Bayesian Data Analysis

Daniel Paulin & Nicolò Margaritella

University of Edinburgh

Semester 2, 2020/2021

1. Introduction to INLA
   - Latent Gaussian Models (LGM)
   - Conditional probabilities
   - Laplace approximation
   - Integrated Nested Laplace Approximation

2. R-INLA Package
   - General information
   - Examples
   - Posterior sampling in INLA
   - Model checking in INLA

Happy Lunar New Year of the Ox!

Latent Gaussian Models (LGM)

- Consider the general inference problem assuming a probability model for the observed data as a function of some relevant model parameters $\boldsymbol{x}$ and θ,

$$\boldsymbol{y}|\boldsymbol{x},\theta \sim f(\boldsymbol{y}|\boldsymbol{x},\theta) = \prod_{i=1}^{n} f(y_i|\boldsymbol{x},\theta).$$

- For a surprisingly large class of models, it is sensible to choose a prior distribution such that

$$\boldsymbol{x}|\theta \sim \pi(\boldsymbol{x}|\theta) = N(0,\Sigma(\theta)) = N(0,\boldsymbol{\tau}(\theta)^{-1}),$$

where the precision matrix $\boldsymbol{\tau}(\theta) = \Sigma(\theta)^{-1}$ is often sparse (i.e. most of its elements are zero). Very efficient computations in high dimensions based on sparse Cholesky factorization.

- Multivariate normal distributions with sparse precision matrices are called Gaussian Markov Random Fields (GMRF).

Latent Gaussian Models (LGM)

- $\boldsymbol{\theta}$: vector of hyperparameters, $\boldsymbol{x}$: vector of latent effects.
- In general, we can partition $\boldsymbol{\theta} = (\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)})$ as follows.

$$
\begin{aligned}
\boldsymbol{\theta} &\sim \pi(\boldsymbol{\theta}) && \text{(Hyperprior)} \\
\boldsymbol{x}|\boldsymbol{\theta} &\sim \pi(\boldsymbol{x}|\boldsymbol{\theta}) = N(0, \boldsymbol{\Sigma}(\boldsymbol{\theta}^{(2)})) && \text{(GMRF prior)} \\
\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{x} &\sim f(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{i=1}^{n} f(y_i|\boldsymbol{x}, \boldsymbol{\theta}^{(1)}) && \text{(data model)}
\end{aligned}
$$

- Hence the data model only depends on $\boldsymbol{\theta}^{(1)}$, while the GMRF prior only depends on $\boldsymbol{\theta}^{(2)}$. Usually we will just write $\boldsymbol{\theta}$ everywhere for simplicity.
- The dimension of $\boldsymbol{\theta}$ is small (up to 15), while the dimension of $\boldsymbol{x}$ can be large (up to 10^{12}).

Latent Gaussian Models (LGM)

- Let $\mu_i = \mathbb{E}(y_i|\boldsymbol{\theta}, \boldsymbol{x})$ be the mean of the observation i given the model parameters, for $1 \leq i \leq n$. For a large class of models (called GLMs), this mean is connected to another random variable η_i called the linear predictor of observation i by an invertible link function g, i.e.

$$\eta_i = g(\mu_i), \quad \mu_i = g^{-1}(\eta_i), \quad \eta_i = \beta_0 + \sum_{j=1}^{n_\beta} \beta_j z_{ji} + \sum_{k=1}^{n_f} f^{(k)}(u_{ki}) + \epsilon_i, \qquad \text{where}$$

- $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)$ are error terms, assumed to be Gaussian (and usually i.i.d.).
- $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_n)$ is the vector of linear predictors.
- β_0 is the intercept.
- $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_{n_\beta})$ is the regression coefficient quantifying the effect of the covariates $\boldsymbol{z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_\beta})$ on the linear predictor η_i ($\boldsymbol{z}_j = (z_{j1}, z_{j2}, \ldots, z_{jn})$ collects the values of covariate j for all of the n observations).
- $\boldsymbol{f} = (\boldsymbol{f}^{(1)}, \ldots, \boldsymbol{f}^{(n_f)}) = (f^{(1)}(u_{1,1}), f^{(1)}(u_{12}), \ldots, f^{(n_f)}(u_{n_f n}))$ is a set of random functions (Gaussian random fields) defined in terms of some covariates $\boldsymbol{u} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n_f})$.
- These parts form the vector of latent effects $\boldsymbol{x} = (\eta, \boldsymbol{\beta}, \boldsymbol{f})$. It is assumed that

$$\boldsymbol{x}|\boldsymbol{\theta} \sim \pi(\boldsymbol{x}|\boldsymbol{\theta}) = N(0, \boldsymbol{\Sigma}(\boldsymbol{\theta})).$$

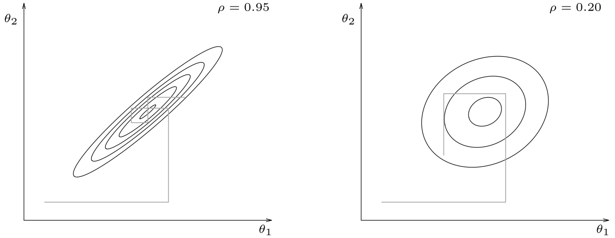
Latent Gaussian Models (LGM)

By varying the form of the functions $f^{(k)}$, this framework can accommodate a wide range of models

- $\hookrightarrow$ Standard regression ($f^{(k)}(\cdot) = 0$).
- $\hookrightarrow$ Hierarchical models
- $\hookrightarrow$ Spatial models
- $\hookrightarrow$ Temporal models
- $\hookrightarrow$ Spatio-temporal models
- $\hookrightarrow$ Spline smoothing
- $\hookrightarrow$ Survival models / log-Gaussian Cox processes

MCMC for LGMs

- ↪ MCMC methods can become very slow when applied to high dimensional LGMs.
- ↪ This is due to the correlations between the components of the latent Gaussian field $\boldsymbol{x}$.
- ↪ When the number of observations n is large, the latent Gaussian field and the hyperparameter vector $\boldsymbol{\theta}$ also become highly correlated.
- ↪ Such correlations can slow down the convergence of MCMC methods.
- ↪ Centering, standardization, extending the parameter space (in a problem specific way), and blocking can somewhat speed up convergence. However, they require some additional effort, and may be difficult to scale in very high dimensions.
- ↪ INLA allows to perform Bayesian inference for high dimensional LGMs by doing some clever approximations, exploiting the sparsity of the precision matrices, and using efficient optimization methods.



Conditional Probabilities

- Let X, Z, W be random variables defined on the same probability space.
- Suppose that these random variables have densities (with respect to the Lebesgue measure), and their densities are denoted by $p(x)$, $p(z)$ and $p(w)$, respectively.
- Similarly, we denote the joint densities of X and Z by $p(x, z)$, etc.
- By the definition of conditional probability, assuming that $p(z) > 0$, we have (almost surely)

$$p(x|z) := \frac{p(x,z)}{p(z)}, \text{ and consequently,}$$

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

- Similarly, a conditioned version can also be shown to hold. Assuming that $p(w) > 0$ and $p(z, w) > 0$, we have

$$p(z|w) := \frac{p(x,z|w)}{p(x|z,w)}. \tag{1}$$

This form is very useful for constructing approximations for Bayesian inference.

Laplace Approximation

- The second key tool in the INLA approach is Laplace approximation.

- Main idea: for some unnormalized probability density $q(x)$, we we approximate $\log q(x)$ by a quadratic function centered at the mode $\hat{x}$. By Taylor series expansion of order 2 around the mode $\hat{x}$, we have

$$\begin{aligned}\log q(x) &\approx \log q(\hat{x}) + (\log q)'(\hat{x})(x-\hat{x}) + \frac{1}{2}(\log q)''(\hat{x})(x-\hat{x})^2 \\ &= \log q(\hat{x}) + \frac{1}{2}(\log q)''(\hat{x})(x-\hat{x})^2,\end{aligned}$$

since $(\log q)'(\hat{x}) = 0$ because $\hat{x}$ is the mode. Let $\hat{\sigma}^2 := -((\log q)''(\hat{x}))^{-1}$, then

$$\log q(x) \approx \log q(\hat{x}) - \frac{1}{2\hat{\sigma}^2}(x-\hat{x})^2, \quad \text{hence} \quad q \approx N(\hat{x}, \hat{\sigma}^2).$$

- The integral of $q(x)$ can be approximated as

$$\int_x q(x)dx \approx q(\hat{x}) \int_x \exp\left(-\frac{1}{2\hat{\sigma}^2}(x-\hat{x})^2\right) dx = q(\hat{x})\sqrt{2\pi\hat{\sigma}^2}.$$

- Laplace approximation is also applicable in higher dimensions, with $\hat{\sigma}^2$ replaced by the covariance matrix $\hat{\Sigma} = -(\nabla^2(\log q)(\hat{x}))^{-1}$, and $\int_x q(x)dx \approx q(\hat{x})(2\pi)^{d/2}(\det\hat{\Sigma})^{1/2}$.

Laplace Approximation - Example

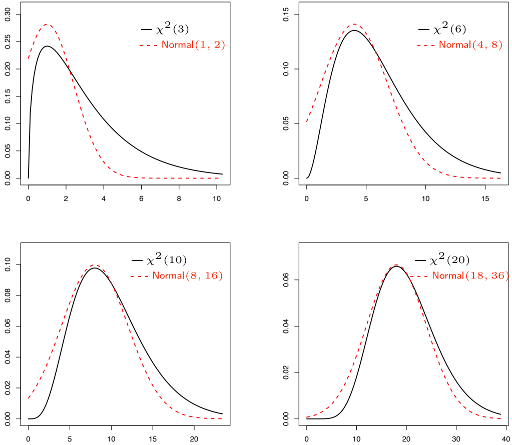
- Consider the χ^2 distribution with k d.o.f.: $p(x) = \frac{q(x)}{Z} = \frac{x^{\frac{k}{2}-1} \cdot e^{-\frac{x}{2}}}{Z}$ for $x \geq 0$.

$$\log q(x) = \left(\frac{k}{2} - 1\right) \log(x) - \frac{x}{2}$$
$$(\log q)'(x) = \left(\frac{k}{2} - 1\right) x^{-1} - \frac{1}{2}$$
$$(\log q)''(x) = -\left(\frac{k}{2} - 1\right) x^{-2}.$$

- We can find the mode analytically in this case by solving $(\log q)'(x) = 0$. This leads to $\hat{x} = k - 2$, and $\hat{\sigma}^2 = -((\log q)''(\hat{x}))^{-1} = 2(k-2)$.

- Thus the Laplace approximation in this case is $p \approx \tilde{p} = N(k-2, 2(k-2))$, i.e. a Normal distribution with mean $k-2$ and variance $2(k-2)$. This approximation is illustrated on the figures of the next slide.

