NTNU
Norwegian University of
Science and Technology

Introduction to INLA

Andrea Riebler <andrea.riebler@math.ntnu.no>
Department of Mathematical Sciences, NTNU

Outline

Introduction

Bayesian hierarchical models

Latent Gaussian models

Deterministic inference

R-INLA

http://www.r-bloggers.com/inla-bayes-goes-to-norway/

What?

The short answer:

INLA is a fast method to do Bayesian inference with latent Gaussian models and R-INLA is an R-package that implements this method with a flexible and simple interface.

A much longer answer:

Rue, Martino, and Chopin (2009) "Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations." *Journal of the royal statistical society: Series B*. 319–392

INLA - how it all began

- INLA is the result of many years of work, mainly driven by Håvard Rue
- the first “user-friendly” implementation was built on a C-library in 2007
- The first prototype of the R-interface came in Jan/Feb 2008
- today, the complete source-code is about 100 000 lines, in R/C/C++ (view/track/download from inla.googlecode.com)

... and the INLA group increased

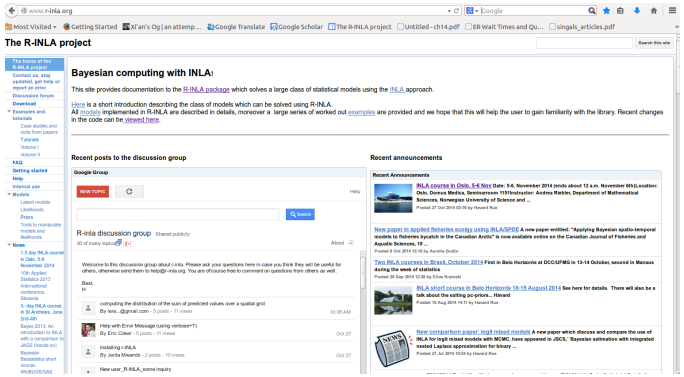
(photo 2011)

There are more:

...

Where?

The software, information, examples and help can be found at `http://www.r-inla.org`



A complete documentation about INLA is in progress.

So... When can `R-INLA` be used?

- What type of problems can it solve?
- What type of models can be used?

The core

We have observed something.

We have questions.

We want answers.

How do we find answers?

We need to make choices:

Bayesian or frequentist?

How do we model the data?

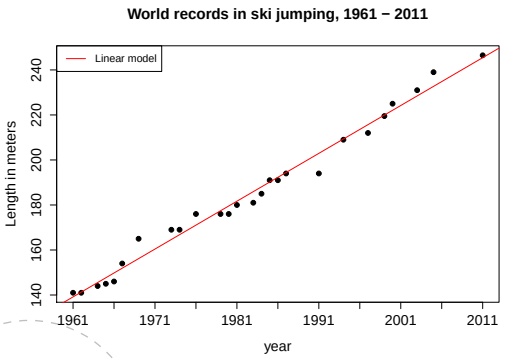
How do we compute the answer?

These questions are *not* independent.

Example: Ski flying records

Assume a simple linear regression model with Gaussian observations $\boldsymbol{y} = (y_1, \ldots, y_n)$, where

$$E(y_i) = \mu + \beta x_i, \quad \text{Var}(y_i) = \tau^{-1}, \quad 1, \ldots, n$$

