



NTNU
Norwegian University of
Science and Technology

Bayesian computation using INLA

Daniel Simpson

Institutt for matematiske fag

May, 2011

Outline

Introduction

A brief introduction to MCMC

The problem with MCMC scheme

A case for ‘approximate’ inference

Latent Gaussian Models

Gaussian Markov random fields

Gaussian Markov random fields

Computing with GMRFs

A case for approximate inference

Hierarchical Bayesian models

Hierarchical models are an extremely useful tool in Bayesian model building.

Three parts:

- *Observations (\mathbf{y})*: Encodes information about observed data, including design and collection issues.

Hierarchical Bayesian models

Hierarchical models are an extremely useful tool in Bayesian model building.

Three parts:

- *Observations (y)*: Encodes information about observed data, including design and collection issues.
- *The latent process (x)*: The unobserved process. May be the focus of the study, or may be included to reduce autocorrelation.

Hierarchical Bayesian models

Hierarchical models are an extremely useful tool in Bayesian model building.

Three parts:

- *Observations (y)*: Encodes information about observed data, including design and collection issues.
- *The latent process (x)*: The unobserved process. May be the focus of the study, or may be included to reduce autocorrelation.
- *The Parameter model (θ)*: Models for all of the parameters in the observation and latent processes.

First example: Spatial logistic regression

Observation process:

$$y_i \sim \text{Bin}(1, p_i)$$

Latent field:

$$\text{logit}(p_i) \sim N(0, \phi^{-1} \Sigma)$$

Parameters:

$$\phi \sim \text{log-gamma}(\alpha)$$

Bayesian computing

We are interested in the posterior marginal quantities like $\pi(x_i|\mathbf{y})$ and $\pi(\theta_i|\mathbf{y})$.

This requires the evaluation of integrals of the form

$$\pi(x_i|\mathbf{y}) \propto \int_{x_{\{-i\}}} \int_{\theta} \pi(y|\mathbf{x}, \theta) \pi(\mathbf{x}|\theta) \pi(\theta) d\theta dx_{\{-i\}}$$

The computation of massively high dimensional integrals is at the core of Bayesian computing.

But surely we can already do this

- The issue of Bayesian computing is not “solved” even though MCMC is available
- Hierarchical models are more difficult for MCMC
- A main obstacle for Bayesian modelling is the issue of “Bayesian computing”
- Currently, only one generic tool is available OpenBUGS, based on MCMC
- Specific tools are of course available for specific models, like BayesX, mainly based on MCMC.

MCMC

Construct a Markov chain

$$\eta^{(1)}, \eta^{(2)}, \dots, \eta^{(k)}, \dots$$

that converges (under some conditions) to $\pi(\eta)$.

MCMC

Construct a Markov chain

$$\eta^{(1)}, \eta^{(2)}, \dots, \eta^{(k)}, \dots$$

that converges (under some conditions) to $\pi(\eta)$.

1. Start with $\eta^{(0)}$ where $\pi(\eta^{(0)}) > 0$. Set $k = 1$.

MCMC

Construct a Markov chain

$$\boldsymbol{\eta}^{(1)}, \boldsymbol{\eta}^{(2)}, \dots, \boldsymbol{\eta}^{(k)}, \dots$$

that converges (under some conditions) to $\pi(\boldsymbol{\eta})$.

1. Start with $\boldsymbol{\eta}^{(0)}$ where $\pi(\boldsymbol{\eta}^{(0)}) > 0$. Set $k = 1$.
2. Generate a *proposal* $\boldsymbol{\eta}^*$ from some *proposal kernel* $q(\boldsymbol{\eta}^* | \boldsymbol{\eta}^{(k-1)})$.
Set $\boldsymbol{\eta}^{(k)} = \boldsymbol{\eta}^*$ with probability

$$\alpha = \min \left\{ 1, \frac{\pi(\boldsymbol{\eta}^*)}{\pi(\boldsymbol{\eta}^{(k-1)})} \frac{q(\boldsymbol{\eta}^{(k-1)} | \boldsymbol{\eta}^*)}{q(\boldsymbol{\eta}^* | \boldsymbol{\eta}^{(k-1)})} \right\};$$

otherwise set $\boldsymbol{\eta}^{(k)} = \boldsymbol{\eta}^{(k-1)}$.

MCMC

Construct a Markov chain

$$\boldsymbol{\eta}^{(1)}, \boldsymbol{\eta}^{(2)}, \dots, \boldsymbol{\eta}^{(k)}, \dots$$

that converges (under some conditions) to $\pi(\boldsymbol{\eta})$.

1. Start with $\boldsymbol{\eta}^{(0)}$ where $\pi(\boldsymbol{\eta}^{(0)}) > 0$. Set $k = 1$.
2. Generate a *proposal* $\boldsymbol{\eta}^*$ from some *proposal kernel* $q(\boldsymbol{\eta}^* | \boldsymbol{\eta}^{(k-1)})$.
Set $\boldsymbol{\eta}^{(k)} = \boldsymbol{\eta}^*$ with probability

$$\alpha = \min \left\{ 1, \frac{\pi(\boldsymbol{\eta}^*)}{\pi(\boldsymbol{\eta}^{(k-1)})} \frac{q(\boldsymbol{\eta}^{(k-1)} | \boldsymbol{\eta}^*)}{q(\boldsymbol{\eta}^* | \boldsymbol{\eta}^{(k-1)})} \right\};$$

otherwise set $\boldsymbol{\eta}^{(k)} = \boldsymbol{\eta}^{(k-1)}$.

3. Set $k = k + 1$ and go back to 2

Depending on the specific choice of the proposal kernel $q(\eta^*|\eta)$, very different algorithms result.

Depending on the specific choice of the proposal kernel $q(\eta^*|\eta)$, very different algorithms result.

- When $q(\eta^*|\eta)$ does not depend on the current value of η proposal is called an *independence proposal*.

Depending on the specific choice of the proposal kernel $q(\eta^*|\eta)$, very different algorithms result.

- When $q(\eta^*|\eta)$ does not depend on the current value of η proposal is called an *independence proposal*.
- When $q(\eta^*|\eta) = q(\eta|\eta^*)$ we have a *Metropolis proposal*. These includes so-called *random-walk proposals*.

Depending on the specific choice of the proposal kernel $q(\eta^*|\eta)$, very different algorithms result.

- When $q(\eta^*|\eta)$ does not depend on the current value of η proposal is called an *independence proposal*.
- When $q(\eta^*|\eta) = q(\eta|\eta^*)$ we have a *Metropolis proposal*. These includes so-called *random-walk proposals*.

The rate of convergence toward $\pi(\eta)$ and the degree of dependence between successive samples of the Markov chain (*mixing*) will depend on the chosen proposal.

So what's wrong with MCMC?

It is { very
extremely
unspeakably
unbelievably
exceptionally slow.
extraordinarily
terrifically
remarkably
impractically }

So what's wrong with MCMC?

This is actually a problem with *any* Monte-Carlo scheme.

Error in expectations

The Monte-Carlo error is

$$\text{Var} \left(\mathbb{E}(f(X)) - \frac{1}{N} \sum_{i=1}^N f(x_i) \right) = \mathcal{O} \left(\frac{1}{\sqrt{N}} \right)$$

In practical terms, to reduce the variance to $\mathcal{O}(10^{-p})$ you need $\mathcal{O}(10^{2p})$ samples!

This can be optimistic!

So what's good about MCMC?

Sometimes it's the only thing we can do.

A case for ‘approximate’ inference

Except for the (trivial) cases when everything can be computed exactly (maybe up to very small integration error), *we can never do exact inference.*

Facetious principle

If we’re going to be wrong, we might as well be wrong quickly.

MCMC method violate this principle.

Be more narrow: Don't look for universal algorithms

So how do we design fast approximate inference schemes?

- Focus on specific model classes—MCMC ‘works’ for everything, but it doesn’t work well.
- Don’t be afraid of complicated algorithms—Anyone can program basic MCMC, this does not make the method better.
- **Provide good, user friendly software.**

R—A language for statistical computing

R is the most popular computing language in applied statistics.

- It is open source and its popularity stems from its extendibility and the range of packages available.
- Solve GLM-type models very well.
- It is a complete programming language, albeit one with somewhat idiosyncratic ‘features’.
- It plays nicely with compiled languages like C++ and FORTRAN.

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

Fits the linear model

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

Our first INLA example: Binomial regression with random effects

This is a Winbugs/ Openbugs example.

- Two types of seeds were planted and treated with one of two root extracts on one of 21 plates arranged in a 2×2 factorial design.
- The number that germinated was measured.
- The sampling model is $y_i|\eta_i, n_i \sim \text{Bin}(n_i, p_i)$
- The probabilities p_i are modelled through a logit link

$$\text{logit}(p_i) = \mu + \beta_1 x_1 + \beta_w x_2 + \beta_3 x_1 x_2 + f(\text{plate}),$$

where x_1 is the seed type and x_2 is the root extract.

- The random effect $f(\text{plate}_i)|\tau \sim N(0, \tau^{-1})$... A random intercept model

And now to r-INLA

The file is seeds.R.

```
require(INLA)

#Load up the data - in the INLA package
data(Seeds)

# Define your formula. response ~ Latent field
#Random effects are formulated through the f( ) function

formula = r ~ x1 + x2 +x1*x2 + f(plate,model="iid")
```

And now to r-INLA

The file is seeds.R.

```
#Run INLA.  
#‘family’ defines the link function  
#‘Ntrials’ - the number of trials (just for binomial)  
#data - the data ;)  
mod.seeds = inla(formula,family="binomial",Ntrials=n,data=Se  
  
#View the results  
summary(mod.seeds)  
  
#We can make the inference better  
hyp.seeds = inla.hyperpar(mod.seeds)  
summary(hyp.seeds)  
plot(hyp.seeds)
```

Results

```
> summary(hyp.seeds)
```

Call:

```
"inla(formula = formula, data = Seeds)"
```

Time used:

Pre-processing	Running inla	Post-processing	Total
0.2993040	0.4662671	0.3413241	1.1068952

Fixed effects:

	mean	sd	0.025quant	0.5quant	0.975quant	kld
(Intercept)	20.137156	5.965280	8.420878	20.116364	32.09810	0.000000e+00
x1	-11.105922	8.773308	-28.114186	-11.155184	6.23786	0.000000e+00
x2	11.894063	8.500429	-4.692228	11.888688	28.58836	1.966702e-16
x1:x2	-7.634212	11.330760	-29.712071	-7.718478	14.95064	0.000000e+00

Random effects:

Name	Model	Max KLD
plate	IID model	

Model hyperparameters:

	mean	sd	0.025quant	0.5quant	0.975quant
Precision for the Gaussian observations	5.600e-03	1.783e-03	2.663e-03	5.411e-03	9.617e-03
Precision for plate	2.283e+04	1.771e+04	4.259e+03	1.741e+04	7.068e+04

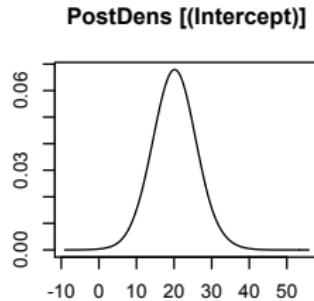
Expected number of effective parameters(std dev): 3.75(0.06926)

Number of equivalent replicates : 5.60

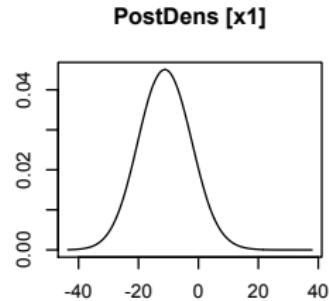
Marginal Likelihood: -90.06

Warning: Interpret the marginal likelihood with care if the prior model is improper.

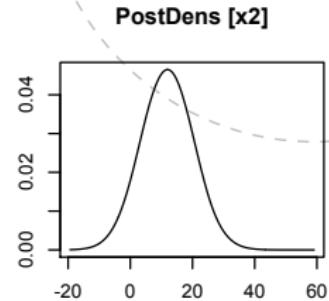
Results



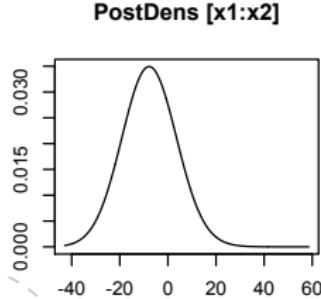
Mean = 20.137 SD = 5.965



Mean = -11.106 SD = 8.773



Mean = 11.894 SD = 8.5



Mean = -7.634 SD = 11.331

Course outline

Long Lecture 1: Introduction, Latent Gaussian Models, GMRFs, GMRF Computations

Lunch

Long Lecture 2: GMRF Computations, Intrinsic GMRFs and SPDEs, The INLA approximation, Examples.

Outline

Introduction

Latent Gaussian Models

Example III

Latent Gaussian Models

GMRFs

Summing up

Illustrative Example

Gaussian Markov random fields

Gaussian Markov random fields

Computing with GMRFs

A case for approximate inference

Latent Gaussian Models

Question: What type of models will we be trying to fit?

Answer: Latent Gaussian models (LGMs).

Latent Gaussian Models

Question: What type of models will we be trying to fit?

Answer: Latent Gaussian models (LGMs).

Question: What is a Latent Gaussian Model?

Answer: Proof by example.

GxxMs—Different names for the same thing

GLM/GAM/GLMM/GAMM/++

- Perhaps the most important class of statistical models
- Many “models” can be cast in to this class without knowing
- No good (enough) MCMC solution around
- Even frequentist approaches does not scale well computationally

Bayesian GLM/GAM/GLMM/GAMM/+++

Linear predictor

$$\eta = \mu \mathbf{1} + \mathbf{A}\beta + \sum_i \mathbf{B}_i \mathbf{v}_i + \epsilon$$

where

- \mathbf{A} : covariates, fixed effects β
- \mathbf{B}_i : weights, random effects $\mathbf{v}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$
- ϵ_i : Possible noise

Observations

$$\mathbf{y} \sim \pi(\mathbf{y} | \eta,) = \prod_i \pi(y_i | \eta_i)$$

A closer look at that latent field

$$\eta_{ij} = \mu + c_{ij}\beta + u_i + v_j + w_{ij}, \quad i = 1, \dots, N, \quad j = 1, \dots, M$$

where \mathbf{u} , \mathbf{v} and \mathbf{w} are “random effects”.

If we assign Gaussian priors on μ , β , \mathbf{u} and \mathbf{v} , then

$$\mathbf{x} = (\mu, \beta, \mathbf{u}, \mathbf{v}, \boldsymbol{\eta})$$

is jointly Gaussian.

Reaching for a framework

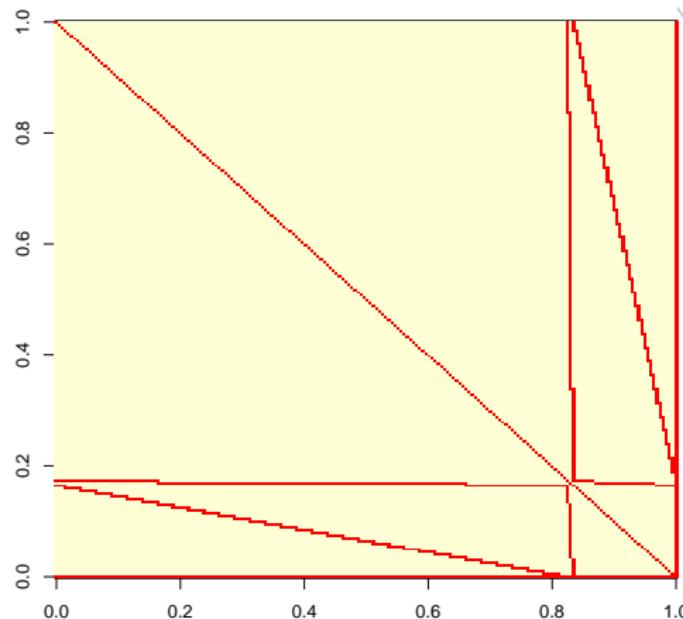
We can reinterpret the model as

$$\begin{aligned}\boldsymbol{\theta} &\sim \pi(\boldsymbol{\theta}) \\ \mathbf{x} | \boldsymbol{\theta} &\sim \pi(\mathbf{x} | \boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\theta})) \\ \mathbf{y} | \mathbf{x}, \boldsymbol{\theta} &\sim \prod_i \pi(y_i | \eta_i, \boldsymbol{\theta})\end{aligned}$$

- $\dim(\mathbf{x})$ could be large 10^2 - 10^5
- $\dim(\boldsymbol{\theta})$ is small 1-5

What does Σ^{-1} look like?

Precision matrix $(\eta, \mathbf{u}, \mathbf{v}, \mu, \beta) N = 100, M = 5.$



Smoothing binary time series

- Data is sequence of 0 and 1s
- Probability for a 1 at time t , p_t , depends on time

$$p_t = \frac{\exp(\eta_t)}{1 + \exp(\eta_t)}$$

- Linear predictor

$$\eta_t = \mu + \beta c_t + u_t + v_t, \quad t = 1, \dots, n$$

Smoothing binary time series

Prior models

- μ and β are Normal
- \boldsymbol{u} AR-model, like

$$u_t = \phi u_{t-1} + \epsilon_t$$

with parameters $(\phi, \sigma_\epsilon^2)$.

- \boldsymbol{v} is an unstructured term or a “random effect”

$$v_i \stackrel{\text{i.i.d.}}{\sim} N(0, \sigma_v^2)$$

Smoothing binary time series

Prior models

- μ and β are Normal
- \mathbf{u} AR-model, like

$$u_t = \phi u_{t-1} + \epsilon_t$$

with parameters $(\phi, \sigma_\epsilon^2)$.

- \mathbf{v} is an unstructured term or a “random effect”

$$v_i \stackrel{\text{i.i.d.}}{\sim} N(0, \sigma_v^2)$$

This gives

$$\mathbf{x} = (\mu, \beta, \mathbf{u}, \mathbf{v}, \eta)$$

is jointly Gaussian.

Hyperparameters

$$\boldsymbol{\theta} = (\phi, \sigma_\epsilon^2, \sigma_v^2)$$

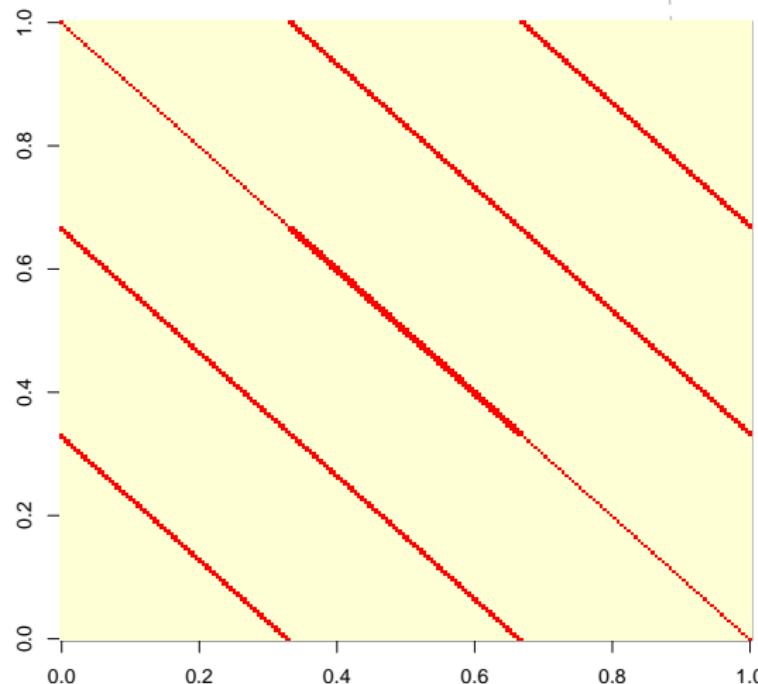
(Still) Reaching for a framework

We can reinterpret the model as

$$\begin{aligned}\boldsymbol{\theta} &\sim \pi(\boldsymbol{\theta}) \\ \mathbf{x} | \boldsymbol{\theta} &\sim \pi(\mathbf{x} | \boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\theta})) \\ \mathbf{y} | \mathbf{x}, \boldsymbol{\theta} &\sim \prod_i \pi(y_i | \eta_i, \boldsymbol{\theta})\end{aligned}$$

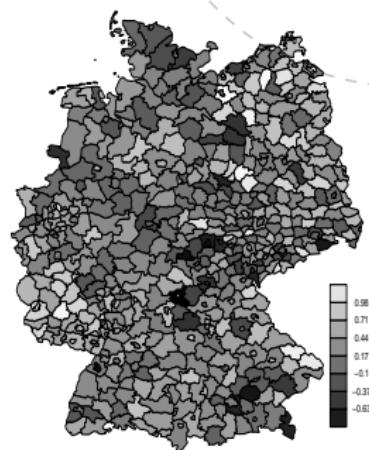
- $\dim(\mathbf{x})$ could be large 10^2 - 10^5
- $\dim(\boldsymbol{\theta})$ is small 1-5

Precision matrix $(\eta, \mathbf{u}, \mathbf{v}, \mu, \beta)$ $N = 100, M = 5$.



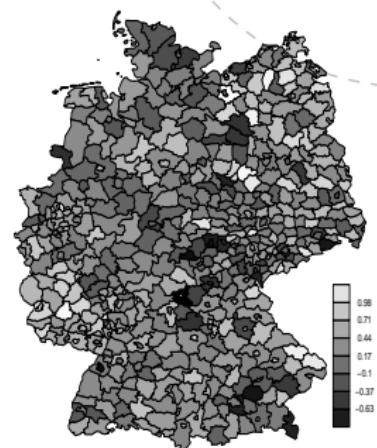
Yet Another Example (I): Disease mapping

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



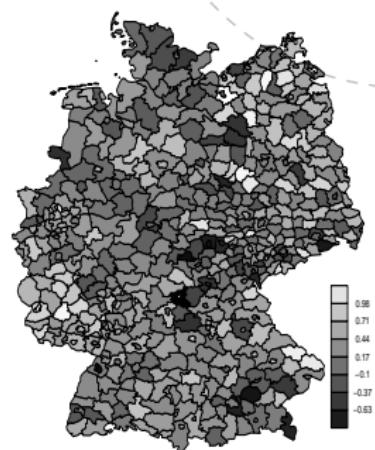
Yet Another Example (I): Disease mapping

- Data $y_i \sim \text{Poisson}(E_i \exp(\eta_i))$
- Log-relative risk
$$\eta_i = \mu + u_i + v_i + f(c_i)$$



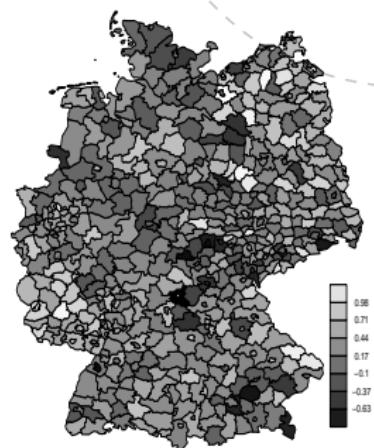
Yet Another Example (I): Disease mapping

- Data $y_i \sim \text{Poisson}(E_i \exp(\eta_i))$
- Log-relative risk
$$\eta_i = \mu + u_i + v_i + f(c_i)$$
- Structured component \boldsymbol{u}



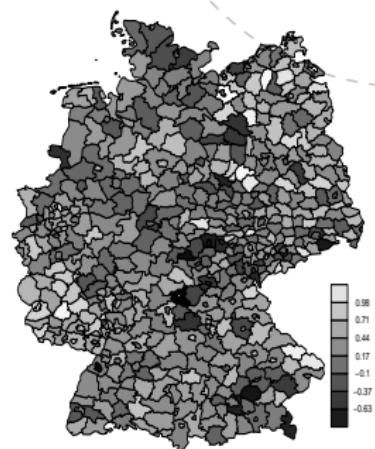
Yet Another Example (I): Disease mapping

- Data $y_i \sim \text{Poisson}(E_i \exp(\eta_i))$
- Log-relative risk
$$\eta_i = \mu + u_i + v_i + f(c_i)$$
- Structured component \boldsymbol{u}
- Unstructured component \boldsymbol{v}

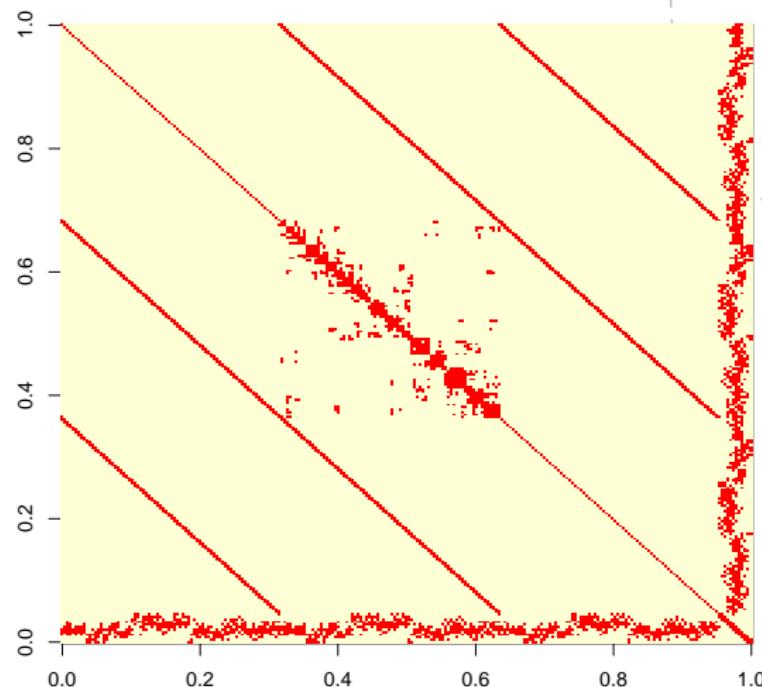


Yet Another Example (I): Disease mapping

- Data $y_i \sim \text{Poisson}(E_i \exp(\eta_i))$
- Log-relative risk
$$\eta_i = \mu + u_i + v_i + f(c_i)$$
- Structured component \mathbf{u}
- Unstructured component \mathbf{v}
- Smooth effect of a covariate \mathbf{c}



Precision matrix ($\eta, \mathbf{u}, \mathbf{v}, \mu, \mathbf{f}$)



We'll just keep reaching for a framework

We can reinterpret the model as

$$\begin{aligned}\boldsymbol{\theta} &\sim \pi(\boldsymbol{\theta}) \\ \mathbf{x} | \boldsymbol{\theta} &\sim \pi(\mathbf{x} | \boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\theta})) \\ \mathbf{y} | \mathbf{x}, \boldsymbol{\theta} &\sim \prod_i \pi(y_i | \eta_i, \boldsymbol{\theta})\end{aligned}$$

- $\dim(\mathbf{x})$ could be large 10^2 - 10^5
- $\dim(\boldsymbol{\theta})$ is small 1-5

A more surprising example: Log-Gaussian Cox processes

Spatial point processes model the spatial organisation of individuals.

- They focus on the random location at which events happen.
- They make excellent models for ‘presence only’ data when coupled with an appropriate observation process.
- Realistic models can be quite complicated.
- Inference is hard—standard MCMC falls in a hole for even simple models.

The likelihood (aka why point processes are hard to infer)

The likelihood *in the most boring case* is

$$\log(\pi(Y|\eta)) = |\Omega| - \int_{\Omega} \Lambda(s) ds + \sum_{s_i \in Y} \Lambda(s_i),$$

where Y is the set of observed locations and $\Lambda(s) = \exp(Z(s))$, and $Z(s)$ is a Gaussian random field.

This is very different to the previous examples!

Or is it?

It's worth thinking about what the ugly ugly likelihood actually means. It says that the number of points in a region R is Poisson distributed with mean $\int_R \Lambda(s) ds$.

Or is it?

It's worth thinking about what the ugly ugly likelihood actually means. It says that the number of points in a region R is Poisson distributed with mean $\int_R \Lambda(s) ds$.

- Divide the ‘observation window’ into N rectangles.
- Let y_i be the number of points in rectangle i . Then

$$y_i | x_i, \theta \sim Po(e^{x_i}),$$

where x_i is a representative values of $Z(s)$ in the i th rectangle.

- The log-risk surface is replaced with \mathbf{x} , which is conditionally Gaussian

$$\mathbf{x} | \theta \sim N(\mu(\theta), \Sigma(\theta)).$$

- The parameters θ are as they always are.

What we have learned so far (I)

This model-construct

$$\begin{aligned}\boldsymbol{\theta} &\sim \pi(\boldsymbol{\theta}) \\ \mathbf{x} | \boldsymbol{\theta} &\sim \pi(\mathbf{x} | \boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\theta})) \\ \mathbf{y} | \mathbf{x}, \boldsymbol{\theta} &\sim \prod_i \pi(y_i | \eta_i, \boldsymbol{\theta})\end{aligned}$$

occurs in many, seemingly unrelated, statistical models.

We call these models *latent Gaussian models!*

Further Examples

- Dynamic linear models

Further Examples

- Dynamic linear models
- Stochastic volatility models (famously difficult with MCMC)

Further Examples

- Dynamic linear models
- Stochastic volatility models (famously difficult with MCMC)
- Generalised linear (mixed) models

Further Examples

- Dynamic linear models
- Stochastic volatility models (famously difficult with MCMC)
- Generalised linear (mixed) models
- Generalised additive (mixed) models

Further Examples

- Dynamic linear models
- Stochastic volatility models (famously difficult with MCMC)
- Generalised linear (mixed) models
- Generalised additive (mixed) models
- Spline smoothing

Further Examples

- Dynamic linear models
- Stochastic volatility models (famously difficult with MCMC)
- Generalised linear (mixed) models
- Generalised additive (mixed) models
- Spline smoothing
- Semiparametric regression

Further Examples

- Dynamic linear models
- Stochastic volatility models (famously difficult with MCMC)
- Generalised linear (mixed) models
- Generalised additive (mixed) models
- Spline smoothing
- Semiparametric regression
- Space-varying (semiparametric) regression models

Further Examples

- Dynamic linear models
- Stochastic volatility models (famously difficult with MCMC)
- Generalised linear (mixed) models
- Generalised additive (mixed) models
- Spline smoothing
- Semiparametric regression
- Space-varying (semiparametric) regression models
- Disease mapping

Further Examples

- Dynamic linear models
- Stochastic volatility models (famously difficult with MCMC)
- Generalised linear (mixed) models
- Generalised additive (mixed) models
- Spline smoothing
- Semiparametric regression
- Space-varying (semiparametric) regression models
- Disease mapping
- Log-Gaussian Cox-processes

Further Examples

- Dynamic linear models
- Stochastic volatility models (famously difficult with MCMC)
- Generalised linear (mixed) models
- Generalised additive (mixed) models
- Spline smoothing
- Semiparametric regression
- Space-varying (semiparametric) regression models
- Disease mapping
- Log-Gaussian Cox-processes
- Model-based geostatistics (*)

Further Examples

- Dynamic linear models
- Stochastic volatility models (famously difficult with MCMC)
- Generalised linear (mixed) models
- Generalised additive (mixed) models
- Spline smoothing
- Semiparametric regression
- Space-varying (semiparametric) regression models
- Disease mapping
- Log-Gaussian Cox-processes
- Model-based geostatistics (*)
- Spatio-temporal models

Further Examples

- Dynamic linear models
- Stochastic volatility models (famously difficult with MCMC)
- Generalised linear (mixed) models
- Generalised additive (mixed) models
- Spline smoothing
- Semiparametric regression
- Space-varying (semiparametric) regression models
- Disease mapping
- Log-Gaussian Cox-processes
- Model-based geostatistics (*)
- Spatio-temporal models
- Survival analysis

Further Examples

- Dynamic linear models
- Stochastic volatility models (famously difficult with MCMC)
- Generalised linear (mixed) models
- Generalised additive (mixed) models
- Spline smoothing
- Semiparametric regression
- Space-varying (semiparametric) regression models
- Disease mapping
- Log-Gaussian Cox-processes
- Model-based geostatistics (*)
- Spatio-temporal models
- Survival analysis
- +++

So how do we make this efficient

In the end, we have a big Gaussian random vector laying around and, as such we need to be careful.

- In order to do efficient computations with large matrices, you need something to be sparse.

So how do we make this efficient

In the end, we have a big Gaussian random vector laying around and, as such we need to be careful.

- In order to do efficient computations with large matrices, you need something to be sparse.
- Option 1: The covariance matrix—this happens sometimes, but enforces independence.

So how do we make this efficient

In the end, we have a big Gaussian random vector laying around and, as such we need to be careful.

- In order to do efficient computations with large matrices, you need something to be sparse.
- Option 1: The covariance matrix—this happens sometimes, but enforces independence.
- Option 2: The precision matrix

$$\mathbf{Q}(\theta) = \boldsymbol{\Sigma}(\theta)^{-1}.$$

This enforces a *conditional independence* structure.