Laplace Approximation - Example



Laplace approximation to the χ^2 distribution with varying d.o.f.

Integrated Nested Laplace Approximation (INLA)

- The general idea of INLA is to repeatedly use the approximation

$$p(z|w) \approx \frac{p(x, z|w)}{\tilde{p}(x|z, w)},$$

where $\tilde{p}(x|z, w)$ is the Laplace approximation to the conditional density $p(x|z, w)$.

- This approximation can be used for any x, but since the Laplace approximation is most accurate near the mode, we will always use this at $x = \hat{x}(z, w)$ (i.e. the mode of $p(x|z, w)$).

- This idea is quite broadly applicable, and significantly extends the usefulness of Laplace approximation (i.e. $p(z|w)$ does not have to be approximately Gaussian, only $p(x|z, w)$).

Integrated Nested Laplace Approximation (INLA)

↪ We are often interested in computing the marginals of the posterior distributions for LGMs. These can be written as

$$p(x_i|\boldsymbol{y}) = \int p(x_i, \boldsymbol{\theta}|\boldsymbol{y})d\boldsymbol{\theta} = \int p(\boldsymbol{\theta}|\boldsymbol{y})p(x_i|\boldsymbol{\theta}, \boldsymbol{y})d\boldsymbol{\theta},$$

$$p(\theta_j|\boldsymbol{y}) = \int p(\boldsymbol{\theta}|\boldsymbol{y})d\boldsymbol{\theta}_{-j}.$$

Here $\boldsymbol{\theta}_{-j} = \{\theta_k : k \neq j\}$ denotes the components of $\boldsymbol{\theta}$ excluding θ_j.

↪ In order to proceed, we will need to compute

(I) $p(\boldsymbol{\theta}|\boldsymbol{y})$, from which $p(\theta_j|\boldsymbol{y})$ can be obtained by integrating out $\boldsymbol{\theta}_{-j}$.

(II) $p(x_i|\boldsymbol{\theta}, \boldsymbol{y})$, which is needed for computing $p(x_i|\boldsymbol{y})$.

Integrated Nested Laplace Approximation (INLA)

$\hookrightarrow$ (I) can be estimated as

$$
\begin{aligned}
p(\theta|\boldsymbol{y}) &= \frac{p(\theta, \boldsymbol{x}|\boldsymbol{y})}{p(\boldsymbol{x}|\theta, \boldsymbol{y})} = \\
&= \frac{p(\theta, \boldsymbol{x}, \boldsymbol{y})}{m(\boldsymbol{y})} \cdot \frac{1}{p(\boldsymbol{x}|\theta, \boldsymbol{y})} \\
&\propto \frac{\pi(\theta)\pi(\boldsymbol{x}|\theta)f(\boldsymbol{y}|\theta, \boldsymbol{x})}{p(\boldsymbol{x}|\theta, \boldsymbol{y})} \\
&\approx \left.\frac{\pi(\theta)\pi(\boldsymbol{x}|\theta)f(\boldsymbol{y}|\theta, \boldsymbol{x})}{\tilde{p}(\boldsymbol{x}|\theta, \boldsymbol{y})}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}(\theta)} := \underline{p}(\theta|\boldsymbol{y}),
\end{aligned}
$$

where $\tilde{p}(\boldsymbol{x}|\theta, \boldsymbol{y})$ is the Laplace approximation of $p(\boldsymbol{x}|\theta, \boldsymbol{y})$, and $\hat{\boldsymbol{x}}(\theta, \boldsymbol{y})$ is the mode (maximizer) of $p(\boldsymbol{x}|\theta, \boldsymbol{y})$.

$\hookrightarrow$ We will denote this approximation by $\underline{p}(\theta|\boldsymbol{y})$.

Integrated Nested Laplace Approximation (INLA)

- (II) is more complex, because the dimension of $\boldsymbol{x}$ can be large, meaning that many marginals need to be computed.
- One possibility is to approximate $p(x_i|\boldsymbol{\theta}, \boldsymbol{y})$ directly by a Gaussian, based on the Hessian of $p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{y})$ at its mode $\hat{\boldsymbol{x}}(\boldsymbol{\theta}, \boldsymbol{y})$. Although this is fast, the approximation might not be very accurate.
- Alternatively, we can write $\boldsymbol{x} = (x_j, \boldsymbol{x}_{-j})$ ($-j$ refers to all the indices except j), and

$$\begin{aligned}
p(x_j|\boldsymbol{\theta}, \boldsymbol{y}) &= \frac{p(x_j, \boldsymbol{x}_{-j}|\boldsymbol{\theta}, \boldsymbol{y})}{p(\boldsymbol{x}_{-j}|x_j, \boldsymbol{\theta}, \boldsymbol{y})} = \frac{p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{y})}{p(\boldsymbol{x}_{-j}|x_j, \boldsymbol{\theta}, \boldsymbol{y})} \\
&\propto \frac{\pi(\boldsymbol{\theta})\pi(\boldsymbol{x}|\boldsymbol{\theta})f(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})}{p(\boldsymbol{x}_{-j}|x_j, \boldsymbol{\theta}, \boldsymbol{y})} \\
&\approx \left.\frac{\pi(\boldsymbol{\theta})\pi(\boldsymbol{x}|\boldsymbol{\theta})f(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})}{\tilde{p}(\boldsymbol{x}_{-j}|x_j, \boldsymbol{\theta}, \boldsymbol{y})}\right|_{\boldsymbol{x}_{-j}=\hat{\boldsymbol{x}}_{-j}(x_j, \boldsymbol{\theta}, \boldsymbol{y})} := \underline{p}(x_j|\boldsymbol{\theta}, \boldsymbol{y}),
\end{aligned}$$

where $\tilde{p}(\boldsymbol{x}_{-j}|x_j, \boldsymbol{\theta}, \boldsymbol{y})$ is the Laplace approximation of $p(\boldsymbol{x}_{-j}|x_j, \boldsymbol{\theta}, \boldsymbol{y})$, and $\hat{\boldsymbol{x}}_{-j}(x_j, \boldsymbol{\theta}, \boldsymbol{y})$ denotes the mode of $p(\boldsymbol{x}_{-j}|x_j, \boldsymbol{\theta}, \boldsymbol{y})$.

Integrated Nested Laplace Approximation (INLA)

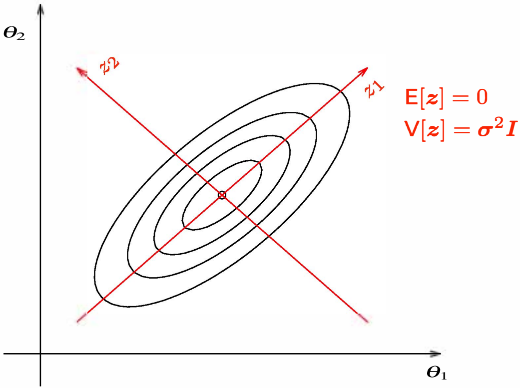
- The approximations $\underline{p}(x_j|\boldsymbol{\theta}, \boldsymbol{y})$ and $\underline{p}(\boldsymbol{\theta}|\boldsymbol{y})$ detailed above are called Integrated Nested Laplace Approximation because the Laplace approximation is only applied within the denominator, not directly on the distributions $p(x_j|\boldsymbol{\theta}, \boldsymbol{y})$ and $p(\boldsymbol{\theta}|\boldsymbol{y})$.
- These approximations work well for most LGMs encountered in practice. However, computing $\underline{p}(x_j|\boldsymbol{\theta}, \boldsymbol{y})$ for every j can be somewhat computationally expensive when $\boldsymbol{x}$ is very high dimensional.
- A faster alternative method for approximating $p(x_j|\boldsymbol{\theta}, \boldsymbol{y})$ is called the "Simplified Laplace Approximation". This is based on a Taylor series expansion approximation (up to third order) of both the numerator and the denominator of $\underline{p}(x_j|\boldsymbol{\theta}, \boldsymbol{y})$.
- This effectively corrects the Gaussian approximation of $p(x_j|\boldsymbol{\theta}, \boldsymbol{y})$ for location and skewness, leading to better accuracy.
- Simplified Laplace Approximation is the default option in R-INLA, but users can choose to do the full Laplace approximation (i.e. compute $\underline{p}(x_j|\boldsymbol{\theta}, \boldsymbol{y})$), at the expense of longer running time.

Steps in INLA's Operation

(I) Explore the marginal of the hyperparameters, $p(\boldsymbol{\theta}|\boldsymbol{y})$.

1. Locate the mode $\hat{\boldsymbol{\theta}}$ of $\underline{p}(\boldsymbol{\theta}|\boldsymbol{y})$ by maximizing $\log \underline{p}(\boldsymbol{\theta}|\boldsymbol{y})$ using a variant of Newton's method.
2. Compute the Hessian of $\log \underline{p}(\boldsymbol{\theta}|\boldsymbol{y})$ at $\hat{\boldsymbol{\theta}}$, and change coordinates to standardize the variables. This improves conditioning, and simplifies numerical integration.

Standardizing the variables by change of coordinates



Steps in INLA's Operation

(II) Explore $\log \underline{p}(\boldsymbol{\theta}|\boldsymbol{y})$ and produce H points $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(H)}$ associated with the bulk of the mass of the density $\underline{p}(\boldsymbol{\theta}|\boldsymbol{y})$, together with the corresponding area weights $\Delta^{(1)}, \Delta^{(2)}, \ldots, \Delta^{(H)}$. In addition to this, for each point $\boldsymbol{\theta}^{(h)}$ for $1 \leq h \leq H$, we

1. Evaluate the marginal posterior of the hyperparameter $\underline{p}(\boldsymbol{\theta}^{(h)}|\boldsymbol{y})$.
2. Evaluate the marginals of the latent field $\underline{p}(x_j|\boldsymbol{\theta}^{(h)}, \boldsymbol{y})$ on a grid of selected values of x_j, for every j (either simplified Laplace approximation or full Laplace approximation can be used here).

(III) Obtain the marginals $\underline{p}(\theta_i|\boldsymbol{y})$ at selected gridpoints for θ_i by interpolation, using $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(H)}$ and $\underline{p}(\boldsymbol{\theta}^{(1)}|\boldsymbol{y}), \ldots, \underline{p}(\boldsymbol{\theta}^{(H)}|\boldsymbol{y})$.