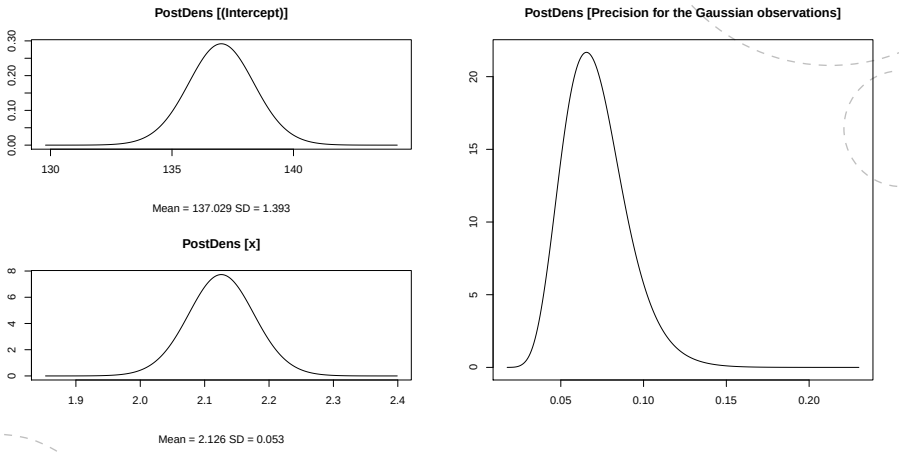


Estimates

Intercept: 137.03 (1.42195),
β: 2.13 (0.05)

The Bayesian approach

Assign priors to the parameters α, β and τ and calculate posteriors:



Real-world datasets are usually much more complicated!

Using a Bayesian framework:

- Build (hierarchical) models to account for potentially complicated dependency structures in the data.
- Attribute uncertainty to model parameters and latent variables using priors.

Two main challenges:

- Need computationally efficient methods to calculate posteriors.
- Select priors in a sensible way.

Bayesian hierarchical models

INLA can be used with Bayesian hierarchical models where we model in different stages or levels:

Stage 1: What is the distribution of the responses?

Stage 2: What is the distribution of the underlying unobserved (latent) components?

Stage 3: What are our prior beliefs about the parameters controlling the components in the model?

Stage 1

How is our data ($\boldsymbol{y}$) generated from the underlying components ($\boldsymbol{x}$) and hyperparameters ($\boldsymbol{\theta}$) in the model:

- Gaussian response?
- Count data? (E.g. Poisson, negative binomial)
- Spatial point pattern?
- Binary data?
- ...

This information is placed into our *likelihood* $\pi(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$

Stage 2

The underlying unobserved components $\boldsymbol{x}$ are called **latent components** and can be:

- Covariates
- Unstructured random effects (individual effects, group effects)
- Structured random effects, for example an autorgressive process of order 1 (AR(1)-process).

These are linked to the responses in the likelihood through linear predictors.

Stage 3

The likelihood and the latent model typically have hyperparameters that control their behavior. The hyperparameters $\boldsymbol{\theta}$ can include:

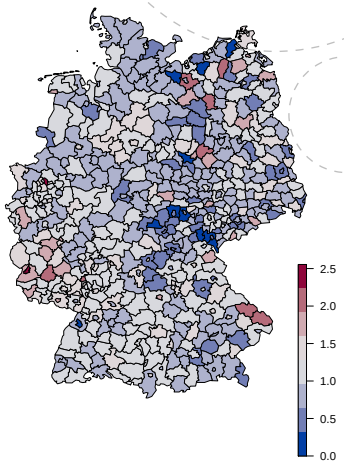
- Variance of observation noise
- Variance of unstructured effects
- Correlation of multivariate effects
- Autocorrelation parameter

Example: Disease mapping in Germany

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

Information available:

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



Stage 1: The data

First we decide on the likelihood for our data $\boldsymbol{y}$

- Our responses are counts
- We decide to model our responses as

$$y_i \mid \eta_i \sim \text{Poisson}(E_i \exp(\eta_i))$$

- η_i is a linear function of the latent components

Stage 2: The latent model

The latent field $\boldsymbol{x}$ consists of two parts:

1. One fixed effect: the intercept μ
2.
   - The spatially structured effect f_s.
   - The unstructured effect $\boldsymbol{u}$ which accounts for non-observed variability
   - The unknown effect $f(c_i)$ of the exposure covariate which assumes value c_i for district i.

