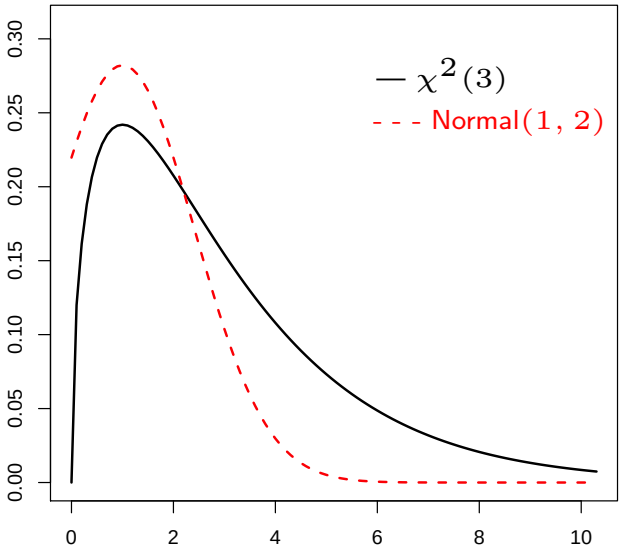
- Consider a χ^2 distribution: $p(x) = \frac{g(x)}{c} = \frac{x^{\frac{k}{2}-1} e^{\frac{-x}{2}}}{c}$
  1. $l(x) = \log g(x) = \left(\frac{k}{2} - 1\right) \log x - \frac{x}{2}$
  2. $l'(x) = \frac{\partial \log g(x)}{\partial x} = \left(\frac{k}{2} - 1\right) x^{-1} - \frac{1}{2}$
  3. $l''(x) = \frac{\partial^2 \log g(x)}{\partial x^2} = -\left(\frac{k}{2} - 1\right) x^{-2}$

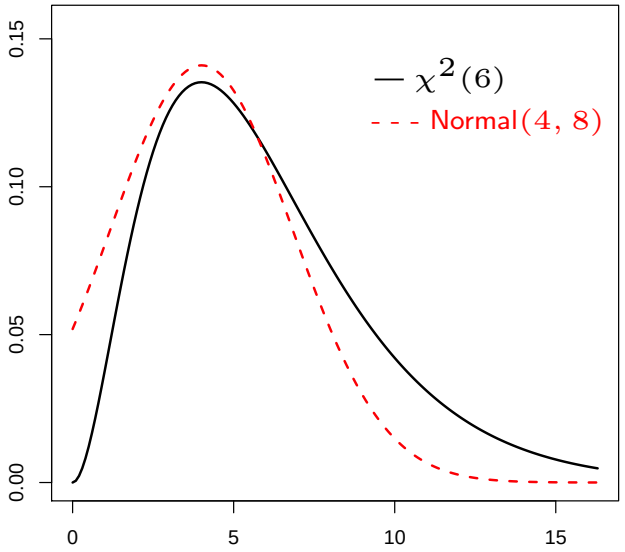
- Then
  - Solving $l'(x) = 0$ we find the mode: $\hat{x} = k - 2$
  - Evaluating $-\frac{1}{l''(x)}$ at the mode gives $\hat{\sigma}^2 = 2(k-2)$

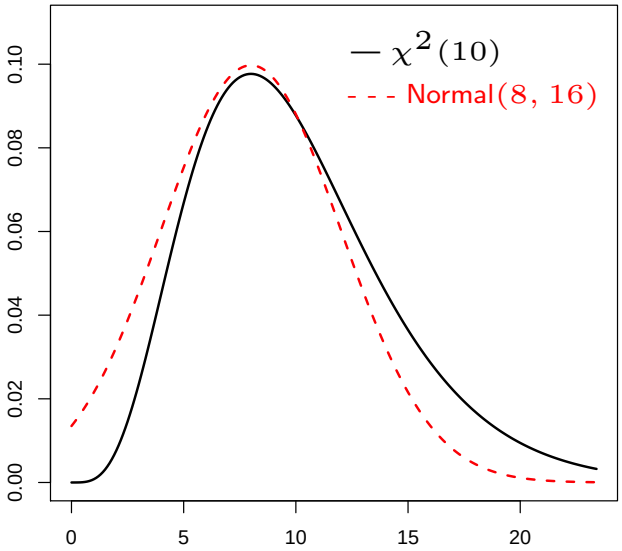
- Consequently, we can approximate $p(x)$ as

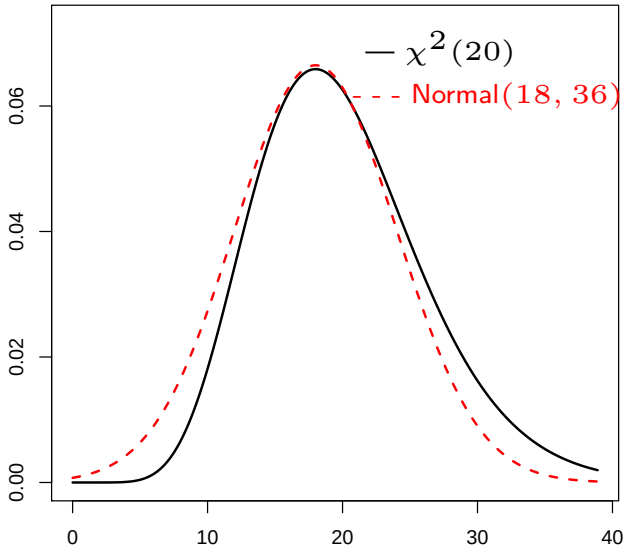
$$p(x) \approx \tilde{p}(x) = \text{Normal}(k-2, 2(k-2))$$

Laplace approximation — example









- The general idea is that using the fundamental probability equations, we can approximate a generic conditional (posterior) distribution as

$$\tilde{p}(z \mid w) = \frac{p(x, z \mid w)}{\tilde{p}(x \mid z, w)},$$

where $\tilde{p}(x \mid z, w)$ is the Laplace approximation to the conditional distribution of x given z, w

- The general idea is that using the fundamental probability equations, we can approximate a generic conditional (posterior) distribution as

$$\tilde{p}(z \mid w) = \frac{p(x, z \mid w)}{\tilde{p}(x \mid z, w)},$$

  where $\tilde{p}(x \mid z, w)$ is the Laplace approximation to the conditional distribution of x given z, w

- This idea can be used to approximate any generic required posterior distribution

Objective of Bayesian estimation

- In a Bayesian LGM, the required distributions are

$$p(\theta_j \mid \boldsymbol{y}) = \int p(\theta_j, \boldsymbol{\psi} \mid \boldsymbol{y}) d\boldsymbol{\psi} = \int p(\boldsymbol{\psi} \mid \boldsymbol{y}) p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y}) d\boldsymbol{\psi}$$

$$p(\psi_k \mid \boldsymbol{y}) = \int p(\boldsymbol{\psi} \mid \boldsymbol{y}) d\boldsymbol{\psi}_{-k}$$

Objective of Bayesian estimation

- In a Bayesian LGM, the required distributions are

$$p(\theta_j \mid \boldsymbol{y}) = \int p(\theta_j, \boldsymbol{\psi} \mid \boldsymbol{y}) d\boldsymbol{\psi} = \int p(\boldsymbol{\psi} \mid \boldsymbol{y}) p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y}) d\boldsymbol{\psi}$$

$$p(\psi_k \mid \boldsymbol{y}) = \int p(\boldsymbol{\psi} \mid \boldsymbol{y}) d\boldsymbol{\psi}_{-k}$$

- Thus we need to estimate:

  (1.) $p(\boldsymbol{\psi} \mid \boldsymbol{y})$, from which also all the relevant marginals $p(\psi_k \mid \boldsymbol{y})$ can be obtained;

Objective of Bayesian estimation

- In a Bayesian LGM, the required distributions are

$$p(\theta_j \mid \boldsymbol{y}) = \int p(\theta_j, \boldsymbol{\psi} \mid \boldsymbol{y}) d\boldsymbol{\psi} = \int p(\boldsymbol{\psi} \mid \boldsymbol{y}) p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y}) d\boldsymbol{\psi}$$

$$p(\psi_k \mid \boldsymbol{y}) = \int p(\boldsymbol{\psi} \mid \boldsymbol{y}) d\boldsymbol{\psi}_{-k}$$

- Thus we need to estimate:

  (1.) $p(\boldsymbol{\psi} \mid \boldsymbol{y})$, from which also all the relevant marginals $p(\psi_k \mid \boldsymbol{y})$ can be obtained;

  (2.) $p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y})$, which is needed to compute the marginal posterior for the parameters

(1.) can be easily estimated as

$$p(\boldsymbol{\psi} \mid \boldsymbol{y}) \quad = \quad \frac{p(\boldsymbol{\theta}, \boldsymbol{\psi} \mid \boldsymbol{y})}{p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})}$$

(1.) can be easily estimated as

$$
\begin{aligned}
p(\boldsymbol{\psi} \mid \boldsymbol{y}) &= \frac{p(\boldsymbol{\theta}, \boldsymbol{\psi} \mid \boldsymbol{y})}{p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})} \\
&= \frac{p(\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{\psi}) p(\boldsymbol{\theta}, \boldsymbol{\psi})}{p(\boldsymbol{y})} \frac{1}{p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})}
\end{aligned}
$$

(1.) can be easily estimated as

$$
\begin{aligned}
p(\boldsymbol{\psi} \mid \boldsymbol{y}) &= \frac{p(\boldsymbol{\theta}, \boldsymbol{\psi} \mid \boldsymbol{y})}{p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})} \\
&= \frac{p(\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{\psi}) p(\boldsymbol{\theta}, \boldsymbol{\psi})}{p(\boldsymbol{y})} \frac{1}{p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})} \\
&= \frac{p(\boldsymbol{y} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\psi}) p(\boldsymbol{\psi})}{p(\boldsymbol{y})} \frac{1}{p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})}
\end{aligned}
$$

(1.) can be easily estimated as

$$
\begin{aligned}
p(\boldsymbol{\psi} \mid \boldsymbol{y}) &= \frac{p(\boldsymbol{\theta}, \boldsymbol{\psi} \mid \boldsymbol{y})}{p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})} \\
&= \frac{p(\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{\psi}) p(\boldsymbol{\theta}, \boldsymbol{\psi})}{p(\boldsymbol{y})} \frac{1}{p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})} \\
&= \frac{p(\boldsymbol{y} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\psi}) p(\boldsymbol{\psi})}{p(\boldsymbol{y})} \frac{1}{p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})} \\
&\propto \frac{p(\boldsymbol{\psi}) p(\boldsymbol{\theta} \mid \boldsymbol{\psi}) p(\boldsymbol{y} \mid \boldsymbol{\theta})}{p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})}
\end{aligned}
$$

(1.) can be easily estimated as

$$
\begin{aligned}
p(\boldsymbol{\psi} \mid \boldsymbol{y}) &= \frac{p(\boldsymbol{\theta}, \boldsymbol{\psi} \mid \boldsymbol{y})}{p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})} \\
&= \frac{p(\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{\psi}) p(\boldsymbol{\theta}, \boldsymbol{\psi})}{p(\boldsymbol{y})} \frac{1}{p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})} \\
&= \frac{p(\boldsymbol{y} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\psi}) p(\boldsymbol{\psi})}{p(\boldsymbol{y})} \frac{1}{p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})} \\
&\propto \frac{p(\boldsymbol{\psi}) p(\boldsymbol{\theta} \mid \boldsymbol{\psi}) p(\boldsymbol{y} \mid \boldsymbol{\theta})}{p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})} \\
&\approx \left. \frac{p(\boldsymbol{\psi}) p(\boldsymbol{\theta} \mid \boldsymbol{\psi}) p(\boldsymbol{y} \mid \boldsymbol{\theta})}{\tilde{p}(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})} \right|_{\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}(\boldsymbol{\psi})} =: \tilde{p}(\boldsymbol{\psi} \mid \boldsymbol{y})
\end{aligned}
$$

where

- $\tilde{p}(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})$ is the Laplace approximation of $p(\boldsymbol{\theta} \mid \boldsymbol{\psi}, \boldsymbol{y})$
- $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}(\boldsymbol{\psi})$ is its mode

(2.) is slightly more complex, because in general there will be more elements in $\boldsymbol{\theta}$ than there are in $\boldsymbol{\psi}$ and thus this computation is more expensive

(2.) is slightly more complex, because in general there will be more elements in $\boldsymbol{\theta}$ than there are in $\boldsymbol{\psi}$ and thus this computation is more expensive

- One easy possibility is to approximate $p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y})$ directly using a Normal distribution, where the precision matrix is based on the Cholesky decomposition of the precision matrix $\boldsymbol{Q}$. While this is very fast, the approximation is generally not very good

(2.) is slightly more complex, because in general there will be more elements in $\boldsymbol{\theta}$ than there are in $\boldsymbol{\psi}$ and thus this computation is more expensive

- One easy possibility is to approximate $p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y})$ directly using a Normal distribution, where the precision matrix is based on the Cholesky decomposition of the precision matrix $\boldsymbol{Q}$. While this is very fast, the approximation is generally not very good

- Alternatively, we can write $\boldsymbol{\theta} = \{\theta_j, \boldsymbol{\theta}_{-j}\}$, use the definition of conditional probability and again Laplace approximation to obtain

$$p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y}) = \frac{p\left(\{\theta_j, \boldsymbol{\theta}_{-j}\} \mid \boldsymbol{\psi}, \boldsymbol{y}\right)}{p(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})}$$

(2.) is slightly more complex, because in general there will be more elements in $\boldsymbol{\theta}$ than there are in $\boldsymbol{\psi}$ and thus this computation is more expensive

- One easy possibility is to approximate $p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y})$ directly using a Normal distribution, where the precision matrix is based on the Cholesky decomposition of the precision matrix $\boldsymbol{Q}$. While this is very fast, the approximation is generally not very good

- Alternatively, we can write $\boldsymbol{\theta} = \{\theta_j, \boldsymbol{\theta}_{-j}\}$, use the definition of conditional probability and again Laplace approximation to obtain

$$p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y}) = \frac{p\left(\{\theta_j, \boldsymbol{\theta}_{-j}\} \mid \boldsymbol{\psi}, \boldsymbol{y}\right)}{p(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})} = \frac{p\left(\{\theta_j, \boldsymbol{\theta}_{-j}\}, \boldsymbol{\psi} \mid \boldsymbol{y}\right)}{p(\boldsymbol{\psi} \mid \boldsymbol{y})} \frac{1}{p(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})}$$

(2.) is slightly more complex, because in general there will be more elements in $\boldsymbol{\theta}$ than there are in $\boldsymbol{\psi}$ and thus this computation is more expensive

- One easy possibility is to approximate $p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y})$ directly using a Normal distribution, where the precision matrix is based on the Cholesky decomposition of the precision matrix $\boldsymbol{Q}$. While this is very fast, the approximation is generally not very good

- Alternatively, we can write $\boldsymbol{\theta} = \{\theta_j, \boldsymbol{\theta}_{-j}\}$, use the definition of conditional probability and again Laplace approximation to obtain

$$
\begin{aligned}
p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y}) &= \frac{p\left(\{\theta_j, \boldsymbol{\theta}_{-j}\} \mid \boldsymbol{\psi}, \boldsymbol{y}\right)}{p(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})} = \frac{p\left(\{\theta_j, \boldsymbol{\theta}_{-j}\}, \boldsymbol{\psi} \mid \boldsymbol{y}\right)}{p(\boldsymbol{\psi} \mid \boldsymbol{y})} \frac{1}{p(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})} \\
&\propto \frac{p\left(\boldsymbol{\theta}, \boldsymbol{\psi} \mid \boldsymbol{y}\right)}{p(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})}
\end{aligned}
$$

(2.) is slightly more complex, because in general there will be more elements in $\boldsymbol{\theta}$ than there are in $\boldsymbol{\psi}$ and thus this computation is more expensive

- One easy possibility is to approximate $p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y})$ directly using a Normal distribution, where the precision matrix is based on the Cholesky decomposition of the precision matrix $\boldsymbol{Q}$. While this is very fast, the approximation is generally not very good

- Alternatively, we can write $\boldsymbol{\theta} = \{\theta_j, \boldsymbol{\theta}_{-j}\}$, use the definition of conditional probability and again Laplace approximation to obtain

$$
\begin{aligned}
p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y}) &= \frac{p\left(\{\theta_j, \boldsymbol{\theta}_{-j}\} \mid \boldsymbol{\psi}, \boldsymbol{y}\right)}{p(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})} = \frac{p\left(\{\theta_j, \boldsymbol{\theta}_{-j}\}, \boldsymbol{\psi} \mid \boldsymbol{y}\right)}{p(\boldsymbol{\psi} \mid \boldsymbol{y})} \frac{1}{p(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})} \\
&\propto \frac{p\left(\boldsymbol{\theta}, \boldsymbol{\psi} \mid \boldsymbol{y}\right)}{p(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})} \quad \propto \frac{p(\boldsymbol{\psi})p(\boldsymbol{\theta} \mid \boldsymbol{\psi})p(\boldsymbol{y} \mid \boldsymbol{\theta})}{p(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})}
\end{aligned}
$$

(2.) is slightly more complex, because in general there will be more elements in $\boldsymbol{\theta}$ than there are in $\boldsymbol{\psi}$ and thus this computation is more expensive

- One easy possibility is to approximate $p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y})$ directly using a Normal distribution, where the precision matrix is based on the Cholesky decomposition of the precision matrix $\boldsymbol{Q}$. While this is very fast, the approximation is generally not very good

- Alternatively, we can write $\boldsymbol{\theta} = \{\theta_j, \boldsymbol{\theta}_{-j}\}$, use the definition of conditional probability and again Laplace approximation to obtain

$$
\begin{aligned}
p(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y}) &= \frac{p\left(\{\theta_j, \boldsymbol{\theta}_{-j}\} \mid \boldsymbol{\psi}, \boldsymbol{y}\right)}{p(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})} = \frac{p\left(\{\theta_j, \boldsymbol{\theta}_{-j}\}, \boldsymbol{\psi} \mid \boldsymbol{y}\right)}{p(\boldsymbol{\psi} \mid \boldsymbol{y})} \frac{1}{p(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})} \\
&\propto \frac{p\left(\boldsymbol{\theta}, \boldsymbol{\psi} \mid \boldsymbol{y}\right)}{p(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})} \propto \frac{p(\boldsymbol{\psi}) p(\boldsymbol{\theta} \mid \boldsymbol{\psi}) p(\boldsymbol{y} \mid \boldsymbol{\theta})}{p(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})} \\
&\approx \left. \frac{p(\boldsymbol{\psi}) p(\boldsymbol{\theta} \mid \boldsymbol{\psi}) p(\boldsymbol{y} \mid \boldsymbol{\theta})}{\tilde{p}(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})} \right|_{\boldsymbol{\theta}_{-j} = \hat{\boldsymbol{\theta}}_{-j}(\theta_j, \boldsymbol{\psi})} =: \tilde{p}(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y})
\end{aligned}
$$

- Because $(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})$ are reasonably Normal, the approximation works generally well
- However, this strategy can be computationally expensive

- Because $(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})$ are reasonably Normal, the approximation works generally well
- However, this strategy can be computationally expensive

- The most efficient algorithm is the "**Simplified Laplace Approximation**"
  - Based on a Taylor's series expansion up to the third order of both numerator and denominator for $\tilde{p}(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y})$
  - This effectively "corrects" the Gaussian approximation for location and skewness to increase the fit to the required distribution

- Because $(\boldsymbol{\theta}_{-j} \mid \theta_j, \boldsymbol{\psi}, \boldsymbol{y})$ are reasonably Normal, the approximation works generally well
- However, this strategy can be computationally expensive

- The most efficient algorithm is the "**Simplified Laplace Approximation**"
  - Based on a Taylor's series expansion up to the third order of both numerator and denominator for $\tilde{p}(\theta_j \mid \boldsymbol{\psi}, \boldsymbol{y})$
  - This effectively "corrects" the Gaussian approximation for location and skewness to increase the fit to the required distribution

- This is the algorithm implemented by default by `R-INLA`, but this choice can be modified
  - If extra precision is required, it is possible to run the full Laplace approximation — of course at the expense of running time!