(IV) Obtain the marginals $\underline{p}(x_j|\boldsymbol{y})$ at selected gridpoints for x_j by

$$\underline{p}(x_j|\boldsymbol{y}) \approx \sum_{h=1}^{H} \underline{p}(x_j|\boldsymbol{\theta}^{(h)}, \boldsymbol{y})\underline{p}(\boldsymbol{\theta}^{(h)}|\boldsymbol{y})\Delta^{(h)}.$$

Steps in INLA's Operation

(II) Explore $\log \underline{p}(\boldsymbol{\theta}|\boldsymbol{y})$ and produce H points $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(H)}$ associated with the bulk of the mass of the density $\underline{p}(\boldsymbol{\theta}|\boldsymbol{y})$, together with the corresponding area weights $\Delta^{(1)}, \Delta^{(2)}, \ldots, \Delta^{(H)}$. In addition to this, for each point $\boldsymbol{\theta}^{(h)}$ for $1 \le h \le H$, we

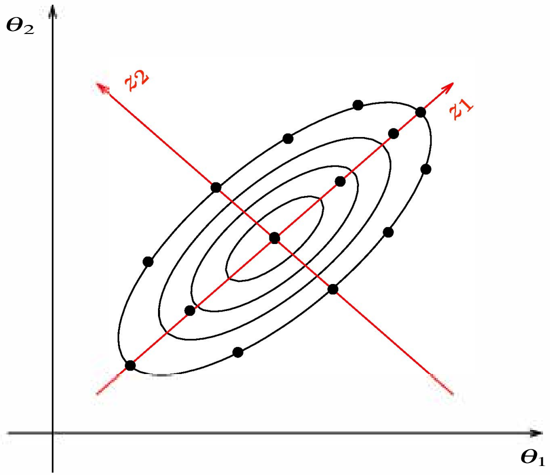
1. Evaluate the marginal posterior of the hyperparameter $\underline{p}(\boldsymbol{\theta}^{(h)}|\boldsymbol{y})$.
2. Evaluate the marginals of the latent field $\underline{p}(x_j|\boldsymbol{\theta}^{(h)}, \boldsymbol{y})$ on a grid of selected values of x_j, for every j (either simplified Laplace approximation or full Laplace approximation can be used here).

(III) Obtain the marginals $\underline{p}(\theta_i|\boldsymbol{y})$ at selected gridpoints for θ_i by interpolation, using $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(H)}$ and $\underline{p}(\boldsymbol{\theta}^{(1)}|\boldsymbol{y}), \ldots, \underline{p}(\boldsymbol{\theta}^{(H)}|\boldsymbol{y})$.

(IV) Obtain the marginals $\underline{p}(x_j|\boldsymbol{y})$ at selected gridpoints for x_j by

$$\underline{p}(x_j|\boldsymbol{y}) \approx \sum_{h=1}^{H} \underline{p}(x_j|\boldsymbol{\theta}^{(h)}, \boldsymbol{y})\underline{p}(\boldsymbol{\theta}^{(h)}|\boldsymbol{y})\Delta^{(h)}.$$

Selecting gridpoints for approximating $\underline{p}(\boldsymbol{\theta}|\boldsymbol{y})$



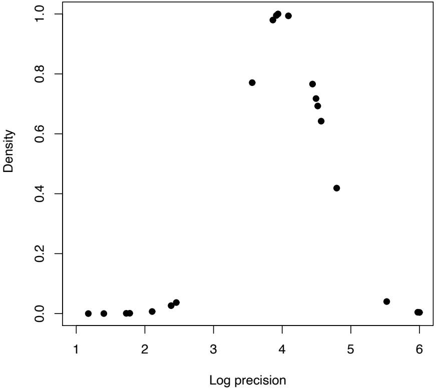
Example for INLA's Operation

↪ Consider the following simple model

$$
\begin{aligned}
y_{ij}|\boldsymbol{x}, \theta &\sim N(x_j, \sigma_0^2) \text{ for } 1 \leq i \leq n, 1 \leq j \leq n_j \quad (\sigma_0^2 \text{ is known}) \\
x_j|\theta &\sim N(0, \theta^{-1}) \quad (\theta \text{ corresponds to the precision}) \\
\theta &\sim \Gamma(a, b)
\end{aligned}
$$

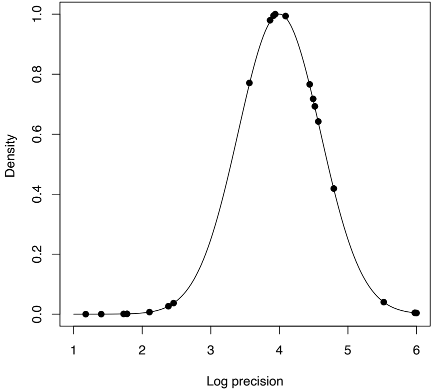
↪ In the following figures, the INLA approximation for computing the posterior marginal approximations $\underline{p}(\theta|\boldsymbol{y})$ and $\underline{p}(x_i|\boldsymbol{y})$ is illustrated.

Example for INLA's Operation



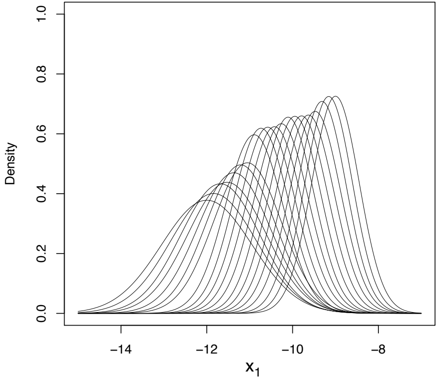
$\underline{p}(\theta|\boldsymbol{y})$ computed at H gridpoints $\theta^{(1)}, \ldots, \theta^{(H)}$

Example for INLA's Operation



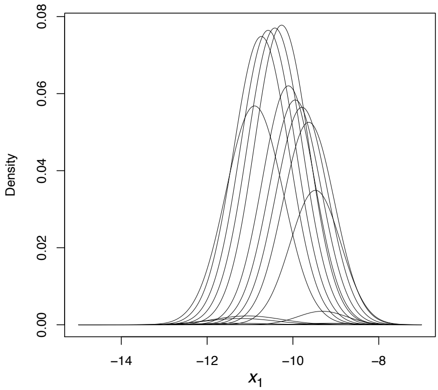
$\underline{p}(\theta|\boldsymbol{y})$ interpolated

Example for INLA's Operation



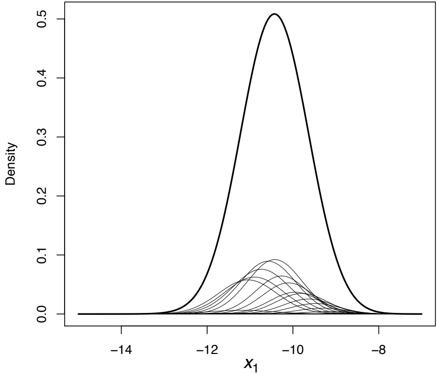
$\underline{p}(x_1|\theta^{(h)}, \boldsymbol{y})$ is computed for each gridpoint $1 \leq h \leq H$

Example for INLA's Operation



$\underline{p}(x_1|\theta^{(h)}, \boldsymbol{y})$ weighted for each $1 \le h \le H$ according to $\underline{p}(\boldsymbol{\theta}^{(h)}|\boldsymbol{y})\Delta^{(h)}$

Example for INLA's Operation



$\underline{p}(x_1 | y)$ obtained by summing up the weighted conditional densities

Summary of INLA

↪ The basic idea behind INLA is quite simple.

- Use of Laplace approximation repeatedly in a nested manner, i.e. not directly on the posterior, but on some of the conditional distributions that arise in calculations.
- Take advantage of the structure of the Latent Gaussian Model (such as the sparsity of the GMRF prior) to speed up calculations.
- Use numerical integration over the hyperparameter space. The grid can be further refined if the precision is not yet sufficient (by increasing the number of gridpoints).

Some possible complications:

- In the case of markedly non-Gaussian observations, if the number of observations is small, the Laplace approximations might not be very accurate. Nevertheless, several remedies were proposed that increase the accuracy in such situations, at the expense of increased computational cost.
- When the number of hyperparameters exceeds 10, numerical integration can become slow. To overcome this issue, several authors have proposed the combination of MCMC and INLA (i.e. by sampling using MCMC only on the hyperparameter space), see e.g. “Markov chain Monte Carlo with the Integrated Nested Laplace Approximation" by Gómez-Rubio & Rue.

The INLA package for R

- The procedures that form INLA are implemented in the R-INLA package. This is an R package that also installs two C components that do the actual calculations:
  - The GMRFLib library, which is a C library for fast simulation of GMRFs. It is able to sample from unconditional GMRFs and conditional GMRFs, and evaluate the corresponding log-densities and related quantities.
  - The inla program, which is a standalone C program that interfaces with GMRF, and performs the required calculations for INLA.
- The R-INLA package processes the data in R into the format required by the inla program, and then collects the results and returns them in the R format. The library is available at `www.r-inla.org`. It runs natively on Linux, Windows and Mac.
- The necessary commands for loading R-INLA in Kaggle are included the code for this lecture and Workshop 3.

The INLA package for R

↪ The INLA method for Bayesian inference on LGMs can be applied using the command

```
m <- inla(formula, data, family, ...) ,
```

where

- `formula` describes the regression model between the linear predictor and the covariates, including the specification of the random effects
- `data` contains a dataframe including the response variables and the covariates
- `family` describes the likelihood model of the observations y_i (i.e. the distribution of $y_i|\boldsymbol{x}, \boldsymbol{\theta}$).
- There are many possible additional parameters that can specify the link function (each `family` has a default one), the priors, and indicate to INLA that we would like some additional quantities to be computed.

↪ Once INLA has done the calculations, various summary statistics can be printed out using the `summary(m)` command. Other information (such as posterior marginals, etc.) can also be extracted from `m` using the appropriate elements (accessed through `m$element`).

Example: Bayesian Linear regression (mtcars dataset)

$\hookrightarrow$ In Lecture 2, we looked at the mtcars dataset which describes the fuel consumption of cars (mpg), and several other aspects of car construction and performance. We picked 3 covariates (Rear axle ratio, weight, and 1/4 mile time) and used them in a Bayesian linear regression model in JAGS with fuel consumption as the response variable.

$\hookrightarrow$ As a reminder, the model was of the following form,

$$\begin{aligned} y_i \mid \mu_i, \sigma^2 &\overset{\text{ind}}{\sim} \mathrm{N}(\mu_i, \sigma^2), \qquad i = 1, \ldots, n, \\ \mu_i &= \beta_0 + \sum_{j=1}^{n_\beta} \beta_j z_{j,i}, \\ \beta_j &\sim \mathrm{N}(\mu_{\beta_j}, \sigma^2_{\beta_j}), \quad j = 0, \ldots, p, \\ \tau &\sim \mathrm{Gamma}(a, b). \end{aligned}$$

$\hookrightarrow$ Here y_i are the response variables, β_j are the regression coefficients (β_0 is the intercept), and $z_{j,i}$ are the covariates.