These are combined for each location to give a linear predictor

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

The latent field is $\boldsymbol{x} = (\mu, \{f_s(\cdot)\}, \{f(\cdot)\}, u_1, u_2, \ldots, u_n)$

Stage 3: Hyperparameters

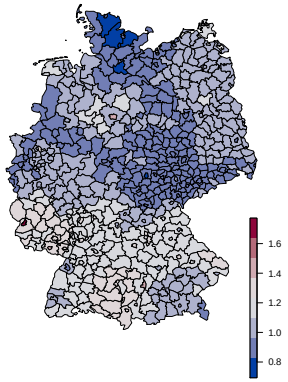
The structured and unstructured spatial effect as well as the smooth covariate effect will be each controlled by one parameter

— $\tau_c, \tau_f, \tau_\eta$: The precisions (inverse variances) of the covariate effect, spatial effect and unstructured effect, respectively.

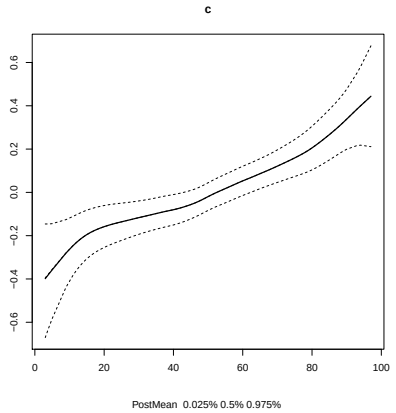
The hyperparameters are $\boldsymbol{\theta} = (\tau_c, \tau_f, \tau_\eta)$, and must be given a prior $\pi(\tau_c, \tau_f, \tau_\eta)$.

Quantities of interest

Structured spatial effect $exp(f_s(s_i))$



Covariate effect $f(c_i)$



Latent Gaussian models

This example is just one example of a very useful class of models called **Latent Gaussian models**.

— The characteristic property is that the latent part of the hierarchical model is Gaussian, $\boldsymbol{x}|\boldsymbol{\theta} \sim N(0, \mathbf{Q}^{-1})$

— The expected value is **0**

— The *precision* matrix (inverse covariance matrix) is **Q**

The general set-up

The set up contains GLMs, GLMMs, GAMs, GAMMs, and more. The mean of the observation i, μ_i, is connected to the linear predictor, η_i, through a link function g,

$$\eta_i = g(\mu_i) = \mu + \boldsymbol{z}_i^\top \boldsymbol{\beta} + \sum_\gamma w_{\gamma,i} f_\gamma(c_{\gamma,i}) + u_i, \quad i = 1, 2, \ldots, n$$

where

μ : Intercept
$\boldsymbol{\beta}$: Fixed effects of covariates $\boldsymbol{z}$
$\{f_\gamma(\cdot)\}$: Non-linear/smooth effects of covariates $\boldsymbol{c}$
$\{w_{\gamma,i}\}$: Known weights defined for each observed data point
$\boldsymbol{u}$: Unstructured error terms

Specification of the latent field

- Collect all parameters (random variables) in the linear predictor in a latent field $\boldsymbol{x} = \{\mu, \boldsymbol{\beta}, \{f_\gamma(\cdot)\}, \boldsymbol{\eta}\}$.
- A latent Gaussian model is obtained by assigning Gaussian priors to all elements of $\boldsymbol{x}$.
- Very flexible due to many different forms of the unknown functions $\{f_\lambda(\cdot)\}$:
- Hyperparameters account for variability and length/strength of dependence

Flexibility through f-functions

The functions $\{f_\gamma\}$ in the linear predictor make it possible to capture very different types of random effects in the same framework:

- $f(\texttt{time})$: For example, an AR(1) process, a random walk of first or second order (RW1 or RW2)
- $f(\texttt{spatial location})$: For example, a Matérn field
- $f(\texttt{covariate})$: For example, a RW1 or RW2 on the covariate values
- $f(\texttt{time}, \texttt{spatial location})$ can be a spatio-temporal effect
- And much more

Additivity

- One of the most useful features of the framework is the additivity.
- Effects can easily be removed and added without difficulty.
- Each component might add a new latent part and might add new hyperparameters, but the modelling framework and computations stay the same.

