

Bayesian Data Analysis

Daniel Paulin & Nicolò Margaritella

University of Edinburgh



Semester 2, 2020/2021

With thanks to Jonathan Gair, Ruben Amoros-Salvador, Ken Newman, Vanda Inácio and Natalia Bochkina for much of the material.

Scope

- ↪ The goal of this course is to provide practical experience of applying Bayesian analyses to a range of statistical models.
- ↪ The statistical analyses will be conducted using the widely used computer packages JAGS and R-INLA.
- ↪ Topics:
 - ↪ Brief overview of main Bayesian ideas.
 - ↪ Introduction to JAGS.
 - ↪ Introduction to INLA.
 - ↪ Linear and generalised linear models (fixed effects).
 - ↪ Hierarchical Bayesian models: linear and generalised linear models with random effects.
 - ↪ Further topics (which might include spatial and temporal models).

General information

- ↪ **Lecturers:** Daniel Paulin & Nicolò Margaritella
- ↪ **Lectures:** Recordings posted by Monday mornings on Learn during odd weeks.
- ↪ **Workshops:** Fridays even weeks on Zoom:
22 January, 5 February, 26 February, 12 March, 26 March.
3 available time slots: 9:00-11:00, 11:00 - 13:00, 14:00 - 16:00.
- ↪ **Additional information:**

Workshops will be conducted using www.kaggle.com (similar environment to Noteable). Please go to www.kaggle.com now, register, and open a new notebook. Select File/Language as R and File/Editor Type as Notebook. Click on File/Upload to add the Workshop problems from Learn.

The workshop problems will be available 2 weeks before the workshops on Learn.

At the end of each workshop, you must download your work from Kaggle, and upload it in Learn in Workshop Submissions.

General information

- ↪ **Office hours:** Tuesdays every week from 12:00-13:00 on Zoom.
Thursdays weeks 5-6 and 10-11 from 9:00-10:00 on Zoom.
- ↪ **Piazza:** The course has a Piazza page, feel free to post short questions of common interest there. For more complex questions that require some context to formulate, communication through Piazza can be very cumbersome. You should pose them during the workshops or office hours.
- ↪ **Email:** Other forms of communication such as discussions during office hours, workshops or on Piazza are preferred due to the large number of students enrolled in this course.

Assessment

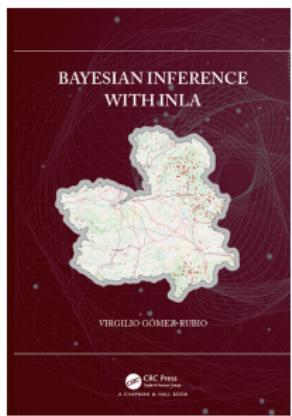
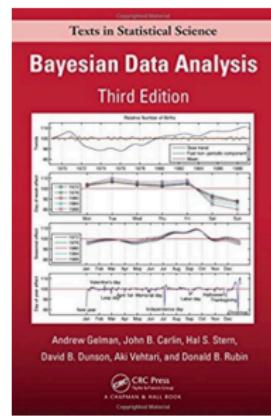
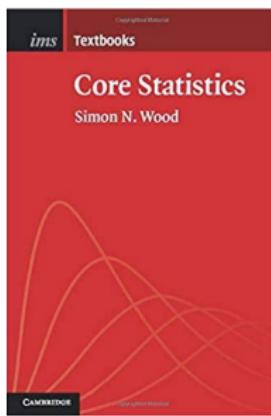
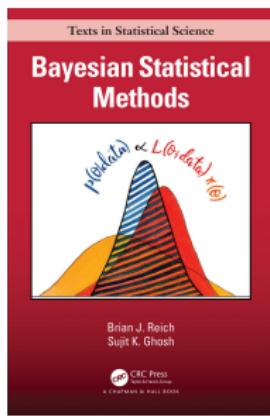
- ↪ 100% coursework: two homework assignments.
- ↪ Homework assignment 1 (50%) will be issued in week 5 (week beginning February 8). Solutions should be uploaded to Learn by 17:00 on Tuesday 2 March, 2021 (week 8).
- ↪ Homework assignment 2 (50%) will be issued in week 10 (week beginning March 22). Solutions should be uploaded to Learn by 17:00 on Monday 12 April, 2021 (week 13).
- ↪ Both homework assignments are to be done **individually** and consist of a sample of applied problems requiring the use of statistical software. A report containing all the analyses and conclusions should be delivered, along with your code that should run without error and fully reproduce all results in the report. Reports containing work that has not been done individually will be sent to the Academic Misconduct Officer.
- ↪ The homework assignments will be impossible to solve without a good understanding of the material in the lectures and workshops. **Regular study is essential.**
- ↪ **Students who discuss their solutions to the homework problems on Piazza will have their posts removed, and their access to the Piazza page revoked.**

Study plan (100 hours in total)

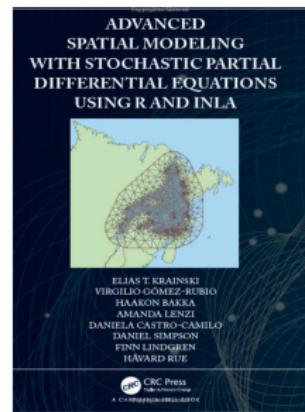
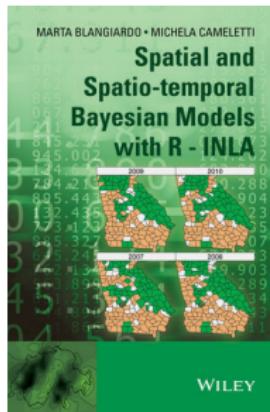
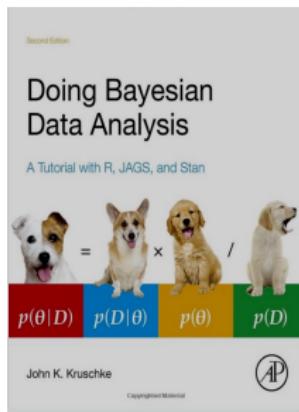
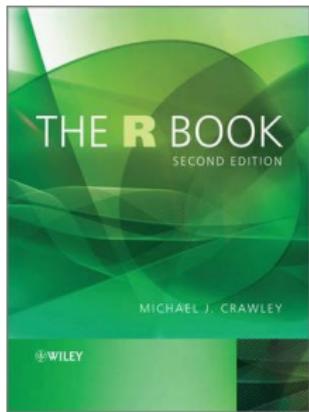
- ↪ Weeks 1 - 2 : 4 h study of lecture materials, 6 h work on Workshop 1 problems, 2 h participation in Workshop 1, 1 h on Piazza/office hours
- ↪ Weeks 3 - 4 : 4 h study of lecture materials, 6 h work on Workshop 2 problems, 2 h participation in Workshop 2, 1 h on Piazza/office hours
- ↪ Weeks 5 - 6 (3 weeks as include Flexible learning week) :
4 h study of lecture materials, 6 h work on Workshop 3 problems, 2 h participation in Workshop 3, **15 h work on Homework 1**, 1 h on Piazza/office hours
- ↪ Weeks 7 - 8 : 4 h study of lecture materials, 6 h working on Workshop 4 problems, 2 h participating in Workshop 4, 1 h on Piazza/office hours
- ↪ Weeks 9 - 12 : 8 h study of lecture materials (Lectures 5 and 6), 6 h work on Workshop 5 problems, 2 h participation in Workshop 5, **15 h work on Homework 2**, 2 h on Piazza/office hours

Recommended textbooks

- The course material (slides, worksheets, and other support material) contain all the information needed for the course.
- However, there are a wide variety of books, at all levels, that cover the material in this course that may be of interest to supplement the lecture material and/or provide additional examples.
- Suitable books include

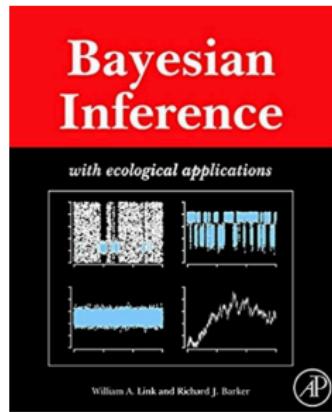
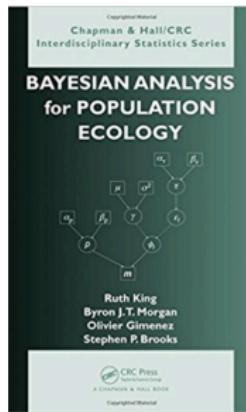
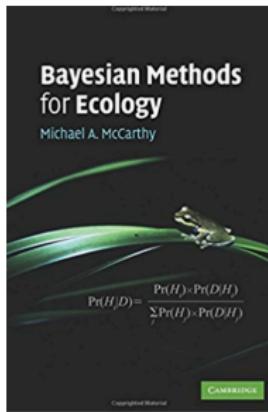


Additional textbooks



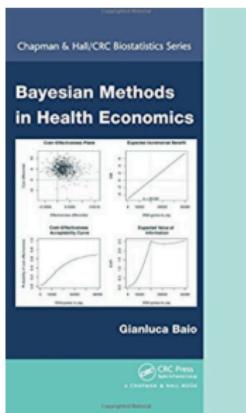
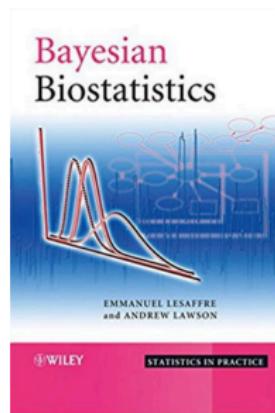
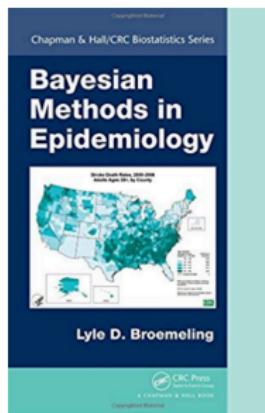
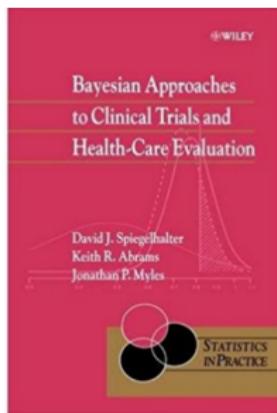
Additional textbooks

Ecological applications (covers from [amazon.co.uk](https://www.amazon.co.uk))



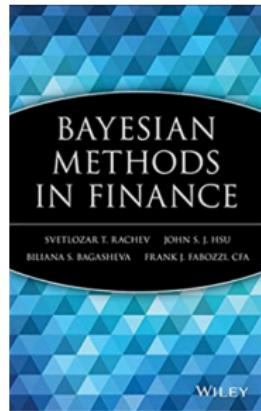
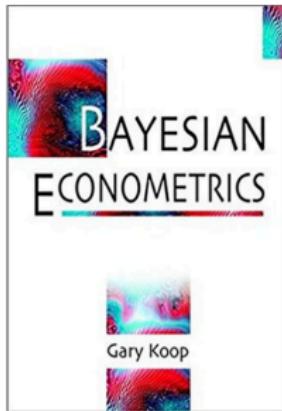
Additional textbooks

Medical/epidemiological applications (covers from amazon.co.uk)



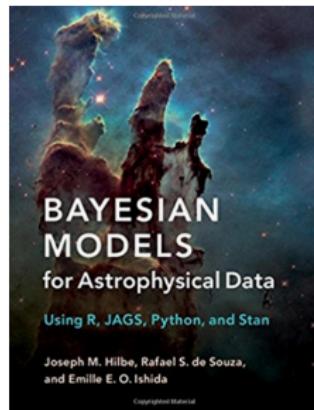
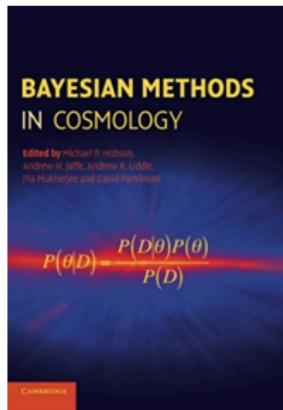
Additional textbooks

Finance/business/econometric applications (covers from [amazon.co.uk](https://www.amazon.co.uk))



Additional textbooks

Astrophysics/cosmology applications (covers from [amazon.co.uk](https://www.amazon.co.uk))



Outline

1 Review of Bayesian inference

- Choice of prior distribution
- Predictive inference

2 Bayesian computing

- Deterministic approximation methods
- Monte Carlo methods
- MCMC convergence diagnostics

Review of Bayesian inference

- ↪ Bayesian methods have been widely applied in many areas of science (e.g., medicine, finance, ecology, physics, psychology, etc).
- ↪ Motivations for adopting Bayesian approach vary:
 - ↪ natural and coherent way of thinking about science and learning,
 - ↪ pragmatic choice that is suitable for the problem in hand.
- ↪ Spiegelhalter et al. (2004) define a Bayesian approach as
 - 'the explicit use of external information in the design, monitoring, analysis, interpretation, and reporting of a [scientific investigation].'
- ↪ These authors argue that a Bayesian approach is
 - ↪ more flexible in adapting to each unique situation,
 - ↪ more efficient in using all available evidence,
 - ↪ more useful in providing relevant quantitative summaries,than traditional (frequentist) methods.

Review of Bayesian inference

Bayesian vs frequentist statistics

- ↪ The **frequentist** approach can be regarded as a procedure that quantifies uncertainties (p-value, confidence interval, etc) if the process that generated the data is repeated many times.
- ↪ Parameters are fixed and unknown, only the data is random.
- ↪ Aims to be objective.
- ↪ Both approaches have pros and cons. When both are applicable they are unlikely to give different answers.
- ↪ **Bayesian** represent their uncertainty about parameters with probability distributions and treat them as random variables.
- ↪ We can thus, under a Bayesian framework, make probability statements about model parameters.
- ↪ This is in contrast with the frequentist framework where probability statements only concern the data.

Review of Bayesian inference

Main components

- ↪ Suppose we have a parameter $\theta \in \Theta$ on which we wish to make inference.
- ↪ The main ‘ingredients’ of Bayesian inference are:
 - ↪ The prior distribution, $\pi(\theta)$, which represents the initial beliefs concerning the parameter prior to any data being observed.
 - ↪ The likelihood $f(\mathbf{y} | \theta)$, which plays a key role in all statistical inference, both Bayesian and frequentist. Represents the information contained in the data \mathbf{y} about the parameter θ .
 - ↪ The posterior distribution, $p(\theta | \mathbf{y})$, which updates the prior beliefs with respect to θ , following the data \mathbf{y} being observed.
- ↪ Bayesian learning combines past experience (prior) with new data (likelihood) in a mathematically coherent way (Bayes’ Theorem) to obtain the current state of knowledge (posterior).

Review of Bayesian inference

Bayes' theorem

↪ Bayes' theorem (continuous version) tells us

$$p(\theta | \mathbf{y}) = \frac{f(\mathbf{y} | \theta)\pi(\theta)}{m(\mathbf{y})},$$

where $m(\mathbf{y}) = \int_{\Theta} f(\mathbf{y} | \theta)\pi(\theta)d\theta$ is the evidence (the marginal distribution of the data).

↪ Since the marginal distribution does not depend on θ , we can write

$$p(\theta | \mathbf{y}) \propto f(\mathbf{y} | \theta)\pi(\theta),$$

i.e., the posterior distribution is proportional to the likelihood times the prior distribution.

Review of Bayesian inference

Summarising posterior distributions

- ↪ All inference about the parameter(s) of interest, θ , is (are) based on the posterior distribution.
- ↪ The information contained in the posterior distribution can be summarised in different ways as appropriate to the inference goal, e.g.
 - ↪ Means, standard deviations, medians.
 - ↪ Probability of exceeding a certain threshold, say θ_0 , $\Pr(\theta > \theta_0 | \mathbf{y})$.
 - ↪ Credibility intervals.

Review of Bayesian inference

Decision theory

- ↪ Quantities derived from the posterior can be used as estimators of the unknown parameter(s). But, which of the possible summary statistics is ‘best’?
- ↪ This is only a meaningful question if we have some notion of the ‘cost’ of making an error when estimating the value of the parameter. Decision theory provides such a framework by introducing the notion of a ‘loss function’.
- ↪ Let $L(\theta, a)$ be the loss associated with using a as the estimate, when the true value is θ . Note that for simplicity we only write a but, in fact, it is $a(\mathbf{y})$.
- ↪ The corresponding Bayes estimator is then chosen to minimise the expected loss with respect to the posterior distribution.
- ↪ Mathematically, the Bayes estimator, $\hat{\theta}$, is defined such that

$$\begin{aligned}\hat{\theta}(\mathbf{y}) &= \operatorname{argmin}_{a \in \Theta} \mathbb{E}_{\text{post.}} [L(\theta, a)] \\ &= \operatorname{argmin}_{a \in \Theta} \left[\int_{\theta \in \Theta} L(\theta, a) p(\theta | \mathbf{y}) d\theta \right].\end{aligned}$$

Review of Bayesian inference

Decision theory

- Three commonly used loss functions and corresponding Bayes estimators are:

Loss	Bayes estimate
$L(\theta, a) = (\theta - a)^2$ (quadratic loss)	$\hat{\theta} = \mathbb{E}_{\text{post.}}(\theta) = \int_{\theta \in \Theta} \theta p(\theta \mathbf{y}) d\theta$
$L(\theta, a) = \theta - a $ (absolute error loss)	$\hat{\theta} = \text{median}_{\text{post}}(\theta)$
$L(\theta, a) = I(\theta \neq a)$ (zero/one loss)	$\hat{\theta} = \arg \max_{\theta} p(\theta \mathbf{y})$

- Appropriate loss functions for hypothesis testing and interval estimation also exist.

Choice of prior distribution

- ↪ Picking the prior is obviously important and uniquely Bayesian. Priors must be specified on all the parameters that the statistician is interested in.
- ↪ Common types of priors include (these categories are not mutually exclusive):
 - ↪ Informative/expert priors
 - ↪ Non-informative/vague priors
 - ↪ Conjugate priors
- ↪ It is also important to try several priors in a sensitivity analysis.

Choice of prior distribution

Informative/expert priors

- ↪ A major advantage of the Bayesian approach is the ability to include expert prior information.
- ↪ This can be designed to reflect the subjective opinion of an expert in the field or by using past data (literature, pilot study, etc).
- ↪ The process of extracting prior knowledge in a suitable manner to permit the formulation of a prior distribution that represents the expert/historical information as accurately as possible is called **elicitation**.
- ↪ Spiegelhalter et al. (2004), Sections 5.2, 5.3, and 5.4, contain a good discussion on how priors might be elicited from experts or historical data.

Choice of prior distribution

Informative/expert priors

- ↪ What about if we ask more than one expert and they don't agree?
- ↪ Suppose we are interested in a quantity θ and that expert j recommends prior $\theta \sim N(\mu_j, \sigma_j)$.
- ↪ One approach is to weight the experts using a *mixture model*

$$\pi(\theta) = \sum_{j=1}^J \omega_j N(\theta | \mu_j, \sigma_j^2),$$

where ω_j is the weight given to expert j ($\sum_{j=1}^J \omega_j = 1$).

Choice of prior distribution

Non-informative/vague priors

- ↪ What should we do if we do not have any prior information concerning the parameter of interest?
- ↪ Bayes himself suggested that when this is the case, the Uniform prior should be used, so that $\pi(\theta) = c$, for all θ .

- ↪ In this case we clearly have

$$p(\theta | \mathbf{y}) \propto f(\mathbf{y} | \theta),$$

i.e., the posterior distribution has the same shape as the likelihood function.

- ↪ In this case the mode of the posterior distribution coincides with the MLE.
- ↪ If the parameter space Θ is unbounded, this prior will be improper for any choice of c , i.e., $\int_{\Theta} \pi(\theta) d\theta = \infty$. Improper priors can be used but we must check if the resulting posterior is proper (i.e., if it can be normalized to integrate to one).

Choice of prior distribution

Non-informative/vague priors

- ↪ As argued by Raiffa and Schlaiffer (1961), if one is ignorant about θ , one should also be ignorant about θ^2 , and one cannot find a distribution that is uniform on both θ and θ^2 .
- ↪ To see this, suppose that we place a Uniform prior on $\theta \in [0, 1]$, so that $\pi(\theta) = 1$. The corresponding prior on $\psi = \theta^2$ is $p(\psi) = \frac{1}{2\sqrt{\psi}}$, which is obviously non-uniform on ψ .

Choice of prior distribution

Non-informative/vague priors: Jeffreys' prior

→ Jeffreys (1961) proposed a class of priors that are invariant to transformations.

→ Jeffreys' prior is $\pi(\theta) \propto \sqrt{I(\theta)}$, where $I(\theta)$ is the expected Fisher information.

→ Remember that

$$I(\theta) = \mathbb{E} \left(\frac{d}{d\theta} \log f(\mathbf{Y} | \theta) \right)^2,$$

which in regular cases equals to $I(\theta) = -\mathbb{E} \left(\frac{d^2}{d\theta^2} \log f(\mathbf{Y} | \theta) \right)$.

→ For the case of a multivariate parameter vector, say $\boldsymbol{\theta} = (\theta_1, \dots, \theta_p)^T$, Jeffreys' prior is given by

$$\pi(\boldsymbol{\theta}) \propto \sqrt{\det I(\boldsymbol{\theta})},$$

with $I(\boldsymbol{\theta}) = -\mathbb{E} \left(\frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \log f(\mathbf{Y} | \boldsymbol{\theta}) \right)$.

Choice of prior distribution

Non-informative/vague priors: Jeffreys' prior

- ↪ We will now calculate the Jeffreys' prior for binomial data.
- ↪ Let us suppose that $Y \sim \text{Bin}(n, \theta)$, where Y denotes the number of 'successes' out of n trials, and where the probability of success is θ .
- ↪ The likelihood is

$$f(y | \theta) = \binom{n}{y} \theta^y (1 - \theta)^{n-y},$$

and the log likelihood is

$$\log f(y | \theta) = C + y \log \theta + (n - y) \log(1 - \theta).$$

- ↪ The first derivative is

$$\frac{d}{d\theta} \log f(y | \theta) = \frac{y}{\theta} - \frac{n - y}{1 - \theta},$$

and the second derivative is

$$\frac{d^2}{d\theta^2} \log f(y | \theta) = -\frac{y}{\theta^2} - \frac{n - y}{(1 - \theta)^2}.$$

Choice of prior distribution

Non-informative/vague priors: Jeffreys' prior

↪ Thus,

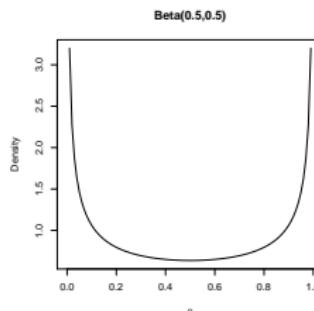
$$I(\theta) = -\mathbb{E} \left(\frac{d^2}{d\theta^2} \log f(Y | \theta) \right) = \frac{1}{\theta^2} \mathbb{E}(Y) + \frac{1}{(1-\theta)^2} (n - \mathbb{E}(Y)).$$

↪ Remember that $Y \sim \text{Bin}(n, \theta)$, implies $E(Y) = n\theta$.

↪ Therefore,

$$I(\theta) = \frac{n}{\theta(1-\theta)}.$$

↪ We can then conclude that Jeffreys's prior is $\pi(\theta) \propto \theta^{-1/2}(1-\theta)^{-1/2}$, which is a Beta(1/2, 1/2) distribution and gives greater plausibility to values near 0 and 1 than to values in between (see figure below).



Choice of prior distribution

Non-informative/vague priors: Jeffreys' prior

- ↪ Let us now compute Jeffreys' prior for a normal distribution with mean θ (assuming that the variance σ^2 is known).
- ↪ With $Y_1, \dots, Y_n \stackrel{\text{iid}}{\sim} N(\theta, \sigma^2)$, the likelihood is

$$\begin{aligned} f(\mathbf{y} \mid \theta) &= \prod_{i=1}^n \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{1}{2\sigma^2} (y_i - \theta)^2 \right\} \right] \\ &= (2\pi\sigma^2)^{-n/2} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \theta)^2 \right\}, \end{aligned}$$

and the corresponding log likelihood is

$$\log f(\mathbf{y} \mid \theta) = C - \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \theta)^2.$$

Choice of prior distribution

Non-informative/vague priors: Jeffreys' prior

- ↪ The first and second derivatives are, respectively

$$\frac{d}{d\theta} \log f(\mathbf{y} | \theta) = \frac{n}{\sigma^2} (\bar{y} - \theta),$$

and

$$\frac{d^2}{d\theta^2} \log f(\mathbf{y} | \theta) = -\frac{n}{\sigma^2}.$$

- ↪ The expected Fisher information is then $I(\theta) = \frac{n}{\sigma^2}$, which does not depend on θ , thus implying that $\pi(\theta) \propto 1, \forall \theta$ (\Rightarrow improper prior).

Choice of prior distribution

Non-informative/vague priors: Jeffreys' prior

- ↪ We will now consider the Poisson case. Let $Y_1, \dots, Y_n \stackrel{\text{iid}}{\sim} \text{Poisson}(\theta)$, $\theta > 0$.
- ↪ The likelihood and log likelihood are, respectively, given by

$$f(\mathbf{y} | \theta) = \prod_{i=1}^n \left\{ \frac{e^{-\theta} \theta^{y_i}}{y_i!} \right\} = \frac{e^{-n\theta} \theta^{\sum_{i=1}^n y_i}}{\prod_{i=1}^n y_i!},$$

and

$$\log f(\mathbf{y} | \theta) = -n\theta + \sum_{i=1}^n y_i \log \theta + C.$$

- ↪ The first derivative is

$$\frac{d}{d\theta} \log f(\mathbf{y} | \theta) = -n + \frac{1}{\theta} \sum_{i=1}^n y_i,$$

while the second derivative is

$$\frac{d^2}{d\theta^2} \log f(\mathbf{y} | \theta) = -\frac{\sum_{i=1}^n y_i}{\theta^2}.$$

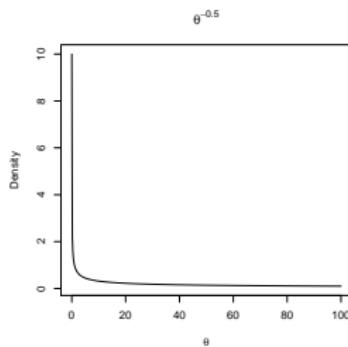
Choice of prior distribution

Non-informative/vague priors: Jeffreys' prior

↪ The expected Fisher information is then

$$I(\theta) = \frac{1}{\theta^2} n \mathbb{E}[Y] = \frac{n}{\theta},$$

implying that $\pi(\theta) \propto \theta^{-1/2}$.



↪ Jeffreys' prior is improper in this case, but it can be approximated by a Gamma distribution with parameters $\alpha = 1/2$ and $\beta \rightarrow 0$.

Choice of prior distribution

Non-informative/vague priors: Jeffreys' prior

- ↪ Jeffreys' prior is objective in that there is no prior tuning. It is a means of constructing a prior in the absence of prior information.
- ↪ Although Jeffreys' prior has the desirable property of being invariant to reparameterisations it can lead to improper priors.
- ↪ Alternative vague or non-informative prior distributions often have a reasonable mean for the distribution, but with a large variance parameter.
- ↪ Several different priors may be considered, each of which may be described to be vague or non-informative, and the sensitivity of the posterior on these priors investigated.

Choice of prior distribution

Conjugate priors

- ↪ A conjugate prior leads to a posterior from the same parametric family as the prior.
- ↪ There are long lists of conjugacies that we should be aware of

https://en.wikipedia.org/wiki/Conjugate_prior

- ↪ Conjugate priors are used often for computational convenience because the posterior has a closed form.
- ↪ In fancier models, conjugate priors facilitate Gibbs sampling which is the easiest Bayesian computational algorithm.

Choice of prior distribution

Conjugate priors: beta-binomial model

↪ Let us suppose $Y \sim \text{Bin}(n, \theta)$, with likelihood

$$f(y | \theta) = \binom{n}{y} \theta^y (1 - \theta)^{n-y}.$$

↪ It is mathematically convenient to use a Beta(a, b) prior distribution for θ because it has a similar form to the binomial likelihood. Its density function is

$$\pi(\theta) = \frac{1}{B(a, b)} \theta^{a-1} (1 - \theta)^{b-1} = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \theta^{a-1} (1 - \theta)^{b-1}.$$

↪ If $\theta \sim \text{Beta}(a, b)$, then

$$\mathbb{E}(\theta) = \frac{a}{a+b},$$

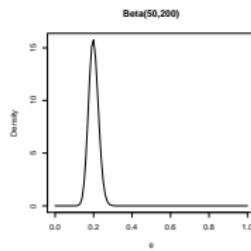
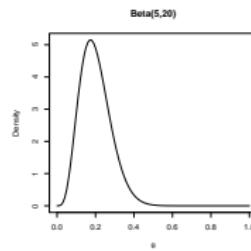
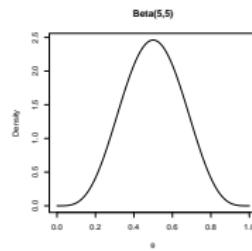
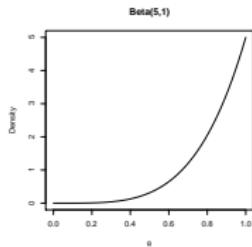
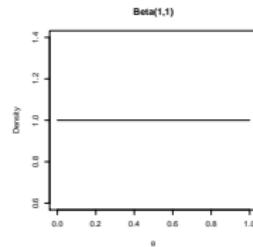
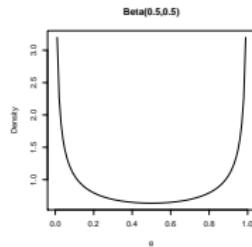
and variance

$$\text{var}(\theta) = \frac{ab}{(a+b)^2(a+b+1)}.$$

Choice of prior distribution

Conjugate priors: beta-binomial model

→ The beta distribution can take several different shapes.



Choice of prior distribution

Conjugate priors: beta-binomial model

- Combining the beta prior distribution for θ with the binomial likelihood results in the following posterior distribution

$$\begin{aligned} p(\theta | y) &\propto f(y | \theta)\pi(\theta) \\ &= \binom{n}{y} \theta^y (1-\theta)^{n-y} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \theta^{a-1} (1-\theta)^{b-1} \\ &\propto \theta^{a+y-1} (1-\theta)^{b+n-y-1}. \end{aligned}$$

- That is, the posterior distribution is another Beta distribution

$$\theta | y \sim \text{Beta}(a + y, b + n - y)$$

Choice of prior distribution

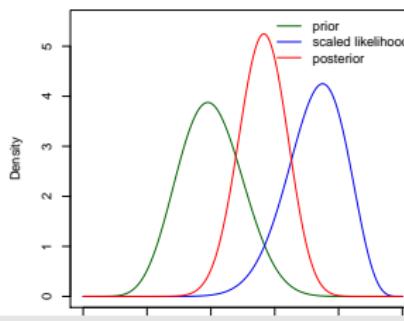
Conjugate priors: beta-binomial model

- ↪ The prior mean is $E(\theta) = \frac{a}{a+b}$.
- ↪ The data mean (MLE) is y/n .
- ↪ The posterior mean $\mathbb{E}[\theta | y] = \frac{a+y}{a+b+n}$.
- ↪ We can interpret prior information as being equivalent to having observed a successes in $a + b$ prior trials.
- ↪ With fixed a and b , as y and n increase, $\mathbb{E}[\theta | y] \rightarrow \frac{y}{n}$ (the MLE), and the variance tends to zero.
- ↪ This is a general phenomenon: as n increases, the posterior distribution gets more concentrated and the likelihood dominates the prior.

Choice of prior distribution

Conjugate priors: beta-binomial model

- ↪ Consider a drug to be given for relief of chronic pain.
- ↪ Experience with similar compounds has suggested that response rates, say θ , between 0.2 and 0.6 could be feasible.
- ↪ One way to turn this information into a prior is to interpret it as a distribution with mean $(0.2 + 0.6)/2 = 0.4$ and standard deviation $(0.6 - 0.2)/4 = 0.1$.
- ↪ A Beta(9.2, 13.8) distribution has these properties.
- ↪ Suppose we treat $n = 20$ volunteers with the compound and observe $y = 15$ positive responses.
- ↪ The parameters of the Beta distribution are updated to $9.2 + 15 = 24.2$ and $13.8 + 20 - 15 = 18.8$.



Choice of prior distribution

Conjugate priors: beta-binomial model

- ↪ Note that the likelihood, although a function of the parameter, it is not a density and so, in particular, does not integrate to one.
- ↪ In order to plot the likelihood along with the prior and posterior distributions, it is convenient that the three are in the same scale
- ↪ Therefore, we have rescaled the likelihood function so that it integrates to one.
- ↪ See implementation in the `R` script for Lecture 1.
- ↪ An alternative way to interpret the information that “response rates between 0.2 and 0.6 could be feasible” as a prior is to consider the effect of the prior, i.e. given the prior $\text{Beta}(a, b)$, the posterior is $\theta | y \sim \text{Beta}(a + y, b + n - y)$. This means that the prior effectively corresponds to data confirming a successes out of $a + b$ trials. Since the interval for the feasible response rates ($[0.2, 0.6]$) is rather large, it makes sense to choose the number of trials ($a+b$) rather low. So for example $\text{Beta}(4, 6)$ would be sensible, as this corresponds to 4 successes out of 10 trials. Such an approach can be applied even when it is difficult to link the information to the parameters directly.

Choice of prior distribution

Conjugate priors: poisson-gamma model

↪ Suppose we have $Y_1, \dots, Y_n \stackrel{\text{iid}}{\sim} \text{Poisson}(\theta)$.

↪ We have already seen that the likelihood is

$$f(\mathbf{y} | \theta) = \prod_{i=1}^n \left\{ \frac{e^{-\theta} \theta^{y_i}}{y_i!} \right\}$$

↪ The kernel of the Poisson likelihood (as a function of θ) has the same form as that of a Gamma(a, b) prior for θ

$$\pi(\theta) = \frac{b^a}{\Gamma(a)} \theta^{a-1} e^{-b\theta}.$$

↪ This parameterisation of the Gamma distribution in terms of the shape parameter a and rate parameter b , has mean a/b and variance a/b^2 .

Choice of prior distribution

Conjugate priors: poisson-gamma model

↪ With this prior, the posterior is

$$\begin{aligned} p(\theta | \mathbf{y}) &\propto f(\theta | \mathbf{y})\pi(\theta) \\ &= \left\{ \prod_{i=1}^n \frac{e^{-\theta} \theta^{y_i}}{y_i!} \right\} \frac{b^a}{\Gamma(a)} \theta^{a-1} e^{-b\theta} \\ &\propto e^{-n\theta} \theta^{\sum_{i=1}^n y_i} \theta^{a-1} e^{-b\theta} \\ &= \theta^{a+\sum_{i=1}^n y_i - 1} e^{-\theta(b+n)} \end{aligned}$$

↪ We recognise this as the kernel of a gamma distribution with parameters $a + n\bar{y}$ and $b + n$, that is

$$\theta | \mathbf{y} \sim \text{Gamma}(a + n\bar{y}, b + n).$$

Choice of prior distribution

Conjugate priors: poisson-gamma model

↪ Note that

$$\mathbb{E}(\theta | \mathbf{y}) = \frac{a + n\bar{y}}{b + n} = \bar{y} \left(\frac{n}{n + b} \right) + \frac{a}{b} \left(1 - \frac{n}{n + b} \right).$$

↪ The posterior mean is then a compromise between prior mean a/b and the MLE \bar{y} .