This will be our focus.

A justification: Building models through conditioning

Consider the joint density for (\mathbf{x}, \mathbf{y}) , where

- $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_x^{-1})$
- $\mathbf{y}|\mathbf{x} \sim \mathcal{N}(\mathbf{x}, \mathbf{Q}_y^{-1})$

then

$$\mathbf{Q}_{(\mathbf{x}, \mathbf{y})} = \begin{bmatrix} \mathbf{Q}_x + \mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{Q}_y \end{bmatrix}$$

A justification: Building models through conditioning

Consider the joint density for (\mathbf{x}, \mathbf{y}) , where

- $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_x^{-1})$
- $\mathbf{y}|\mathbf{x} \sim \mathcal{N}(\mathbf{x}, \mathbf{Q}_y^{-1})$

then

$$\mathbf{Q}_{(\mathbf{x}, \mathbf{y})} = \begin{bmatrix} \mathbf{Q}_x + \mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{Q}_y \end{bmatrix}$$

The corresponding expressions using covariance matrices are very ugly!

Conclusion: This is the natural parameterisation for building hierarchical models.

Computational benefits

Models we have seen have

- sparse precision matrix
- sparse Cholesky triangle (coming soon!)

These are much faster to compute than dense matrices

Special cases

Computational benefits

Models we have seen have

- sparse precision matrix
- sparse Cholesky triangle (coming soon!)

These are much faster to compute than dense matrices

Special cases

- Algorithms for ARMA-models in time-series

Computational benefits

Models we have seen have

- sparse precision matrix
- sparse Cholesky triangle (coming soon!)

These are much faster to compute than dense matrices

Special cases

- Algorithms for ARMA-models in time-series
- Kalman-filter algorithms

Numerical algorithms for sparse matrices: scaling properties

— Time: $\mathcal{O}(n)$

This is to be compared with general $\mathcal{O}(n^3)$ algorithms for general dense matrices.

Numerical algorithms for sparse matrices: scaling properties

- Time: $\mathcal{O}(n)$
- Space: $\mathcal{O}(n^{3/2})$

This is to be compared with general $\mathcal{O}(n^3)$ algorithms for general dense matrices.

Numerical algorithms for sparse matrices: scaling properties

- Time: $\mathcal{O}(n)$
- Space: $\mathcal{O}(n^{3/2})$
- Space-time: $\mathcal{O}(n^2)$

This is to be compared with general $\mathcal{O}(n^3)$ algorithms for general dense matrices.

Gaussian Markov random fields

- Gaussians with a sparse precision matrix are called *Gaussian Markov random fields* (GMRFs)

Gaussian Markov random fields

- Gaussians with a sparse precision matrix are called *Gaussian Markov random fields* (GMRFs)
- Good computational properties through numerical algorithms for sparse matrices

Gaussian Markov random fields

- Gaussians with a sparse precision matrix are called *Gaussian Markov random fields* (GMRFs)
- Good computational properties through numerical algorithms for sparse matrices
- Very useful for doing MCMC-based inference as well (BayesX, for example)

Summary

Three main ingredients in INLA

- Gaussian Markov random fields (Next section)
- Latent Gaussian models
- Laplace approximations (Coming attraction!)

which together (and +++++...) gives a very very nice tool for Bayesian inference

- quick
- accurate (relative error)
- good scaling properties
- +++

Spatial survival: example

Leukaemia survival data (Henderson et al, 2002, JASA), 1043 cases.



Fig. 1. Leukaemia survival data: districts of Northwest England and locations of the observations.

Spatial survival: example

$\log(\text{hazard}) = \log(\text{baseline})$
+ $f(\text{age})$
+ $f(\text{white blood cell count})$
+ $f(\text{deprivation index})$
+ $f(\text{spatial})$
+ sex



Fig. 1. Leukaemia survival data: districts of Northwest England and locations of the observed

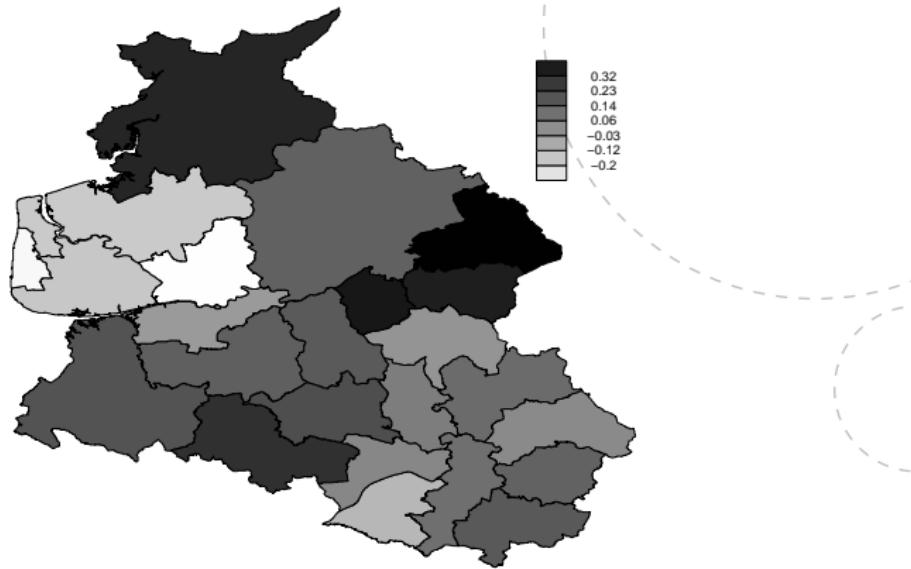
R-code

```
data(Leuk)
g = system.file("demodata/Leuk.graph", package="INLA")

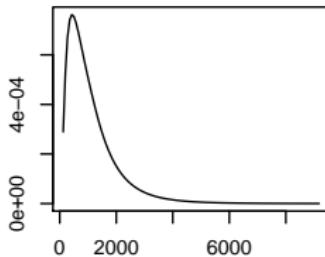
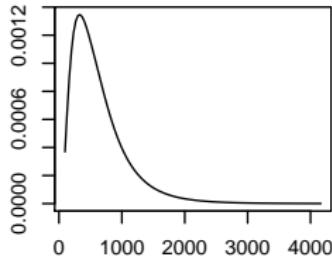
formula = inla.surv(Leuk$time, Leuk$cens) ~ sex + age +
  f(inla.group(wbc), model="rw1")+
  f(inla.group(tpi), model="rw2")+
  f(district, model="besag", graph.file = g)

result = inla(formula, family="coxph", data=Leuk)

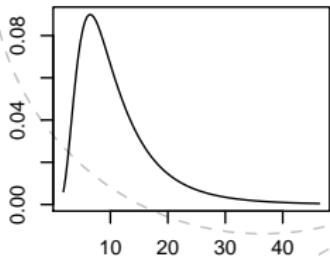
source(system.file("demodata/Leuk-map.R", package="INLA"))
Leuk.map(result$summary.random$district$mean)
plot(result)
```



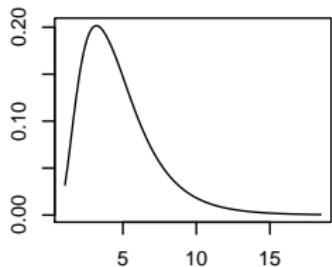
PostDens [Precision for inla.group(vPostDens [Precision for inla.group(



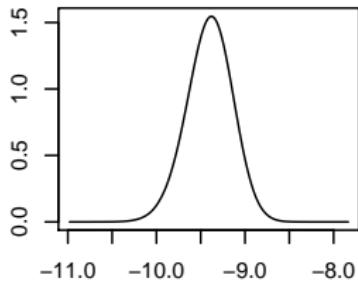
PostDens [Precision for distribution]



PostDens [Precision for baseline.ha:

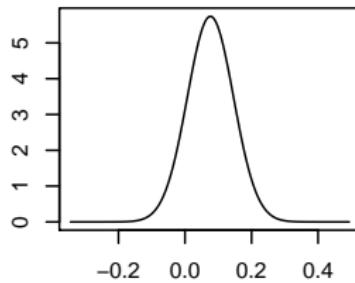


PostDens [(Intercept)]



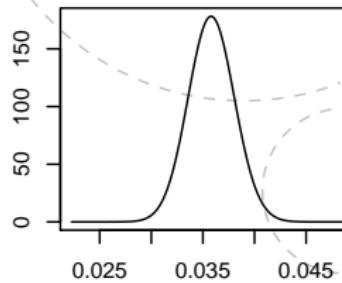
Mean = -9.401 SD = 0.261

PostDens [sex]



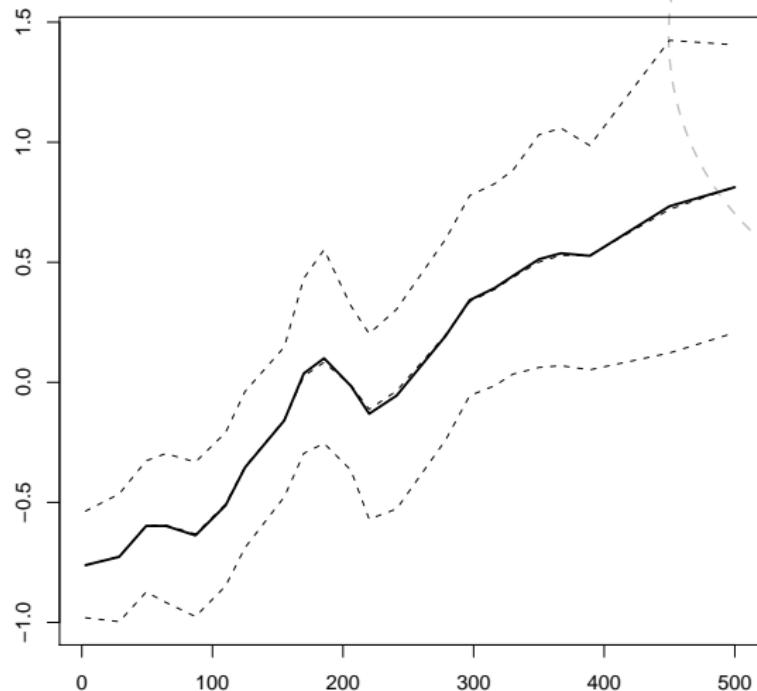
Mean = 0.077 SD = 0.069

PostDens [age]



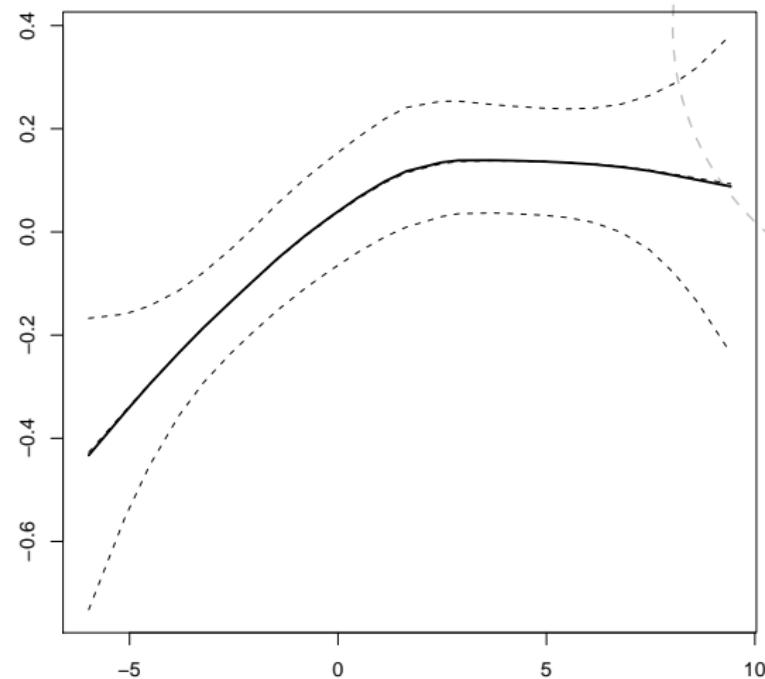
Mean = 0.036 SD = 0.002

inla.group(wbc)



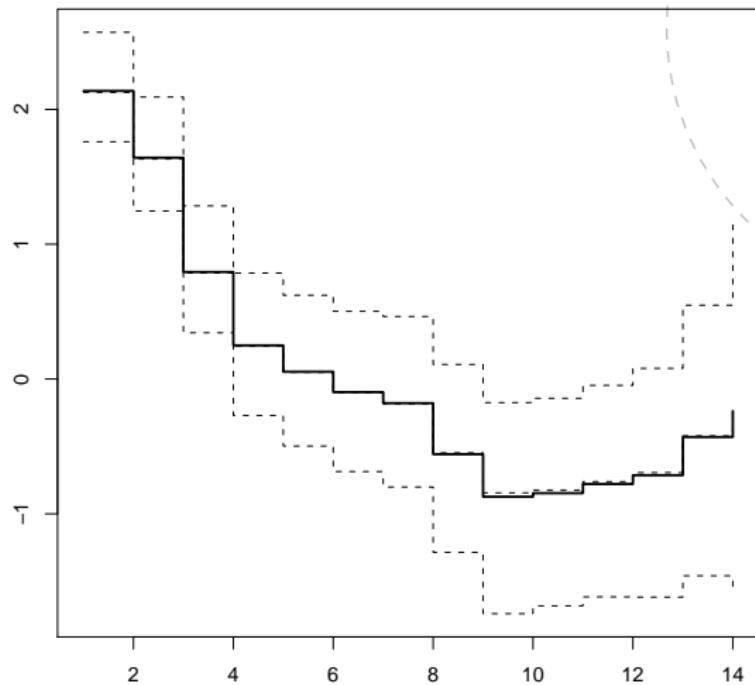
PostMean 0.025% 0.5% 0.975%

inla.group(tpi)



PostMean 0.025% 0.5% 0.975%

baseline.hazard



PostMean 0.025% 0.5% 0.975%

Some internal statistics

Running time on my laptop: 12 seconds

- Factorise \mathbf{Q} (dim = 2453): 518 times
- Solve $\mathbf{Q}\mathbf{x} = \mathbf{b}$: 4 500 times

Outline

Introduction

Latent Gaussian Models

Gaussian Markov random fields

Gaussian Markov random fields

Computing with GMRFs

A case for approximate inference

INLA

Intrinsic GMRFs

Outline

Introduction

Latent Gaussian Models

Gaussian Markov random fields

Gaussian Markov random fields

What is a GMRF?

The precision matrix

Example: Auto-regressive process

Definition of a GMRF

Main features of GMRFs

Properties of GMRFs

Interpretation of elements of \mathbf{Q}

Markov properties

What is a Gaussian Markov random field (GMRF)?

A GMRF is a simple construct

- A normal distributed random vector

$$\mathbf{x} = (x_1, \dots, x_n)^T$$

- Additional Markov properties:

$$x_i \perp x_j \mid \mathbf{x}_{-ij}$$

x_i and x_j are conditional independent (CI).

If $x_i \perp x_j \mid \mathbf{x}_{-ij}$ for a set of $\{i, j\}$, then we need to constrain the parametrisation of the GMRF.

- Covariance matrix: difficult
- Precision matrix: easy

Conditional independence and the precision matrix

The density of a zero mean Gaussian

$$\pi(\mathbf{x}) \propto |\mathbf{Q}|^{1/2} \exp\left(-\frac{1}{2}\mathbf{x}^T \mathbf{Q} \mathbf{x}\right)$$

Constraining the parametrisation to obey CI properties

Theorem

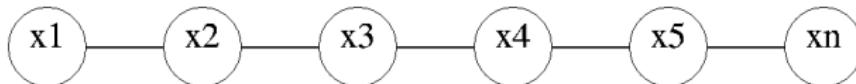
$$x_i \perp x_j \mid \mathbf{x}_{-ij} \iff Q_{ij} = 0$$

Simple example of a GMRF

Auto-regressive process of order 1

$$x_t \mid x_{t-1}, \dots, x_1 \sim \mathcal{N}(\phi x_{t-1}, 1), \quad t = 2, \dots, n$$

and $x_1 \sim \mathcal{N}(0, (1 - \phi^2)^{-1})$.

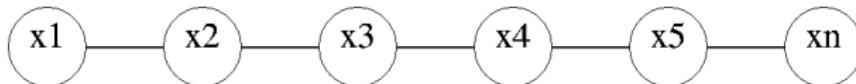


Simple example of a GMRF

Auto-regressive process of order 1

$$x_t \mid x_{t-1}, \dots, x_1 \sim \mathcal{N}(\phi x_{t-1}, 1), \quad t = 2, \dots, n$$

and $x_1 \sim \mathcal{N}(0, (1 - \phi^2)^{-1})$.



Tridiagonal precision matrix

$$\mathbf{Q} = \begin{pmatrix} 1 & -\phi & & & & \\ -\phi & 1 + \phi^2 & -\phi & & & \\ & \ddots & \ddots & \ddots & & \\ & & -\phi & 1 + \phi^2 & -\phi & \\ & & & -\phi & 1 & \end{pmatrix}$$

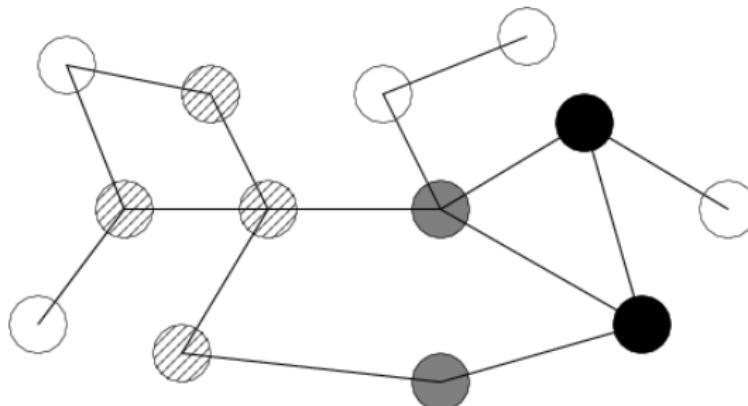
The Markov property on a Graph

Let \mathbf{x} be a GMRF wrt $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

The global Markov property:

$$\mathbf{x}_A \perp \mathbf{x}_B \mid \mathbf{x}_C$$

for all disjoint sets A , B and C where C separates A and B , and A and B are non-empty.



Use a (undirected) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to represent the CI properties,

\mathcal{V} Vertices: $1, 2, \dots, n$.

\mathcal{E} Edges $\{i, j\}$

- No edge between i and j if $x_i \perp x_j \mid \mathbf{x}_{-ij}$.
- An edge between i and j if $x_i \not\perp x_j \mid \mathbf{x}_{-ij}$.

Key point: A graph defines the sparsity structure of \mathbf{Q} !

Definition of a GMRF

Definition (GMRF)

A random vector $\mathbf{x} = (x_1, \dots, x_n)^T$ is called a GMRF wrt the graph $\mathcal{G} = (\mathcal{V} = \{1, \dots, n\}, \mathcal{E})$ with mean μ and precision matrix $\mathbf{Q} > 0$, iff its density has the form

$$\mathbf{x} \sim N(\mu, \mathbf{Q}^{-1})$$

and

$$Q_{ij} \neq 0 \iff \{i, j\} \in \mathcal{E} \quad \text{for all } i \neq j.$$

Main features

- Analytical tractable
- Modelling using conditional independence
- Merging GMRFs using conditioning (hierarchical models)
- **Unified framework** for
 - understanding
 - representation
 - computation using numerical methods for sparse matrices

Interpretation of elements of \mathbf{Q}

Let \mathbf{x} be a GMRF wrt $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with mean $\boldsymbol{\mu}$ and precision matrix $\mathbf{Q} > 0$, then

$$\mathbb{E}(x_i | \mathbf{x}_{-i}) = \mu_i - \frac{1}{Q_{ii}} \sum_{j:j \sim i} Q_{ij}(x_j - \mu_j),$$

$$\text{Prec}(x_i | \mathbf{x}_{-i}) = Q_{ii} \quad \text{and}$$

$$\text{Corr}(x_i, x_j | \mathbf{x}_{-ij}) = -\frac{Q_{ij}}{\sqrt{Q_{ii}Q_{jj}}}, \quad i \neq j.$$

Outline

Introduction

Latent Gaussian Models

Gaussian Markov random fields

Gaussian Markov random fields

Computing with GMRFs

Simulation algorithms for GMRFs

Basic numerical linear algebra

Cholesky factorisation and the Cholesky triangle

Solving linear equations

Avoid computing the inverse

Unconditional sampling

Simulation algorithm

Example: auto-regressive processes

Cholesky factorisation

If $\mathbf{A} > 0$ be a $n \times n$ positive definite matrix, then there exists a unique Cholesky triangle \mathbf{L} , such that \mathbf{L} is a lower triangular matrix, and

$$\mathbf{A} = \mathbf{LL}^T$$

Computing \mathbf{L} costs $n^3/3$ flops.

This factorisation is the basis for *solving* systems like

$$\mathbf{Ax} = \mathbf{b} \quad \text{or} \quad \mathbf{AX} = \mathbf{B}$$

for k right hand sides, or equivalently, computing

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \quad \text{or} \quad \mathbf{X} = \mathbf{A}^{-1}\mathbf{B}$$

Solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ where $\mathbf{A} > 0$

- 1: Compute the Cholesky factorisation, $\mathbf{A} = \mathbf{L}\mathbf{L}^T$
- 2: Solve $\mathbf{L}\mathbf{v} = \mathbf{b}$
- 3: Solve $\mathbf{L}^T\mathbf{x} = \mathbf{v}$
- 4: **Return \mathbf{x}**

Solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ where $\mathbf{A} > 0$

- 1: Compute the Cholesky factorisation, $\mathbf{A} = \mathbf{L}\mathbf{L}^T$
- 2: Solve $\mathbf{L}\mathbf{v} = \mathbf{b}$
- 3: Solve $\mathbf{L}^T\mathbf{x} = \mathbf{v}$
- 4: **Return \mathbf{x}**

Step 2 is called *forward-substitution* and cost $\mathcal{O}(n^2)$ flops.



The solution \mathbf{v} is computed in a forward-loop

$$v_i = \frac{1}{L_{ii}}(b_i - \sum_{j=1}^{i-1} L_{ij}v_j), \quad i = 1, \dots, n \tag{1}$$

Solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ where $\mathbf{A} > 0$

- 1: Compute the Cholesky factorisation, $\mathbf{A} = \mathbf{L}\mathbf{L}^T$
- 2: Solve $\mathbf{L}\mathbf{v} = \mathbf{b}$
- 3: Solve $\mathbf{L}^T\mathbf{x} = \mathbf{v}$
- 4: **Return \mathbf{x}**

Step 3 is called *back-substitution* and costs $\mathcal{O}(n^2)$ flops.



The solution \mathbf{x} is computed in a backward-loop

$$x_i = \frac{1}{L_{ii}}(v_i - \sum_{j=i+1}^n L_{ji}x_j), \quad i = n, \dots, 1 \tag{2}$$

To compute $\mathbf{A}^{-1}\mathbf{B}$ where \mathbf{B} is a $n \times k$ matrix, we do this by computing the solution \mathbf{X} of

$$\mathbf{AX}_j = \mathbf{B}_j$$

for each of the k columns of \mathbf{X} .

Sample $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$

If $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$ and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, then \mathbf{x} defined by

$$\mathbf{L}^T \mathbf{x} = \mathbf{z}$$

has covariance

$$\text{Cov}(\mathbf{x}) = \text{Cov}(\mathbf{L}^{-T} \mathbf{z}) = (\mathbf{L}\mathbf{L}^T)^{-1} = \mathbf{Q}^{-1}$$

Sample $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$

If $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$ and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, then \mathbf{x} defined by

$$\mathbf{L}^T \mathbf{x} = \mathbf{z}$$

has covariance

$$\text{Cov}(\mathbf{x}) = \text{Cov}(\mathbf{L}^{-T} \mathbf{z}) = (\mathbf{L}\mathbf{L}^T)^{-1} = \mathbf{Q}^{-1}$$

Sampling $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$

- 1: Compute the Cholesky factorisation, $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$
- 2: Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 3: Solve $\mathbf{L}^T \mathbf{v} = \mathbf{z}$
- 4: Compute $\mathbf{x} = \boldsymbol{\mu} + \mathbf{v}$
- 5: **Return \mathbf{x}**

Numerical methods for sparse matrices

Computations on GMRFs can be expressed such that the main tasks are

1. compute the Cholesky factorisation of $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$, and
2. solve $\mathbf{L}\mathbf{v} = \mathbf{b}$ and $\mathbf{L}^T\mathbf{x} = \mathbf{z}$.

The second task is much-faster than the first, but sparsity will be of advantage also here.

The goal is to explain

- **why** a sparse \mathbf{Q} allow for fast factorisation
- **how** we can take advantage of it,
- **why** we gain if we permute the vertices before factorising the matrix,
- **how** statisticians can benefit for recent research in this area by the numerical mathematicians.

Statisticians and numerical methods for sparse matrices

Gupta (2002) summarises his findings on recent advances for sparse linear solvers:

... recent sparse solvers have significantly improved the state of the art of the direct solution of general sparse systems.

... recent years have seen some remarkable advances in the general sparse direct-solver algorithms and software.

- Good news
- We just use their software

Example: AR(1)-process (c.f. Kalman Filter)

$$x_t \mid \mathbf{x}_{1:(t-1)} \sim \mathcal{N}(\phi x_{t-1}, \sigma^2), \quad t = 1, \dots, n$$

Example: AR(1)-process (c.f. Kalman Filter)

$$x_t \mid \mathbf{x}_{1:(t-1)} \sim \mathcal{N}(\phi x_{t-1}, \sigma^2), \quad t = 1, \dots, n$$

$$\mathbf{Q} = \begin{pmatrix} \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}$$

Example: AR(1)-process (c.f. Kalman Filter)

$$x_t \mid \mathbf{x}_{1:(t-1)} \sim \mathcal{N}(\phi x_{t-1}, \sigma^2), \quad t = 1, \dots, n$$

$$\mathbf{Q} = \begin{pmatrix} \times & \times \\ \times & \times & \times \\ \times & \times & \times & \times \end{pmatrix} \quad \mathbf{L} = \begin{pmatrix} \times \\ \times & \times \\ \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{pmatrix}$$

Observation: Bandwidth is preserved

Similarly, for an AR(p)-process

- \mathbf{Q} have bandwidth p .
- \mathbf{L} have lower-bandwidth p .

Observation: Bandwidth is preserved

Similarly, for an AR(p)-process

- \mathbf{Q} have bandwidth p .
- \mathbf{L} have lower-bandwidth p .



Observation: Bandwidth is preserved

Similarly, for an AR(p)-process

- \mathbf{Q} have bandwidth p .
- \mathbf{L} have lower-bandwidth p .



We should exploit this ...easy to modify existing Cholesky-factorisation code to use only entries where $|i - j| \leq p$.

Can we exploit this for general sparse matrices?

We can permute the vertexes!

select one of the $n!$ possible permutations, define the corresponding permutation matrix \mathbf{P} , such that $\mathbf{i}^P = \mathbf{Pi}$, where $\mathbf{i} = (1, \dots, n)^T$, is the new ordering of the vertexes.

Chose \mathbf{P} , if possible, such that

$$\mathbf{Q}^P = \mathbf{PQP}^T \tag{3}$$

is a band-matrix with a small bandwidth.

Reorderings in practice

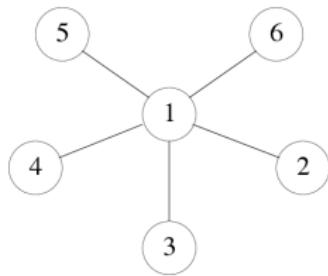
- Impossible in general to obtain the optimal permutation, $n!$ is too large!
- A sub-optimal ordering will do as well.
- Solve $\mathbf{Q}\boldsymbol{\mu} = \mathbf{b}$ as follows:
 - $\mathbf{b}^P = \mathbf{P}\mathbf{b}$.
 - Solve $\mathbf{Q}^P \boldsymbol{\mu}^P = \mathbf{b}^P$
 - Map the solution back, $\boldsymbol{\mu} = \mathbf{P}^T \boldsymbol{\mu}^P$.

Better reorderings

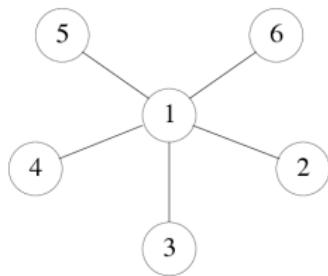
The idea behind a minimal bandwidth reordering is that we can guarantee that there will be no new fill in outside the band width.

Idea: Just find an ordering that minimises the fill-in!

More optimal reordering schemes

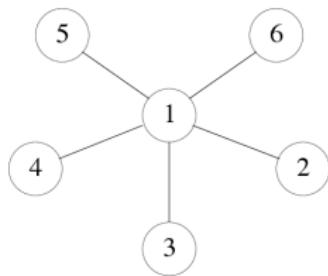


More optimal reordering schemes



$$\begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & & & & \\ \times & & \times & & & \\ \times & & & \times & & \\ \times & & & & \times & \\ \times & & & & & \times \end{pmatrix}$$

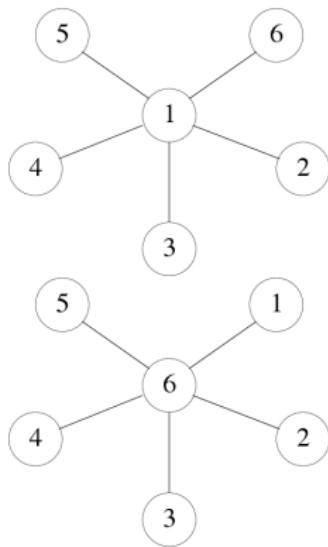
More optimal reordering schemes



$$\begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & & & & \\ \times & & \times & & & \\ \times & & & \times & & \\ \times & & & & \times & \\ \times & & & & & \times \end{pmatrix} \quad \begin{pmatrix} \times & & \times & & & \\ \times & & \checkmark & & & \\ \times & \checkmark & & \times & & \\ \times & \checkmark & \checkmark & \times & & \\ \times & \checkmark & \checkmark & \checkmark & \times & \\ \times & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark \end{pmatrix}$$

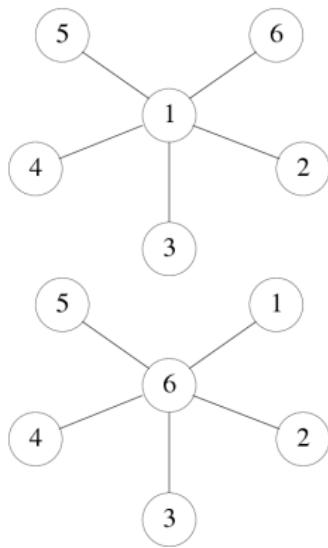
The first matrix represents the original adjacency matrix of the graph. The second matrix represents a reordered version where node 1 is at index 1, node 2 is at index 5, node 3 is at index 4, node 4 is at index 3, node 5 is at index 2, and node 6 is at index 6. The reordered matrix shows that edges between nodes 1 and 2, 1 and 3, 1 and 4, 1 and 5, and 1 and 6 are preserved, while the edge between 2 and 3 is removed.

More optimal reordering schemes



$$\begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & & & & \\ \times & & \times & & & \\ \times & & & \times & & \\ \times & & & & \times & \\ \times & & & & & \times \end{pmatrix} \quad \begin{pmatrix} \times & & \times & & & \\ \times & & \checkmark & & & \\ \times & \checkmark & & \times & & \\ \times & \checkmark & & \checkmark & & \times \\ \times & \checkmark & & \checkmark & & \checkmark \\ \times & \checkmark & & \checkmark & & \checkmark \end{pmatrix}$$

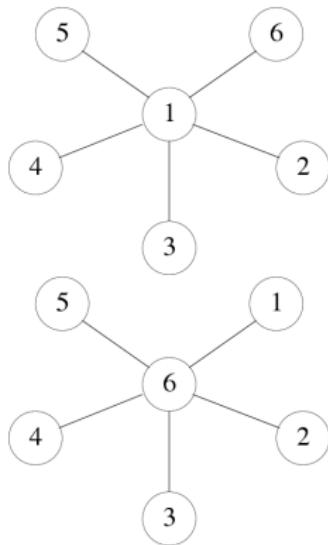
More optimal reordering schemes



$$\begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & & & & \\ \times & & \times & & & \\ \times & & & \times & & \\ \times & & & & \times & \\ \times & & & & & \times \end{pmatrix}
 \begin{pmatrix} \times & & \times & & & \\ \times & \times & & & & \\ \times & & \checkmark & & \times & \\ \times & & \checkmark & \checkmark & \times & \\ \times & & \checkmark & \checkmark & \checkmark & \times \\ \times & & \checkmark & \checkmark & \checkmark & \checkmark \end{pmatrix}$$

$$\begin{pmatrix} \times & & & & & \times \\ & \times & & & & \times \\ & & \times & & & \times \\ & & & \times & & \times \\ & & & & \times & \times \\ & & & & & \times \end{pmatrix}$$

More optimal reordering schemes



$$\begin{array}{c}
 \left(\begin{array}{cccccc} \times & \times & \times & \times & \times & \times \\ \times & \times & & & & \\ \times & & \times & & & \\ \times & & & \times & & \\ \times & & & & \times & \\ \times & & & & & \times \end{array} \right) \quad \left(\begin{array}{cccccc} \times & & & & & \\ \times & \times & & & & \\ \times & & \checkmark & & & \\ \times & & & \checkmark & & \\ \times & & & & \checkmark & \\ \times & & & & & \checkmark \end{array} \right) \\
 \\[10pt]
 \left(\begin{array}{cccccc} \times & & & & & \times \\ & \times & & & & \times \\ & & \times & & & \times \\ & & & \times & & \times \\ & & & & \times & \times \\ & & & & & \times \end{array} \right) \quad \left(\begin{array}{cccccc} \times & & & & & \\ & \times & & & & \\ & & \times & & & \\ & & & \times & & \\ & & & & \times & \\ & & & & & \times \end{array} \right)
 \end{array}$$

Nested dissection reordering

The idea generalise as follows.