A small point to think about

From a Bayesian point of view fixed effects and random effects are all the same.

- Fixed effects are also random
- They only differ in the prior we put on them

Example: Smoothing binary time-series

- Have observed a sequence $y_1, y_2, \ldots, y_n$ of 0s and 1s
- Each time t has an associated covariate x_t
- We want to smooth the time series by inferring the sequence p_t, for $t = 1, 2, \ldots, n$, of probabilities for 1s at each time step

Example: Smoothing time series

Stage 1: We choose a Bernoulli distribution for the responses, so that

$$y_t|\eta_t \sim \text{Bernoulli}\left(\frac{\exp(\eta_t)}{1+\exp(\eta_t)}\right)$$

Stage 2: Covariates, AR(1) component, i.e. $a_t = \rho a_{t-1} + \epsilon_t$, and random noise are connected to likelihood by

$$\eta_t = \beta_0 + \beta_1 x_t + a_t + v_t$$

Stage 3: ρ: Dependence parameter in AR(1) process

σ_a^2: Marginal variance in AR(1) process

σ_v^2: Variance of unstructured term

Loads of examples

- Generalized linear and additive (mixed) models
- Disease mapping
- Survival analysis
- Log-Gaussian Cox-processes
- Spatio and spatio-temporal models
- Stochastic volatility models
- Measurement error models
- And more!

Computations

So...

Now we have a modelling framework

But how do we get our answers?

What do we care about?

It depends on the problem!

- A single element of the latent field (e.g. the sign or quantiles of a fixed effect)
- A linear combination of elements from the latent field (the average over an area of a spatial effect, the difference of two effects)
- A single hyperparameter (the correlation)
- Predictions at unobserved locations

What do we need to compute?

Often we are interested in the posterior probability density of an element of the latent field

$$\pi(x_i|\boldsymbol{y})$$

or the posterior probability density of an element of the hyperparameters

$$\pi(\theta_j|\boldsymbol{y})$$

or some other statistics

$$\pi(f(\boldsymbol{x},\boldsymbol{\theta})|\boldsymbol{y})$$

But, as always in Bayesian statistics, we need to do high-dimensional integrals that cannot be computed analytically.

Traditional approach: MCMC*

Based on sampling. Construct Markov chains with the target posterior as stationary distribution.

- Extensively used within Bayesian inference since the 1980's.
- Flexible and general, sometimes the only thing we can do!
- A generic tool is available with `JAGS`/`OpenBUGS`.
- Tools for specific models are of course available, e.g. `BayesX` and `stan`.

* Markov chain Monte Carlo

Approximate inference

Bayesian inference can (almost) never be done exactly. Some form of approximation must always be done.

- MCMC "works" for everything, but it can be incredibly slow
- Is it possible to make a quicker, more specialized inference scheme which only needs to work for this limited class of models?

Recall: What is our model framework?

Latent Gaussian models

$$\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta} \sim \prod_i \pi(y_i|\eta_i,\boldsymbol{\theta})$$

$$\boldsymbol{x}|\boldsymbol{\theta} \sim \pi(\boldsymbol{x}|\boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}(\boldsymbol{\theta})^{-1}) \qquad \text{Gaussian!}$$

$$\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta}) \qquad \text{Not Gaussian}$$

where the precision matrix $\mathbf{Q}(\boldsymbol{\theta})$ is sparse. Generally these "sparse" Gaussian distributions are called Gaussian Markov random fields (GMRFs).

The sparseness can be exploited for very quick computations for the Gaussian part of the model through numerical algorithms for sparse matrices.