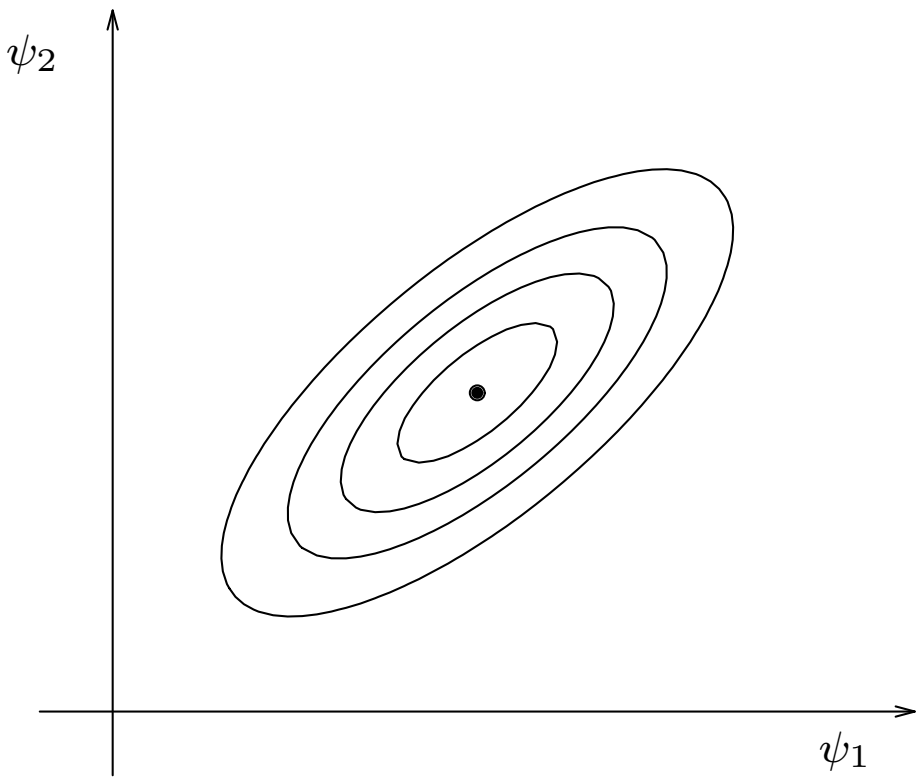
Integrated Nested Laplace Approximation (INLA)

Operationally, the INLA algorithm proceeds with the following steps:

i. Explore the marginal joint posterior for the hyper-parameters $\tilde{p}(\boldsymbol{\psi} \mid \boldsymbol{y})$

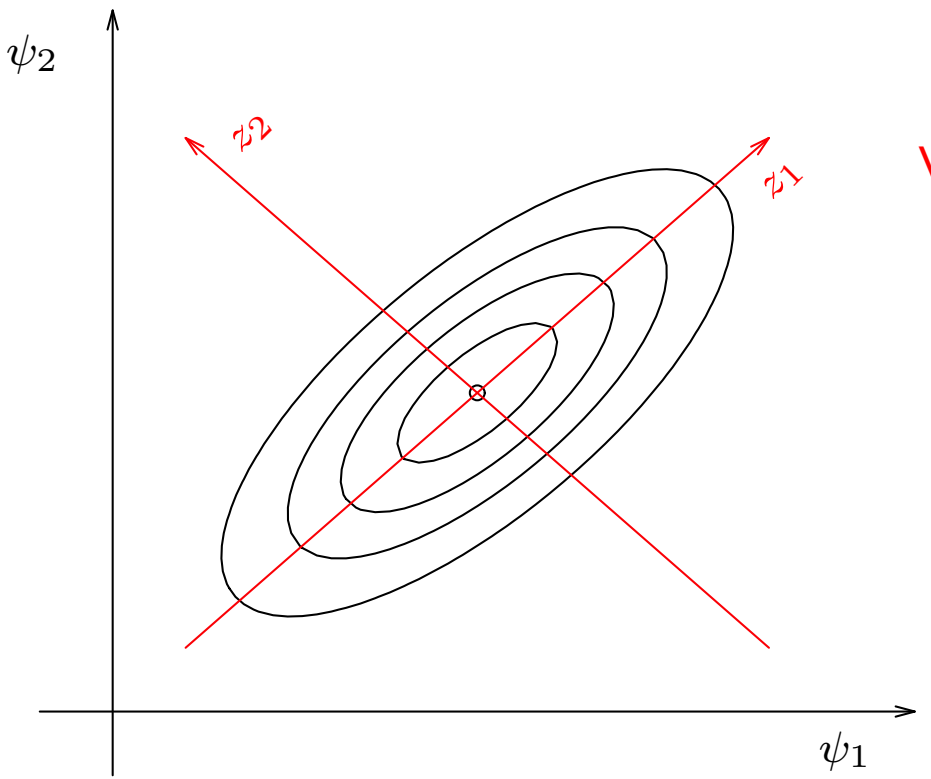
Operationally, the INLA algorithm proceeds with the following steps:

i. Explore the marginal joint posterior for the hyper-parameters $\tilde{p}(\boldsymbol{\psi} \mid \boldsymbol{y})$
   - Locate the mode $\hat{\boldsymbol{\psi}}$ by optimising $\log \tilde{p}(\boldsymbol{\psi} \mid \boldsymbol{y})$, eg using Newton-like algorithms



Operationally, the INLA algorithm proceeds with the following steps:

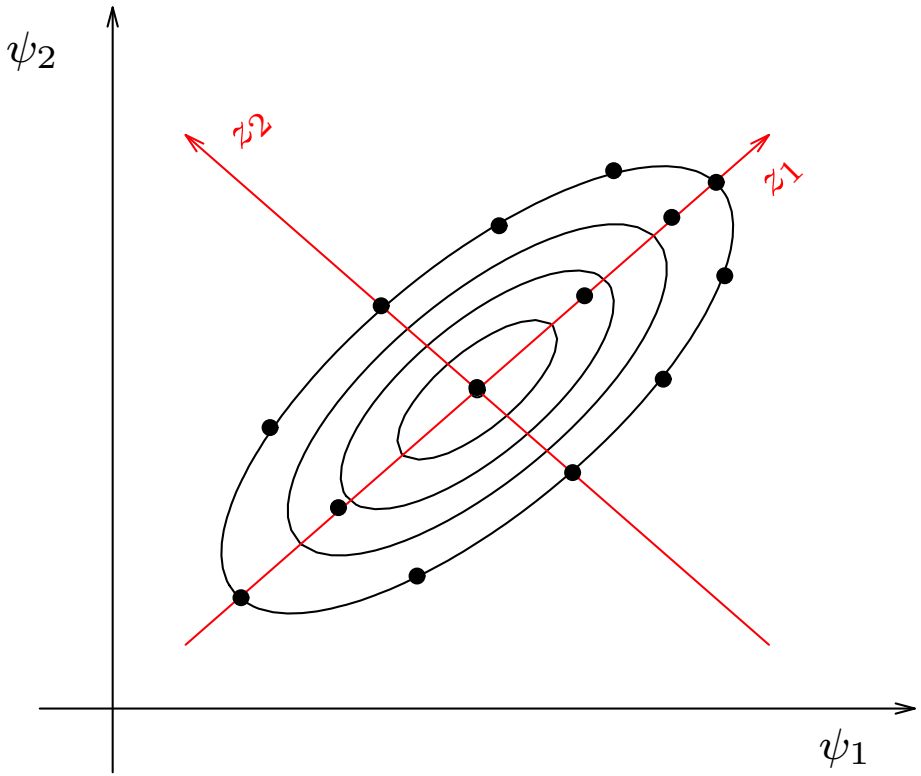
i. Explore the marginal joint posterior for the hyper-parameters $\tilde{p}(\boldsymbol{\psi} \mid \boldsymbol{y})$
   - Locate the mode $\hat{\boldsymbol{\psi}}$ by optimising $\log \tilde{p}(\boldsymbol{\psi} \mid \boldsymbol{y})$, eg using Newton-like algorithms
   - Compute the Hessian at $\hat{\boldsymbol{\psi}}$ and change co-ordinates to standardise the variables; this corrects for scale and rotation and simplifies integration



$$\mathsf{E}[\boldsymbol{z}] = 0$$
$$\mathsf{V}[\boldsymbol{z}] = \boldsymbol{\sigma}^2 \boldsymbol{I}$$

Operationally, the INLA algorithm proceeds with the following steps:

i. Explore the marginal joint posterior for the hyper-parameters $\tilde{p}(\boldsymbol{\psi} \mid \boldsymbol{y})$
   - Locate the mode $\hat{\boldsymbol{\psi}}$ by optimising $\log \tilde{p}(\boldsymbol{\psi} \mid \boldsymbol{y})$, eg using Newton-like algorithms
   - Compute the Hessian at $\hat{\boldsymbol{\psi}}$ and change co-ordinates to standardise the variables; this corrects for scale and rotation and simplifies integration
   - Explore $\log \tilde{p}(\boldsymbol{\psi} \mid \boldsymbol{y})$ and produce a grid of H points $\{\psi_h^*\}$ associated with the bulk of the mass, together with a corresponding set of area weights $\{\Delta_h\}$



ii. For each element ψ_h^* in the grid,
   - Obtain the marginal posterior $\tilde{p}(\psi_h^* \mid \boldsymbol{y})$, using interpolation and possibly correcting for (probable) skewness by using log-splines;
   - Evaluate the conditional posteriors $\tilde{p}(\theta_j \mid \psi_h^*, \boldsymbol{y})$ on a grid of selected values for θ_j;

ii. For each element ψ_h^* in the grid,
   - Obtain the marginal posterior $\tilde{p}(\psi_h^* \mid \boldsymbol{y})$, using interpolation and possibly correcting for (probable) skewness by using log-splines;
   - Evaluate the conditional posteriors $\tilde{p}(\theta_j \mid \psi_h^*, \boldsymbol{y})$ on a grid of selected values for θ_j;

iii. Marginalise ψ_h^* to obtain the marginal posteriors $\tilde{p}(\theta_j \mid \boldsymbol{y})$ using **numerical integration**

$$\tilde{p}(\theta_j \mid \boldsymbol{y}) \approx \sum_{h=1}^{H} \tilde{p}(\theta_j \mid \psi_h^*, \boldsymbol{y}) \tilde{p}(\psi_h^* \mid \boldsymbol{y}) \Delta_h$$

Integrated Nested Laplace Approximation (INLA)

So, it's all in the name...

Integrated Nested Laplace Approximation

- Because Laplace approximation is the basis to estimate the unknown distributions
- Because the Laplace approximations are nested within one another
  - Since (2.) is needed to estimate (1.)
  - NB: Consequently the estimation of (1.) might not be good enough, but it can be refined
- Because the required marginal posterior distributions are obtained by (numerical) integration

So, it's all in the name...

Integrated Nested **Laplace Approximation**

- Because Laplace approximation is the basis to estimate the unknown distributions
- Because the Laplace approximations are nested within one another
  - Since (2.) is needed to estimate (1.)
  - NB: Consequently the estimation of (1.) might not be good enough, but it can be refined
- Because the required marginal posterior distributions are obtained by (numerical) integration

So, it's all in the name...

Integrated **Nested** Laplace Approximation

- Because Laplace approximation is the basis to estimate the unknown distributions
- Because the Laplace approximations are nested within one another
  - Since (2.) is needed to estimate (1.)
  - NB: Consequently the estimation of (1.) might not be good enough, but it can be refined
- Because the required marginal posterior distributions are obtained by (numerical) integration

So, it's all in the name...

Integrated Nested Laplace Approximation

- Because Laplace approximation is the basis to estimate the unknown distributions
- Because the Laplace approximations are nested within one another
  - Since (2.) is needed to estimate (1.)
  - NB: Consequently the estimation of (1.) might not be good enough, but it can be refined
- Because the required marginal posterior distributions are obtained by (numerical) integration

INLA — example

- Suppose we want to make inference on a very simple model

$$
\begin{aligned}
y_i \mid \theta, \psi &\sim \text{Normal}(\theta, \psi) && (\psi = \sigma^{-2} \text{ is the precision}) \\
\theta \mid \psi &\sim \text{Normal}(\theta_0, \psi_0) && (\theta_0, \psi_0 \text{ known}) \\
\psi &\sim \text{Gamma}(a_0, b_0) && (a_0, b_0 \text{ known})
\end{aligned}
$$

INLA — example

- Suppose we want to make inference on a very simple model

$$
\begin{aligned}
y_i \mid \theta, \psi &\sim \text{Normal}(\theta, \psi) && (\psi = \sigma^{-2} \text{ is the precision}) \\
\theta \mid \psi &\sim \text{Normal}(\theta_0, \psi_0) && (\theta_0, \psi_0 \text{ known}) \\
\psi &\sim \text{Gamma}(a_0, b_0) && (a_0, b_0 \text{ known})
\end{aligned}
$$

- So, the model is made by a three-level hierarchy:
  1. Data $\boldsymbol{y} = (y_i)$ for $i = 1, \ldots, n$
  2. Parameter θ
  3. Hyper-parameter ψ

- Suppose we want to make inference on a very simple model

$$\begin{aligned} y_i \mid \theta, \psi &\sim \text{Normal}(\theta, \psi) && (\psi = \sigma^{-2} \text{ is the precision}) \\ \theta \mid \psi &\sim \text{Normal}(\theta_0, \psi_0) && (\theta_0, \psi_0 \text{ known}) \\ \psi &\sim \text{Gamma}(a_0, b_0) && (a_0, b_0 \text{ known}) \end{aligned}$$

- So, the model is made by a three-level hierarchy:
  1. Data $\boldsymbol{y} = (y_i)$ for $i = 1, \ldots, n$
  2. Parameter θ
  3. Hyper-parameter ψ

- **NB**: This model is in fact semi-conjugated, (θ, ψ) are *a priori* independent and so inference is possible numerically or using simple MCMC algorithms
  - $\theta \mid \boldsymbol{y}, \psi \sim \text{Normal}\,(\theta_n, \psi_n)$, with $\theta_n = \dfrac{\psi \sum_{i=1}^n y_i + \psi_0 \theta_0}{n\psi + \psi_0}$ and $\psi_n = n\psi + \psi_0$
  - $\psi \mid \theta, \boldsymbol{y} \sim \text{Gamma}(a_n, b_n)$, with $a_n = a_0 + \dfrac{n}{2}$ and $b_n = b_0 + \dfrac{\sum_{i=1}^n (y_i - \theta)^2}{2}$

INLA — example

- More generally, because of semi-conjugacy, we know that

$$\theta, \boldsymbol{y} \mid \psi \sim \text{Normal}(\cdot, \cdot) \quad \text{and} \quad \theta \mid \boldsymbol{y}, \psi \sim \text{Normal}(\cdot, \cdot)$$

- More generally, because of semi-conjugacy, we know that

$$\theta, \boldsymbol{y} \mid \psi \sim \text{Normal}(\cdot, \cdot) \quad \text{and} \quad \theta \mid \boldsymbol{y}, \psi \sim \text{Normal}(\cdot, \cdot)$$

- Thus

$$\begin{aligned} p(\psi \mid \boldsymbol{y}) &\propto p(\boldsymbol{y} \mid \psi) p(\psi) \\ &\propto \frac{\overbrace{p(\theta, \boldsymbol{y} \mid \psi)}^{\text{Gaussian}} p(\psi)}{\underbrace{p(\theta \mid \boldsymbol{y}, \psi)}_{\text{Gaussian}}} \end{aligned}$$

- More generally, because of semi-conjugacy, we know that

$$\theta, \boldsymbol{y} \mid \psi \sim \text{Normal}(\cdot, \cdot) \quad \text{and} \quad \theta \mid \boldsymbol{y}, \psi \sim \text{Normal}(\cdot, \cdot)$$

- Thus

$$\begin{aligned} p(\psi \mid \boldsymbol{y}) &\propto p(\boldsymbol{y} \mid \psi) p(\psi) \\ &\propto \frac{\overbrace{p(\theta, \boldsymbol{y} \mid \psi)}^{\text{Gaussian}} p(\psi)}{\underbrace{p(\theta \mid \boldsymbol{y}, \psi)}_{\text{Gaussian}}} \end{aligned}$$