- Select a (small) set of nodes whose removal divides the graph into two disconnected subgraphs of almost equal size.
- Order the nodes chosen *after* ordering all the nodes in both subgraphs.
- Apply this procedure recursively to the nodes in each subgraph.

Nested dissection reordering

The idea generalise as follows.

- Select a (small) set of nodes whose removal divides the graph into two disconnected subgraphs of almost equal size.
- Order the nodes chosen *after* ordering all the nodes in both subgraphs.
- Apply this procedure recursively to the nodes in each subgraph.

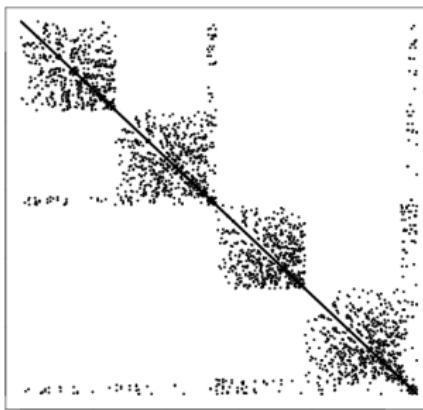
Costs in the spatial case

- Factorisation $\mathcal{O}(n^{3/2})$
- Fill-in $\mathcal{O}(n \log n)$
- Optimal in the order sense.

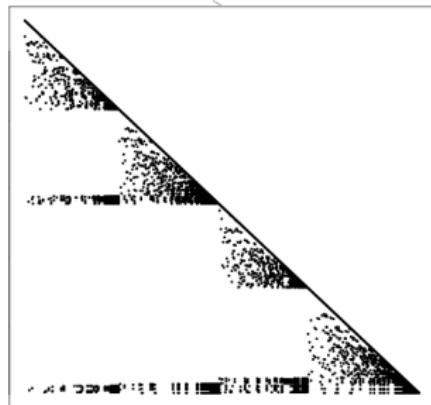
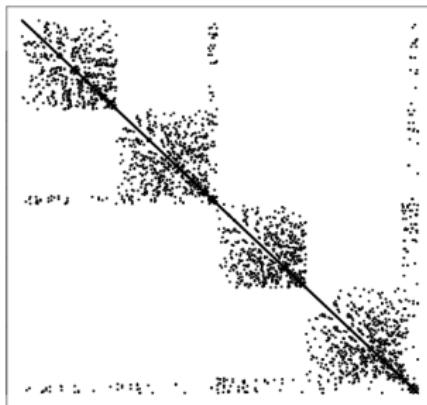
Nested dissection reordering



Nested dissection reordering



Nested dissection reordering



Outline

Introduction

Latent Gaussian Models

Gaussian Markov random fields

Gaussian Markov random fields

Computing with GMRFs

A case for approximate inference

MCMC – single-site

MCMC – blocking

MCMC – blocking scheme I

MCMC – blocking scheme II

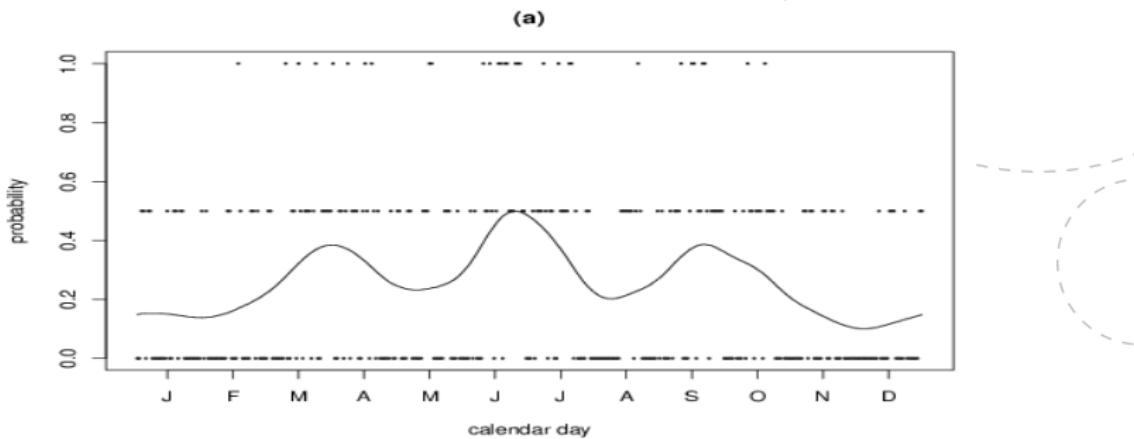
MCMC – blocking without auxiliary variables

MCMC – independence sampler

An MCMC case-study

- Study a seemingly trivial hierarchical model
 - Latent temporal Gaussian, with
 - Binary observations
- Develop a “standard” MCMC-algorithm for inference
 - Auxiliary variables
 - (Conjugate) single-site updates
- ..and study empirically its properties.

Tokyo rainfall data

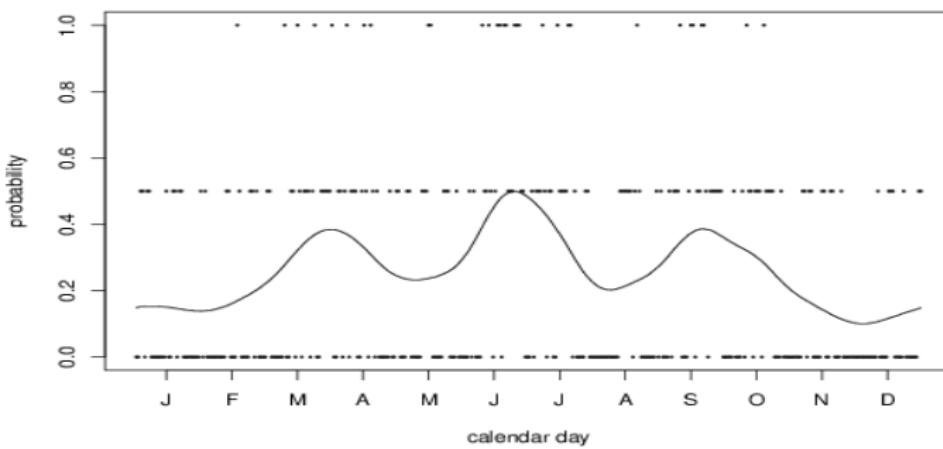


Stage 1 Binomial data

$$y_i \sim \begin{cases} \text{Binomial}(2, p(x_i)) \\ \text{Binomial}(1, p(x_i)) \end{cases}$$

Tokyo rainfall data

(a)

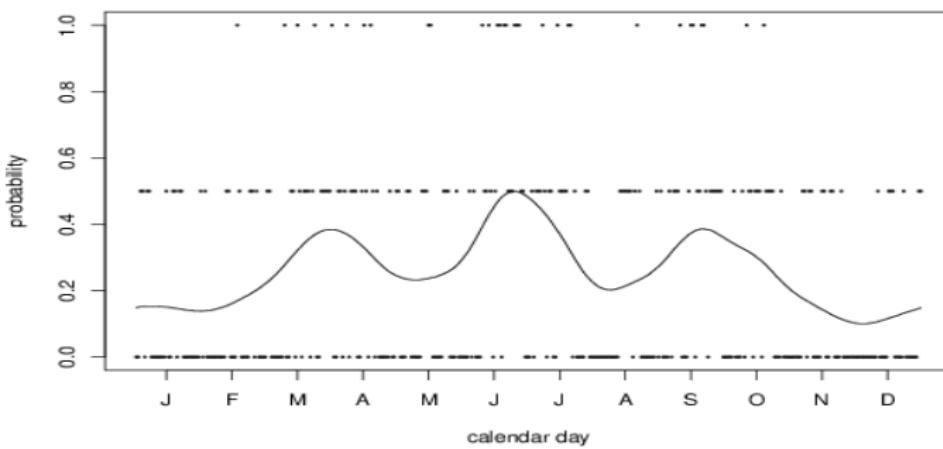


Stage 2 Assume a smooth latent \mathbf{x} ,

$$\mathbf{x} \sim RW2(\kappa), \quad \text{logit}(p_i) = x_i$$

Tokyo rainfall data

(a)



Stage 3 Gamma(α, β)-prior on κ

Model summary

$$\pi(\mathbf{x} \mid \kappa) \pi(\kappa) \prod_i \pi(y_i \mid x_i)$$

where

- $\mathbf{x} \mid \kappa$ is Gaussian (Markov) with dimension 366
- κ is Gamma
- $y_i|x_i$ is Binomial with $p(x_i)$

Construction of nice full conditionals

A popular approach is to introduce auxiliary variables w , so that

$x \mid \text{the rest}$

is Gaussian.

Normal-mixture representation

Theorem (Kerker, 1971)

If x has density $f(x)$ symmetric around 0, then there exist independent random variables z and v , with z standard normal such that $x = z/v$ iff the derivatives of $f(x)$ satisfy

$$\left(-\frac{d}{dy}\right)^k f(\sqrt{y}) \geq 0$$

for $y > 0$ and for $k = 1, 2, \dots$

- Student- t
- Logistic and Laplace

Corresponding mixing distribution for the precision parameter λ that generates these distributions as scale mixtures of normals.

Distribution of x	Mixing distribution of λ
Student- t_ν	$\mathcal{G}(\nu/2, \nu/2)$
Logistic	$1/(2K)^2$ where K is Kolmogorov-Smirnov distributed
Laplace	$1/(2E)$ where E is exponential distributed

Example: Binary regression

GMRF \mathbf{x} and Bernoulli data

$$\begin{aligned} y_i &\sim \mathcal{B}(g^{-1}(x_i)) \\ g(p) &= \Phi(p) \quad \text{probit link} \end{aligned}$$

Equivalent representation using auxiliary variables \mathbf{w}

$$\begin{aligned} \epsilon_i &\stackrel{\text{iid}}{\sim} \mathcal{N}(0, 1) \\ w_i &= x_i + \epsilon_i \\ y_i &= \begin{cases} 1 & \text{if } w_i > 0 \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

for the probit-link.

Single-site Gibbs sampling

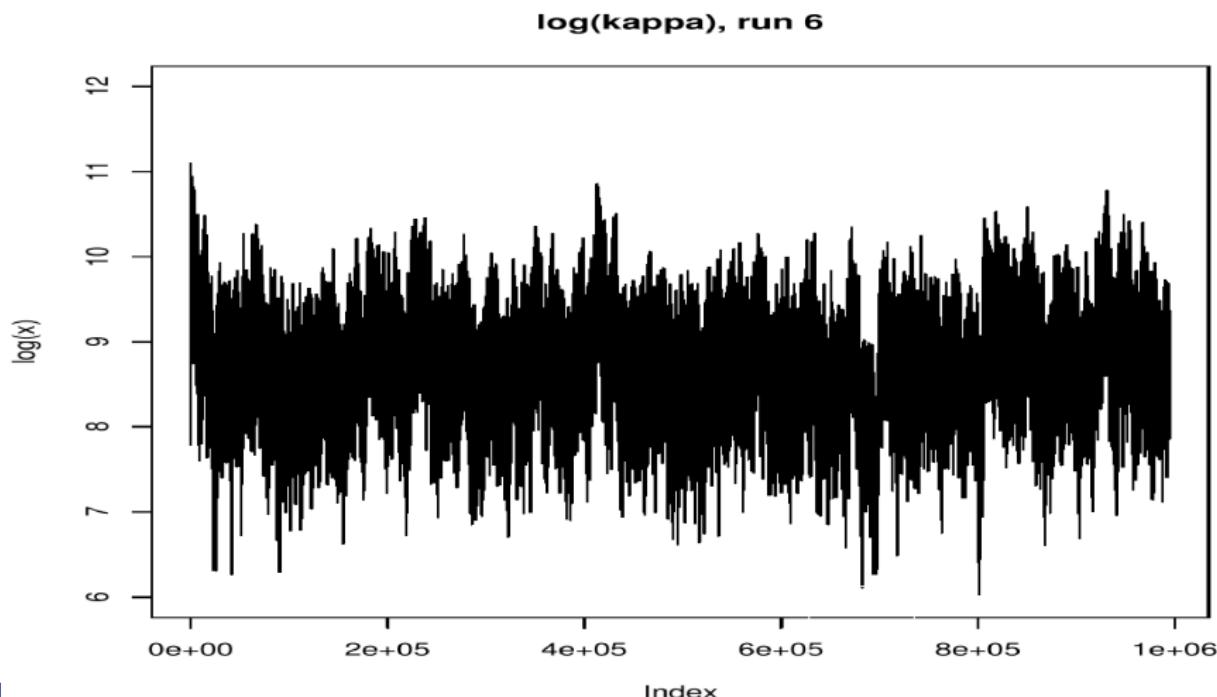
Auxiliary variables can be introduced for the logit-link¹, to achieve this sampler:

- $\kappa \sim \Gamma(\cdot, \cdot)$
- for each i
 - $x_i \sim \mathcal{N}(\cdot, \cdot)$
- for each i
 - $w_i \sim \mathcal{W}(\cdot)$

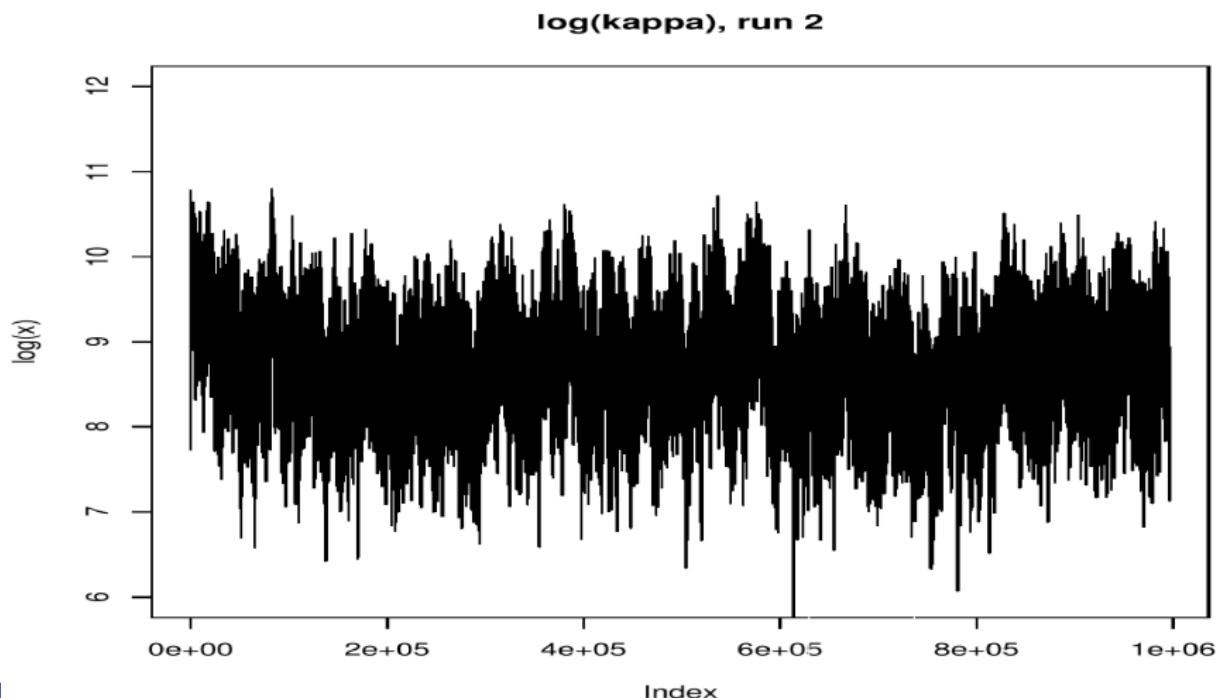
It is fully automatic; no tuning!!!

¹Held & Holmes (2006)

Results: hyper-parameter $\log(\kappa)$

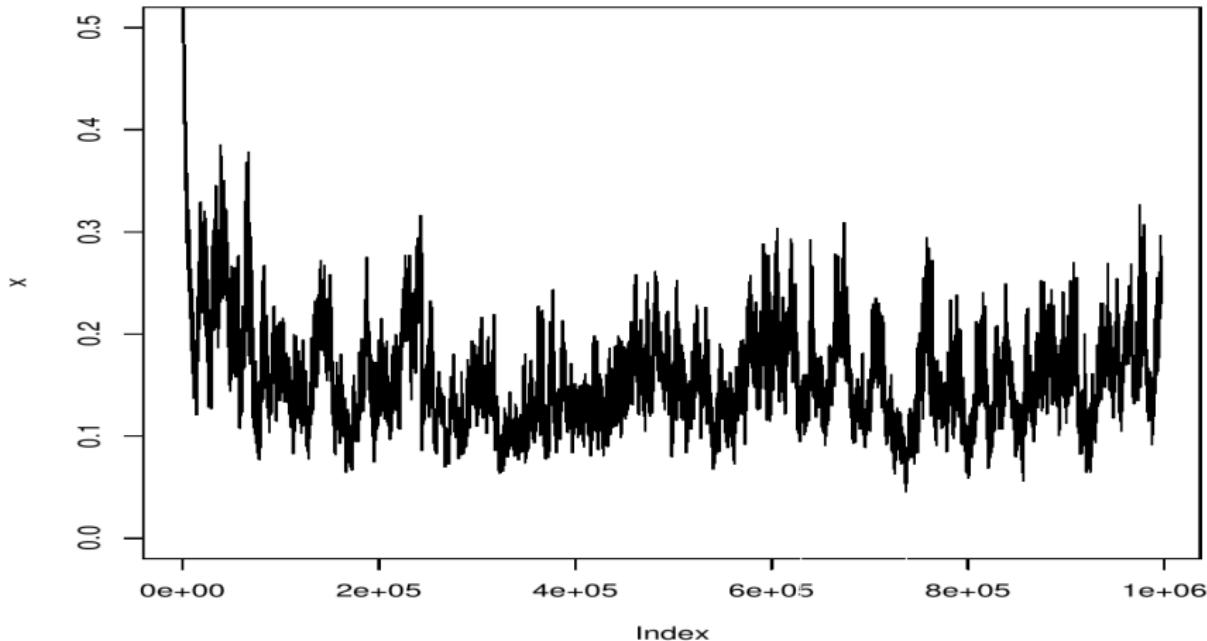


Results: hyper-parameter $\log(\kappa)$



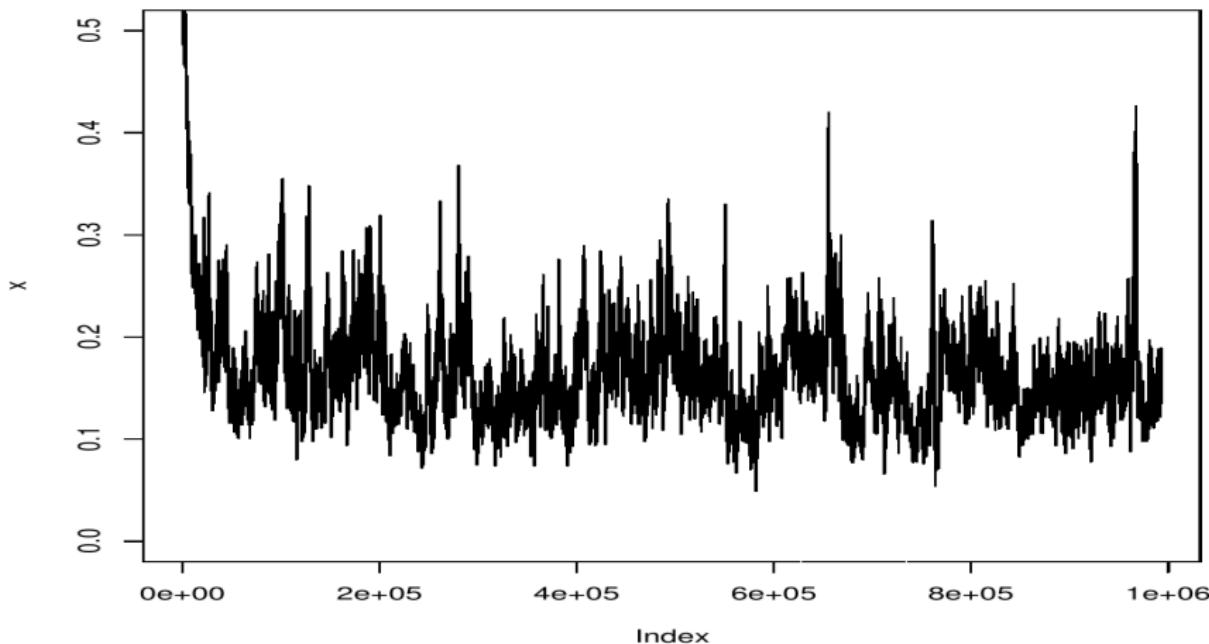
Results: latent node x_{10}

x[10], run 5

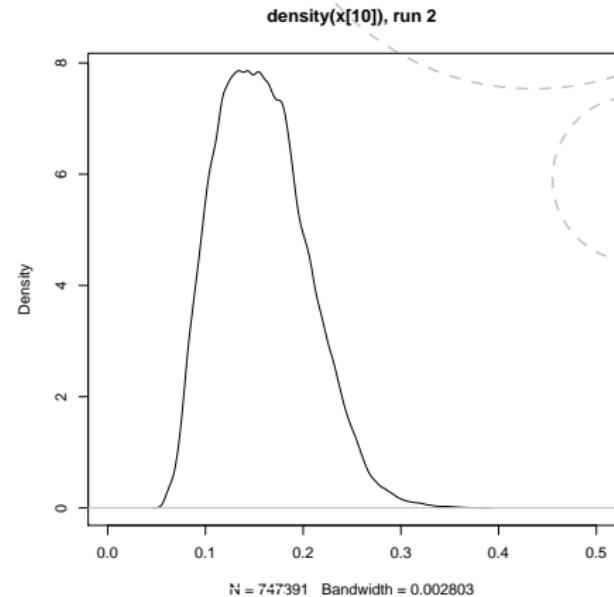
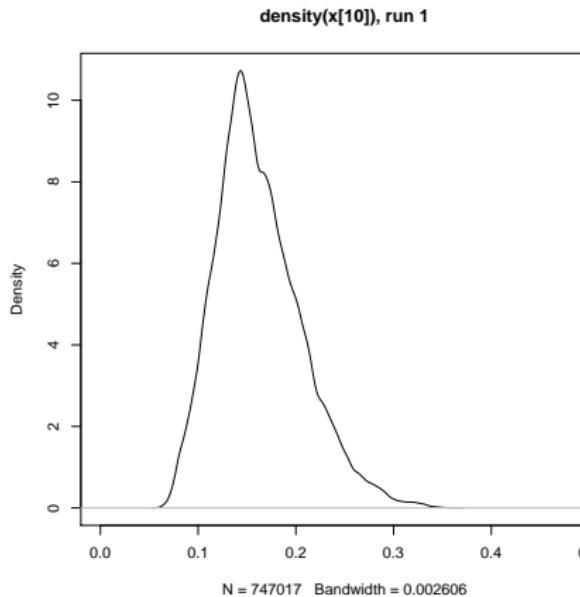


Results: latent node x_{10}

x[10], run 3



Results: density for latent node x_{10}



Discussion

Single-site sampler with auxiliary variables:

- Even *long runs* shows large variation
- “Long” range dependence
- *Very* slowly mixing

But:

- Easy to be “fooled” running shorter chains
- The variability can be underestimated.

What is causing the problem?

Two issues

1. Slow mixing within the latent field \mathbf{x}
2. Slow mixing between the latent field \mathbf{x} and θ .

Blocking is the “usual” approach to resolve such issues, if possible.

Note: blocking mainly helps within the block only.

Strategies for blocking

Slow mixing due to the latent field \mathbf{x} only:

- Block \mathbf{x}

Slow mixing due to the interaction between the latent field \mathbf{x} and θ :

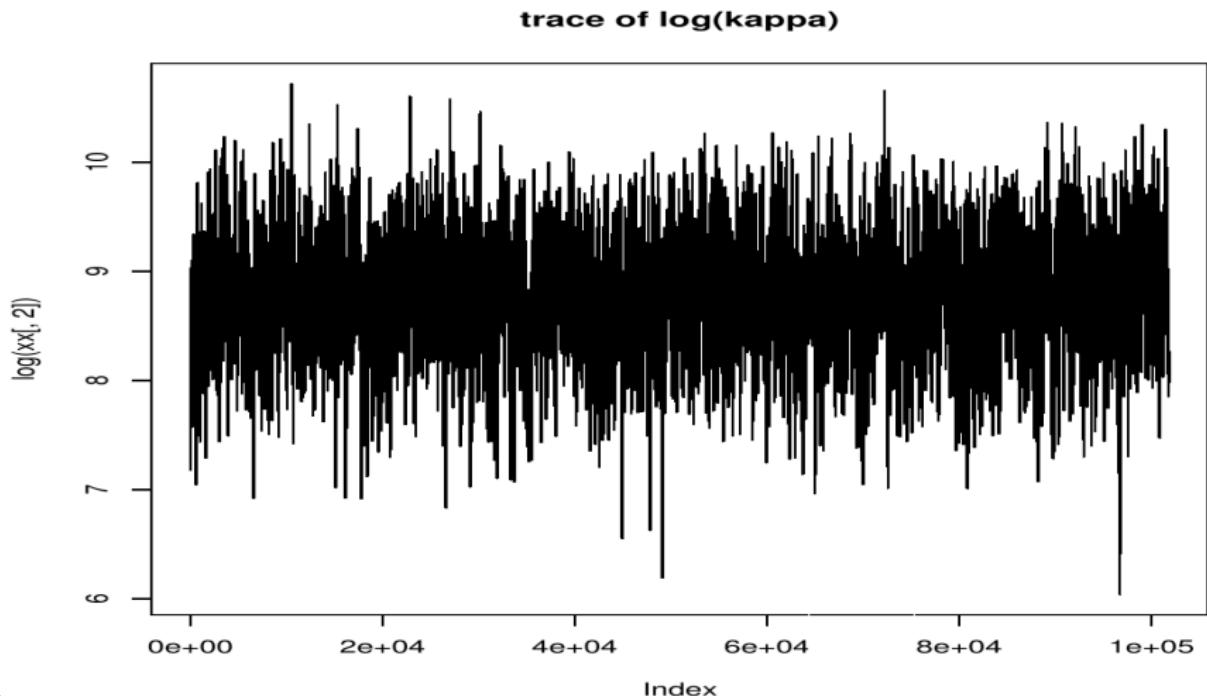
- Block (\mathbf{x}, θ) .

In most cases: if you can do one, you can do both.

Blocking scheme I

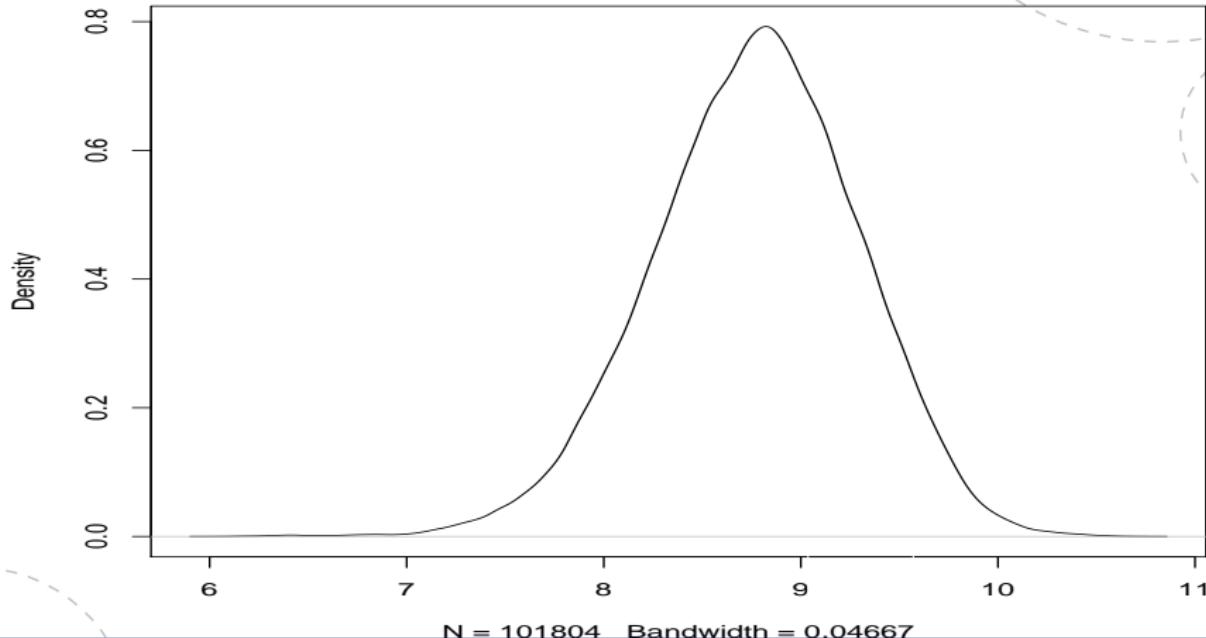
- $\kappa \sim \Gamma(\cdot, \cdot)$
- $\mathbf{x} \sim \mathcal{N}(\cdot, \cdot)$
- $\mathbf{w} \sim \mathcal{W}(\cdot)$ (conditional independent)

Results

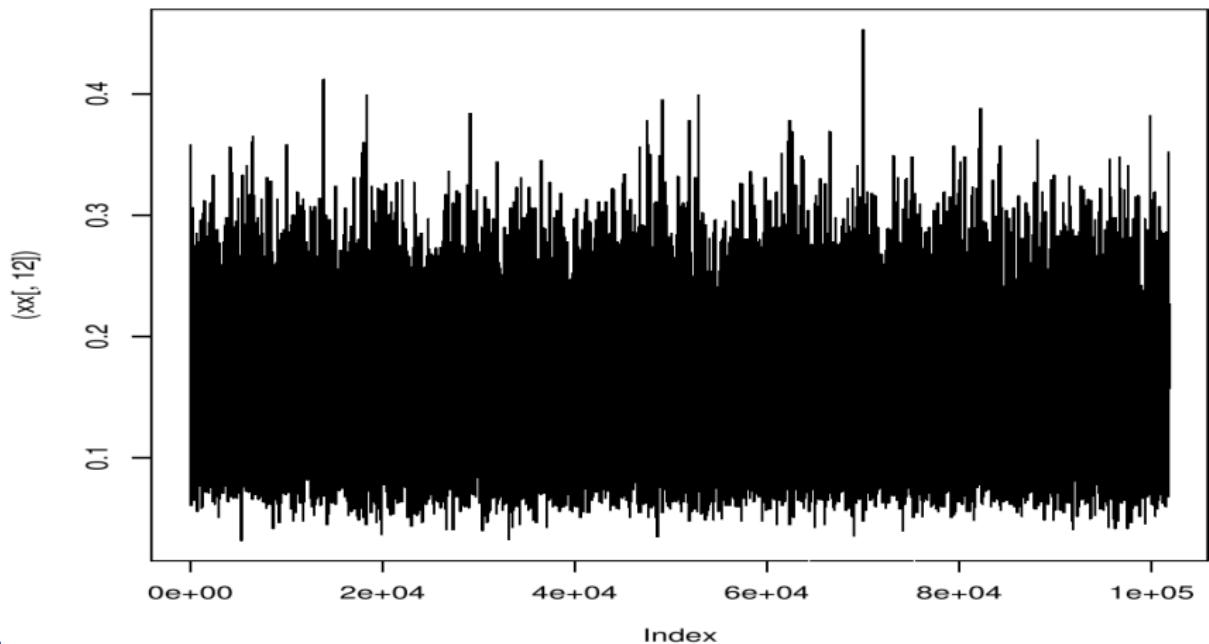


Results

density of log(kappa)