The INLA idea

Use the posterior distribution

$$\pi(\boldsymbol{x},\boldsymbol{\theta} \mid \boldsymbol{y}) \propto \pi(\boldsymbol{\theta})\pi(\boldsymbol{x} \mid \boldsymbol{\theta})\pi(\boldsymbol{y} \mid \boldsymbol{x},\boldsymbol{\theta})$$

to approximate the posterior marginals

$$\pi(x_i \mid \boldsymbol{y}) \quad \text{and} \quad \pi(\theta_j \mid \boldsymbol{y})$$

directly.

Let us consider a toy example to illustrate the ideas.

Smoothing noisy observations (I)

Observations

$$y_i = m(i) + \epsilon_i, \qquad i = 1,\dots,n$$

for Gaussian iid noise ϵ_i with *known* precision.

Will assume $m(i)$ is a smooth function wrt i

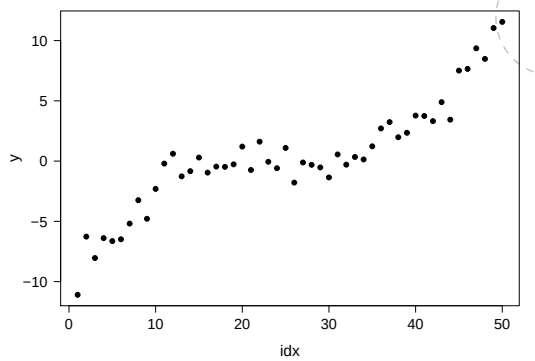
How does INLA work?

Observations

$$y_i = m(i) + \epsilon_i, \qquad i = 1,\dots,n$$

Here, we assume that $m(i)$ is a smooth function wrt i and $\epsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0,\tau_0)$ with *known* precision τ_0.

```
1 n = 50
2 idx = 1:n
3 fun = 100*((idx-n/2)/n)^3
4 y = fun + rnorm(n, mean
      =0, sd=1)
5 plot(idx, y)
```



Assumed hierarchical model

1. Data: Gaussian observations with known precision

$$y_i \mid x_i, \theta \sim \mathcal{N}(x_i, \tau_0)$$

2. Latent model: A Gaussian model for the smooth function[1]

$$\pi(\boldsymbol{x} \mid \theta) \propto \theta^{(n-2)/2} \exp\left(-\frac{\theta}{2}\sum_{i=2}^{n}(x_i - 2x_{i-1} + x_{i-2})^2\right)$$

3. Hyperparameter: The smoothing parameter θ which we assign a $\Gamma(a, b)$ prior

$$\pi(\theta) \propto \theta^{a-1} \exp(-b\theta), \quad \theta > 0$$

[1] `model="rw2"`

Derivation of posterior marginals (I)

Since

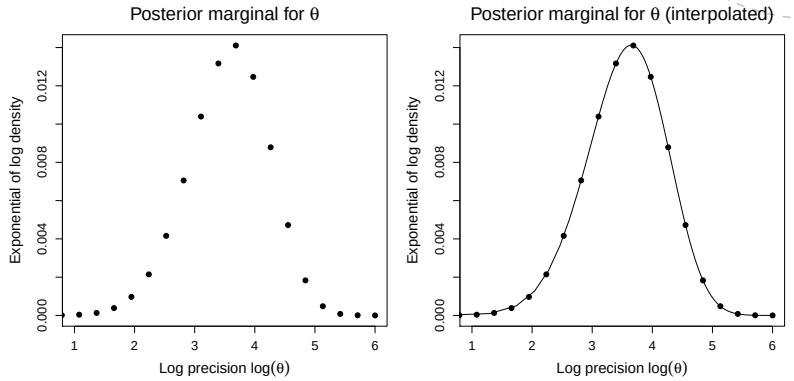
$$\boldsymbol{x}, \boldsymbol{y} \mid \theta \sim \mathcal{N}(\cdot, \cdot)$$