Choice of prior distribution

Conjugate priors: normal-normal model

- ↪ Let us now consider $Y_1, \dots, Y_n \stackrel{\text{iid}}{\sim} N(\theta, \sigma^2)$, with σ^2 known. Further, let $\theta \sim N(\mu_0, \sigma_0^2)$.
- ↪ We've already seen the likelihood

$$f(\mathbf{y} | \theta) = (2\pi\sigma^2)^{-n/2} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \theta)^2 \right\}.$$

- ↪ The posterior is

$$\begin{aligned} p(\theta | \mathbf{y}, \sigma^2) &\propto f(\mathbf{y} | \theta) \pi(\theta) \\ &\propto \exp \left\{ -\frac{1}{2\sigma^2} \sum_i (y_i - \theta)^2 \right\} \exp \left\{ -\frac{1}{2\sigma_0^2} (\theta - \mu_0)^2 \right\} \\ &\propto \exp \left\{ -\frac{1}{2\sigma^2 \sigma_0^2} \left[\theta^2(n\sigma_0^2 + \sigma^2) - 2\theta(n\bar{y}\sigma_0^2 + \mu_0\sigma^2) \right] \right\} \end{aligned}$$

Choice of prior distribution

Conjugate priors: normal-normal model

- ↪ This can be recognised as the density of a normal distribution with mean

$$\mu_n = \frac{n\bar{y}\sigma_0^2 + \mu_0\sigma^2}{n\sigma_0^2 + \sigma^2} = \frac{\frac{\mu_0}{\sigma_0^2} + \frac{n}{\sigma^2}\bar{y}}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}},$$

and variance

$$\sigma_n^2 = \frac{\sigma^2\sigma_0^2}{n\sigma_0^2 + \sigma^2} = \frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}}.$$

- ↪ The posterior parameters μ_n and σ_n^2 combine the prior parameters μ_0 and σ_0^2 with terms from the data.
- ↪ For instance, notice that

$$\mu_n = \frac{\tau_0}{\tau_0 + n\tau}\mu_0 + \frac{n\tau}{\tau_0 + n\tau}\bar{y},$$

where $\tau_0 = 1/\sigma_0^2$ (prior precision) and $\tau = 1/\sigma^2$ (sampling precision). The posterior mean is an average of the prior mean and the sample mean, weighted by their precisions.

Choice of prior distribution

Conjugate priors: normal-gamma model

- ↪ Suppose now that θ is known (which is an unrealistic scenario in practice), but the variance σ^2 is unknown.
- ↪ It is often convenient in Bayesian statistics to work with the precision, $\tau = 1/\sigma^2$.
- ↪ Let us take $\tau \sim \text{Gamma}(a, b)$.
- ↪ Then the posterior of τ is

$$\begin{aligned} p(\tau | \mathbf{y}, \theta) &\propto f(\mathbf{y} | \tau) \pi(\tau) \\ &\propto \tau^{n/2} \exp\left\{-\frac{\tau}{2} \sum_{i=1}^n (y_i - \theta)^2\right\} \tau^{a-1} e^{-b\tau} \\ &= \tau^{a+n/2-1} \exp\left\{-\tau \left(b + \frac{1}{2} \sum_i (y_i - \theta)^2\right)\right\}. \end{aligned}$$

- ↪ That is,

$$\tau | \mathbf{y}, \theta \sim \text{Gamma}\left(a + \frac{n}{2}, b + \frac{1}{2} \sum_i (y_i - \theta)^2\right).$$

Choice of prior distribution

Conjugate priors: normal-gamma model

↪ A common choice is to take both a and b very small, then the posterior is approximately

$$\tau \mid \mathbf{y}, \theta \sim \text{Gamma} \left(\frac{n}{2}, \frac{1}{2} \sum_i (y_i - \theta)^2 \right),$$

and so

$$\mathbb{E}[\tau \mid \mathbf{y}, \theta] = \left(\frac{1}{n} \sum_i (y_i - \theta)^2 \right)^{-1},$$

so that the posterior expectation of the precision is (approximately) the sample precision (but with a divisor of n and not $n - 1$).

Choice of prior distribution

Conjugate priors: mixture priors

- Conjugate priors are convenient, but sometimes they might be not flexible enough.
- Mixtures of conjugate priors are a good alternative and they are actually quite flexible.
- Fortunately, mixtures of conjugate priors are also conjugate. A mixture prior is

$$\pi(\theta) = \sum_{j=1}^J \omega_j p_j(\theta | \psi_j),$$

where p_j are conjugate priors with different hyperparameters, ψ_j , and the mixture weights $\omega_j \in [0, 1]$ sum to one.

- **Question:** If $Y \sim \text{Bin}(n, \theta)$ and $\pi(\theta) = \pi\text{Beta}(\theta | a_1, b_1) + (1 - \pi)\text{Beta}(\theta | a_2, b_2)$, then $\theta | y$ is ... (see first practical lab next week).
- **Additional reading:** see Section 2 of Bayesian Statistical Methods for more examples on the choice of prior distribution.

Predictive inference

- ↪ We now consider prediction.
- ↪ The prior predictive distribution of \mathbf{y} is

$$m(\mathbf{y}) = \int_{\Theta} f(\mathbf{y} \mid \theta) \pi(\theta) d\theta,$$

also called marginal distribution of the data, usually in the frequentist approach.

- ↪ It is useful to check whether the model (likelihood+prior) gives (un)reasonable predictions.
- ↪ The posterior predictive distribution of a future observation z , given the data \mathbf{y} is

$$\begin{aligned} f(z \mid \mathbf{y}) &= \int_{\Theta} f(z \mid \theta, \mathbf{y}) p(\theta \mid \mathbf{y}) d\theta \\ &= \int_{\Theta} f(z \mid \theta) p(\theta \mid \mathbf{y}) d\theta, \end{aligned}$$

where the second equality is due to the conditional independence of Y and Z .

- ↪ The predictive distribution only depends on z and \mathbf{y} .

Predictive inference

- ↪ Let us consider a beta binomial experiment, where $Y \sim \text{Bin}(n, \theta)$ and $\theta \sim \text{Beta}(a, b)$.
- ↪ We have seen that the posterior distribution $\theta | y \sim \text{Beta}(a + y, b + n - y)$.
- ↪ For ease of notation, let $c = a + y$ and $d = b + n - y$.
- ↪ Now let us consider a further random quantity Z for which we judge $Z \sim \text{Bin}(m, \theta)$ and Z and Y are conditionally independent given θ .
- ↪ So, having conducted n trials, we consider conducting a further m trials.
- ↪ We seek the predictive distribution of Z (the number of successes in m new trials) given the observed y .

Predictive inference

↪ We have

$$\begin{aligned}f(z \mid y) &= \int_{\Theta} f(z \mid \theta)p(\theta \mid y)d\theta \\&= \int_0^1 \binom{m}{z} \theta^z (1-\theta)^{m-z} \frac{1}{B(c,d)} \theta^{c-1} (1-\theta)^{d-1} d\theta \\&= \binom{m}{z} \frac{1}{B(c,d)} \int_0^1 \theta^{c+z-1} (1-\theta)^{d+m-z-1} d\theta \\&= \binom{m}{z} \frac{B(c+z, d+m-z)}{B(c,d)} \\&= \binom{m}{z} \frac{\Gamma(c+d)}{\Gamma(c)\Gamma(d)} \frac{\Gamma(c+z)\Gamma(d+m-z)}{\Gamma(c+d+m)}.\end{aligned}$$

↪ We say that $Z \mid y$ is the Binomial-Beta distribution with parameters c , d , and m .

↪ **Additional reading:** see Section 5.2 of Bayesian Theory by Bernardo and Smith for more examples of posterior predictive distributions from the exponential family.

Bayesian computing

- ↪ As we've seen, Bayesian inference centres around the posterior distribution

$$\begin{aligned} p(\theta \mid \mathbf{y}) &= \frac{f(\mathbf{y} \mid \theta)\pi(\theta)}{\int f(\mathbf{y} \mid \theta)\pi(\theta)d\theta} \\ &\propto f(\mathbf{y} \mid \theta)\pi(\theta). \end{aligned}$$

- ↪ Bayesian inference thus requires summarising the posterior distribution.
- ↪ When there are more than a few parameters, this requires advanced computational tools.
- ↪ We will first briefly discuss some methods that are deterministic. These usually rely on some approximation, such as
 - ↪ Bayesian Central Limit Theorem,
 - ↪ Numerical integration.
- ↪ Monte Carlo methods are more general, but they can become slow in high dimensions:
 - ↪ Direct sampling.
 - ↪ Markov chain Monte Carlo methods.

Bayesian computing

Bayesian central limit theorem

- In a frequentist approach it is common to compute the MLE, and then assume (based on asymptotic theory) that its sampling distribution is Gaussian for inference.
- The Bayesian central limit theorem can be used in the same way to summarise a posterior.
- **Bayesian central limit theorem:** Suppose $Y_1, \dots, Y_n \stackrel{\text{iid}}{\sim} f(\cdot | \theta)$ and that the prior $\pi(\theta)$ and the likelihood $f(\mathbf{y} | \theta)$ are positive and twice differentiable near $\hat{\theta}_{\text{post}}$, the posterior mode of θ . Then for large n

$$p(\theta | \mathbf{y}) \sim N\left(\hat{\theta}_{\text{post}}, [I^{\text{post}}(\theta, \mathbf{y})]^{-1}\right),$$

where

$$I^{\text{post}}(\theta, \mathbf{y}) = - \left[\frac{\partial^2}{\partial \theta \partial \theta^T} \log p(\theta | \mathbf{y}) \right]_{\theta=\hat{\theta}_{\text{post}}}.$$

Bayesian computing

Bayesian central limit theorem

- ↪ Other forms of the normal approximation are also often used.
- ↪ In particular, a powerful new type of models called latent Gaussian models have become widely used in the last two decades.
- ↪ In such models, we have a few dimensional (up to 15) non-Gaussian parameters called hyperparameters, and a large dimensional (up to 10^{12} in recent years) latent random vector that is Gaussian when conditioned on the hyperparameters. The observations are allowed to be nonlinear functions of the latent variables.
- ↪ The fact that the latent random vector is conditionally Gaussian allows for efficient and accurate computations using various clever tricks. These form the basis of the INLA (Integrated Nested Laplace Approximations) approach, which will be discussed in detail starting from Lecture 3.

Bayesian computing

Numerical integration

- ↪ The vast majority of posterior summaries are integrals of the posterior (posterior mean, variance, probability above zero, etc).
- ↪ There is a vast literature on approximating integrals.
- ↪ These work great in medium dimensions, say models with 1-15 parameters.
- ↪ The simplest method is a grid approximation. Suppose we have an unnormalised posterior distribution we want to sample from. The grid approximation works as follows:
 - ➊ Divide the unnormalised posterior area into m grids.
 - ➋ Evaluate each grid point at the unnormalised posterior.
 - ➌ Sample grid points with unnormalised posterior ordinates as the probabilities.

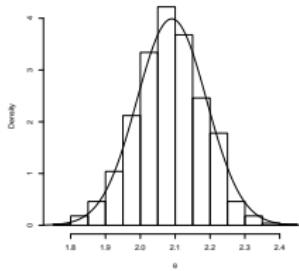
Bayesian computing

Numerical integration

- ↪ Suppose y_1, \dots, y_n is observed data and that $Y_i \stackrel{\text{iid}}{\sim} N(\theta, 1)$, and $\theta \sim N(\mu_0 = 0, \sigma_0^2 = 10^2)$.
- ↪ We do not need numerical integration here since we do know that $p(\theta | \mathbf{y})$ is actually available in closed form, namely

$$\theta | \mathbf{y}, \sigma^2 \sim N\left(\frac{\frac{\mu_0}{\sigma_0^2} + \frac{n\bar{y}}{\sigma^2}}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}}, \frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}}\right).$$

- ↪ This is only a toy example to illustrate the use of the method.



- ↪ See implementation in the R script for Lecture 1.

Bayesian computing

Monte Carlo sampling

- ↪ Monte Carlo sampling is the predominant method of Bayesian inference because it avoids asymptotic approximations and can be used in high-dimensions.
- ↪ The main idea is to approximate posterior summaries by drawing samples from the posterior distribution, and then averaging quantities of interest over these samples to approximate integrals of those quantities over the posterior.
- ↪ We have seen in last lecture that the best estimator of $h(\theta)$ in MSE is $\int_{\theta} h(\theta)p(\theta | \mathbf{y})d\theta$.
- ↪ For example, if $\theta^{(1)}, \dots, \theta^{(S)}$ are samples from $p(\theta | \mathbf{y})$, then the mean $\bar{\theta}$ of S samples can be used to approximate the posterior mean. More generally, we use estimators of the form $(h(\theta^{(1)}) + \dots + h(\theta^{(S)}))/S$ for $\int_{\theta} h(\theta)p(\theta | \mathbf{y})d\theta$.
- ↪ Many argue that this form of approximation is superior to asymptotic approximations because the Bayes CLT requires the sample size of the dataset to go to infinity and the Monte Carlo approximation requires the number of simulated values to go to infinity.
- ↪ In most cases, $S \rightarrow \infty$ is cheaper and more realistic than $n \rightarrow \infty$.
- ↪ But how to draw samples from some arbitrary distribution $p(\theta | \mathbf{y})$?

Bayesian computing

Monte Carlo sampling: Rejection sampling

- ↪ Suppose we wish to generate values $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(S)}$ from the posterior $p(\theta | \mathbf{y})$ but this is not a known distribution.
- ↪ Rejection sampling takes samples from a distribution that resembles the posterior and is easy to sample from (say a normal approximation using the CLT), and thins those samples to obtain draws from the posterior distribution.
- ↪ The approximate density, $g(\theta)$, is called the envelope function.
- ↪ Let M be a constant so that $p(\theta | \mathbf{y}) \leq Mg(\theta)$ for all θ . Then $p(\theta | \mathbf{y})$ resides in the envelope.

Bayesian computing

Monte Carlo sampling: Rejection sampling

↪ The rejection sampling algorithm is:

- 1 Sample $\theta \sim g(\theta)$ and draw $u \sim \text{Unif}(0, 1)$.
- 2 Set $\alpha = \frac{p(\theta | \mathbf{y})}{Mg(\theta)}$.
- 3 If $u \leq \alpha$, accept θ .

Repeat until you have accepted S (pre-determined) values. The accepted values $\theta^{(1)}, \dots, \theta^{(S)}$ are a random sample from $p(\theta | \mathbf{y})$.

Bayesian computing

Monte Carlo sampling: Rejection sampling

- It is sometimes tricky to choose a good envelope/proposal distribution $g(\theta)$. A basic requirement is that it should have support at least as large as $p(\theta | \mathbf{y})$ and preferably heavier tails than $p(\theta | \mathbf{y})$.
- It is desirable to choose M as small as possible for efficiency reasons - in fact, the expected number of samples taken until one is accepted is exactly M . Note that $M \geq \frac{p(\theta | \mathbf{y})}{g(\theta)}$ for all θ , so that the optimal M is simply

$$M^{\text{opt}} = \max_{\theta} \left(\frac{p(\theta | \mathbf{y})}{g(\theta)} \right).$$

- Rejection sampling also works when the normalisation constant is unknown. In this case, we have an unnormalised posterior density $\tilde{p}(\theta | \mathbf{y})$. If it satisfies that for a normalised proposal distribution $g(\theta)$,

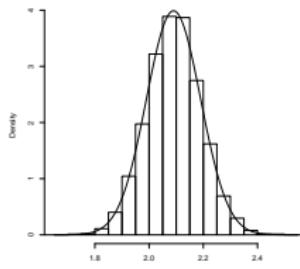
$$\tilde{p}(\theta | \mathbf{y}) \leq Mg(\theta) \quad \text{for every } \theta,$$

then samples from $g(\theta)$ accepted with probability $\alpha = \frac{\tilde{p}(\theta | \mathbf{y})}{Mg(\theta)}$ are distributed according to $p(\theta | \mathbf{y})$. This works because we can essentially incorporate the normalisation constant into the M term. The expected number of samples taken until one is accepted in this case is M/Z , where $Z = \int_{\theta} \tilde{p}(\theta | \mathbf{y}) d\theta$ is the normalising constant.

Bayesian computing

Monte Carlo sampling: Rejection sampling

- ↪ We return to the toy example used to illustrate the grid approximation to the posterior.
- ↪ In this context, we choose as an envelope distribution, a normal distribution with mean equal to the mean of the data and standard deviation equal to the standard deviation of the data.



- ↪ See implementation in the R script for Lecture 1.
- ↪ Although direct sampling methods such as rejection sampling can be very efficient in small dimensions, they become typically exponentially slow in higher dimensions.
- ↪ More general methods are needed.

Bayesian computing

MCMC

- ↪ MCMC techniques are based on the construction of a Markov chain that eventually ‘converges’ to the target distribution (called the stationary or equilibrium distribution) which, in our case, is the posterior distribution $p(\theta | \mathbf{y})$.
- ↪ This is the main way to distinguish MCMC algorithms from direct simulation methods, which provide samples directly from the target posterior distribution.
- ↪ Moreover, the MCMC output is a dependent sample since it is generated from a Markov chain, in contrast to the output of direct methods, which are independent samples.
- ↪ Finally, MCMC methods incorporate the notion of an iterative procedure (for this reason they are frequently called iterative methods) since in every step they produce values depending on the previous one.
- ↪ We will briefly cover two different MCMC methods: Gibbs sampler and Metropolis-Hastings.

Bayesian computing

MCMC: Gibbs sampler

- ↪ Gibbs sampling was proposed in the early 1990s (Geman and Geman, 1984; Gelfand and Smith, 1990) and fundamentally changed Bayesian computing.
- ↪ Gibbs sampling is attractive because it can sample from high-dimensional posteriors.
- ↪ The main idea is to break the problem of sampling from the high-dimensional joint distribution into a series of samples from low-dimensional conditional distributions.
- ↪ The algorithm begins by setting initial values for all parameters, $\theta^{(0)} = (\theta_1^{(0)}, \dots, \theta_p^{(0)})$.
- ↪ Variables are then sampled one at a time from their full conditional distributions

$$p(\theta_j | \theta_1, \dots, \theta_{j-1}, \theta_{j+1}, \dots, \theta_p, \mathbf{y}), \quad j = 1, \dots, p.$$

- ↪ Rather than 1 sample from the p -dimensional joint distribution, we make p samples from the 1-dimensional conditional distributions (sometimes using rejection sampling).
- ↪ The process is repeated until the required number of samples have been generated.

Bayesian computing

MCMC: Gibbs sampler

→ Formally, the algorithm is:

① Set initial values $\theta^{(0)} = (\theta_1^{(0)}, \dots, \theta_p^{(0)})$.

② For $s = 1, \dots, S$:

- Draw $\theta_1^{(s)} \sim p(\theta_1 | \theta_2^{(s-1)}, \theta_3^{(s-1)}, \dots, \theta_p^{(s-1)}, \mathbf{y})$.

- Draw $\theta_2^{(s)} \sim p(\theta_2 | \theta_1^{(s)}, \theta_3^{(s-1)}, \dots, \theta_p^{(s-1)}, \mathbf{y})$.

- ...

- Draw $\theta_p^{(s)} \sim p(\theta_p | \theta_1^{(s)}, \theta_2^{(s)}, \dots, \theta_{p-1}^{(s)}, \mathbf{y})$.

→ Then for s sufficiently large $(\theta_1^{(s)}, \dots, \theta_p^{(s)}) \stackrel{\text{approx.}}{\sim} p(\theta_1, \dots, \theta_p | \mathbf{y})$.

→ The convergence of the p -tuple obtained at iteration s , $(\theta_1^{(s)}, \dots, \theta_p^{(s)})$ to a draw from a joint posterior distribution occurs under mild regularity conditions that are generally satisfied for most statistical models (see, e.g., Geman and Geman, 1984, or Roberts and Smith, 1993).

Bayesian computing

MCMC: Gibbs sampler

- ↪ Suppose we have data y_1, \dots, y_n such that $Y_i \stackrel{\text{iid}}{\sim} N(\mu, \sigma^2)$, where both μ and σ^2 are unknown.
- ↪ We need to specify the joint prior distribution $p(\mu, \sigma^2)$. Common choices are:
 - ① $p(\mu, \sigma^2) = p(\mu | \sigma^2)p(\sigma^2)$.
 - ② $p(\mu, \sigma^2) = p(\mu)p(\sigma^2)$.
- ↪ We will use the independent prior (second option) and assume $\mu \sim N(\mu_0, \sigma_0^2)$, and $\sigma^2 \sim \text{IG}(a, b)$ (inverse Gamma distribution).
- ↪ We have that

$$\begin{aligned}\mu | \sigma^2, \mathbf{y} &\sim N\left(\frac{\frac{\mu_0}{\sigma_0^2} + \frac{n\bar{y}}{\sigma^2}}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}}, \frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}}\right), \\ \sigma^2 | \mu, \mathbf{y} &\sim \text{IG}\left(a + \frac{n}{2}, b + \frac{1}{2} \sum_{i=1}^n (y_i - \mu)^2\right).\end{aligned}$$

Bayesian computing

MCMC: Gibbs sampler

↪ The algorithm is then

1 Set initial values $\mu^{(0)}$ and $(\sigma^{(0)})^2$.

2 For $s = 1, \dots, S$

- Draw $\mu^{(s)} | \mathbf{y}, (\sigma^{(s-1)})^2 \sim N \left(\frac{\frac{\mu_0}{\sigma_0^2} + \frac{n\bar{y}}{(\sigma^{(s-1)})^2}}{\frac{1}{\sigma_0^2} + \frac{n}{(\sigma^{(s-1)})^2}}, \frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{(\sigma^{(s-1)})^2}} \right)$.
- Draw $(\sigma^{(s)})^2 | \mathbf{y}, \mu^{(s)} \sim \text{IG} \left(a + \frac{n}{2}, b + \frac{1}{2} \sum_{i=1}^n (y_i - \mu^{(s)})^2 \right)$.

↪ See implementation in the `R` script for Lecture 1.

Bayesian computing

MCMC: Gibbs sampler

- Later in the lecture we will look more closely at convergence diagnostics.
- One basic diagnostic is to monitor the *traceplots*: plots of the generated values of the parameters versus the iteration number.
- If all values are within a zone without strong periodicity and (especially) a trend, then there is no evidence of lack of convergence.

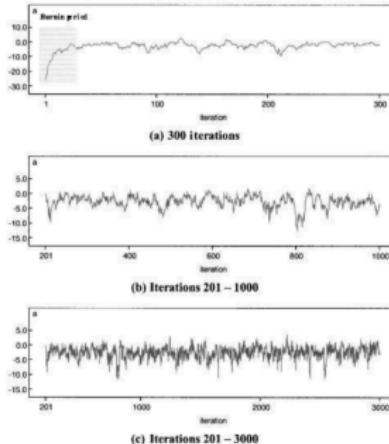


Figure: Traceplots for (a) 300 iterations, (b) 1000 iterations after discarding the first 200, and (c) 3000 iterations after discarding the first 200. Image from Ntzoufras (2009).

Bayesian computing

MCMC: Gibbs sampler

- ↪ In the first trace plot, we can clearly see the burnin period (within the gray box), which must be discarded from the final sample. After this period the generated sampled values are stabilised within a zone.
- ↪ In the second plot, the initial 200 iterations have been discarded to monitor the sampled values which demonstrate much better behaviour with small periodicities (up and down periods in the graph).
- ↪ Finally, generated values of the last trace plot are more convincing in terms of convergence, with all generated values within a parallel zone and no obvious tendencies or periodicities.

Bayesian computing

MCMC: Gibbs sampler

- Updating parameters one at a time can lead to high autocorrelation.
- The lag h autocorrelation function (ACF) for parameter θ_j is

$$\rho_j(h) = \lim_{s \rightarrow \infty} \text{Cor}(\theta_j^{(s)}, \theta_j^{(s-h)}),$$

i.e. it is the correlation between the given parameter value in the Markov chain separated by h iterations. The term $h > 1$ is usually referred to as lag.

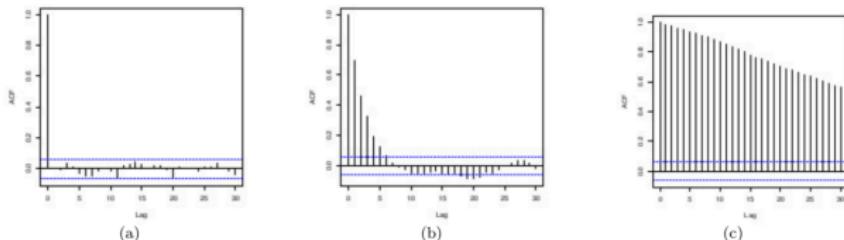


Figure: Sample ACF plots representing (a) ideal mixing, (b) typical good mixing, (c) poor mixing. Image from Prof. R. King.

Bayesian computing

MCMC: Gibbs sampler

- ↪ Note that the autocorrelation function is always equal to 1 for the value $h = 0$, since $\text{Cor}(\theta_j^{(s)}, \theta_j^{(s)}) = 1$.
- ↪ Ideally, for efficient Markov chains, there should be a fast decrease in the value of the autocorrelation function as the lag increases.
- ↪ This would imply that there is little relationship between values of the Markov chain within a small number of iterations.
- ↪ Conversely, poorly mixing chains will typically have a very shallow gradient in the ACF plot, with high autocorrelation values for even relatively large values of h .

Bayesian computing

MCMC: Gibbs sampler

- ↪ A simple, yet reasonably effective, method for dealing with autocorrelation is to only keep every k draws from the posterior and discard the rest; this is known as thinning the chain.
- ↪ The advantages of thinning are both simplicity and a reduction in memory usage—saving and working with large chains can be burdensome.
- ↪ The disadvantage is that we are clearly throwing away information; thinning can never be as efficient as using all the iterations.

Bayesian computing

MCMC: Metropolis–Hastings algorithm

- ↪ Gibbs sampling requires drawing a sample from each full conditional distribution.
- ↪ In cases where the conditional distributions are conjugate sampling is straight-forward. But what if they are not conjugate?
- ↪ We could make draws from the conditional distributions using rejection sampling. This works well if there are only a few non-conjugate parameters but can be difficult to tune.
- ↪ Other methods have been proposed, with Metropolis–Hastings being the most widely used.

Bayesian computing

MCMC: Metropolis–Hastings algorithm

- ↪ Metropolis et al. (1953) first formulated the Metropolis algorithm, and the paper was originally published in the physical sciences.
- ↪ Later, Hastings (1970) generalised the original method to what is now known as the Metropolis–Hastings algorithm.
- ↪ The latter is considered to be the general formulation of all MCMC methods.
- ↪ Green (1995) further generalised the Metropolis–Hastings algorithm by introducing reversible jump Metropolis–Hastings algorithms for sampling from parameter spaces with unknown dimension.

Bayesian computing

MCMC: Metropolis–Hastings algorithm

→ The algorithm is summarised as follows:

- 1 Set initial values $\theta^{(0)}$.
- 2 For $s = 1, \dots, S$, repeat the following steps:
 - Set $\theta = \theta^{(s-1)}$.
 - Draw candidate parameter values θ^* from a proposal distribution $q(\theta^* | \theta)$.
 - Calculate

$$\begin{aligned}\alpha(\theta, \theta^*) &= \min \left\{ 1, \frac{p(\theta^* | \mathbf{y})q(\theta | \theta^*)}{p(\theta | \mathbf{y})q(\theta^* | \theta)} \right\}, \\ &= \min \left\{ 1, \frac{f(\mathbf{y} | \theta^*)p(\theta^*)q(\theta | \theta^*)}{f(\mathbf{y} | \theta)\pi(\theta)q(\theta^* | \theta)} \right\}.\end{aligned}$$

- Generate $u \sim \text{Unif}(0, 1)$.
- Set

$$\theta^{(s)} = \begin{cases} \theta^* & \text{if } u \leq \alpha, \\ \theta & \text{if } u > \alpha. \end{cases}$$

Bayesian computing

MCMC: Metropolis–Hastings algorithm

- ↪ In the original Metropolis algorithm, only symmetric proposals of the type $q(\theta^* | \theta) = q(\theta | \theta^*)$ were considered.
- ↪ Random walk Metropolis is a special case of the algorithm with $q(\theta^* | \theta) = g(\theta^* - \theta)$ and the function g satisfying $g(\mathbf{x}) = g(-\mathbf{x})$.
- ↪ With such a proposal distribution the kernel driving the chain is a random walk since the candidate values are of the form

$$\theta^* = \theta + \mathbf{z}, \quad \mathbf{z} \sim g.$$

- ↪ Both cases result in an acceptance probability that depends only on the posterior (target) distribution

$$\alpha(\theta, \theta^*) = \min \left\{ 1, \frac{f(\mathbf{y} | \theta^*) p(\theta^*)}{f(\mathbf{y} | \theta) \pi(\theta)} \right\}.$$

- ↪ A common proposal of this type is a multivariate normal $q(\theta^* | \theta) = N_p(\theta, \mathbf{S}_\theta)$.

Bayesian computing

MCMC: Metropolis–Hastings algorithm

- ↪ As an implementation example, we consider the case where y_1, \dots, y_n are drawn from a normal distribution with known variance and unknown mean. The prior for the mean is normally distributed. Of course, as we have already seen, we do not need Metropolis type algorithms to sample from the posterior in this case, but it serves as an illustrative example.
- ↪ See implementation in the R script for Lecture 1.

Bayesian computing

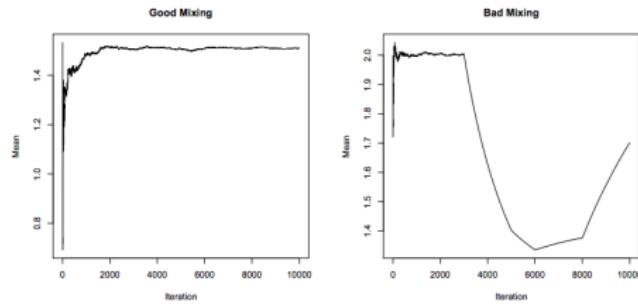
More on MCMC convergence diagnostics

- ↪ From the theory of Markov chains (more details were given in the Bayesian theory course), we expect the chains to eventually converge to the stationary distribution, which is also our target distribution. In this case the target is the posterior distribution.
- ↪ However, there is no guarantee that the chain has converged after S draws.
- ↪ How do we know whether the chain has actually converged?
- ↪ We can never be sure, but there are several tests we can do to check if the chain appears to be converged.

Bayesian computing

More on MCMC convergence diagnostics

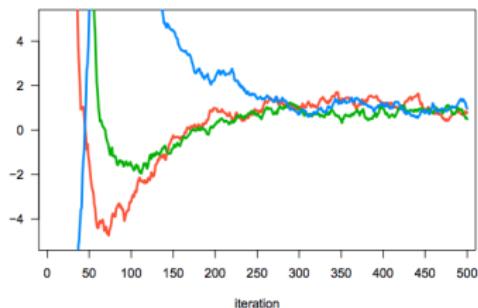
- ↪ We've already seen the use of traceplots and autocorrelation functions.
- ↪ Another quick check is to look at the running mean plots to check how well our chains are mixing.
- ↪ A running mean plot is a plot of the mean of all draws up to the current iteration versus the iteration number.



Bayesian computing

More on MCMC convergence diagnostics

- Many approaches for detecting convergence, both formal and informal, rest on the idea of starting multiple Markov chains and observing whether they come together and start to behave similarly (if they do, we can pool the results from each chain).



- This plot indicates that the three chains converge to the posterior after around $S = 300$ iterations; certainly, however, the number of iterations required to reach convergence depends on the initial values.
- It is typically recommended (e.g., Gelman and Rubin, 1992) to use overdispersed initial values, meaning 'more variable than the target distribution' i.e., the posterior.

Bayesian computing

More on MCMC convergence diagnostics

- ↪ Although looking at trace plots is certainly useful, it is also desirable to obtain an objective, quantifiable measure of convergence.
- ↪ Numerous methods exist, although we will focus on the measure originally proposed in Gelman and Rubin (1992).
- ↪ The basic idea is to quantify the between-chain and the within-chain variability of a quantity of interest. If the chains have converged, these measures will be similar; otherwise, the between-chain variability will be larger.

Bayesian computing

More on MCMC convergence diagnostics

→ The basic idea of the estimator is as follows (the actual estimator makes a number of modifications to account for degrees of freedom):

- Let B denote the standard deviation of the pooled sample of all MS iterations, where M is the number of chains and S is the number of iterations in each chain.
- Let W denotes the average of the within-chain standard deviations.
- Quantify convergence with

$$\widehat{R} = \frac{B}{W}.$$

→ If $\widehat{R} \gg 1$, this is clear evidence that the chains have not converged.

→ As $S \rightarrow \infty$, $\widehat{R} \rightarrow 1$; $\widehat{R} < 1.05$ is widely accepted as implying sufficient convergence for practical purposes.

Bayesian computing

More on MCMC convergence diagnostics

→ More details (along with other convergence measures) are given in the `coda` package, whose `gelman.diag` function provides, in addition to \hat{R} itself:

- An upper confidence interval for \hat{R} .
- A multivariate extension of \hat{R} for quantifying convergence of the entire posterior.

Bayesian computing

More on MCMC convergence diagnostics

- ↪ The obvious downside to running multiple chains is that it is inefficient: we intentionally force our sampler to spend extra time in a non-converged state, which in turn requires much more burn-in.
- ↪ The obvious upside, however, is that it provides us with some measure of confidence that we are actually drawing samples from the posterior.
- ↪ But we reiterate that (without additional assumptions about the posterior) no method can truly prove convergence; diagnostics can only detect failure to converge.

Bayesian computing

More on MCMC convergence diagnostics

- We may use \hat{R} , then, as a guide to how long we must run our chains until convergence.
- The obvious next question is: how long must we run our chains to obtain reasonably accurate estimates of the posterior?
- If we could obtain iid draws from the posterior, estimating the Monte Carlo standard error (at least, of the posterior mean) is straightforward: letting σ_{post} denote the posterior standard deviation, the MCSE is $\sigma_{\text{post}}/\sqrt{S}$.
- But, this will underestimate the true MC standard error due to autocorrelation in the samples generated using MCMC.
- There are various approaches to obtain better estimate of the MC error, with possibly the most popular being the *batches* mean method.
- **Additional reading:** for additional material on Bayesian computing, see Sections 10-12 of Bayesian Data Analysis by Gelman et al.

Bayesian Data Analysis

Daniel Paulin & Nicolò Margaritella

University of Edinburgh



Semester 2, 2020/2021

With thanks to Jonathan Gair, Ruben Amoros-Salvador, Ken Newman, Vanda Inácio and Natalia Bochkina for much of the material.

Outline

1 The BUGS language

- Introduction and historical development
- Running JAGS via `rjags`
- Example: Bayesian Linear Regression
- Example: Nonlinear regression

2 Model checking

- Checking normality of residuals by Q-Q plots
- Posterior predictive checks

The BUGS language

- ↪ We will learn now the basics of the BUGS language.
- ↪ BUGS stands for **B**ayesian **I**nference **U**sing **G**ibbs **S**ampling and began in 1989.
- ↪ Its basic philosophy was to separate the language used to describe a model from the actual programs used to carry out the computations (the *engine*).
- ↪ This approach has two attractive advantages:
 - ↪ One can easily specify complex models without requiring extensive knowledge of Bayesian computational tools.
 - ↪ The language has remained stable and consistent over time, even as the underlying programs which actually do the sampling are constantly changing.

The BUGS language

- The syntax of the BUGS language is relatively straightforward. Every line does one of the following things:
 - Defines a stochastic node
`theta~dbeta(1,1)`
 - Defines a deterministic node
`tau=pow(sigma,-2)`
 - Provides a comment
`## Prior`
 - Defines a loop
`for (i in 1:n)`
- Common distributions in the BUGS language are `dbin`, `dpois`, `dunif`, `dnorm`, `dgamma`.
- The normal is parameterised in terms of its mean and precision (=1/ variance).
- Functions cannot be used as arguments in distributions (one needs to create new nodes).
- For more details on the distributions and functions available in BUGS, check the online manual.

The BUGS language

- ↪ Historically, the most widely used BUGS engine has been a program called WinBUGS. It began in the early 1980s under David Spiegelhalter at the Medical Research Council Biostatistics Unit in Cambridge, UK.
- ↪ Further developments by the developers of WinBUGS, however, have focused on a somewhat different program called OpenBUGS; the features and appearance of the programs are similar, but OpenBUGS is more compatible with different operating systems.
- ↪ Information about WinBUGS and OpenBUGS is available at

<https://www.mrc-bsu.cam.ac.uk/software/bugs/>

- ↪ A separate project, JAGS (Just Another Gibbs Sampler), can be thought of as an engine for running BUGS models, although strictly speaking, the language syntax for the two programs is not always identical.

<http://mcmc-jags.sourceforge.net/>

The BUGS language

- Another two recent engines are STAN and NIMBLE, which can be found, respectively, at

<http://mc-stan.org/>

<https://r-nimble.org>

The BUGS language

→ All engines carry out the following steps for fitting BUGS models:

- 1 Checking model syntax.
- 2 Reading in data.
- 3 Compiling the model.
- 4 Initialising the simulation.
- 5 Sampling.
- 6 Reporting results.