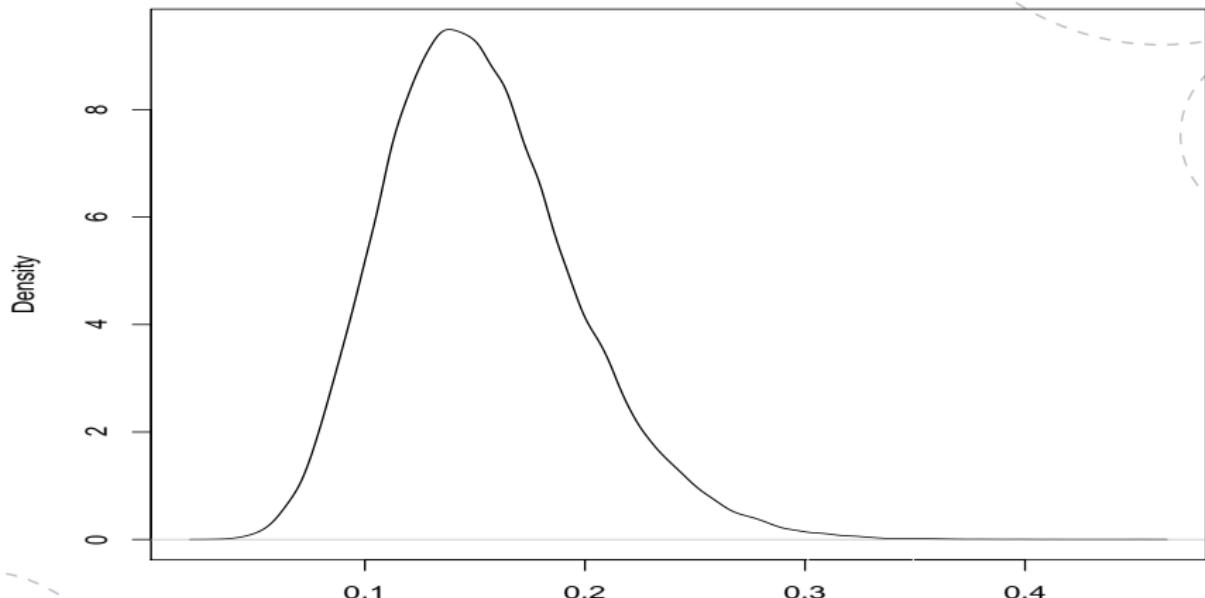


Results



Results

density of $x[10]$



Blocking scheme II

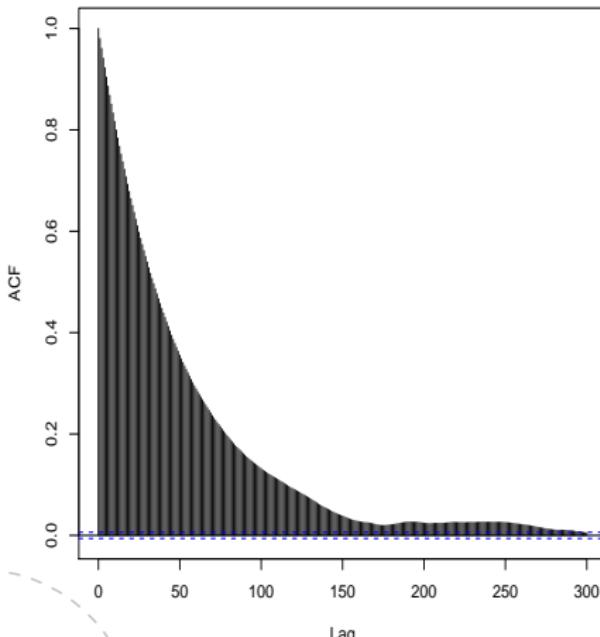
- Sample
 - $\kappa' \sim q(\kappa'; \kappa)$
 - $\mathbf{x}' | \kappa', \mathbf{y} \sim \mathcal{N}(\cdot, \cdot)$
- and then accept/reject (\mathbf{x}', κ') jointly
- $\mathbf{w} \sim \mathcal{W}(\cdot)$ (conditional independent)

Remarks

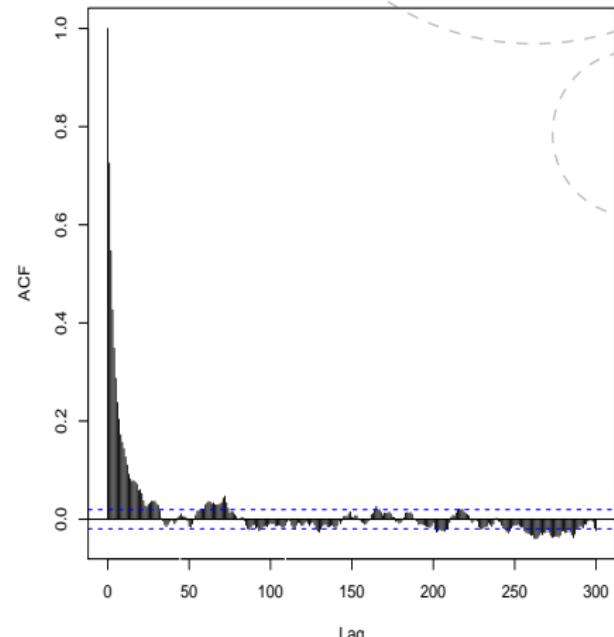
- If the normalising constant for $\mathbf{x}|\cdot$ is available, then this is an EASY FIX of scheme I.
- Usually makes a huge improvement
- Automatic “reparameterisation”
- Doubles the computational costs

Results

ACF(log(kappa) scheme I)



ACF(log(kappa) scheme II)



Removing the auxiliary variables

- The auxiliary variables makes the full conditional for \mathbf{x} Gaussian
- If we do not use them, the full conditional for \mathbf{x} looks like

$$\begin{aligned}\pi(\mathbf{x} | \dots) &\propto \exp\left(-\frac{1}{2}\mathbf{x}^T \mathbf{Q} \mathbf{x} + \sum_i \log(\pi(y_i|x_i))\right) \\ &\approx \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T (\mathbf{Q} + \text{diag}(\mathbf{c}))(\mathbf{x} - \boldsymbol{\mu})\right) \\ &= \pi_G(\mathbf{x} | \dots)\end{aligned}$$

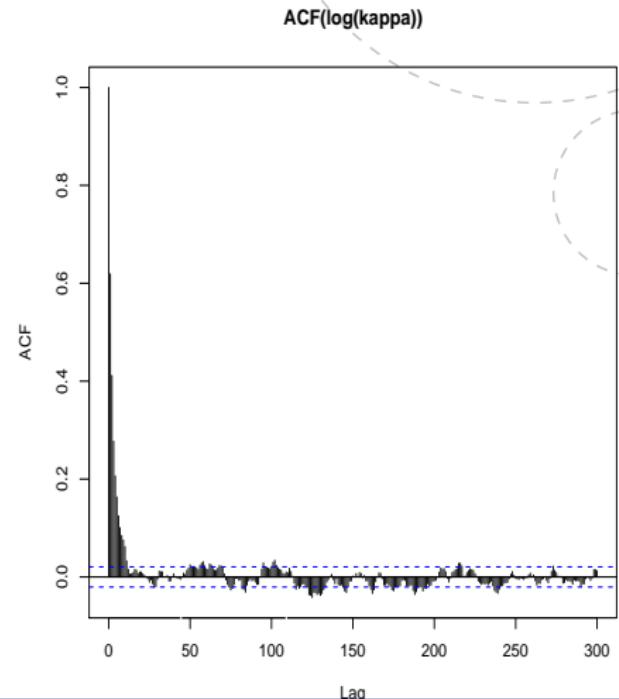
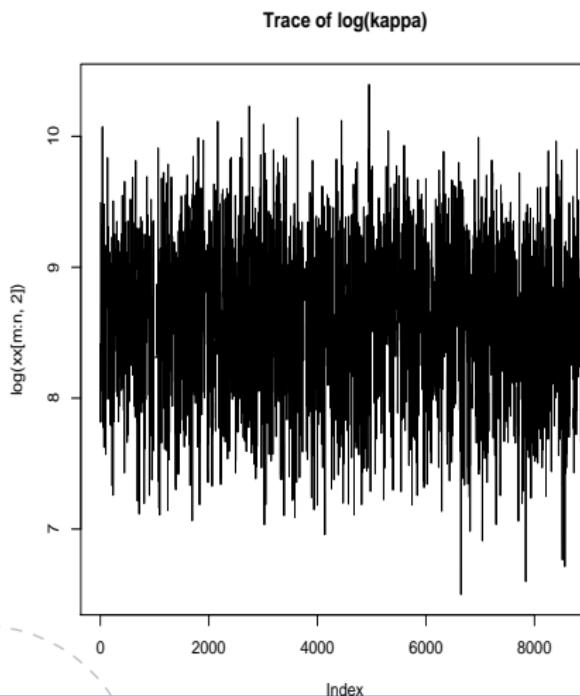
- The Gaussian approximation is constructed by matching the
 - mode, and the
 - curvature at the mode.

Improved one-block scheme

- $\kappa' \sim q(\cdot; \kappa)$
- $\mathbf{x}' \sim \pi_G(\mathbf{x} \mid \kappa', \mathbf{y})$
- Accept/reject (\mathbf{x}', κ') jointly

Note: $\pi_G(\cdot)$ is indexed by κ' , hence we need to compute one for each value of κ' .

Results



Independence sampler

We can construct an independence sampler, using $\pi_G(\cdot)$.

The Laplace-approximation for $\kappa | \mathbf{x}$:

$$\begin{aligned}\pi(\kappa | \mathbf{y}) &\propto \frac{\pi(\kappa) \pi(\mathbf{x}|\kappa) \pi(\mathbf{y}|\mathbf{x})}{\pi(\mathbf{x}|\kappa, \mathbf{y})} \\ &\approx \left. \frac{\pi(\kappa) \pi(\mathbf{x}|\kappa) \pi(\mathbf{y}|\mathbf{x})}{\pi_G(\mathbf{x}|\kappa, \mathbf{y})} \right|_{\mathbf{x}=\text{mode}(\kappa)}\end{aligned}$$

Hence, we do first

- Evaluate the Laplace-approximation at some “selected” points
- Build an interpolation log-spline
- Use this parametric model as $\tilde{\pi}(\kappa|\mathbf{y})$

Independence sampler

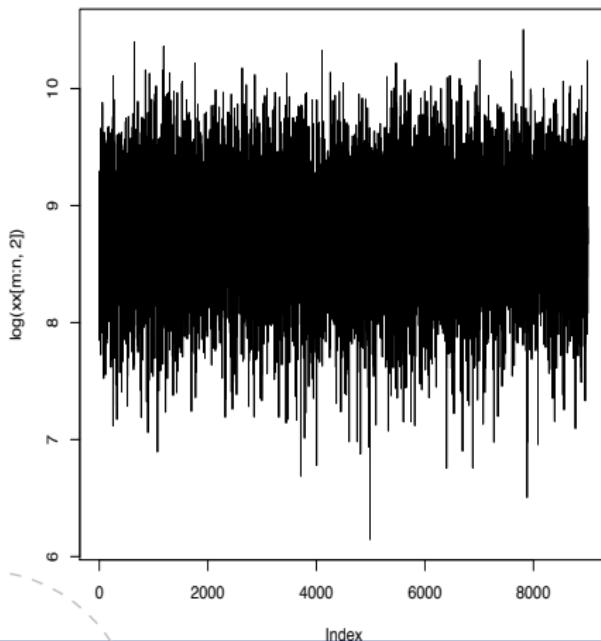
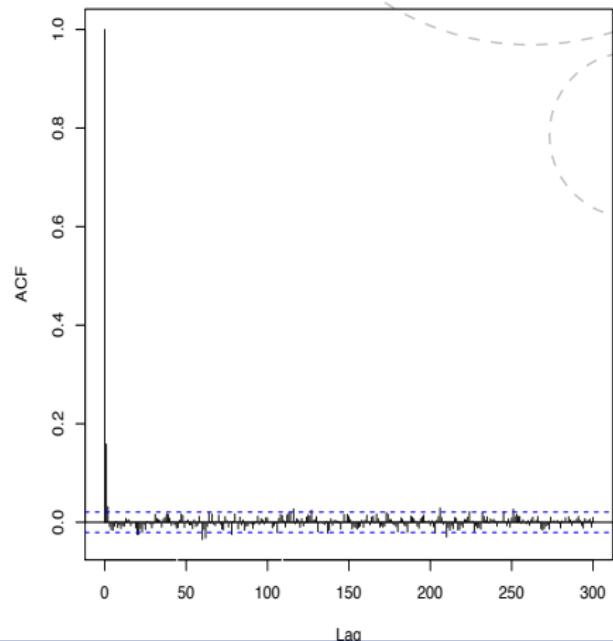
- $\kappa' \sim \tilde{\pi}(\kappa | \mathbf{y})$
- $\mathbf{x}' \sim \pi_G(\mathbf{x} | \kappa', \mathbf{y})$
- Accept/reject (κ', \mathbf{x}') jointly

Note:

$$\text{Corr}(x(t+k), x(t)) \approx (1 - \alpha)^{|k|}$$

In this example, $\alpha = 0.83\dots$

Results

Trace $\log(\kappa)$; independence samplerACF($\log(\kappa)$); independence sampler

Can we improve this sampler?

- Yes, if we are interested in the posterior marginals for κ and $\{x_i\}$.
- The marginals for the Gaussian proposal $\pi_G(\mathbf{x} | \dots)$, are known analytically.
- Just use numerical integration!

Deterministic inference

Posterior marginal for κ :

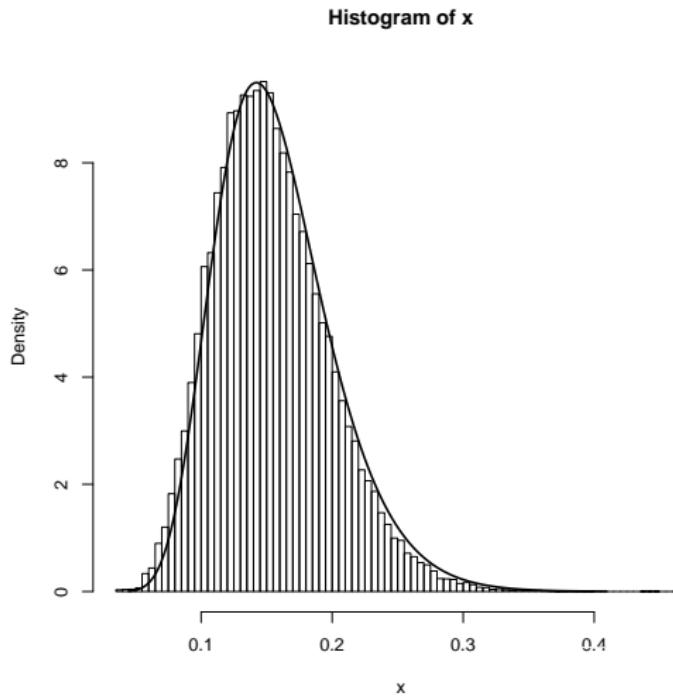
- Compute $\tilde{\pi}(\kappa | \mathbf{y})$

Posterior marginal for x_i :

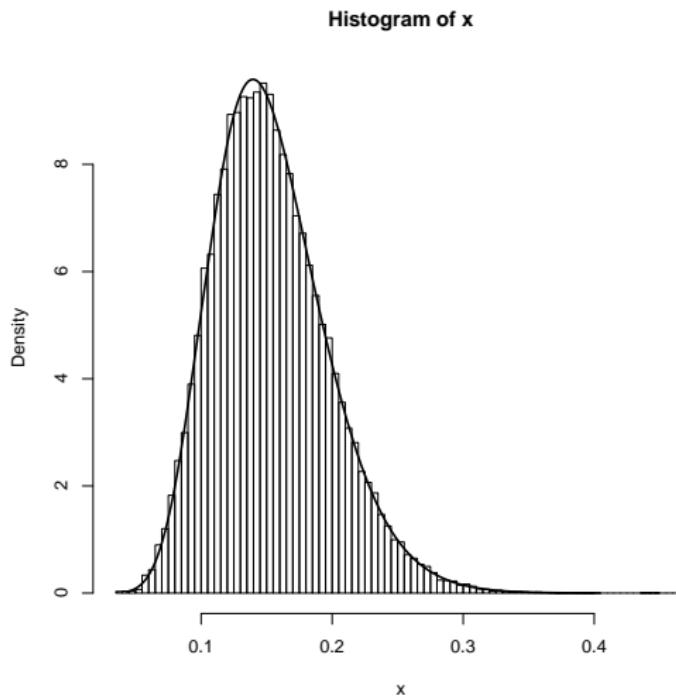
- Use numerical integration

$$\begin{aligned}\pi(x_i | \mathbf{y}) &= \int \pi(x_i | \mathbf{y}, \kappa) \pi(\kappa | \mathbf{y}) d\kappa \\ &\approx \sum_k \mathcal{N}(x_i; \mu_{\kappa_k}, \sigma^2(\kappa_k)) \times \tilde{\pi}(\kappa_k | \mathbf{y}) \times \Delta_k\end{aligned}$$

Results: Mixture of Gaussians



Results: Improved....



What can be learned from this exercise?

For a relative simple model, we have implemented

- single-site with auxiliary variables (looong time; hours)
- various forms for blocking (long time; many minutes)
- independence sampler (long time; many minutes)
- approximate inference (nearly instant; one second)

What can be learned from this exercise?

...

My guess; Most people will implement the single-site scheme, (possible) with auxiliary variables.

Which implies

- Most probably, the results would be not correct.
- They “accept” the long running-time.
- Trouble: such MCMC-schemes is not useful for routine analysis of similar data.

What can be learned from this exercise?

...

- In many cases, the situation is much worse in practice; this was a very simple model.
- Single-site MCMC is still the default choice for the non-expert user.
- Hierarchical models are popular, but they are difficult for MCMC.

Perhaps the development of models is not in sync with the development of inference? We cannot just wait for more powerful computers...

Outline

Introduction

Latent Gaussian Models

Gaussian Markov random fields

Gaussian Markov random fields

Computing with GMRFs

A case for approximate inference

INLA

Overview

Latent Gaussian models

Examples: 2D+

Latent Gaussian models

Latent Gaussian models is a model of the following form

- Observed data \mathbf{y} , $y_i|x_i \sim \pi(y_i|x_i, \theta)$
- Latent Gaussian field $\mathbf{x} \sim \mathcal{N}(\cdot, \Sigma(\theta))$
- Hyperparameters θ
 - variability
 - length/strength of dependence
 - parameters in the likelihood

$$\pi(\mathbf{x}, \theta | \mathbf{y}) \propto \pi(\theta) \pi(\mathbf{x} | \theta) \prod_{i \in \mathcal{I}} \pi(y_i | x_i, \theta)$$

Examples

- 1D Smoothing count data, general spline smoothing, semi-parametric regression, GLM(M), GAM(M), etc
- 2D Disease mapping, log-Gaussian Cox-processes, model-based geostatistics, 1D-models with spatial effect(s)
- (2+1)D Time-series of images, spatio-temporal models.
- 3D ?

Features

- Dimension of the latent Gaussian field, n , is large, $10^2 - 10^5$, but often Markov.
- Dimension of the hyperparameters $\dim(\theta)$ is small, 1 – 5, say.
(Current ‘best’ — 15)
- Dimension of the data $\dim(\mathbf{y})$ might vary, but is often non-Gaussian.

Longitudinal mixed effects model: Epil-example from BUGS

Patient	y ₁	y ₂	y ₃	y ₄	Trt	Base	Age
1	5	3	3	3	0	11	31
2	3	5	3	3	0	11	30
3	2	4	0	5	0	6	25
4	4	4	1	4	0	8	36
...							
8	40	20	21	12	0	52	42
9	5	6	6	5	0	12	37
...							
59	1	4	3	2	1	12	37

Longitudinal mixed effects model: Epil-example from BUGS

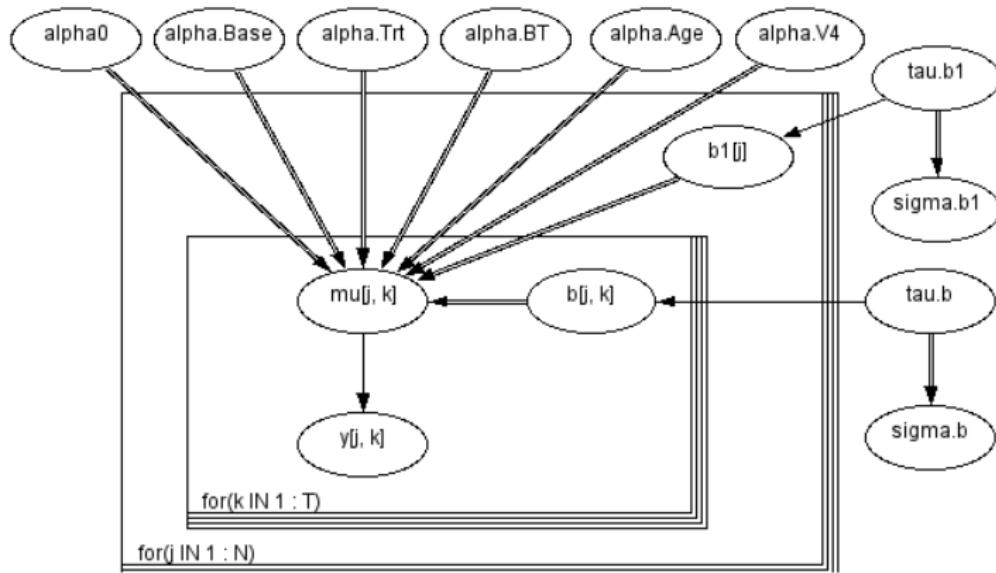
$$y_{jk} \sim \text{Poisson}(m_{jk})$$

$$\log m_{jk} = a_0 + a_{\text{Base}} \log(\text{Base}_j / 4) + a_{\text{Trt}} \text{Trt}_j + a_{\text{BT}} \text{Trt}_j \log(\text{Base}_j / 4) + \\ a_{\text{Age}} \text{Age}_j + a_{\sqrt{4}} \sqrt{4} + b1_j + b_{jk}$$

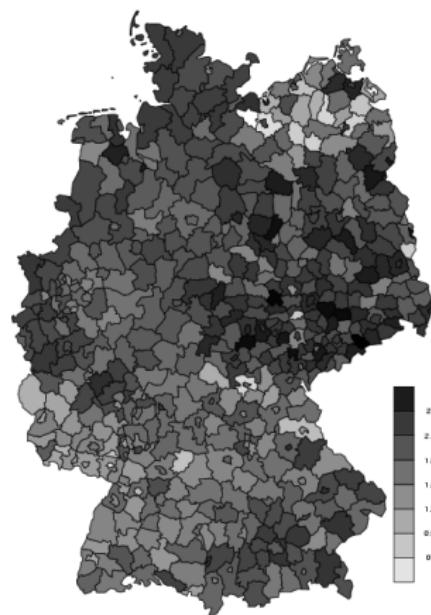
$$b1_j \sim \text{Normal}(0, t_{b1})$$

$$b_{jk} \sim \text{Normal}(0, t_b)$$

Longitudinal mixed effects model: Epil-example from BUGS

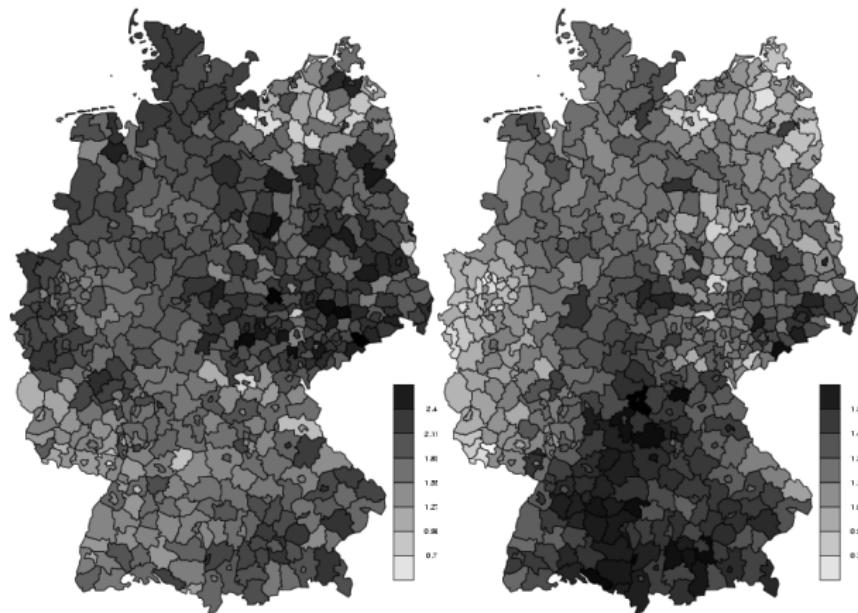


Examples of latent Gaussian models: 2D



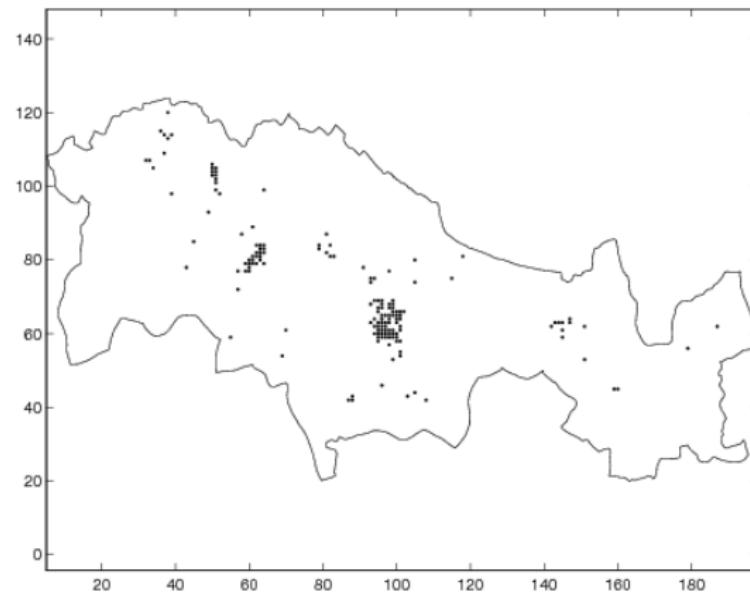
Disease mapping: Poisson data

Examples of latent Gaussian models: 2D



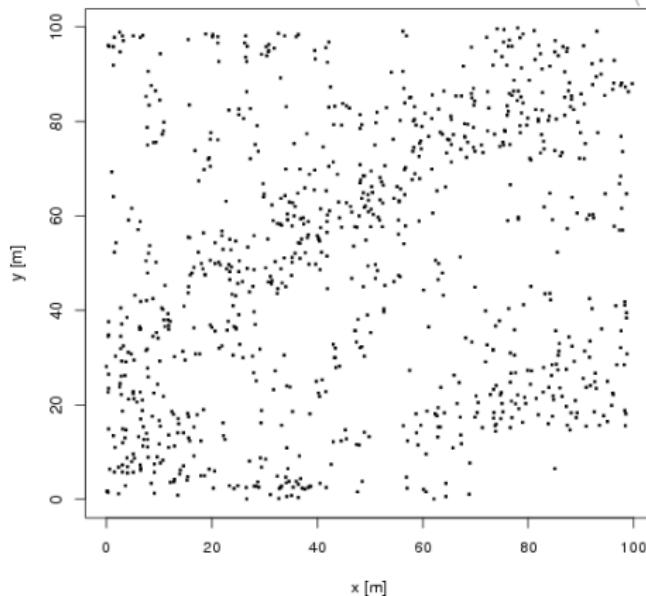
Joint disease mapping: Poisson data

Examples of latent Gaussian models: 2D



Spatial GLM with Binomial data

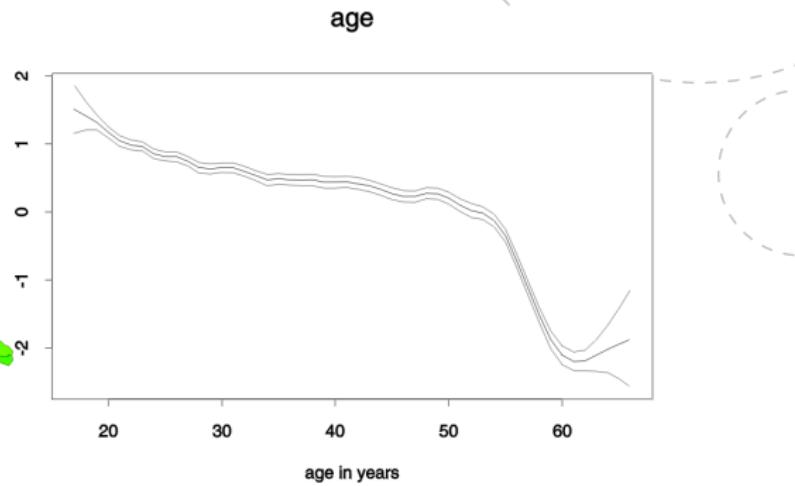
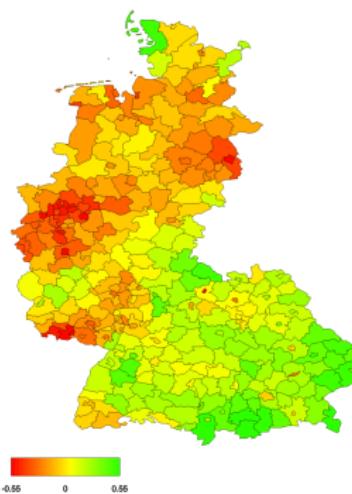
Examples of latent Gaussian models: 2D



Log-Gaussian Cox-process; Oaks-data

Examples of latent Gaussian models: 2D+

structured random effect



Spatial logit-model with semiparametric covariates

Tasks

Compute from

$$\pi(\mathbf{x}, \boldsymbol{\theta} \mid \mathbf{y}) \propto \pi(\boldsymbol{\theta}) \pi(\mathbf{x} \mid \boldsymbol{\theta}) \prod_{i \in \mathcal{I}} \pi(y_i \mid x_i)$$

the posterior marginals:

$$\pi(x_i \mid \mathbf{y}), \quad \text{for some or all } i$$

and/or

$$\pi(\theta_i \mid \mathbf{y}), \quad \text{for some or all } i$$

Our approach: Approximate Bayesian Inference

- Can we compute (approximate) marginals directly without using MCMC?
- YES!
- Gain
 - Huge speedup & accuracy
 - The ability to treat latent Gaussian models properly ;-)

Main ideas (I)

Main ideas are simple and based on the identity

$$\pi(z) = \frac{\pi(x, z)}{\pi(x|z)} \quad \text{leading to} \quad \tilde{\pi}(z) = \frac{\pi(x, z)}{\tilde{\pi}(x|z)}$$

When $\tilde{\pi}(x|z)$ is the Gaussian-approximation, this is the Laplace-approximation.

Main ideas (II)

Construct the approximations to

1. $\pi(\theta|\mathbf{y})$
2. $\pi(x_i|\theta, \mathbf{y})$

then we integrate

$$\pi(x_i|\mathbf{y}) = \int \pi(\theta|\mathbf{y}) \pi(x_i|\theta, \mathbf{y}) d\theta$$

$$\pi(\theta_j|\mathbf{y}) = \int \pi(\theta|\mathbf{y}) d\theta_{-j}$$

The Laplace approximation: The classic case

Compute and approximation to the integral

$$\int \exp(ng(x)) dx$$

where n is the parameter going to ∞ .

Let x_0 be the mode of $g(x)$ and assume $g(x_0) = 0$:

$$g(x) = \frac{1}{2}g''(x_0)(x - x_0)^2 + \dots$$

The Laplace approximation: The classic case...

Then

$$\int \exp(ng(x)) dx = \sqrt{\frac{2\pi}{n(-g''(x_0))}} + \dots$$

- As $n \rightarrow \infty$, then the integrand gets more and more peaked.
- Error should tends to zero as $n \rightarrow \infty$
- Detailed analysis gives

$$\text{relative error}(n) = 1 + \mathcal{O}(1/n)$$

Extension I

$$g_n(x) = \frac{1}{n} \sum_{i=1}^n g_i(x)$$

then the mode x_0 depends on n as well.

Extension II

$$\int \exp(ng(\mathbf{x})) d\mathbf{x}$$

and \mathbf{x} is multivariate, then

$$\int \exp(ng(\mathbf{x})) d\mathbf{x} = \sqrt{\frac{(2\pi)^n}{n|-\mathbf{H}|}}$$

where \mathbf{H} is the hessian (matrix) at the mode

$$H_{ij} = \left. \frac{\partial^2}{\partial x_i \partial x_j} g(\mathbf{x}) \right|_{\mathbf{x}=\mathbf{x}_0}$$

Computing marginals

- Our main issue is to compute marginals
- We can use the Laplace-approximation for this issue as well
- A more “statistical” derivation might be appropriate

Computing marginals...