(derived using $\pi(\boldsymbol{x}, \boldsymbol{y} \mid \theta) \propto \pi(\boldsymbol{y} \mid \boldsymbol{x}, \theta)\, \pi(\boldsymbol{x} \mid \theta)$),
we can compute (numerically) all marginals, using that

$$\pi(\theta \mid \boldsymbol{y}) \propto \frac{\overbrace{\pi(\boldsymbol{x}, \boldsymbol{y} \mid \theta)}^{\text{Gaussian}}\ \pi(\theta)}{\underbrace{\pi(\boldsymbol{x} \mid \boldsymbol{y}, \theta)}_{\text{Gaussian}}}$$

Posterior marginal for hyperparameter

Select a grid of points to represent the density $\theta \mid \boldsymbol{y}$. (Here, they points are chosen to be equi-distant.)



Derivation of posterior marginals (II)

From

$$\boldsymbol{x} \mid \boldsymbol{y}, \theta \sim \mathcal{N}(\cdot, \cdot)$$

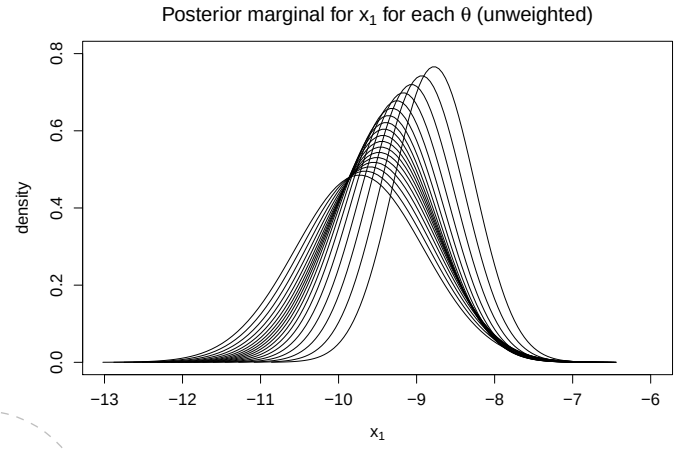
we can compute

$$\pi(x_i \mid \boldsymbol{y}) = \int \underbrace{\pi(x_i \mid \theta, \boldsymbol{y})}_{\text{Gaussian}}\ \pi(\theta \mid \boldsymbol{y})\, d\theta$$
$$\approx \sum_k \pi(x_i \mid \theta_k, \boldsymbol{y}) \pi(\theta_k \mid \boldsymbol{y}) \Delta_k$$

where θ_k, $k = 1, \ldots, K$, correspond to representative points of $\theta \mid \boldsymbol{y}$ and Δ_k are the corresponding weights.

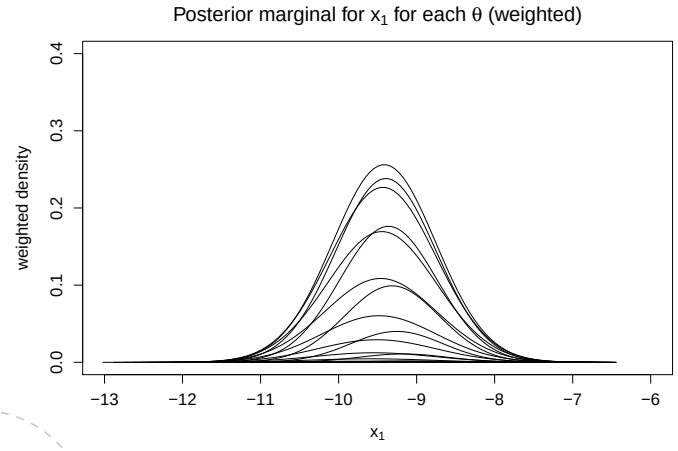
Posterior marginal for latent parameters

Compute the conditional posterior marginal for each x_i given each θ_k.