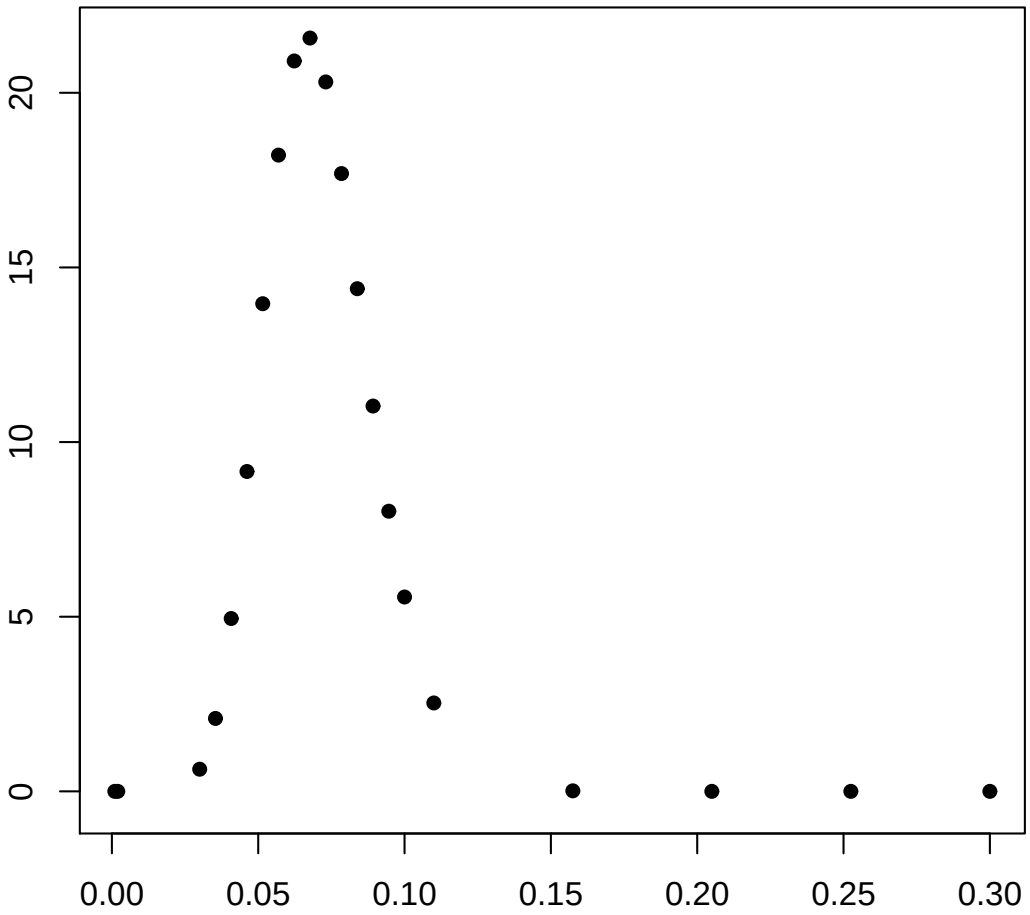
and

$$p(\theta \mid \boldsymbol{y}) = \int \underbrace{p(\theta \mid \boldsymbol{y}, \psi)}_{\text{Gaussian}} \underbrace{p(\psi \mid \boldsymbol{y})}_{\text{Approximated}} \, d\psi$$

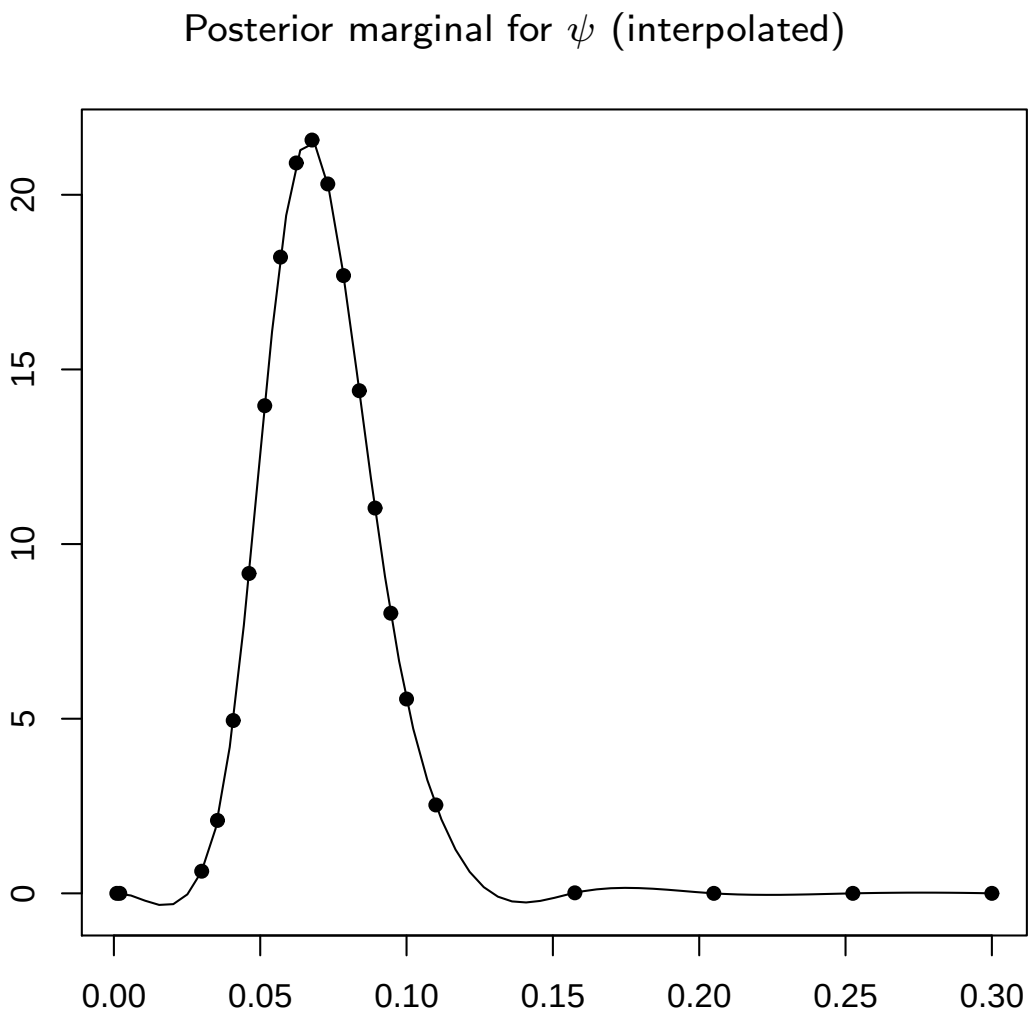
INLA — example

1. Select a grid of H points for ψ ($\{\psi_h^*\}$) and the associated area weights ($\{\Delta_h\}$)

Posterior marginal for ψ : $p(\psi \mid \boldsymbol{y}) \propto \frac{p(\theta, \boldsymbol{y}|\psi)p(\psi)}{p(\theta|\boldsymbol{y},\psi)}$

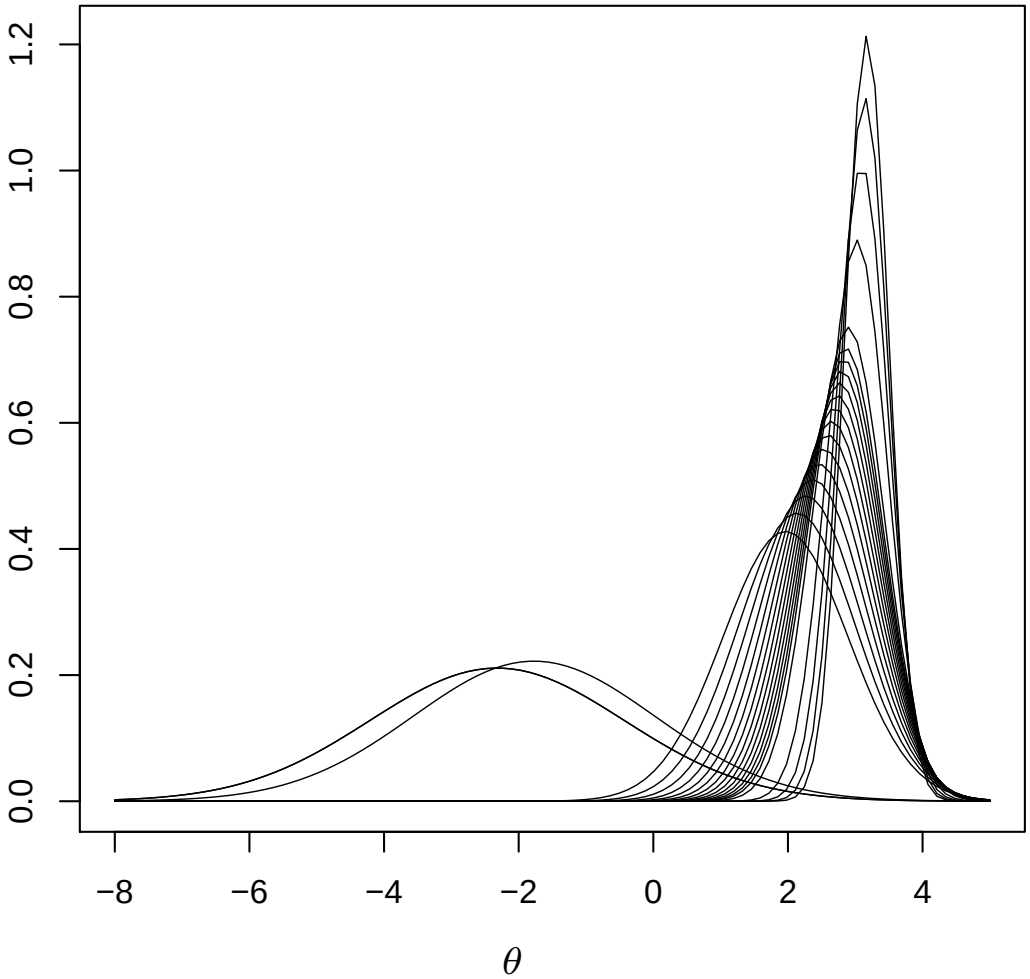


2. Interpolate the posterior density to compute the approximation to the posterior



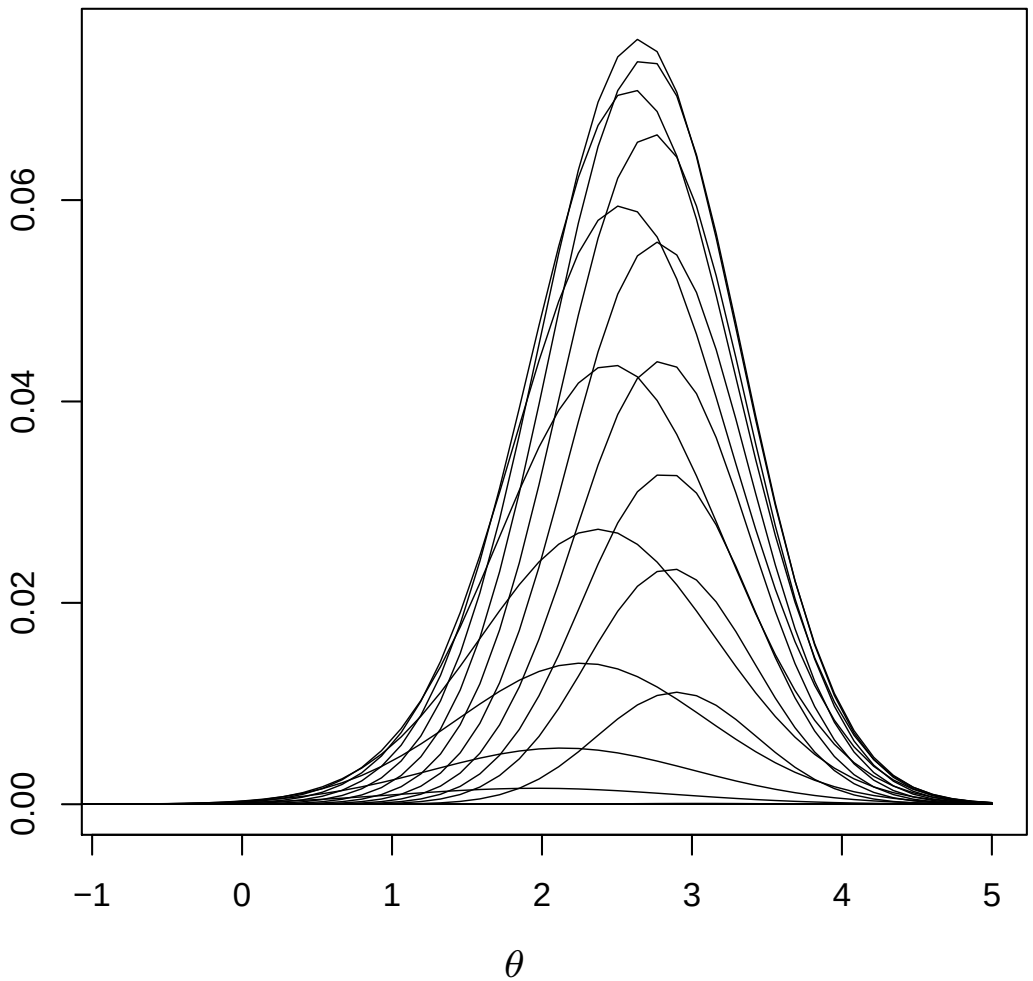
3. Approximates the conditional posterior of θ, given $\psi, \boldsymbol{y}$ on the $H-$dimensional grid

Posterior marginal for θ, conditional on each $\{\psi_h^*\}$ value (unweighted)

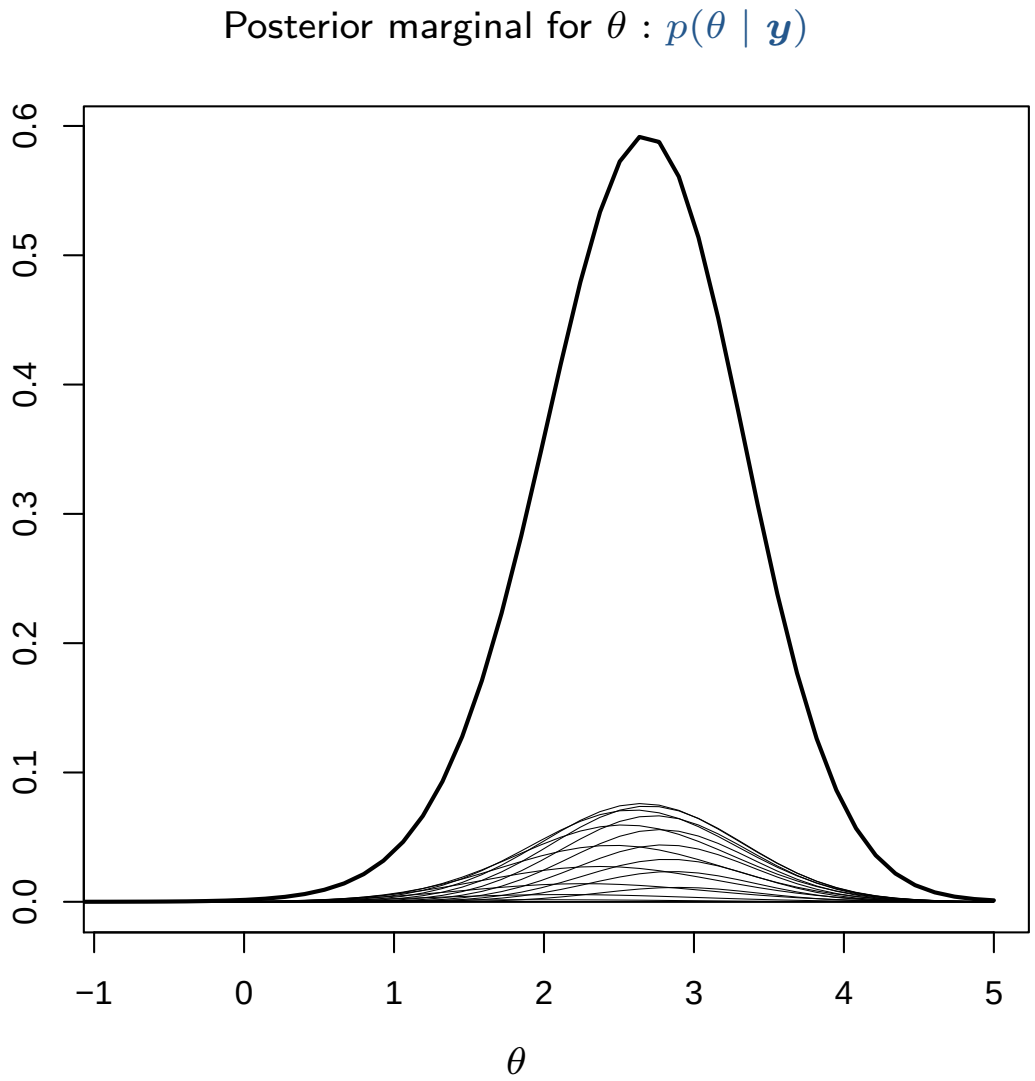


4. Weight the resulting (conditional) marginal posteriors by the density associated with each ψ on the grid

Posterior marginal for θ, conditional on each $\{\psi_h^*\}$ value (weighted)



5. (Numerically) sum over all the conditional densities to obtain the marginal posterior for θ



INLA — Summary

- The basic idea behind the INLA procedure is simple
  - Repeatedly use Laplace approximation and take advantage of computational simplifications due to the structure of the model
  - Use numerical integration to compute the required posterior marginal distributions
  - (If necessary) refine the estimation (eg using a finer grid)

INLA — Summary

- The basic idea behind the INLA procedure is simple
  - Repeatedly use Laplace approximation and take advantage of computational simplifications due to the structure of the model
  - Use numerical integration to compute the required posterior marginal distributions
  - (If necessary) refine the estimation (eg using a finer grid)