The BUGS language

Running JAGS via `rjags`

- In general, the steps to using JAGS to produce samples are:

1 Download JAGS at

<http://sourceforge.net/projects/mcmc-jags/files/>

2 Install the `rjags` package in R, which should automatically find your JAGS installation.

3 Specify the statistical model (likelihood and prior) using the `model` command.

4 Compile the model using `jags.model`.

5 Generate samples using `update` and `coda.samples`.

- As it was already mentioned, there are, at least, 3 R interfaces for JAGS. The model syntax is the same for the three interfaces, what changes is how relevant information is extracted. We will stick with `rjags`, although you are free to choose your preferred interface.
- As we are using Kaggle.com for the workshops in this course, the code for loading JAGS and `rjags` is already included in the workshop notebook file. So no installation is required on your personal computer.

The BUGS language

Running JAGS via `rjags`

- We will learn `rjags` as we go! As an introductory example, we will go back to the example where we have normally distributed data where both the mean and the variance are unknown, i.e., $Y_i \stackrel{\text{iid}}{\sim} N(\mu, \sigma^2)$, with μ and σ^2 unknown (see R script for Lecture 2, JAGS example unknown mean and variance).
- We note again that the BUGS language uses the precision, instead of the variance, in the parameterisation of the normal distribution. Therefore, we assume $\mu \sim N(\mu_0, \sigma_0^2)$ and $\sigma^{-2} \sim \text{Gamma}(a, b)$, with $\mu_0 = 0$, $\sigma_0^2 = 100$, and $a = b = 0.1$.
- First we must load the `rjags` package

```
require(rjags)
```

The BUGS language

Running JAGS via `rjags`

- The program specifying the model (BUGS) code must be put in a separate file which is then read by JAGS. When working in R this is most conveniently done by saving the model in an object, e.g., `model_string="model{...}"` and then using the R function `textConnection` when passing the model to the `jags.model` function.

```
model_string <- "model{

  # Likelihood
  for(i in 1:n){
    y[i]~dnorm(mu,inv.var)
  }

  # Prior for mu
  mu~dnorm(mu0,inv.var0)
  inv.var0=1/sigma02

  # Prior for the inverse variance
  inv.var~dgamma(a, b)

  # Compute the variance
  sigma2=1/inv.var
}"
```

The BUGS language

Running JAGS via rjags

↪ Alternatively we can use the R function `cat`, to put the model in a file, say, `m1.jag`

```
cat ("model{  
  
# Likelihood  
for(i in 1:n){  
y[i]~dnorm(mu,inv.var)  
}  
  
# Prior for mu  
mu~dnorm(mu0,inv.var0)  
inv.var0=1/sigma02  
  
# Prior for the inverse variance  
inv.var~dgamma(a, b)  
  
# Compute the variance  
sigma2=1/inv.var  
}",  
file="m1.jag")
```

The BUGS language

Running JAGS via `rjags`

- ↪ The BUGS language is declarative, i.e., it is not executed as the program runs. Instead it is a specification of the model structure, and after the model is set up JAGS will decide how best to go about the MCMC simulation.
- ↪ We now specify the parameters we will feed into JAGS (data and priors)

```
data=list(y=y, n=n, mu0=mu0, sigma02=sigma02, a=a, b=b)
```

- ↪ Starting values can be supplied, although JAGS will be able to generate them in most cases. If we want to run several chains, for example to monitor convergence, we must supply starting values for each chain.
- ↪ In this example, we use three chains, hence the initial values are a list of three lists. Each of these lists has one named value for each parameter – in this case there are two parameters: `mu` (mean) and `inv.var` (precision).

```
inits=list(list(mu=mean(y), inv.var=1/var(y)), list(mu=0, inv.var=1), list(mu=10, inv.var=0.1))
```

The BUGS language

Running JAGS via `rjags`

- Once the model, the data, and the initial values (in case we want to supply them) have been specified, we use the function `jags.model` to compile and initialise the model.

```
model=jags.model(textConnection(model_string),n.chains=3,data=data,inits=inits)
```

- As this step, we obtain the following

```
Compiling model graph
Resolving undeclared variables
Allocating nodes
Graph information:
  Observed stochastic nodes: 150
  Unobserved stochastic nodes: 2
  Total graph size: 160
```

```
Initializing model
```

- In case we have used the function `cat` to define the model and saved it as file `m1.jag`, we would instead write

```
model=jags.model("m1.jag",n.chains=3,data=data,inits=inits)
```

The BUGS language

Running JAGS via `rjags`

- ↪ To get samples from the posterior distribution of the parameters, we use the `coda.samples` function after first using the `update` function to run the Markov Chain for a burn-in period of a number of specified iterations (in this case 1000).
- ↪ For `coda.samples` we must specify
 - ↪ The variables (nodes) that we want to monitor in the subsequent cycles of the chain. This is done using the argument `variable.names`. In our case the variables we want to monitor are called `mu` and `sigma2`.
 - ↪ How many iterations to run the chain (`n.iter`).
 - ↪ How often we sample the specified parameters and retain the results in memory (`thin`, by default equal to one).

The BUGS language

Running JAGS via `rjags`

↪ We thus type

```
update(model, 1000, progress.bar="none")
res=coda.samples(model, variable.names=c("mu", "sigma2"), n.iter=10000, progress.bar="none")
```

↪ As always in R, the most useful overview comes from the `summary` function

```
> summary(res)
```

Iterations = 12001:22000

Thinning interval = 1

Number of chains = 3

Sample size per chain = 10000

1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:

	Mean	SD	Naive SE	Time-series SE	SE
mu	1.926	0.2336	0.001348		0.001336
sigma2	8.215	0.9591	0.005537		0.005640

2. Quantiles for each variable:

	2.5%	25%	50%	75%	97.5%
mu	1.468	1.767	1.926	2.083	2.384
sigma2	6.541	7.541	8.144	8.812	10.283

The BUGS language

Running JAGS via `rjags`

- ↪ We can also type `plot(res)` to visualise the traceplots and densityplots.
- ↪ For inspection of the autocorrelation function, type `autocorr.plot(res)` and for the effective sample size use `effectiveSize(res)`. The Gelman-Rubin statistic is obtained by typing `gelman.plot(res)` (requires that the number of chains is equal or greater than 2).
- ↪ We can also extract results for each chain and each parameter using the object `res`. For instance, to obtain the values of `mu` of the first chain, we type `res[[1]][,1]`, while for obtaining the `sigma2` values for chain 3, we use `res[[3]][,2]`.
- ↪ We can also collect all the posterior samples from the different chains in one matrix by typing `resmat=as.matrix(res)`.

Regression

Linear regression: general context

- ↪ Linear regression is, perhaps, the most widely used statistical modelling tool.
- ↪ It addresses the following question: How does a quantity of primary interest, y , vary with (depend upon) another quantity, or set of quantities, \mathbf{x} ?
- ↪ The quantity y is called the response or outcome variable. Some people simply refer to it as the dependent variable.
- ↪ The variable(s) $\mathbf{x} = (x_1, x_2, \dots, x_p)$ are called explanatory variables, covariates or simply independent variables.
- ↪ For example, y might be the final mark in the Bayesian data analysis class. Variable x_1 might be the number of workshops attended, x_2 the amount of time spent studying, and so on. Regression allows us to test the effect of, say, the number of workshops attended, but accounting for the time spent studying.

Regression

Linear regression: examples

Some examples (of varying quality!) in the media:

Does chocolate make you clever?

By Charlotte Pritchard
BBC News



Eating more chocolate improves a nation's chances of producing Nobel Prize winners - or at least that's what a recent study appears to suggest. But how much chocolate do Nobel laureates

In today's Magazine

12 March 2012 Last updated at 20:17

Share

Red meat increases death, cancer and heart risk, says study

Comments (729)

A diet high in red meat can shorten life expectancy, according to researchers at Harvard Medical School.

The study of more than 120,000 people suggested red meat increased the risk of death from cancer and heart problems.

Substituting red meat with fish, chicken or nuts lowered the risks, the authors said.

The British Heart Foundation said red meat could still be eaten as part of a balanced diet.

The researchers analysed data from 37,698 men between 1986 and 2008 and 83,644 women between 1980 and 2008.

They said that during the study period, adding an extra portion of unprocessed red meat to someone's daily diet would increase the risk of death by 13%, of fatal cardiovascular disease by 18% and of cancer mortality by 10%. The figures for processed meat were higher. 20% for



Experts advise to choose leaner cuts of red meat

Related Stories

Red meat study: Risks 'very clear'

Cut red meat to lower cancer risk

How much red meat should we eat?

Regression

Linear regression

News > Society > Cancer

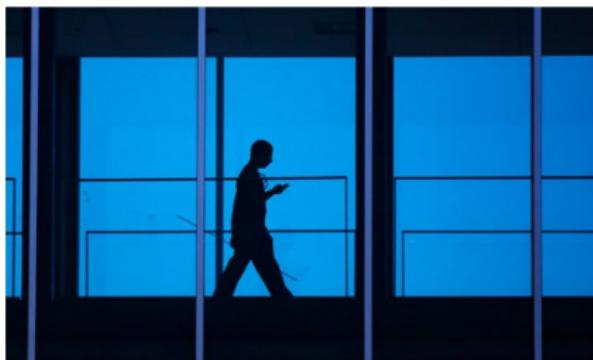
No clear evidence that mobile phone radiation damages health

Largest review yet of published research finds that so far there has been 'no indication' of increased risk

Ian Sample, science correspondent

theguardian.com, Thursday 26 April 2012 06.30 BST

[Jump to comments \(76\)](#)



A report by a Health Protection Agency advisory group has found 'no indication' of an increased risk to health from mobile phone use. Photograph: Mark Blinch/Reuters

How a short nap can raise the risk of diabetes: Study finds people who have a siesta are more likely to have high blood pressure and high cholesterol

- Napping for more than 30 minutes at a time can raise the risk of diabetes, according to a new study
- It can also increase likelihood of high blood pressure and high cholesterol

By PAT HAGAN

PUBLISHED: 01:04, 21 September 2013 | UPDATED: 10:34, 21 September 2013

[Share](#) [Tweet](#) [g+ 1](#) [Share](#) | 598 shares

102 [View comments](#)

They were much favoured by Margaret Thatcher, Albert Einstein and Winston Churchill.

But while afternoon naps may revitalise tired brains, they can also increase the risk of diabetes, according to new research.

A study of more than 27,000 people in China – where taking a post-lunch snooze is very popular – shows napping for more than 30 minutes at a time can raise the chances of developing type two diabetes.

Researchers found men and women taking 40 winks were also more likely to have high blood pressure and raised cholesterol levels compared to those who stayed awake through the day.

Regression

Linear regression: examples

→ The two main aims of regression analysis are

- ① **Estimation:** the estimation of relationships between the explanatory and response variables with the object of both identifying and quantifying this relationship.
- ② **Prediction:** the prediction of new values of y from values of x_1, \dots, x_p .

Regression

Linear regression: formulation

→ We will start by considering regression models of the form

$$y_i \stackrel{\text{ind}}{\sim} N(\mu_i, \sigma^2), \quad i = 1, \dots, n \quad (1)$$

where

$$\mu_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}.$$

- The variables $\beta_0, \beta_1, \dots, \beta_p$ are an unknown set of regression coefficients (to be estimated from the data).
- Equation (1) can also be written as

$$y_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \epsilon_i, \quad i = 1, \dots, n$$

where $\epsilon_i \stackrel{\text{iid}}{\sim} N(0, \sigma^2)$.

Regression

Linear regression: formulation

↪ This is because, since the x_{ij} are known, we can write

$$\begin{aligned}\mathbb{E}(y_i \mid x_{i1}, \dots, x_{ip}) &= \mathbb{E}(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \epsilon_i) \\ &= \beta_0 + \dots + \beta_p x_{ip} \\ &= \mu_i\end{aligned}$$

$$\begin{aligned}\text{var}(y_i \mid x_{i1}, \dots, x_{pi}) &= \text{var}(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \epsilon_i) \\ &= \text{var}(\epsilon_i) \\ &= \sigma^2\end{aligned}$$

↪ In short, $y_i = \mathbf{x}_i^T \boldsymbol{\beta} + \epsilon_i$, where $\mathbf{x}_i = (1, x_{i1}, \dots, x_{ip})^T$ and $\boldsymbol{\beta} = (\beta_0, \beta_1, \dots, \beta_p)^T$.

Regression

Linear regression: formulation

↪ In matrix notation, we have

$$\mathbf{y} = \mathbf{x}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where $\mathbf{y} = (y_1, \dots, y_n)^T$, $\boldsymbol{\varepsilon} = (\varepsilon_1, \dots, \varepsilon_n)^T$,

$$\mathbf{x} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix},$$

and $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \sigma^2 I_n)$.

Regression

Linear regression: formulation

↪ There are $p + 2$ unknown parameters to be estimated:

$$\hookrightarrow \beta = (\beta_0, \beta_1, \dots, \beta_p)^T$$

$$\hookrightarrow \sigma^2$$

↪ The data $(y_i, x_{i1}, x_{i2}, \dots, x_{ip})$ for $i \in \{1, \dots, n\}$ are used to obtain estimates.

↪ Under the classical setting, the estimator for β is given by

$$\hat{\beta} = (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \mathbf{y},$$

assuming that $(\mathbf{x}^T \mathbf{x})$ is invertible. Since we are assuming normal errors, both maximum likelihood and ordinary least squares give rise to this estimator.

Regression

Linear regression: Bayesian formulation

- ↪ We will now turn our attention to Bayesian estimation of β and σ^2 .
- ↪ The Bayesian linear model is given by

$$\begin{aligned}y_i \mid \mu_i, \sigma^2, \mathbf{x}_i &\stackrel{\text{ind}}{\sim} N(\mu_i, \sigma^2), \quad i = 1, \dots, n, \\ \mu_i &= \mathbf{x}_i^T \boldsymbol{\beta}, \\ \boldsymbol{\beta}, \sigma^2 &\sim \pi(\boldsymbol{\beta}, \sigma^2).\end{aligned}$$

- ↪ $\pi(\boldsymbol{\beta}, \sigma^2)$ is the joint prior on the parameters. There are several approaches to model the joint prior distribution.
- ↪ All inference proceeds from $p(\boldsymbol{\beta}, \sigma^2 \mid \mathbf{y})$.
- ↪ The full conditionals $p(\beta_j \mid \tau, \mathbf{y})$ are not difficult, but tedious, to obtain. The full conditional distribution $p(\tau \mid \boldsymbol{\beta}, \mathbf{y})$, where $\tau = \sigma^{-2}$, is easily obtained.

Regression

Linear regression: Bayesian formulation

- In normal regression models, the simplest approach is to assume that all parameters are a priori independent having the structure

$$\pi(\beta, \tau) = \prod_{j=0}^p \pi_j(\beta_j) \pi(\tau), \quad \tau = \sigma^{-2},$$
$$\beta_j \sim N(\mu_{\beta_j}, \sigma_{\beta_j}^2), \quad j = 0, \dots, p,$$
$$\tau \sim \text{Gamma}(a, b).$$

- In this prior setup, again, we have substituted the variance σ^2 by the corresponding precision parameter τ in order to make it compatible to the BUGS notation.

Regression

Linear regression: Bayesian formulation

- ↪ When no information is available, a usual choice for the prior mean is the zero value ($\mu_{\beta_j} = 0$).
- ↪ This choice centers our prior beliefs around zero, which corresponds to the assumption of no effect of x_j on y .
- ↪ In this way, we express our prior doubts about the effect of x_j on y , prompting Spiegelhalter et al. (2004, pp. 90, 158-160) to call this a “sceptical” prior.
- ↪ The prior variance $\sigma_{\beta_j}^2$ of the effect β_j is set equal to a large value (e.g., 10^4) to represent high uncertainty or prior ignorance.
- ↪ A very popular prior for τ , especially among BUGS users, is to use equally low prior parameter values, e.g., $a = b = 0.1$, or $a = b = 0.01$. Both prior configurations lead to a peak in 0^+ . Their use is controversial. For a good discussion on this topic, see

<http://www.stat.columbia.edu/~gelman/research/published/taumain.pdf>

Regression

Linear regression: Bayesian formulation

- ↪ Each regression coefficient represents the effect of explanatory variable x_j on the expectation of the response variable y adjusted for the remaining covariates.
- ↪ To ascertain whether the effect of x_j is important for the prediction or description of y , we initially focus on examining whether the posterior distribution of β_j is scattered around zero (or not).
- ↪ Posterior distributions far away from the zero value indicate an important contribution of x_j to the prediction of the response variable.
- ↪ Although formal Bayesian hypothesis testing is not based on simply examining the posterior distribution and derived credible intervals, such an analysis offers a first and reliable tool for identifying important variables.

Regression

Linear regression: Bayesian formulation

- The magnitude of the effect of variable x_j on y is given by the posterior distribution of β_j ($j = 1, \dots, p$) since

$$\begin{aligned} & \mathbb{E}(y \mid x_1, \dots, x_{j-1}, x_j = x + 1, x_{j+1}, \dots, x_p) - \mathbb{E}(y \mid x_1, \dots, x_{j-1}, x_j = x, x_{j+1}, \dots, x_p) \\ &= \beta_0 + \beta_j(x + 1) + \sum_{k \neq j, k=1}^p \beta_k x_k - \beta_0 - \beta_j x - \sum_{k \neq j, k=1}^p \beta_k x_k \\ &= \beta_j \end{aligned}$$

- Hence the posterior mean or median of β_j will correspond to the corresponding posterior measures of the expected change of the response variable y .
- An increase of one unit of x_j , given that the remaining covariates remain unchanged, induces an a posteriori average change on the expectation of y equal to the posterior mean of β_j .

Regression

Linear regression: Bayesian formulation

- ↪ The interpretation of the intercept parameter, β_0 , is as the expected value of the response variable y when the observed values of all covariates are equal to zero.
- ↪ Frequently such a combination lies outside the range of the observed covariate values and so, in many cases, direct interpretation of β_0 does not lead to realistic and sensible interpretations.
- ↪ An alternative is to center around zero all explanatory variables x_j , by subtracting their sample mean. In this case, the constant β_0 represents the expected value of y when all covariates are equal to the sample means, representing in this way the expected response y for an “average” or “typical” subject within our sample.
- ↪ Note that this does not change the meaning of $\beta_j, j = 1, \dots, p$.
- ↪ Centering (or centering and scaling) has some further advantages: it usually helps the mixing of the MCMC chains and also facilitates prior specification.

Regression

Linear regression

- Let us look at an example. The following dataset (the `mtcars` dataset available on R) gives the fuel consumption of cars, together with three aspects of car construction and performance.

	mpg	Rear axle ratio	Weight (lb/1000)	1/4 mile time
Mazda RX4	21.0	3.90	2.620	16.46
Mazda RX4 Wag	21.0	3.90	2.875	17.02
Datsun 710	22.8	3.85	2.320	18.61
Hornet 4 Drive	21.4	3.08	3.215	19.44
Hornet Sportabout	18.7	3.15	3.440	17.02
Valiant	18.1	2.76	3.460	20.22
Duster 360	14.3	3.21	3.570	15.84
:	:	:	:	:

- Based on these data we are interested in estimating the fuel consumption of a car based on the rear axle ratio, the weight, and the 1/4 mile time.

Regression

Linear regression

- We will fit a Bayesian multiple linear regression model to the `mtcars` dataset.
- The model we will fit is

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \varepsilon_i, \quad \varepsilon_i \stackrel{\text{iid}}{\sim} N(0, \sigma^2), \quad i = 1, \dots, 32,$$

where

- y_i is the miles driven per gallon for the i -th car.
- x_{i1} is the rear axle ratio for the i -th car.
- x_{i2} is the weight of the i -th car (lb/1000).
- x_{i3} is the 1/4 mile time for the i -th car.
- We will consider $\beta_j \sim N(0, 1000)$ (variance of 1000 or precision of 0.001) and $\tau \sim \text{Gamma}(0.1, 0.1)$.

Regression

Linear regression

→ We start by fitting a frequentist regression model (using ordinary least squares).

```
data(mtcars)
help(mtcars)

#creating a new dataframe with only mpg,drat,wt, and qsec as variables
vars=names(mtcars)%in%c("cyl", "disp", "hp", "vs","am","gear","carb") #variables to exclude
mtcars1=mtcars[!vars]

fit=lm(mpg~drat+wt+qsec,data=mtcars1)
summary(fit)
```

Call:

```
lm(formula = mpg ~ drat + wt + qsec, data = mtcars1)
```

Residuals:

Min	1Q	Median	3Q	Max
-4.1152	-1.8273	-0.2696	1.0502	5.5010

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	11.3945	8.0689	1.412	0.16892
drat	1.6561	1.2269	1.350	0.18789
wt	-4.3978	0.6781	-6.485	5.01e-07 ***
qsec	0.9462	0.2616	3.616	0.00116 **

Residual standard error: 2.56 on 28 degrees of freedom
Multiple R-squared: 0.837, Adjusted R-squared: 0.8196
F-statistic: 47.93 on 3 and 28 DF, p-value: 3.723e-11



Regression

Linear regression

- We now implement the model using the package `rjags`. We need to specify the likelihood of our normal multiple linear regression model.

```
require(rjags)
n=nrow(mtcars1)
#covariates and response
y=mtcars1$mpg; drat=mtcars1$drat; wt=mtcars1$wt; qsec=mtcars1$qsec

model_string <- "model{

  # Likelihood
  for(i in 1:n){
    y[i]~dnorm(mu[i],tau)
    mu[i]=beta[1]+beta[2]*drat[i]+beta[3]*wt[i]+beta[4]*qsec[i]
  }

  # Prior for beta
  for(j in 1:4){
    beta[j]~dnorm(mu0,tau0)
  }
  tau0=1/sigma02

  # Prior for the precision
  tau~dgamma(a, b)

  # Compute the variance
  sigma2=1/tau
}"
```

Regression

Linear regression

- We will now pass all information needed and compile the model.

```
#hyperparameters for the betas and tau  
mu0=0; sigma02=1000; a=0.1; b=0.1  
  
# list with data and hyperparameters  
data=list (y=y,drat=drat,wt=wt,qsec=qsec,n=n,mu0=mu0,sigma02=sigma02,a=a,b=b)  
  
#Compile the model  
model=jags.model(textConnection(model_string),n.chains=1,data=data)  
  
Compiling model graph  
  Resolving undeclared variables  
  Allocating nodes  
Graph information:  
  Observed stochastic nodes: 32  
  Unobserved stochastic nodes: 5  
  Total graph size: 254  
  
Initializing model
```

- Note that we have 5 unobserved stochastic nodes (=variables): $\beta_0, \beta_1, \beta_2, \beta_3$, and σ^2 . The 32 observed stochastic nodes correspond to the 32 y's.

Regression

Linear regression

→ We will do a burn-in of 100 000 iterations, run the model for 500 000 iterations, and do a thinning of 50.

```
update(model, 100000, progress.bar="none")
res=coda.samples(model, variable.names=c("beta", "sigma2"), n.iter=500000, thin=50, progress.bar=
summary(res)
```

```
Iterations = 100050:6e+05
Thinning interval = 50
Number of chains = 1
Sample size per chain = 10000
```

1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:

	Mean	SD	Naive SE	Time-series SE
beta[1]	10.3694	8.0796	0.080796	0.234179
beta[2]	1.7771	1.2427	0.012427	0.029145
beta[3]	-4.3354	0.6834	0.006834	0.015359
beta[4]	0.9677	0.2675	0.002675	0.005713
sigma2	6.9785	1.9893	0.019893	0.020370

2. Quantiles for each variable:

	2.5%	25%	50%	75%	97.5%
beta[1]	-5.0983	5.0167	10.3319	15.617	26.349
beta[2]	-0.7206	0.9583	1.7916	2.622	4.166
beta[3]	-5.7024	-4.7828	-4.3340	-3.885	-2.995
beta[4]	0.4487	0.7894	0.9673	1.149	1.493
sigma2	4.1601	5.5733	6.6201	8.016	11.729

Regression

Linear regression

- ↪ Note that the above results might slightly change when you run the model on your computer (stochastic nature of the MCMC).
- ↪ The chains for the regression coefficients were highly correlated, and that is why we have thinned them.
- ↪ The effective sample sizes for each parameter (β_0 , β_1 , β_2 , β_3 , and σ^2) are: 1190, 1817, 1979, 2192, and 9536, respectively.
- ↪ All convergence diagnostics seem ok!
- ↪ As an exercise, try centering and scaling each covariate, i.e., for each covariate subtract its mean and divide by its standard deviation. This can be done by redefining the covariates, e.g.,

```
drat=(mtcars1$drat-mean(mtcars1$drat))/sqrt(var(mtcars1$drat));
```
- ↪ Look at autocorrelations again (no thinning). Does centring and scaling help increase the effective sample size?

Regression

Linear regression

↪ Under the frequentist analysis we have

Parameter	Point Estimate (95% confidence interval)
β_0	11.395 (-5.134, 27.922)
β_1	1.750 (-0.857, 4.169)
β_2	-4.347 (-5.787, -3.009)
β_3	0.946 (0.410, 1.482)
σ^2	6.554

↪ For the Bayesian version results are as follows

Parameter	Posterior mean (95% credible interval)
β_0	10.369 (-5.098, 26.349)
β_1	1.777 (-0.721, 4.166)
β_2	-4.335 (-5.702, -2.995)
β_3	0.968 (0.449, 1.493)
σ^2	6.978 (4.160, 11.729)

↪ Results are pretty similar! However, interpretation of the intervals are completely different.

Regression

Linear regression

- The regression model described earlier can be defined in JAGS using vectors and matrices instead of scalar nodes, by using the function `inprod`.

```
x=cbind(rep(1,n),drat,wt,qsec)

# Likelihood
for(i in 1:n){
y[i]~dnorm(mu[i],tau)
mu[i]=inprod(beta[],x[i,])
}

# Prior for beta
for(j in 1:4){
beta[j]~dnorm(mu0,tau0)
}
tau0=1/sigma02

# Prior for the inverse variance
tau~dgamma(a, b)

# Compute the variance
sigma2=1/tau
}"
```

mu0=0; sigma02=1000; a=0.1; b=0.1
data=list(y=y, x=x, n=n, mu0=mu0, sigma02=sigma02, a=a, b=b)

- Fitting the model in JAGS then proceeds as before.

Regression

Linear regression

- We can also use multivariate normal priors for the regression coefficients $\beta_{p+1} \sim N(\mu_\beta, \Sigma_\beta)$, using the function `dmmnorm`.
- The prior we used before is a particular case of this more general prior, in which the off-diagonal elements are all zero).

```
model_string <- "model{

# Likelihood
for(i in 1:n){
y[i]~dnorm(mu[i],inv.var)
mu[i]=inprod(beta[],x[i,])
}

beta~dmmnorm(mu.beta,tau.beta)

# Prior for the inverse variance
inv.var~dgamma(a, b)

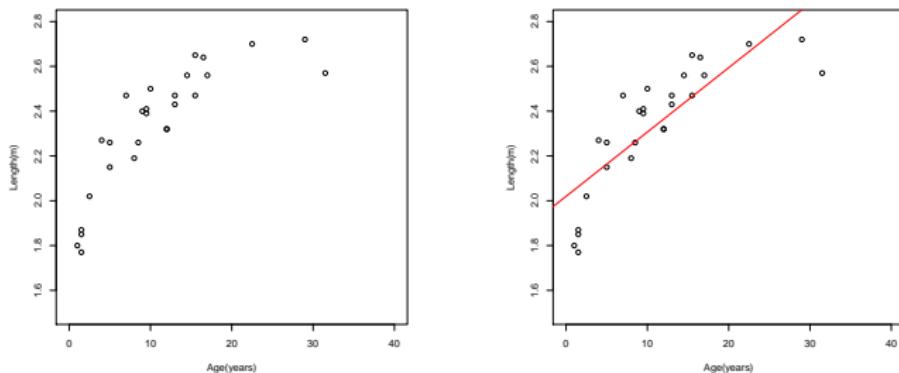
# Compute the variance
sigma2=1/inv.var
}"

#this coincides with the previous prior, as off diagonal entries are zero
mu.beta=rep(0,4); tau.beta=diag(0.0001,4); a=0.001; b=0.001
data=list(y=y,x=x,n=n,mu.beta=mu.beta,tau.beta=tau.beta,a=a,b=b)
```

Regression

Nonlinear regression

- Another desirable extension is the ability to fit a nonlinear model.
- Carlin and Gelfand (1991) consider data on length (y_i) and age (x_i) measurements for $i = 1, \dots, n = 27$ dugongs (sea cows) captured off the coast of Queensland.



- A linear fit to this dataset (shown on the right) clearly does not fit the behaviour for young or old dugongs.

Regression

Nonlinear regression - Dugongs



Regression

Nonlinear regression

- The following nonlinear growth model has been suggested

$$y_i = \alpha - \beta\gamma^{x_i} + \varepsilon_i, \quad i = 1, \dots, n,$$

where $\alpha > 0$, $\beta > 0$, $0 \leq \gamma \leq 1$ and $\varepsilon_i \stackrel{\text{iid}}{\sim} N(0, \sigma^2)$.

- In this model, as explained in Carlin and Louis (2009, p.174), α corresponds to the average length of a fully grown dugong, $(\alpha - \beta)$ is the length of a dugong at birth, and γ determines the growth rate.
- Specifically, lower values of γ produce an initially steep growth curve (rapid progression to adulthood immediately after birth), while higher γ values lead to much more gradual, almost linear growth.
- As stated by these authors, flat priors are suitable for the two “endpoint” parameters α and β , but the harder-to-estimate growth parameter γ benefits from a tighter (albeit uniform) specification.

Regression

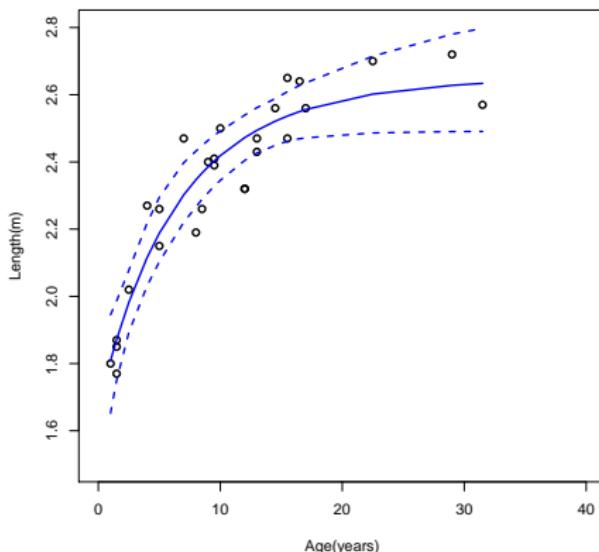
Nonlinear regression

→ The code to specify the model is

```
model_string <- "model{  
  # Likelihood  
  for(i in 1:n){  
    y[i]~dnorm(mu[i],tau)  
    mu[i]=alpha-beta*pow(gamma,x[i])  
  }  
  
  alpha~dunif(0,10)  
  
  beta~dunif(0,10)  
  
  gamma~dunif(0.5,1)  
  
  # Prior for the inverse variance  
  tau~dgamma(a, b)  
  
  # Compute the variance  
  sigma2=1/tau  
}"
```

Regression

Nonlinear regression



→ **Additional reading:** see Sections 6.6.2-6.6.5 of Core Statistics by Simon Wood for more examples on JAGS.

Model checking

Linear regression: model checking

- ↪ The conclusions of any (Bayesian) analysis are conditional on the appropriateness of the assumed statistical model, so we need to be satisfied that our assumptions are a reasonable approximation to reality, even though we do not generally believe any model is actually “true” (Lunn et al. 2012).
- ↪ In defining the multiple linear regression model we have assumed
 - ① **Normality:** the error terms $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$ are normally distributed, with mean 0 and variance σ^2 .
 - ② **Independence:** the error terms $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$ are independent of one another.
 - ③ **Constant variability:** the value of σ^2 does not depend on the values x_{ij} (this is known as homoskedasticity).
 - ④ **Linearity:** the mean μ_i is a linear function of x_{i1}, \dots, x_{ip} .
- ↪ All of these assumptions can be (informally) checked using the Studentized residuals.

Model checking

Linear regression: model checking

- Studentized residuals are often used in frequentist analyses and are defined by

$$\hat{\varepsilon}_i = \frac{y_i - \hat{y}_i}{\hat{\sigma} \sqrt{1 - h_{ii}}}, \quad i = 1, \dots, n$$

where $\hat{y}_i = \sum_{j=0}^p \hat{\beta}_j x_{ij}$ and h_{ii} is the i -th diagonal entry of the 'hat' matrix $H = \mathbf{x}(\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T$.

- This is effectively a rescaling, as $\hat{\sigma} \sqrt{1 - h_{ii}}$ is an estimate of the standard deviation of the residual $y_i - \hat{y}_i$ (\hat{y}_i depends on y_i , hence the correction term $\sqrt{1 - h_{ii}}$).
- Under the assumption of normality of the error terms ε_i , we have the approximation

$$\{\hat{\varepsilon}_i\} \sim N(0, 1)$$

Note: this is an approximation. Although the errors $\{\varepsilon_i\}$ should be independent under our model assumptions, the Studentized residuals $\{\hat{\varepsilon}_i\}$ are not necessarily independent, so tests of Normality are not applied rigorously - it is usually enough to inspect graphs, etc.

Model checking

Linear regression: model checking

- In the Bayesian context, the point estimates $\hat{\beta}_j$ and $\hat{\sigma}$ are replaced by distributions so the Studentized residuals are not single-valued. Lunn et al. (2012, p.141) suggest

"If we want to create a single-valued residual rather than a random quantity, alternative possibilities include using a single draw $\theta^{(j)} = (\beta^{(j)}, \sigma^{(j)})$, plugging in the posterior means of θ , or using the posterior mean residual."
- Here, we opt for the last option (the posterior mean of the residual), that is, for each draw $(\beta^{(j)}, \sigma^{(j)})$ from the posterior distribution, we compute the Studentized residuals and from this ensemble the posterior mean (or median if preferred) is computed.
- The same option is taken by Wang et al. (Bayesian Regression Modeling with INLA, 2018, CRC/Chapman & Hall).

Model checking

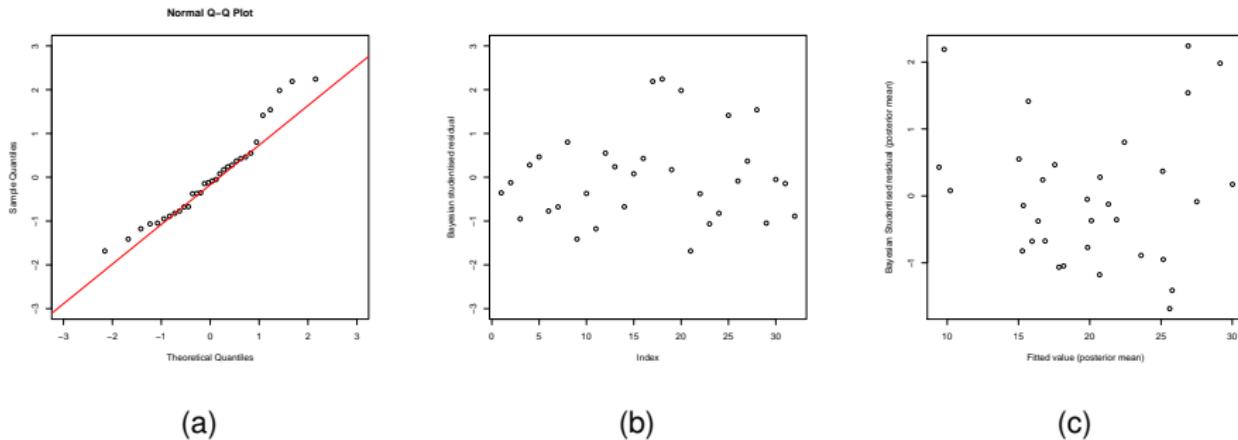
Linear regression: model checking

- ↪ To assess the normality assumption, a Q-Q plot of the Bayesian Studentized residuals must be constructed.
- ↪ To assess the independence of the error terms, a plot of the Bayesian Studentized residual against the index, i , that labels the order in which the observations were gathered, can be used. Any pattern suggests that the errors $\{\varepsilon_i\}_{i=1}^n$ are not independent.
- ↪ To check the assumptions of linearity and constant variance we can plot the Bayesian Studentized residuals against the posterior mean (or median) of the fitted values. This graph should look random, with no obvious patterns, if these two assumptions are valid.
- ↪ Interpreting these plots is, of course, very subjective...