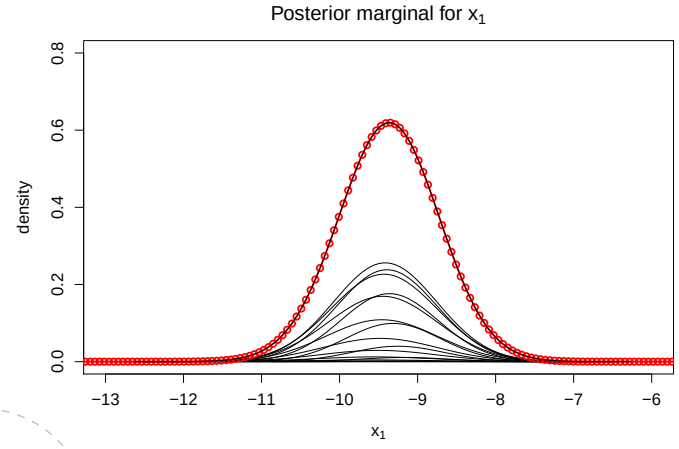
Posterior marginal for latent parameters

Weigh the resulting (conditional) marginal posteriors by the density associated with each θ_k.



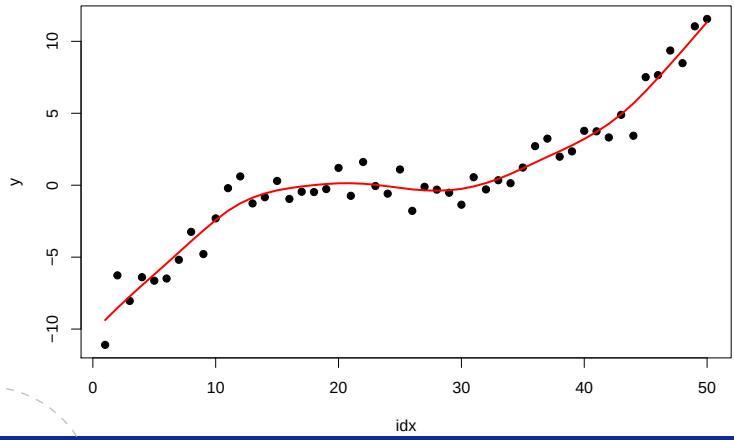
Posterior marginal for latent parameters

Numerically sum over all conditional densities to obtain the posterior marginal for each x_i.



Fitted spline

The posterior marginals are used to calculate summary statistics, like means, variances and credible intervals:



Extensions

This is the basic idea behind INLA. It is quite simple.

However, we need to extend this basic idea so we can deal with

- More than one hyperparameter
- Non-Gaussian observations

The non-Gaussian part of the model

- In many cases $\pi(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{\theta})$ is very close to a Gaussian distribution, and can be replaced with a Laplace approximation
- This means that all the really hard, high-dimensional integrals with respect to the latent field are easy, and only the integrals with respect to the hyperparameters remain
- If the number of hyperparameters is low, these integrals can be done efficiently numerically

Limitations

- The dimension of the latent field $\boldsymbol{x}$ can be large (10^2–10^6)
- But the dimension of the hyperparameters $\boldsymbol{\theta}$ must be small (≤ 9)

In other words, each random effect can be big, but there cannot be too many random effects unless they share parameters.

How to use INLA?

INLA is implemented through the package `R-INLA` in the `R` software which

- has a very user friendly `formula` interface

```
linear_model <- lm(weight ~ group)
```

Fits the linear model

$$\text{weight}_i = \mu + \text{group}_i + \epsilon_i$$

Example: Ski flying records

Steps to run INLA

1. Make an object to store responses and covariates

```
data = list(y = y, x = x)
```

2. Make a formula specifying the model

```
formula = y~x
```

3. Call `INLA`

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


Example: Summary

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

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

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


Summary of INLA

Three main ingredients in INLA

- Latent Gaussian models
- Laplace approximations
- Gaussian Markov random fields

These ingredients give a very nice tool for Bayesian inference which is

- fast
- accurate
- scales well for moderate sizes