Consider the general problem

- θ is hyper-parameter with prior $\pi(\theta)$
- x is latent with density $\pi(x|\theta)$
- y is observed with likelihood $\pi(y|x)$

then

$$\pi(\theta|y) = \frac{\pi(x, \theta|y)}{\pi(x|\theta, y)}$$

for any x !

Computing marginals...

Further,

$$\begin{aligned}
 \pi(\theta|y) &= \frac{\pi(x, \theta|y)}{\pi(x|\theta, y)} \\
 &\propto \frac{\pi(\theta) \pi(x|\theta) \pi(y|x)}{\pi(x|\theta, y)} \\
 &\approx \left. \frac{\pi(\theta) \pi(x|\theta) \pi(y|x)}{\pi_G(x|\theta, y)} \right|_{x=x^*(\theta)}
 \end{aligned}$$

where $\pi_G(x|\theta, y)$ is the Gaussian approximation of $\pi(x|\theta, y)$ and $x^*(\theta)$ is the mode.

Computing marginals...

Error:

With n repeated measurements of the same x , then the error is

$$\tilde{\pi}(\theta|y) = \pi(\theta|y)(1 + \mathcal{O}(n^{-3/2}))$$

after renormalisation.

Relative error is a very nice property!

The GMRF-approximation

$$\begin{aligned}\pi(\mathbf{x} | \mathbf{y}) &\propto \exp\left(-\frac{1}{2}\mathbf{x}^T \mathbf{Q} \mathbf{x} + \sum_i \log \pi(y_i | x_i)\right) \\ &\approx \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T (\mathbf{Q} + \text{diag}(c_i))(\mathbf{x} - \boldsymbol{\mu})\right) = \tilde{\pi}(\mathbf{x} | \boldsymbol{\theta}, \mathbf{y})\end{aligned}$$

Constructed as follows:

- Locate the mode \mathbf{x}^*
- Expand to second order

Markov and computational properties are preserved

Main ideas

Posterior

$$\pi(\mathbf{x}, \boldsymbol{\theta} \mid \mathbf{y}) \propto \pi(\boldsymbol{\theta}) \pi(\mathbf{x} \mid \boldsymbol{\theta}) \prod_{i \in \mathcal{I}} \pi(y_i \mid x_i, \boldsymbol{\theta})$$

Do the integration wrt $\boldsymbol{\theta}$ numerically

Main ideas

Posterior

$$\pi(\mathbf{x}, \boldsymbol{\theta} \mid \mathbf{y}) \propto \pi(\boldsymbol{\theta}) \pi(\mathbf{x} \mid \boldsymbol{\theta}) \prod_{i \in \mathcal{I}} \pi(y_i \mid x_i, \boldsymbol{\theta})$$

Do the integration wrt $\boldsymbol{\theta}$ numerically

$$\pi(x_i \mid \mathbf{y}) = \int \pi(\boldsymbol{\theta} \mid \mathbf{y}) \pi(x_i \mid \boldsymbol{\theta}, \mathbf{y}) d\boldsymbol{\theta}$$

$$\pi(\theta_j \mid \mathbf{y}) = \int \pi(\boldsymbol{\theta} \mid \mathbf{y}) d\boldsymbol{\theta}_{-j}$$

Main ideas

Posterior

$$\pi(\mathbf{x}, \boldsymbol{\theta} \mid \mathbf{y}) \propto \pi(\boldsymbol{\theta}) \pi(\mathbf{x} \mid \boldsymbol{\theta}) \prod_{i \in \mathcal{I}} \pi(y_i \mid x_i, \boldsymbol{\theta})$$

Do the integration wrt $\boldsymbol{\theta}$ numerically

$$\tilde{\pi}(x_i \mid \mathbf{y}) = \int \tilde{\pi}(\boldsymbol{\theta} \mid \mathbf{y}) \tilde{\pi}(x_i \mid \boldsymbol{\theta}, \mathbf{y}) d\boldsymbol{\theta}$$

$$\tilde{\pi}(\theta_j \mid \mathbf{y}) = \int \tilde{\pi}(\boldsymbol{\theta} \mid \mathbf{y}) d\boldsymbol{\theta}_{-j}$$

Remarks

1. Expect $\tilde{\pi}(\theta|\mathbf{y})$ to be accurate, since
 - $\mathbf{x}|\theta$ is *a priori* Gaussian
 - Likelihood models are ‘well-behaved’ so

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

is *almost* Gaussian.

2. There are no distributional assumptions on $\theta|\mathbf{y}$
3. Similar remarks are valid to

$$\tilde{\pi}(x_i \mid \theta, \mathbf{y})$$

The Laplace approximation

The *Laplace approximation* for $\pi(\theta|\mathbf{y})$ is

$$\begin{aligned}\pi(\boldsymbol{\theta} \mid \mathbf{y}) &= \frac{\pi(\mathbf{x}, \boldsymbol{\theta} \mid \mathbf{y})}{\pi(\mathbf{x} \mid \mathbf{y}, \boldsymbol{\theta})} \quad (\text{any } \mathbf{x}) \\ &\approx \frac{\pi(\mathbf{x}, \boldsymbol{\theta} \mid \mathbf{y})}{\widetilde{\pi}(\mathbf{x} \mid \mathbf{y}, \boldsymbol{\theta})} \Big|_{\mathbf{x}=\mathbf{x}^*(\boldsymbol{\theta})} = \widetilde{\pi}(\boldsymbol{\theta} \mid \mathbf{y})\end{aligned}\tag{4}$$

Remarks

The Laplace approximation

$$\tilde{\pi}(\theta|\mathbf{y})$$

turn out to be accurate: $\mathbf{x}|\mathbf{y}, \theta$ appears *almost Gaussian* in most cases, as

- \mathbf{x} is *a priori* Gaussian.
- \mathbf{y} is typically not very informative.
- Observational model is usually ‘well-behaved’.

Note: $\tilde{\pi}(\theta|\mathbf{y})$ itself does *not* look Gaussian. Thus, a Gaussian approximation of (θ, \mathbf{x}) will be inaccurate.

Approximating $\pi(x_i | \mathbf{y}, \boldsymbol{\theta})$

This task is more challenging, since

- dimension of \mathbf{x} , n is large
- and there are potential n marginals to compute, or at least $\mathcal{O}(n)$.

An obvious simple and fast alternative, is to use the GMRF-approximation

$$\tilde{\pi}(x_i | \boldsymbol{\theta}, \mathbf{y}) = \mathcal{N}(x_i; \mu(\boldsymbol{\theta}), \sigma^2(\boldsymbol{\theta}))$$

Computing marginal variances for GMRFs*

Let

$$\mathbf{Q} = \mathbf{V}\mathbf{D}\mathbf{V}^T$$

- where \mathbf{D} is a diagonal matrix, and
- \mathbf{V} is a lower triangular matrix with ones on the diagonal.

The matrix identity

$$\boldsymbol{\Sigma} = \mathbf{D}^{-1} \mathbf{V}^{-1} + (\mathbf{I} - \mathbf{V}^T) \boldsymbol{\Sigma}$$

define *recursions* which can be used to compute

- $\text{Var}(x_i)$ and $\text{Cov}(x_i, x_j)$ for $i \sim j$

essentially without cost when the Cholesky triangle \mathbf{L} is known.
 This is a result from 1973.

General algorithm

for $i = n, \dots, 1$

for decreasing j in $\mathcal{I}(i)$

Compute $\Sigma_{i,j}$ from

$$\Sigma_{ij} = \delta_{ij}/L_{ii}^2 - \frac{1}{L_{ii}} \sum_{k \in \mathcal{I}(i)}^n L_{ki} \Sigma_{kj}, \quad j \geq i, \quad i = n, \dots, 1.$$

Band matrices

for $i = n, \dots, 1$

 for $j = \min(i + b_w, n), \dots, i$

 Compute $\Sigma_{i,j}$ from

$$\Sigma_{ij} = \delta_{ij}/L_{ii}^2 - \frac{1}{L_{ii}} \sum_{k \in \mathcal{I}(i)}^n L_{ki} \Sigma_{kj}, \quad j \geq i, \quad i = n, \dots, 1.$$

Equivalent to Kalman-recursions for smoothing.

Laplace approximation of $\pi(x_i | \theta, y)$

- The Laplace approximation:

$$\tilde{\pi}(x_i | y, \theta) \approx \frac{\pi(\mathbf{x}, \theta | y)}{\tilde{\pi}(\mathbf{x}_{-i} | x_i, y, \theta)} \Big|_{\mathbf{x}_{-i} = \mathbf{x}_{-i}^*(x_i, \theta)}$$

- Again, approximation is very good, as $\mathbf{x}_{-i} | \mathbf{x}_i, \theta$ is ‘almost Gaussian’,
- but it is expensive. In order to get the n marginals:
 - perform n optimisations, and
 - n factorisations of $n - 1 \times n - 1$ matrices.

Can be “solved”.

Simplified Laplace Approximation

An series expansion of the LA for $\pi(x_i|\theta, \mathbf{y})$:

- computational much faster: $\mathcal{O}(n \log n)$ for each i
- correct the Gaussian approximation for error in shift and skewness

$$\log \tilde{\pi}(x_i|\theta, \mathbf{y}) = -\frac{1}{2}x_i^2 + bx_i + \frac{1}{6}d x_i^3 + \dots$$

- Fit a skew-Normal density

$$2\phi(x)\Phi(ax)$$

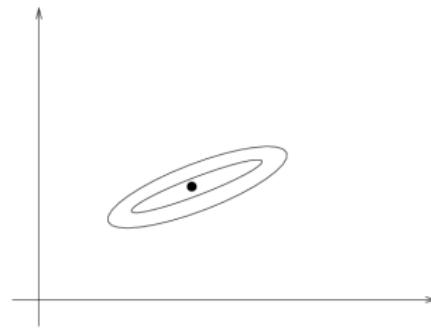
- sufficiently accurate for most applications

The integrated nested Laplace approximation (INLA) I

Step I Explore $\tilde{\pi}(\theta|y)$

The integrated nested Laplace approximation (INLA) I

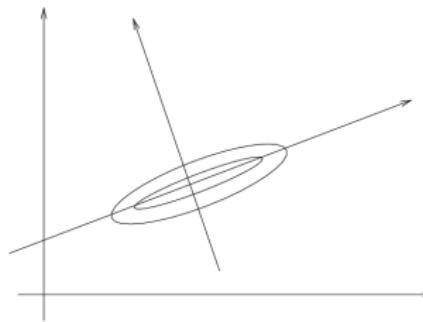
Step I Explore $\tilde{\pi}(\theta|y)$
— Locate the mode



The integrated nested Laplace approximation (INLA) I

Step I Explore $\tilde{\pi}(\theta|y)$

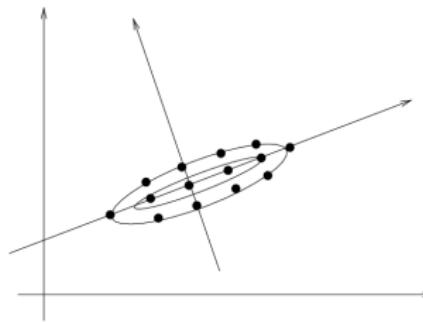
- Locate the mode
- Use the Hessian to construct new variables



The integrated nested Laplace approximation (INLA) I

Step I Explore $\tilde{\pi}(\theta|y)$

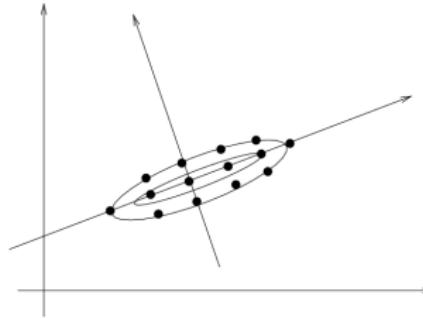
- Locate the mode
- Use the Hessian to construct new variables
- Grid-search



The integrated nested Laplace approximation (INLA) I

Step I Explore $\tilde{\pi}(\theta|y)$

- Locate the mode
- Use the Hessian to construct new variables
- Grid-search
- Can be case-specific



The integrated nested Laplace approximation (INLA) II

Step II For each θ_j

- For each i , evaluate the Laplace approximation for selected values of x_i
- Build a Skew-Normal or log-spline corrected Gaussian

$$\mathcal{N}(x_i; \mu_i, \sigma_i^2) \times \exp(\text{spline})$$

to represent the conditional marginal density.

The integrated nested Laplace approximation (INLA) III

Step III Sum out θ_j

- For each i , sum out θ

$$\tilde{\pi}(x_i \mid \mathbf{y}) \propto \sum_j \tilde{\pi}(x_i \mid \mathbf{y}, \theta_j) \times \tilde{\pi}(\theta_j \mid \mathbf{y})$$

- Build a log-spline corrected Gaussian

$$\mathcal{N}(x_i; \mu_i, \sigma_i^2) \times \exp(\text{spline})$$

to represent $\tilde{\pi}(x_i \mid \mathbf{y})$.

Computing posterior marginals for θ_j (I)

Main idea

- Use the integration-points and build an interpolant
- Use numerical integration on that interpolant

Computing posterior marginals for θ_j (II)

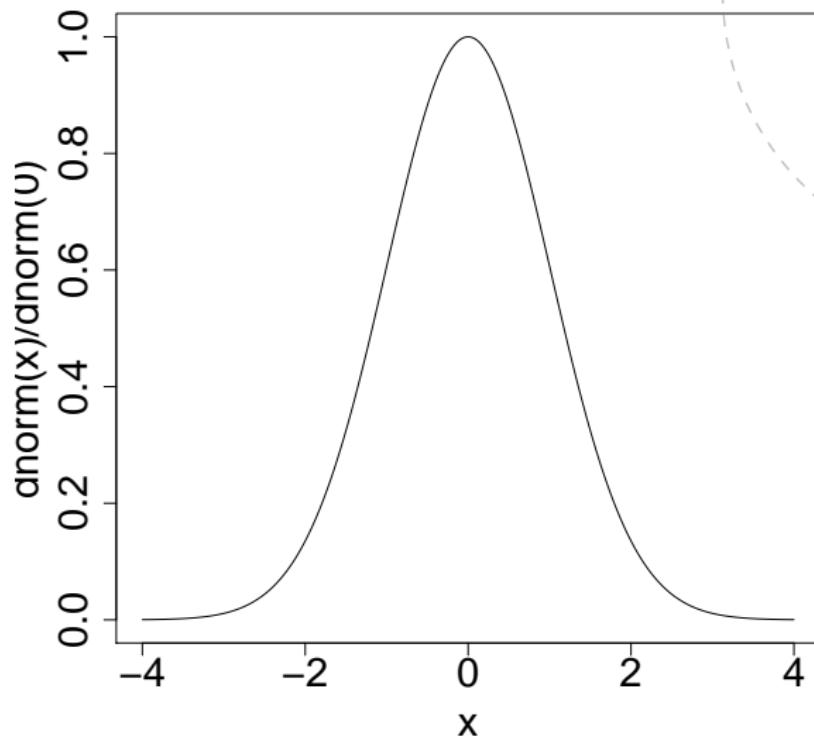
Practical approach (high accuracy)

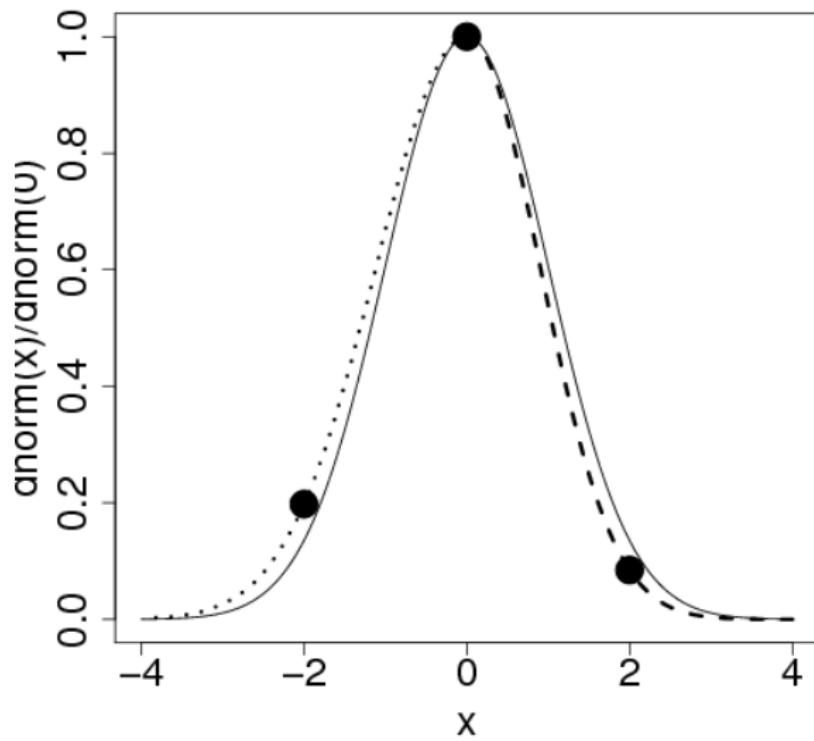
- Rerun using a fine integration grid
- Possibly with no rotation
- Just sum up at grid points, then interpolate

Computing posterior marginals for θ_j (II)

Practical approach (lower accuracy)

- Use the Gaussian approximation at the mode θ^*
- ...BUT, adjust the standard deviation in each direction
- Then use numerical integration





How can we assess the error in the approximations?

Tool 1: Compare a sequence of improved approximations

1. Gaussian approximation
2. Simplified Laplace
3. Laplace

How can we assess the error in the approximations?

Tool 3: Estimate the “effective” number of parameters as defined in the *Deviance Information Criteria*:

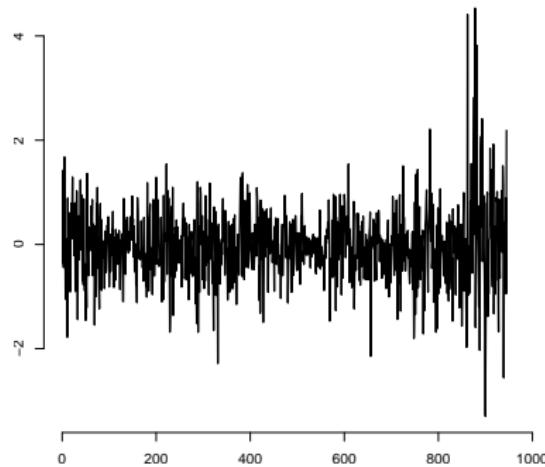
$$p_D(\theta) = \bar{D}(\mathbf{x}; \theta) - D(\bar{\mathbf{x}}; \theta)$$

and compare this with the number of observations.

Low ratio is good.

This criteria has theoretical justification.

Stochastic Volatility model



Log of the daily difference of the pound-dollar exchange rate from October 1st, 1981, to June 28th, 1985.

Stochastic Volatility model

Simple model

$$x_t \mid x_1, \dots, x_{t-1}, \tau, \phi \sim \mathcal{N}(\phi x_{t-1}, 1/\tau)$$

where $|\phi| < 1$ to ensure a stationary process.

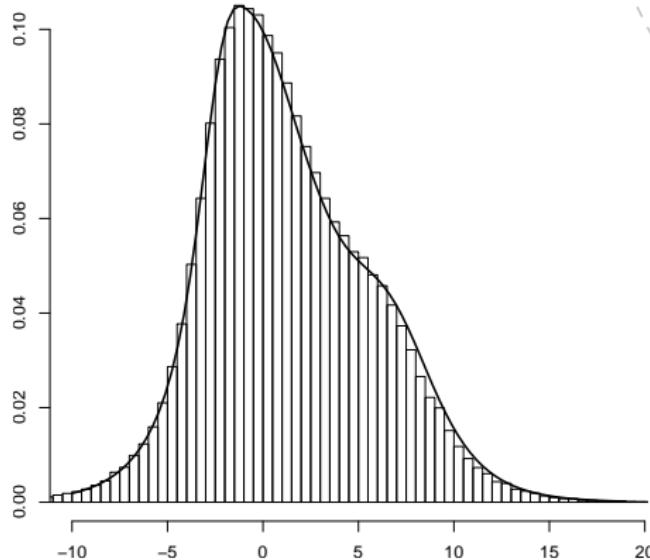
Observations are taken to be

$$y_t \mid x_1, \dots, x_t, \mu \sim \mathcal{N}(0, \exp(\mu + x_t))$$

Results

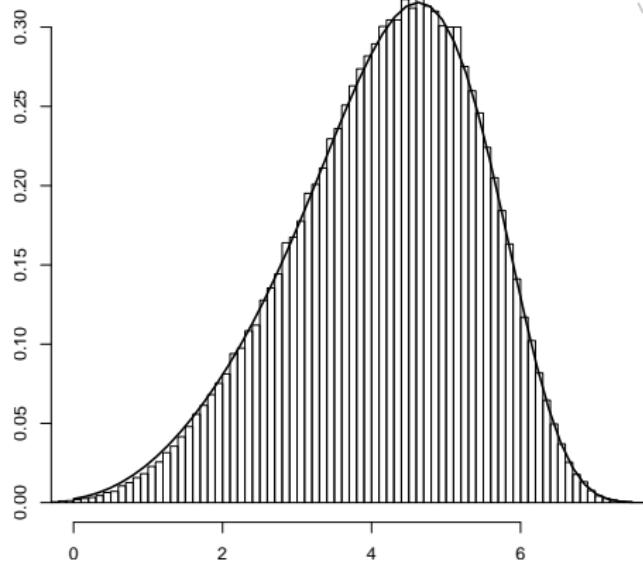
Using just the first 50 data-points only, which makes the problem much harder.

Results



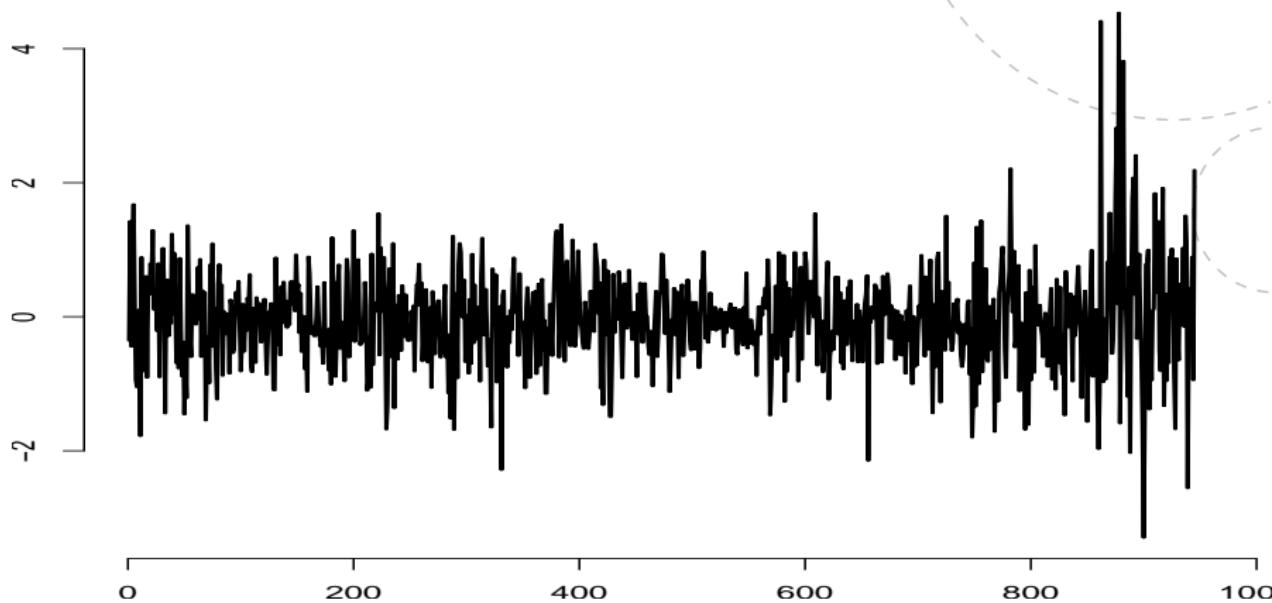
$$\nu = \text{logit}(2\phi - 1)$$

Results



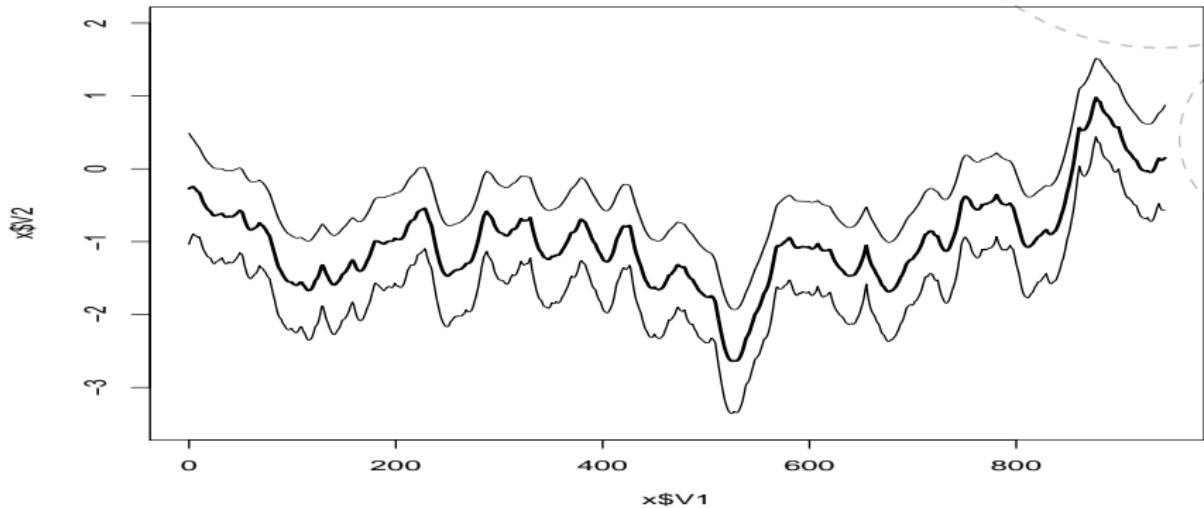
$\log(\kappa_x)$

Using the full dataset



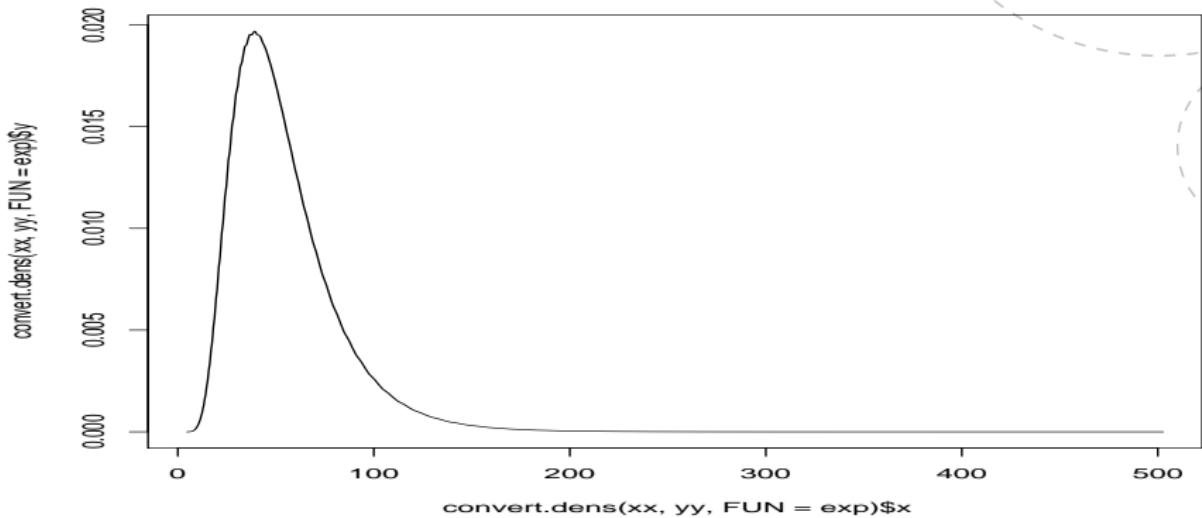
The Pound-Dollar data.

Using the full dataset



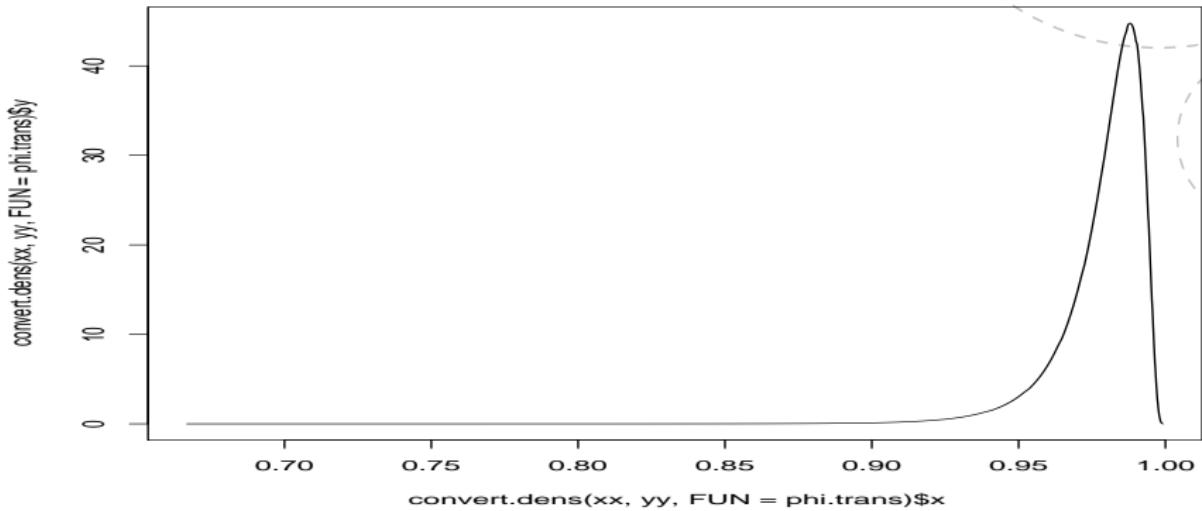
Mean of $x_t + \mu$

Using the full dataset



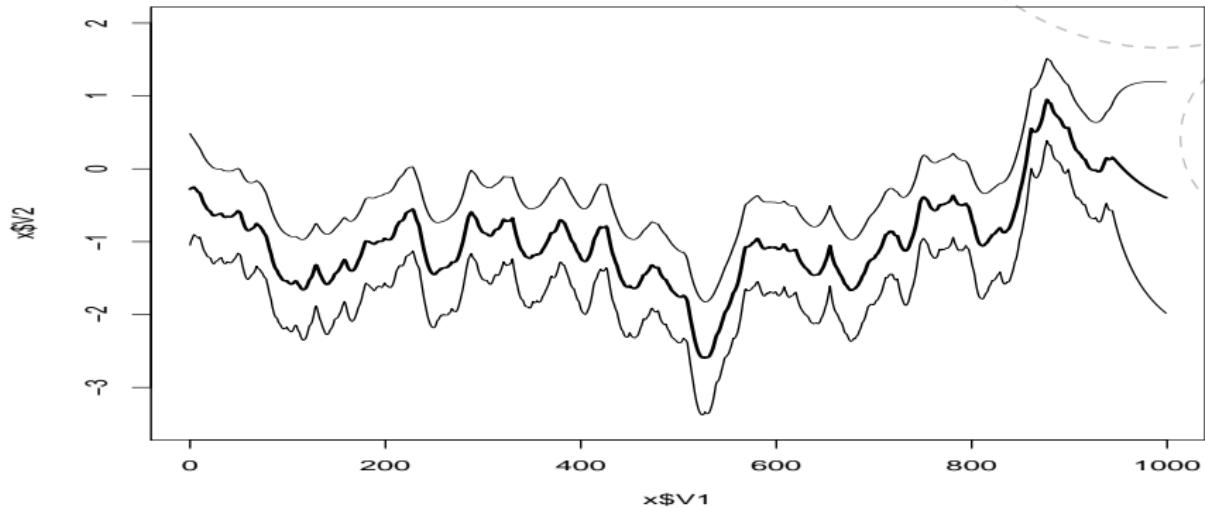
The posterior marginal for the precision.

Using the full dataset



The posterior marginal for the lag-1 correlation.