Example: Bayesian Linear regression (mtcars dataset)

↪ The code below fits this model in INLA using the default priors. By default, the intercept of the model is assigned a Gaussian prior with mean and precision equal to 0. The rest of the fixed effects (regression coefficients) are assigned Gaussian priors with mean equal to 0 and precision equal to 0.001. The default prior for the Gaussian precision τ of the Gaussian likelihood is a Gamma prior with parameters $(1, 0.00005)$. As internally the logarithm of the precision $\theta = log(\tau)$ is stored, this is equivalent to a log-Gamma prior on θ with parameters $(1, 0.00005)$.

```
#We load the dataset, and select the relevant covariates
mtcars1=mtcars[c("mpg","drat","wt","qsec")]

#This is a standard linear model
m.mtcars.linear=lm(mpg~drat+wt+qsec,data=mtcars1)

#This code fits the same model with INLA, using the default priors
m.mtcars.I.defaultprior=inla(mpg~drat+wt+qsec,
                             data=mtcars1,family="gaussian")
#This displays the summary statistics
summary(m.mtcars.I.defaultprior)
```

Example: Bayesian Linear regression (mtcars dataset)

↪ The summary statistics obtained after fitting the model:

```
Call:
   "inla(formula = mpg ~ drat + wt + qsec, family = \"gaussian\", data =
   mtcars1)"
Time used:
    Pre = 0.427, Running = 0.0822, Post = 0.0343, Total = 0.543
Fixed effects:
              mean    sd 0.025quant 0.5quant 0.975quant   mode kld
(Intercept) 11.390 8.037     -4.497   11.390     27.265 11.391   0
drat         1.656 1.222     -0.759    1.656      4.069  1.656   0
wt          -4.397 0.675     -5.732   -4.397     -3.063 -4.397   0
qsec         0.946 0.261      0.431    0.946      1.461  0.946   0

Model hyperparameters:
                                         mean    sd 0.025quant 0.5quant
Precision for the Gaussian observations 0.163 0.042      0.092     0.16
                                        0.975quant  mode
Precision for the Gaussian observations      0.256 0.153

Expected number of effective parameters(stdev): 4.00(0.001)
Number of equivalent replicates : 8.00

Marginal log-Likelihood:  -97.81
```

↪ INLA provides an estimate of the effective number of parameters, a measure of the complexity of the model. The number of equivalent replicates is computed as well, which is the number of observations divided by the effective number of parameters. This is the average number of observations available to estimate each parameter in the model (higher values are better).

Example: Bayesian Linear regression (mtcars dataset)

- We set the priors:

```
prec.prior <- list(prec=list(prior = "loggamma", param = c(0.1, 0.1)))
prior.beta <- list(mean.intercept = 0, prec.intercept = 0.001,
                   mean = 0, prec = 0.001)

m.mtcars.I=inla(mpg~drat+wt+qsec,data=mtcars1,family="gaussian",
control.family=list(hyper=prec.prior),control.fixed=prior.beta)
summary(m.mtcars.I)
```

- You can choose a different mean or precision for the Gaussian prior of the 3 regression coefficients by passing along lists to mean and prec,

```
prior.beta <- list(mean.intercept = 0, prec.intercept = 0.001,
                  mean = list(drat=0.1,wt=0.3,qsec=0.5),
                  prec = list(drat=0.001,wt=0.002,qsec=0.003))
```

Example: Bayesian Linear regression (mtcars dataset)

↪ The summary statistics displayed by INLA are shown below. These are very similar to the ones we got from JAGS in Lecture 2.

```
Call:
   c("inla(formula = mpg ~ drat + wt + qsec, family = \"gaussian\", data =
   mtcars1, ", " control.family = list(hyper = prec.prior), control.fixed
   = prior.beta)" )
Time used:
    Pre = 0.349, Running = 0.126, Post = 0.0298, Total = 0.505
Fixed effects:
              mean    sd 0.025quant 0.5quant 0.975quant   mode kld
(Intercept) 10.659 8.013     -5.238   10.681     26.419 10.724   0
drat         1.741 1.236     -0.694    1.739      4.190  1.734   0
wt          -4.351 0.684     -5.698   -4.352     -2.996 -4.355   0
qsec         0.962 0.265      0.438    0.961      1.487  0.960   0

Model hyperparameters:
                                         mean    sd 0.025quant 0.5quant
Precision for the Gaussian observations 0.154 0.041      0.084     0.15
                                        0.975quant  mode
Precision for the Gaussian observations      0.244 0.143

Expected number of effective parameters(stdev): 3.93(0.018)
Number of equivalent replicates : 8.14

Marginal log-Likelihood:  -93.19
```

Working with marginals

- The `names` command is very useful, it allows us to lists all of the available names in the `inla` object that we are able to use to extract information. These elements can be referred to using the $ notation.

```
names(m.mtcars.I)
---------------
'names.fixed', 'summary.fixed', 'marginals.fixed', 'summary.lincomb',
'marginals.lincomb', 'size.lincomb', 'summary.lincomb.derived',
'marginals.lincomb.derived', 'size.lincomb.derived', 'mlik', 'cpo',
'po', 'waic', 'model.random', 'summary.random', 'marginals.random',
'size.random', 'summary.linear.predictor', 'marginals.linear.predictor',
'summary.fitted.values', 'marginals.fitted.values', 'size.linear.predictor',
'summary.hyperpar', 'marginals.hyperpar', 'internal.summary.hyperpar',
'internal.marginals.hyperpar', 'offset.linear.predictor', 'model.spde2.blc',
'summary.spde2.blc', 'marginals.spde2.blc', 'size.spde2.blc',
'model.spde3.blc', 'summary.spde3.blc', 'marginals.spde3.blc',
'size.spde3.blc', 'logfile', 'misc', 'dic', 'mode', 'neffp', 'joint.hyper',
'nhyper', 'version', 'Q', 'graph', 'ok', 'cpu.used', 'all.hyper', '.args',
'call', 'model.matrix'
```

- We can also apply the `names` command to elements of `inla` objects, such as `marginals.fixed`.

```
names(m.mtcars.I$marginals.fixed)
---------------
'(Intercept)','drat', 'wt', 'qsec'
```

We can access the marginals of the regression coefficient β_0 as `m.mtcars.I$marginals.fixed$'(Intercept)'`, or equivalently `m.mtcars.I$marginals.fixed[[1]]`, and similarly for the other coefficients.

Working with marginals

- We are able to plot the marginals of the regression coefficients using the `plot` function.

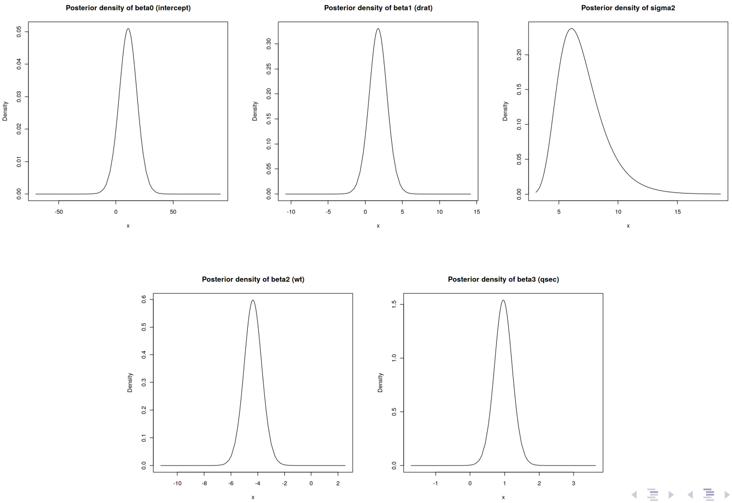
```
plot(m.mtcars.I$marginals.fixed$'(Intercept)',type='l',xlab="x",
ylab="Density",main="Posterior density of beta0 (intercept)")

plot(m.mtcars.I$marginals.fixed$'drat',type='l',xlab="x",
ylab="Density",main="Posterior density of beta1 (drat)")

plot(m.mtcars.I$marginals.fixed$'wt',type='l',xlab="x",
ylab="Density",main="Posterior density of beta2 (wt)")

plot(m.mtcars.I$marginals.fixed$'qsec',type='l',xlab="x",
ylab="Density",main="Posterior density of beta3 (qsec)")
```

- In order to obtain the marginal of the variance σ^2, we need first extract the marginal of the precision parameter τ, and then transform it.

```
marginal.tau=m.mtcars.I$marginals.hyperpar[[1]]
marginal.sigma2 <- inla.tmarginal(function(tau) tau^(-1),marginal.tau)
plot(marginal.sigma2,type='l',xlab="x",ylab="Density",
main="Posterior density of sigma2")
```

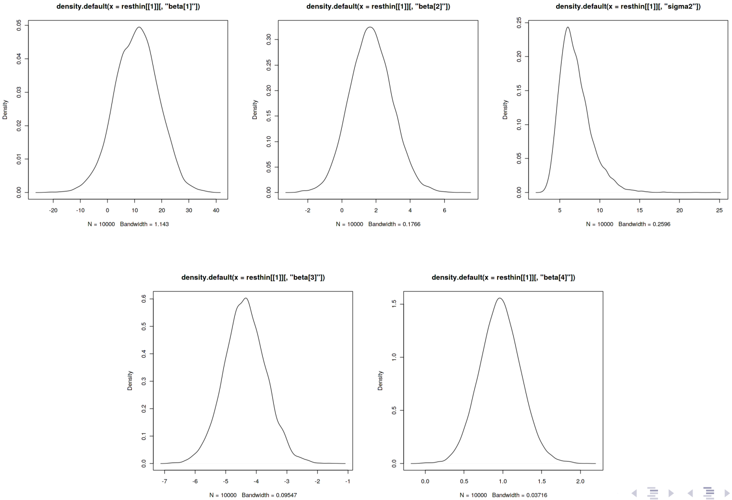
Working with marginals

See the plots of the marginals that we have computed with INLA.



Working with marginals

As a comparison, here are the plots with JAGS (Lecture 2), very similar.



Working with marginals

- We might also be interested in computing summary statistics for σ^2. This can be done using the `inla.zmarginal` command applied on `marginal.sigma2` (which we have computed previously by transforming the marginal of τ using `inla.tmarginal`.

```
cat("Summary statistics of sigma2\n")
inla.zmarginal(marginal.sigma2)
--------------
Summary statistics of sigma2
Mean                6.98976
Stdev               1.97818
Quantile  0.025 4.10781
Quantile  0.25  5.58166
Quantile  0.5   6.65383
Quantile  0.75  8.01779
Quantile  0.975 11.805
```

- As a comparison, these were the same summary statistics for sigma2 obtained by JAGS:

```
Summary statistics of sigma2 (JAGS)
Mean                6.9908
Stdev               1.9881
Quantile  0.025 4.1073
Quantile  0.25  5.5928
Quantile  0.5   6.6414
Quantile  0.75  8.031
Quantile  0.975 11.747
```

- We can see that these are virtually identical, showing that the INLA approximation is very accurate in this case.