- Complications are mostly computational and occur when
  - Extending to more than one hyper-parameter
  - Markedly non-Gaussian observations

Using the package R-INLA

The INLA package for R

Good news is that all the procedures needed to perform INLA are implemented in a R package. This is effectively made by two components

The INLA package for R

Good news is that all the procedures needed to perform INLA are implemented in a R package. This is effectively made by two components

1. The GMRFLib library
   - This is a C library for fast and exact simulation of GMRFs, used to perform
     - Unconditional simulation of a GMRF;
     - Various types of conditional simulation from a GMRF;
     - Evaluation of the corresponding log-density;
     - Generation of blockupdates in MCMC-algorithms using GMRF-approximations or auxilliary variables, construction of non-Gaussian approximations to hidden GMRFs, approximate inference using INLA

The INLA package for R

Good news is that all the procedures needed to perform INLA are implemented in a R package. This is effectively made by two components

1. The GMRFLib library
   - This is a C library for fast and exact simulation of GMRFs, used to perform
     - Unconditional simulation of a GMRF;
     - Various types of conditional simulation from a GMRF;
     - Evaluation of the corresponding log-density;
     - Generation of blockupdates in MCMC-algorithms using GMRF-approximations or auxilliary variables, construction of non-Gaussian approximations to hidden GMRFs, approximate inference using INLA
2. The inla program
   - A standalone C program that
     - Interfaces with GMRFLib
     - Performs the relevant computation and returns the results in a standardised way

The INLA package for R

Good news is that all the procedures needed to perform INLA are implemented in a R package. This is effectively made by two components

1. The GMRFLib library
   - This is a C library for fast and exact simulation of GMRFs, used to perform
     - Unconditional simulation of a GMRF;
     - Various types of conditional simulation from a GMRF;
     - Evaluation of the corresponding log-density;
     - Generation of blockupdates in MCMC-algorithms using GMRF-approximations or auxilliary variables, construction of non-Gaussian approximations to hidden GMRFs, approximate inference using INLA
2. The inla program
   - A standalone C program that
     - Interfaces with GMRFLib
     - Performs the relevant computation and returns the results in a standardised way

NB: Because the package R-INLA relies on a standalone C program, it is not available directly from CRAN

The INLA package for R — Installation

- Visit the website

  www.r-inla.org

  and follow the instructions
- The website contains source code, examples, papers and reports discussing the theory and applications of INLA

The INLA package for R — Installation

- Visit the website

  www.r-inla.org

  and follow the instructions
- The website contains source code, examples, papers and reports discussing the theory and applications of INLA

- From R, installation is performed typing
  ```
  source("http://www.math.ntnu.no/inla/givemeINLA.R")
  ```
- Later, you can upgrade the package by typing
  ```
  inla.upgrade()
  ```
- A test-version (which may contain unstable updates/new functions) can be obtained by typing
  ```
  inla.upgrade(testing=TRUE)
  ```

The INLA package for R — Installation

- Visit the website

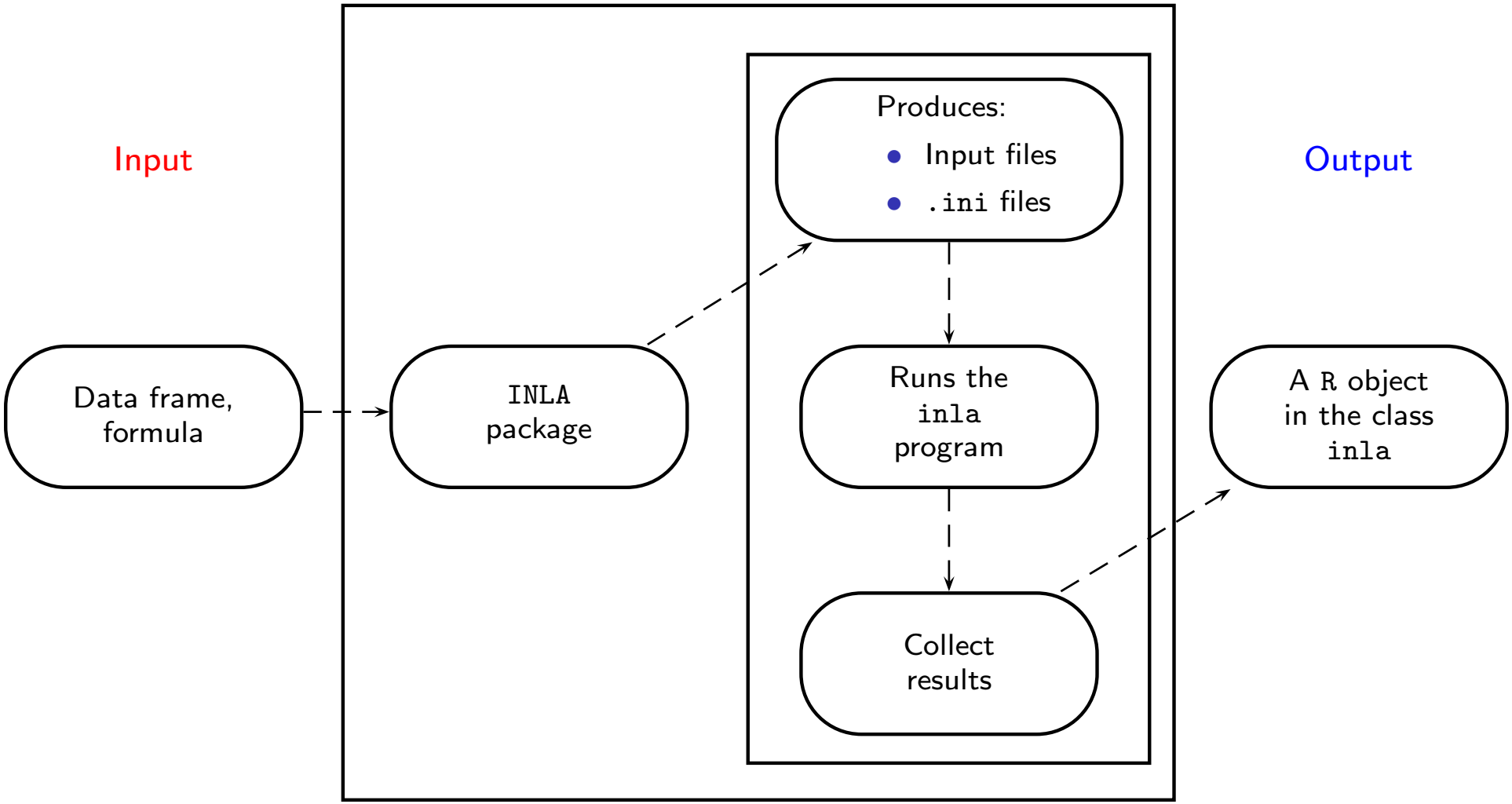
  www.r-inla.org

  and follow the instructions
- The website contains source code, examples, papers and reports discussing the theory and applications of INLA

- From R, installation is performed typing
  ```
  source("http://www.math.ntnu.no/inla/givemeINLA.R")
  ```
- Later, you can upgrade the package by typing
  ```
  inla.upgrade()
  ```
- A test-version (which may contain unstable updates/new functions) can be obtained by typing
  ```
  inla.upgrade(testing=TRUE)
  ```

- R-INLA runs natively under Linux, Windows and Mac and it is possible to do multi-threading using OpenMP

The INLA package for R — How does it work?



The INLA package for R — Documentation

- There has been a great effort lately in producing quite a lot of user-frienly(-*ish*) documentation
- Tutorials are (or will shortly be) available on
  - Basic INLA (probably later this year)
  - SPDE (spatial models based on stochastic partial differential equations) models
- Blangiardo & Cameletti *Spatial and spatio-temporal Bayesian models with R-INLA* (out end of 2014)

The INLA package for R — Documentation

- There has been a great effort lately in producing quite a lot of user-frienly(-*ish*) documentation
- Tutorials are (or will shortly be) available on
  - Basic INLA (probably later this year)
  - SPDE (spatial models based on stochastic partial differential equations) models
- Blangiardo & Cameletti *Spatial and spatio-temporal Bayesian models with R-INLA* (out end of 2014)

- Much of the recent development in R-INLA is devoted to extending the applications of INLA for spatial and spatio-temporal models as well as producing detailed information

- The website also has a discussion forum and a FAQ page

1. The first thing to do is to **specify the model**

- For example, assume we have a generic model

$$\begin{aligned} y_i &\overset{iid}{\sim} p(y_i \mid \theta_i) \\ \eta_i &= g(\theta_i) = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + f(z_i) \end{aligned}$$

where

- $\boldsymbol{x} = (x_1, x_2)$ are observed covariates for which we are assuming a linear effect on some function $g(\cdot)$ of the parameter θ_i
- $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2) \sim \text{Normal}(0, \tau_1^{-1})$ are unstructured ("fixed") effects
- z is an **index**. This can be used to include structured ("random"), spatial, spatio-temporal effect, etc.
- $f \sim \text{Normal}(0, \boldsymbol{Q}_f^{-1}(\tau_2))$ is a suitable function used to model the structured effects

1. The first thing to do is to **specify the model**

- For example, assume we have a generic model

$$\begin{aligned} y_i &\overset{iid}{\sim} p(y_i \mid \theta_i) \\ \eta_i &= g(\theta_i) = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + f(z_i) \end{aligned}$$

where

  - $\boldsymbol{x} = (x_1, x_2)$ are observed covariates for which we are assuming a linear effect on some function $g(\cdot)$ of the parameter θ_i
  - $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2) \sim \text{Normal}(0, \tau_1^{-1})$ are unstructured ("fixed") effects
  - z is an **index**. This can be used to include structured ("random"), spatial, spatio-temporal effect, etc.
  - $f \sim \text{Normal}(0, \boldsymbol{Q}_f^{-1}(\tau_2))$ is a suitable function used to model the structured effects

- As mentioned earlier, this formulation can actually be used to represent quite a wide class of models!

Step by step guide to using R-INLA

- The model is translated in R code using a formula
- This is sort of standard in R (you would do pretty much the same for calls to functions such as lm, or glm, or lmer)

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

Step by step guide to using R-INLA

- The model is translated in R code using a formula
- This is sort of standard in R (you would do pretty much the same for calls to functions such as lm, or glm, or lmer)

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

- The f() function can account for several structured effects
- This is done by specifying a different model
  - iid, iid1d, iid2d, iid3d specify random effects
  - rw1, rw2, ar1 are smooth effect of covariates or time effects
  - seasonal specifies a seasonal effect
  - besag models spatially structured effects (CAR)
  - generic is a user-defined precision matrix

2. Call the function `inla`, specifying the data and options (more on this later), eg

```
m = inla(formula, data=data.frame(y,x1,x2,z))
```

Step by step guide to using R-INLA

2. Call the function `inla`, specifying the data and options (more on this later), eg

```
m = inla(formula, data=data.frame(y,x1,x2,z))
```

- The data need to be included in a suitable `data.frame`
- `R` returns an object `m` in the class `inla`, which has some methods available
  - `summary()`
  - `plot()`
- The options let you specify the priors and hyperpriors, together with additional output

```
names(m)
 [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"                          "model.random"
[13] "summary.random"               "marginals.random"
[15] "size.random"                  "summary.linear.predictor"
[17] "marginals.linear.predictor"   "summary.fitted.values"
[19] "marginals.fitted.values"      "size.linear.predictor"
[21] "summary.hyperpar"             "marginals.hyperpar"
[23] "internal.summary.hyperpar"    "internal.marginals.hyperpar"
[25] "si"                           "offset.linear.predictor"
[27] "model.spde2.blc"              "summary.spde2.blc"
[29] "marginals.spde2.blc"          "size.spde2.blc"
[31] "logfile"                      "misc"
[33] "dic"                          "mode"
[35] "neffp"                        "joint.hyper"
[37] "nhyper"                       "version"
[39] "Q"                            "graph"
[41] "cpu.used"                     ".args"
[43] "call"                         "model.matrix"
```

First, generate some data from an assumed model

$$y_i \sim \text{Binomial}(\pi_i, N_i), \qquad \text{for } i = 1, \ldots, n = 12$$

```
library(INLA)

# Data generation
n=12
Ntrials = sample(c(80:100), size=n, replace=TRUE)
eta = rnorm(n,0,0.5)
prob = exp(eta)/(1 + exp(eta))
y = rbinom(n, size=Ntrials, prob = prob)
data=data.frame(y=y,z=1:n,Ntrials)
```

Example — Binary data with individual random effect

```
data
```

	y	z	Ntrials
1	50	1	95
2	37	2	97
3	36	3	93
4	47	4	96
5	39	5	80
6	67	6	97
7	60	7	89
8	57	8	84
9	34	9	89
10	57	10	96
11	46	11	87
12	48	12	98

Example — Binary data with individual random effect

data

	y	z	Ntrials
1	50	1	95
2	37	2	97
3	36	3	93
4	47	4	96
5	39	5	80
6	67	6	97
7	60	7	89
8	57	8	84
9	34	9	89
10	57	10	96
11	46	11	87
12	48	12	98

We want to fit the following model

$$
\begin{aligned}
y_i &\sim \text{Binomial}(\pi_i, N_i), && \text{for } i = 1, \ldots, n = 12 \\
\text{logit}(\pi_i) &= \alpha + f(z_i) \\
\alpha &\sim \text{Normal}(0, 1\,000) && (\text{“fixed” effect}) \\
f(z_i) &\sim \text{Normal}(0, \sigma^2) && (\text{“random” effect}) \\
p(\sigma^2) &\propto \sigma^{-2} = \tau && (\text{“non-informative” prior}) \\
&\approx \log \sigma \sim \text{Uniform}(0, \infty)
\end{aligned}
$$

data

	y	z	Ntrials
1	50	1	95
2	37	2	97
3	36	3	93
4	47	4	96
5	39	5	80
6	67	6	97
7	60	7	89
8	57	8	84
9	34	9	89
10	57	10	96
11	46	11	87
12	48	12	98

We want to fit the following model

$$
\begin{aligned}
y_i &\sim \text{Binomial}(\pi_i, N_i), && \text{for } i = 1, \ldots, n = 12 \\
\text{logit}(\pi_i) &= \alpha + f(z_i) \\
\alpha &\sim \text{Normal}(0, 1\,000) && (\text{“fixed” effect}) \\
f(z_i) &\sim \text{Normal}(0, \sigma^2) && (\text{“random” effect}) \\
p(\sigma^2) &\propto \sigma^{-2} = \tau && (\text{“non-informative” prior}) \\
&\approx \log \sigma \sim \text{Uniform}(0, \infty)
\end{aligned}
$$

This can be done by typing in R

```
formula = y ~ f(z,model="iid",
                hyper=list(list(prior="flat")))
m=inla(formula, data=data,
       family="binomial",
       Ntrials=Ntrials,
       control.predictor = list(compute = TRUE))
summary(m)
```

```
data
```

	y	z	Ntrials
1	50	1	95
2	37	2	97
3	36	3	93
4	47	4	96
5	39	5	80
6	67	6	97
7	60	7	89
8	57	8	84
9	34	9	89
10	57	10	96
11	46	11	87
12	48	12	98

We want to fit the following model

$$
\begin{aligned}
y_i &\sim \text{Binomial}(\pi_i, N_i), && \text{for } i = 1, \ldots, n = 12 \\
\text{logit}(\pi_i) &= \alpha + f(z_i) \\
\alpha &\sim \text{Normal}(0, 1\,000) && (\text{“fixed” effect}) \\
f(z_i) &\sim \text{Normal}(0, \sigma^2) && (\text{“random” effect}) \\
p(\sigma^2) &\propto \sigma^{-2} = \tau && (\text{“non-informative” prior}) \\
&\approx \log \sigma \sim \text{Uniform}(0, \infty)
\end{aligned}
$$

This can be done by typing in R

```
formula = y ~ f(z,model="iid",
                hyper=list(list(prior="flat")))
m=inla(formula, data=data,
       family="binomial",
       Ntrials=Ntrials,
       control.predictor = list(compute = TRUE))
summary(m)
```

Example — Binary data with individual random effect

```
Call:
c("inla(formula = formula, family = \"binomial\", data = data, Ntrials = Ntrials,
  "control.predictor = list(compute = TRUE))")

Time used:
 Pre-processing    Running inla Post-processing           Total
         0.2258          0.0263          0.0744          0.3264

Fixed effects:
               mean    sd 0.025quant 0.5quant 0.975quant kld
(Intercept) -0.0021 0.136     -0.272  -0.0021      0.268   0

Random effects:
Name   Model
 z   IID model

Model hyperparameters:
                mean   sd     0.025quant 0.5quant 0.975quant
Precision for z  7.130  4.087  2.168      6.186    17.599

Expected number of effective parameters(std dev): 9.494(0.7925)
Number of equivalent replicates : 1.264

Marginal Likelihood:  -54.28
CPO and PIT are computed

Posterior marginals for linear predictor and fitted values computed
```

Exploring the R-INLA output

```
Fixed effects:
               mean    sd 0.025quant 0.5quant 0.975quant kld
(Intercept) -0.0021 0.136     -0.272  -0.0021      0.268   0
```

Exploring the R-INLA output

```
Fixed effects:
              mean    sd 0.025quant 0.5quant 0.975quant kld
(Intercept) -0.0021 0.136     -0.272  -0.0021      0.268   0
```

- For each unstructured ("fixed") effect, R-INLA reports a set of summary statistics from the posterior distribution

Exploring the R-INLA output

```
Fixed effects:
               mean    sd 0.025quant 0.5quant 0.975quant kld
(Intercept) -0.0021 0.136     -0.272  -0.0021      0.268   0
```

- For each unstructured ("fixed") effect, R-INLA reports a set of summary statistics from the posterior distribution
- The value of the Kullback-Leibler divergence (KLD) describes the difference between the standard Gaussian and the Simplified Laplace Approximation to the marginal posterior densities
  - Small values indicate that the posterior distribution is well approximated by a Normal distribution
  - If so, the more sophisticated SLA gives a "good" error rate and therefore there is no need to use the more computationally intensive "full" Laplace approximation