Model checking

Linear regression: model checking

- For the mtcars dataset, the following figures were produced



- Figure (a) shows a normal Q-Q plot, checking **Normality** of the studentized residuals. Figure (b) plots the studentized residuals versus their index, showing a random behaviour with mean close to 0, meaning that the **Linearity** and **Independence** assumptions seem to hold. Figure (c) plots the posterior mean of the fitted value (of the response variable) versus the posterior mean of the studentized residual. Residuals do not show clear dependence on the fitted value, justifying **Constant variability** assumption.

Model checking

Linear regression: model checking

- ↪ Predictive checks are also very popular in Bayesian statistics. Basically, for an overall assessment of model fit we can generate replicate data sets (of the same size as the original data) from the posterior predictive distribution, and compare to the real data using specific test quantities (Gelman et al. 2013, chapter 6).
- ↪ Let \mathbf{y}_{rep} denotes the replicated datasets. The distribution of \mathbf{y}_{rep} is the posterior predictive distribution

$$p(\mathbf{y}_{\text{rep}} \mid \mathbf{y}) = \int p(\mathbf{y}_{\text{rep}} \mid \theta)p(\theta \mid \mathbf{y})d\theta.$$

- ↪ We do not usually evaluate this integral but instead draw samples from $p(\mathbf{y}_{\text{rep}} \mid \mathbf{y})$:

- 1 Draw $\theta^{(j)}$ from $p(\theta \mid \mathbf{y})$, $j = 1, \dots, M$.
- 2 Draw $\mathbf{y}_{\text{rep}}^{(j)} \sim p(\mathbf{y}_{\text{rep}} \mid \theta^{(j)})$, $j = 1, \dots, M$,

where M is the number of MCMC iterations.

- ↪ The chosen test statistics are then computed for each replicated dataset and compared to the one derived from the observed data.

Model checking

Linear regression: model checking

- ↪ We distinguish between the replicated data \mathbf{y}_{rep} and the predictive outcomes $\tilde{\mathbf{y}}$.
- ↪ The variable $\tilde{\mathbf{y}}$ is any *future* observable value of the outcome. For example, in a linear regression model $\tilde{\mathbf{y}}$ can have its own set of explanatory variables $\tilde{\mathbf{x}}$.
- ↪ On the other hand, \mathbf{y}_{rep} must have the same explanatory variables \mathbf{x} as those used in the model for the observed data \mathbf{y} . In this sense, \mathbf{y}_{rep} is similar to “predicting the observed data”.

Model checking

Linear regression: model checking

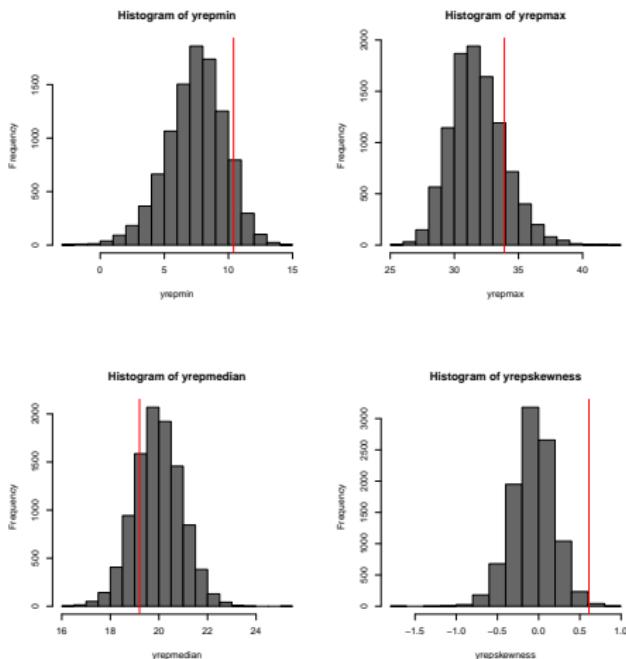
- ↪ A very good and concise discussion about this topic can be found in the paper

<https://arxiv.org/pdf/1709.01449.pdf>

- ↪ As the authors of the aforementioned paper state: "*Posterior predictive checking makes use of the data twice, once for the fitting and once for the checking. Therefore it is a good idea to choose statistics that are orthogonal to the model parameters. If the test statistic is related to one of the model parameters, e.g., if the mean statistic is used for a Gaussian model with a location parameter, the posterior predictive checks may be less able to detect conflicts between the data and the model.*"
- ↪ For the `mtcars` example, since we are using a location-scale normal model, let us pick up as test statistics the minimum, the maximum, the median, and the skewness.

Model checking

Linear regression: model checking



- The solid red vertical lines indicate the corresponding statistic computed from the observed data.

Model checking

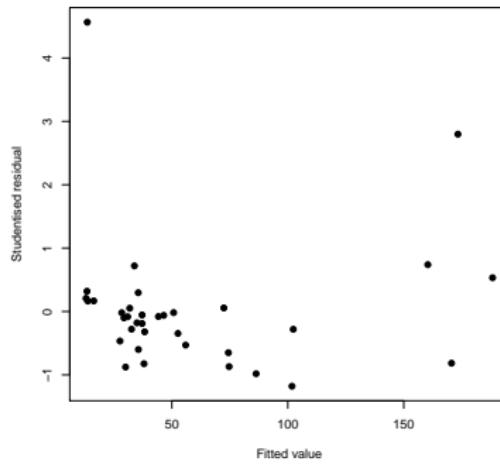
Robustifying linear regression

- ↪ Let us now consider robust regression. The normality assumption of the error terms renders the model quite sensitive to outliers, so we now consider modifications to that assumption.
- ↪ A classic data set in the literature pertains to hill racing (a somewhat popular sport here in Scotland).
- ↪ The data set `hills.txt` contains information on the winning times (in minutes) in 1984 for 35 Scottish hill races, as well as two factors which presumably influence the duration of the race:
 - ↪ `dist`: The distance of the race (in miles).
 - ↪ `climb`: The elevation change (in feet).

Model checking

Robustifying linear regression

- Fitting a simple linear regression model for time assuming additive linear relationships for dist and climb, we obtain the following residuals



- The point at the top left corner, seems suspicious

Model checking

Robustifying linear regression

- ↪ To reduce the impact of the outlier on the fit of the model, we can replace the normal distribution with a fat-tailed distribution.
- ↪ A natural choice is the t -distribution, which can be implemented by simply replacing the normal likelihood with

```
y[i] ~ dt(mu[i], tau, nu)
```

- ↪ Recall that as $\nu \rightarrow \infty$, the t -distribution resembles the normal, but for small ν has considerably fatter tails.
- ↪ Let us start by trying $\nu = 5$.

Model checking

Robustifying linear regression

↪ The code for defining this model is

```
hills=read.table("hills.txt",header=TRUE)
y=hills$time; climb=hills$climb; dist=hills$dist

model_string <- "model{

# Likelihood
for(i in 1:n){
y[i]~dt(mu[i],tau,5)
mu[i]=beta[1]+beta[2]*climb[i]+beta[3]*dist[i]
}

# Prior for beta
for(j in 1:3){
beta[j]~dnorm(mu0,tau0)
}
tau0=1/sigma02

# Prior for the inverse variance
tau~dgamma(a, b)

# Compute the variance
sigma2=1/tau
}"
```

Model checking

Robustifying linear regression

- Contrasting the results from a normal model versus a t_5 model (posterior mean and standard deviation)

	Climb	Dist
Normal	0.011 (0.0021)	6.204 (0.6150)
t_5	0.0081 (0.0015)	6.565 (0.3089)

- There is a modest change in the posterior means (the posterior mean for distance goes up, while the one for climb goes down), and a sizeable drop (roughly 2-fold) in the posterior standard deviation.

Model checking

Robustifying linear regression

- ↪ Of course, one may ask, why a t_5 distribution?
- ↪ Since we do not actually know ν , it would be more reasonable to include ν as a parameter in our model; the only condition is that we must place a prior on it.
- ↪ There are several possible options. For instance we could use a similar prior as the one used for the precision.

Model checking

Robustifying linear regression

→ The modified model definition is now

```
model_string <- "model{
  # Likelihood
  for(i in 1:n){
    y[i]~dt(mu[i],tau,nu)
    mu[i]=beta[1]+beta[2]*climb[i]+beta[3]*dist[i]
  }

  # Prior for beta
  for(j in 1:3){
    beta[j]~dnorm(mu0,tau0)
  }
  tau0=1/sigma02

  # Prior for the inverse variance
  tau~dgamma(a, b)

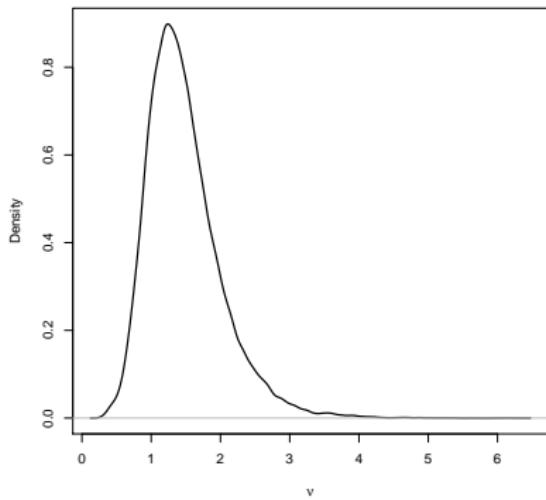
  # Compute the variance
  sigma2=1/tau

  #Prior for nu
  nu~dgamma(c, d)
}"

mu0=0; sigma02=1000; a=0.1; b=0.1; c=0.1; d=0.1
data=list(y=y,climb=climb,dist=dist,n=n,mu0=mu0,sigma02=sigma02,a=a,b=b,c=c,d=d)
```

Model checking

Robustifying linear regression



- The plot on the left shows the posterior density of the degrees of freedom parameter ν .
- The posterior mean (standard deviation) for regression coefficient of climb is now 0.0069 (0.0011).
- The posterior mean (standard deviation) for regression coefficient of dist is now 6.543 (0.300).
- **Additional reading:** see Section 7.2 of Core Statistics by Simon Wood and Sections 6-7 of Bayesian Data Analysis by Gelman et al. for various approaches to model checking and model comparison.

Bayesian Data Analysis

Daniel Paulin & Nicolò Margaritella

University of Edinburgh



Semester 2, 2020/2021

1

Introduction to INLA

- Latent Gaussian Models (LGM)
- Conditional probabilities
- Laplace approximation
- Integrated Nested Laplace Approximation

2

R-INLA Package

- General information
- Examples
- Posterior sampling in INLA
- Model checking in INLA

Happy Lunar New Year of the Ox!



Latent Gaussian Models (LGM)

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

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

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

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

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

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

Latent Gaussian Models (LGM)

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

$$\theta \sim \pi(\theta) \quad \text{(Hyperprior)}$$

$$\mathbf{x}|\theta \sim \pi(\mathbf{x}|\theta) = N(0, \Sigma(\theta^{(2)})) \quad \text{(GMRF prior)}$$

$$\mathbf{y}|\theta, \mathbf{x} \sim f(\mathbf{y}|\mathbf{x}, \theta) = \prod_{i=1}^n f(y_i|\mathbf{x}, \theta^{(1)}) \quad \text{(data model)}$$

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

Latent Gaussian Models (LGM)

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

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

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

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

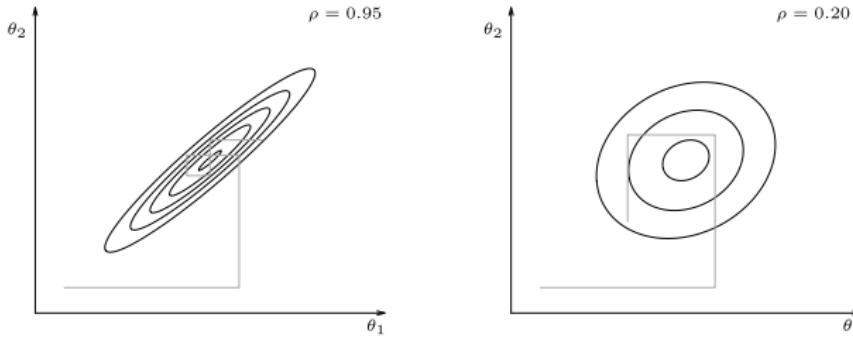
Latent Gaussian Models (LGM)

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

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

MCMC for LGMs

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



Conditional Probabilities

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

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

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

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

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

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

Laplace Approximation

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

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

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

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

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

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

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

Laplace Approximation - Example

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

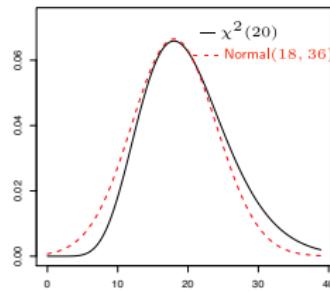
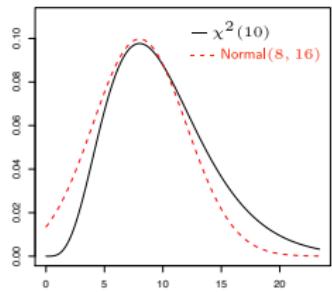
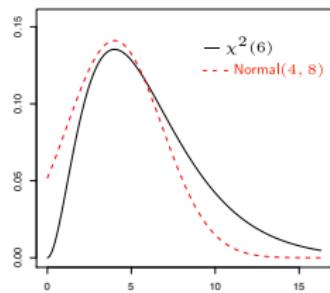
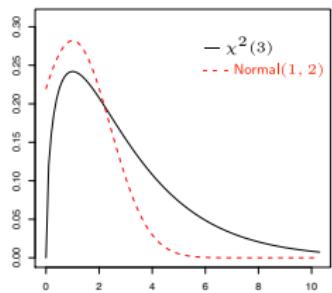
$$\log q(x) = \left(\frac{k}{2} - 1\right) \log(x) - \frac{x}{2}$$

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

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

- ↪ We can find the mode analytically in this case by solving $(\log q)'(x) = 0$. This leads to $\hat{x} = k - 2$, and $\hat{\sigma}^2 = -((\log q)''(\hat{x}))^{-1} = 2(k - 2)$.
- ↪ Thus the Laplace approximation in this case is $p \approx \tilde{p} = N(k - 2, 2(k - 2))$, i.e. a Normal distribution with mean $k - 2$ and variance $2(k - 2)$. This approximation is illustrated on the figures of the next slide.

Laplace Approximation - Example



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

Integrated Nested Laplace Approximation (INLA)

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

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

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

- This approximation can be used for any x , but since the Laplace approximation is most accurate near the mode, we will always use this at $x = \hat{x}(z, w)$ (i.e. the mode of $p(x|z, w)$).
- This idea is quite broadly applicable, and significantly extends the usefulness of Laplace approximation (i.e. $p(z|w)$ does not have to be approximately Gaussian, only $p(x|z, w)$).

Integrated Nested Laplace Approximation (INLA)

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

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

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

- In order to proceed, we will need to compute
 - (I) $p(\boldsymbol{\theta}|\mathbf{y})$, from which $p(\theta_j|\mathbf{y})$ can be obtained by integrating out $\boldsymbol{\theta}_{-j}$.
 - (II) $p(x_i|\boldsymbol{\theta}, \mathbf{y})$, which is needed for computing $p(x_i|\mathbf{y})$.

Integrated Nested Laplace Approximation (INLA)

→ (I) can be estimated as

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

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

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

Integrated Nested Laplace Approximation (INLA)

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

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

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



Integrated Nested Laplace Approximation (INLA)

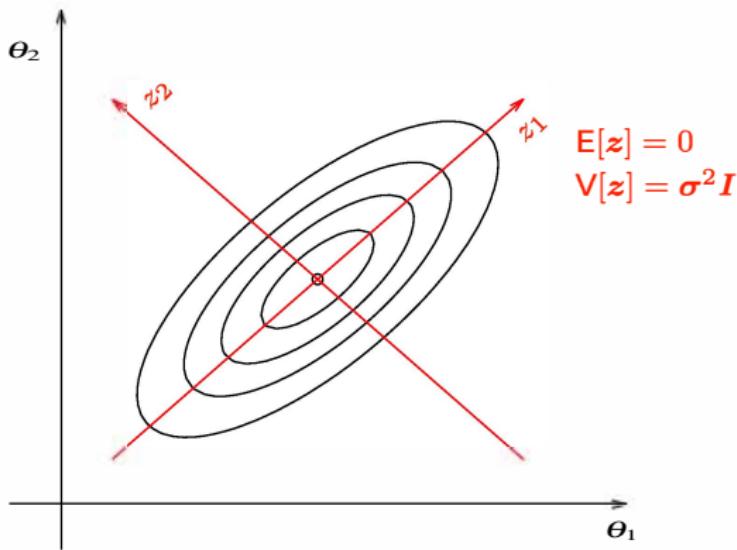
- The approximations $\underline{p}(x_j|\theta, \mathbf{y})$ and $\underline{p}(\theta|\mathbf{y})$ detailed above are called Integrated Nested Laplace Approximation because the Laplace approximation is only applied within the denominator, not directly on the distributions $p(x_j|\theta, \mathbf{y})$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

Steps in INLA's Operation

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

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

Standardizing the variables by change of coordinates



Steps in INLA's Operation

- (II) Explore $\log \underline{p}(\theta|\mathbf{y})$ and produce H points $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(H)}$ associated with the bulk of the mass of the density $\underline{p}(\theta|\mathbf{y})$, together with the corresponding area weights $\Delta^{(1)}, \Delta^{(2)}, \dots, \Delta^{(H)}$. In addition to this, for each point $\theta^{(h)}$ for $1 \leq h \leq H$, we
- ① Evaluate the marginal posterior of the hyperparameter $\underline{p}(\theta^{(h)}|\mathbf{y})$.
 - ② Evaluate the marginals of the latent field $\underline{p}(x_j|\theta^{(h)}, \mathbf{y})$ on a grid of selected values of x_j , for every j (either simplified Laplace approximation or full Laplace approximation can be used here).
- (III) Obtain the marginals $p(\theta_i|\mathbf{y})$ at selected gridpoints for θ_i by interpolation, using $\theta^{(1)}, \dots, \theta^{(H)}$ and $\underline{p}(\theta^{(1)}|\mathbf{y}), \dots, \underline{p}(\theta^{(H)}|\mathbf{y})$.
- (IV) Obtain the marginals $\underline{p}(x_j|\mathbf{y})$ at selected gridpoints for x_j by

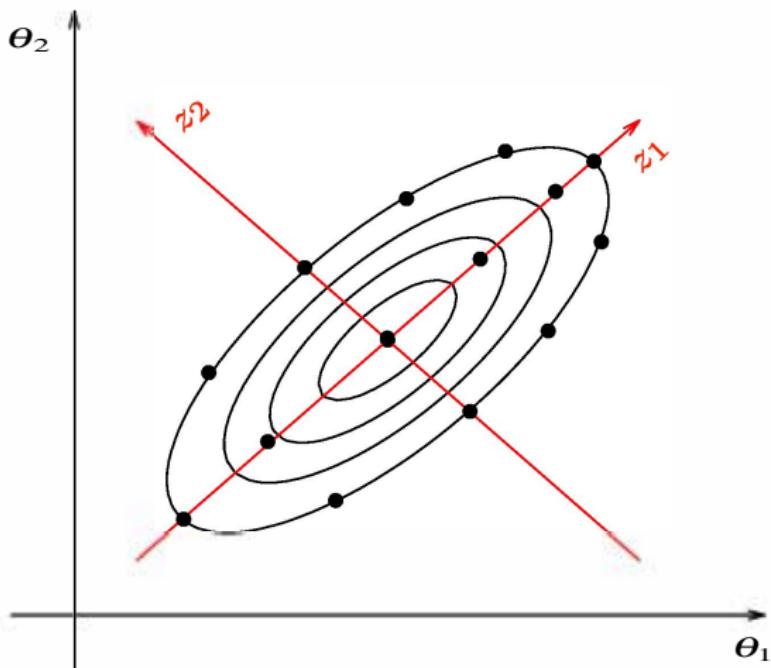
$$\underline{p}(x_j|\mathbf{y}) \approx \sum_{h=1}^H \underline{p}(x_j|\theta^{(h)}, \mathbf{y}) \underline{p}(\theta^{(h)}|\mathbf{y}) \Delta^{(h)}.$$

Steps in INLA's Operation

- (II) Explore $\log \underline{p}(\theta|\mathbf{y})$ and produce H points $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(H)}$ associated with the bulk of the mass of the density $\underline{p}(\theta|\mathbf{y})$, together with the corresponding area weights $\Delta^{(1)}, \Delta^{(2)}, \dots, \Delta^{(H)}$. In addition to this, for each point $\theta^{(h)}$ for $1 \leq h \leq H$, we
- ① Evaluate the marginal posterior of the hyperparameter $\underline{p}(\theta^{(h)}|\mathbf{y})$.
 - ② Evaluate the marginals of the latent field $\underline{p}(x_j|\theta^{(h)}, \mathbf{y})$ on a grid of selected values of x_j , for every j (either simplified Laplace approximation or full Laplace approximation can be used here).
- (III) Obtain the marginals $p(\theta_i|\mathbf{y})$ at selected gridpoints for θ_i by interpolation, using $\theta^{(1)}, \dots, \theta^{(H)}$ and $\underline{p}(\theta^{(1)}|\mathbf{y}), \dots, \underline{p}(\theta^{(H)}|\mathbf{y})$.
- (IV) Obtain the marginals $\underline{p}(x_j|\mathbf{y})$ at selected gridpoints for x_j by

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

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



Example for INLA's Operation

→ Consider the following simple model

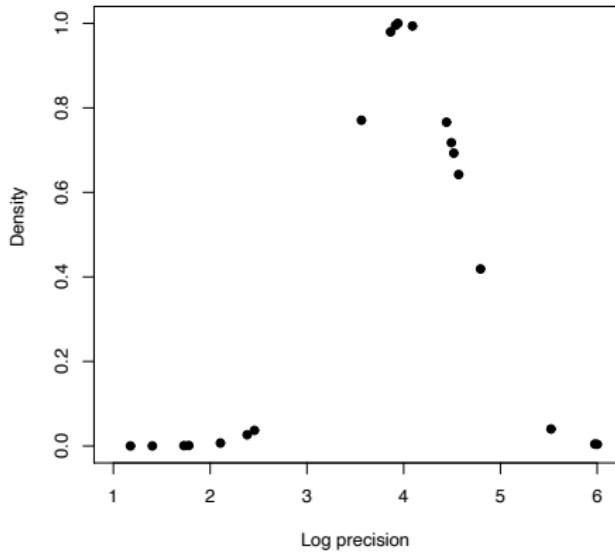
$$y_{ij} | \mathbf{x}, \theta \sim N(x_j, \sigma_0^2) \text{ for } 1 \leq i \leq n, 1 \leq j \leq n_j \quad (\sigma_0^2 \text{ is known})$$

$$x_j | \theta \sim N(0, \theta^{-1}) \quad (\theta \text{ corresponds to the precision})$$

$$\theta \sim \Gamma(a, b)$$

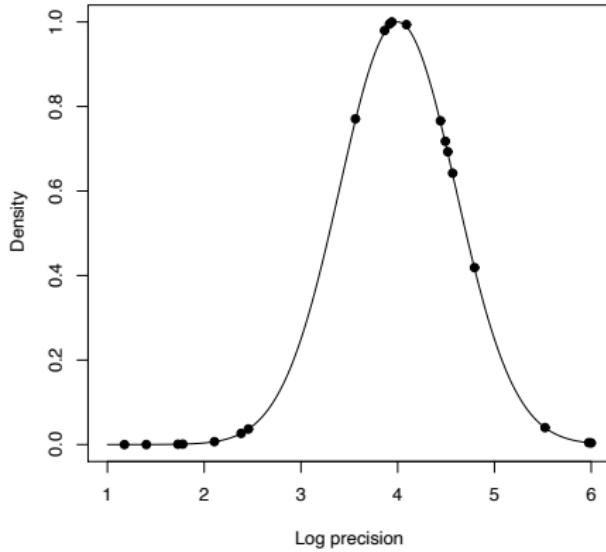
→ In the following figures, the INLA approximation for computing the posterior marginal approximations $\underline{p}(\theta|\mathbf{y})$ and $\underline{p}(x_i|\mathbf{y})$ is illustrated.

Example for INLA's Operation



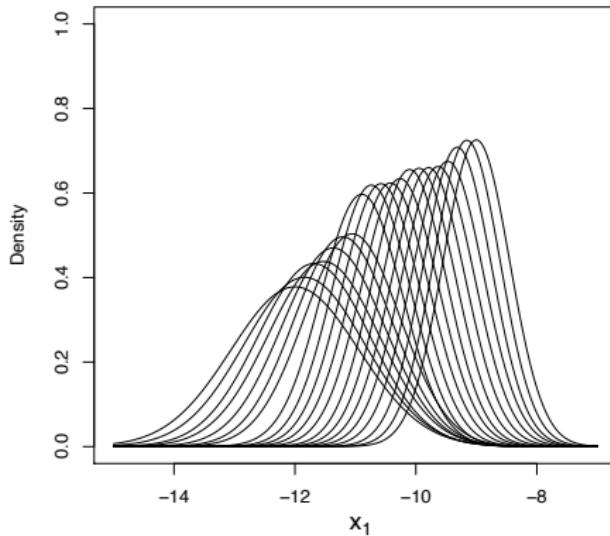
$p(\theta|\mathbf{y})$ computed at H gridpoints $\theta^{(1)}, \dots, \theta^{(H)}$

Example for INLA's Operation



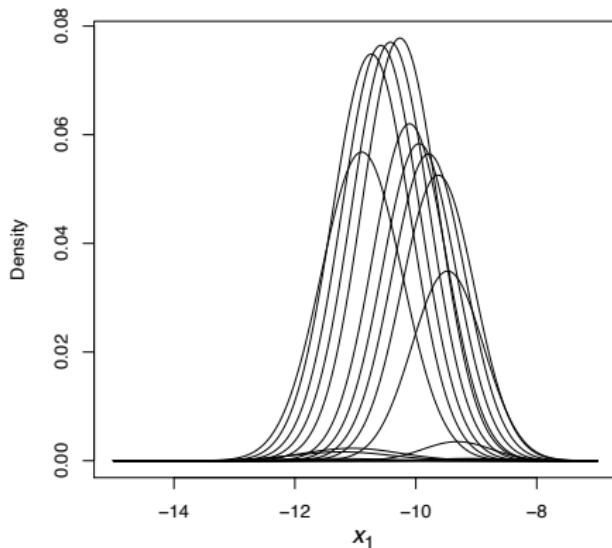
$\underline{p(\theta|\mathbf{y})}$ interpolated

Example for INLA's Operation



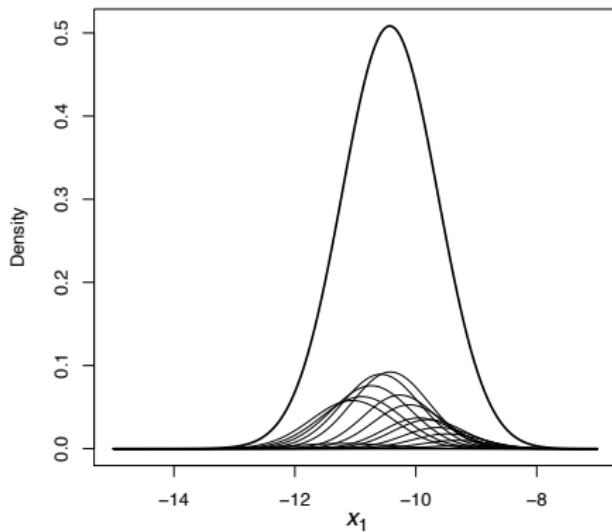
$p(x_1 | \theta^{(h)}, \mathbf{y})$ is computed for each gridpoint $1 \leq h \leq H$

Example for INLA's Operation



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

Example for INLA's Operation



$p(x_1|y)$ obtained by summing up the weighted conditional densities

Summary of INLA

→ The basic idea behind INLA is quite simple.

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

Some possible complications:

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

The INLA package for R

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

The INLA package for R

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

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

where

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

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

Example: Bayesian Linear regression (mtcars dataset)

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

$$y_i \mid \mu_i, \sigma^2 \stackrel{\text{ind}}{\sim} \mathcal{N}(\mu_i, \sigma^2), \quad i = 1, \dots, n,$$

$$\mu_i = \beta_0 + \sum_{j=1}^{n_\beta} \beta_j z_{j,i},$$

$$\beta_j \sim \mathcal{N}(\mu_{\beta_j}, \sigma_{\beta_j}^2), \quad j = 0, \dots, p,$$

$$\tau \sim \text{Gamma}(a, b).$$

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

Example: Bayesian Linear regression (mtcars dataset)

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

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

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

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

#This displays the summary statistics
summary(m.mtcars.I.defaultprior)
```

Example: Bayesian Linear regression (mtcars dataset)

- The summary statistics obtained after fitting the model:

Call:

```
"inla(formula = mpg ~ drat + wt + qsec, family = \"gaussian\", data =
mtcars1)"
```

Time used:

Pre = 0.427, Running = 0.0822, Post = 0.0343, Total = 0.543

Fixed effects:

	mean	sd	0.025quant	0.5quant	0.975quant	mode	kld
(Intercept)	11.390	8.037	-4.497	11.390	27.265	11.391	0
drat	1.656	1.222	-0.759	1.656	4.069	1.656	0
wt	-4.397	0.675	-5.732	-4.397	-3.063	-4.397	0
qsec	0.946	0.261	0.431	0.946	1.461	0.946	0

Model hyperparameters:

	mean	sd	0.025quant	0.5quant
Precision for the Gaussian observations	0.163	0.042	0.092	0.16
			0.975quant	mode

Precision for the Gaussian observations 0.256 0.153

Expected number of effective parameters(stdev) : 4.00(0.001)

Number of equivalent replicates : 8.00

Marginal log-Likelihood: -97.81

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

Example: Bayesian Linear regression (mtcars dataset)

- ↪ We set the priors:

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

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

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

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

Example: Bayesian Linear regression (mtcars dataset)

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

Call:

```
c("inla(formula = mpg ~ drat + wt + qsec, family = \"gaussian\", data =
  mtcars1, ", " control.family = list(hyper = prec.prior), control.fixed
  = prior.beta) " )
```

Time used:

Pre = 0.349, Running = 0.126, Post = 0.0298, Total = 0.505

Fixed effects:

	mean	sd	0.025quant	0.5quant	0.975quant	mode	kld
(Intercept)	10.659	8.013	-5.238	10.681	26.419	10.724	0
drat	1.741	1.236	-0.694	1.739	4.190	1.734	0
wt	-4.351	0.684	-5.698	-4.352	-2.996	-4.355	0
qsec	0.962	0.265	0.438	0.961	1.487	0.960	0

Model hyperparameters:

	mean	sd	0.025quant	0.5quant
Precision for the Gaussian observations	0.154	0.041	0.084	0.15
			0.975quant	mode

Precision for the Gaussian observations	0.244	0.143
---	-------	-------

Expected number of effective parameters(stdev): 3.93(0.018)

Number of equivalent replicates : 8.14

Marginal log-Likelihood: -93.19

Working with marginals

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

```
names(m.mtcars.I)
-----
```

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

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

```
names(m.mtcars.I$marginals.fixed)
-----
```

```
'(Intercept)', 'drat', 'wt', 'qsec'
```

We can access the marginals of the regression coefficient β_0 as

`m.mtcars.I$marginals.fixed$(Intercept)`, or equivalently

`m.mtcars.I$marginals.fixed[[1]]`, and similarly for the other coefficients.

Working with marginals

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

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

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

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

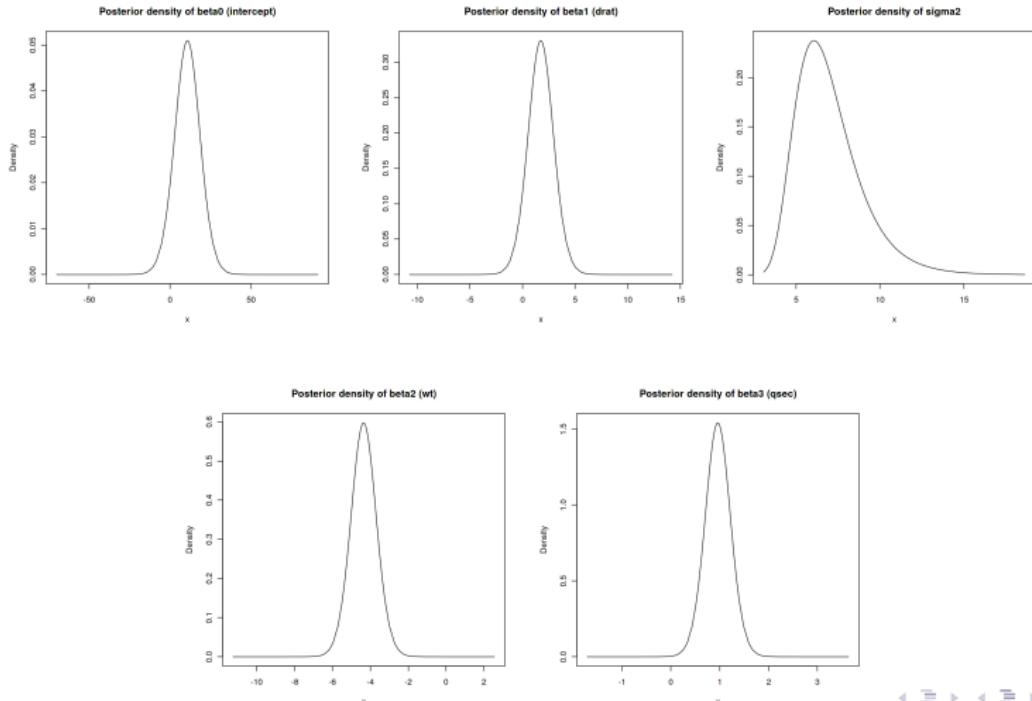
```
plot(m.mtcars.I$marginals.fixed$qsec',type='l',xlab="x",
ylab="Density",main="Posterior density of beta3 (qsec)")
```

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

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

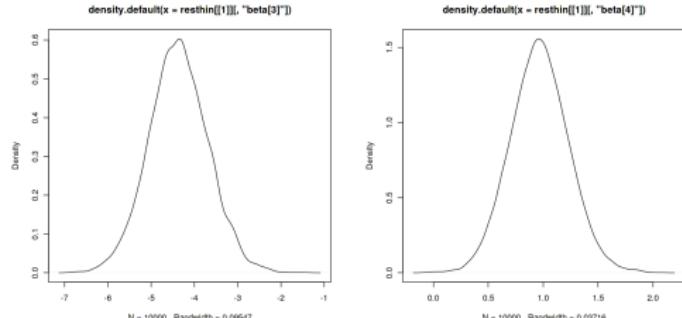
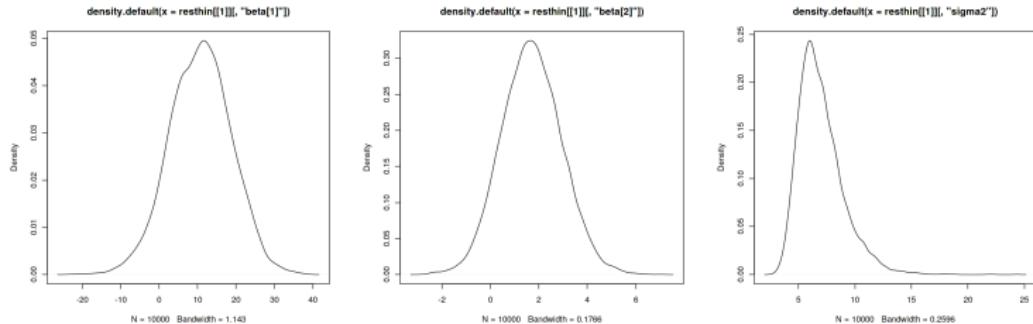
Working with marginals

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



Working with marginals

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



Working with marginals

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