Using the full dataset



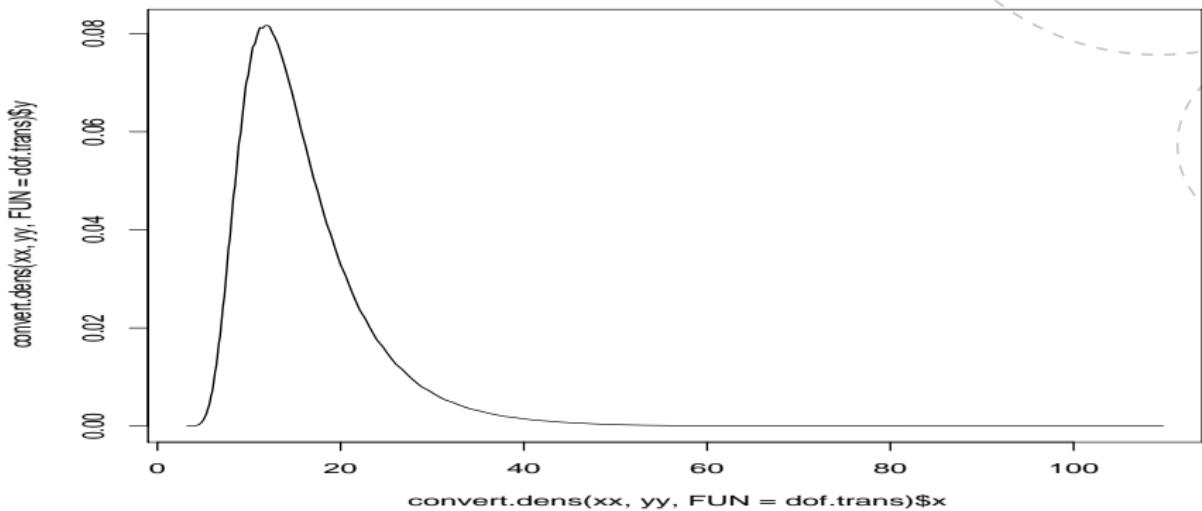
Predictions for $\mu + x_{t+k}$

New data-model: Student-t_ν

Now extend the model to use Student-t_ν distribution

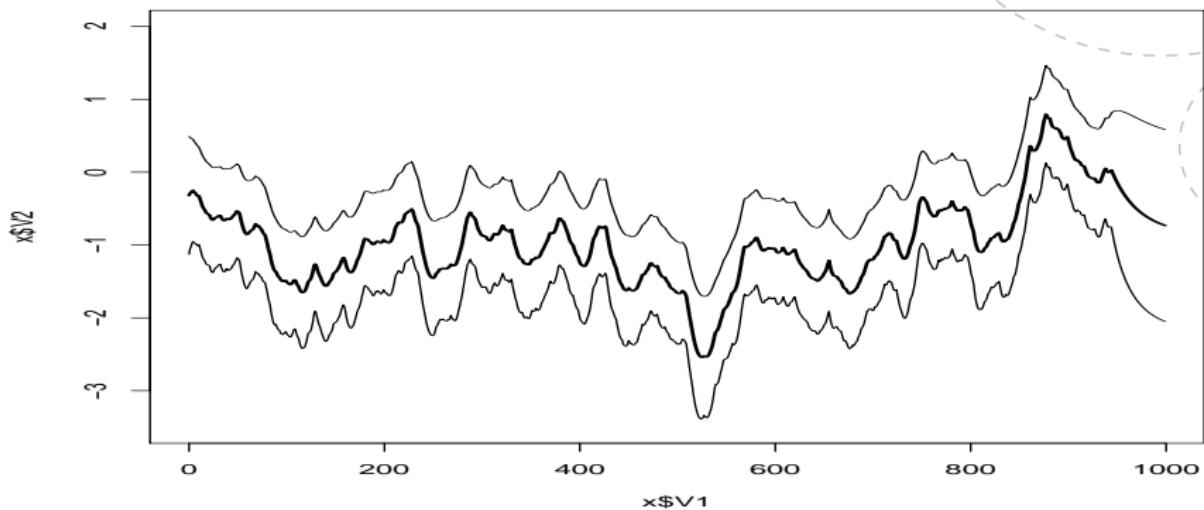
$$y_t \mid x_1, \dots, x_t \sim \exp(\mu/2 + x_t/2) \times \text{Student-}t_{\nu} / \sqrt{\nu/(\nu - 2)}$$

Student- t_{ν}



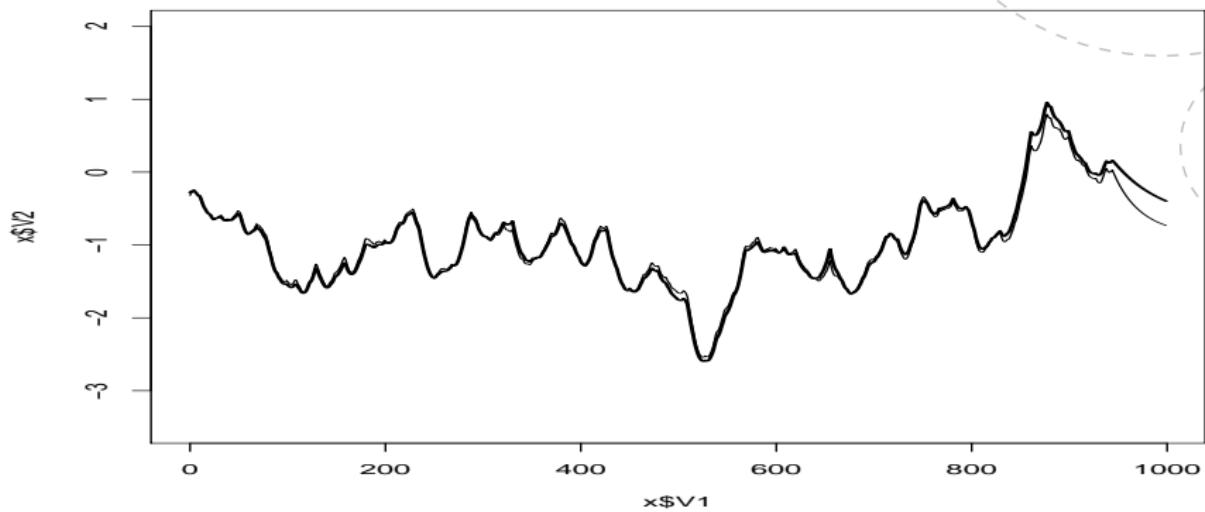
Posterior marginal for ν .

Student- t_ν



Predictions

Student- t_ν



Comparing predictions with Student- t_ν and Gaussian

Student- t_ν

However,

- No support for Student- t_ν in the data
 - Bayes-factor
 - Deviance Information Criteria

Epil-example from Win/Open-BUGS

Patient	y ₁	y ₂	y ₃	y ₄	Trt	Base	Age
1	5	3	3	3	0	11	31
2	3	5	3	3	0	11	30
3	2	4	0	5	0	6	25
4	4	4	1	4	0	8	36
....							
8	40	20	21	12	0	52	42
9	5	6	6	5	0	12	37
....							
59	1	4	3	2	1	12	37

Epil-example from Win/Open-BUGS

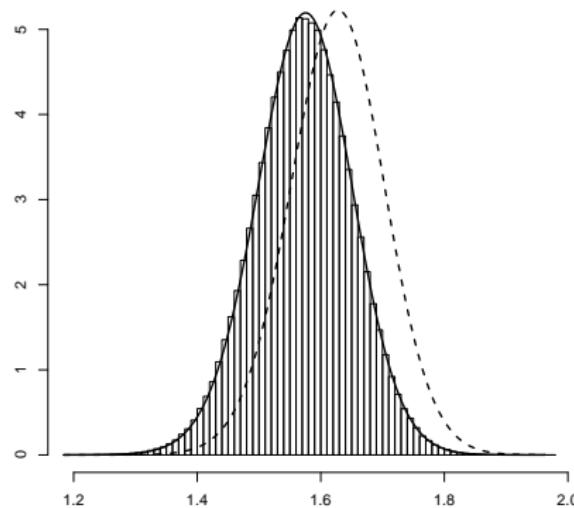
$$y_{jk} \sim \text{Poisson}(m_{jk})$$

$$\log m_{jk} = a_0 + a_{\text{Base}} \log(\text{Base}_j / 4) + a_{\text{Trt}} \text{Trt}_j + a_{\text{BT}} \text{Trt}_j \log(\text{Base}_j / 4) + a_{\text{Age}} \text{Age}_j + a_{\sqrt{4}} \sqrt{4} + b1_j + b_{jk}$$

$$b1_j \sim \text{Normal}(0, t_{b1})$$

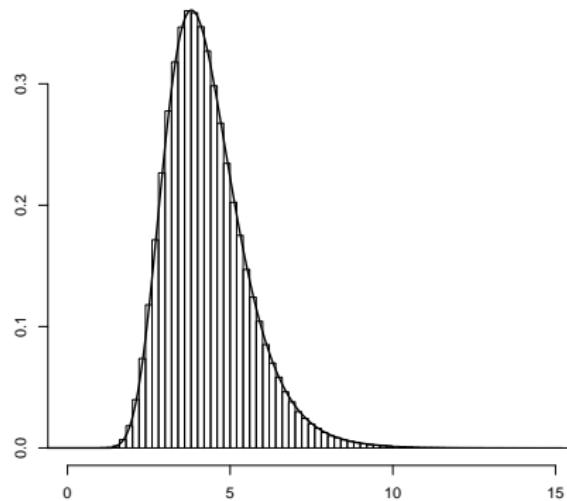
$$b_{jk} \sim \text{Normal}(0, t_b)$$

Epil-example from Win/Open-BUGS



Marginals for a_0

Epil-example from Win/Open-BUGS



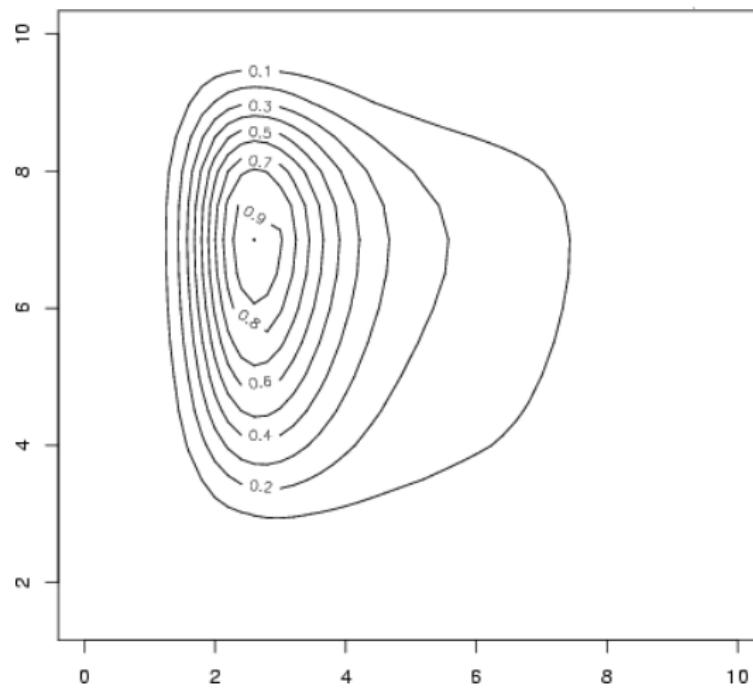
Marginals for τ_{b1}

Disease mapping: The BYM-model

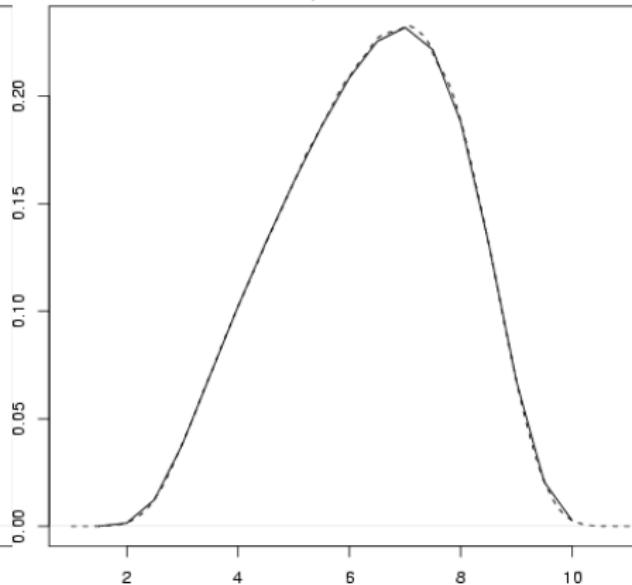
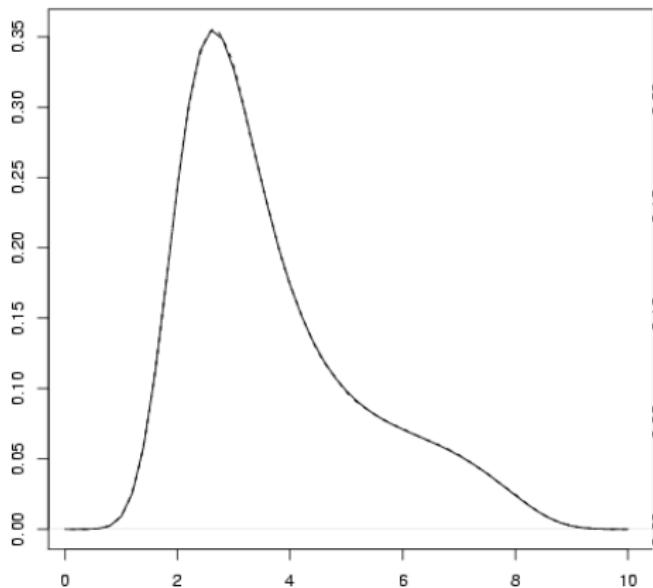
- Data $y_i \sim \text{Poisson}(E_i \exp(\eta_i))$
- Log-relative risk $\eta_i = u_i + v_i$
- Structured component \boldsymbol{u}
- Unstructured component \boldsymbol{v}
- Log-precisions $\log \kappa_u$ and $\log \kappa_v$
- A hard case: Insulin Dependent Diabetes Mellitus in 366 districts of Sardinia. Few counts.
- $\dim(\theta) = 2$.



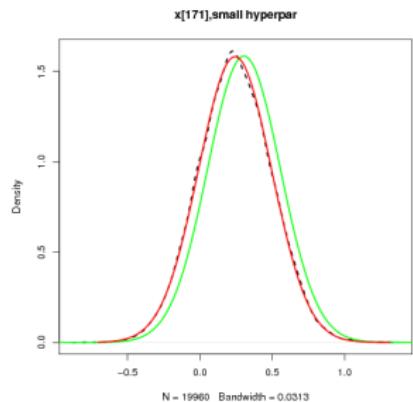
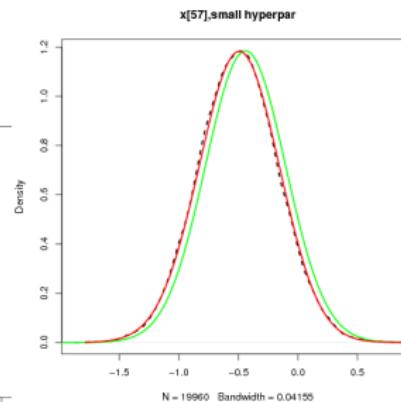
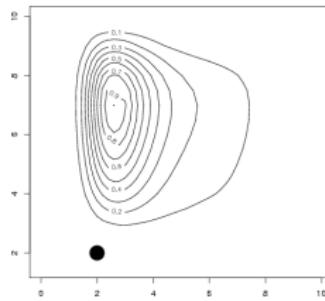
Marginals for $\theta|y$



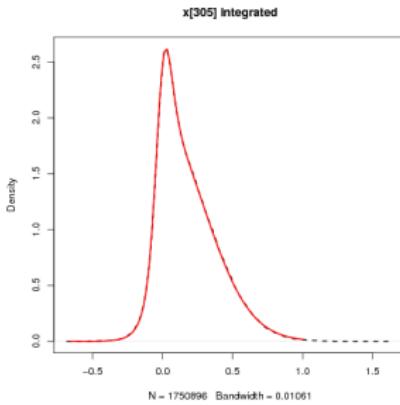
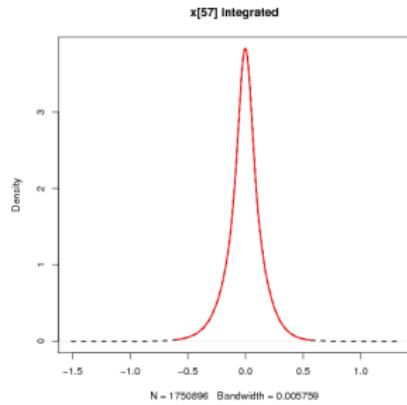
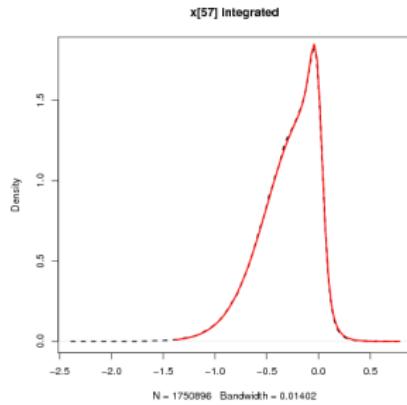
Marginals for $\theta|y$



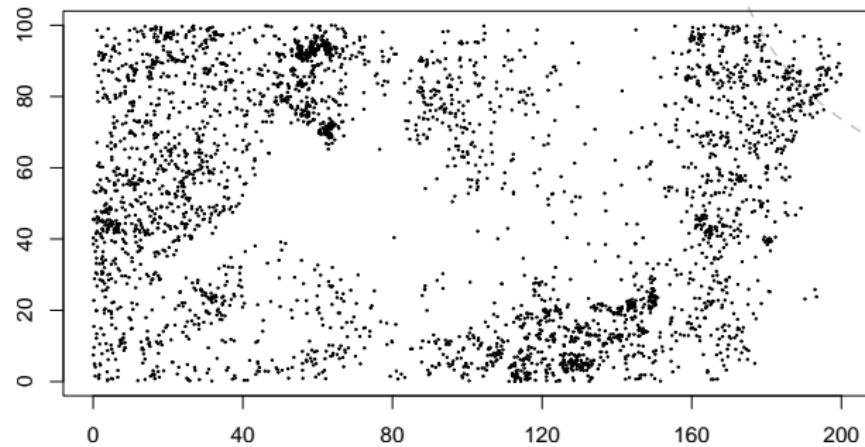
Marginals for $x_i|\theta, y$



Marginals for $x_i|y$

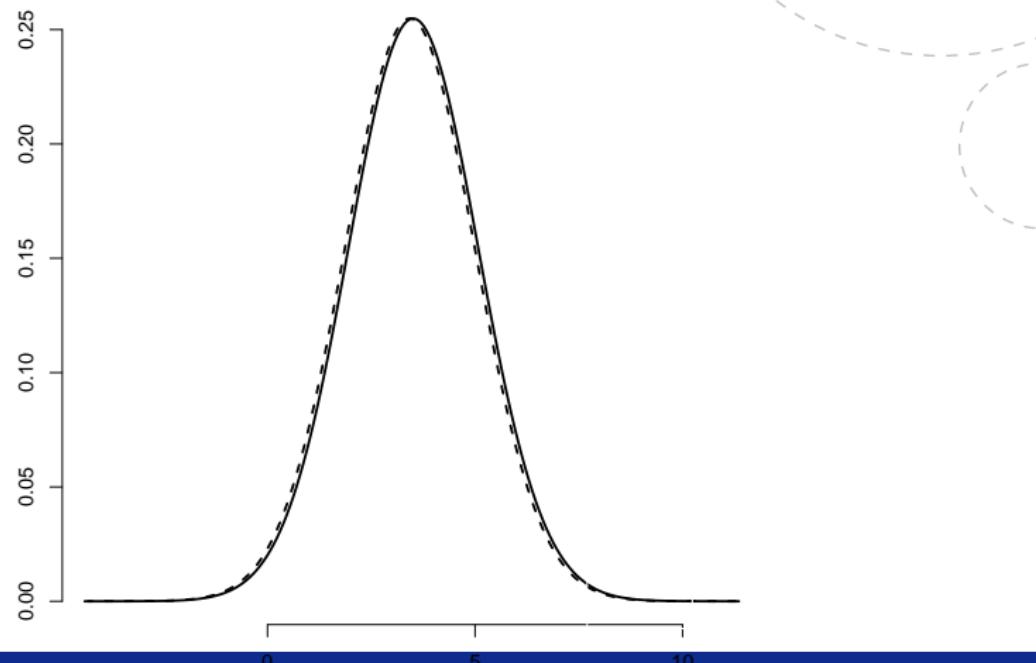


Log-Gaussian Cox process

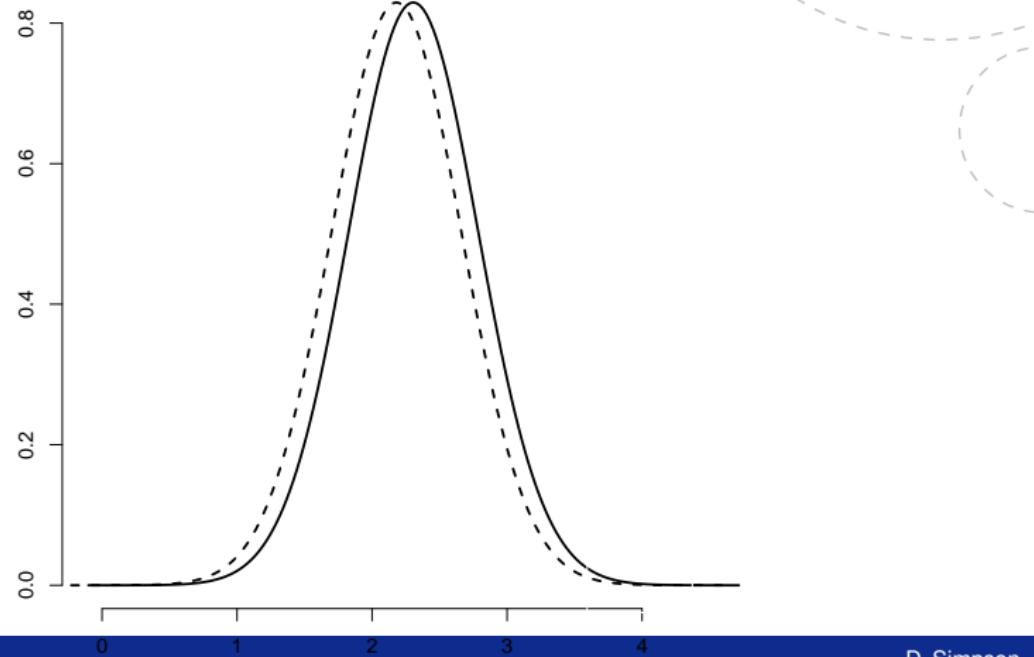


Locations of trees of a particular type: Data comes from a 50-hectare permanent tree plot which was established in 1980 in the tropical moist forest of Barro Colorado Island in Gatun Lake in central Panama.

Log-Gaussian Cox process



Log-Gaussian Cox process



Model

Model for log-density at each “pixel” in a 200×100 lattice

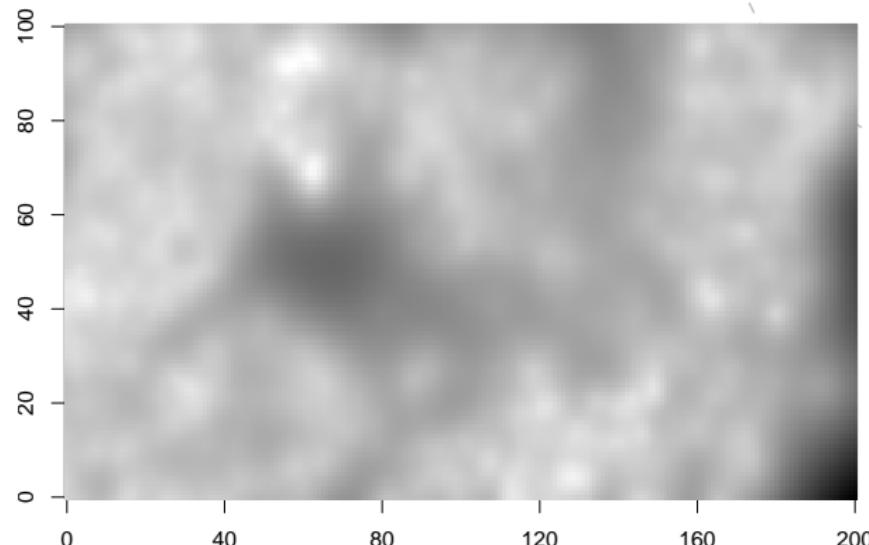
$$\eta_i = \beta_0 + \beta_1 c_{1i} + \beta_2 c_{2i} + u_i + v_i, \quad \sum_i u_i = 0$$

The spatial term is an IGMRF

$$\mathbb{E}(u_i | \mathbf{u}_{-i}) = \frac{1}{20} \left(8 \begin{array}{cccccc} \circ & \circ & \circ & \circ & \circ \\ \circ & \bullet & \circ & \circ & \circ \\ \circ & \circ & \bullet & \circ & \circ \\ \circ & \circ & \circ & \bullet & \circ \\ \circ & \circ & \circ & \circ & \circ \end{array} - 2 \begin{array}{cccccc} \circ & \circ & \circ & \circ & \circ \\ \circ & \bullet & \circ & \bullet & \circ \\ \circ & \circ & \circ & \circ & \bullet \\ \circ & \bullet & \circ & \circ & \bullet \\ \circ & \circ & \circ & \circ & \circ \end{array} - 1 \begin{array}{cccccc} \circ & \circ & \bullet & \circ & \circ \\ \circ & \circ & \circ & \bullet & \circ \\ \circ & \bullet & \circ & \circ & \bullet \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \bullet & \circ & \circ \end{array} \right)$$

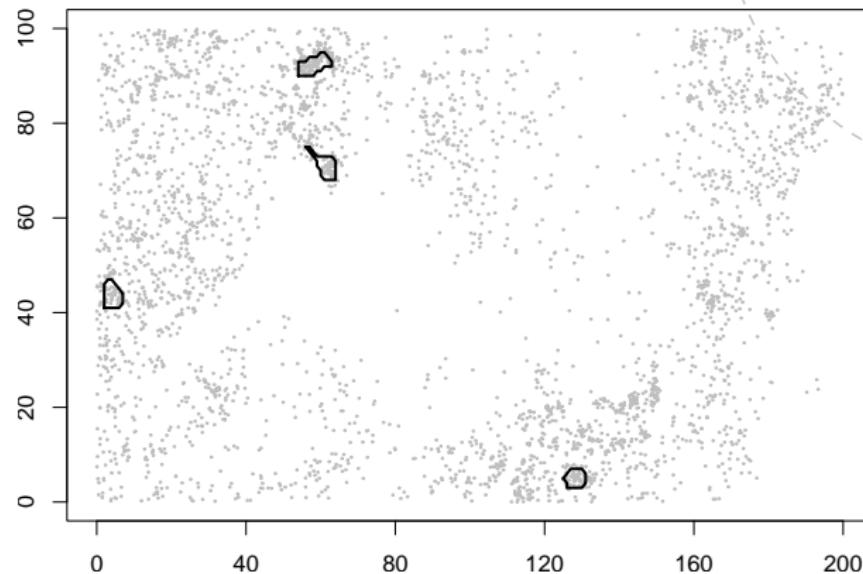
$$\text{Prec}(u_i | \mathbf{u}_{-i}) = 20\kappa_u$$

Results



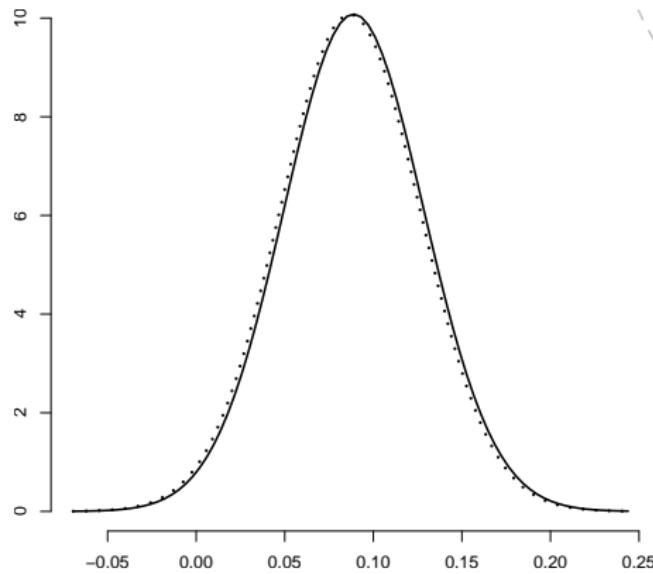
The posterior expectation of the spatial field

Results



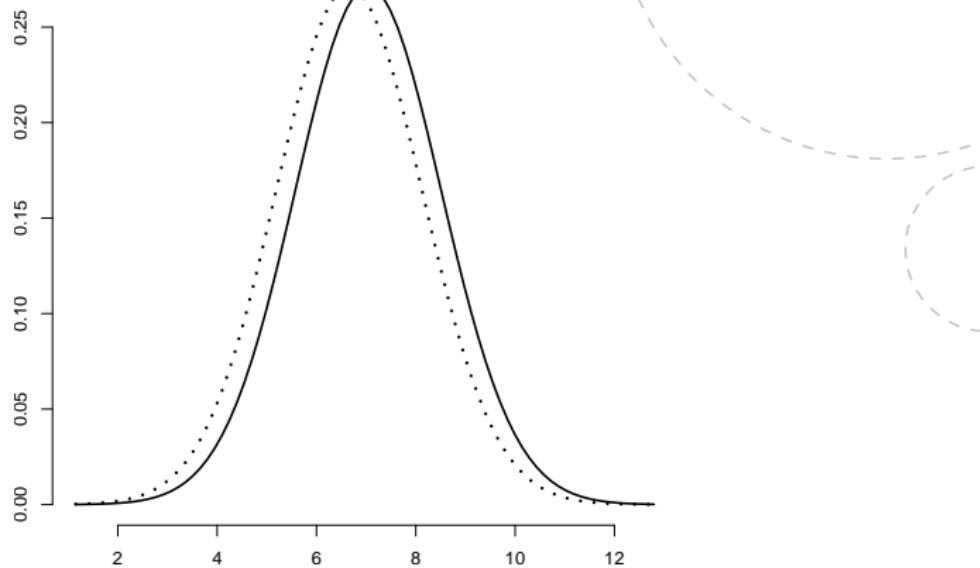
Locations with high KLD

Results



Effect of altitude

Results



Effect of norm of the gradient

Tokyo driver: Redux!

```
require(INLA)
data(Tokyo)

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

## Once more: the call to inla in verbose mode
result = inla(formula, family="binomial", Ntrials=n, data=Tokyo, verbose = TRUE)

## Summarise the results
summary(result)

## Plot the results
plot(result)

## Improve estimate of the hyperparameters
h = inla.hyperpar(result)

## Summarise improved estimates
summary(h)

## Plot improved estimates
plot(h)
```

Tokyo driver: Summary

```
> summary(h)

Call:
c("inla(formula = formula, family = \"binomial\")", data = Tokyo, Ntrials = n, ", "
  "verbose = TRUE)")

Time used:
Pre-processing    Running inla Post-processing          Total
0.1194000       0.4812071      0.2001090      0.8007162

The model has no fixed effects

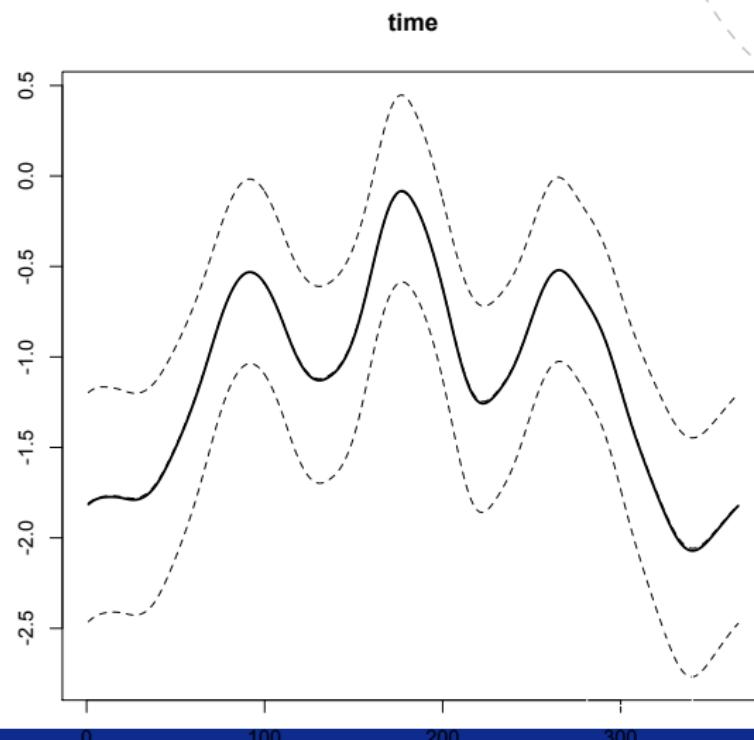
Random effects:
Name    Model    Max KLD
time   RW2 model

Model hyperparameters:
           mean      sd    0.025quant 0.5quant 0.975quant
Precision for time 13228.05 8484.76 3148.07 11122.37 35549.25

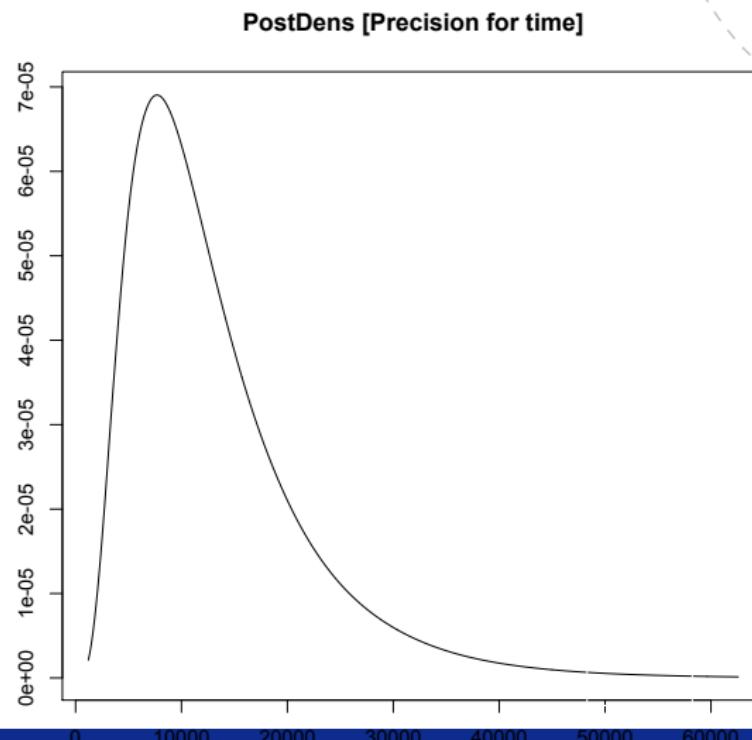
Expected number of effective parameters(std dev): 9.787(1.204)
Number of equivalent replicates : 37.39

Marginal Likelihood: -331.29
Warning: Interpret the marginal likelihood with care if the prior model is improper.
```

Posterior for temporal effect



Posterior for precision



Outline

Introduction

Latent Gaussian Models

Gaussian Markov random fields

Gaussian Markov random fields

Computing with GMRFs

A case for approximate inference

INLA

Intrinsic GMRFs

Intrinsic GMRFs (IGMRF)

- IGMRFs are *improper*, i.e., they have precision matrices not of full rank.
- Often used as prior distributions in various applications.
- Of particular importance are IGMRFs that are invariant to any trend that is a polynomial of the locations of the nodes up to a specific order.