Working with marginals

- Besides `inla.tmarginal` and `inla.zmarginal`, there are several other very useful functions in INLA for working with marginals. The standard "d", "p", "r" and "q" functions for distributions in R have their INLA equivalents,

```
inla.dmarginal(x, marginal, log = FALSE)
inla.pmarginal(q, marginal, normalize = TRUE, len = 1024)
inla.qmarginal(p, marginal, len = 1024)
inla.rmarginal(n, marginal)
```

  The marginal parameter will be a marginal object, such `marginal.sigma2` or `m.mtcars.I$marginals.fixed$'drat'` in our previous code.

- These for functions evaluate the density, the CDF, the quantiles, and generate *n* random samples from the marginal distribution.

- We can compute the expected value of an arbitrary function according to the marginal using `inla.emarginal(fun, marginal)`.

- There are several other functions available, see `https://rdrr.io/github/andrewzm/INLA/man/marginal.html` for a complete list.

- We have seen that INLA can be applied using the command

```
m <- inla(formula, data, family, control.fixed,
          control.family,...)
```

- `family` specifies the likelihood, the next slides show the list of available likelihoods. These all come with a default link function g depending on the likelihood. In some cases this can be changed using the `control.link` option.
- `control.fixed` allows us to set the prior on the fixed effects (regression coefficients). Only Gaussian priors are possible on the fixed effects, and these can be set in the form `control.fixed=list(mean.intercept = 0, prec.intercept = 0.001, mean = 0, prec = 0.001)`, as we have seen in the mtcars example.
- `control.family` allows controlling different options, including the priors on the hyperparameters. These can be specified by `control.family=list(hyper = list(hyperparameter = list(prior="prior name",param= parameter values))` in `inla`. We will show the list of available hyperparameter priors in a few slides.
- Besides specifying the prior on the hyperparameters, we sometimes want to use some random effects with a GMRF prior (these are also called latent effects). They are set in INLA inside the `formula` term, such as `y∼ x1+...+xn+f(covariates, model="name of latent model")`. We show the list of such models as well.

Available likelihoods in INLA, page 1

The available likelihoods in INLA are listed by `inla.list.models("likelihood")`:

```
beta                 The Beta likelihood
betabinomial         The Beta-Binomial likelihood
betabinomialna       The Beta-Binomial Normal approximation likelihood
binomial             The Binomial likelihood
cbinomial            The clustered Binomial likelihood
cenpoisson           Then censored Poisson likelihood
circularnormal       The circular Gaussian likelihoood
coxph                Cox-proportional hazard likelihood
dgp                  Discrete generalized Pareto likelihood
exponential          The Exponential likelihood
exponentialsurv      The Exponential likelihood (survival)
gamma                The Gamma likelihood
gammacount           A Gamma generalisation of the Poisson likelihood
gammasurv            The Gamma likelihood (survival)
gaussian             The Gaussian likelihoood
gev                  The Generalized Extreme Value likelihood
gev2                 The Generalized Extreme Value likelihood (2nd variant)
gp                   Generalized Pareto likelihood
gpoisson             The generalized Poisson likelihood
logistic             The Logistic likelihoood
loglogistic          The loglogistic likelihood
loglogisticsurv      The loglogistic likelihood (survival)
```

Available likelihoods in INLA, page 2

`lognormal`	`The log-Normal likelihood`
`lognormalsurv`	`The log-Normal likelihood (survival)`
`logperiodogram`	`Likelihood for the log-periodogram`
`nbinomial`	`The negBinomial likelihood`
`nbinomial2`	`The negBinomial2 likelihood`
`nmix`	`Binomial-Poisson mixture`
`nmixnb`	`NegBinomial-Poisson mixture`
`poisson`	`The Poisson likelihood`
`pom`	`Likelihood for the proportional odds model`
`qkumar`	`A quantile version of the Kumar likelihood`
`qloglogistic`	`A quantile loglogistic likelihood`
`qloglogisticsurv`	`A quantile loglogistic likelihood (survival)`
`simplex`	`The simplex likelihood`
`skewnormal`	`The Skew-Normal likelihoood`
`sn`	`The Skew-Normal likelihoood`
`sn2`	`The Skew-Normal likelihoood (alt param)`
`stochvol`	`The Gaussian stochvol likelihood`
`stochvolnig`	`The Normal inverse Gaussian stochvol likelihood`
`stochvolt`	`The Student-t stochvol likelihood`
`t`	`Student-t likelihood`
`tstrata`	`A stratified version of the Student-t likelihood`

Available likelihoods in INLA, page 3

```
weibull                        The Weibull likelihood
weibullcure                    The Weibull-cure likelihood (survival)
weibullsurv                    The Weibull likelihood (survival)
wrappedcauchy                  The wrapped Cauchy likelihoood
xpoisson                       The Poisson likelihood (expert version)
zeroinflatedbetabinomial0      Zero-inflated Beta-Binomial, type 0
zeroinflatedbetabinomial1      Zero-inflated Beta-Binomial, type 1
zeroinflatedbetabinomial2      Zero inflated Beta-Binomial, type 2
zeroinflatedbinomial0          Zero-inflated Binomial, type 0
zeroinflatedbinomial1          Zero-inflated Binomial, type 1
zeroinflatedbinomial2          Zero-inflated Binomial, type 2
zeroinflatednbinomial0         Zero inflated negBinomial, type 0
zeroinflatednbinomial1         Zero inflated negBinomial, type 1
zeroinflatednbinomial1strata2 Zero inflated negBinomial, type 1, strata 2
zeroinflatednbinomial1strata3 Zero inflated negBinomial, type 1, strata 3
zeroinflatednbinomial2         Zero inflated negBinomial, type 2
zeroinflatedpoisson0           Zero-inflated Poisson, type 0
zeroinflatedpoisson1           Zero-inflated Poisson, type 1
zeroinflatedpoisson2           Zero-inflated Poisson, type 2
zeroninflatedbinomial2         Zero and N inflated binomial, type 2
zeroninflatedbinomial3         Zero and N inflated binomial, type 3
```

↪ You can use these by writing `family` = "name of likelihood" when calling `inla`.
↪ More info and examples for each likelihood is available at
`https://inla.r-inla-download.org/r-inla.org/doc/likelihood/`.

Available priors for hyperparameters, page 1

The available GMRF priors for the latent effects in INLA are listed by `inla.list.models("prior")`:

```
betacorrelation        Beta prior for the correlation
dirichlet              Dirichlet prior
expression:            A generic prior defined using expressions
flat                   A constant prior
gamma                  Gamma prior
gaussian               Gaussian prior
invalid                Void prior
jeffreystdf            Jeffreys prior for the doc
logflat                A constant prior for log(theta)
loggamma               Log-Gamma prior
logiflat               A constant prior for log(1/theta)
logitbeta              Logit prior for a probability
logtgaussian           Truncated Gaussian prior
logtnormal             Truncated Normal prior
mvnorm                 A multivariate Normal prior
none                   No prior
normal                 Normal prior
pc                     Generic PC prior
pc.alphaw              PC prior for alpha in Weibull
pc.ar                  PC prior for the AR(p) model
pc.cor0                PC prior correlation, basemodel cor=0
pc.cor1                PC prior correlation, basemodel cor=1
```

Available priors hyperparameters, page 2

pc.dof	PC prior for log(dof-2)
pc.fgnh	PC prior for the Hurst parameter in FGN
pc.gamma	PC prior for a Gamma parameter
pc.gammacount	PC prior for the GammaCount likelihood
pc.gevtail	PC prior for the tail in the GEV likelihood
pc.matern	PC prior for the Matern SPDE
pc.mgamma	PC prior for a Gamma parameter
pc.prec	PC prior for log(precision)
pc.range	PC prior for the range in the Matern SPDE
pc.sn	PC prior for the skew-normal
pc.spde.GA	(experimental)
pom	#classes-dependent prior for the POM model
ref.ar	Reference prior for the AR(p) model, p<=3
table:	A generic tabulated prior
wishart1d	Wishart prior dim=1
wishart2d	Wishart prior dim=2
wishart3d	Wishart prior dim=3
wishart4d	Wishart prior dim=4
wishart5d	Wishart prior dim=5

↪ Specified by `control.family=list(hyper = list(hyperparameter = list(prior="prior name",param= parameter values))` in `inla`. Important to understand the internal parametrisation of the likelihood model (explained in the documentation of the likelihood), as priors need to be specified on the internal parameters.

↪ More info and examples: `https://inla.r-inla-download.org/r-inla.org/doc/prior/`.

Available latent effects models (GMRF priors), page 1

The available GMRF priors for the latent effects in INLA are listed by `inla.list.models("latent")`:

```
ar              Auto-regressive model of order p (AR(p))
ar1             Auto-regressive model of order 1 (AR(1))
ar1c            Auto-regressive model of order 1 w/covariates
besag           The Besag area model (CAR-model)
besag2          The shared Besag model
besagproper     A proper version of the Besag model
besagproper2    An alternative proper version of the Besag model
bym             The BYM-model (Besag-York-Mollier model)
bym2            The BYM-model with the PC priors
clinear         Constrained linear effect
copy            Create a copy of a model component
crw2            Exact solution to the random walk of order 2
dmatern         Dense Matern field
fgn             Fractional Gaussian noise model
fgn2            Fractional Gaussian noise model (alt 2)
generic         A generic model
generic0        A generic model (type 0)
generic1        A generic model (type 1)
generic2        A generic model (type 2)
generic3        A generic model (type 3)
iid             Gaussian random effects in dim=1
iid1d           Gaussian random effect in dim=1 with Wishart prior
iid2d           Gaussian random effect in dim=2 with Wishart prior
iid3d           Gaussian random effect in dim=3 with Wishart prior
iid4d           Gaussian random effect in dim=4 with Wishart prior
```

Available latent effects models (GMRF priors), page 2

`iid5d`	Gaussian random effect in dim=5 with Wishart prior
`intslope`	Intecept-slope model with Wishart-prior
`linear`	Alternative interface to an fixed effect
`log1exp`	A nonlinear model of a covariate
`logdist`	A nonlinear model of a covariate
`matern2d`	Matern covariance function on a regular grid
`meb`	Berkson measurement error model
`mec`	Classical measurement error model
`ou`	The Ornstein-Uhlenbeck process
`revsigm`	Reverse sigmoidal effect of a covariate
`rgeneric`	Generic latent model specified using R
`rw1`	Random walk of order 1
`rw2`	Random walk of order 2
`rw2d`	Thin-plate spline model
`rw2diid`	Thin-plate spline with iid noise
`seasonal`	Seasonal model for time series
`sigm`	Sigmoidal effect of a covariate
`slm`	Spatial lag model
`spde`	A SPDE model
`spde2`	A SPDE2 model
`spde3`	A SPDE3 model
`z`	The z-model in a classical mixed model formulation

↪ Specified by `y~ x1+...+xn+f(covariates, model="name of latent model")` in formula in `inla`.