```
cat("Summary statistics of sigma2\n")  
inla.zmarginal(marginal.sigma2)  
-----
```

```
Summary statistics of sigma2  
Mean           6.98976  
Stdev          1.97818  
Quantile  0.025 4.10781  
Quantile  0.25  5.58166  
Quantile  0.5   6.65383  
Quantile  0.75  8.01779  
Quantile  0.975 11.805
```

- As a comparison, these were the same summary statistics for σ^2 obtained by JAGS:

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

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

Working with marginals

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

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

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

- These four functions evaluate the density, the CDF, the quantiles, and generate n random samples from the marginal distribution.
- We can compute the expected value of an arbitrary function according to the marginal using `inla.emarginal(fun, marginal)`.
- There are several other functions available, see <https://rdrr.io/github/andrewzm/INLA/man/marginal.html> for a complete list.

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

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

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

Available likelihoods in INLA, page 1

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

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

Available likelihoods in INLA, page 2

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

Available likelihoods in INLA, page 3

weibull	The Weibull likelihood
weibullcure	The Weibull-cure likelihood (survival)
weibullsurv	The Weibull likelihood (survival)
wrappedcauchy	The wrapped Cauchy likelihood
xpoisson	The Poisson likelihood (expert version)
zeroinflatedbetabinomial0	Zero-inflated Beta-Binomial, type 0
zeroinflatedbetabinomial1	Zero-inflated Beta-Binomial, type 1
zeroinflatedbetabinomial2	Zero inflated Beta-Binomial, type 2
zeroinflatedbinomial0	Zero-inflated Binomial, type 0
zeroinflatedbinomial1	Zero-inflated Binomial, type 1
zeroinflatedbinomial2	Zero-inflated Binomial, type 2
zeroinflatednbnomial0	Zero inflated negBinomial, type 0
zeroinflatednbnomial1	Zero inflated negBinomial, type 1
zeroinflatednbnomial1strata2	Zero inflated negBinomial, type 1, strata 2
zeroinflatednbnomial1strata3	Zero inflated negBinomial, type 1, strata 3
zeroinflatednbnomial2	Zero inflated negBinomial, type 2
zeroinflatedpoisson0	Zero-inflated Poisson, type 0
zeroinflatedpoisson1	Zero-inflated Poisson, type 1
zeroinflatedpoisson2	Zero-inflated Poisson, type 2
zeroninflatedbinomial2	Zero and N inflated binomial, type 2
zeroninflatedbinomial3	Zero and N inflated binomial, type 3

→ You can use these by writing `family = "name of likelihood"` when calling `inla`.

→ More info and examples for each likelihood is available at

<https://inla.r-inla-download.org/r-inla.org/doc/likelihood/>.

Available priors for hyperparameters, page 1

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

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

Available priors hyperparameters, page 2

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

- Specified by `control.family=list(hyper = list(hyperparameter = list(prior="prior name", param= parameter values)))` in `inla`. Important to understand the internal parametrisation of the likelihood model (explained in the documentation of the likelihood), as priors need to be specified on the internal parameters.
- More info and examples: <https://inla.r-inla-download.org/r-inla.org/doc/prior/>.

Available latent effects models (GMRF priors), page 1

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

ar	Auto-regressive model of order p (AR(p))
ar1	Auto-regressive model of order 1 (AR(1))
arlc	Auto-regressive model of order 1 w/covariates
besag	The Besag area model (CAR-model)
besag2	The shared Besag model
besagproper	A proper version of the Besag model
besagproper2	An alternative proper version of the Besag model
bym	The BYM-model (Besag-York-Mollier model)
bym2	The BYM-model with the PC priors
clinear	Constrained linear effect
copy	Create a copy of a model component
crw2	Exact solution to the random walk of order 2
dmatern	Dense Matern field
fgn	Fractional Gaussian noise model
fgn2	Fractional Gaussian noise model (alt 2)
generic	A generic model
generic0	A generic model (type 0)
generic1	A generic model (type 1)
generic2	A generic model (type 2)
generic3	A generic model (type 3)
iid	Gaussian random effects in dim=1
iidl1d	Gaussian random effect in dim=1 with Wishart prior
iid2d	Gaussian random effect in dim=2 with Wishart prior
iid3d	Gaussian random effect in dim=3 with Wishart prior
iid4d	Gaussian random effect in dim=4 with Wishart prior

Available latent effects models (GMRF priors), page 2

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

- Specified by `y ~ x1+...+xn+f(covariates, model="name of latent model")` in `formula` in `inla`.
- More info and examples: <https://inla.r-inla-download.org/r-inla.org/doc/latent/>.

Example: Robust regression (Scottish hill racing data)

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

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

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

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

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

Example: Robust regression (Scottish hill racing data)

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

Call:

```
c("inla(formula = time ~ 1 + climb + dist, family = \"T\", data =  
hills, ", " control.family = list(hyper = prior.t), control.fixed =  
prior.fixed) " )
```

Time used:

```
Pre = 0.349, Running = 0.111, Post = 0.0296, Total = 0.49
```

Fixed effects:

	mean	sd	0.025quant	0.5quant	0.975quant	mode	kld
(Intercept)	-9.526	2.276	-14.285	-9.439	-5.271	-9.276	0
climb	0.008	0.001	0.006	0.008	0.011	0.008	0
dist	6.582	0.266	6.028	6.590	7.087	6.607	0

Model hyperparameters:

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

Expected number of effective parameters(stdev): 3.00(0.002)

Number of equivalent replicates : 11.69

Marginal log-Likelihood: -149.73

Example: Poisson regression (ship incident data)

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

```

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

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

```



Example: Poisson regression (ship incident data)

- A simple Poisson regression model can be written as follows. For every observation $1 \leq i \leq n$,

$$y_i \sim \text{Poisson}(\lambda_i)$$

$$\eta_i = \log(\lambda_i) = \beta_0 + \sum_{j=1}^{n_\beta} \beta_j z_{ji},$$

where z_{ji} are covariates, and $\boldsymbol{\beta} = (\beta_0, \dots, \beta_{n_\beta})$ are the regression coefficients (fixed effects).

- The mean of observation i is $\mathbb{E}(y_i|\boldsymbol{\beta}) = \lambda_i$, which is linked to the linear predictor η_i through the log link function $\eta_i = \log(\lambda_i)$. This fits in the LGM framework.
- In such Poisson regression models, y_i typically refers to the number of events during a time period. We often encounter situations where the time period T_i is different for different observations y_i . In such cases, it is more appropriate to use the slightly modified model

$$y_i \sim \text{Poisson}(T_i \rho_i)$$

$$\eta_i = \log(T_i \rho_i) = \beta_0 + \sum_{j=1}^{n_\beta} \beta_j z_{ji} + \log(T_i).$$

Example: Poisson regression (ship incident data)

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

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

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

Example: Poisson regression (ship incident data)

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

Call:

```
c("inla(formula = formula.inla, family = \"poisson\", data =  
  ShipsIncidents, ", " E = months)")
```

Time used:

```
Pre = 0.47, Running = 0.0936, Post = 0.0294, Total = 0.593
```

Fixed effects:

	mean	sd	0.025quant	0.5quant	0.975quant	mode	kld
(Intercept)	-6.416	0.217	-6.852	-6.413	-5.998	-6.406	0
typeB	-0.543	0.178	-0.882	-0.546	-0.185	-0.553	0
typeC	-0.689	0.329	-1.366	-0.677	-0.075	-0.655	0
typeD	-0.075	0.290	-0.664	-0.069	0.476	-0.055	0
typeE	0.326	0.236	-0.141	0.327	0.785	0.330	0
built65-69	0.696	0.150	0.406	0.695	0.993	0.692	0
built70-74	0.818	0.170	0.487	0.818	1.153	0.816	0
built75-79	0.453	0.233	-0.012	0.455	0.904	0.460	0
oper75-79	0.384	0.118	0.153	0.384	0.617	0.383	0

Expected number of effective parameters(stdev): 9.00(0.00)

Number of equivalent replicates : 3.78

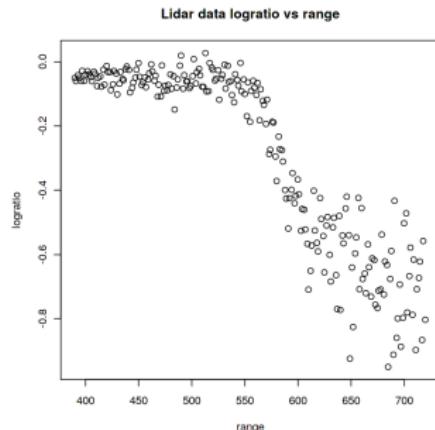
Marginal log-Likelihood: -111.81

Example: Poisson regression (ship incident data)

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

Example: Smoothing (Lidar data)

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



Example: Smoothing (Lidar data)

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

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

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

Example: Smoothing (Lidar data)

→ By printing out the summary, we obtain

Call:

```
c("inla(formula = logratio ~ 1 + range + I(range^2) + I(range^3), ", "
  data = lidar, control.predictor = list(compute = TRUE))" )
```

Time used:

```
Pre = 0.309, Running = 0.193, Post = 0.0701, Total = 0.572
```

Fixed effects:

	mean	sd	0.025quant	0.5quant	0.975quant	mode	kld
(Intercept)	-13.443	1.554	-16.498	-13.443	-10.391	-13.443	0
range	0.074	0.009	0.057	0.074	0.091	0.074	0
I(range^2)	0.000	0.000	0.000	0.000	0.000	0.000	0
I(range^3)	0.000	0.000	0.000	0.000	0.000	0.000	0

Model hyperparameters:

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

Expected number of effective parameters(stdev): 4.14(0.014)

Number of equivalent replicates : 53.35

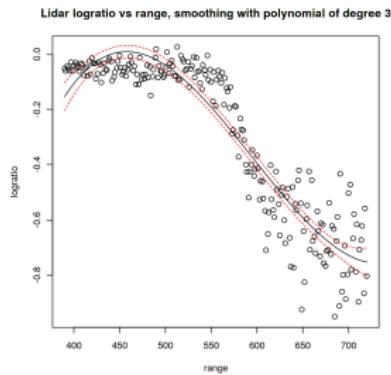
Marginal log-Likelihood: 140.18

Posterior marginals for the linear predictor and
the fitted values are computed

Example: Smoothing (Lidar data)

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

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



Example: Smoothing (Lidar data)

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

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

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

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

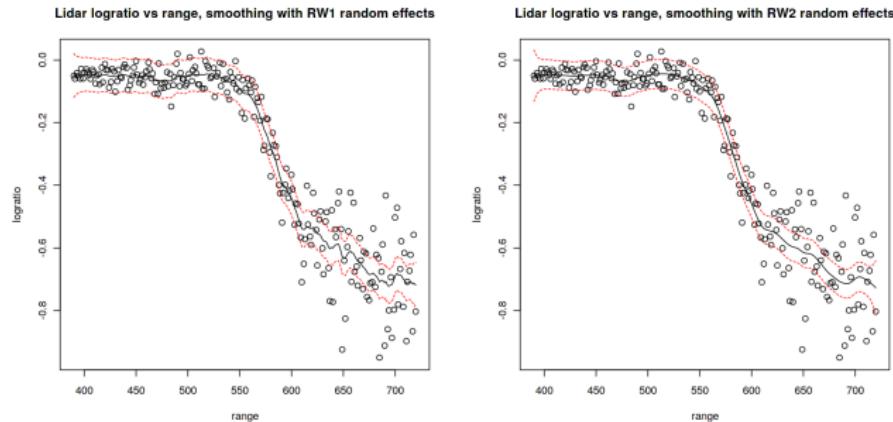
Marginal log-Likelihood: 251.75
```

Example: Smoothing (Lidar data)

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

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

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



Sampling from the posterior

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

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

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

Sampling from the posterior

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

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

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

Sampling from the posterior

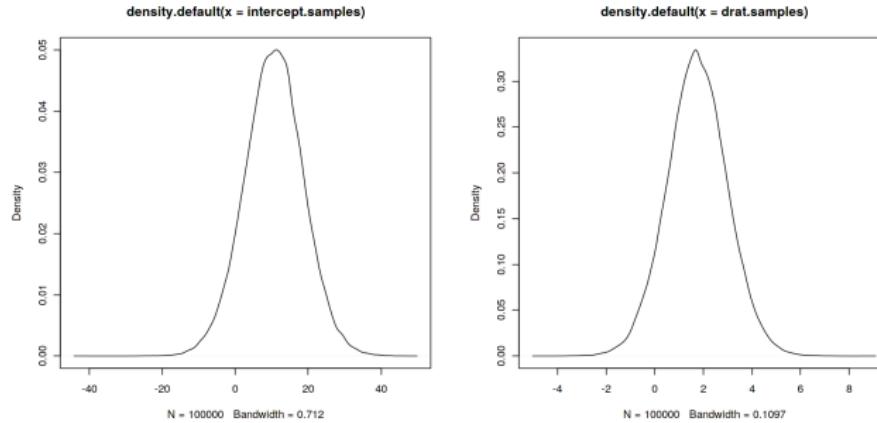
- We can access the value of the hyperparameter by `mtcars.samples[[1]]$hyperpar`, while the value of the latent variables can be accessed by
`mtcars.samples[[1]]$latent[1], ..., mtcars.samples[[1]]$latent[36]`
(there are 36 latent variables in total).
- If we want to collect all of the samples from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

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

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

Sampling from the posterior

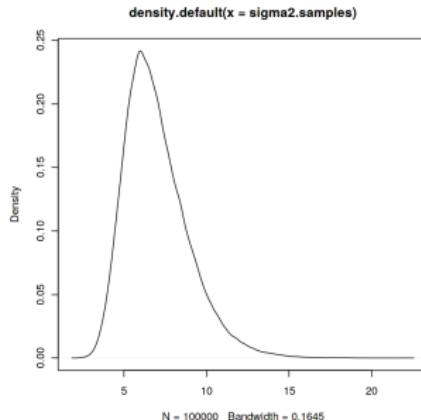
- We can see that the plots are very similar to what we got previously by plotting the marginals computed by INLA directly.



Sampling from the posterior

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

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



Posterior predictive distributions in INLA

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



Posterior predictive distributions in INLA

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

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

- The second step is to fit the model.

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

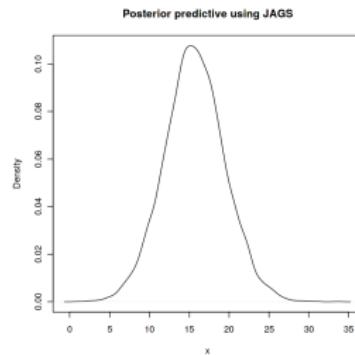
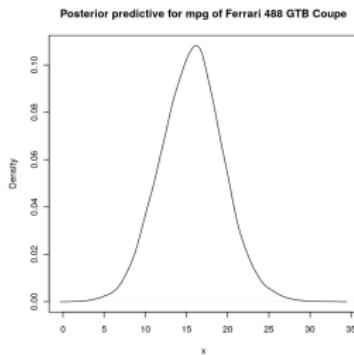
- After this, we obtain posterior samples for the Predictor variable for this new datapoint (located in row 33). The Predictor variables in the sample correspond to the linear predictor η_i in the LGM model. They are not samples from the posterior predictive. The linear predictor is linked to the mean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

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

Posterior predictive distributions in INLA

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

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



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

Model checking in INLA

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

Model checking in INLA

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

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

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

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

Model checking in INLA

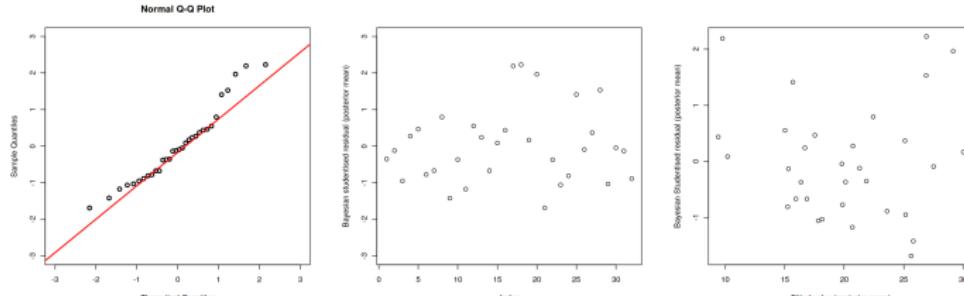
- Using these samples, the fitted values can be computed just as before,

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

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

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

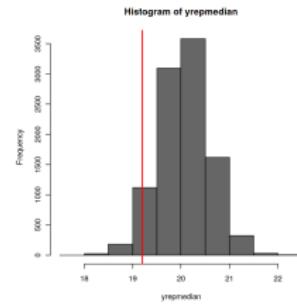
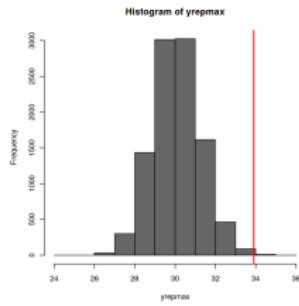
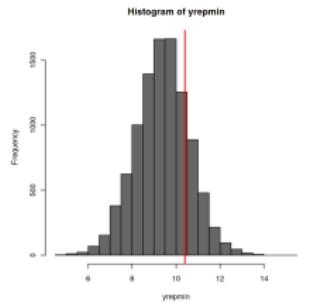
- From these, we are able to compute the studentized residuals, and then do the Q-Q plot, and plot the residuals in terms of their order, and also in terms of the fitted value. The results are similar to what we got from JAGS.



Model checking in INLA

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

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



Model checking in INLA

- ↪ INLA provides a number of Bayesian criteria for model assessment.
- ↪ Marginal likelihood is an useful criteria when comparing different models. It is defined as

$$m(\mathbf{y}) = \int_{\mathbf{x}, \boldsymbol{\theta}} p(\mathbf{y} | \mathbf{x}, \boldsymbol{\theta}) \pi(\mathbf{x}, \boldsymbol{\theta}),$$

and it describes the overall fit of the model, including the prior distribution, on the data. INLA computes $\log(m(\mathbf{y}))$ by default, and displays it in the summary of the model. Larger $\log(m(\mathbf{y}))$ values correspond to better model fit.

- ↪ Conditional predictive ordinate (CPO) is a cross-validation type model assessment criterion, which is defined as

$$CPO_i = p(y_i | y_{-i}),$$

for every observation $1 \leq i \leq n$. This quantifies how likely is the observation i given the rest of the observations given the model. We can summaries these values in a single number by computing

$$NLSCPO = - \sum_{i=1}^n \log(p(y_i | y_{-i})).$$

Smaller values correspond to better model fit.

Model checking in INLA

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

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

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

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

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

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

Model checking in INLA

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

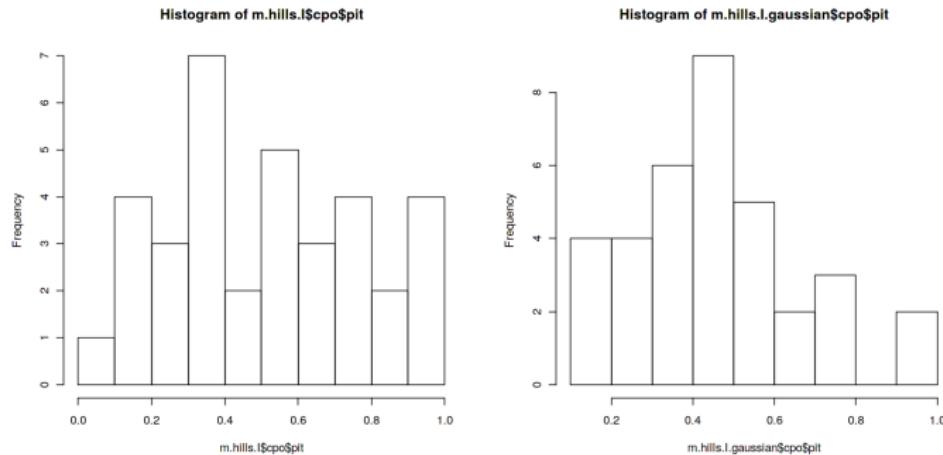
```
m.hills.I <- inla(time ~ 1+climb+dist,family="T",data=hills,
                     control.family=list(hyper=prior.t),
                     control.fixed=prior.fixed,
                     control.compute=list(cpo=TRUE, dic=TRUE))
cat ("DIC:",m.hills.I$dic$dic,"\n")
cat ("NSLCPO:",-sum(log(m.hills.I$cpo$cpo)),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

Model checking in INLA

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

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

Model checking in INLA



↪ The robust model is better according to all 4 criteria.

Summary

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

Bayesian Data Analysis

Daniel Paulin & Nicolò Margaritella

University of Edinburgh



Semester 2, 2020/2021

With thanks to Jonathan Gair, Rubén Amorós-Salvador, Ken Newman, Vanda Inácio and
Natalia Bochkina for much of the material

Outline

- 1 Normal linear regression (recall)
- 2 Departures from Normality: Poisson data
- 3 Departures from Normality: Binary or Bernoulli Data
- 4 Basic structure of GLMs
- 5 Frequentist inference for GLMs
- 6 Bayesian inference for the Poisson GLM
- 7 Bayesian inference for the Binomial GLM
- 8 Categorical Covariates
- 9 Computation of posterior and predictive probabilities
- 10 Bayesian GLM: from JAGS to INLA

Section 1

Normal linear regression (recall)

Generalised Linear Models

Normal linear regression (recall)

- In Linear Regression we assumed that a quantity of primary interest, a random variable Y , has an expected value that is a linear combination of another quantity, x , or a vector of quantities, \mathbf{x} .

We label Y =response variable, x 's=covariates. (Y =random variable, y =observed value.)

$$\rightarrow \mathbb{E}[Y_i] = \mu_i = \beta_0 + \beta_1 x_{i,1} + \dots + \beta_p x_{i,p}, i=1, \dots, n.$$

$$\rightarrow \text{Abbreviated notation for } \mu_i = \mathbf{x}_i^T \boldsymbol{\beta}, \text{ where}$$

$$\mathbf{x}_i = \begin{bmatrix} 1 \\ x_{i,1} \\ \vdots \\ x_{i,p} \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$

- This is a model for the **mean** structure of a random variable Y .

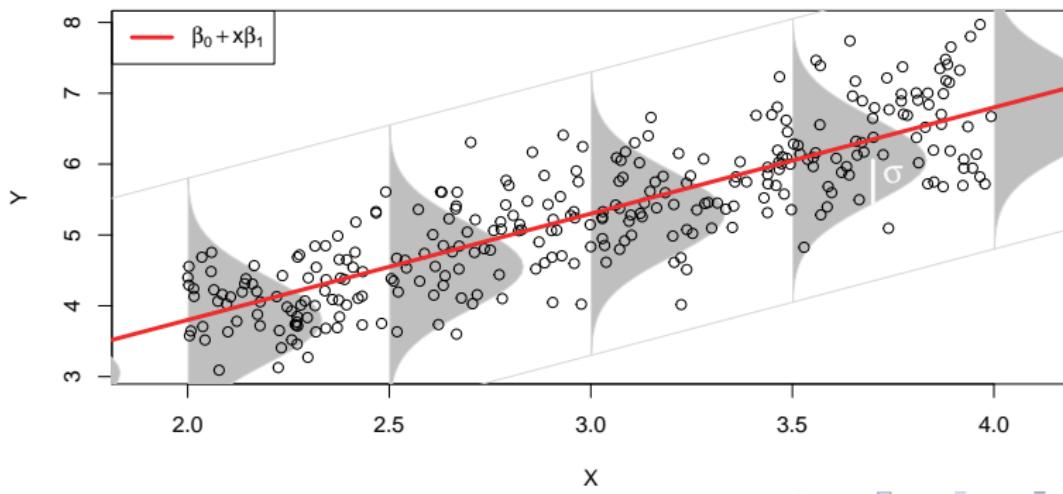
Generalised Linear Models

Normal linear regression

↪ We further assumed a probability distribution of Y_i given \mathbf{x}_i and β :

$$Y_i | \mathbf{x}_i, \beta, \sigma^2 \sim \text{Normal}(\mathbf{x}_i^T \beta, \sigma^2), \quad i = 1, \dots, n$$

With a common variance σ^2 .



Generalised Linear Models

Normal linear regression

→ We can then estimate β and σ^2 by frequentist (classical) methods, namely maximum likelihood:

$$\begin{aligned}\hat{\beta} &= (X^T X)^{-1} X^T \mathbf{y} \\ \hat{\sigma}^2 &= \frac{\sum_{i=1}^n (y_i - \mathbf{x}_i^T \hat{\beta})^2}{n}\end{aligned}$$

where

$$X = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{bmatrix}$$

Generalised Linear Models

Normal linear regression

↪ Or by Bayesian estimation of β and σ^2 , with the posterior distribution:

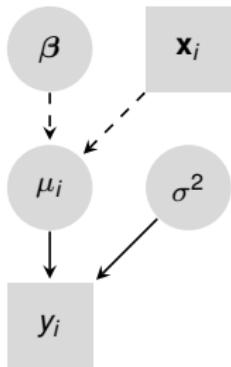
$$\begin{aligned} p(\beta, \sigma^2 | \mathbf{y}) &= \frac{p(\beta, \sigma^2, \mathbf{y})}{m(\mathbf{y})} = \frac{f(\mathbf{y} | \beta, \sigma^2)\pi(\beta, \sigma^2)}{\int \dots \int f(\mathbf{y} | \beta, \sigma^2)\pi(\beta, \sigma^2) d\beta_0 \dots d\sigma^2} \\ &\propto f(\mathbf{y} | \beta, \sigma^2)\pi(\beta, \sigma^2) \end{aligned}$$

Generalised Linear Models

Normal linear regression

→ Example of a full model and directed acyclic graph (DAG) representation:

$$\begin{aligned}
 y_i | \mu_i, \sigma^2, \mathbf{x}_i &\sim N(\mu_i, \sigma^2), \quad i = 1, \dots, n \\
 \mu_i &= \mathbf{x}_i^T \boldsymbol{\beta} \\
 \beta_j &\sim N(0, \sigma_{\beta_j}^2), \quad j = 0, \dots, p \\
 \sigma &\sim Unif(0, u_\sigma)
 \end{aligned}$$



Section 2

Departures from Normality: Poisson data

Generalised Linear Models

Departures from Normality: Poisson data

- An example is the number of “satellite” male crabs that a female Horseshoe crab attracts in addition to a “primary” male crab, y , compared to the width of the carapace (the larger outer shell), x .

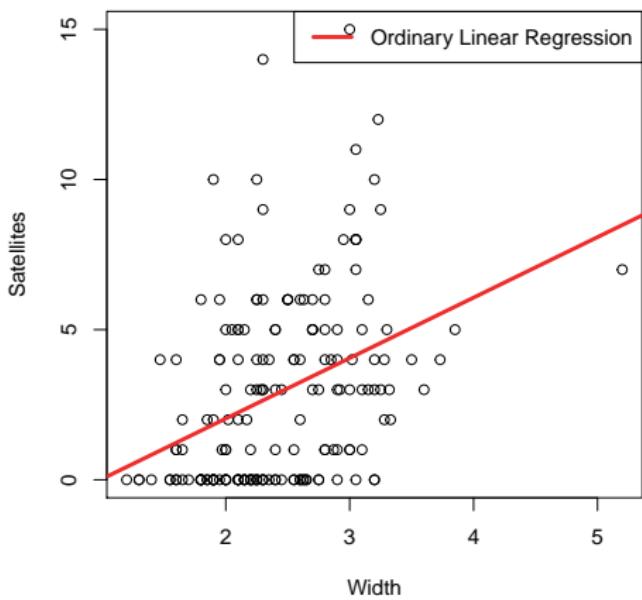


¹pictures NWF; Jane Brockwell

Generalised Linear Models

Departures from Normality: Poisson data

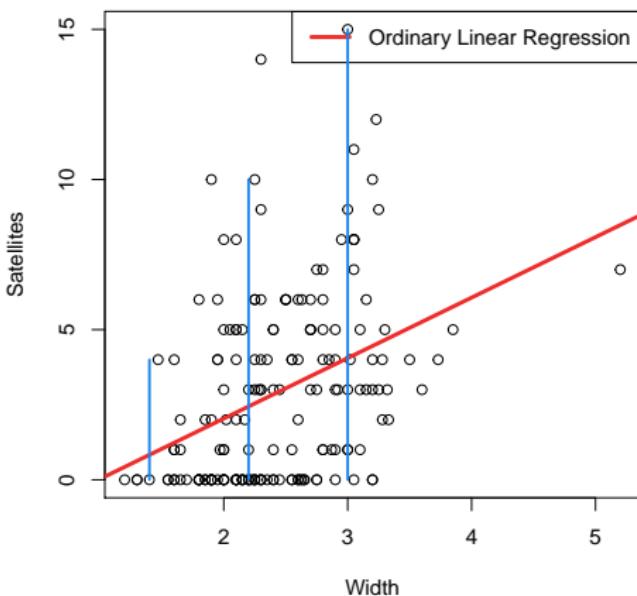
Female Horseshoe Crab's Satellites vs Carapace Width



Generalised Linear Models

Departures from Normality: Poisson data

Female Horseshoe Crab's Satellites vs Carapace Width



- Discrete response
- Lower boundary
- Heteroskedasticity
(non-constant variance)

Generalised Linear Models

Departures from Normality: Poisson data

- ↪ We may consider the data to come from a Poisson distribution.
- ↪ Frequentist usual Normal linear regression assumes that the variance of y_i is a constant, i.e., $\text{Var}(y_i) = \sigma^2$, but in a Poisson distribution $\text{Var}(y_i) = E(y_i) = \mu_i$.
- ↪ But modeling μ_i as in the Normal model does not ensure positive mean, i.e.

$$\mu_i = \beta_0 + \beta_1 x_i > 0$$

is not guaranteed.

- ↪ One solution to this is to model the logarithm of μ_i :

$$\ln(\mu_i) = \beta_0 + \beta_1 x_i$$

Then

$$\mu_i = \exp(\beta_0 + \beta_1 x_i) > 0$$

Generalised Linear Models

Departures from Normality: Poisson data

- ↪ The likelihood of the Poisson model with mean $\mu_i = \exp(\beta_0 + \beta_1 x_i)$ is:

$$f(\mathbf{y} | \beta_0, \beta_1, \mathbf{x}) \propto \prod_i e^{-\mu_i} \mu_i^{y_i} = \prod_i e^{-\exp(\beta_0 + \beta_1 x_i)} (\exp(\beta_0 + \beta_1 x_i))^{y_i}$$

- ↪ Under the frequentist paradigm, maximum (log)likelihood with respect to β_0 and β_1 :

$$\ln(f(\mathbf{y} | \beta_0, \beta_1, \mathbf{x})) \propto \sum_i (-\exp(\beta_0 + \beta_1 x_i) + y_i(\beta_0 + \beta_1 x_i))$$

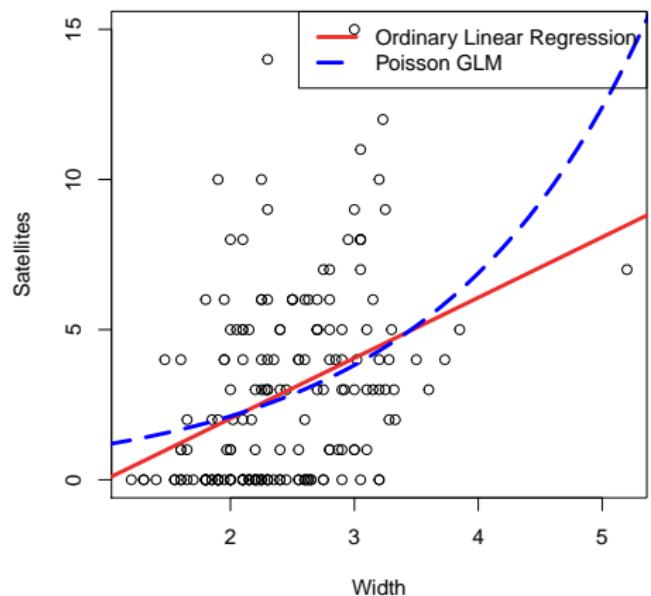
- ↪ Under the Bayesian paradigm, a prior distribution for β_0 and β_1 must be defined and the posterior of the parameters is found in the usual way:

$$p(\beta_0, \beta_1 | \mathbf{y}, \mathbf{x}) \propto f(\mathbf{y} | \beta_0, \beta_1, \mathbf{x}) \pi(\beta_0, \beta_1)$$

Generalised Linear Models

Departures from Normality: Poisson data

Female Horseshoe Crab's Satellites vs Carapace Width



Section 3

Departures from Normality: Binary or Bernoulli Data

Generalised Linear Models

Departures from Normality: Binary or Bernoulli Data

- ↪ Binary Data: Success or Failure, 1 or 0: $Y \sim \text{Bernoulli}(\theta)$. Thus $\mathbb{E}[Y] = \mu = \theta$.
- ↪ The probability of success, θ , may be a function of covariates.
- ↪ Can you think of some examples of Binary Data and corresponding covariates that could affect θ ?
- ↪ For example, is a randomly chosen Scot in favour of Scottish independence? Let $\theta = \Pr(\text{in favour}) = \Pr(Y = 1)$. What factors might affect θ ?
- ↪ Suppose $x = \text{Age}$ is a factor. What might be wrong with the following model?

$$\mu_i \equiv \theta_i = \beta_0 + \beta_1 \text{Age}_i$$

Generalised Linear Models

Departures from Normality: Binary or Bernoulli Data

- Again a solution for this is to construct a 1:1 function of μ , $g(\mu)$, as a linear combination of covariates,

$$g(\mu_i) \equiv g(\theta_i) = \eta_i = \beta_0 + \beta_1 \text{Age}_i$$

where the inverse of g maps into the allowable range of θ :

$$0 < [g^{-1}(\beta_0 + \beta_1 \text{Age}_i) = \theta_i] < 1$$

- A particular function that can be used for this case is the *logit* function, $\ln(x/(1-x))$:

$$g(\theta_i) = \ln(\theta_i/(1 - \theta_i)) = \beta_0 + \beta_1 \text{Age}_i$$

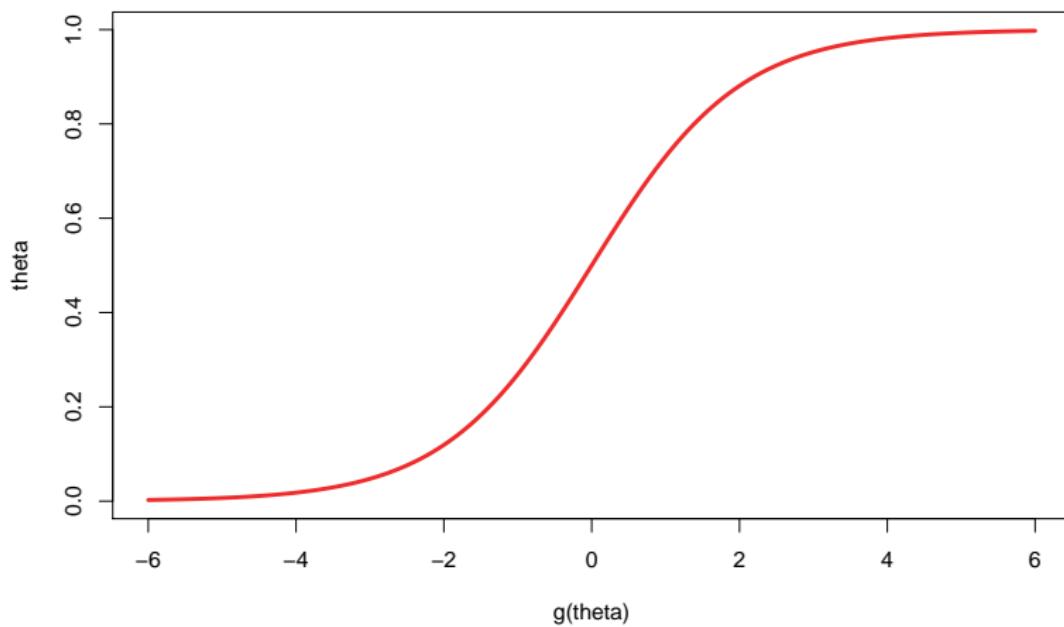
Then

$$0 < [g^{-1}(\beta_0 + \beta_1 \text{Age}_i) \equiv \theta_i = \frac{1}{1 + \exp(-\beta_0 - \beta_1 \text{Age}_i)}] < 1$$

Generalised Linear Models

Departures from Normality: Binary or Bernoulli Data

Inverse-Logit function



Generalised Linear Models

Departures from Normality: Binary or Bernoulli Data

- ↪ The likelihood of the Bernoulli model with probability of success $\theta_i = \frac{1}{1+\exp(-\beta_0-\beta_1x_i)}$ is:

$$f(\mathbf{y} | \beta_0, \beta_1, \mathbf{x}) = \prod_i \theta_i^{y_i} (1 - \theta_i)^{(1-y_i)} = \prod_i \left(1 + e^{-\beta_0 - \beta_1 x_i}\right)^{-y_i} \left(1 + e^{\beta_0 + \beta_1 x_i}\right)^{-(1-y_i)}$$

- ↪ Under the frequentist paradigm, maximum (log)likelihood with respect to β_0 and β_1 :

$$\ln(f(\mathbf{y} | \beta_0, \beta_1, \mathbf{x})) = \sum_i (y_i(\beta_0 + \beta_1 x_i) - \ln(1 + \exp(\beta_0 + \beta_1 x_i)))$$

- ↪ Under the Bayesian paradigm, a prior distribution for β_0 and β_1 must be defined and the posterior of the parameters is found in the usual way:

$$p(\beta_0, \beta_1, | \mathbf{y}, \mathbf{x}) \propto f(\mathbf{y} | \beta_0, \beta_1, \mathbf{x}) \pi(\beta_0, \beta_1)$$

Section 4

Basic structure of GLMs

Generalised Linear Models

Basic Structure

Generalised Linear Models extend the Normal Linear Model in two ways:

- ➊ The distribution of Y is a member of the *Exponential Family Distributions*.
- ➋ A 1:1 'link' function of $\mathbb{E}[Y] = \mu$, $g(\mu)$, is a linear combination of covariates, $g(\mu) = \mathbf{x}^T \boldsymbol{\beta}$

$$\begin{aligned} Y &\sim f(y | \mu, \dots) \\ g(\mu) &= \eta \\ \eta &= \mathbf{x}^T \boldsymbol{\beta} \end{aligned}$$

Generalised Linear Models

Exponential Family of Distributions

- ↪ Exponential Family of Distributions include Normal, Poisson, Bernoulli, Binomial, Multinomial, Exponential, Gamma, Beta, Lognormal, Dirichlet.
- ↪ Precise Definition for Single Parameter Case

$$f_Y(y \mid \theta) = h(y) \exp(\eta(\theta) \cdot T(y) - A(\theta))$$

- ↪ Precise Definition for Multiple Parameter Case

$$f_Y(y \mid \theta) = h(y) \exp\left(\sum_{i=1}^s \eta_i(\theta) T_i(y) - A(\theta)\right)$$

- ↪ $h(y)$ and $T(y)$ are functions of y
- ↪ $\eta(\theta)$ and $A(\theta)$ are functions of θ
- ↪ More on this topic can be found in Generalised Regression Models course.
- ↪ NOTE: Exponential family distributions have conjugate priors!

Generalised Linear Models

Two Examples: Exponential Family of Distributions

$$f_Y(y | \theta) = h(y) \exp(\eta(\theta) \cdot T(y) - A(\theta))$$

↪ Poisson(θ)²

$$f_Y(y | \theta) = \frac{\exp(-\theta)\theta^y}{y!} = \frac{1}{y!} \exp(\ln(\theta) \cdot y - \theta)$$

↪ Exponential(θ)³

$$f_Y(y | \theta) = \theta \exp(-\theta y) = \exp(-\theta \cdot y - (-\ln(\theta)))$$

²where $h(y)=1/y!$; $\eta(\theta)=\ln(\theta)$; $T(y)=y$; and $A(\theta)=\theta$.

³where $h(y)=1$; $\eta(\theta)=-\theta$; $T(y)=y$; and $A(\theta)=-\ln(\theta)$.

Generalised Linear Models

Link Function

$$g(\mu) = \eta = \mathbf{x}_i^T \boldsymbol{\beta}$$

- ↪ The Link Function g is not a unique function.
- ↪ g is usually (but not always) defined so that $g^{-1}(\mathbf{x}_i^T \boldsymbol{\beta})$ yields only allowable values for μ .
 - For example, the “identity” link function $g(\mu) = \mu$ can be specified for a Poisson but that does not ensure that $g^{-1}(\mathbf{x}_i^T \boldsymbol{\beta}) = \mathbf{x}_i^T \boldsymbol{\beta} > 0$. In practice, some constraints must be added to the estimates $\mathbf{x}_i^T \boldsymbol{\beta}$.

Generalised Linear Models

Exponential Family of Distributions and Link Functions

Some examples of link functions for certain distributions:

$$\text{Distribution} \quad g(\mu) = \mathbf{x}^t \boldsymbol{\beta}$$

Normal(μ, σ^2) Identity: $g(\mu) = \mu$

Poisson(μ) Log: $g(\mu) = \ln(\mu)$

Bernoulli(μ) Logit: $g(\mu) = \ln(\mu/(1 - \mu))$

C-log-log: $g(\mu) = \ln(-\ln(1 - \mu))$

Probit: $g(\mu) = \Phi^{-1}(\mu)$

Probit link: $\Phi(x)$ = CDF for a Standard Normal random variable:

$$\Phi(x) = \int_{-\infty}^x \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right)$$

$\Phi : \mathcal{R}^1 \rightarrow [0, 1]$ and is 1:1, thus $\Phi^{-1} : [0, 1] \rightarrow \mathcal{R}^1$.

Section 5

Frequentist inference for GLMs

Generalised Linear Models

Frequentist Inference with R

- We will first discuss how to do frequentist inference for GLMs using the R function `glm`, and then show how to carry out a parallel Bayesian analysis using JAGS.
- We will look at two examples: a Poisson model (using the Horseshoe crabs data) and a Binomial model
- The `glm` function has multiple arguments:

```
glm(formula, family = gaussian, data, weights, subset,  
na.action, start = NULL, etastart, mustart, offset, ...)
```

Generalised Linear Models

Frequentist Inference with R

- formula is an expression for the mean structure that is similar to that for normal linear regression

```
formula = y ~ x           # a single covariate,  $b_0 + b_1 \cdot x$ 
formula = y ~ x1 + x2    # two covariates,  $b_0 + b_1 \cdot x_1 + b_2 \cdot x_2$ 
formula = y ~ x1*x2      # with interac.:  $b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + b_3 \cdot (x_1 \cdot x_2)$ 
```

- While interpretation of the right-hand side is the same as for lm: $\mathbf{x}^T \boldsymbol{\beta}$, the left-hand side is $g(\mu)$ (i.e. the function of $\mathbb{E}[Y | \mathbf{x}]$).

```
formula = ``g(\mu)'' ~ x1 + x2
```

Generalised Linear Models

Frequentist Inference with R

- family specifies what exponential family distribution is used:

```
family=gaussian # the default value
family=poisson
family=binomial
```

Each family has an implicit link function, $g(\mu)$ used by default. For example, here are the default link functions made explicit for some distributions:

```
gaussian(link = "identity")
poisson(link = "log")
binomial(link = "logit")
Gamma(link = "inverse")
inverse.gaussian(link = "1/mu^2")
```

- One can override these defaults by typing an alternative link function between brackets.
For example:

```
glm(formula=y ~ x, family=poisson(link=sqrt))
```

Generalised Linear Models

Frequentist Inference: Poisson and Horseshoe crabs

↪ The `crabs` data set has 6 attributes:

```
head(crabs) [,-1]
```

	Colour	Spine	Weight	Width	Satellites
## 1	med	both.bad	28.3	3.05	8
## 2	dk.med	both.bad	26.0	2.60	4
## 3	dk.med	both.bad	25.6	2.15	0
## 4	dk	one.bad	21.0	1.85	0
## 5	med	both.bad	29.0	3.00	1
## 6	lt.med	one.bad	25.0	2.30	3

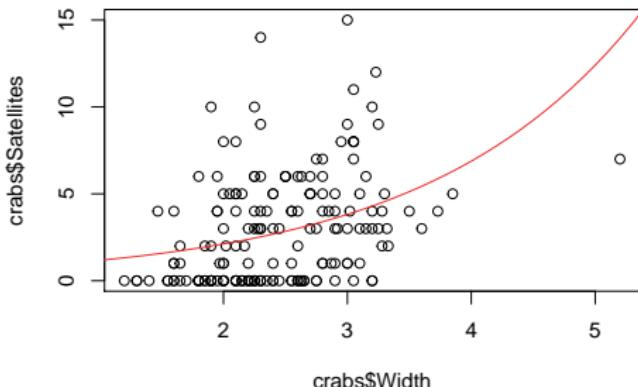
To begin we just focus on modelling the number of satellites as a function of carapace width.

Generalised Linear Models

Frequentist Inference: Poisson and Horseshoe crabs

R code for Poisson model

```
m1 <- glm(formula = Satellites ~ Width, family=poisson(link=log),  
          data=crabs )  
newdata <- data.frame(Width=seq(1,6,0.1))  
pred1 <- predict(m1, newdata=newdata, type="response")  
plot(crabs$Width, crabs$Satellites)  
lines(newdata$Width, pred1, col="firebrick2")
```



Generalised Linear Models

Frequentist Inference: Poisson and Horseshoe crabs

Different link functions:

```

m1.identity <- glm(formula = Satellites ~ Width,
                     family=poisson(link=identity), data=crabs, start=c(0,1))
m1.sqrt <- glm(formula = Satellites ~ Width,
                 family=poisson(link=sqrt), data=crabs )
tabb<-round(rbind(coef(m1),coef(m1.identity),coef(m1.sqrt)),2)
row.names(tabbb)<-c("log","identity","sqrt")
tabbb

##          (Intercept) Width
## log           -0.43   0.59
## identity      -2.59   2.26
## sqrt           0.16   0.62

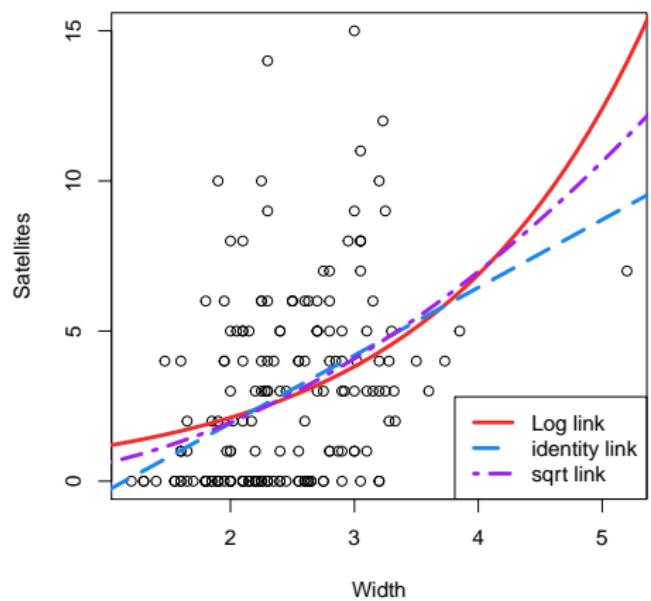
```

- Different link functions lead to different estimated parameters with very different interpretations.

Generalised Linear Models

Frequentist Inference: Poisson and Horseshoe crabs

Prediction for different link functions:



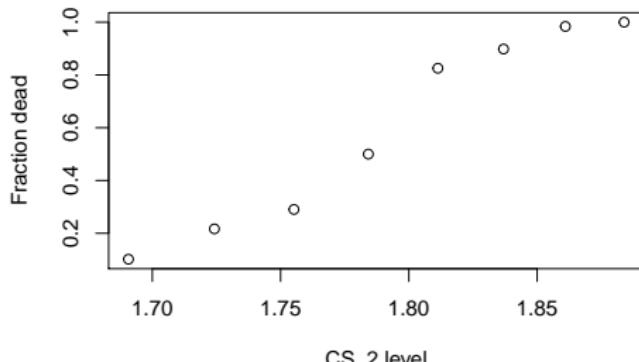
Generalised Linear Models

Frequentist Inference: Binomial and Dead Beetles

- Bliss (1935) presented results of an experiment on the fraction of confused flour beetles (*Tribolium confusum*) that were dead 5 hours after being exposed to gaseous Carbon Disulphide (CS_2). Eight different concentrations of CS_2 were used and about 60 beetles were exposed to each concentration.



- Fraction dead vs exposure:



Generalised Linear Models

Frequentist Inference: Binomial and Dead Beetles

```

n.exposed <- c(59, 60, 62, 56, 63, 59, 62, 60)
CS2.level <- c(1.691, 1.724, 1.755, 1.784, 1.811, 1.837, 1.861, 1.884)
y.dead <- c(6, 13, 18, 28, 52, 53, 61, 60)
beetles <- data.frame(n.exposed, CS2.level, y.dead)
m.logit<- glm(cbind(y.dead, n.exposed-y.dead) ~ CS2.level,
                family=binomial(link=logit), data = beetles)
m.cloglog<- glm(cbind(y.dead, n.exposed-y.dead) ~ CS2.level,
                  family=binomial(link=cloglog), data = beetles)
m.probit<- glm(cbind(y.dead, n.exposed-y.dead) ~ CS2.level,
                 family=binomial(link=probit), data = beetles)
tabb2<- round(rbind(coef(m.logit),coef(m.cloglog),coef(m.probit)),3))
row.names(tabb2)<-c("logit","cloglog","probit")
colnames(tabb2)<-c("beta_0", "beta_1 (CS2.level)")
tabb2

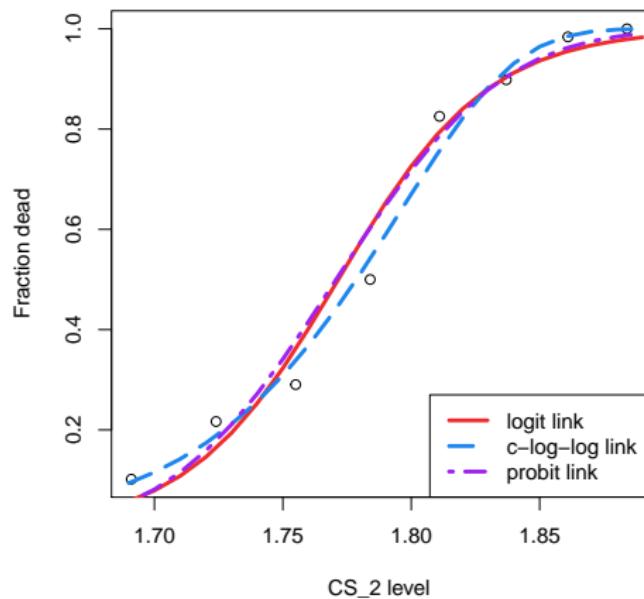
##          beta_0 beta_1 (CS2.level)
## logit     -60.740      34.286
## cloglog   -39.522      22.015
## probit    -34.956      19.741

```

Generalised Linear Models

Frequentist Inference: Poisson and Horseshoe crabs

Prediction for different link functions:



Section 6

Bayesian inference for the Poisson GLM

Generalised Linear Models

Bayesian Inference

- As usual, for the Bayesian approach we need to specify priors for the unknown parameters. In the case of GLMs, that is primarily priors for the β s in the model for the expected response via the link function formulation: $g(\mu) = \mathbf{x}^T \boldsymbol{\beta}$.
- The posterior distribution for $\boldsymbol{\beta}$:

$$p(\boldsymbol{\beta} | \mathbf{y}) \propto f(\mathbf{y} | \boldsymbol{\beta})\pi(\boldsymbol{\beta})$$

But now the β will appear in the likelihood via the link function. For example, with a Poisson and a single covariate:

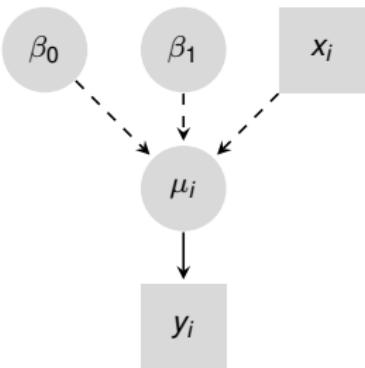
$$\begin{aligned} p(\mathbf{y} | \beta_0, \beta_1, \mathbf{x}) &\propto \prod_i e^{-\mu_i} \mu_i^{y_i} = \exp\left(-\sum_{i=1}^n \mu_i\right) \prod_i \mu_i^{y_i} \\ &= \exp\left(-\sum_i g^{-1}(\beta_0 + \beta_1 x_i)\right) \prod_i \left(g^{-1}(\beta_0 + \beta_1 x_i)\right)^{y_i} \end{aligned}$$

Generalised Linear Models

Bayesian Inference: Poisson and Horseshoe Crabs

→ The Poisson full model for the Horseshoe Crabs and its DAG representation:

$$\begin{aligned} y_i | \mu_i, \mathbf{x}_i &\sim \text{Poisson}(\mu_i), \quad i = 1, \dots, n \\ \log(\mu_i) &= \beta_0 + \beta_1 \text{Width}_i \\ \beta_j &\sim N(0, \sigma_{\beta_j}^2), \quad j = 0, 1 \end{aligned}$$



Generalised Linear Models

Bayesian Inference: Parameters and Priors

Data Centering

- High correlation between parameters can slow the MCMC. With many correlated parameters, the time necessary to generate enough effective samples for each parameter might be too long.
- Centering the covariates usually helps $(x - \bar{x})$, especially by reducing the dependence of β_0 with the rest of the β s.

Prior Selection

- The selection of “non-informative” priors is highly dependent on the possible values of the response variable, the covariates and the link function.
- Numerical issues might arise with too sparse priors for the β s and extreme initial values (see, for example, the problem of separation in logistic regression⁴)
- On the other hand, too concentrated priors can be too informative or even restrictive to the support of μ .
- A useful diagnostic for priors for GLMs is to examine the induced priors for the mean parameter μ for typical and extreme individuals.
- Always check sensitivity to priors!

⁴Gelman A, Carlin JB, Stern HS, et al. Bayesian Data Analysis, Third Edition, CRC Press. Chapter 16, section 16.3

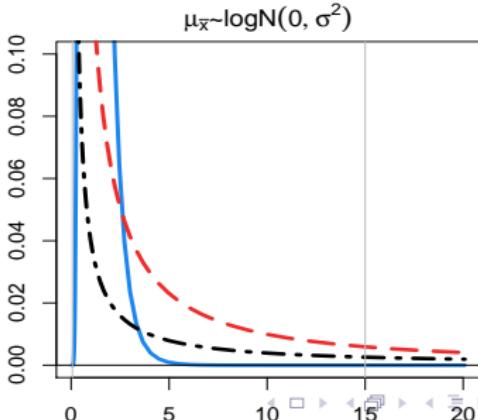
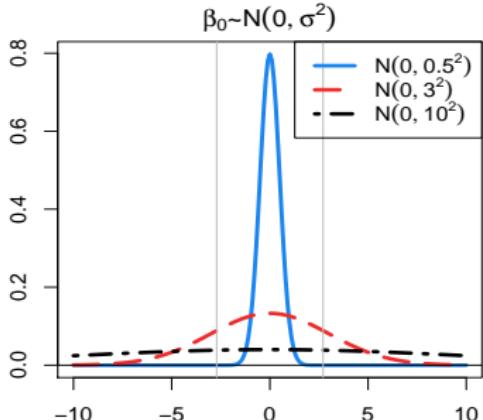


Generalised Linear Models

Bayesian Inference: Parameters and Priors

- ↪ $\log(\mu_i) = \beta_0 + \beta_1(x_i - \bar{x})$ or equivalently $\mu_i = e^{\beta_0 + (x_i - \bar{x})\beta_1}$
- ↪ If the covariates are centered, β_0 determines the expected value of y_i for a typical individual ($x_i = \bar{x}$).

$$\mu_{\bar{x}} = g^{-1}(\beta_0 + \beta_1(\bar{x} - \bar{x})) = g^{-1}(\beta_0) = e^{\beta_0}$$
- ↪ For example, usual values for the number of Horseshoe crab satellites (y_i) are between 0 and 15. With a log-link function, we could try to be uninformative by giving high probability density for the induced variable $\mu_{\bar{x}} = e^{\beta_0}$ on the interval $[1/15, 15]$ and surroundings; or equivalently for β_0 on the interval $[-\ln(15), \ln(15)]$.



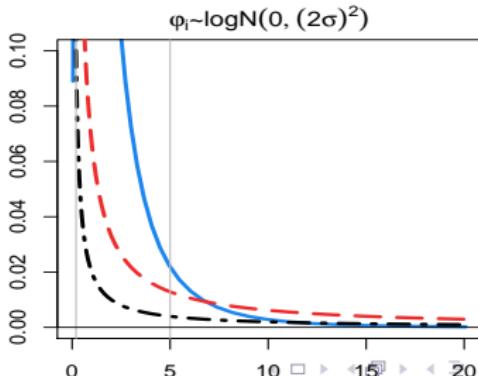
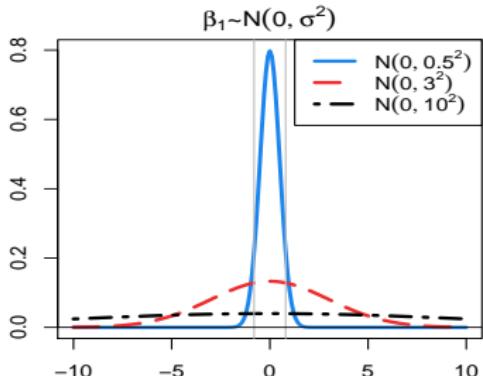
Generalised Linear Models

Bayesian Inference: Parameters and Priors

↪ And for the prior of β_1 , with a log-link function, we have that

$$\mu_i = e^{\beta_0 + (x_i - \bar{x})\beta_1} = \mu_{\bar{x}} e^{(x_i - \bar{x})\beta_1} \equiv \mu_{\bar{x}} \varphi_i$$

- ↪ Therefore β_1 and x_i determine a factor φ_i that indicates, for any individual i , how many times larger μ_i is with respects of $\mu_{\bar{x}}$.
- ↪ Usual values of the width of Horseshoe crabs are between 1 and 5, so $\max |x_i - \bar{x}| \approx 2$. For extreme values of x_i , $\mu_{\bar{x}}$ will be multiplied or divided by a factor $\varphi_i = e^{2\beta_1}$.
- ↪ If we consider that an extreme individual can have a mean at most $\varphi_i = 5$ times the mean of a typical individual, then $\beta_1 = \ln(\varphi_i) / |x_{extreme} - \bar{x}|$ would be in $[-\ln(5)/2, \ln(5)/2]$.



Generalised Linear Models

Bayesian Inference: Poisson and Horseshoe Crabs

- ↪ Using JAGS, we repeat the analysis of the number of Horseshoe Crab Satellites as a function of carapace width, again assuming a Poisson distribution, in a Bayesian framework.
- ↪ The log link function is used and the covariate will be centred:
 $\log(\mu) = \beta_0 + \beta_1(Width - \bar{Width})$.
- ↪ The priors for the β s will be $\text{Normal}(0, \tau)$, where $\tau=0.01$.
- ↪ For JAGS we need three components: a data block, initial values for the parameters, and a model statement.

Generalised Linear Models

Bayesian Inference: Poisson and Horseshoe Crabs

↪ Data block and Initial Values

```
n <- length(crabs$Obs)
crabs.data <- list(n=n, Satellites=crabs$Satellites, Width=crabs$Width)

crabs.inits <- list(list(beta0=-1, beta1=-1),
                      list(beta0=0, beta1=1),
                      list(beta0=3, beta1=2))
```

Generalised Linear Models

Bayesian Inference: Poisson and Horseshoe Crabs

↪ Model statement

```
crabs.model <- "model {  
  # Hyperparameters  
  beta.mu.0    <- 0  
  beta.tau.0   <- 0.01  
  
  # prior  
  beta0 ~ dnorm(beta.mu.0,beta.tau.0)  
  betal ~ dnorm(beta.mu.0,beta.tau.0)  
  
  #Likelihood  
  for(i in 1:n) {  
    # Note: link function on LHS of fn assignment  
    log(mu[i])  <- beta0+betal*(Width[i]-mean(Width[]))  
    Satellites[i] ~ dpois(mu[i])  
  }  
}"
```

Generalised Linear Models

Bayesian Inference: Poisson and Horseshoe Crabs

↪ Call from R to JAGS:

```
# Run JAGS to the completion of the "adaption" stage
results.crabs.A <- jags.model(file=textConnection(crabs.model),
                                data=crabs.data, inits=crabs.inits,
                                n.chains=3)

# Burn-in of 5000 iterations
update(results.crabs.A, n.iter=5000)

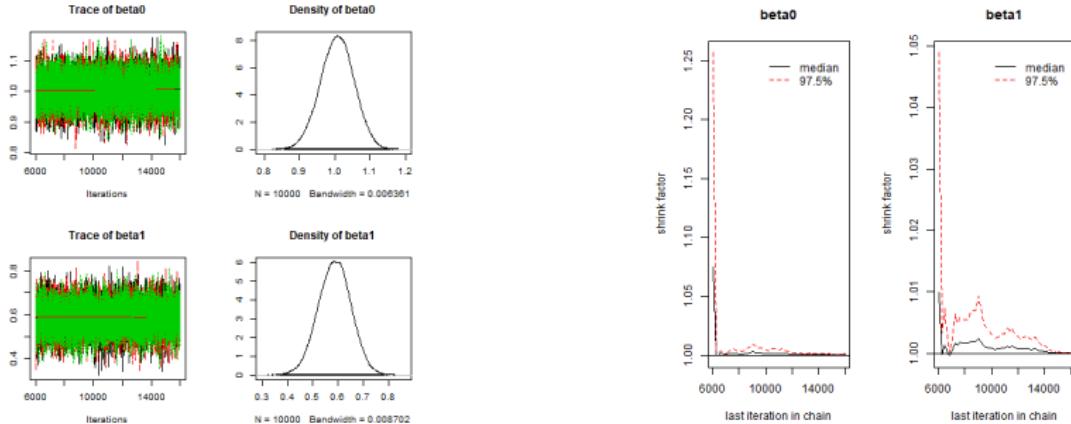
# Longer run for making inferences, assuming chains have converged
results.crabs.B <- coda.samples(results.crabs.A,
                                   variable.names=c("beta0", "beta1"), n.iter=10000)
```

Generalised Linear Models

Bayesian Inference: Poisson and Horseshoe Crabs

→ Diagnostics before looking at the summary statistics

```
# Trace plots and density
plot(results.crabs.B)
# Brooks-Gelman-Rubin statistic (want a value near 1)
gelman.plot(results.crabs.B)
```



Generalised Linear Models

Bayesian Inference: Poisson and Horseshoe Crabs

- ↪ Other measures of "mixing"- effective sample size given 10,000

```
effectiveSize(results.crabs.B[[1]][, "beta0"])
```

```
##      var1  
## 5254.064
```

```
effectiveSize(results.crabs.B[[1]][, "beta1"])
```

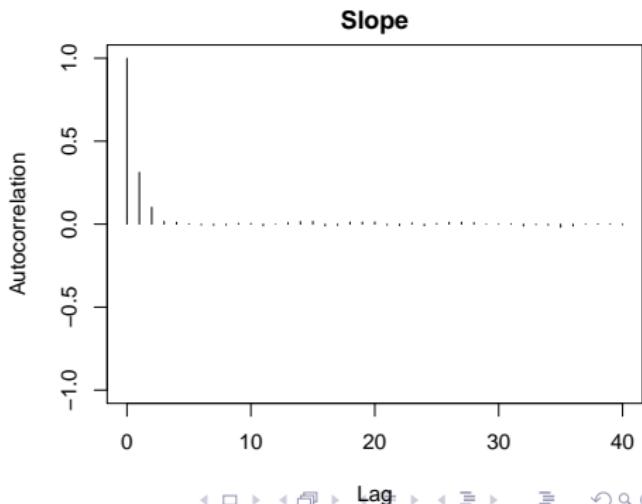
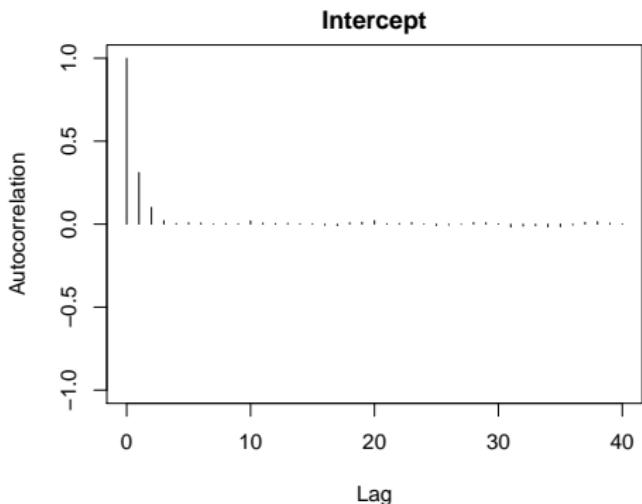
```
##      var1  
## 5236.409
```

Generalised Linear Models

Bayesian Inference: Poisson and Horseshoe Crabs

↪ And autocorrelation within each chain.

```
autocorr.plot(results.crabs.B[[1]][, "beta0"], main="Intercept")
autocorr.plot(results.crabs.B[[1]][, "beta1"], main="Slope")
```



Generalised Linear Models

Bayesian Inference: Poisson and Horseshoe Crabs

- Now look at the posteriors and compare to MLEs:

```
summary(results.crabs.B)
##           Mean      SD  Naive SE Time-series SE
## beta0 1.0067 0.04717 0.0002723      0.0003780
## beta1 0.5864 0.06452 0.0003725      0.0005275

## vs MLEs #####
##             Estimate Std. Error
## (Intercept) 1.00782   0.04691
## ctr.Width   0.58919   0.06500
```

- Monte Carlo error: error in the calculated mean due to random sampling. Can be reduced by increasing the chain length
- A “rule of thumb” (in the OpenBUGS manual) is to have a chain length such that the MC error is no more than 1/20th the size of the standard deviation of the parameter.⁵

⁵The reduction is not proportional; in this case, double chain length reduces MC error by 28% ↗

Generalised Linear Models

Bayesian Inference: Poisson and Horseshoe Crabs

↪ Interpretation

```
# Joinning all the chains in one data.frame
results.crabs.output <- do.call(rbind.data.frame, results.crabs.B)
#interpretation
cat("E[exp(beta0)] is", round(mean(exp(results.crabs.output$beta0)), 3))

## E[exp(beta0)] is 2.739

cat("E[exp(beta1)] is", round(mean(exp(results.crabs.output$beta1)), 3))

## E[exp(beta1)] is 1.801
```

- ↪ 2.739 is the posterior expected number of satellite male crabs for a female with an average carapace width
- ↪ For a unit increase in the carapace width of a female crab, the expected number of her satellite male crabs increases of 1.801 times, i.e. a 80% increase.

Section 7

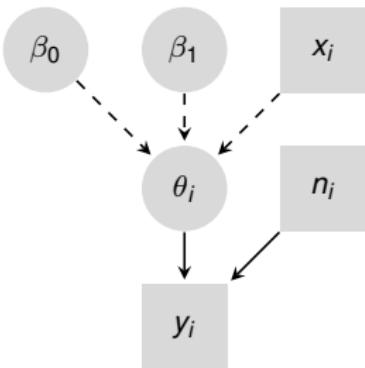
Bayesian inference for the Binomial GLM

Generalised Linear Models

Bayesian Inference: Binomial and Dead Beetles

→ The Binomial full model for the beetles and its DAG representation:

$$\begin{aligned} y_i | \mu_i, \mathbf{x}_i &\sim \text{Bin}(\theta_i, n_i), \quad i = 1, \dots, l \\ g(\theta_i) &= \beta_0 + \beta_1 Dose_i \\ \beta_j &\sim N(0, \sigma_{\beta_j}^2), \quad j = 0, 1 \end{aligned}$$



Generalised Linear Models

Bayesian Inference: Parameters and Priors

- ↪ $g(\theta_i) = \ln(\theta_i / (1 - \theta_i)) = \beta_0 + \beta_1(x_i - \bar{x})$
- ↪ β_0 is the log odds of beetle death for the mean concentration \bar{x} of CS₂
- ↪ β_1 is the log odds-ratio (i.e. the expected change in the log odds) of beetle death for a one-unit increase in the concentration x of CS₂
- ↪ When setting priors we have to consider the minimum distance in x , $x_u - x_l$, for which we think the change of probability can happen. We then set a prior that ensures that $|\beta_1|$ can be larger than 10 over that minimum change distance.

$$-5 \approx \ln(0.005 / (1 - 0.005)) = \beta_0 + \beta_1(x_l - \bar{x})$$

$$5 \approx \ln(0.995 / (1 - 0.995)) = \beta_0 + \beta_1(x_u - \bar{x})$$

- ↪ For the prior of β_0 , we have to ensure that $|\beta_0|$ can be larger than $(\bar{x} - x_m)|\beta_1|^6$ where $x_m = (x_u + x_l)/2$.
- ↪ When in doubt, try vaguer priors and check sensitivity. If results do not change, you are probably being uninformative enough.

⁶To work out this relationship, start by adding the two equations in this slide



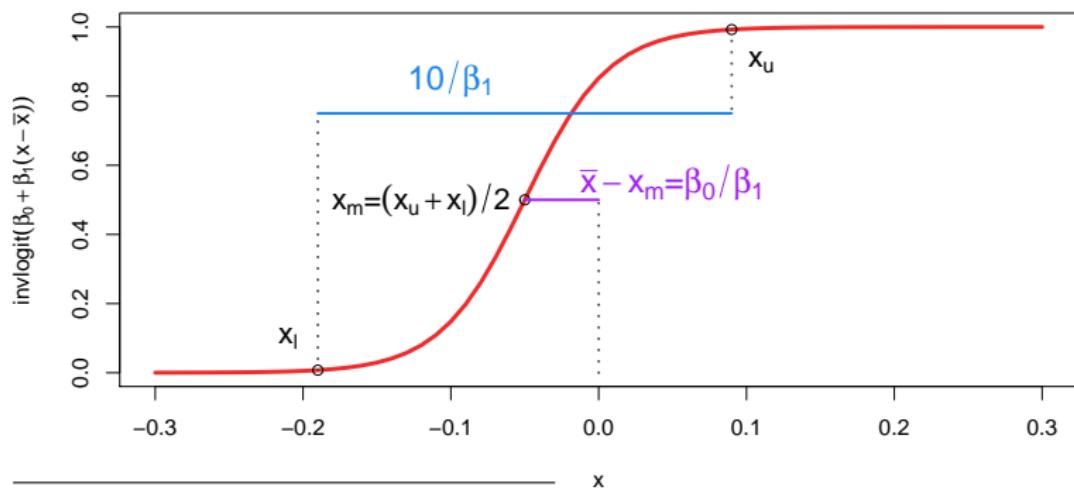
Generalised Linear Models

Bayesian Inference: Binomial and Dead Beetles

- β_1 determines how fast the change in probability happens with respects to x . β_0 determines the horizontal displacement of this change.⁷

$$-5 \approx \text{logit}(0.005) = \beta_0 + \beta_1(x_l - \bar{x})$$

$$5 \approx \text{logit}(0.995) = \beta_0 + \beta_1(x_u - \bar{x})$$



⁷Example with $\beta_0 = 1.75$ and $\beta_1 = 35$

Generalised Linear Models

Bayesian Inference: Binomial and Dead Beetles

- For the Bayesian analysis of beetle mortality rate, we need three components: data, initial values, and model statement (with priors and likelihood).
- The election of different initial values allows us to detect possible multimodalities of the model.
- If one has an idea of the range of possible values for β_0 and β_1 , five initial values corresponding to the 0.05, 0.25, 0.5, 0.75, 0.95 quantiles are one possibility.
- Another option is to randomly generate the initial values (from the priors or narrower distributions) using a function that generates a list of random initial values.