Exploring the R-INLA output

```
Random effects:
Name   Model
 z   IID model

Model hyperparameters:
                  mean    sd     0.025quant 0.5quant 0.975quant
Precision for z  7.130  4.087  2.168       6.186    17.599
```

- Also for each hyper-parameter, the summary statistics are reported to describe the posterior distribution
- **NB**: INLA reports results on the **precision** scale (more on this later)

Exploring the R-INLA output

```
Expected number of effective parameters(std dev): 9.494(0.7925)
Number of equivalent replicates : 1.264
```

Exploring the R-INLA output

```
Expected number of effective parameters(std dev): 9.494(0.7925)
Number of equivalent replicates : 1.264
```

- The expected number of effective parameters is basically the number of **independent** parameters included in the model
  - In a hierarchical model, because of shrinkage, information is shared across parameters
  - Example: in this case there are 14 actual parameters ($\alpha, \sigma^2, f(1), \ldots, f(12)$). However, because the structured effects are exchangeable (ie correlated) the "effective" number of parameters is (on average) just 9.5

Exploring the R-INLA output

```
Expected number of effective parameters(std dev): 9.494(0.7925)
Number of equivalent replicates : 1.264
```

- The expected number of effective parameters is basically the number of **independent** parameters included in the model
  - In a hierarchical model, because of shrinkage, information is shared across parameters
  - Example: in this case there are 14 actual parameters $(\alpha, \sigma^2, f(1), \ldots, f(12))$. However, because the structured effects are exchangeable (ie correlated) the "effective" number of parameters is (on average) just 9.5

- The number of equivalent replicates indicates the available information (in terms of sample size) per effective parameter
  - Example: there are 12 data points and on average 9.5 parameters; so each is estimated using on average $12/9.5 \approx 1.3$ data points
  - Low values (with respect to the overall sample size) are indicative of poor fit

Exploring the R-INLA output

```
Marginal Likelihood:  -54.28
CPO and PIT are computed
```

- R-INLA can produce two types of "leave-one-out" measures of fit
  1. Conditional Predictive Ordinate (CPO): $p(y_i \mid \boldsymbol{y}_{-i})$
     - "Extreme" values for CPO indicate a surprising observation
  2. Probability Integral Transforms (PIT): $\Pr(y_i^{\text{new}} \leq y_i \mid \boldsymbol{y}_{-i})$
     - "Extreme" values for PIT indicate outliers
     - A histogram of PIT that does not look Uniformly distributed indicate lack of fit for the current model

Exploring the R-INLA output

```
Marginal Likelihood:  -54.28
CPO and PIT are computed
```

- R-INLA can produce two types of "leave-one-out" measures of fit
  1. Conditional Predictive Ordinate (CPO): $p(y_i \mid \boldsymbol{y}_{-i})$
     - "Extreme" values for CPO indicate a surprising observation
  2. Probability Integral Transforms (PIT): $\Pr(y_i^{\text{new}} \leq y_i \mid \boldsymbol{y}_{-i})$
     - "Extreme" values for PIT indicate outliers
     - A histogram of PIT that does not look Uniformly distributed indicate lack of fit for the current model

- If the option

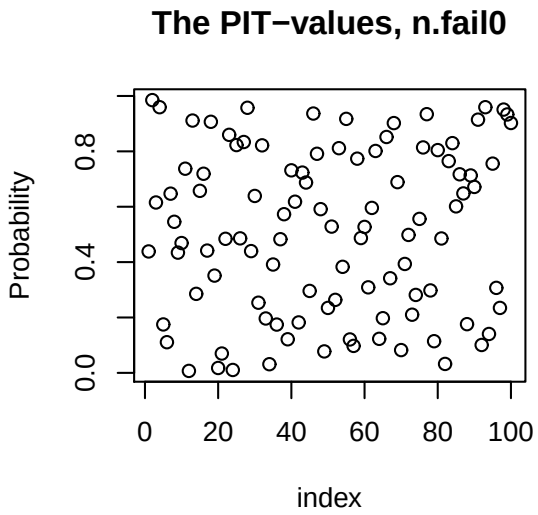
  ```
  control.compute=list(cpo=TRUE)
  ```

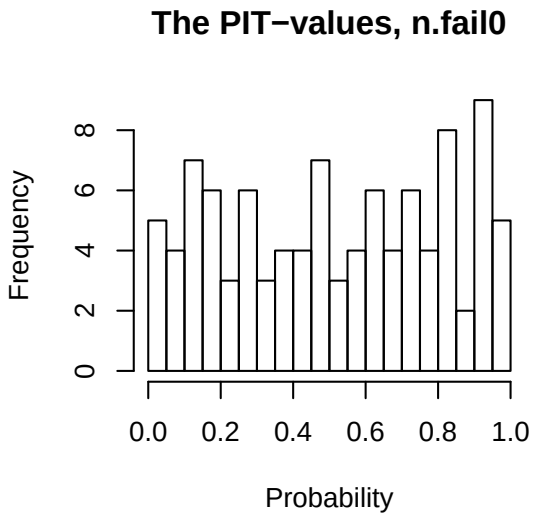
  is added to the call to the function `inla` then the resulting object contains values for CPO and PIT, which can then be post-processed
  - **NB**: for the sake of model checking, it is useful to to increase the accuracy of the estimation for the tails of the marginal distributions
  - This can be done by adding the option

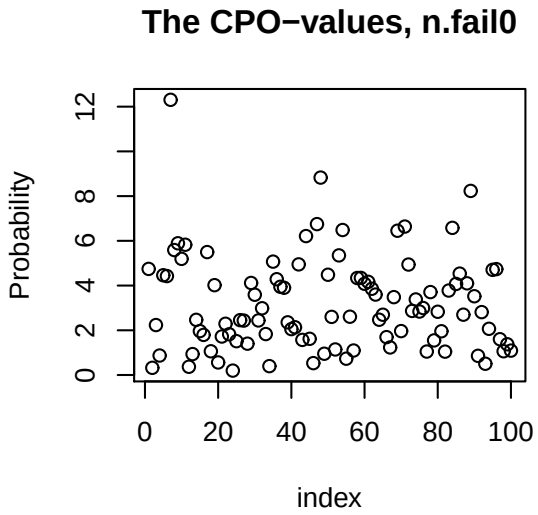
    ```
    control.inla = list(strategy = "laplace", npoints = 21)
    ```

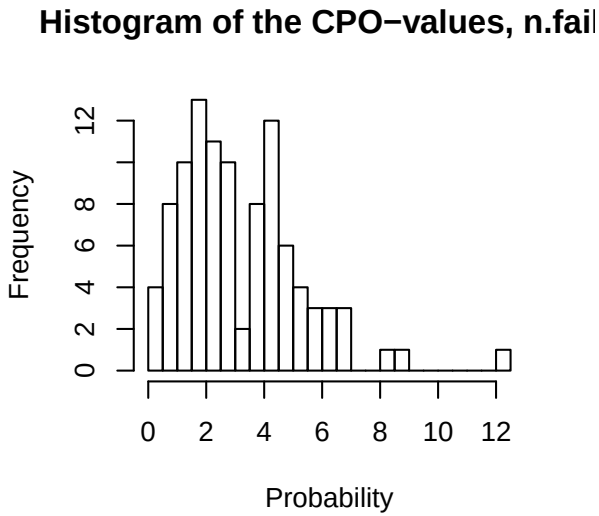
    to add more evaluation points (npoints=21) instead of the default npoints=9

Exploring the R-INLA output

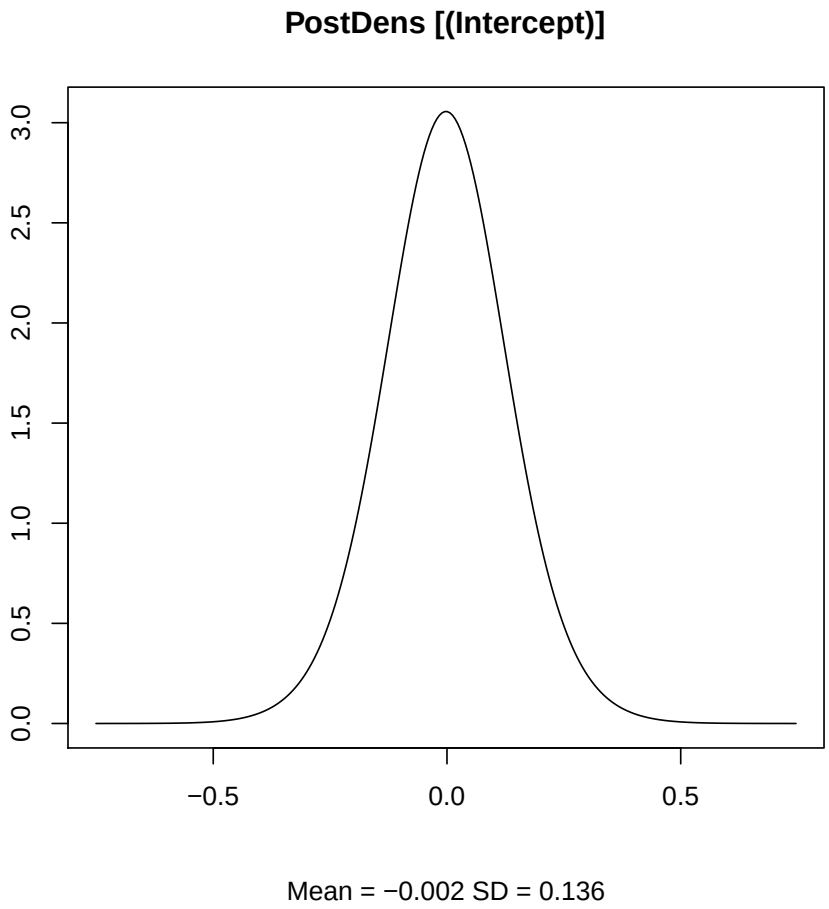






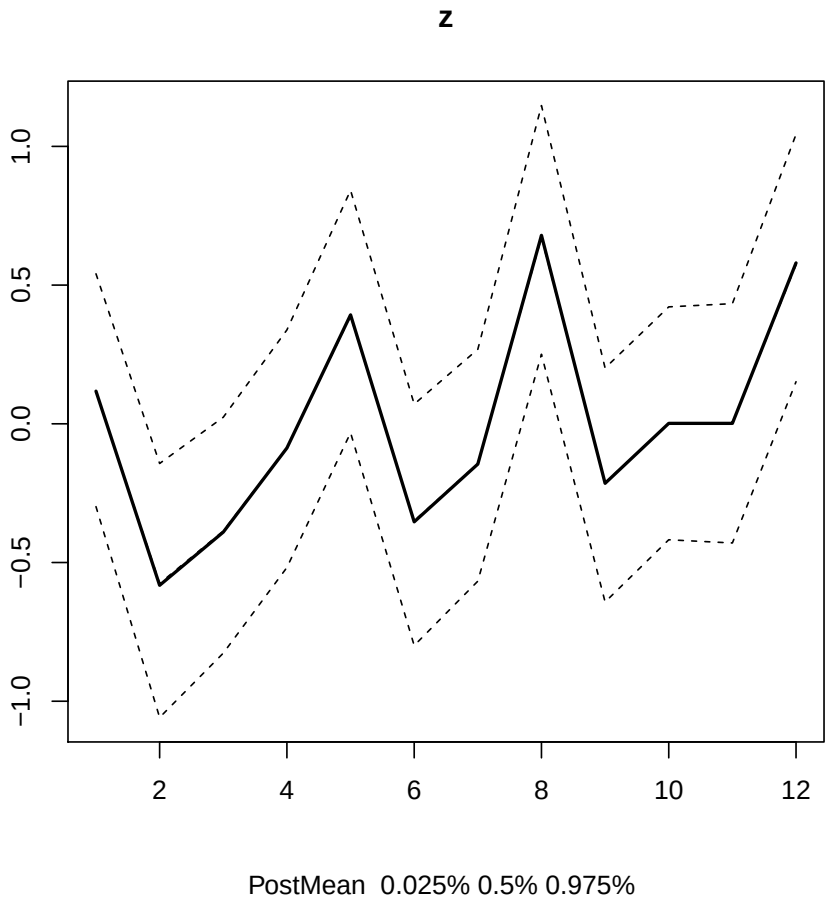


```
plot(m)


plot(m,
     plot.fixed.effects = TRUE,
     plot.lincomb = FALSE,
     plot.random.effects = FALSE,
     plot.hyperparameters = FALSE,
     plot.predictor = FALSE,
     plot.q = FALSE,
     plot.cpo = FALSE
)


plot(m,single = TRUE)
```



```
plot(m)
```

```
plot(m,
     plot.fixed.effects = FALSE,
     plot.lincomb = FALSE,
     plot.random.effects = TRUE,
     plot.hyperparameters = FALSE,
     plot.predictor = FALSE,
     plot.q = FALSE,
     plot.cpo = FALSE
)
```



Manipulating the results from R-INLA

- The elements of the object `m` can be used for post-processing

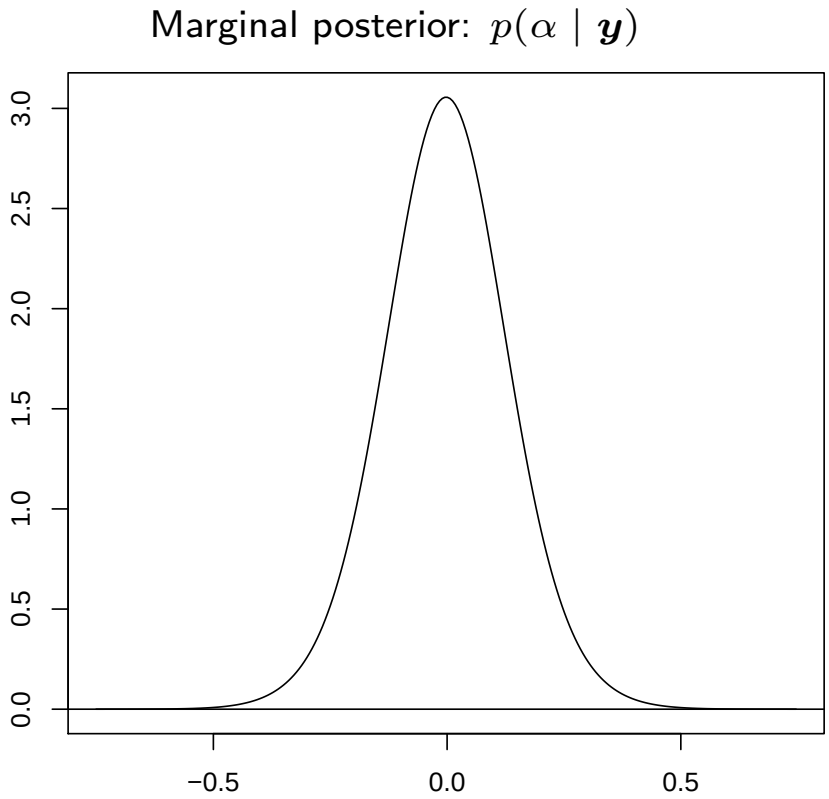
```
m$summary.fixed
```

```
                     mean        sd 0.025quant     0.5quant 0.975quant          kld
(Intercept) -0.002092578 0.1360447 -0.2720331 -0.002101465  0.2680023 1.866805e-08
```

```
m$summary.random

$z
   ID         mean        sd  0.025quant     0.5quant  0.975quant          kld
1   1  0.117716597 0.2130482 -0.29854459  0.116540837  0.54071007 1.561929e-06
2   2 -0.582142549 0.2328381 -1.05855344 -0.575397613 -0.14298960 3.040586e-05
3   3 -0.390419424 0.2159667 -0.82665552 -0.386498698  0.02359256 1.517773e-05
4   4 -0.087199172 0.2174477 -0.51798771 -0.086259111  0.33838724 7.076793e-07
5   5  0.392724605 0.2220260 -0.03217954  0.388462164  0.84160800 1.604348e-05
6   6 -0.353323459 0.2210244 -0.79933142 -0.349483252  0.07088015 1.242953e-05
7   7 -0.145238917 0.2122322 -0.56726042 -0.143798605  0.26859415 2.047815e-06
8   8  0.679294456 0.2279863  0.25076022  0.672226639  1.14699903 4.145645e-05
9   9 -0.214441626 0.2141299 -0.64230245 -0.212274011  0.20094086 4.577080e-06
10 10  0.001634115 0.2131451 -0.41797579  0.001622300  0.42152562 4.356243e-09
11 11  0.001593724 0.2190372 -0.42961274  0.001581019  0.43309253 3.843622e-09
12 12  0.580008923 0.2267330  0.15173745  0.573769187  1.04330359 3.191737e-05
```

Manipulating the results from R-INLA

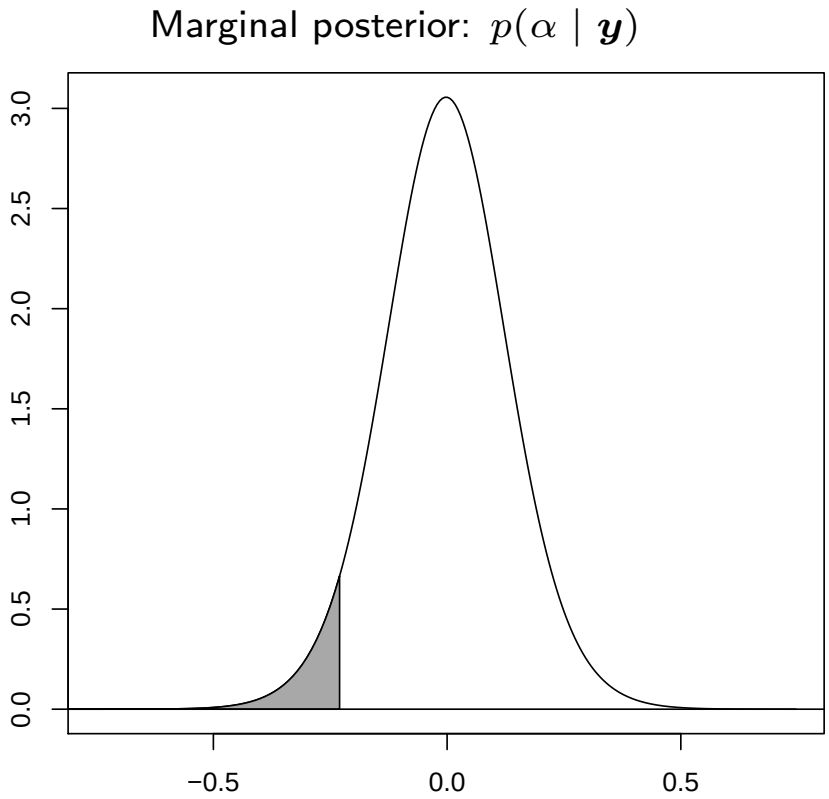
```
alpha <- m$marginals.fixed[[1]]
plot(inla.smarginal(alpha),t="l")
```