↪ More info and examples: `https://inla.r-inla-download.org/r-inla.org/doc/latent/`.

Example: Robust regression (Scottish hill racing data)

- The data set `hills.txt` contains information on the winning times (in minutes) in 1984 for 35 Scottish hill races, as well as two factors which presumably influence the duration of the race:
  - dist: The distance of the race (in miles).
  - climb: The elevation change (in feet).
- We looked at this in Lecture 2 and fit a robust linear regression model with dof parameter $\nu = 5$ using JAGS. Here we repeat the analysis using INLA, and display the summary.

```
hills=read.table("hills.txt",header=TRUE)

prior.t <- list(prec=list(prior = "loggamma", param = c(0.1, 0.1)),
                dof = list(initial=log(5-2), fixed=TRUE) )

prior.fixed <- list(mean.intercept = 0, prec.intercept = 0.001,
                    mean = 0, prec = 0.001)

m.hills.I <- inla(time ~ 1+climb+dist,family="T",data=hills,
                  control.family=list(hyper=prior.t),
                  control.fixed=prior.fixed)
summary(m.hills.I)
```

Example: Robust regression (Scottish hill racing data)

- Below are summary statistics displayed by INLA. These are virtually identical to the previous results from JAGS for this example.

```
Call:
   c("inla(formula = time ~ 1 + climb + dist, family = \"T\", data =
   hills, ", " control.family = list(hyper = prior.t), control.fixed =
   prior.fixed)" )
Time used:
    Pre = 0.349, Running = 0.111, Post = 0.0296, Total = 0.49
Fixed effects:
              mean    sd 0.025quant 0.5quant 0.975quant   mode kld
(Intercept) -9.526 2.276    -14.285   -9.439     -5.271 -9.276   0
climb        0.008 0.001      0.006    0.008      0.011  0.008   0
dist         6.582 0.266      6.028    6.590      7.087  6.607   0

Model hyperparameters:
                                         mean    sd 0.025quant 0.5quant
precision for the student-t observations 0.016 0.006      0.007    0.015
                                         0.975quant  mode
precision for the student-t observations       0.03 0.013

Expected number of effective parameters(stdev): 3.00(0.002)
Number of equivalent replicates : 11.69

Marginal log-Likelihood:  -149.73
```

Example: Poisson regression (ship incident data)

↪ The following dataset (Ships.csv) was originally provided by J. Crilley and L.N. Heminway of Lloyd's Register of Shipping, and appeared in Generalized Linear Models (1989) by McCullagh and Nelder. The dataset contains categorical variables describing ship type (`type`), construction period (`built`), operation period (`oper`), number of incidents (`y`), as well as the number of months in operation (`months`). Incidents in this dataset mean damage to the hull caused by strong waves. Each row of the dataset refers to a group of essentially identical ships, i.e. type, construction period, and operation period is the same. The number of months in operation and the number of incidents refers to the total number summed up among the ships in this particular group.

```
ShipsIncidents <- read.csv("Ships.csv",sep=",")
head(ShipsIncidents)
 type  built    oper    months   y    id
 <fct> <fct>    <fct>   <int>  <int> <int>
1  A     60-64    60-74    127      0     1
2  A     60-64    75-79    63       0     2
3  A     65-69    60-74    1095     3     3
4  A     65-69    75-79    1095     4     4
5  A     70-74    60-74    1512     6     5
6  A     70-74    75-79    3353     18    6

summary(ShipsIncidents)
type     built      oper         months             y                 id
 A:7    60-64: 9   60-74:15   Min.    :   45   Min.    : 0.00   Min.    : 1.00
 B:7    65-69:10   75-79:19   1st Qu.:  371   1st Qu.: 1.00   1st Qu.: 9.25
 C:7    70-74:10              Median : 1095   Median : 4.00   Median :17.50
 D:7    75-79: 5              Mean   : 4811   Mean   :10.47   Mean   :17.50
 E:6                          3rd Qu.: 2223   3rd Qu.:11.75   3rd Qu.:25.75
                              Max.   :44882   Max.   :58.00   Max.   :34.00
```

Example: Poisson regression (ship incident data)

- A simple Poisson regression model can be written as follows. For every observation $1 \leq i \leq n$,
$$y_i \sim \text{Poisson}(\lambda_i)$$
$$\eta_i = \log(\lambda_i) = \beta_0 + \sum_{j=1}^{n_\beta} \beta_i z_{ji},$$
where z_{ji} are covariates, and $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_{n_\beta})$ are the regression coefficients (fixed effects).
- The mean of observation i is $\mathbb{E}(y_i|\boldsymbol{\beta}) = \lambda_i$, which is linked to the linear predictor η_i through the log link function $\eta_i = \log(\lambda_i)$. This fits in the LGM framework.
- In such Poisson regression models, y_i typically refers to the number of events during a time period. We often encounter situations where the time period T_i is different for different observations y_i. In such cases, it is more appropriate to use the slightly modified model
$$y_i \sim \text{Poisson}(T_i \rho_i)$$
$$\eta_i = \log(T_i \rho_i) = \beta_0 + \sum_{j=1}^{n_\beta} \beta_i z_{ji} + \log(T_i).$$

Example: Poisson regression (ship incident data)

- In this case, each linear regression equation has an additional constant term $\log(T_i)$ called an offset, that has regression coefficient fixed at 1.
- There are two equivalent ways to write this model in INLA, either using the `offset` parameter, or using the Poisson model specific `E` parameter.

```
formula.ships <-  y ~ 1 + type + built + oper

m.ships.poisson.I <- inla(formula.inla,family="poisson",
                         data=ShipsIncidents, offset=log(months))
#or equivalently
m.ships.poisson.I <- inla(formula.inla,family="poisson",
                              data=ShipsIncidents, E=months)
```

Example: Poisson regression (ship incident data)

The INLA results are printed by `summary(m.ships.poisson.I)`:

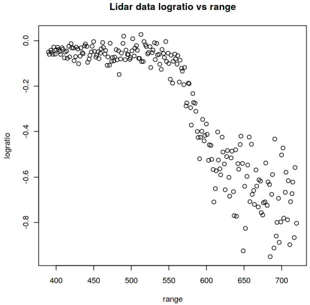
```
Call:
   c("inla(formula = formula.inla, family = \"poisson\", data =
   ShipsIncidents, ", " E = months)")
Time used:
    Pre = 0.47, Running = 0.0936, Post = 0.0294, Total = 0.593
Fixed effects:
              mean    sd 0.025quant 0.5quant 0.975quant   mode kld
(Intercept) -6.416 0.217     -6.852   -6.413     -5.998 -6.406   0
typeB       -0.543 0.178     -0.882   -0.546     -0.185 -0.553   0
typeC       -0.689 0.329     -1.366   -0.677     -0.075 -0.655   0
typeD       -0.075 0.290     -0.664   -0.069      0.476 -0.055   0
typeE        0.326 0.236     -0.141    0.327      0.785  0.330   0
built65-69   0.696 0.150      0.406    0.695      0.993  0.692   0
built70-74   0.818 0.170      0.487    0.818      1.153  0.816   0
built75-79   0.453 0.233     -0.012    0.455      0.904  0.460   0
oper75-79    0.384 0.118      0.153    0.384      0.617  0.383   0

Expected number of effective parameters(stdev): 9.00(0.00)
Number of equivalent replicates : 3.78

Marginal log-Likelihood:  -111.81
```

Example: Poisson regression (ship incident data)

- You can notice that although we have used only 3 covariates ship type (`type`), construction period (`built`), operation period (`oper`), there are 9 regression coefficients, including the intercept.
- This is because the covariates are categorical (called factor in R language), i.e. they only take a finite number of different values (A-E for `type`, 60-64, 65-69, 70-74 or 75-79 for `built`, and 60-74 or 75-79 for `oper`).
- If we have a categorical variables with c possible categories, in general, it is not a good idea to directly replace the categories with integers $1, 2, \ldots, c$, and use a single regression coefficient for it.
- Instead of this, each different category has a separate contribution, and so a different regression coefficient. In order to avoid model identifiability issues, the first category is always set to have 0 coefficient, and we let the remaining $c - 1$ categories have a separate coefficient each.

Example: Smoothing (Lidar data)

- LIDAR (Light Detection And Ranging) is a remote-sensing technique. It can be used for example to obtain measurements about the distribution of different gas molecules in the atmosphere. The `lidar` dataset (available in the `SemiPar` package) contains measurements on the concentration of atmospheric atomic mercury in an Italian geothermal field (see Holst et al., Environmetrics 7.4 (1996): 401-416 for more details).
- The dataset contains measurements of of two variables, `range` is the distance traveled before the light is reflected back to its source, while `logratio` is the logarithm of the ratio of received light from two laser sources. There seem to be a quite nonlinear dependency between the two variables. We are interested in smoothing this data and find the relation between these two variables.