Example: Smoothing

Say we want to fit a smooth trend. One way of specifying smoothness is to ensure that the value of the random field at each point is close to its neighbours' values.

$$x_i - \sum_{j \in \mathcal{N}_i} x_j \sim N(0, \sigma^2)$$

This is known as the first order *Simultaneous Autoregression* or SAR(1) model.

Convince yourself that \mathbf{Q} is singular.

Improper GMRF

Informal Definition

An *Improper GMRF* is a GMRF where the precision matrix \mathbf{Q} is singular (i.e. it has at least one zero eigenvalue)

- An improper GMRF is always proper on an appropriate subspace.
- The eigenvectors corresponding to the zero eigenvalues define the directions that the GMRF ‘doesn’t care about’.
- *We can use this indifference!*

Markov properties

The Markov properties are interpreted as those obtained from the limit of a proper density.

Let the columns of \mathbf{A}^T span the null space of \mathbf{Q}

$$\mathbf{Q}(\gamma) = \mathbf{Q} + \gamma \mathbf{A}^T \mathbf{A}. \quad (5)$$

$\mathbf{Q}(\gamma)$ tends to the corresponding one in \mathbf{Q} as $\gamma \rightarrow 0$.

$$\mathbb{E}(x_i \mid \mathbf{x}_{-i}) = \mu_i - \frac{1}{Q_{ii}} \sum_{j \sim i} Q_{ij} (x_j - \mu_j)$$

is interpreted as $\gamma \rightarrow 0$.

IGMRF of first order

An intrinsic GMRF of first order is an improper GMRF of rank $n - 1$ where $\mathbf{Q}\mathbf{1} = \mathbf{0}$.

The condition $\mathbf{Q}\mathbf{1} = \mathbf{0}$ means that

$$\sum_j Q_{ij} = 0, \quad i = 1, \dots, n$$

Let $\mu = \mathbf{0}$ so

$$\mathbb{E}(x_i | \mathbf{x}_{-i}) = -\frac{1}{Q_{ii}} \sum_{j:j \sim i} Q_{ij} x_j \quad (6)$$

$$-\sum_{j:j \sim i} Q_{ij}/Q_{ii} = 1$$

- The conditional mean of x_i is a weighted mean of its neighbours.
- No shrinking towards an overall level.
- Many IGMRFs are constructed such that the *deviation* from the overall level is a smooth curve in time or a smooth surface in space.

Definition (Forward difference)

Define the first-order forward difference of a function $f(\cdot)$ as

$$\Delta f(z) = f(z + 1) - f(z).$$

Higher-order forward differences are defined recursively:

$$\Delta^k f(z) = \Delta \Delta^{k-1} f(z)$$

so

$$\Delta^2 f(z) = f(z + 2) - 2f(z + 1) + f(z) \tag{7}$$

and in general for $k = 1, 2, \dots,$

$$\Delta^k f(z) = (-1)^k \sum_{j=0}^k (-1)^j \binom{k}{j} f(z + j).$$

For a vector $\mathbf{z} = (z_1, z_2, \dots, z_n)^T$, $\Delta\mathbf{z}$ has elements

$$\Delta z_i = z_{i+1} - z_i, \quad i = 1, \dots, n-1$$

The forward difference of k th order as an approximation to the k th derivative of $f(z)$,

$$f'(z) = \lim_{h \rightarrow 0} \frac{f(z + h) - f(z)}{h}$$

IGMRFs of first order on the line

Location of the node i is i (think “time”).

Assume *independent increments*

$$\Delta x_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \kappa^{-1}), \quad i = 1, \dots, n-1, \quad \text{so} \quad (8)$$

$$x_j - x_i \sim \mathcal{N}(0, (j-i)\kappa^{-1}) \text{ for } i < j. \quad (9)$$

If the intersection between $\{i, \dots, j\}$ and $\{k, \dots, l\}$ is empty for $i < j$ and $k < l$, then

$$\text{Cov}(x_j - x_i, x_l - x_k) = 0. \quad (10)$$

Properties coincide with those of a *Wiener process*.

The density for \mathbf{x} is

$$\pi(\mathbf{x} \mid \kappa) \propto \kappa^{(n-1)/2} \exp\left(-\frac{\kappa}{2} \sum_{i=1}^{n-1} (\Delta x_i)^2\right) \quad (11)$$

$$= \kappa^{(n-1)/2} \exp\left(-\frac{\kappa}{2} \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2\right), \quad (12)$$

or

$$\pi(\mathbf{x}) \propto \kappa^{(n-1)/2} \exp\left(-\frac{\kappa}{2} \mathbf{x}^T \mathbf{R} \mathbf{x}\right) \quad (13)$$

The *structure matrix*

$$\mathbf{R} = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ -1 & 2 & -1 & & \\ \ddots & \ddots & \ddots & & \\ & -1 & 2 & -1 & \\ & -1 & 2 & -1 & \\ & -1 & 1 & & \end{pmatrix}. \quad (14)$$

- The $n - 1$ independent increments ensure that the rank of \mathbf{Q} is $n - 1$.
- Denote this model by CAR(1) or RW1(κ) or short RW1.

Full conditionals

$$x_i \mid \mathbf{x}_{-i}, \kappa \sim \mathcal{N}\left(\frac{1}{2}(x_{i-1} + x_{i+1}), 1/(2\kappa)\right), \quad 1 < i < n, \quad (15)$$

Alternative interpretation:

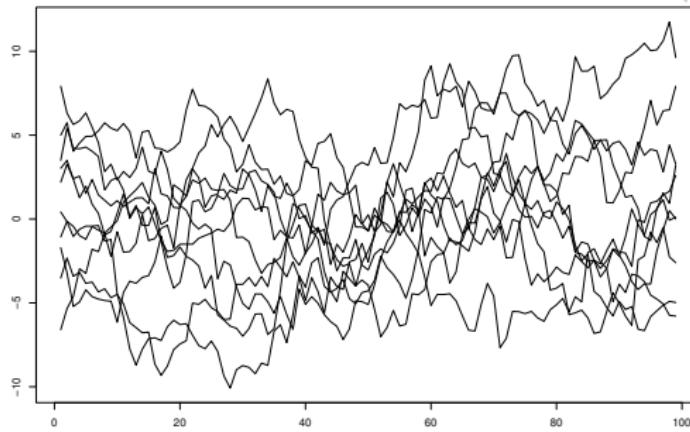
- fit a first order polynomial

$$p(j) = \beta_0 + \beta_1 j, \quad (16)$$

locally through the points $(i - 1, x_{i-1})$ and $(i + 1, x_{i+1})$.

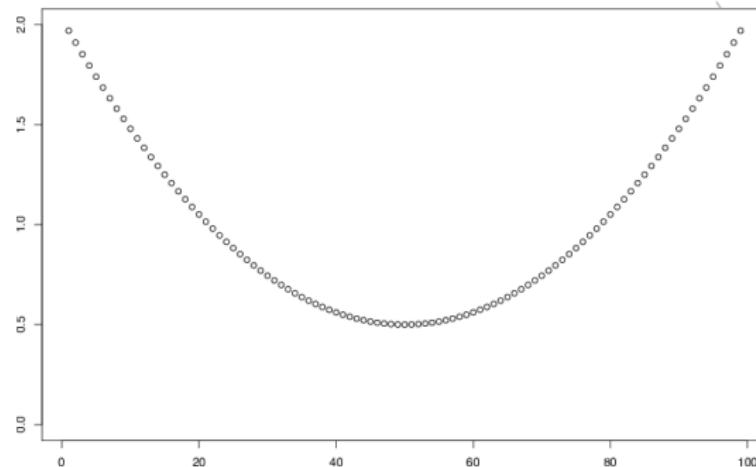
- The conditional mean is $p(i)$.

Example



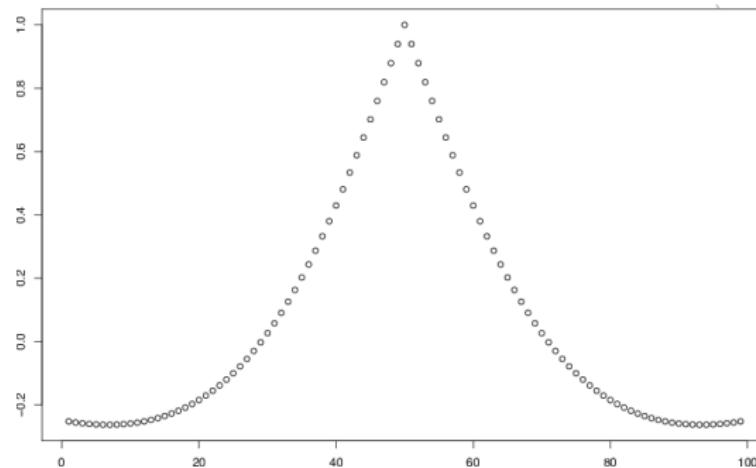
Samples with $n = 99$ and $\kappa = 1$ by conditioning on the constraint
 $\sum x_i = 0$.

Example



Marginal variances $\text{Var}(x_i)$ for $i = 1, \dots, n$.

Example



$\text{Corr}(x_{n/2}, x_i)$ for $i = 1, \dots, n$

CAR(1) on a regular lattice

It's the same as CAR(1) on a line!

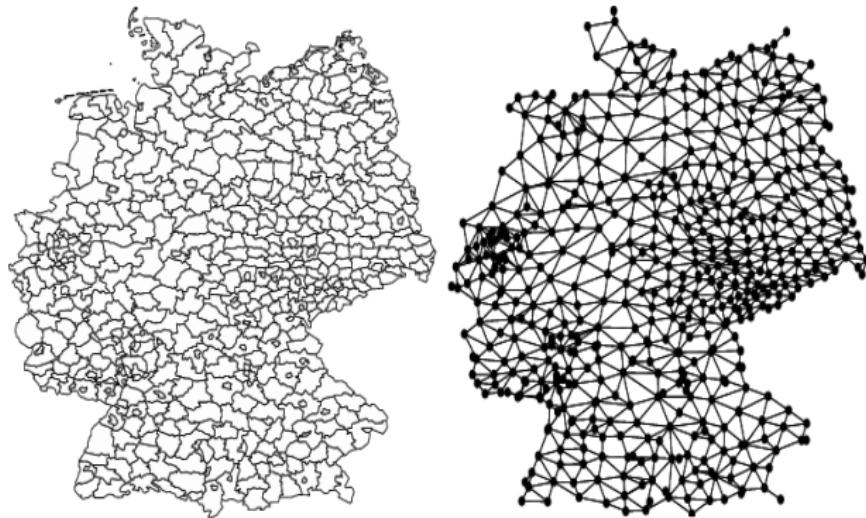
The differencing is now

$$\begin{matrix} & & -1 \\ -1 & 4 & -1 \\ & -1 \end{matrix}$$

Things don't tend to happen on regular lattices!

Consider the map of the 544 regions in Germany.

Two regions are *neighbours* if they share a common border.



Between neighbouring regions i and j , say, we define a “independent” Gaussian increment

$$x_i - x_j \sim \mathcal{N}(0, \kappa^{-1}) \quad (17)$$

Assume independent increments yields

$$\pi(\mathbf{x}) \propto \kappa^{(n-1)/2} \exp\left(-\frac{\kappa}{2} \sum_{i \sim j} (x_i - x_j)^2\right). \quad (18)$$

“ $i \sim j$ ” denotes the set of all *unordered* pairs of neighbours.

Number of increments $|i \sim j|$ is larger than n , but the rank of the corresponding precision matrix is still $n - 1$.

There are hidden constraints in the increments due to the more complicated geometry on a lattice than on the line.

Example

Let $n = 3$ where all nodes are neighbours. Then $x_1 - x_2 = \epsilon_1$, $x_2 - x_3 = \epsilon_2$, and $x_3 - x_1 = \epsilon_3$, where ϵ_1 , ϵ_2 and ϵ_3 are the increments.

This implies that

$$\epsilon_1 + \epsilon_2 + \epsilon_3 = 0$$

which is the ‘hidden’ linear constraint.

Let n_i denote the number of neighbours of region i .
The precision matrix \mathbf{Q} is

$$Q_{ij} = \kappa \begin{cases} n_i & i = j, \\ -1 & i \sim j, \\ 0 & \text{otherwise,} \end{cases} \quad (19)$$

$$x_i \mid \mathbf{x}_{-i}, \kappa \sim \mathcal{N}\left(\frac{1}{n_i} \sum_{j:j \sim i} x_j, \frac{1}{n_i \kappa}\right). \quad (20)$$

Example



Surely this is too easy...

- In general there is no longer an underlying continuous stochastic process that we can relate to this density.
- If we change the spatial resolution or split a region into two new ones, we change the model.

This is bad!

First order IGMRFs on regular lattices

For a lattice \mathcal{I}_N with $n = n_1 n_2$ nodes, let $i = (i_1, i_2)$ denote the node in the i_1 th row and i_2 th column.

Use the nearest four sites of i as its neighbours

$$(i_1 + 1, i_2), (i_1 - 1, i_2), (i_1, i_2 + 1), (i_1, i_2 - 1).$$

The precision matrix is

$$Q_{ij} = \kappa \begin{cases} n_i & i = j, \\ -1 & i \sim j, \\ 0 & \text{otherwise,} \end{cases} \quad (21)$$

and the full conditionals for x_i are

$$x_i | \mathbf{x}_{-i}, \kappa \sim \mathcal{N}\left(\frac{1}{n_i} \sum_{j:j \sim i} x_j, \frac{1}{n_i \kappa}\right). \quad (22)$$

Limiting behaviour

This model have an important property

*The process converge to the **de Wijs**-process: a Gaussian process with variogram*

$\log(\text{distance})$

This is important

- there is a limiting process
- The de Wijs process has dense precision matrix whereas the IGMRF is very sparse!

The RW2 model for regular locations

Let $s_i = i$ for $i = 1, \dots, n$, with a constant distance between consecutive nodes.

Use the second order increments

$$\Delta^2 x_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \kappa^{-1}) \quad (23)$$

for $i = 1, \dots, n - 2$,

to define the joint density of \mathbf{x}

$$\pi(\mathbf{x}) \propto \kappa^{(n-2)/2} \exp\left(-\frac{\kappa}{2} \sum_{i=1}^{n-2} (x_i - 2x_{i+1} + x_{i+2})^2\right) \quad (24)$$

$$= \kappa^{(n-2)/2} \exp\left(-\frac{\kappa}{2} \mathbf{x}^T \mathbf{R} \mathbf{x}\right) \quad (25)$$

$$R = \begin{pmatrix} 1 & -2 & 1 & & & & \\ -2 & 5 & -4 & 1 & & & \\ 1 & -4 & 6 & -4 & 1 & & \\ & 1 & -4 & 6 & -4 & 1 & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 6 & -4 & 1 \\ & & & 1 & -4 & 6 & -4 & 1 \\ & & & & 1 & -4 & 5 & -2 \\ & & & & & 1 & -2 & 1 \end{pmatrix}. \quad (26)$$

- Verify directly that $QS_1 = \mathbf{0}$ and that the rank of Q is $n - 2$.
- IGMRF of second order: invariant to the adding line to x .
- Known as the second order random walk model, denoted by $RW2(\kappa)$ or simply RW2 model.

Remarks

- This is the RW2 model defined and used in the literature.
- We can extend it consistently (numerically) to the case where the locations are irregular.
- There exists an exact and somewhat more involved formulation for arbitrary locations. (Markov)

The conditional mean and precision is

$$\mathbb{E}(x_i \mid \mathbf{x}_{-i}, \kappa) = \frac{4}{6}(x_{i+1} + x_{i-1}) - \frac{1}{6}(x_{i+2} + x_{i-2}), \quad (27)$$

$$\text{Prec}(x_i \mid \mathbf{x}_{-i}, \kappa) = 6\kappa, \quad (28)$$

respectively for $2 < i < n - 2$.

Consider the second order polynomial

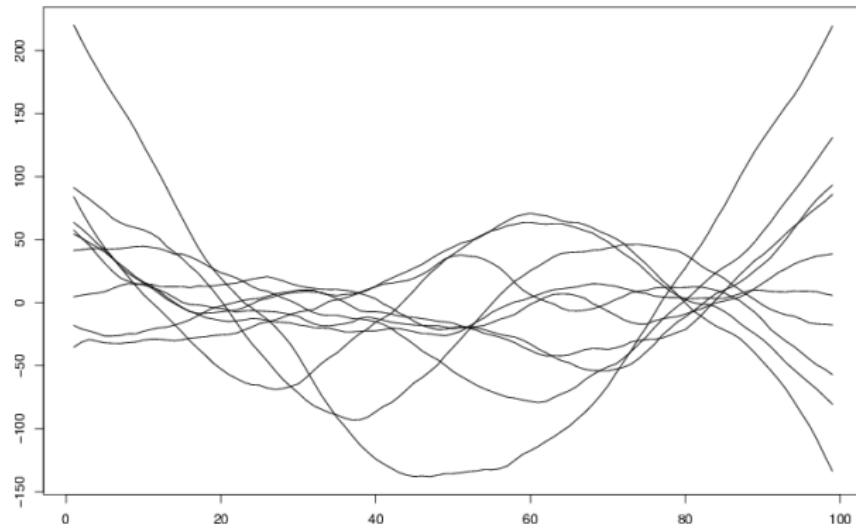
$$p(j) = \beta_0 + \beta_1 j + \frac{1}{2} \beta_2 j^2 \quad (29)$$

and compute the coefficients by a local least squares fit to the points

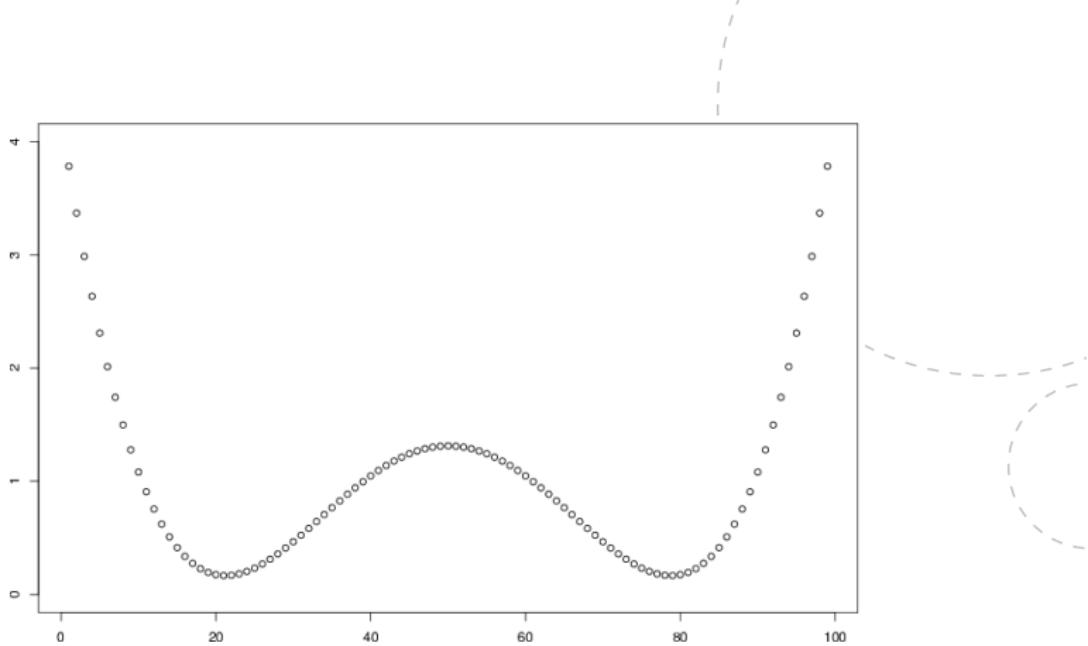
$$(i-2, x_{i-2}), (i-1, x_{i-1}), (i+1, x_{i+1}), (i+2, x_{i+2}). \quad (30)$$

Turns out that

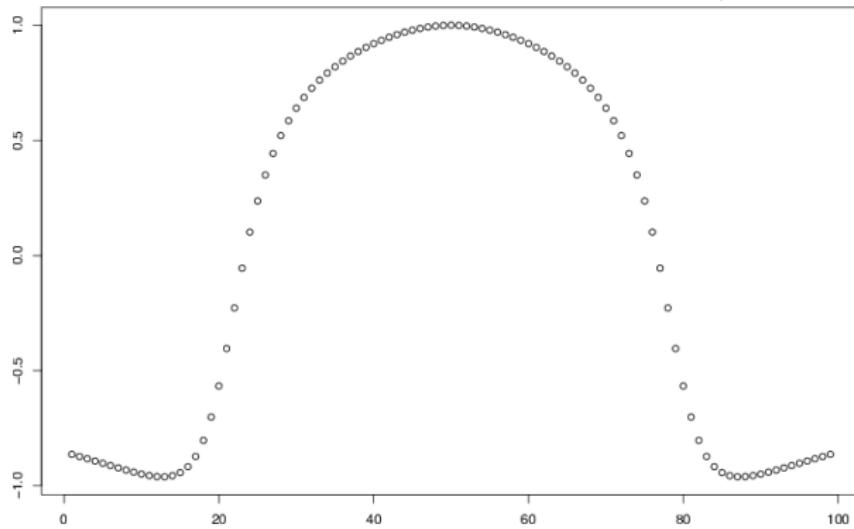
$$\mathbb{E}(x_i | \mathbf{x}_{-i}, \kappa) = p(i)$$



Samples with $n = 99$ and $\kappa = 1$ by conditioning on the constraints.



Marginal variances $\text{Var}(x_i)$ for $i = 1, \dots, n$.



$\text{Corr}(x_{n/2}, x_i)$ for $i = 1, \dots, n$

IGMRFs of higher order on regular lattices

The precision matrix is orthogonal to a polynomial design matrix of a certain degree.

Definition (IGMRFs of order k in dimension d)

An IGMRF of order k in dimension d , is an improper GMRF of rank $n - m_{k-1,d}$ where $\mathbf{QS}_{k-1,d} = \mathbf{0}$.

A second order IGMRF in two dimensions

Consider a regular lattice \mathcal{I}_N in $d = 2$ dimensions where $s_{i_1} = i_1$ and $s_{i_2} = i_2$.

Choose the independent increments

$$(x_{(i_1+1, i_2)} + x_{(i_1-1, i_2)} + x_{(i_1, i_2+1)} + x_{(i_1, i_2-1)}) - 4x_{i_1, i_2} \quad (31)$$

The motivation for this choice is that (31) is

$$\left(\Delta_{i_1}^2 + \Delta_{i_2}^2\right) x_{i_1-1, i_2-1} \quad (32)$$

Invariant to adding a first order polynomial

$$p_{1,2}(i_1, i_2) = \beta_{00} + \beta_{10}i_1 + \beta_{01}i_2,$$

The precision matrix (apart from boundary effects) should have non-zero elements

$$-\left(\Delta_{i_1}^2 + \Delta_{i_2}^2\right)^2 = -\left(\Delta_{i_1}^4 + 2\Delta_{i_1}^2\Delta_{i_2}^2 + \Delta_{i_2}^4\right) \quad (33)$$

which is a negative difference approximation to the *biharmonic* differential operator

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)^2 = \frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial x^2\partial y^2} + \frac{\partial^4}{\partial y^4}. \quad (34)$$

The fundamental solution of the biharmonic equation

$$\left(\frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial x^2\partial y^2} + \frac{\partial^4}{\partial y^4}\right)\phi(x, y) = 0 \quad (35)$$

is the *thin plate spline*.

Full conditionals in the interior

$$\mathbb{E}(x_i | \mathbf{x}_{-i}) = \frac{1}{20} \left(8 \begin{array}{cccccc} \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \bullet & \circ & \circ \\ \circ & \bullet & \circ & \bullet & \circ \\ \circ & \circ & \bullet & \circ & \circ \\ \circ & \circ & \circ & \bullet & \circ \\ \circ & \circ & \circ & \circ & \circ \end{array} - 2 \begin{array}{cccccc} \circ & \circ & \circ & \circ & \circ \\ \circ & \bullet & \circ & \bullet & \circ \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \bullet & \circ & \bullet & \circ \\ \circ & \circ & \circ & \circ & \circ \end{array} - 1 \begin{array}{cccccc} \circ & \circ & \bullet & \circ & \circ \\ \circ & \bullet & \circ & \circ & \bullet \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \bullet & \circ & \circ & \bullet \\ \circ & \circ & \bullet & \circ & \circ \end{array} \right) \quad (36)$$

and

$$\text{Prec}(x_i | \mathbf{x}_{-i}) = 20\kappa. \quad (37)$$

Outline

Introduction

Latent Gaussian Models

Gaussian Markov random fields

Gaussian Markov random fields

Computing with GMRFs

A case for approximate inference

INLA

Intrinsic GMRFs

Moving beyond lattices

The general IGMRF construction we have seen so far has been of the form

The k th order increments are independently distributed.

Questions:

- How do we generalise ‘increments’ so they limit correctly?

Moving beyond lattices

The general IGMRF construction we have seen so far has been of the form

The k th order increments are independently distributed.

Questions:

- How do we generalise ‘increments’ so they limit correctly?
- How do we generalise ‘independent’ so it limits?

Moving beyond lattices

The general IGMRF construction we have seen so far has been of the form

The k th order increments are independently distributed.

Questions:

- How do we generalise ‘increments’ so they limit correctly?
- **How do we generalise ‘independent’ so it limits?**

WHITE NOISE!

White Noise

White noise is the derivative of Brownian motion.

White Noise

White noise is the derivative of Brownian motion.

A better definition of White Noise

Let $W(\cdot)$ to be the random function that takes *sets* as inputs that has the following properties

- For a set $A \subset \mathbb{R}$, let $W(A) \sim N(0, |A|)$.
- If A and B are disjoint, $W(A \cup B) = W(A) + W(B)$.
- If A and B are disjoint, $\text{Var}(W(A), W(B)) = 0$.

$W(\cdot)$ is *White Noise* on \mathbb{R} .

Sets of the same size have the same amount of randomness.

Yuck!

Why define white noise this way?

- It is correct!
- It is correct on *any** space: \mathbb{R}^d , the sphere, a potato,
- It's the version that allows us to do all of the stuff we need to do!

But is it still the derivative of Brownian motion?

Yes!

But is it still the derivative of Brownian motion?

Yes!

But the ‘derivative of Brownian motion’ does not make sense.

But is it still the derivative of Brownian motion?

Yes!

But the ‘derivative of Brownian motion’ does not make sense.

Brownian motion is the integral of white noise!

White noise integrals

Definition

Let $f(s)$ and $g(s)$ be ok functions. The white noise integral $W(f) \equiv \int_{\mathbb{R}} f(s) dW(s)$ satisfies

- $W(f)$ is normally distributed with mean zero.
- $\text{Var}(W(f), W(g)) = \int_{\mathbb{R}} f(s)g(s) ds.$

This works for any other space as well. Just change the domain of integration and the ‘ ds ’.

Exercise: Convince yourself $B(t) = \int_0^t dW(s)$ is a Brownian motion.

So now we've got the independent thing done...

How on earth do we define 'increments'??

So now we've got the independent thing done...

How on earth do we define 'increments'??

Serious principle

If you want something to make sense in the continuous limit,
define it that way!

The region-based models don't limit to anything because we didn't construct them to.

Don't build a car and get surprised when it doesn't fly!

Gaussian random fields (GRFs)

The standard model for continuous structured spatial effects is a Gaussian field $x(s)$.

- There is a mean function $\mu(s) = \mathbb{E}(x(s))$.
 - This is usually modelled using covariates or known spatial basis functions.
- There is a covariance function $c(s, t) = \mathbb{E}(x(s)x(t))$.
 - This is usually specified parametrically.

Computational considerations

For a generic covariance function $c(s, t)$, imagine that we are interested in the latent field only at a set of N points $\{s_i\}$ that includes all of the data points.

The latent field sampled at these points becomes

$$\mathbf{x} \sim N(\mathbf{0}, \boldsymbol{\Sigma}),$$

where $\Sigma_{ij} = c(s_i, s_j)$ is a *dense* $N \times N$ matrix.

Working with dense matrices

— Storage:

- $\mathcal{O}(N^2)$
- 2500 points for 20 years requires ~ 20 Gbytes

Working with dense matrices

- Storage:
 - $\mathcal{O}(N^2)$
 - 2500 points for 20 years requires ~ 20 Gbytes
- Computation:
 - Each sample, solve, or determinant costs $\mathcal{O}(N^3)$.
 - We always need quite a few of these (likelihood, VB, INLA)
 - If we use MCMC we need a gargantuan number!
 - Remember 1,000,000 samples give ~ 3 decimal places of accuracy.

Working with dense matrices

- Storage:
 - $\mathcal{O}(N^2)$
 - 2500 points for 20 years requires ~ 20 Gbytes
- Computation:
 - Each sample, solve, or determinant costs $\mathcal{O}(N^3)$.
 - We always need quite a few of these (likelihood, VB, INLA)
 - If we use MCMC we need a gargantuan number!
 - Remember 1,000,000 samples give ~ 3 decimal places of accuracy.

Clearly this won't work if N is large.

What if our matrices magically became sparse?

Sparse covariance matrices can be formed using covariance tapering or compactly supported covariance functions.

— Storage:

- $\mathcal{O}(N)$
- 2500 points for 20 years requires ~ 400 Kilobytes

What if our matrices magically became sparse?

Sparse covariance matrices can be formed using covariance tapering or compactly supported covariance functions.

- Storage:
 - $\mathcal{O}(N)$
 - 2500 points for 20 years requires ~ 400 Kilobytes
- Computation:
 - Each sample, solve, or determinant costs $\sim \mathcal{O}(N^{3/2})$.

Searching for a CAR a carpark full of GRFs

Peter Whittle and Julian Besag considered limiting random fields corresponding to CAR models.