```
n <- length(n.exposed)
beetles.data <- list(n=n, n.exposed=n.exposed,
                      y.dead=y.dead, CS2.level=CS2.level*100)
beetles.inits <- function() {
  beta0 <- rnorm(1, 0, 1)
  beta1 <- rnorm(1, 0, 1)
  return( list(beta0=beta0, beta1=beta1) )
}
```

Generalised Linear Models

Bayesian Inference: Binomial and Dead Beetles

↪ Model Statement

```
beetles.model <- "model {  
  #Hyperparameters  
  beta.mu.0    <- 0  
  beta.tau.0   <- 0.0001  
  
  # prior  
  beta0 ~ dnorm(beta.mu.0,beta.tau.0)  
  betal ~ dnorm(beta.mu.0,beta.tau.0)  
  
  #Likelihood  
  for(i in 1:n) {  
    logit(mu[i])  <- beta0+betal*(CS2.level[i]-mean(CS2.level[]))  
    y.dead[i] ~ dbin(mu[i],n.exposed[i])  
  }  
}"
```

Generalised Linear Models

Bayesian Inference: Binomial and Dead Beetles

↪ Call from R to JAGS

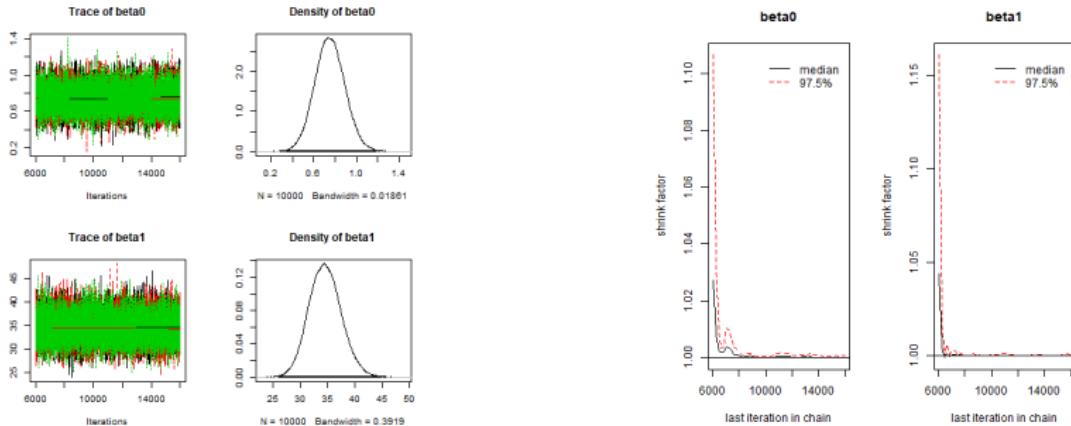
```
# Run JAGS to the completion of the "adaption" stage
results.beetles.A <- jags.model(file=textConnection(beetles.model),
data=beetles.data, inits=beetles.inits, n.chains=3)
# Burn-in of 5000 iterations
update(results.beetles.A, n.iter=5000)
# Longer run for making inferences, assuming chains have converged
results.beetles.B <- coda.samples(results.beetles.A,
variable.names=c("beta0", "beta1"), n.iter=10000)
```

Generalised Linear Models

Bayesian Inference: Binomial and Dead Beetles

→ Trace plots and Brooks-Gelman-Rubin (BGR) statistic

```
# Trace plots and density
plot(results.beetles.B)
# Brooks-Gelman-Rubin statistic (want a value near 1)
gelman.plot(results.beetles.B)
```



Generalised Linear Models

Bayesian Inference: Binomial and Dead Beetles

↪ Effective sample size

```
effectiveSize(results.beetles.B[[1]][, "beta0"])

##      var1
## 4985.016

effectiveSize(results.beetles.B[[1]][, "beta1"])

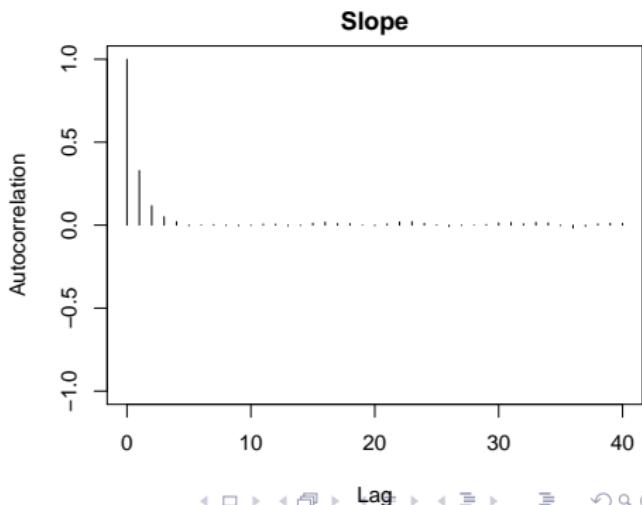
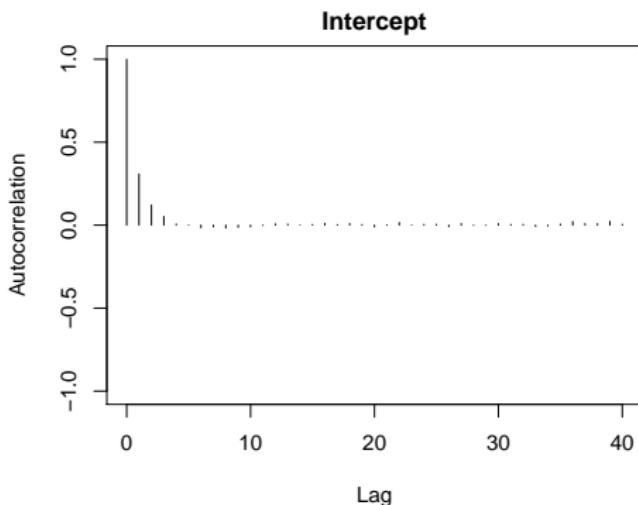
##      var1
## 5046.473
```

Generalised Linear Models

Bayesian Inference: Binomial and Dead Beetles

↪ Autocorrelation Function (ACF).

```
autocorr.plot(results.beetles.B[[1]][, "beta0"], main="Intercept")
autocorr.plot(results.beetles.B[[1]][, "beta1"], main="Slope")
```



Generalised Linear Models

Bayesian Inference: Binomial and Dead Beetles

→ Now look at the posteriors and compare to MLEs:

```
summary(results.beetles.B)
```

```
##           Mean      SD  Naive SE Time-series SE
## beta0    0.7561 0.140 0.0008065          0.0011320
## beta1    0.3464 0.029 0.0001689          0.0002375
```

```
## vs MLEs #####
## (Intercept)     ctr.CS2
##   0.7474     0.34286
```

→ The Monte Carlo error (Time-series SE) is much lower than 1/20th the size of the standard deviation of the parameters.

Generalised Linear Models

Bayesian Inference: Binomial and Dead Beetles

↪ Interpretation

```
# Joinning all the chains in one data.frame
results.beetles.output <- do.call(rbind.data.frame, results.beetles.B)
#interpretation
ilogit=function(x) { 1/(1+exp(-x)) }
#
cat("E[ilogit(beta0)] is", round(mean(ilogit(results.beetles.output$beta0)))

## E[ilogit(beta0)] is 0.68

cat("E[exp(beta1)] is", round(mean(exp(results.beetles.output$beta1))), 3))

## E[exp(beta1)] is 1.415
```

- ↪ The estimated posterior mean probability of a dead beetle for the average CS2 level (1.79) is 0.68.
- ↪ for a one-unit increase in the CS2 level (*100) we expect a 41.5% increase in the odds of a dead beetle.

Section 8

Categorical Covariates

Generalised Linear Models

Bayesian Analysis: Categorical Covariates

- ↪ Sometimes a covariate is categorical with K possible values. For example, there are $K=4$ crab carapace colours: light medium, medium, dark medium, and dark.
- ↪ To include such covariates in a regression model, one must create $K - 1$ “indicator” variables.
- ↪ The effect of indicator (“dummy”) variables is to shift the model intercept up or down, and there are different ways to construct indicators.
- ↪ Each indicator below takes value=1 for the specific colour is assigned to (Light Medium, Medium and Dark Medium) and value=0 otherwise.

$$g(\mu) = \beta_0 + \beta_1(Width - \overline{Width}) + \beta_2 I_{LtMed} + \beta_3 I_{Med} + \beta_4 I_{DkMed}$$

Thus, for a light medium crab, $g(\mu) = \beta_0 + \beta_1(Width - \overline{Width}) + \beta_2$ and similarly for the other cases.

- ↪ The category without a dummy variable (Dark) will be the reference category. This means that $\exp(\beta_0)$ is now interpreted as the expected number of satellite male crabs for a female crab with a dark carapace of average width.

Generalised Linear Models

Bayesian Inference: Categorical Covariates

↪ Data block⁸

```
n <- length(crabs$Obs)
crabs.mult.data <- list(n=n, Satellites=crabs$Satellites,
                         Width=crabs$Width,
                         lt.med.Ind=crabs$Colour=="lt.med",
                         med.Ind=crabs$Colour=="med",
                         dk.med.Ind=crabs$Colour=="dk.med")
```

⁸If we do not provide initial values, the same initial values are set for all the chains. Only to be done if one knows that there is no multimodality in the posterior.

Generalised Linear Models

Bayesian Analysis: Categorical Covariates

↪ Model Statement

```
crabs.mult.model <- "model {  
  #Hyperparameters  
  beta.mu.0    <- 0  
  beta.tau.0   <- 0.001  
  # prior  
  beta0        ~ dnorm(beta.mu.0,beta.tau.0)  
  beta.width   ~ dnorm(beta.mu.0,beta.tau.0)  
  beta.lt.med ~ dnorm(beta.mu.0,beta.tau.0)  
  beta.med    ~ dnorm(beta.mu.0,beta.tau.0)  
  beta.dk.med ~ dnorm(beta.mu.0,beta.tau.0)  
  #Likelihood  
  for(i in 1:n) {  
    log(mu[i])  <- beta0+beta.width*(Width[i]-mean(Width[])) +  
                  beta.lt.med*lt.med.Ind[i] + beta.med*med.Ind[i] +  
                  beta.dk.med*dk.med.Ind[i]  
    Satellites[i] ~ dpois(mu[i])  
  }  
}"
```

Generalised Linear Models

Bayesian Inference: Categorical Covariates

→ Call from R to JAGS:

```
# Run JAGS to the completion of the "adaption" stage
results.crabs.mult.A <- jags.model(file=textConnection(crabs.mult.model),
                                         data=crabs.mult.data,
                                         n.chains=3)

# Burn-in of 5000 iterations
update(results.crabs.mult.A, n.iter=5000)

# Longer run for making inferences, assuming chains have converged
results.crabs.mult.B <- coda.samples(results.crabs.mult.A,
                                         variable.names=c("beta0", "beta.width", "beta.lt.med",
                                                          "beta.med", "beta.dk.med"),
                                         n.iter=10000)
```

Generalised Linear Models

Bayesian Inference: Categorical Covariates

→ Now look at the summary of the posteriors:

```
summary(results.crabs.mult.B)
```

	Mean	SD	Naive SE	Time-series SE	
## beta.dk.med	0.0103	0.18561	0.0010716	0.0058468	
## beta.lt.med	0.4553	0.21197	0.0012238	0.0059372	
## beta.med	0.2581	0.16901	0.0009758	0.0059801	
## beta.width	0.5443	0.06831	0.0003944	0.0005911	
## beta0	0.8159	0.15520	0.0008960	0.0054734	
##					
	2.5%	25%	50%	75%	97.5%
## beta.dk.med	-0.34411	-0.1164	0.006755	0.1345	0.3839
## beta.lt.med	0.04373	0.3120	0.452149	0.5967	0.8727
## beta.med	-0.06382	0.1417	0.255108	0.3699	0.5993
## beta.width	0.40846	0.4989	0.544873	0.5907	0.6761
## beta0	0.49562	0.7127	0.821404	0.9232	1.1067

→ We can immediately spot an increasing pattern towards lighter carapace colours.



Generalised Linear Models

Bayesian Inference: Categorical Covariates

↪ Interpretation

```
# Joinging all the chains in one data.frame
results.crabs.mult.output <- do.call(rbind.data.frame, results.crabs.mult.B)
#interpretation
cat("E[exp(beta_LightMed)] is",
    round(mean(exp(results.crabs.mult.output$beta.lt.med)), 3))

## E[exp(beta_LightMed)] is 1.613

#... similarly for the other colours
```

- ↪ A female crab with a light-medium coloured carapace attracts on average 1.613 times more satellite male crabs than a female with dark coloured carapace, *while the other variables are held constant.*
- ↪ Equivalently, there is a 61% increase in the expected number of satellite male crabs attracted by a female crab with light medium carapace compared to a female with dark carapace, *while the other variables are held constant.*

Generalised Linear Models

Bayesian Analysis: Categorical Covariates

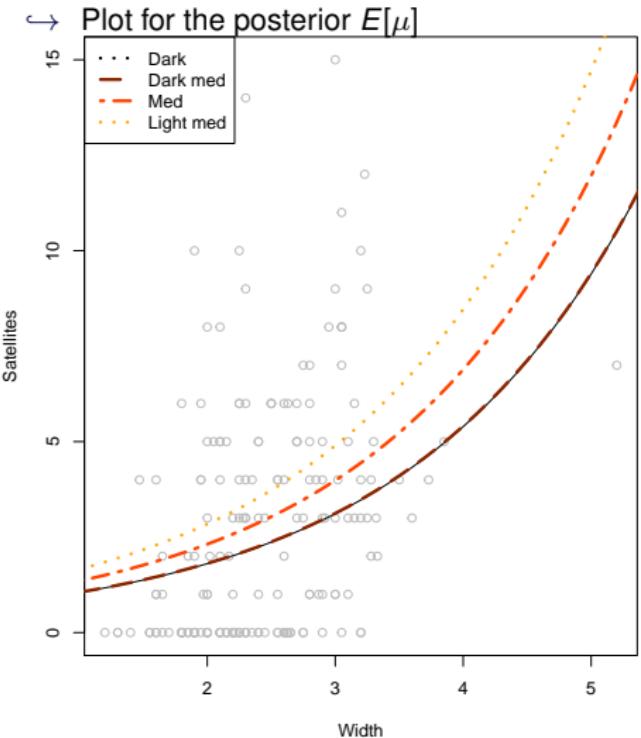
→ Plot for the posterior $E[\mu]$ (R code)

```
x <- seq(1, 6, 0.1)
m.crabs <- summary(results.crabs.mult.B)$statistics[, "Mean"]
plot(crabs$Width, crabs$Satellites, col="gray", xlab="Width", ylab="Satellites")
PostMean_mu=matrix(NA, 4, length(x))
#
for( k in 1:length(x)) {
PostMean_mu[1,k]=mean(exp(results.crabs.mult.output$beta0+
results.crabs.mult.output$beta.width*(x[k]-mean(crabs$Width))))
PostMean_mu[2,k]=mean(exp(results.crabs.mult.output$beta0+
results.crabs.mult.output$beta.width*(x[k]-mean(crabs$Width))+

results.crabs.mult.output$beta.dk.med))
#...
}
lines(x,PostMean_mu[1,],lwd=4,lty=3)
lines(x,PostMean_mu[2,],col="orangered4", lwd=3, lty=5)
#...
legend("topleft", c("Dark", "Dark med", "Med", "Light med"), col = c("black",
"orangered4", "orangered", "orange"), lty = c(1,5,6,3), lwd = 3)
```

Generalised Linear Models

Bayesian Analysis: Categorical Covariates



Section 9

Computation of posterior and predictive probabilities

Generalised Linear Models

Bayesian Analysis: Direct calculations using the simulations

- ↪ to easily extract simulation chains

```
library(runjags)
fit.crabs <- as.data.frame(combine.mcmc(results.crabs.mult.B))
# or use function do.call as shown in slide 71
```

- ↪ It can be used to calculate probabilities, e.g. $Pr(\beta_{Med} > 0 | y)$.

```
mean(fit.crabs$beta.med>0)

## [1] 0.9388333
```

- ↪ One can also obtain simulations for the predictive distribution $\int f(y^*|\beta)p(\beta|y)d\beta$ by plugging-in posterior samples of β into the likelihood of the new observation y^* , for example $p(y^*|y, \text{Width}^* = 3, \text{Colour}^* = Med)$:

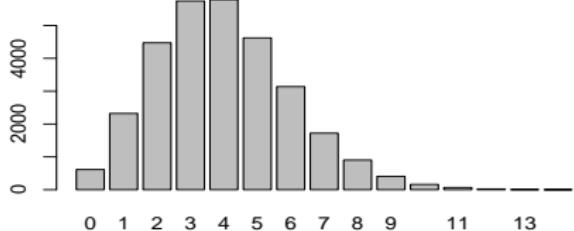
```
mu_star <- exp(fit.crabs$beta0 +
                  fit.crabs$beta.width*(3-mean(crabs$Width)) +
                  fit.crabs$beta.med)
y_star <- rpois(length(mu_star), mu_star)
```

Generalised Linear Models

Bayesian Analysis: Direct calculations using the simulations

- ↪ In this way we can plot the distribution $p(y^*|y, \text{Width}^* = 3, \text{Colour}^* = \text{Med})$

```
barplot(table(y_star) )
```



- ↪ And calculate predictive probabilities such as $Pr(y^* \geq 3|y, \text{Width}^* = 3, \text{Colour}^* = \text{Med})$

```
mean(y_star>=3)
```

```
## [1] 0.7530667
```

Section 10

Bayesian GLM: from JAGS to INLA

Generalised Linear Models

From JAGS to INLA: Categorical Covariates

- We repeat the analysis of the crab data with categorical covariates.
- In INLA, the log link function is the only available option for Poisson likelihood. This is the default link, so it does not have to be specified explicitly
- In INLA, the K-1 indicators are automatically created by the system when we use categorical variables in the regression formula

```
####Crab data with categorical covariates - INLA ####
#
#Creating centered Width covariate
crabs$Width.ctr=crabs$Width-mean(crabs$Width)
#Priors for the regression coefficients.
prior.beta <- list(mean.intercept = 0, prec.intercept = 0.01,
                     mean = 0, prec = 0.01)
#Fitting the model in INLA
#"control.fixed=prior.beta" sets regression coeff. priors.
m2.I <- inla(formula = Satellites ~ Width.ctr+Colour, family="poisson",
              data=crabs, control.fixed=prior.beta)
```

Generalised Linear Models

From JAGS to INLA: Categorical Covariates

→ Now look at the summary of the posteriors:

```
summary(m2.I)
```

```
##               mean      sd 0.025quant 0.5quant 0.975quant mode kld
## (Intercept) 0.828 0.149       0.523    0.832     1.109 0.841  0
## Width.ctr   0.546 0.068       0.410    0.547     0.678 0.548  0
## Colourdk.med 0.003 0.180      -0.344   0.000     0.364 -0.004  0
## Colourlt.med 0.453 0.208       0.045    0.452     0.863 0.452  0
## Colourmed   0.248 0.163      -0.062   0.244     0.578 0.237  0
```

→ The summary statistics are essentially identical to what we got from JAGS.

Generalised Linear Models

From JAGS to INLA: Categorical Covariates

- Now we are going to plot the posterior mean of μ , as a function of the width, for a set of equally spaced widths, separately for each 4 colours.
- This can be done by including new rows in the dataset with the response variable (Satellites) set to NA.

```
#Create a sequence of widths at which the posterior means
#of the fitted values is computed
x=seq(1, 6, 0.1)
lx=length(x)

#Create new rows in the dataset for these widths,
#for all 4 colours, with the response Satellites set to NA
newdata <- rbind(data.frame(Width=x,Colour=as.factor("dk"),
    Satellites=NA), data.frame(Width=x,Colour=as.factor("dk.med"),
    Satellites=NA), data.frame(Width=x,Colour=as.factor("med"),
    Satellites=NA), data.frame(Width=x,Colour=as.factor("lt.med"),
    Satellites=NA))

#Join the new rows with the original dataframe,
#and include the centered width covariate
newdata=rbind(newdata, data.frame(Satellites=crabs$Satellites,
    Width=crabs$Width, Colour=crabs$Colour))
newdata$Width.ctr=newdata$Width-mean(crabs$Width)
```

Generalised Linear Models

From JAGS to INLA: Categorical Covariates

- ↪ We fit the model in INLA. Note that we need to set **control.predictor = list(compute = TRUE, link=1)**
- ↪ **link=1** tells INLA to compute the marginals and means of the fitted values μ_i for the rows where the response is set to NA
- ↪ normally, when the response is set to NA, the statistics in **inla.model\$summary.fitted.values** will refer to the linear predictor η_i , and not μ_i

#whenever we use a model with link function that is not the identity
 #and we want to compute the posterior marginal and mean of mu_i
 #and not eta_i,we need to set link=1 in control.predictor

```
m3.I <- inla(formula = Satellites ~ Width.ctr+Colour, family="poisson",
  data=newdata, control.fixed=prior.beta,
  control.predictor = list(compute = TRUE, link=1),
  control.compute=list(config=TRUE))
```

Generalised Linear Models

From JAGS to INLA: Categorical Covariates

- We extract the posterior mean of the fitted values by `m3.I$summary.fitted.values$mean`, and plot them.

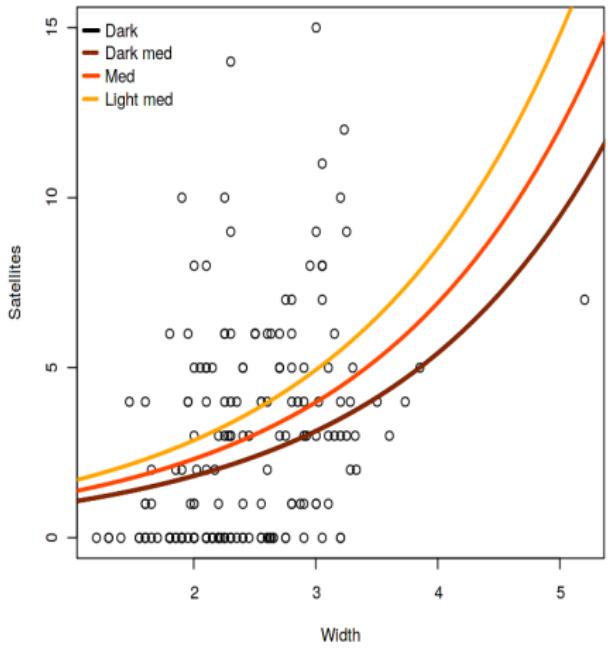
```
#Rows 1:lx contain the results for colour Light med,
#rows lx+1:2*lx contain the results for colour Med, etc.
```

```
fittedvaluesm=m3.I$summary.fitted.values$mean

plot(crabs$Width, crabs$Satellites, col="black", xlab="Width",
ylab="Satellites")
legend("topleft", c("Dark", "Dark med", "Med", "Light med"),
col = c("black", "orangered4", "orangered", "orange"),
lty = c(1,5,6,3), lwd = 3,bty = "n")
lines(x, fittedvaluesm[1:lx], col="black",lwd=3)
lines(x, fittedvaluesm[(lx+1):(2*lx)], col="orangered4", lwd=3, lty=5)
lines(x, fittedvaluesm[(2*lx+1):(3*lx)], col="orangered", lwd=3, lty=6)
lines(x, fittedvaluesm[(3*lx+1):(4*lx)], col="orange", lwd=3, lty=3)
```

Generalised Linear Models

From JAGS to INLA: Categorical Covariates



Generalised Linear Models

From JAGS to INLA: posterior and predictive probabilities

→ We look at the posterior and posterior predictive probabilities

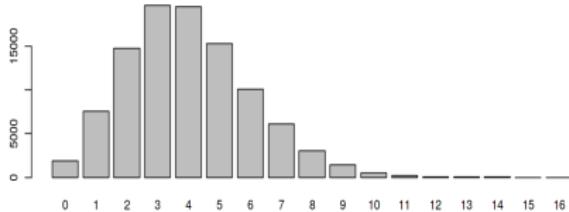
```
#We have already included the Width=3, Colour = Med
#in the newdata dataframe in row 51*2+21=123
#
#We only include this single linear predictor
#and the regression coefficient for colour med
#in the output of inla.posterior.sample by including
#selection=list(Predictor=123,Colourmed=1)
#
nbsamp=100000
crab.samples=inla.posterior.sample(n=nbsamp, result=m3.I,
                                     selection=list(Predictor=123,Colourmed=1))
#POSTERIOR PROBABILITY OF beta_MED > 0
#
beta.med.samples=inla.posterior.sample.eval(function(...) {Colourmed},
crab.samples)
mean(beta.med.samples>0)
#0.93691
```

Generalised Linear Models

From JAGS to INLA: posterior and predictive probabilities

→ and $P(y^*|y, \text{width}^* = 3, \text{Colour}^* = \text{Med})$

```
predictor.samples=inla.posterior.sample.eval(function(...) {Predictor}, crab.samples)
post.pred.samples=rpois(n=nbsamp, lambda=exp(predictor.samples))
#
barplot(table(post.pred.samples))
#
mean(post.pred.samples>=3)
#0.75814
```



The R code containing all of the analyses in Lecture 4 using JAGS and INLA is available on Learn.

Generalised Linear Models

Bayesian Analysis: further readings on Bayesian GLMs

- ↪ Reich BJ, Ghosh SK. Bayesian Statistical Methods. CRC press, 2019. Chapter 4, Section 4.3 and subsections.
- ↪ Gelman A, Carlin JB, Stern HS, et al. Bayesian Data Analysis, Third Edition. CRC press, 2014. Chapter 16, Introduction, Section 16.1, 16.2 "Offsets" subsection, and 16.4.

Bayesian Data Analysis

Daniel Paulin & Nicolò Margaritella

University of Edinburgh



Semester 2, 2020/2021

With thanks to Jonathan Gair, Rubén Amorós-Salvador, Ken Newman, Vanda Inácio and
Natalia Bochkina for much of the material

Outline

- 1 Review
- 2 Introduction
- 3 Salmon Survival Example
- 4 General Remarks
- 5 Salmon Survival Example (II)
- 6 Radon Levels Example
- 7 Identical Model
- 8 Independent Model
- 9 Hierarchical Model
- 10 Hierarchical Models: from JAGS to INLA

Section 1

Review

Hierarchical Models

Review

Structure of the Bayesian models considered thus far:

- 1 **The Likelihood or observation distribution** is a probability distribution for n independent observations, $\mathbf{y} = y_i, i=1, \dots, n$, that depends on one or more unknown parameters, $\boldsymbol{\theta} = (\theta_1, \dots, \theta_q)$.

$$f(y_1, y_2, \dots, y_n | \boldsymbol{\theta}) = \prod_{i=1}^n p(y_i | \boldsymbol{\theta}) \quad (1)$$

- With frequentist inference, we treat those parameters as constants $f(y_1, y_2, \dots, y_n | \boldsymbol{\theta}) = L(\boldsymbol{\theta} | \mathbf{y})$. A typical point estimate of $\boldsymbol{\theta}$ is the value of $\boldsymbol{\theta}$ maximizing the “probability” of \mathbf{y} , the MLE.

$$\text{MLE: } \hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} L(\boldsymbol{\theta} | y_1, \dots, y_n) \quad (2)$$

Hierarchical Models

Review

The Bayesian models considered thus far have had the following structure:

- 2 **The prior distribution** for θ reflects our uncertainty about θ and is in turn a function of hyperparameters ψ (so far, we consider them fixed).

$$\pi(\theta|\psi) \tag{3}$$

- 3 **The posterior distribution** for θ , calculated via the Bayes Theorem, i.e., the conditional distribution for θ given \mathbf{y} .

$$p(\theta|\psi, \mathbf{y}) = \frac{p(\theta, \mathbf{y}|\psi)}{m(\mathbf{y})} = \frac{f(\mathbf{y}|\theta)\pi(\theta|\psi)}{m(\mathbf{y})} \propto f(\mathbf{y}|\theta)\pi(\theta|\psi) \tag{4}$$

- 4 Two special cases you know well:

$$y \sim \text{Binomial}(n, p), p \sim \text{Beta}(\alpha, \beta) \rightarrow p|y \sim \text{Beta}(\alpha + y, \beta + n - y)$$

$$y_i \sim \text{Poisson}(\mu), i = 1, \dots, n; \mu \sim \text{Gamma}(\alpha, \beta) \rightarrow \mu|y \sim \text{Gamma}\left(\alpha + \sum_{i=1}^n y_i, \beta + n\right)$$

Hierarchical Models

Review

- 5 **Observation Distribution given Covariates:** Sometimes, there are explanatory variables, x , that affect the expected value of the observations: $\mathbb{E}[Y] = \mu = g^{-1}(x, \theta)$.

Special Cases of distributions and link functions $g()$ that you've worked with:

$$\text{Normal Linear Model: } y|x, \beta_0, \beta_1 \sim \text{Normal}(\mu = \beta_0 + \beta_1 x, \sigma^2) \quad (5)$$

$$\text{Poisson 'Loglinear' Model: } y|x, \beta_0, \beta_1 \sim \text{Poisson}(\mu = \exp(\beta_0 + \beta_1 x)) \quad (6)$$

$$\text{Bernoulli 'Logit' Model: } y|x, \beta_0, \beta_1 \sim \text{Bernoulli}\left(\mu = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)}\right) \quad (7)$$

Section 2

Introduction

Hierarchical Models

Introduction

Hierarchical models extend what we've done so far:

- There are j groups for our observations $y_{i,j}$.
- The behaviour of all groups is supposed to be similar (but not equal).
- The parameters θ_j for every group come from a common probability distribution (whose parameters need to be estimated) that can be considered a **Bayesian random effect**.

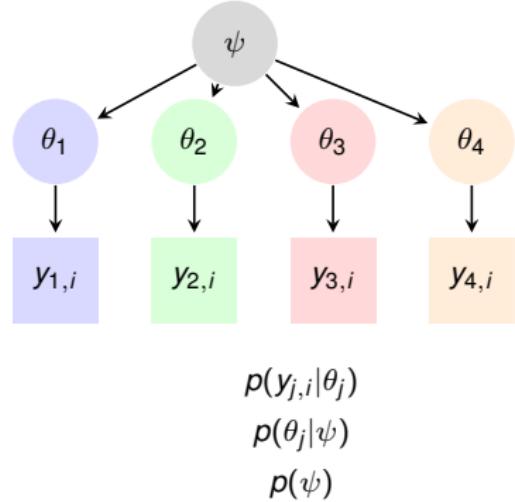
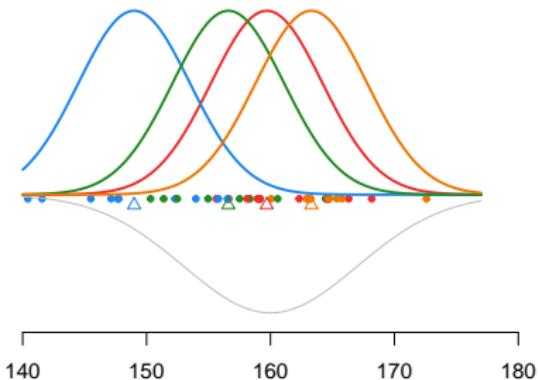
Examples:

- $y_{j,i}$ = household income in city j and household i in Scotland.
Household incomes within a city vary less than between cities.
- $y_{j,i}$ = score on a standardized mathematics exam in school j for a 10-year-old student i .
Scores vary more between different schools than between students in the same school.
- $y_{j,i}$ = indicator that infant has low birthweight ($<2000\text{mg}$) for infant i born in hospital j .
The probability that a newly born infant has a low birthweight differs between hospitals.

Hierarchical Models

Hierarchical Models

Another example: Height of women in different cities.

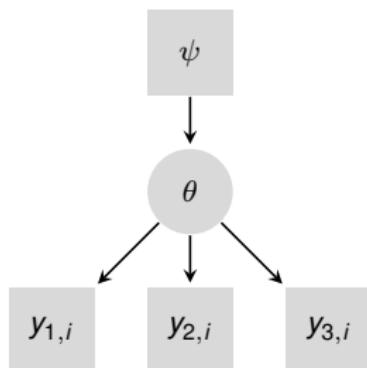


- The height distribution for each city is represented on top in colours.
- The mean for each city (triangle) is a realization of a common distribution of the means (represented below in gray).

Hierarchical Models

Common effect vs random effects vs fixed effects

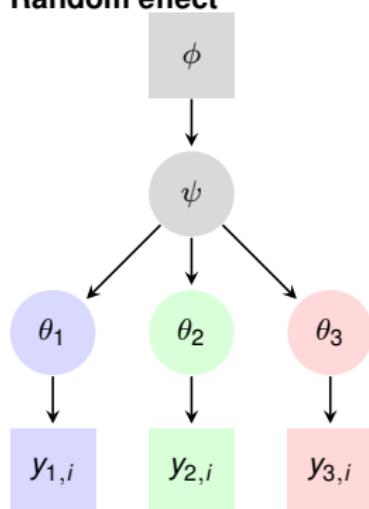
Common effect



$$y_{j,i}|\theta \sim dist(\theta)$$

$$\theta|\psi \sim dist(\psi)$$

Random effect

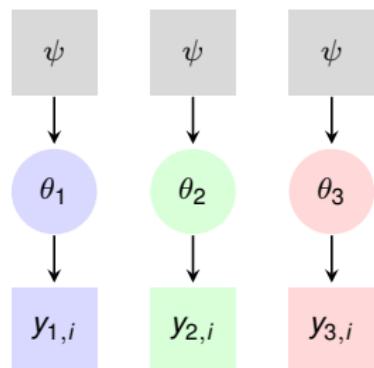


$$y_{j,i}|\theta_j \sim dist(\theta_j)$$

$$\theta_j|\psi \sim dist(\psi)$$

$$\psi|\phi \sim dist(\phi)$$

Fixed effect



$$y_{j,i}|\theta_j \sim dist(\theta_j)$$

$$\theta_j|\psi \sim dist(\psi)$$

Section 3

Salmon Survival Example

Hierarchical Models

Salmon Survival Example

- In a given year, several fish hatcheries located along rivers in Washington state, USA raise coho salmon from eggs to a juvenile stage. **Each hatchery releases a batch of juvenile fish** into the rivers. The fish then travel to the ocean and **some of them return** to the hatchery 3 years later.
- The **probability** that a juvenile salmon returns **varies between hatcheries** due to different hatchery practices and river conditions at the point of release.
- Let $J = \# \text{Hatcheries}$ and $p_j = \Pr(\text{Salmon from hatchery } j \text{ returns})$.
- Assume the p_j are independent random variables from a $\text{Beta}(\alpha, \beta)$, where α and β are unknown (NOT known and specified hyperparameters).



Photo from USGS.

Hierarchical Models

Salmon Survival Example

- Let n_j be the number of salmon released from hatchery j . Let y_j be the number of salmon that return 3 years later.
- This leads to a hierarchical model.

$$\begin{array}{lcl}
 \text{Hyperparameters } \alpha, \beta & : & ? \\
 \text{Parameters } p_j, j = 1, \dots, J & \stackrel{iid}{\sim} & \text{Beta}(\alpha, \beta) \\
 \text{Observations } I_{j,i}, j = 1, \dots, J, i = 1, \dots, n_j & \stackrel{\text{indep}}{\sim} & \text{Bernoulli}(p_j)
 \end{array}$$

where $I_{j,i}=1$ if fish returns, 0 otherwise.

Note: assuming independence between fish

$$y_j = \sum_{i=1}^{n_j} I_{j,i} \sim \text{Binomial}(n_j, p_j)$$

Hierarchical Models

Salmon Survival Example

- ↪ The Bayesian solution adds another layer, *hyperprior distributions* for the hyperparameters α and β . Let $h(\alpha, \beta|\phi)$ denote the joint prior for α and β with ϕ denoting the specified “hyper”-hyperparameters
- ↪ Then

$$\begin{aligned}
 \text{Hyperparameters } \alpha, \beta &\sim h(\alpha, \beta|\phi) \\
 \text{Parameters } p_j, j = 1, \dots, J &\stackrel{iid}{\approx} \text{Beta}(\alpha, \beta) \\
 \text{Observations } y_j, j = 1, \dots, J &\stackrel{indep}{\sim} \text{Binomial}(n_j, p_j)
 \end{aligned}$$

Section 4

General Remarks

Hierarchical Models

General Remarks

- ↪ Note that there is a hierarchy to this model, hence the name.
- ↪ The hierarchy can be viewed from the observed outcomes up:

$$\text{Data \& Likelihood} \quad f(y|\theta) \quad \Rightarrow \quad \text{Random Params \& Prior} \quad g(\theta|\psi) \quad \Rightarrow \quad \text{Hyperparameters \& Hyperprior} \quad h(\psi|\phi)$$

- ↪ More levels can be imagined: person → household → city → country
- ↪ Another name: Multi-Level Models
- ↪ Classical Frequentist Framework: Random Effects Models.

$$\begin{aligned} y_{j,i} &\sim \text{Normal} \left(\mu_i, \sigma^2 \right), j = 1, \dots, J, \quad i = 1, \dots, n_j \\ \mu_i &\sim \text{Normal} (\psi_1, \psi_2) \end{aligned}$$

where the μ_i are called random effects.

Hierarchical Models

General Remarks

- The model for the observations can include both random and non-random parameters; e.g.,

$$y_{j,i} \sim \text{Normal}(\theta_j + \beta x_{j,i}, \sigma^2)$$

where $\theta_j \sim \text{Normal}(\mu, \eta^2)$ and β is a constant. This is sometimes called a *Mixed Effects* model.

Hierarchical Models

General Remarks

- When the hierarchy for a model is Normal for the observations and Normal for the random parameters, there is often interest in the relative magnitude of the variances in both models:

$$y_{j,i} \sim \text{Normal}(\theta_j, \sigma_y^2), j = 1, \dots, J, i = 1, \dots, n_j$$

$$\theta_j \sim \text{Normal}(\mu, \sigma_\theta^2), j = 1, \dots, J$$

E.g., is the between observation variance, σ_y^2 , much larger than the between group variance, σ_θ^2 ? This is sometimes expressed by the *intraclass correlation*:

$$ICC = \frac{\sigma_\theta^2}{\sigma_\theta^2 + \sigma_y^2} \quad (8)$$

“If members of a group are unrelated, $ICC \rightarrow 0$ (the grouping contains no information); if members of a group are identical, $ICC \rightarrow 1$ (the grouping contains all the information)”
 Such an analysis is sometimes called *Variance Components* analysis.

Section 5

Salmon Survival Example (II)

Hierarchical Models

Returning to the Salmon Survival Example

A simulation example will be used because one can see how the estimated results differ from the true values.

- The survival probabilities are independent Beta random variables with an average survival probability, $\mathbb{E}[p]=0.15$, and a Coefficient of Variation (CV) of 0.30. Thus $p \sim \text{Beta}(9.3, 52.7)$.
- There are $J = 10$ hatcheries which have released a different number of fish n_j .
- The numbers surviving from hatchery j , $y_j \sim \text{Binomial}(n_j, p_j)$.