Example: Smoothing (Lidar data)

↪ The relationship between `logratio` and `range` is clearly nonlinear. A first approach to model this would be using a polynomial regression, i.e. try to fit a model of the form $\texttt{logratio} \sim N(\beta_0 + \beta_1 \texttt{range} + \beta_2 \texttt{range}^2 + \beta_3 \texttt{range}^3, \sigma^2)$. This is achieved in INLA as follows.

```
library("SemiPar")
data(lidar)

m.lidar.poly <- inla(logratio ~ 1 + range +  I(range^2) + I(range^3),
data = lidar, control.predictor = list(compute = TRUE))
```

Example: Smoothing (Lidar data)

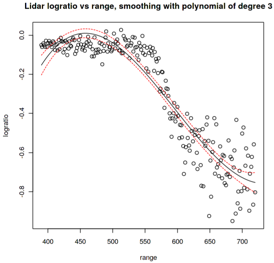
↪ By printing out he summary, we obtain

```
Call:
   c("inla(formula = logratio ~ 1 + range + I(range^2) + I(range^3), ", "
   data = lidar, control.predictor = list(compute = TRUE))" )
Time used:
    Pre = 0.309, Running = 0.193, Post = 0.0701, Total = 0.572
Fixed effects:
               mean    sd 0.025quant 0.5quant 0.975quant    mode kld
(Intercept) -13.443 1.554    -16.498  -13.443    -10.391 -13.443   0
range         0.074 0.009      0.057    0.074      0.091   0.074   0
I(range^2)    0.000 0.000      0.000    0.000      0.000   0.000   0
I(range^3)    0.000 0.000      0.000    0.000      0.000   0.000   0

Model hyperparameters:
                                          mean    sd 0.025quant 0.5quant
Precision for the Gaussian observations 106.76 10.19      87.72   106.43
                                        0.975quant   mode
Precision for the Gaussian observations     127.67 105.77

Expected number of effective parameters(stdev): 4.14(0.014)
Number of equivalent replicates : 53.35

Marginal log-Likelihood:  140.18
Posterior marginals for the linear predictor and
the fitted values are computed
```

Example: Smoothing (Lidar data)

- Fitted values in INLA always refer to the mean of the observations $\mu_i = \mathbb{E}(y_i)$. This is can be different from the linear predictor $\eta_i = g(\mu_i)$ in case the link function g is not the identity function.
- We are going to plot the posterior mean of the fitted values, as well as 95% credible intervals around each position according to the posterior distribution of the fitted values (i.e. $\mu_i = \beta_0 + \beta_1 \texttt{range} + \beta_2 \texttt{range}^2 + \beta_3 \texttt{range}^3$).
- This is easy in INLA as they are contained in `m.lidar.poly$summary.fitted.values`.

```
plot(lidar,main="Lidar logratio vs range,
smoothing with polynomial of degree 3")
lines(lidar$range,m.lidar.poly$summary.fitted.values$mean,type='l')
lines(lidar$range,m.lidar.poly$summary.fitted.values$'0.025quant',lty=2,col='red')
lines(lidar$range,m.lidar.poly$summary.fitted.values$'0.975quant',lty=2,col='red')
```



Example: Smoothing (Lidar data)

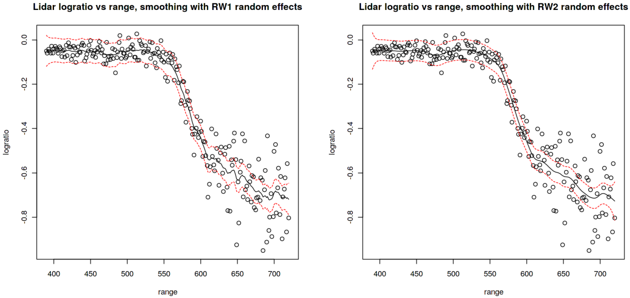
- An alternative, more flexible approach to polynomial regression is smoothing with random effects.
- Let $r_1, \ldots, r_n$ be the values of the `range` variable in increasing order, and $l_1, \ldots, l_n$ be the corresponding log-ratios.
- We are going to consider a random function f such that $f(r_1), f(r_2), \ldots, f(r_n)$ are jointly Gaussian random variables.
- The observations are distributed as $y_i|\boldsymbol{f} \sim N(f(r_i), \sigma^2)$.
- We consider two types of prior distributions for $\boldsymbol{f}$:
  - RW1 model: $f(r_{i+1}) - f(r_i) \sim N(0, \sigma_f^2)$ i.i.d. for every $1 \leq i \leq n-1$
  - RW2 model: $f(r_{i+1}) - 2f(r_i) + f(r_{i-1}) \sim N(0, \sigma_f^2)$ i.i.d. for every $2 \leq i \leq n-1$

```
↪ m.lidar.rw1 <- inla(logratio ~ 0 + f(range, model = "rw1", constr = FALSE),
    data = lidar)
  summary(m.lidar.rw1)
  -----------
  Call:
     c("inla(formula = logratio ~ 0 + f(range, model = \"rw1\", constr =
     FALSE), ", " data = lidar)")
  Time used:
      Pre = 0.339, Running = 0.801, Post = 0.0402, Total = 1.18
  Random effects:
    Name    Model
      range RW1 model

  Model hyperparameters:
                                             mean      sd 0.025quant 0.5quant
  Precision for the Gaussian observations  166.64   17.21     134.99   165.90
  Precision for range                     4409.53 1356.52    2283.68  4235.20
                                          0.975quant    mode
  Precision for the Gaussian observations     202.68  164.60
  Precision for range                        7551.69 3903.24

  Expected number of effective parameters(stdev): 27.74(4.65)
  Number of equivalent replicates : 7.97

  Marginal log-Likelihood:  251.75
```

Example: Smoothing (Lidar data)

↪ As previously, we are able to plot the posterior mean of the fitted values, as well as 95% credible intervals around each position according to the posterior distribution.

```
plot(lidar,main="Lidar logratio vs range,
smoothing with RW1 random effects")

lines(lidar$range,m.lidar.rw1$summary.fitted.values$mean,type='l')
lines(lidar$range,m.lidar.rw1$summary.fitted.values$'0.025quant',
lty=2,col='red')
lines(lidar$range,m.lidar.rw1$summary.fitted.values$'0.975quant',
lty=2,col='red')
```



Sampling from the posterior

- Sampling from the posterior distribution can be done using the `inla.posterior.sample` function. Note that INLA is based on some deterministic approximations, so the samples will be from an approximate posterior (however, in most cases, the approximation error is very small).
- Before using this function, we need to tell INLA to do the calculations needed for posterior sampling. This is done by selecting the option `control.compute = list(config = TRUE)`. The following example obtains some posterior samples from our linear regression model for the mtcars dataset.

```
m.mtcars.I.post=inla(mpg~drat+wt+qsec,data=mtcars1,
                family="gaussian",
                control.family=list(hyper=prec.prior),
                control.fixed=prior.beta,
                control.compute = list(config = TRUE))

nsamp=10000;
mtcars.samples=inla.posterior.sample(n=nsamp, result=m.mtcars.I.post)
```

- Here we are using the priors defined previously for this example. The function `inla.posterior.sample` obtains the posterior samples. It has at minimum two parameters:
  - `n`, the number of samples, and
  - `result`, the INLA model that was fitted previously.

Sampling from the posterior

- The samples can be accessed in several ways. The first approach is to access them directly, with `mtcars.samples[[i]]` contains all of the variables in sample *i*. By printing it out, we get the following.

```
mtcars.samples[[1]]
-------------------
$hyperpar
    Precision for the Gaussian observations: 0.126030219845555
$latent
    A matrix: 36 x 1 of type dbl sample1
    Predictor:1 23.2415031
    Predictor:2 22.6354758
    Predictor:3 26.5603511
    ....
    Predictor:32 24.9711345
    (Intercept):1 11.9322632
    drat:1 1.8406525
    wt:1 -4.4685918
    qsec:1 0.9622053
$logdens

    $hyperpar
        2.20882306711378
    $latent
        146.778726736302
    $joint
        148.987549803416
```

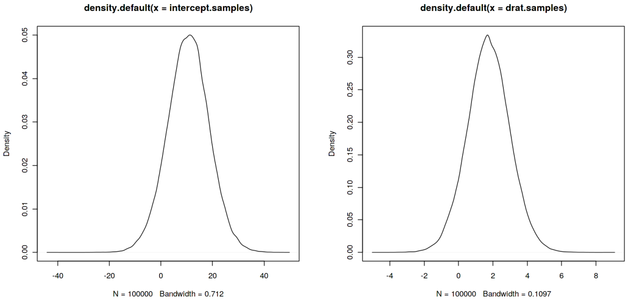
Sampling from the posterior

- We can access the value of the hyperparameter by `mtcars.samples[[1]]$hyperpar`, while the value of the latent variables can be accessed by `mtcars.samples[[1]]$latent[1]`,. . . , `mtcars.samples[[1]]$latent[36]` (there are 36 latent variables in total).
- If we want to collect all of the samples from a single variable (such as the regression coefficient for the intercept, or drat), this can be done more efficiently using the inla.posterior.sample.eval function. This function allows us to evaluate any function on the samples. In particular, it can be used to extract the samples corresponding to a single variable.
- In the code below, we extract the samples for the regression coefficients of intercept, and drat, and plot the estimated densities.

```
intercept.samples=
inla.posterior.sample.eval(function(...) {(Intercept)},
   mtcars.samples)
drat.samples=inla.posterior.sample.eval(function(...) {drat},
   mtcars.samples)

plot(density(intercept.samples))
plot(density(drat.samples))
```

Sampling from the posterior

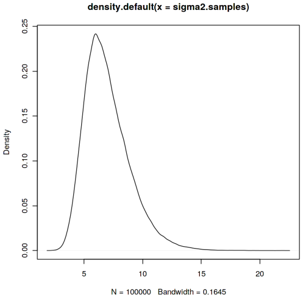
$\hookrightarrow$ We can see that the plots are very similar to what we got previously by plotting the marginals computed by INLA directly.



Sampling from the posterior

↪ Obtaining samples from the hyperparameters is also possible using the samples returned by `inla.posteror.sample`. However, it is faster and more accurate to use the function `inla.hyperpar.sample` instead for this. This function has the same parameters `n` and `result` as previously. We illustrate its usage by sampling from the precision variable, and using the samples to plot the posterior density of the variance parameter σ^2. The plot is very similar to what we have obtained previously using the marginals computed by INLA.

```
precision.samples=
inla.hyperpar.sample(n=nsamp,result=m.mtcars.I.post)
sigma2.samples=1/precision.samples
plot(density(sigma2.samples))
```



Posterior predictive distributions in INLA

- We are going to compute the posterior predictive of the response `mpg` (miles per gallon) for a new car, the Ferrari 488 GTB Coupe.
- The covariates for this car are `drat`= 5.14, `wt`=3.252, `qsec`=10.6.
- We will add this to the dataframe as a new row, with response `mpg` set as NA.