Manipulating the results from R-INLA

```
alpha <- m$marginals.fixed[[1]]
plot(inla.smarginal(alpha),t="l")
```

```
inla.qmarginal(0.05,alpha)
[1] -0.2257259
```

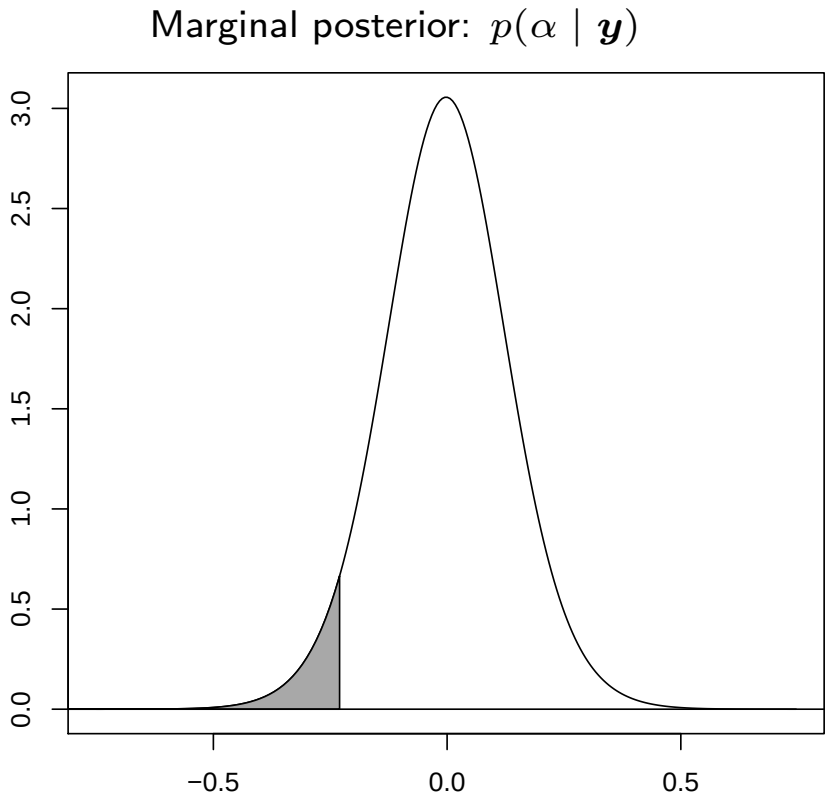


Manipulating the results from R-INLA

```
alpha <- m$marginals.fixed[[1]]
plot(inla.smarginal(alpha),t="l")
```

```
inla.qmarginal(0.05,alpha)
[1] -0.2257259
```

```
inla.pmarginal(-.2257259,alpha)
[1] 0.04999996
```



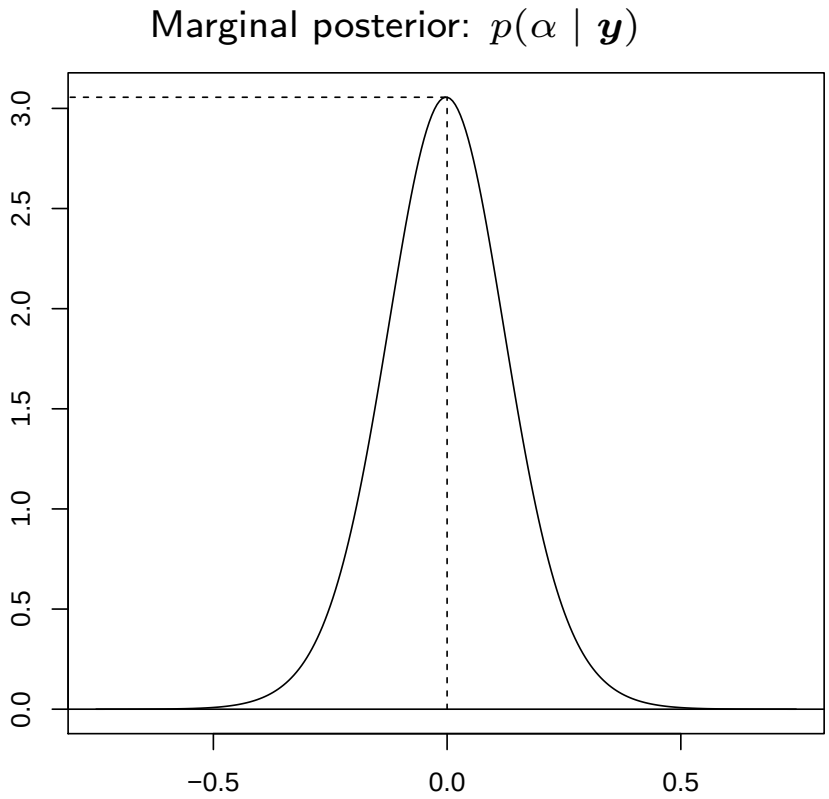
Manipulating the results from R-INLA

```
alpha <- m$marginals.fixed[[1]]
plot(inla.smarginal(alpha),t="l")
```

```
inla.qmarginal(0.05,alpha)
[1] -0.2257259
```

```
inla.pmarginal(-.2257259,alpha)
[1] 0.04999996
```

```
inla.dmarginal(0,alpha)
[1] 3.055793
```



Manipulating the results from R-INLA

```
alpha <- m$marginals.fixed[[1]]
plot(inla.smarginal(alpha),t="l")
```

```
inla.qmarginal(0.05,alpha)
[1] -0.2257259
```

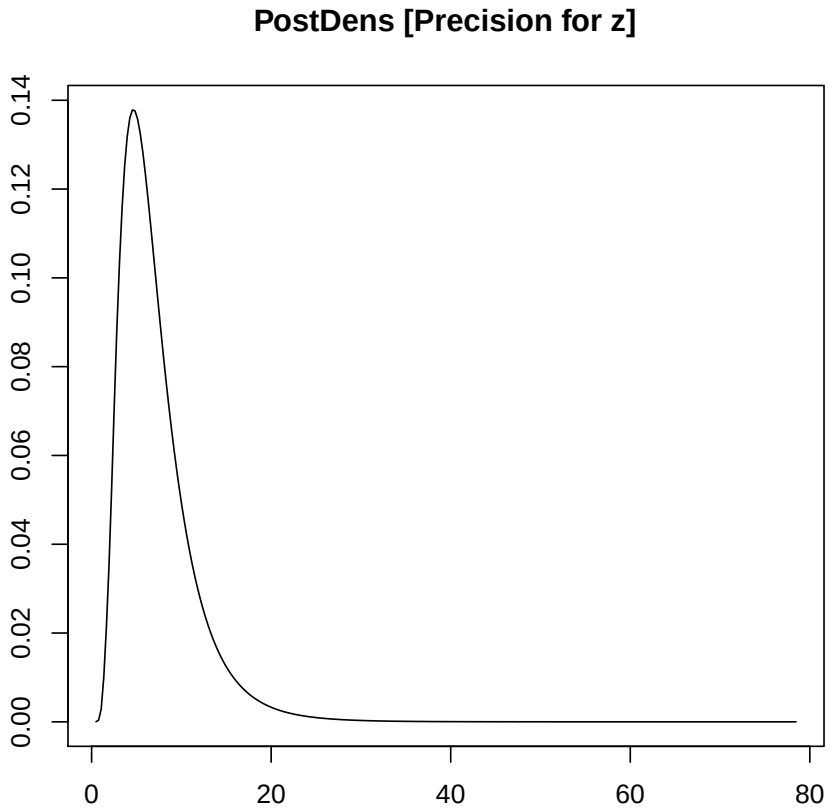
```
inla.pmarginal(-.2257259,alpha)
[1] 0.04999996
```

```
inla.dmarginal(0,alpha)
[1] 3.055793
```

```
inla.rmarginal(4,alpha)
[1]  0.05307452  0.07866796  -0.09931744 -0.02027463
```

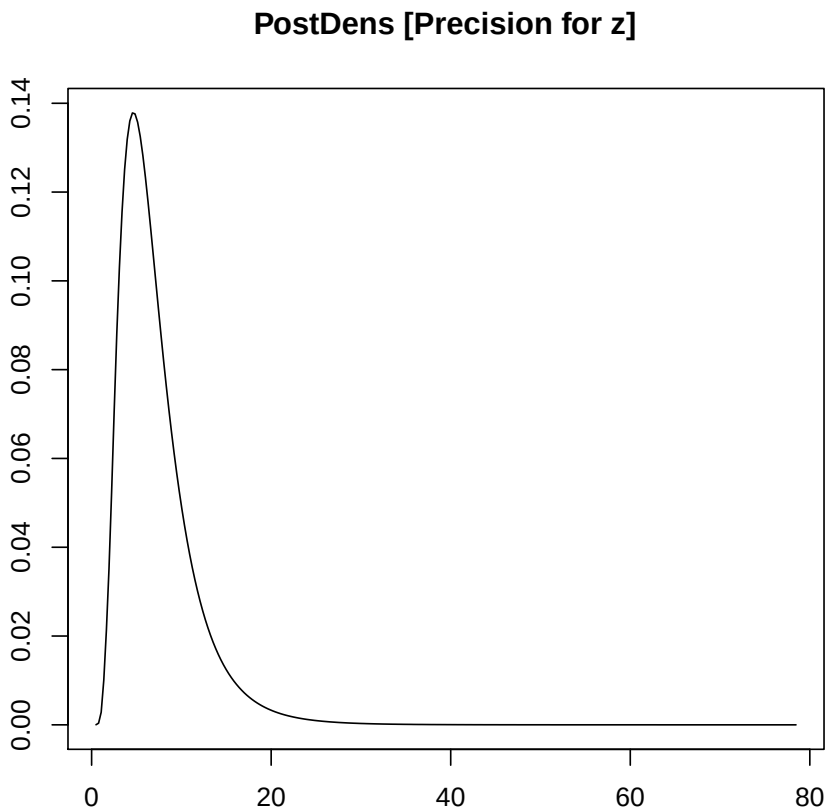
NB: INLA works by default with **precisions**

```
plot(m,
     plot.fixed.effects = FALSE,
     plot.lincomb = FALSE,
     plot.random.effects = FALSE,
     plot.hyperparameters = TRUE,
     plot.predictor = FALSE,
     plot.q = FALSE,
     plot.cpo = FALSE
)
```



NB: INLA works by default with **precisions**

```
plot(m,
     plot.fixed.effects = FALSE,
     plot.lincomb = FALSE,
     plot.random.effects = FALSE,
     plot.hyperparameters = TRUE,
     plot.predictor = FALSE,
     plot.q = FALSE,
     plot.cpo = FALSE
)
```



Problem: usually, we want to make inference on more interpretable parameters, eg **standard deviations**

- Using some built-in INLA functions
  - model$marginals.hyperpar
  - inla.expectation
  - inla.rmarginal

  it is possible to compute the structured variability, for example on the standard deviation scale, based on nsamples (default=1000) MC simulations from the estimated precision

```
s <- inla.contrib.sd(m,nsamples=1000)
s$hyper
```

```
                mean        sd      2.5%     97.5%
sd for z 0.416862 0.1098968 0.2332496 0.6478648
```

Example — Binary data with individual random effect

- Using some built-in INLA functions
  - model$marginals.hyperpar
  - inla.expectation
  - inla.rmarginal

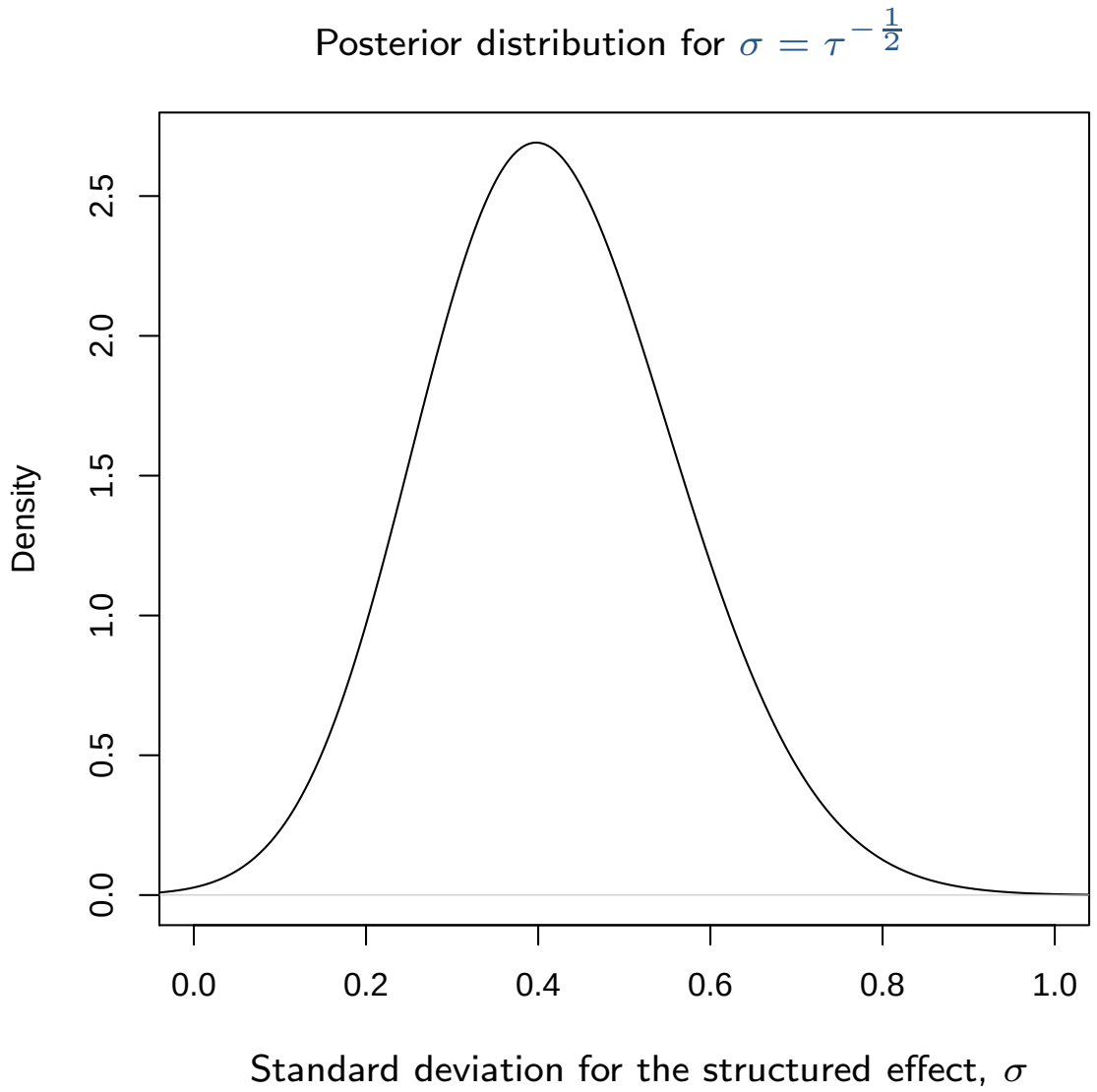
  it is possible to compute the structured variability, for example on the standard deviation scale, based on nsamples (default=1000) MC simulations from the estimated precision

```
s <- inla.contrib.sd(m,nsamples=1000)
s$hyper
```

```
                mean        sd      2.5%     97.5%
sd for z 0.416862 0.1098968 0.2332496 0.6478648
```

- The object s contains a vector of simulations from the induced posterior distribution for the standard deviation scale, than can then be used for plots

```
hist(s$samples)
plot(density(s$samples,bw=.1),xlab="sigma",main="")
```


Posterior distribution for $\sigma = \tau^{-\frac{1}{2}}$
Density
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
Standard deviation for the structured effect, σ

Example — Binary data with individual random effect

If we wanted to perform MCMC on this model, we could

1. Program it in JAGS/BUGS and save it as `model.txt`

```
model {
    for (i in 1:n) {
        y[i] ~ dbinom(pi[i],Ntrials[i])
        logit(pi[i]) <- alpha+f[i]
        f[i] ~ dnorm(0,tau)
    }
    alpha ~ dnorm(0,.001)
    log.sigma ~ dunif(0,10000)
    sigma <- exp(log.sigma)
    tau <- pow(sigma,-2)
}
```