GMRF	Covariance	
CAR(2)	$\propto \kappa \ s\ K_1(\kappa \ s\)$	Whittle (1954)
CAR(1)	$\frac{1}{2\pi} K_0(\kappa \ s\)$	Besag (1981)
ICAR(1)	$-\frac{1}{2} \log(\ s\)$	Besag & Mondal (2005)
CAR(k)	$\propto (\kappa \ s\)^\nu K_\nu(\kappa \ s\)$	Lindgren, Rue & Lindström (2011)

$$\nu = k - d/2.$$

Theory Slide!!

Which GRFs ‘are’ GMRFs?

Spectral representation: $x(s) = \int \exp(is \cdot k) \hat{x}(k) dk$

Power spectrum: $S_x(k) = \mathbb{E}(|\hat{x}(k)|^2)$.

Rozanov, 1977

$x(s)$ is Markov if and only if its power spectrum is a constant divided by a non-negative symmetric polynomial.

Lesson: If we want to get GMRFs, we need to check the power spectrum

The Matérn covariance functions

An incredibly popular family of covariance functions:

$$c(x, y) \propto (\kappa \|x - y\|)^{\nu} K_{\nu}(\kappa \|x - y\|).$$

- The parameter ν controls the smoothness of the field, while κ controls its range.

The Matérn covariance functions

An incredibly popular family of covariance functions:

$$c(x, y) \propto (\kappa \|x - y\|)^{\nu} K_{\nu}(\kappa \|x - y\|).$$

- The parameter ν controls the smoothness of the field, while κ controls its range.
- The spectrum of a Matérn field is $S_x(k) = (2\pi)^{-d}(\kappa^2 + \|k\|^2)^{-\alpha}$, where $\nu = \alpha - d/2 > 0$.

Matérn fields are Markov when $\alpha = \nu + d/2$ is an integer.

So how do we exploit this?

We need an unusual characterisation of the Matérn field:

Scary maths

It is the stationary solution to the SPDE

$$(\kappa^2 - \Delta)^{\frac{\nu+d/2}{2}} x(s) = W(s),$$

where $\Delta = \sum_{i=1}^d \frac{\partial^2}{\partial s_i^2}$ is the Laplacian and $W(s)$ is spatial white noise.

Quiet aside: What not to do

We can write down the solution (trust me)

$$x(s) = C_\eta \int_{\mathbb{R}^2} (\kappa \|s - t\|)^\eta K_\eta(\kappa \|s - t\|) dW(t),$$

where $\eta = (\nu - d/2)/2$.

- This suggests the use of convolution approaches (Higdon, '98)

$$x(s) \approx C_\eta \sum_{i=1}^n (\kappa \|s - t_i\|)^\eta K_\eta(\kappa \|s - t_i\|) \xi_i,$$

where ξ_i are i.i.d. Normals.

Quiet aside: What not to do

We can write down the solution (trust me)

$$x(s) = C_\eta \int_{\mathbb{R}^2} (\kappa \|s - t\|)^\eta K_\eta(\kappa \|s - t\|) dW(t),$$

where $\eta = (\nu - d/2)/2$.

- This suggests the use of convolution approaches (Higdon, '98)

$$x(s) \approx C_\eta \sum_{i=1}^n (\kappa \|s - t_i\|)^\eta K_\eta(\kappa \|s - t_i\|) \xi_i,$$

where ξ_i are i.i.d. Normals.

- **This does not work.** (S, Lindgren, Rue, '10, Bolin and Lindgren '10)

Unpacking the SPDE

We have

$$(\kappa^2 - \Delta)^{\frac{\alpha}{2}} x(s) = W(s),$$

where α is an integer.

- We know what to do with white noise (particularly integrals of it)
- That means that $(\kappa^2 - \Delta)^{\alpha/2}$ must be the continuous versions α th increments.
- When α is even, this isn't too scary (it's just derivatives)

Unpacking the SPDE

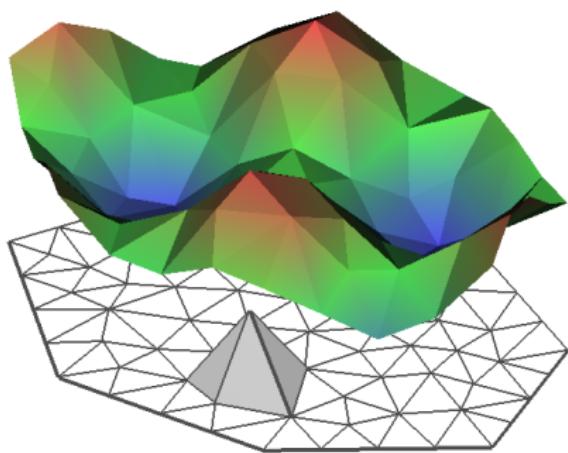
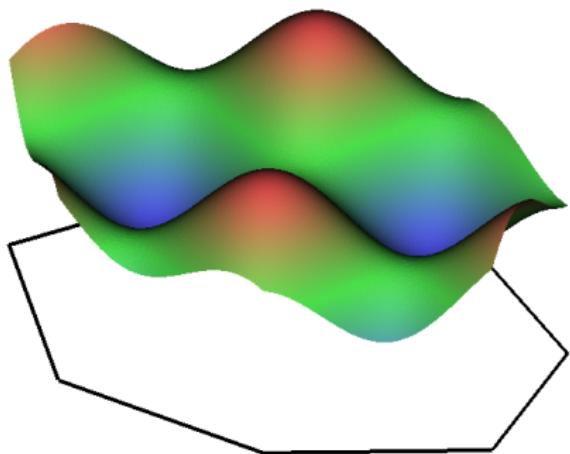
We have

$$(\kappa^2 - \Delta)^{\frac{\alpha}{2}} x(s) = W(s),$$

where α is an integer.

- We know what to do with white noise (particularly integrals of it)
- That means that $(\kappa^2 - \Delta)^{\alpha/2}$ must be the continuous versions α th increments.
- When α is even, this isn't too scary (it's just derivatives)
- When α is odd, we need to do a touch more work.

Approximation of surfaces



Key point: We are approximating the random field $x(s)$ by a piecewise continuous random field $x_n(s)$.

Stochastic weak representation

We are looking for the piecewise linear random field

$$x_n(s) = \sum_{i=1}^n w_i \phi_i(s)$$

that *best* approximates the solution to $(\kappa^2 - \Delta)x(s) = W(s)$.
 We get the system of linear equations

$$\int_{\Omega} \phi_j(s)(\kappa^2 - \Delta) \left(\sum_i w_i \phi_i(s) \right) ds \stackrel{D}{=} \int_{\Omega} \phi_j(s) dW(s)$$

Stochastic weak representation

We are looking for the piecewise linear random field

$$x_n(s) = \sum_{i=1}^n w_i \phi_i(s)$$

that *best* approximates the solution to $(\kappa^2 - \Delta)x(s) = W(s)$.
 We get the system of linear equations

$$\int_{\Omega} \phi_j(s) (\kappa^2 - \Delta) \left(\sum_i w_i \phi_i(s) \right) ds \stackrel{D}{=} \int_{\Omega} \phi_j(s) dW(s)$$

This is good— LHS has things we can compute, RHS has integrals of white noise.

What comes out?

We get two matrices:

- $\mathbf{C}_{ii} = \int_{\Omega} \phi_i(s) ds$ (the constant terms)
- $\mathbf{K}_{ij} = \int_{\Omega} \nabla \phi_i(s) \cdot \nabla \phi_j(s) ds$ (the Laplacian term)

What comes out?

We get two matrices:

- $\mathbf{C}_{ii} = \int_{\Omega} \phi_i(s) ds$ (the constant terms)
- $\mathbf{K}_{ij} = \int_{\Omega} \nabla \phi_i(s) \cdot \nabla \phi_j(s) ds$ (the Laplacian term)

The (scary) SPDE becomes the (normal) equation

$$(\kappa^2 \mathbf{C} + \mathbf{K}) \mathbf{w} \sim \mathcal{N}(0, \mathbf{C}^{-1})$$

and therefore \mathbf{w} is a GMRF with precision matrix

$$\mathbf{Q} = (\kappa^2 \mathbf{C} + \mathbf{K})^T \mathbf{C}^{-1} (\kappa^2 \mathbf{C} + \mathbf{K}).$$

A fake data example

```
require(INLA)
require(rgl) #For plotting!
require(boot) #for inverse logit

# Make some fake data
N=500
points = matrix(runif(2*N),ncol=2)

#The 'true' surface
fcn = function(loc){
  return( cos(loc[,1])*sin(4*loc[,2]))
}

noisy_data = rbinom(N,rep(1,N),inv.logit(fcn(points)))
```

A fake data example

```
## Build a mesh

bnd = inla.mesh.segment(matrix(c(0,0,1,0,1,1,1,0,1),ncol=2,byr

mesh = inla.mesh.create(points,boundary=bnd,refine=list(max

plot(mesh)

##Make the SPDE

spde = inla.spde.create(mesh,model="imatern")
```

A fake data example

```
## Fit the model
fake_data = list(y=noisy_data, data_points = mesh$idx$loc)

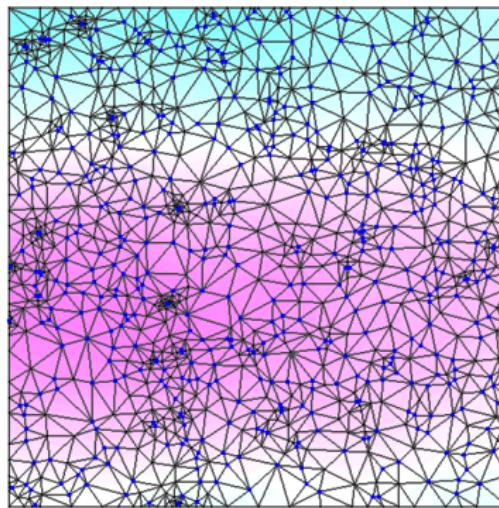
formula = y ~ f(data_points, model=spde)-1

r = inla(formula, family="binomial", Ntrials=rep(1,N), data=fake_data)

## And some output
summary(r)
plot(r, plot.random.effects=FALSE)
plot(old.mesh.class(mesh), r$summary.random$data_points$mean)

#Calculate the MSE
print(sqrt(var(r$summary.random$data_points$mean - fcn(mesh$idx$loc, mesh$beta))))
```

Posterior mean

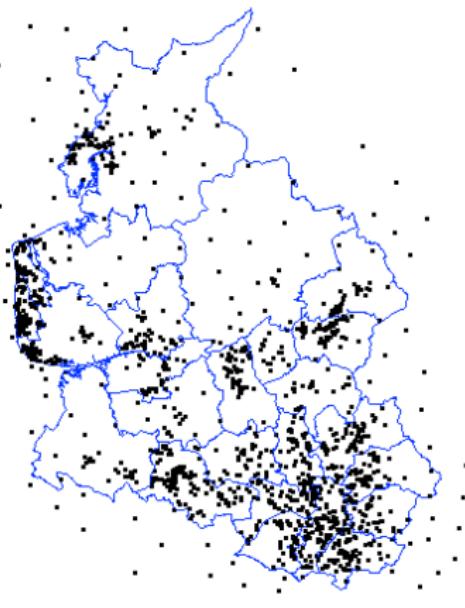


Leukaemia Survival in the UK

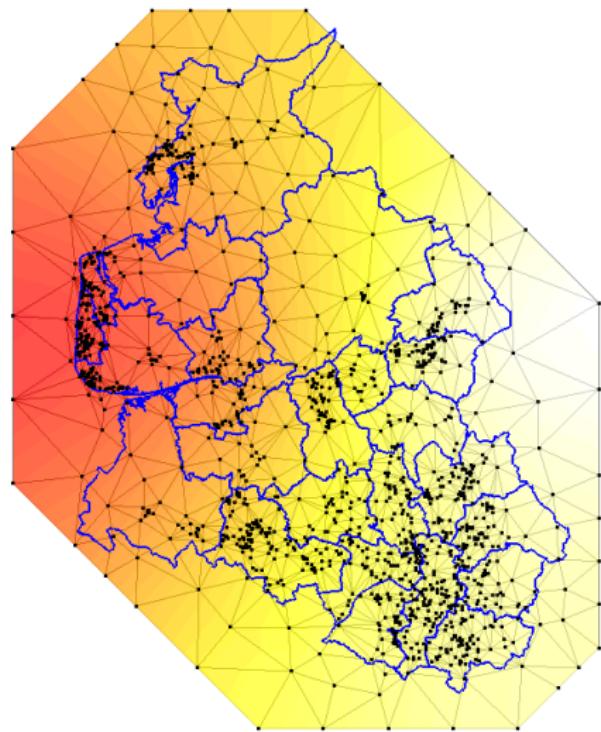
- Survival data at 1043 locations for acute myeloid leukaemia in adults diagnosed between 1982 and 1998 in Northwest England.
- Our model is

$$\log(\text{hazard}) = \log(\text{baseline}(\text{time})) + \\ \text{intercept} + \text{sex} + \text{age} + \text{wbc} + \text{tpi} + \text{spatial}(\text{location})$$

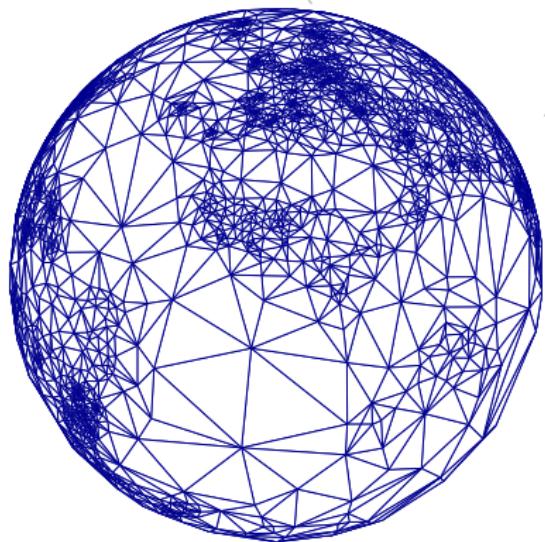
Data



Posterior Mean (Kriging estimate)



Manifolds/curved-spaces



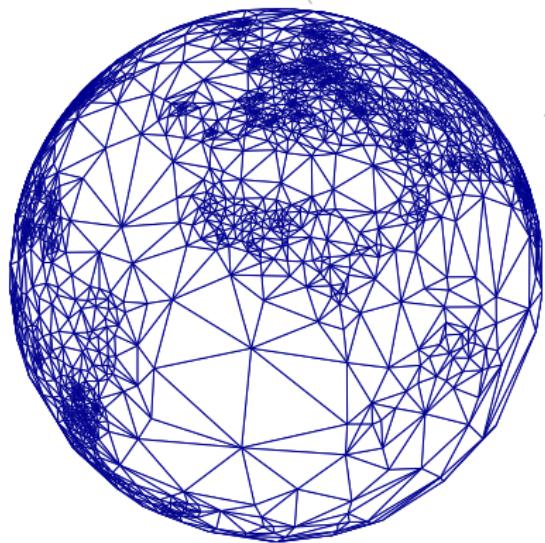
Manifolds/curved-spaces

Define Matérn fields using

$$(\kappa^2 - \Delta)^{\alpha/2} \mathbf{x}(\mathbf{s}) = \mathbf{W}(\mathbf{s})$$

on the manifold \mathcal{S} , driven by Gaussian “white noise” on \mathcal{S}

$$\text{Cov}(\epsilon(A_i), \epsilon(A_j)) = \int_{A_i \cap A_j} d\mathcal{S}(\mathbf{s})$$



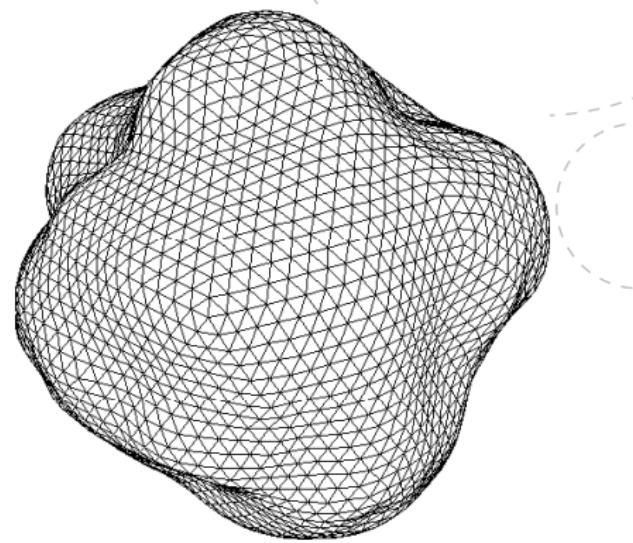
Manifolds/curved-spaces

Define Matérn fields using

$$(\kappa^2 - \Delta)^{\alpha/2} \mathbf{x}(\mathbf{s}) = \mathbf{W}(\mathbf{s})$$

on the manifold \mathcal{S} , driven by Gaussian “white noise” on \mathcal{S}

$$\text{Cov}(\epsilon(A_i), \epsilon(A_j)) = \int_{A_i \cap A_j} d\mathcal{S}(\mathbf{s})$$

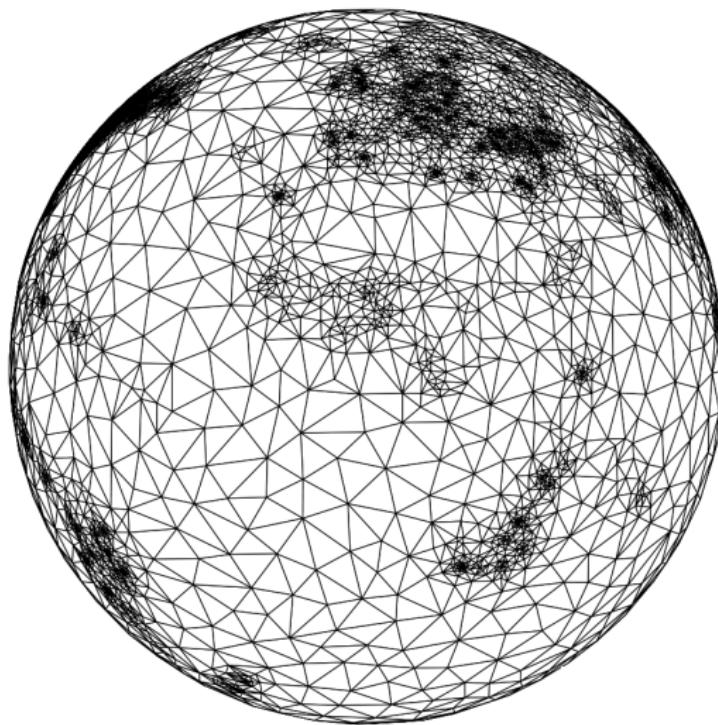


The advantage of SPDE approaches

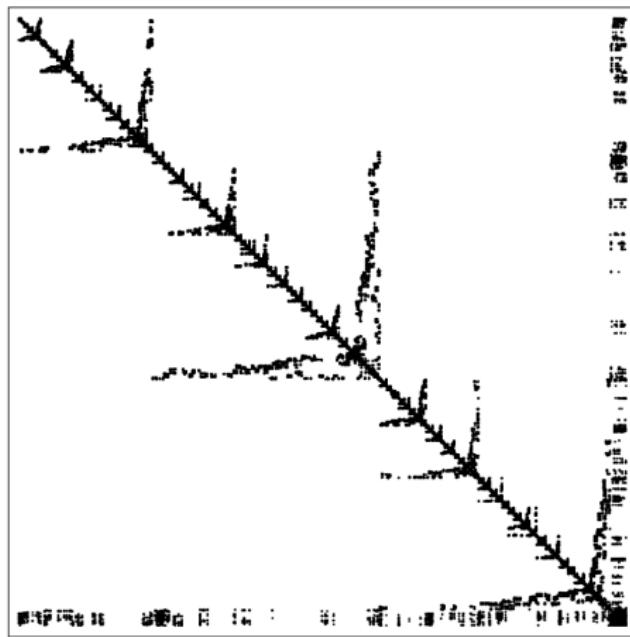
Everything stays the same

(just change ds to the surface measure)

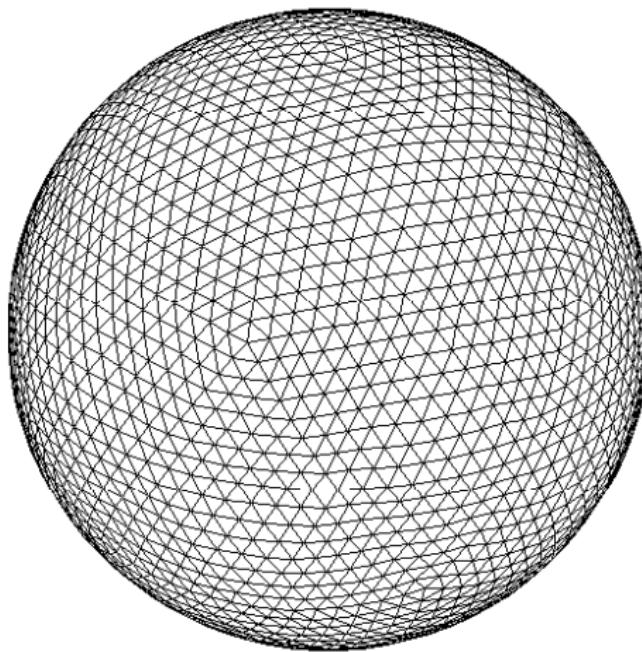
Example $\nu = 1$ and $\kappa^2 = 9$



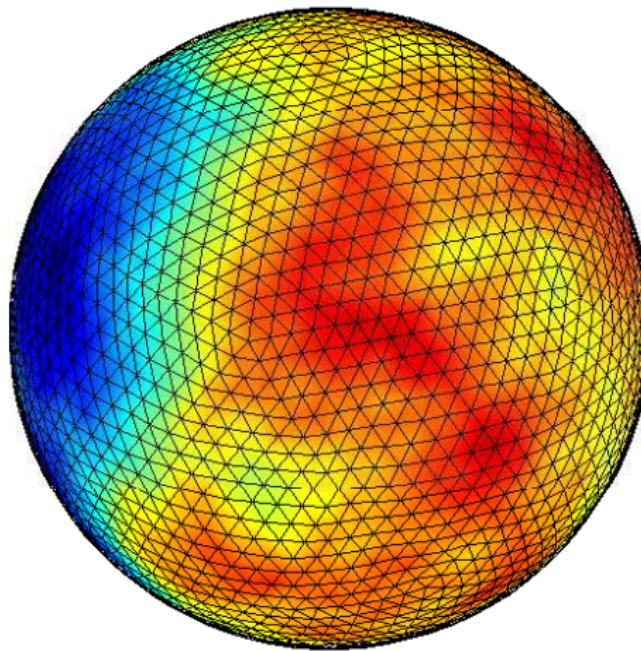
Example $\nu = 1$ and $\kappa^2 = 9$



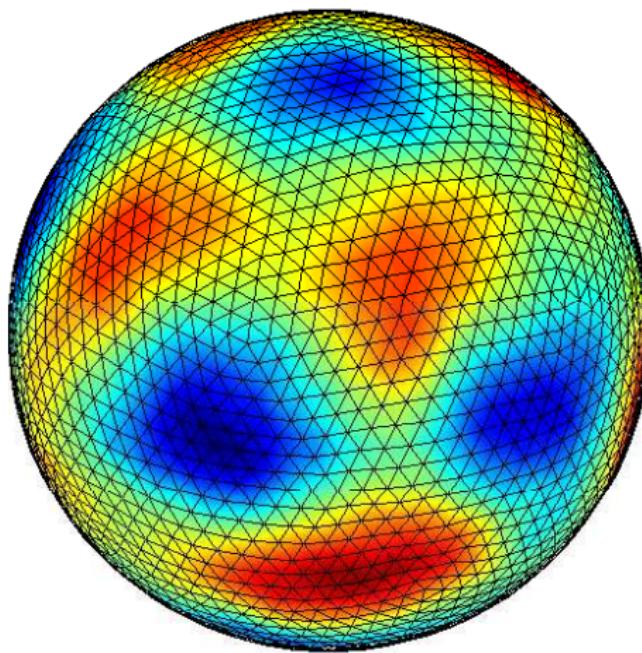
Example: Sphere



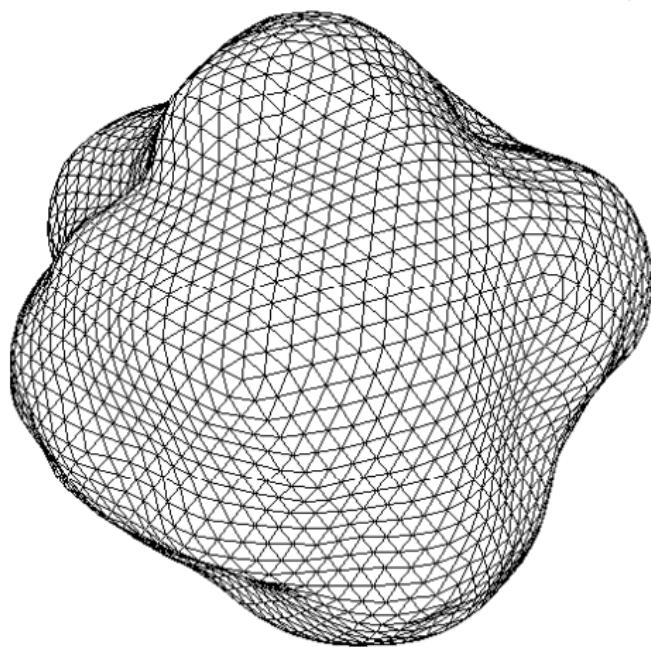
Example: $\kappa^2 = 2$, $\alpha = 2$



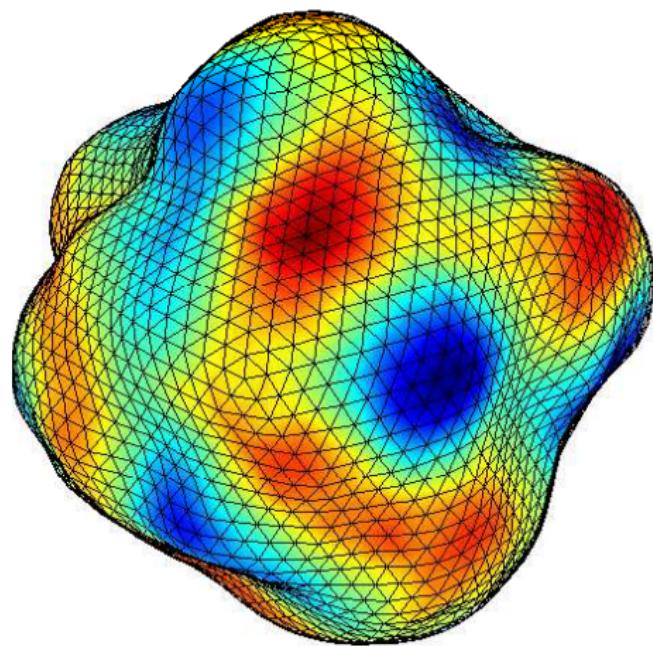
Example: $\kappa = -30 + 4i$, $\alpha = 2$



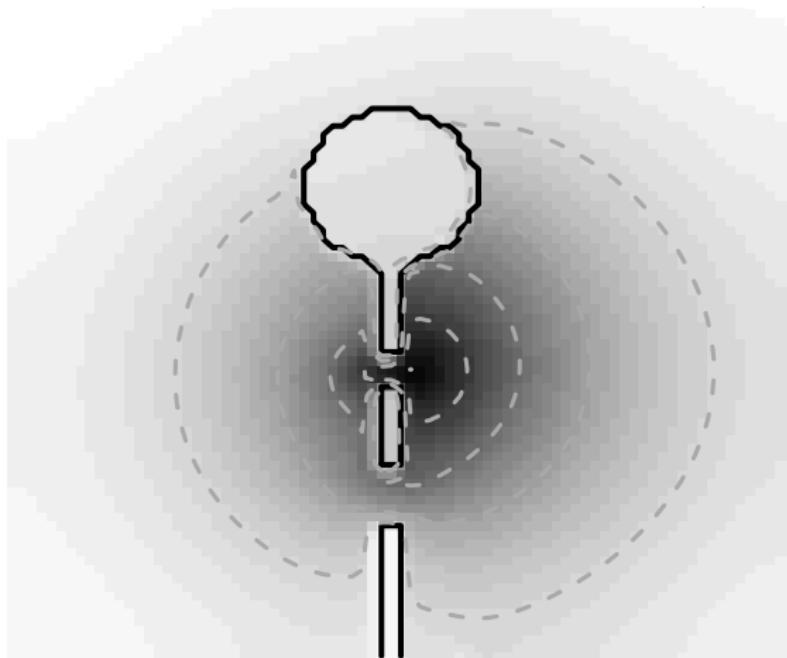
Example: Potato



Example: $\kappa = -30 + 4i$, $\alpha = 2$



Incorporating topography

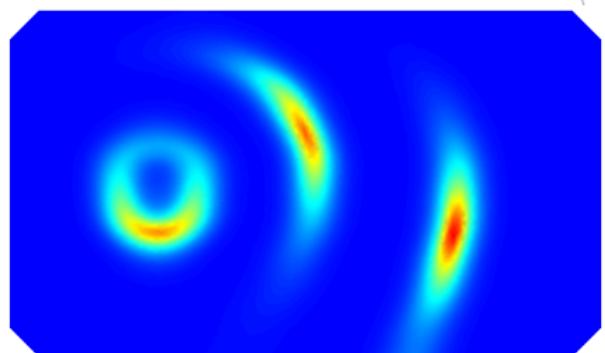
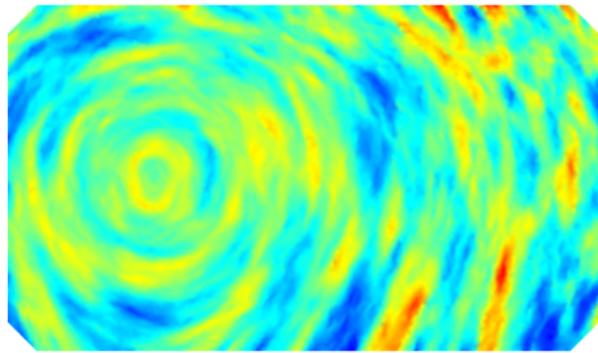


(S, Ottavi, Lingren and Rue, In prep.)

Non-stationary models

$$\kappa(s)\tau(s)x(s) + \nabla \cdot (\beta(s)x(s)) - \nabla \cdot (\mathbf{D}(s)\nabla(\tau(s)x(s))) = W(s)$$

(Lindgren, Rue and Lindström, 2011, and other work in. prep)



Spatiotemporal random fields

- Vector AR(1) process $x_t = \phi x_{t-1} + \epsilon_t$, where ϵ_t is generated by an SPDE.
- Space-time SPDE

$$\frac{\partial x}{\partial t} + (\kappa^2 - \Delta)^{\alpha/2} x(s, t) = W(s, t).$$

- Take space-time noise to be separable, white in time and Markov in space.
- Leads to a vector AR(1) process like

$$x_t = \mathbf{A}x_{t-1} + \epsilon_t,$$

where \mathbf{A} is constructed from the standard FE building blocks (**C** and **G**) and $\epsilon_t \approx \int_{t-\delta t}^t W(s, t) dt$.

Multivariate random fields

Solve a *system* of SPDEs

$$\begin{aligned} b_{11}(\kappa_{11}^2 - \Delta)^{\alpha_{11}/2}x_1(s) + b_{12}(\kappa_{12}^2 - \Delta)^{\alpha_{12}/2}x_2(s) &= f_1(s) \\ b_{22}(\kappa_{22}^2 - \Delta)^{\alpha_{22}/2}x_2(s) + b_{21}(\kappa_{21}^2 - \Delta)^{\alpha_{21}/2}x_1(s) &= f_2(s). \end{aligned}$$

- If we take $\alpha_{ij} \in 2\mathbb{N}$ and $f_i(s)$ to be Markovian noise, we get Markov random fields.
(Hu, S, Lingren, Rue, in prep.)

Outline

Introduction

Latent Gaussian Models

Gaussian Markov random fields

Gaussian Markov random fields

Computing with GMRFs

A case for approximate inference

INLA

Intrinsic GMRFs

Closing thoughts

We have covered a massive amount of material.

Key points:

- Be careful with latent Gaussian models!
- The Markov property is a wonderful thing.
- Inference needs to keep up with modelling. MCMC isn't enough (and neither is INLA)
- Continuous problems require continuous formulation.

Closing thoughts: Consistent GMRFs

- SPDEs: A continuous version of the precision matrix.
- We have necessarily gone beyond the usual comp-stats toolbox, but we're not too far into the woods.
- The approximation is theoretically, practically and numerically well motivated (no alarms and no surprises).
- The framework is incredibly flexible.
- You don't need to know how it works to use it! Everything's in r-INLA. (<http://www.r-inla.org>)

Give it a try and email Finn, Håvard or me if you hit trouble.

{finn.lindgren, havard.rue, daniel.simpson}@math.ntnu.no