Posterior predictive distributions in INLA

- Now we are going to describe the process to obtain samples from the posterior predictive for the response variable (mpg) of this new data point.
- The first step is to add these covariates as a new row to the dataset, and then set the response variable as NA.

```
mtcars_new=data.frame(mpg=NA, drat= 5.14,wt=3.252,qsec=10.6)
row.names(mtcars_new)<-'Ferrari 488 GTB Coupe'
mtcars2=rbind(mtcars1,mtcars_new)
```

- The second step is to fit the model.

```
m.mtcars.I.post2=inla(mpg~drat+wt+qsec,data=mtcars2,family="gaussian",
                control.family=list(hyper=prec.prior),
                control.fixed=prior.beta,
                control.compute = list(config = TRUE),
                control.predictor = list(compute = TRUE))
```

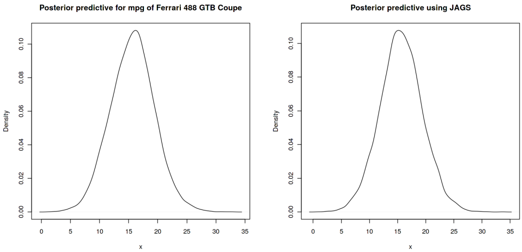
- After this, we obtain posterior samples for the Predictor variable for this new datapoint (located in row 33). The Predictor variables in the sample correspond to the linear predictor η_i in the LGM model. They are not samples from the posterior predictive. The linear predictor is linked to the mean of the observations by $\eta_i = g(\mu_i)$. For this linear regression model, $\eta_i = \mu_i$.

```
nsamp=10000
mtcars.samples2=inla.posterior.sample(n=nsamp, result=m.mtcars.I.post2,
selection = list(Predictor=33))
#selection = list(Predictor=33) means that we only want
#the samples for the linear predictor eta_i of the new datapoint
predictor.samples=inla.posterior.sample.eval(function(...) {Predictor},
mtcars.samples2)
```

Posterior predictive distributions in INLA

- Since our likelihood model is $y_i \sim N(\mu_i, \sigma^2)$, in order to create samples from the posterior predictive, we need to add some Gaussian noise to the samples from the mean $\mu_i = \eta_i$.
- Since σ is also a parameter from the model that is different for each sample, we need to extract σ from the output of `inla.posterior.sample`, and then add the corresponding noise, see our code below.

```
sigma.samples=1/sqrt(
inla.posterior.sample.eval(function(...) {theta}, mtcars.samples2))
post.pred.samples=predictor.samples
+rnorm(n=nsamp,mean=0,sd=sigma.samples)
plot(density(post.pred.samples),xlab="x",ylab="Density",
main="Posterior predictive for mpg of new datapoint")
```



- The true mpg of this car in city is approximately 16.0, which is close to the posterior predictive mean of 15.7.

Model checking in INLA

- We are going to look at several ways of checking models in INLA.
- First, we will redo the standard Q-Q plot and residual checks from Lecture 2 with INLA.
- After this, we will redo the posterior predictive checks from Lecture 2 with INLA.
- Finally, we will look at some new approaches to model checking, using the marginal likelihood, and CPO scores.

Model checking in INLA

- On the `mtcars` Bayesian linear regression example, we can easily obtain posterior samples from the regression coefficients, and the variance parameter σ^2.

```
m.mtcars.I.post=inla(mpg~drat+wt+qsec,data=mtcars1,
                 family="gaussian",
                 control.family=list(hyper=prec.prior),
                 control.fixed=prior.beta,
                 control.compute = list(config = TRUE),
                 control.predictor = list(compute = TRUE))
nsamp=10000
mtcars.samples=inla.posterior.sample(n=nsamp, result=m.mtcars.I.post)

beta0=inla.posterior.sample.eval(function(...) {(Intercept)},
  mtcars.samples)
beta1=inla.posterior.sample.eval(function(...) {drat},
  mtcars.samples)
beta2=inla.posterior.sample.eval(function(...) {wt},
  mtcars.samples)
beta3=inla.posterior.sample.eval(function(...) {qsec},
  mtcars.samples)

sigma2=1/(inla.posterior.sample.eval(function(...) {theta},
  mtcars.samples))
```

Model checking in INLA

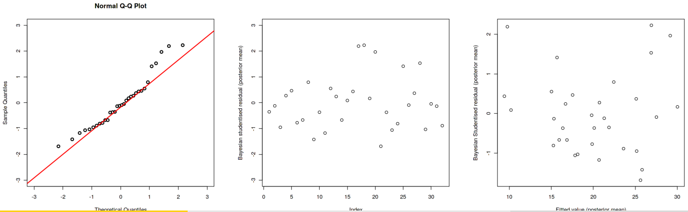
- Using these samples, the fitted values can be computed just as before,

```
fittedvalues=matrix(0,nrow=n,ncol=nsamp)
for(l in 1:nsamp){
fittedvalues[,l]=beta0[l]*x[,1]+beta1[l]*x[,2]
               +beta2[l]*x[,3]+beta3[l]*x[,4] }
```

- Alternatively, we could have obtained the fitted values directly from the samples of the linear predictor without working with the regression coefficients and covariates. In this model the link function is the identity, so fitted values are the same as the linear predictors ($\mathbb{E}(y_i) = \mu_i = \eta_i$)

```
fittedvalues=inla.posterior.sample.eval(function(...) {Predictor},
mtcars.samples)
```

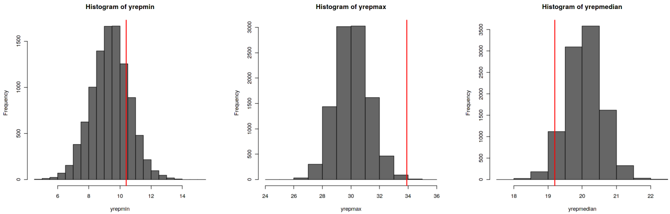
- From these, we are able to compute the studentized residuals, and then do the Q-Q plot, and plot the residuals in terms of their order, and also in terms of the fitted value. The results are similar to what we got from JAGS.



Model checking in INLA

- To obtain the replicate samples in INLA, we will using the Predictor variables from the samples obtained from `inla.posterior.sample`. These store samples from the linear predictors η_i for each datapoint. We need to add some Gaussian noise according to the samples from σ, as we have done for the posterior predictive previously.

```
predictor.samples=inla.posterior.sample.eval(function(...) {Predictor},
   mtcars.samples)
sigma.samples=1/sqrt(inla.posterior.sample.eval(function(...) {theta},
   mtcars.samples))
yrep=matrix(0,nrow=n,ncol=nsamp)
for(row.num in 1:n){
    yrep[row.num, ]<-
    predictor.samples[row.num, ]+rnorm(n=nsamp,mean=0,sd=sigma.samples)
}
```



Model checking in INLA

- INLA provides a number of Bayesian criteria for model assessment.
- Marginal likelihood is an useful criteria when comparing different models. It is defined as
$$m(\boldsymbol{y}) = \int_{\boldsymbol{x},\boldsymbol{\theta}} p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})\pi(\boldsymbol{x},\boldsymbol{\theta}),$$
and it describes the overall fit of the model, including the prior distribution, on the data. INLA computes $log(m(\boldsymbol{y}))$ by default, and displays it in the summary of the model. Larger $log(m(\boldsymbol{y}))$ values correspond to better model fit.
- Conditional predictive ordinate (CPO) is a cross-validation type model assessment criterion, which is defined as
$$CPO_i = p(y_i|y_{-i}),$$
for every observation $1 \leq i \leq n$. This quantifies how likely is the observation i given the rest of the observations given the model. We can summaries these values in a single number by computing
$$NLSCPO = -\sum_{i=1}^{n} \log(p(y_i|y_{-i})).$$
Smaller values correspond to better model fit.

Model checking in INLA

- Predictive integral transform (PIT) measures for each observation the value of the CDF of the posterior predictive distribution of this observation evaluated at the observation value. It is defined as

$$PIT_i = p(y_i^{new} \leq y_i | y_{-i}).$$

  In case of a perfect model, PIT_i are uniformly distributed on $[0, 1]$ for every i. Hence we can evaluate the model fit by looking at the distribution of $PIT_1, \ldots, PIT_n$.

- Deviance information criterion (DIC) was introduced by Spiegelhalter et al. (2002). It is similar to AIC. It takes into account the goodness of fit of the model, and adds a penalty term that is based on the complexity of the model via the estimated number of parameters. It is defined as

$$DIC = D(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\theta}}) + 2p_D,$$

  where D is the deviance function, $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{\theta}}$ are the posterior means of the hyperparameters $\boldsymbol{\theta}$ and latent effects $\boldsymbol{x}$, and p_D is the effective number of parameters, defined as $p_D = \mathbb{E}(D(\boldsymbol{x}, \boldsymbol{\theta})|\boldsymbol{y}) - D(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\theta}})$. Smaller DIC values correspond to better fit.

Model checking in INLA

↪ These model assessment criteria can be computed in INLA by setting `control.compute=list(cpo=TRUE, dic=TRUE)`. We do this for the robust regression example on the Scottish hills racing dataset.

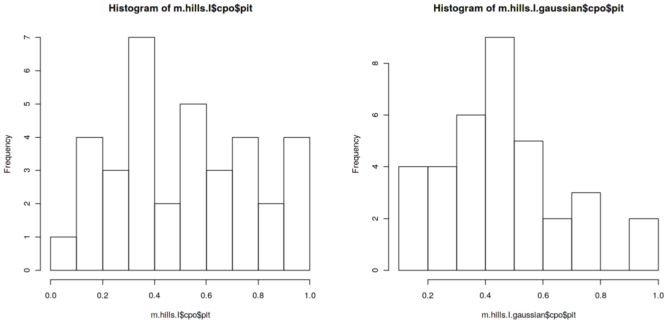
```
m.hills.I <- inla(time ~ 1+climb+dist,family="T",data=hills,
                  control.family=list(hyper=prior.t),
                  control.fixed=prior.fixed,
                  control.compute=list(cpo=TRUE, dic=TRUE))
cat("DIC:",m.hills.I$dic$dic,"\n")
cat("NSLCPO:",-sum(log(m.hills.I$cpo$cpo)),"\n")
cat("Log marginal likelihood:",m.hills.I$mlik[1],"\n")
hist(m.hills.I$cpo$pit)
------
DIC: 264.55
NSLCPO: 133.6297
Log marginal likelihood: -149.7289
```

Model checking in INLA

↪ We also fit a standard linear regression model with Gaussian noise.

```
m.hills.I.gaussian <-
inla(time ~ 1+climb+dist,family="gaussian",
data=hills,control.family=list(hyper=prec.prior),
control.fixed=prior.fixed,
control.compute=list(cpo=TRUE, dic=TRUE))
cat("DIC:",m.hills.I.gaussian$dic$dic,"\n")
cat("NSLCPO:",-sum(log(m.hills.I.gaussian$cpo$cpo)),"\n")
cat("Log marginal likelihood:",m.hills.I.gaussian$mlik[1],"\n")
#We display a histogram of the PIT values
hist(m.hills.I.gaussian$cpo$pit)
------
DIC: 292.7035
NSLCPO: 152.6384
Log marginal likelihood: -162.1553
```

Model checking in INLA



$\hookrightarrow$ The robust model is better according to all 4 criteria.

Summary

- ↪ INLA allows for computationally efficient Bayesian inference for a large class of LGMs.
- ↪ There are many useful models implemented, and all of the usual Bayesian computations can be done, including sampling from the posterior, and posterior predictives.
- ↪ INLA also allows for model checking and comparison using various statistics (marginal likelihood, CPO, DIC).
- ↪ JAGS is more flexible than INLA, and it allows for almost any Bayesian model. Moreover, there are no deterministic approximations used, so as the number of MCMC samples tends to infinity, the samples become exactly from the posterior.
- ↪ A drawback is that mixing can become slow in high dimensions, especially when there are strong correlations between the model variables.