Example — Binary data with individual random effect

If we wanted to perform MCMC on this model, we could

1. Program it in JAGS/BUGS and save it as model.txt

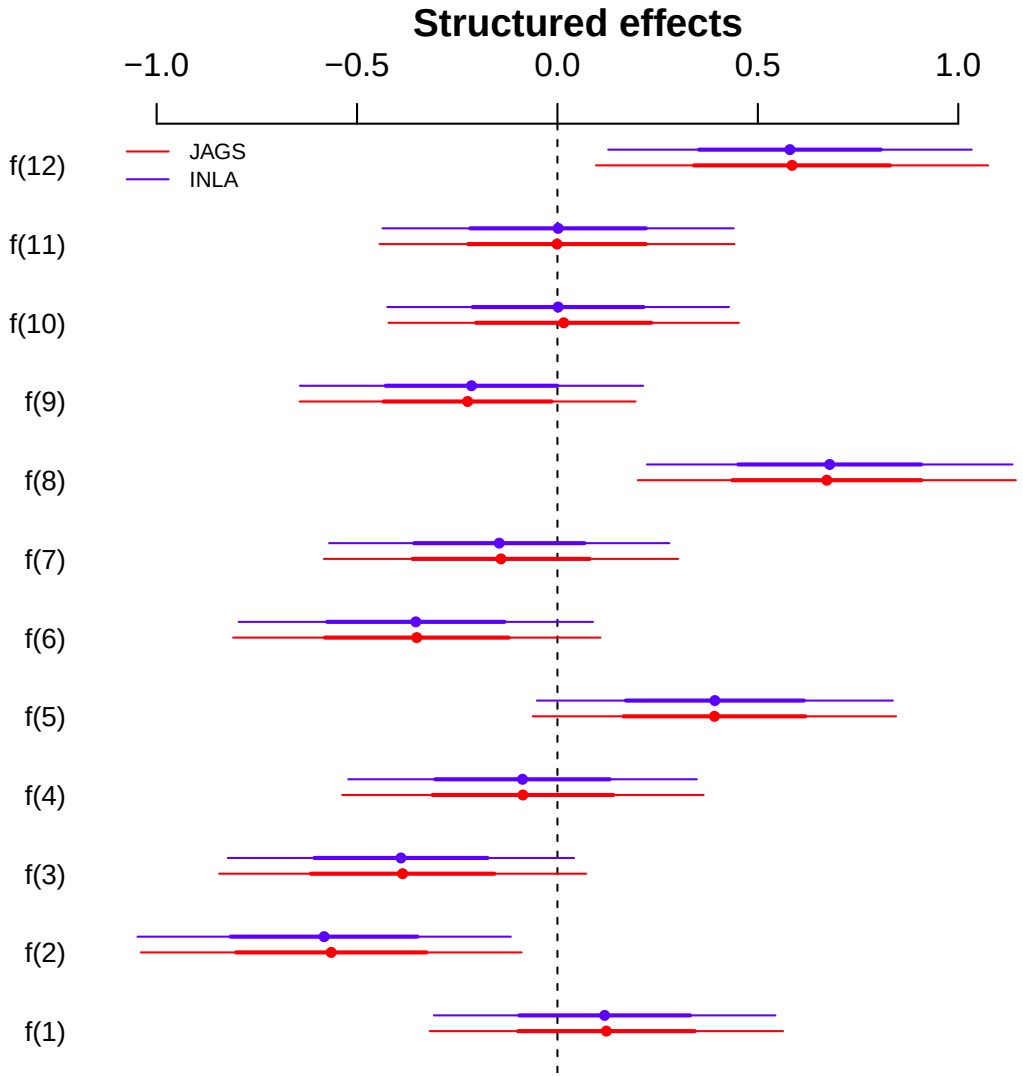
```
model {
    for (i in 1:n) {
        y[i] ~ dbinom(pi[i],Ntrials[i])
        logit(pi[i]) <- alpha+f[i]
        f[i] ~ dnorm(0,tau)
    }
    alpha ~ dnorm(0,.001)
    log.sigma ~ dunif(0,10000)
    sigma <- exp(log.sigma)
    tau <- pow(sigma,-2)
}
```

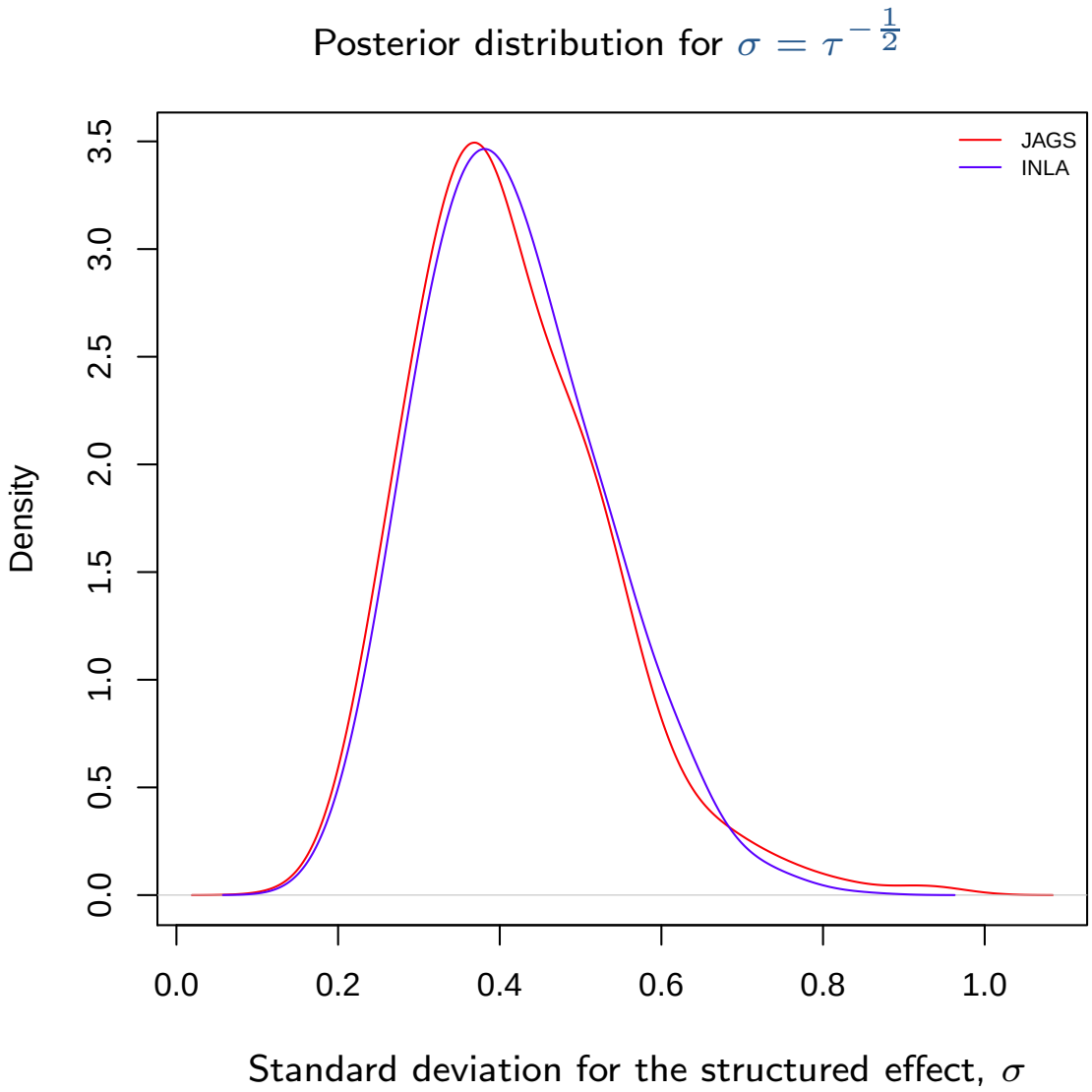
2. In R, use the library R2jags (or R2WinBUGS) to interface with the MCMC software

```
library(R2jags)
filein <- "model.txt"
dataJags <- list(y=y,n=n,Ntrials=Ntrials,prec=prec)
params <- c("sigma","tau","f","pi","alpha")
inits <- function(){ list(log.sigma=runif(1),alpha=rnorm(1),f=rnorm(n,0,1)) }
n.iter <- 100000
n.burnin <- 9500
n.thin <- floor((n.iter-n.burnin)/500)
mj <- jags(dataJags, inits, params, model.file=filein,n.chains=2, n.iter, n.burnin,
    n.thin, DIC=TRUE, working.directory=working.dir, progress.bar="text")
print(mj,digits=3,intervals=c(0.025, 0.975))
```

Example — Binary data with individual random effect

```
Inference for Bugs model at "model.txt", fit using jags,
 2 chains, each with 1e+05 iterations (first 9500 discarded), n.thin = 181
 n.sims = 1000 iterations saved                    (Time to run: 4.918 sec)

          mu.vect sd.vect   2.5%  97.5%  Rhat n.eff
alpha      -0.005   0.146 -0.270  0.292 1.001  1000
f[1]        0.122   0.220 -0.347  0.582 1.001  1000
f[2]       -0.564   0.238 -1.051 -0.115 1.008   190
f[3]       -0.386   0.229 -0.880  0.050 1.000  1000
f[4]       -0.086   0.225 -0.549  0.367 1.002   780
f[5]        0.392   0.227 -0.047  0.828 1.002   870
f[6]       -0.351   0.229 -0.805  0.081 1.000  1000
f[7]       -0.141   0.221 -0.578  0.286 1.001  1000
f[8]        0.672   0.236  0.246  1.200 1.002   860
f[9]       -0.224   0.210 -0.643  0.178 1.000  1000
f[10]       0.016   0.219 -0.396  0.463 1.006  1000
f[11]      -0.001   0.221 -0.441  0.416 1.002   780
f[12]       0.585   0.245  0.153  1.093 1.001  1000
sigma       0.414   0.120  0.230  0.693 1.000  1000
tau         7.415   4.546  2.080 18.951 1.000  1000
deviance   72.378   5.497 64.016 84.715 1.000  1000

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 15.1 and DIC = 87.5
DIC is an estimate of expected predictive error (lower deviance is better).
```


Structured effects
−1.0
−0.5
0.0
0.5
1.0
JAGS
INLA
f(12)
f(11)
f(10)
f(9)
f(8)
f(7)
f(6)
f(5)
f(4)
f(3)
f(2)
f(1)


Posterior distribution for $\sigma = \tau^{-\frac{1}{2}}$
JAGS
INLA
Density
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0.0
0.2
0.4
0.6
0.8
1.0
Standard deviation for the structured effect, σ

Example — Binary data with individual random effect

- R-INLA allows to make predictive inference based on the observed model
- Suppose for example that the $(n+1)-$th value is not (yet) observed for the response variable y
  - **NB**: for R-INLA, a missing value in the response means no likelihood contribution

Example — Binary data with individual random effect

- R-INLA allows to make predictive inference based on the observed model
- Suppose for example that the $(n+1)-$th value is not (yet) observed for the response variable y
  - **NB**: for R-INLA, a missing value in the response means no likelihood contribution

- We can code this in R, by augmenting the original dataset

```
y[n+1] <- NA
Ntrials[n+1] <- sample(c(80:100),size=1,replace=TRUE)
data2 <- data.frame(y=y,z=1:(n+1),Ntrials=Ntrials)

formula2 = y ~ f(z,model="iid",hyper=list(list(prior="flat")))
m2=inla(formula2,data=data2,
        family="binomial",
        Ntrials=Ntrials,
        control.predictor = list(compute = TRUE))

summary(m2)
```

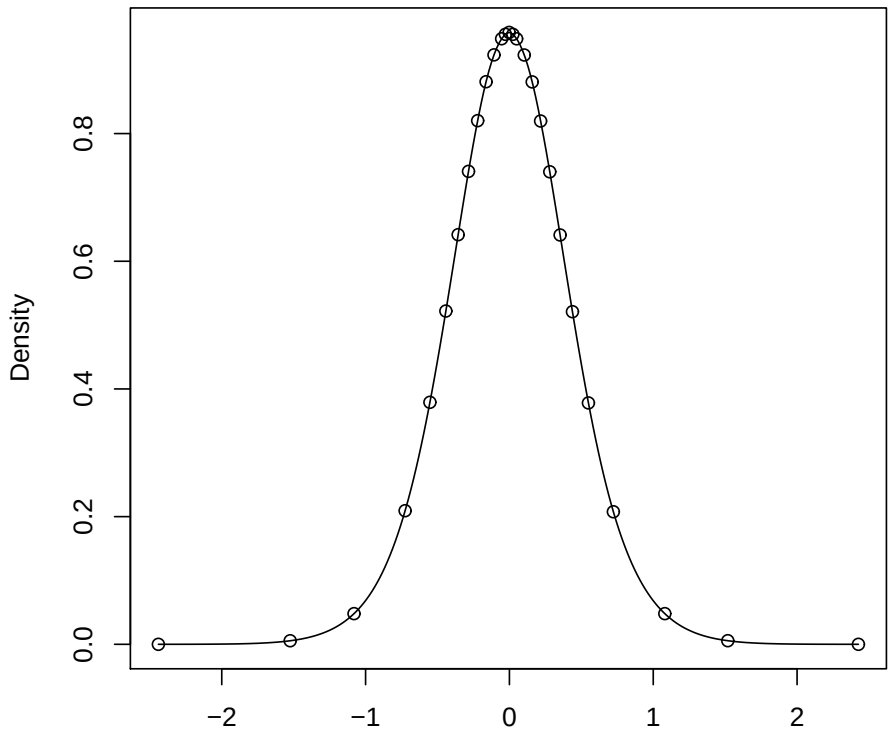
Example — Binary data with individual random effect

```
Time used:
 Pre-processing    Running inla Post-processing           Total
         0.0883          0.0285          0.0236          0.1404
        (0.2258)        (0.0263)        (0.0744)        (0.3264)

 Fixed effects:
               mean      sd  0.025quant  0.5quant  0.975quant  kld
(Intercept) -0.0021   0.136      -0.272   -0.0021       0.268    0
           (-0.0021) (0.136)    (-0.272) (-0.0021)     (0.268)  (0)

Random effects:
Name   Model
 z   IID model

Model hyperparameters:
                mean     sd     0.025quant 0.5quant 0.975quant
Precision for z  7.130   4.087   2.168      6.186    17.599
                (7.130) (4.087) (2.168)    (6.168) (17.599)

Expected number of effective parameters(std dev): 9.494(0.7925)
Number of equivalent replicates : 1.264

Marginal Likelihood:  -54.28
CPO and PIT are computed

Posterior marginals for linear predictor and fitted values computed
```

- The estimated value for the predictive distribution can be retrieved using the following code

```
pred <- m2$marginals.linear.predictor[[n+1]]
plot(pred,xlab="",ylab="Density")
lines(inla.smarginal(pred))
```

which can be used to generate, eg a graph of the predictive density



Specifying the model — options

- It is possible to specify link functions that are different from the default used by R-INLA
- This is done by specifying suitable values for the option control.family to the call to inla, eg

```
m = inla(formula, data=data, family="binomial", Ntrials=Ntrials,
         control.predictor=list(compute=TRUE),
         control.family = list(link = "probit"))
```

Specifying the model — options

- It is possible to specify link functions that are different from the default used by R-INLA
- This is done by specifying suitable values for the option control.family to the call to inla, eg

```
m = inla(formula, data=data, family="binomial", Ntrials=Ntrials,
         control.predictor=list(compute=TRUE),
         control.family = list(link = "probit"))
```

- More details are available on the R-INLA website:
  - http://www.r-inla.org/models/likelihoods

Specifying the model — options

- R-INLA has a set of default priors for the different components of the LGM/GMRF
- For example, in a standard hierarchical formulation, R-INLA assumes
  - Unstructured ("fixed") effects: $\beta \sim \text{Normal}(0, 0.001)$
  - Structured ("random") effects: $f(z_i) \sim \text{Normal}(0, \tau)$
    $\log \tau \sim \text{logGamma}(1, 0.00005)$
- **NB**: It is possible to see the default settings using the function

```
inla.model.properties(<name>, <section>)
```

Specifying the model — options

- R-INLA has a set of default priors for the different components of the LGM/GMRF
- For example, in a standard hierarchical formulation, R-INLA assumes
  - Unstructured ("fixed") effects: $\beta \sim \text{Normal}(0, 0.001)$
  - Structured ("random") effects: $f(z_i) \sim \text{Normal}(0, \tau)$
    $\log \tau \sim \text{logGamma}(1, 0.00005)$
- **NB**: It is possible to see the default settings using the function

```
inla.model.properties(<name>, <section>)
```

- However, there is a wealth of possible formulations that the user can specify, especially for the hyperpriors
- More details are available on the R-INLA website:
  - http://www.r-inla.org/models/likelihoods
  - http://www.r-inla.org/models/latent-models
  - http://www.r-inla.org/models/priors

Models for the observed data

Model	Name
Negative Binomial	nbinomial
Poisson	poisson
Binomial	binomial
Clustered Binomial	cbinomial
Gaussian	gaussian
Skew Normal	sn
Laplace	laplace
Student-t	T
Gaussian model for stochastic volatility	stochvol
Student-t model for stochastic volatility	stochvol.t
NIG model for stochastic volatility	stochvol.nig
Zero inflated Poisson	zeroinflated.poisson.x (x=0,1,2)
Zero inflated Binomial	zeroinflated.binomial.x (x=0,1)
Zero inflated negative Binomial	zeroinflated.nbinomial.x (x=0,1,2)
Zero inflated beta binomial (type 2)	zeroinflated.betabinomial.2
Generalised extreme value distribution (GEV)	gev
Beta	beta
Gamma	gamma
Beta-Binomial	betabinomial
Logistic distribution	logistic
Exponential (Survival models)	exponential
Weibull (Survival model)	weibull
LogLogistic (Survival model)	loglogistic
LogNormal (Survival model)	lognormal
Cox model (Survival model)	coxph

Models for the GMRF

Model	Name
Independent random variables	iid
Linear	linear
Random walk of order 1	rw1
Random walk of order 2	rw2
Continuous random walk of order 2	crw2
Model for seasonal variation	seasonal
Model for spatial effect	besag
Model for spatial effect	besagproper
Model for weighted spatial effects	besag2
Model for spatial effect + random effect	bym
Autoregressive model of order 1	ar1
Autoregressive model of order p	ar
The Ornstein-Uhlenbeck process	ou
User defined structure matrix, type 0	generic0
User defined structure matrix, type1	generic1
User defined structure matrix, type2	generic2
Model for correlated effects with Wishart prior (dimension 1, 2, 3, 4 and 5).	iid1d, iid2d, iid3d, iid4d, iid5d
(Quite) general latent model	z
Random walk of 2nd order on a lattice	rw2d
Gaussian field with Matern covariance function	matern2d
Classical measurement error model	mec
Berkson measurement error model	meb

Models for the hyper-parameters

Model	Name
Normal distribution	normal, gaussian
Log-gamma distribution	loggamma
Improper flat prior	flat
Truncated Normal distribution	logtnormal, logtgaussian
Improper flat prior on the log scale	logflat
Improper flat prior on the 1/ log scale	logiflat
Wishart prior	wishart
Beta for correlations	betacorrelation
Logit of a Beta	logitbeta
Define your own prior	expression:

Internal vs user scale

- Hyper-parameters (eg correlation coefficients ρ or precisions τ) are represented internally using a suitable transformation, eg

$$\psi_1 = \log(\tau)$$

or

$$\psi_2 = \log\left(\frac{1+\rho}{1-\rho}\right)$$

to improve symmetry and approximate Normality

Internal vs user scale

- Hyper-parameters (eg correlation coefficients ρ or precisions τ) are represented internally using a suitable transformation, eg

$$\psi_1 = \log(\tau)$$

or

$$\psi_2 = \log\left(\frac{1+\rho}{1-\rho}\right)$$

to improve symmetry and approximate Normality

- Initial values are given on the internal scale
- Priors are also defined on the internal scale
- So, when specifying custom values, care is needed!

- Current work on a more general class of priors — "penalised complexity priors" (see `http://arxiv.org/pdf/1403.4630v1.pdf`)

Specifying the prior (1)

Consider the model

$$
\begin{aligned}
y_i \mid \theta_i, \sigma^2 &\sim \text{Normal}(\theta_i, \sigma^2) \\
\theta_i &= \alpha + \beta x_i \\
\alpha, \beta &\overset{iid}{\sim} \text{Normal}(0, 0.001) \\
\log \tau = -\log \sigma^2 &\sim \text{logGamma}(1, 0.01)
\end{aligned}
$$

```
n=100
a = 1; b = 1
x = rnorm(n)
eta = a + b*x
tau = 100
scale = exp(rnorm(n))
prec = scale*tau
y = rnorm(n, mean = eta, sd = 1/sqrt(prec))
data = list(y=y, x=x)
formula = y ~ 1 + x
result = inla(formula, family = "gaussian", data = data,
              control.family = list(hyper = list(
                 prec = list(prior = "loggamma",param = c(1,0.01),initial = 2))),
              scale=scale, keep=TRUE)
summary(result)
```

Specifying the prior (1)

```
Time used:
 Pre-processing    Running inla Post-processing           Total
         0.0776          0.0828          0.0189          0.1793

Fixed effects:
              mean     sd 0.025quant 0.5quant 0.975quant kld
(Intercept) 1.0013 0.0074     0.9868   1.0013     1.0158   0
x           0.9936 0.0075     0.9788   0.9936     1.0083   0

The model has no random effects

Model hyperparameters:
                                          mean    sd    0.025quant 0.5quant 0.975quant
Precision for the Gaussian observations 108.00 15.34 80.60      107.09   140.74

Expected number of effective parameters(std dev): 2.298(0.0335)
Number of equivalent replicates : 43.52

Marginal Likelihood:  83.86
CPO and PIT are computed
```

Specifying the prior (2)

Consider the model

$$
\begin{aligned}
y_i \mid \mu, \sigma^2 &\sim \text{Normal}(\mu, \sigma^2) \\
\mu &\sim \text{Normal}(0, 0.001) \\
\log \tau = -\log \sigma^2 &\sim \text{Normal}(0, 1)
\end{aligned}
$$

```
n = 10
y = rnorm(n)
formula = y ~ 1
result = inla(formula, data = data.frame(y),
              control.family = list(hyper = list(
                 prec = list(prior = "normal",param = c(0,1))))
)
summary(result)
```

Specifying the prior (2)

Consider the model

$$\begin{aligned} y_i \mid \mu, \sigma^2 &\sim \text{Normal}(\mu, \sigma^2) \\ \mu &\sim \text{Normal}(0, 0.001) \\ \log \tau = -\log \sigma^2 &\sim \text{Normal}(0, 1) \end{aligned}$$

```
n = 10
y = rnorm(n)
formula = y ~ 1
result = inla(formula, data = data.frame(y),
              control.family = list(hyper = list(
                 prec = list(prior = "normal",param = c(0,1))))
)
summary(result)
```

- **NB**: INLA thinks in terms of LGMs and GMRFs
- Thus, the common mean for all the observations is specified in terms of a regression!

Specifying the prior (2)

```
Time used:
 Pre-processing    Running inla Post-processing           Total
         0.0740          0.0214          0.0221          0.1175

Fixed effects:
               mean     sd 0.025quant 0.5quant 0.975quant kld
(Intercept) -0.3853 0.4077    -1.1939  -0.3853     0.4237   0

The model has no random effects

Model hyperparameters:
                                          mean    sd    0.025quant 0.5quant 0.975quant
Precision for the Gaussian observations 0.6512 0.268 0.2590     0.6089   1.2919

Expected number of effective parameters(std dev): 1.00(0.00)
Number of equivalent replicates : 9.999

Marginal Likelihood:  -17.30
CPO and PIT are computed
```

Specifying the prior (2)

Running the model in JAGS

```
model {
    for (i in 1:n) {
        y[i] ~ dnorm(mu,tau)
    }
    mu ~ dnorm(0,0.001)
    log.tau ~ dnorm(0,1)
    tau <- exp(log.tau)
}
```

produces similar results

```
Inference for Bugs model at "modelHyperPriorNormal.txt", fit using jags,
 2 chains, each with 1e+05 iterations (first 9500 discarded), n.thin = 181
 n.sims = 1000 iterations saved
         mu.vect sd.vect   2.5%  97.5%  Rhat n.eff
mu        -0.384   0.447 -1.293  0.555 1.000  1000
tau        0.642   0.258  0.233  1.240 1.006  1000
deviance  34.507   1.930 32.645 39.897 1.002   650

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 1.9 and DIC = 36.4
DIC is an estimate of expected predictive error (lower deviance is better).
```

Specifying the prior (2)

- We can also assume different priors for the unstructured ("fixed") effects, eg suppose we want to fit the model

$$\begin{aligned} y_i \mid \mu, \sigma^2 &\sim \text{Normal}(\mu, \sigma^2) \\ \mu &\sim \text{Normal}(10, 4) \\ \log \tau = -\log \sigma^2 &\sim \text{Normal}(0, 1) \end{aligned}$$

Specifying the prior (2)

- We can also assume different priors for the unstructured ("fixed") effects, eg suppose we want to fit the model

$$\begin{aligned} y_i \mid \mu, \sigma^2 &\sim \text{Normal}(\mu, \sigma^2) \\ \mu &\sim \text{Normal}(10, 4) \\ \log \tau = -\log \sigma^2 &\sim \text{Normal}(0, 1) \end{aligned}$$

- This can be done by using the option `control.fixed`, eg

```
result = inla(formula, data = data.frame(y),
              control.family = list(hyper = list(
                 prec = list(prior = "normal",param = c(0, 1))))
              control.fixed=list(
                 mean.intercept=10,prec.intercept=4)
)
```

Specifying the prior (2)

- We can also assume different priors for the unstructured ("fixed") effects, eg suppose we want to fit the model

$$\begin{aligned} y_i \mid \mu, \sigma^2 &\sim \text{Normal}(\mu, \sigma^2) \\ \mu &\sim \text{Normal}(10, 4) \\ \log \tau = -\log \sigma^2 &\sim \text{Normal}(0, 1) \end{aligned}$$

- This can be done by using the option `control.fixed`, eg

```
result = inla(formula, data = data.frame(y),
              control.family = list(hyper = list(
                 prec = list(prior = "normal",param = c(0, 1))))
              control.fixed=list(
                 mean.intercept=10,prec.intercept=4)
)
```

- **NB**: If the model contains fixed effects for some covariates, non-default priors can be included using the option

```
control.fixed=list(mean=list(value),prec=list(value))
```

Specifying the prior (2)

```
Time used:
 Pre-processing    Running inla Post-processing           Total
         0.0747          0.0311          0.0164          0.1222

Fixed effects:
              mean    sd 0.025quant 0.5quant 0.975quant kld
(Intercept) 9.5074 0.502     8.5249   9.5067    10.4935   0
           -0.3853 0.407    -1.1939  -0.3853     0.4237   0

The model has no random effects

Model hyperparameters:
                                          mean   sd    0.025quant 0.5quant 0.975quant
Precision for the Gaussian observations 0.0218 0.007 0.0105     0.0208   0.0391
                                        0.6512 0.268 0.2590     0.6089   1.2919

Expected number of effective parameters(std dev): 0.0521(0.0129)
                                                  1.0000(0.0000)
Number of equivalent replicates : 192.05
                                  9.9999

Marginal Likelihood:  153.84
                     -17.30
CPO and PIT are computed
```

Improving the estimation of the hyperparameters

- As mentioned earlier, for computational reasons, by default INLA uses a relatively rough grid to estimate the marginal posterior for the hyperparameters $p(\boldsymbol{\psi} \mid \boldsymbol{y})$
- This is generally good enough, but the procedure can be refined

Improving the estimation of the hyperparameters

- As mentioned earlier, for computational reasons, by default INLA uses a relatively rough grid to estimate the marginal posterior for the hyperparameters $p(\boldsymbol{\psi} \mid \boldsymbol{y})$
- This is generally good enough, but the procedure can be refined

- After the model has been estimated using the standard procedure, it is possible to increase precision in the estimation by re-fitting it using the command

```
inla.hyperpar(m, options)
```

- This modifies the estimation for the hyperparameters and (potentially, but not necessarily!) that for the parameters

A more complex model

- Consider the classic model for seizure counts in a RCT of anti-conversant therapy in epilepsy ("Epil" in the BUGS manual)
- The data are as follows

Patient	Visit 1	Visit 2	Visit 3	Visit 4	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

- We replicate the model presented in the BUGS manual, which uses slightly modified version of the covariates

A more complex model

We model

$$
\begin{aligned}
y_{jk} &\sim \text{Poisson}(\mu_{jk}) \\
\log(\mu_{jk}) &= \alpha_0 + \alpha_1 \log(B_j/4) + \alpha_2 Trt_j + \\
&\quad \alpha_3 Trt_j \times \log(B_j/4) + \alpha_4 \log(Age_j) + \\
&\quad \alpha_5 V4_k + u_j + v_{ik} \\
\alpha_0, \ldots \alpha_5 &\overset{iid}{\sim} \text{Normal}(0, \tau_\alpha), \qquad \tau_\alpha \text{ known} \\
u_j &\sim \text{Normal}(0, \tau_u), \qquad \tau_u \sim \text{Gamma}(a_u, b_u) \\
v_{jk} &\sim \text{Normal}(0, \tau_v), \qquad \tau_v \sim \text{Gamma}(a_v, b_v)
\end{aligned}
$$

A more complex model

We model

$$
\begin{aligned}
y_{jk} &\sim \text{Poisson}(\mu_{jk}) \\
\log(\mu_{jk}) &= \alpha_0 + \alpha_1 \log(B_j/4) + \alpha_2 Trt_j + \\
&\quad \alpha_3 Trt_j \times \log(B_j/4) + \alpha_4 \log(Age_j) + \\
&\quad \alpha_5 V4_k + u_j + v_{ik} \\
\alpha_0, \ldots \alpha_5 &\overset{iid}{\sim} \text{Normal}(0, \tau_\alpha), \qquad \tau_\alpha \text{ known} \\
u_j &\sim \text{Normal}(0, \tau_u), \qquad \tau_u \sim \text{Gamma}(a_u, b_u) \\
v_{jk} &\sim \text{Normal}(0, \tau_v), \qquad \tau_v \sim \text{Gamma}(a_v, b_v)
\end{aligned}
$$

$\boldsymbol{\alpha} = (\alpha_0, \ldots \alpha_5)$ indicates a set of "fixed" effects for the relevant (re-scaled) covariates

A more complex model

We model

$$
\begin{aligned}
y_{jk} &\sim \text{Poisson}(\mu_{jk}) \\
\log(\mu_{jk}) &= \alpha_0 + \alpha_1 \log(B_j/4) + \alpha_2 Trt_j + \\
&\quad \alpha_3 Trt_j \times \log(B_j/4) + \alpha_4 \log(Age_j) + \\
&\quad \alpha_5 V4_k + u_j + v_{ik} \\
\alpha_0, \ldots \alpha_5 &\overset{iid}{\sim} \text{Normal}(0, \tau_\alpha), \qquad \tau_\alpha \text{ known} \\
u_j &\sim \text{Normal}(0, \tau_u), \qquad \tau_u \sim \text{Gamma}(a_u, b_u) \\
v_{jk} &\sim \text{Normal}(0, \tau_v), \qquad \tau_v \sim \text{Gamma}(a_v, b_v)
\end{aligned}
$$

$\boldsymbol{\alpha} = (\alpha_0, \ldots \alpha_5)$ indicates a set of "fixed" effects for the relevant (re-scaled) covariates

u_j is an individual "random" effect

A more complex model

We model

$$
\begin{aligned}
y_{jk} &\sim \text{Poisson}(\mu_{jk}) \\
\log(\mu_{jk}) &= \alpha_0 + \alpha_1 \log(B_j/4) + \alpha_2 Trt_j + \\
&\quad \alpha_3 Trt_j \times \log(B_j/4) + \alpha_4 \log(Age_j) + \\
&\quad \alpha_5 V4_k + u_j + v_{ik} \\
\alpha_0, \ldots \alpha_5 &\overset{iid}{\sim} \text{Normal}(0, \tau_\alpha), \qquad \tau_\alpha \text{ known} \\
u_j &\sim \text{Normal}(0, \tau_u), \qquad \tau_u \sim \text{Gamma}(a_u, b_u) \\
v_{jk} &\sim \text{Normal}(0, \tau_v), \qquad \tau_v \sim \text{Gamma}(a_v, b_v)
\end{aligned}
$$

$\boldsymbol{\alpha} = (\alpha_0, \ldots \alpha_5)$ indicates a set of "fixed" effects for the relevant (re-scaled) covariates

u_j is an individual "random" effect

v_{jk} is a subject by visit "random" effect, which accounts for extra-Poisson variability

A more complex model

```
data(Epil)
head(Epil,n=3)
     y Trt Base Age V4 rand Ind
   1 5   0   11  31  0    1   1
   2 3   0   11  31  0    2   1
   3 3   0   11  31  0    3   1

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

m <- inla(formula, family="poisson", data = Epil)
```

- **NB**: The variable `Ind` indicates the individual random effect u_j, while the variable `rand` is used to model the subject by visit random effect v_{jk}
- Interactions can be indicated in the R formula using the notation

  `I(var1 * var2)`

- The model assumes that the two structured effects are independent. This can be relaxed and a joint model can be used instead

A more complex model

```
 Pre-processing    Running inla Post-processing           Total
         0.3672          0.2780           0.1276          0.7728

Fixed effects:
                       mean     sd 0.025quant 0.5quant 0.975quant    kld
(Intercept)         -1.3877 1.2107    -3.7621  -1.3913     1.0080 0.0055
log(Base/4)          0.8795 0.1346     0.6144   0.8795     1.1447 0.0127
Trt                 -0.9524 0.4092    -1.7605  -0.9513    -0.1498 0.0021
I(Trt * log(Base/4)) 0.3506 0.2081    -0.0586   0.3504     0.7611 0.0011
log(Age)             0.4830 0.3555    -0.2206   0.4843     1.1798 0.0007
V4                  -0.1032 0.0853    -0.2705  -0.1032     0.0646 0.0003

Random effects:
Name    Model
 Ind    IID model
rand    IID model

Model hyperparameters:
                  mean    sd     0.025quant 0.5quant 0.975quant
Precision for Ind   4.635  1.343  2.591       4.436     7.808
Precision for rand  8.566  2.115  5.206       8.298    13.458

Expected number of effective parameters(std dev): 118.97(8.586)
Number of equivalent replicates : 1.984

Marginal Likelihood:  -670.55
```

A more complex model

```
 Pre-processing    Running inla Post-processing           Total
         0.3672          0.2780           0.1276          0.7728
                                                 (MCMC: approximately 30 mins)
Fixed effects:
                       mean     sd 0.025quant 0.5quant 0.975quant    kld
(Intercept)         -1.3877 1.2107    -3.7621  -1.3913     1.0080 0.0055
log(Base/4)          0.8795 0.1346     0.6144   0.8795     1.1447 0.0127
Trt                 -0.9524 0.4092    -1.7605  -0.9513    -0.1498 0.0021
I(Trt * log(Base/4)) 0.3506 0.2081    -0.0586   0.3504     0.7611 0.0011
log(Age)             0.4830 0.3555    -0.2206   0.4843     1.1798 0.0007
V4                  -0.1032 0.0853    -0.2705  -0.1032     0.0646 0.0003

Random effects:
Name    Model
 Ind    IID model
rand    IID model

Model hyperparameters:
                  mean    sd     0.025quant 0.5quant 0.975quant
Precision for Ind   4.635  1.343  2.591       4.436     7.808
Precision for rand  8.566  2.115  5.206       8.298    13.458

Expected number of effective parameters(std dev): 118.97(8.586)
Number of equivalent replicates : 1.984

Marginal Likelihood:  -670.55
```

Conclusions

- Integrated Nested Laplace Approximation is a very effective tool to estimate LGMs
  - Estimation time can be much lower than for standard MCMC
  - Precision of estimation is usually higher than for standard MCMC

Conclusions

- Integrated Nested Laplace Approximation is a very effective tool to estimate LGMs
  - Estimation time can be much lower than for standard MCMC
  - Precision of estimation is usually higher than for standard MCMC
- MCMC still provides a slightly more flexible approach
  - Virtually any model can be fit using `JAGS`/`BUGS`
  - The range of priors available is wider in an MCMC setting than in `INLA`
  - Documentation and examples is more extensive for standard MCMC models

Conclusions

- Integrated Nested Laplace Approximation is a very effective tool to estimate LGMs
  - Estimation time can be much lower than for standard MCMC
  - Precision of estimation is usually higher than for standard MCMC
- MCMC still provides a slightly more flexible approach
  - Virtually any model can be fit using `JAGS/BUGS`
  - The range of priors available is wider in an MCMC setting than in `INLA`
  - Documentation and examples is more extensive for standard MCMC models
- Nevertheless, `INLA` proves to be a very flexible tool, which is able to fit a very wide range of models
  - Generalised linear (mixed) models
  - Log-Gaussian Cox processes
  - Survival analysis
  - Spline smoothing
  - Spatio-temporal models
- The `INLA` setup can be highly specialised (choice of data models, priors and hyperpriors) although this is a bit less intuitive than (most) MCMC models

Thank you!