```
set.seed(9931)
true.a <- 9.3;  true.b <- 52.7
J <- 10
n.vector <- sample(x=50:100, size=J, replace=TRUE)
p.vector <- rbeta(J, shape1=true.a, shape2=true.b )
y.vector <- rbinom(n=J, size=n.vector, prob=p.vector)
# n =  86  83  71  60  96  85  58  95  63 100
# y =    8   12   10    8   16   14   10   24   12   17
```

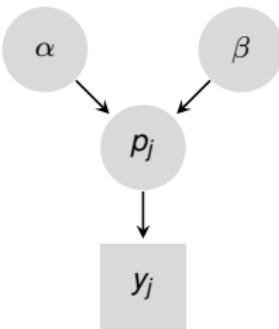
- The resulting values of p and the empirical survival probabilities, $\hat{p} = y/n$:

```
# p.vector      0.11 0.15 0.11 0.11 0.22 0.20 0.13 0.18 0.18 0.22
# phat.vector  0.09 0.14 0.14 0.13 0.17 0.16 0.17 0.25 0.19 0.17
```

Hierarchical Models

Salmon Survival Example

$$\begin{aligned}
 \text{Hyperparameters } \alpha, \beta &\sim h(\alpha, \beta | \psi) \\
 \text{Parameters } p_j, j = 1, \dots, J &\stackrel{iid}{\sim} \text{Beta}(\alpha, \beta) \\
 \text{Observations } y_j, j = 1, \dots, J &\stackrel{\text{indep}}{\sim} \text{Binomial}(n_j, p_j)
 \end{aligned}$$



Setting priors for the hyperparameters:

- ↪ Prior distribution $h(\alpha, \beta | \psi)$ is not a trivial decision.
- ↪ Can have a joint distribution or two independent marginal distributions.
- ↪ α and β must stay positive; e.g., lognormal, gamma, exponential or uniform (lowbounded by 0) priors are some possibilities.
- ↪ How to parameterize the prior distribution? Speculate on the induced values of p
- ↪ Suppose $\mathbb{E}[p] = 0.25$ with a coefficient of variation, $CV = \sqrt{\text{Var}(p)} / \mathbb{E}[p]$, of 0.30. Thus $\text{Var}(p)$ is $(\mathbb{E}[p]^2 CV^2) = (0.25 * 0.30)^2 = 9e-04$.
- ↪ Given $p \sim \text{Beta}(\alpha, \beta)$ and $\mathbb{E}[p]$ and $\text{Var}(p)$, can solve for α and β : $\alpha=8.1$ and $\beta=21.3$.

Hierarchical Models

Salmon Survival Example

Parameterize the lognormal prior distributions for α and β such that the median values are 8.1 for α and 21.3 for β . Then choose “relatively” large CV for the lognormal, e.g., $CV=0.8$, then lognormal $\sigma = \sqrt{\ln(CV^2 + 1)} = 0.70$:¹

$$\alpha \sim \text{Lognormal}(\ln(8.1), 0.70) \quad (9)$$

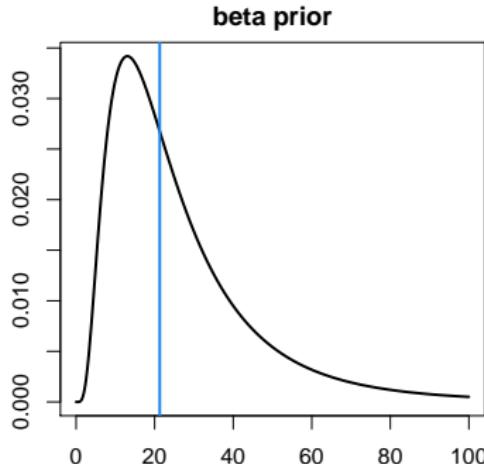
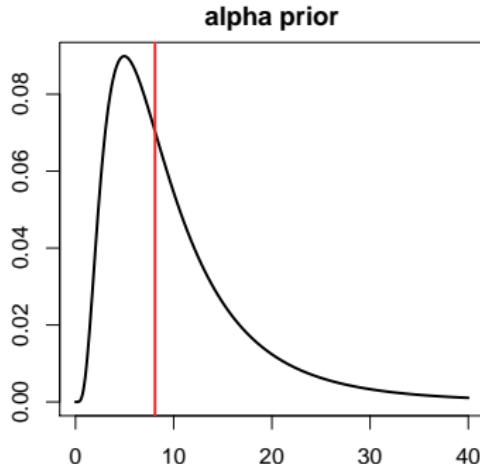
$$\beta \sim \text{Lognormal}(\ln(21.3), 0.70) \quad (10)$$

¹If $X \sim \text{Lognormal}(\mu, \sigma^2) \rightarrow CV(X) = \sqrt{\exp(\sigma^2) - 1}$

Hierarchical Models

Salmon Survival Example

The resulting priors for α and β with median denoted by red and blue vertical lines.



Hierarchical Models

Salmon Survival Example

The R and JAGS code

- Data Block: passing hyperparameters, too

```
salmon.data <- list(J=10, n=n.vector, y=y.vector,
                      ln.mu.alpha=log(8.1), ln.mu.beta=log(21.3),
                      sigma.theta=0.70)
```

- Initial Values Block: 5 IVs for α and β are chosen systematically from the lognormal prior distributions: the 0.05, 0.30, 0.50, 0.80, and 0.95 quantiles

```
prob.vec <- c(0.05, 0.30, 0.50, 0.80, 0.95)
alpha.IVs <- qlnorm(p=prob.vec, meanlog=log(8.1), sdlog=0.70)
# 2.6 5.6 8.1 14.6 25.6
beta.IVs <- qlnorm(p=prob.vec, meanlog=log(21.3), sdlog=0.70)
# 6.7 14.8 21.3 38.4 67.4
salmon.inits <- list()
for(i in 1:5) {
  salmon.inits[[i]] <- list(alpha=alpha.IVs[i],
                             beta=beta.IVs[i],
                             p=rbeta(J, 2, 3)) #initial values for p's
}
```

Hierarchical Models

Salmon Survival Example

↪ Model Block

```
salmon.model <- "model{

  #-- Prior for alpha and beta
  tau.theta    <- 1/pow(sigma.theta,2)
  alpha ~ dlnorm(ln.mu.alpha,tau.theta)
  beta   ~ dlnorm(ln.mu.beta,tau.theta)

  #-- pdf for survival probabilities
  # and the likelihood for the observed returns
  for(j in 1:J) {
    p[j] ~ dbeta(alpha,beta)
    y[j] ~ dbin(p[j],n[j])
  }

  expect.survival <- alpha/(alpha+beta)
} "
```

Hierarchical Models

Salmon Survival Example

↪ Call to JAGS:

```
burnin           <- 5000
inference.length <- 50000
salmon.res.A <- jags.model(file=textConnection(salmon.model),
                             data=salmon.data, inits=salmon.inits,
                             n.chains=length(salmon.inits),
                             quiet = TRUE)
update(salmon.res.A,n.iter=burnin)
salmon.res.B <- coda.samples(salmon.res.A,
                             variable.names=c("alpha", "beta", "expect.survival", "p"),
                             n.iter=inference.length)
```

Hierarchical Models

Salmon Survival Example

↪ Diagnostics:

```
# BGR statistic
gelman.diag(salmon.res.B)
# Number of effective size of the simulations
effectiveSize(salmon.res.B)
# Trace plots
plot(salmon.res.B)
# BGR plots
gelman.plot(salmon.res.B)
# Autocorrelation plots
autocorr.plot(salmon.res.B)
```

Hierarchical Models

Salmon Survival Example

↪ Summary statistics

```
summary(salmon.res.B)
# ...
#           Mean        SD   Naive SE Time-series SE
# alpha     9.5578  4.38067 8.761e-03    6.834e-02
# beta      47.8301 22.34306 4.469e-02   3.489e-01
# expect.survival 0.1682  0.02141 4.281e-05   7.209e-05
# ...
```

Compared to the true values of $\alpha = 9.3$, $\beta = 52.7$, and $E[p] = 0.15$.

↪ Monte Carlo simulation error relative to the standard deviations of the parameters.

$< 1/20 = 0.05?$

alpha MC/SD = 0.016,

beta MC/SD = 0.015,

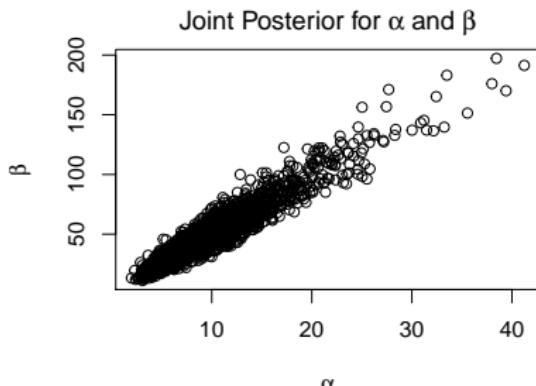
exp.S MC/SD = 0.0025

Hierarchical Models

Salmon Survival Example

The parameters α and β may be correlated which can affect the mixing rates.

```
# Another way of pooling information from all 5 chains
salmon.output <- do.call(rbind.data.frame,salmon.res.B)
# Just work with a random sample from the joint posterior
x <- sample(1:nrow(salmon.output),2000)
plot(salmon.output$alpha[x],salmon.output$beta[x],
      xlab=expression(alpha),ylab=expression(beta),
      main=expression(paste("Joint Posterior for ",alpha," and ",beta)))
```

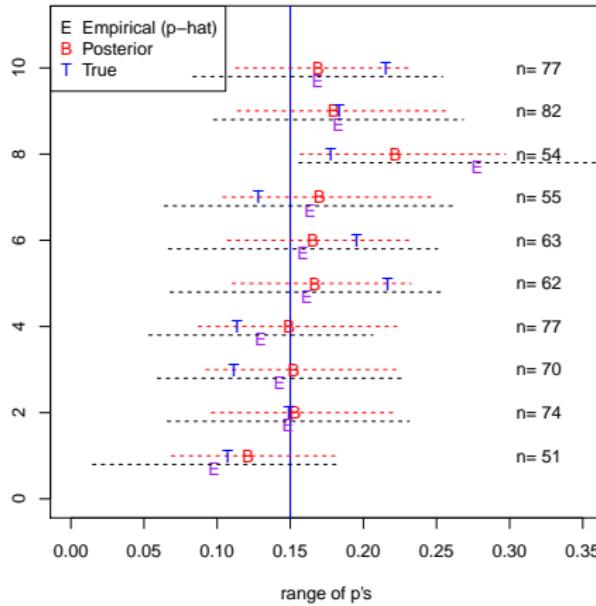


Hierarchical Models

Salmon Survival Example

Contrast the posterior estimates of p to the individual MLEs

Posterior vs empirical estimates for p's



Hierarchical Models

Salmon Survival Example

This plot demonstrates some important features of Bayesian HM:

Shrinkage: Compared to the individual group MLEs, the posterior means for the random parameters move toward their expected values.

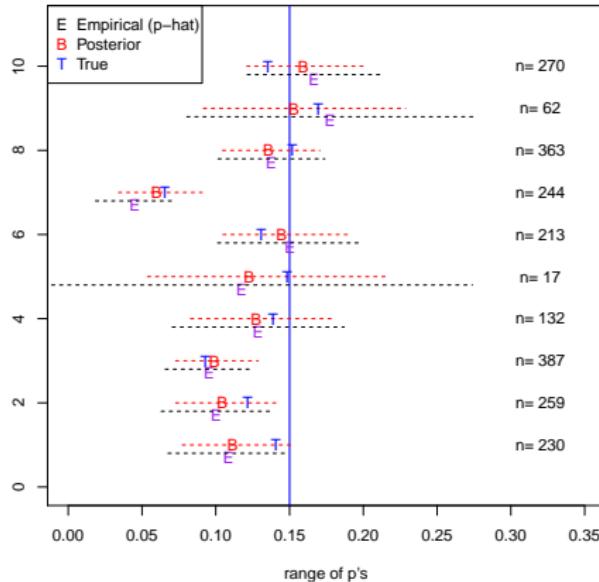
Increased Precision: The 95% (posterior) intervals are narrower than the 95% (confidence) intervals for the individual group-based estimates. This increased precision is often most noticeable for groups with smaller sample sizes.

Borrowing of Strength: this is a related point—an estimate of p_j for a group j with smaller n_j is indirectly getting additional information about p_j from the estimated distribution of all the p_j 's (which is based on all the data from the J hatcheries).

Hierarchical Models

Salmon Survival Example

Demonstration of Shrinkage and Borrowing of Strength with smaller n_j 's:



Section 6

Radon Levels Example

Hierarchical Models

Radon Levels: Hierarchical Normal

- ↪ “Radon is a radioactive gas, we can't see, smell or taste it: you need special equipment to detect it. It comes from the rocks and soil found everywhere in the UK. The radon level in the air we breathe outside is very low but can be higher inside buildings. [...] Radon is everywhere but some parts of the country are more likely to have high levels in buildings than others. [...] Any exposure to this type of radiation is a risk to health - radiation is a form of energy and can cause damage in living tissues increasing the risk of cancer.”²
- ↪ The data we will examine here comes from a large United States survey and we just examine radon measurements from houses in counties in the state of Minnesota.³

²From Public Health England: <http://www.ukradon.org/information/>

³Data source and examples from Patrick Breheny, University of Iowa.

Hierarchical Models

Radon Levels: Hierarchical Normal

- 919 observations from randomly chosen houses from 85 of the 87 counties in Minnesota (1986-1988).
- Radon measurements (in pCi/l: picoCuries per liter) were made by “using charcoal canisters deployed during the winter months on the lowest livable area of the house”.
- The first few records of the reduced data set:

```
radon.df <- read.csv(file="Minnesota_radon_data.csv", header=TRUE)
radon.df$logradon <- pmax(log(0.1), log(radon.df$radon) )
radon.df$j.county <- as.numeric(as.factor(radon.df$county))
head(radon.df)

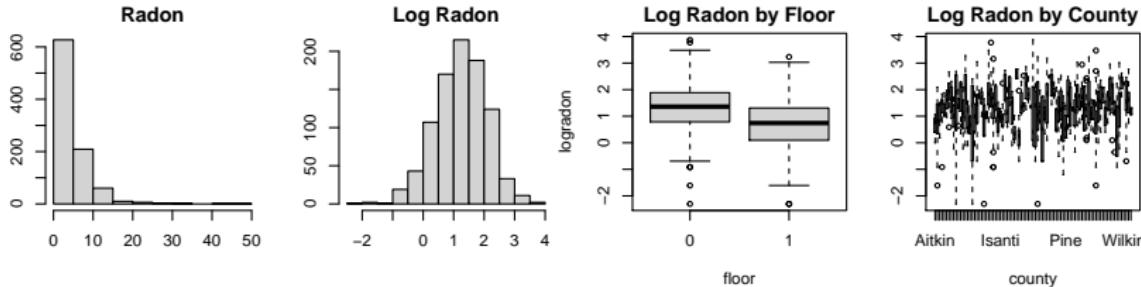
##      radon county floor room logradon j.county
## 1    2.2 Aitkin     1     3  0.7884574      1
## 2    2.2 Aitkin     0     4  0.7884574      1
## 3    2.9 Aitkin     0     4  1.0647107      1
## 4    1.0 Aitkin     0     4  0.0000000      1
## 5    3.1 Anoka      0     4  1.1314021      2
## 6    2.5 Anoka      0     4  0.9162907      2
```

- The floor variable = 0 if the lowest level was a basement and = 1 if it was the ground floor.

Hierarchical Models

Radon Levels: Hierarchical Normal

The radon data are strongly right-skewed and a natural log transformation was applied (after adding 0.1 to all measurements to handle the 3 cases with values = 0).



- ↪ Radon levels tend to be higher in houses with basements
- ↪ High variability in radon levels within the counties.
- ↪ Considerable variability number of houses sampled per county.

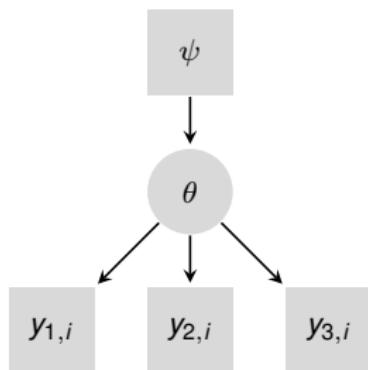
```
summary(as.numeric(table(radon.df$county)))
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	1.00	3.00	5.00	10.81	10.00	116.00

Hierarchical Models

Common effect vs random effects vs fixed effects

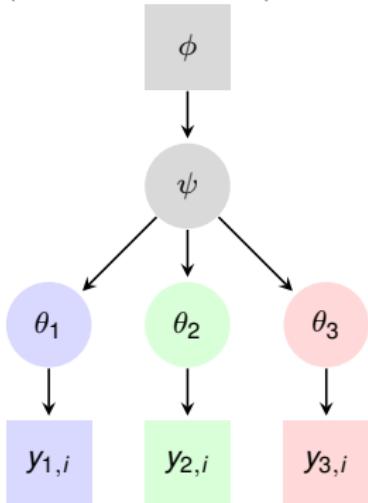
Common effect
(Identical model)



$$y_{j,i} | \theta \sim dist(\theta)$$

$$\theta | \psi \sim dist(\psi)$$

Random effect
(Hierarchical model)

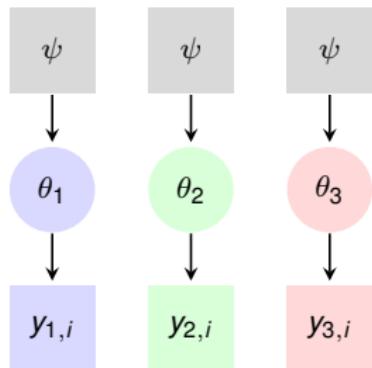


$$y_{j,i} | \theta_j \sim dist(\theta_j)$$

$$\theta_j | \psi \sim dist(\psi)$$

$$\psi | \phi \sim dist(\phi)$$

Fixed effect
(Independent model)



$$y_{j,i} | \theta_j \sim dist(\theta_j)$$

$$\theta_j | \psi \sim dist(\psi)$$

Hierarchical Models

Radon Levels: Hierarchical Normal

Mimicking Breheny, we now modeled log radon, y , as a function of the ground floor indicator in three different models. All of these assume an effect of not having basement ($I_{\text{ground.floor}} = 1$):

- ↪ **Identical model:** Same intercept for all the counties. Only non-basement has an effect.

$$y_i \stackrel{\text{indep}}{\sim} \text{Normal} \left(\alpha + \beta I_{\text{ground.floor},i}, \sigma^2 \right) \quad (11)$$

- ↪ **Hierarchical model:** Each county has a different intercept but the intercepts are random draws from the same distribution.

$$y_{j,i} \stackrel{\text{indep}}{\sim} \text{Normal} \left(\alpha_j + \beta I_{\text{ground.floor},j,i}, \sigma^2 \right), i = 1, \dots, n_j \quad (12)$$

$$\alpha_j \stackrel{\text{indep}}{\sim} \text{Normal} \left(\mu_\alpha, \sigma_\alpha^2 \right), j = 1, \dots, 85 \quad (13)$$

- ↪ **Independent model:** Each county has a different conditionally independent intercept.

$$y_{j,i} \stackrel{\text{indep}}{\sim} \text{Normal} \left(\mu_j + \beta I_{\text{ground.floor},j,i}, \sigma^2 \right), j = 1, \dots, 85, \quad i = 1, \dots, n_j \quad (14)$$

Section 7

Identical Model

Hierarchical Models

Radon Levels: Identical Model

Identical Model. The model and remarks.

There is no distinction according to county, the focus is on the general effect of having a basement or not.

$$y_i \stackrel{\text{indep}}{\sim} \text{Normal} \left(\alpha + \beta I_{\text{ground.floor},i}, \sigma^2 \right), i = 1, \dots, N \quad (15)$$

Hierarchical Models

Radon Levels: Identical Model

Identical Model. Data and Initials.

Priors for 3 parameters:

- $\alpha \sim \text{Normal}(0, \tau=0.01)$
- $\beta \sim \text{Normal}(0, \tau=0.01)$
- $\sigma \sim \text{Uniform}(0, 10)$. NOTE: specifying prior on σ not τ .

```
# Data block #includes hyperparameters
tau.alpha <- tau.beta <- 0.01
sigma.ub  <- 10
radon.ident.data <- list(n=nrow(radon.df), log.radon=radon.df$logradon,
                           floor=radon.df$floor, sigma.ub=sigma.ub,
                           tau.alpha=tau.alpha, tau.beta=tau.beta)

# Initial values
radon.ident.inits <- function(){
  list(alpha=rnorm(1, 0, 1/sqrt(tau.alpha)),
       beta=rnorm(1, 0, 1/sqrt(tau.beta)),
       sigma=runif(1, 0, sigma.ub))
}
```

Hierarchical Models

Radon Levels: Identical Model

Identical Model. Model.

```
# Model
radon.ident.model <- "model {
  #likelihood
  for(i in 1:n) {
    log.radon[i] ~ dnorm(mu[i],tau)
    mu[i] <- alpha + beta*floor[i]
  }

  # priors for intercept, slope and sigma
  alpha ~ dnorm(0,tau.alpha)
  beta ~ dnorm(0,tau.beta)
  tau <- 1/pow(sigma,2)
  sigma ~ dunif(0,sigma.ub)
}"
```

Hierarchical Models

Radon Levels: Identical Model

Identical Model. Inference.

```
radon.ident.res.A <- jags.model(file=textConnection(radon.ident.model),  
                                 data=radon.ident.data, inits=radon.ident.inits, n.chains=3,  
                                 quiet = TRUE)  
update(radon.ident.res.A, n.iter=2000)  
radon.ident.res.B <- coda.samples(radon.ident.res.A,  
                                    variable.names=c("alpha", "beta", "sigma"), n.iter=10000)
```

Hierarchical Models

Radon Levels: Identical Model

Identical Model. Results.

```
summary(radon.ident.res.B)
#           Mean      SD  Naive SE Time-series SE
# alpha    1.3270 0.02937 0.0001696      0.0001994
# beta     -0.6129 0.07299 0.0004214      0.0005010
# sigma    0.8238 0.01920 0.0001109      0.0001383
```

- ↪ Expected log radon in basement → $E(\alpha|y) = 1.33 \text{ pCi/l}$
- ↪ Expected log radon in ground floor → $1.33 - 0.61 = 0.71 \text{ pCi/l}$
- ↪ Large relative variations:
 - CV in basement $\sigma/1.33 = 62\%$
 - CV in ground floor $\sigma/0.71 = 115\%$

Section 8

Independent Model

Hierarchical Models

Radon Levels: Independent Model

Independent Model. The model and remarks.

Each county has a different intercept, i.e. its own basement level radon, but the ground floor effect is identical.

$$y_{j,i} \stackrel{\text{indep}}{\sim} \text{Normal}(\alpha_j + \beta I_{\text{ground.floor},j,i}, \sigma^2), j = 1, \dots, 85, \quad i = 1, \dots, n_j \quad (16)$$

Instead of creating 85 dummy variables, we will use the index j to define all the α_j with a `for` loop in JAGS.

Hierarchical Models

Radon Levels: Independent Model

Independent Model. Data and Initial values.

Priors for $85+2=87$ parameters:

- $\mu_j \sim \text{Normal}(0, \tau=0.01)$, $j=1, \dots, 85$
- $\beta \sim \text{Normal}(0, \tau=0.01)$
- $\sigma \sim \text{Uniform}(0, 10)$.

```
# Data block #includes hyperparameters
radon.indep.data <- list(n=nrow(radon.df), log.radon=radon.df$logradon,
                         floor=radon.df$floor, sigma.ub=10,
                         tau.alpha=0.01, tau.beta=0.01,
                         j.county=radon.df$j.county,
                         J=max(radon.df$j.county))

# Initial values
radon.indep.inits <- function(){
  list(alpha=rnorm(max(radon.df$j.county), 0, 1/sqrt(tau.alpha)),
       beta=rnorm(1, 0, 1/sqrt(tau.beta)),
       sigma=runif(1, 0, sigma.ub))
}
```

Hierarchical Models

Radon Levels: Independent Model

Independent Model. Model.

```
# Model
radon.indep.model <- "model {
  #likelihood
  for(i in 1:n) {
    log.radon[i] ~ dnorm(mu[i],tau)
    mu[i] <- alpha[j.county[i]] + beta*floor[i]
  }

  # priors for independent intercepts per county
  for(j in 1:J){
    alpha[j] ~ dnorm(0,tau.alpha)
  }

  # priors for intercept, slope and sigma
  beta ~ dnorm(0,tau.beta)
  tau <- 1/pow(sigma,2)
  sigma ~ dunif(0,sigma.ub)
}"
```

Hierarchical Models

Radon Levels: Independent Model

Independent Model. Inference.

```
radon.indep.res.A <- jags.model(file=textConnection(radon.indep.model),  
                                 data=radon.indep.data, inits=radon.indep.inits, n.chains=3,  
                                 quiet = TRUE)  
update(radon.indep.res.A, n.iter=2000)  
radon.indep.res.B <- coda.samples(radon.indep.res.A,  
                                    variable.names=c("alpha", "beta", "sigma"), n.iter=10000)
```

Hierarchical Models

Radon Levels: Independent Model

Independent Model. Results.

```
summary(radon.indep.res.B)
#               Mean        SD   Naive SE Time-series SE
# alpha[1]    0.8392  0.37854  0.0021855      0.0021957
# alpha[2]    0.8749  0.10514  0.0006070      0.0006115
# alpha[3]    1.5219  0.43982  0.0025393      0.0025874
# ...
# alpha[85]   1.1830  0.53301  0.0030773      0.0030774
# beta       -0.7190  0.07322  0.0004227      0.0005843
# sigma      0.7573  0.01856  0.0001071      0.0001461
```

- Expected difference of log radon in ground floor with respect of the basement → 0.72 piC/l, compared to 0.61 piC/l for the Identical Model.
- Mean of the expected log radon in basement for all counties → $\frac{1}{J} \sum_j E(\alpha_j | y) = 1.50$ piC/l, compared to 1.33 piC/ for the Identical Model.
- Reduction in relative variation: CV in basement 50% (vs 62% IM); CV in ground floor 97% (vs 115% IM)

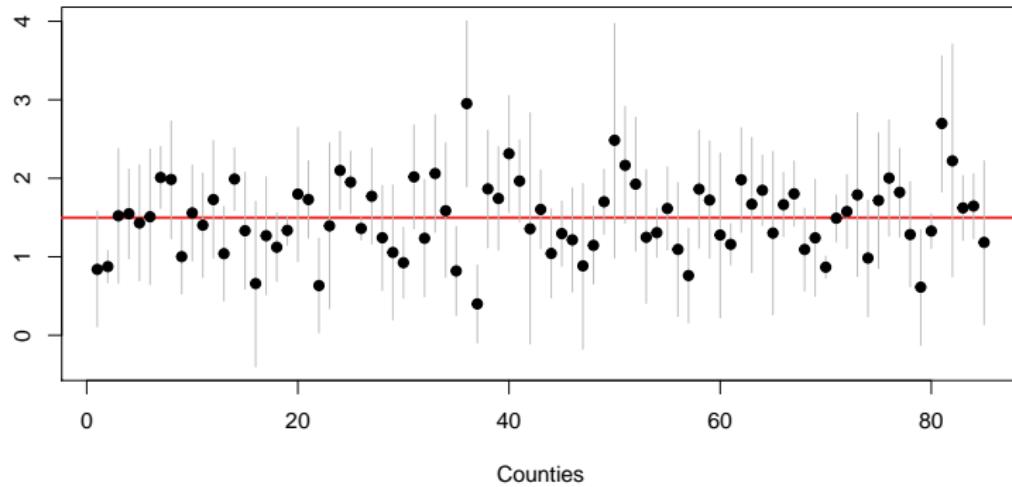
Hierarchical Models

Radon Levels: Independent Model

Independent Model. Results (II).

There is large variability among counties for $\alpha_j|y$ (posterior log radon in basements).

County intercepts: Independent Model



Section 9

Hierarchical Model

Hierarchical Models

Radon Levels: Hierarchical Model

Hierarchical Model.

- Each county has a different intercept but the intercepts are random draws from the same distribution.

$$y_{j,i} \stackrel{\text{indep}}{\sim} \text{Normal} \left(\alpha_j + \beta I_{\text{basement},j,i}, \sigma_{\text{obs}}^2 \right), i = 1, \dots, n_j \quad (17)$$

$$\alpha_j \stackrel{\text{indep}}{\sim} \text{Normal} \left(\mu_\alpha, \sigma_\alpha^2 \right), j = 1, \dots, 85 \quad (18)$$

- Now there are 4 parameters needing priors:

- $\mu_\alpha \sim \text{Normal}(0, \tau=0.01)$
- $\sigma_\alpha \sim \text{Uniform}(0, 20)$
- $\beta \sim \text{Normal}(0, \tau=0.01)$
- $\sigma_{\text{obs}} \sim \text{Uniform}(0, 20)$

Hierarchical Models

Radon Levels: Hierarchical Model

Hierarchical Model. Data and Initial values.

```
# Data block #includes hyperparameters
radon.hier.data <- list(n=nrow(radon.df), log.radon=radon.df$logradon,
                      floor=radon.df$floor,
                      sigma.ub=10, sigma.alpha.ub=20,
                      tau.mu.alpha=0.01, tau.beta=0.01,
                      j.county=radon.df$j.county,
                      J=max(radon.df$j.county))

# Initial values
radon.hier.inits <- function() {
  list(mu.alpha=rnorm(1, 0, 1/sqrt(0.01)),
       beta=rnorm(1, 0, 1/sqrt(0.01)),
       sigma=runif(1, 0, 10),
       sigma.alpha=runif(1, 0, 20))
}
```

Hierarchical Models

Radon Levels: Hierarchical Model

Hierarchical Model. Model.

```
# Model
radon.hier.model <- "model {
  ## likelihood
  for(i in 1:n) {
    log.radon[i] ~ dnorm(mu[i],tau)
    mu[i] <- alpha[j.county[i]] + beta*floor[i]
  }
  #### priors for independent intercepts per county
  for(j in 1:J){ alpha[j] ~ dnorm(mu.alpha, tau.alpha)      }
  #### Hyperpriors for mu.alpha and tau.alpha
  mu.alpha      ~ dnorm(0, tau.mu.alpha)
  tau.alpha     <- 1/pow(sigma.alpha,2)
  sigma.alpha   ~ dunif(0, sigma.alpha.ub)
  #### priors for intercept, slope and sigma
  beta   ~ dnorm(0,tau.beta)
  tau    <- 1/pow(sigma,2)
  sigma  ~ dunif(0,sigma.ub)
}"
```

Hierarchical Models

Radon Levels: Hierarchical Model

Hierarchical Model. Inference.

```
radon.hier.res.A <- jags.model(file=textConnection(radon.hier.model),
                                 data=radon.hier.data, inits=radon.hier.inits, n.chains=3,
                                 quiet = TRUE)
update(radon.hier.res.A, n.iter=2000)
radon.hier.res.B <- coda.samples(radon.hier.res.A,
                                   variable.names=c("alpha", "mu.alpha", "sigma.alpha", "beta", "sigma"),
                                   n.iter=10000)
```

Hierarchical Models

Radon Levels: Hierarchical Model

Hierarchical Model. Results.

```
summary(radon.hier.res.B)
```

	Mean	SD	Naive SE	Time-series SE
# alpha[1]	1.1853	0.25442	0.0014689	0.0015879
# alpha[2]	0.9282	0.10094	0.0005828	0.0005978
# alpha[3]	1.4811	0.26858	0.0015506	0.0016270
# ...				
# alpha[85]	1.3829	0.28523	0.0016468	0.0016915
# beta	-0.6935	0.07136	0.0004120	0.0006041
# mu.alpha	1.4624	0.05340	0.0003083	0.0005780
# sigma	0.7566	0.01849	0.0001068	0.0001437
# sigma.alpha	0.3359	0.04682	0.0002703	0.0007318

- The effects on radon of the ground floor are quite the same as before (-0.69 vs -0.71 for Ind.M), as is the expected basement effect (1.46 vs 1.50 for Ind.M).
- The Intraclass Correlation Coefficient:

$$ICC \approx \frac{\sigma_{\alpha}^2}{\sigma_{\alpha}^2 + \sigma_{obs}^2} = \frac{0.3359^2}{0.3359^2 + 0.7566^2} = 0.165$$

indicating that the between county variation is not as high as the within county variation. ↗ ↘ ↙

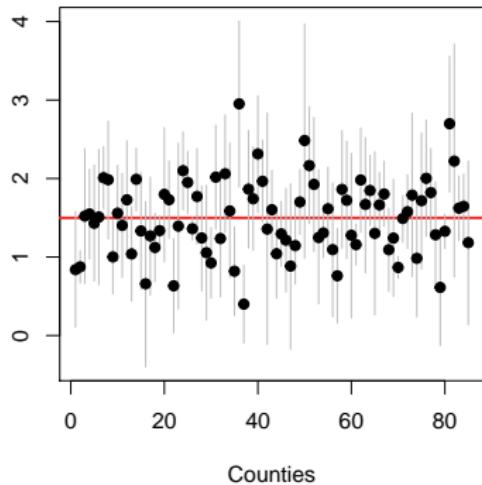
Hierarchical Models

Radon Levels: Hierarchical Model

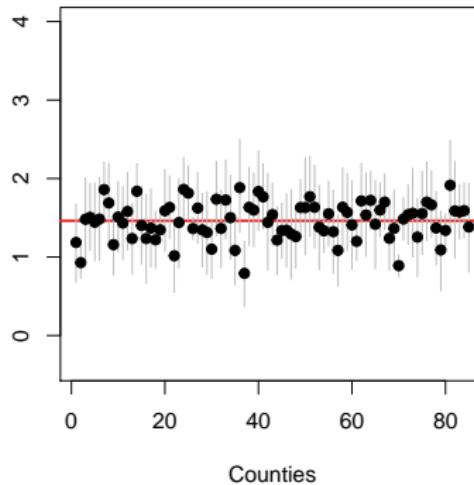
Hierarchical Model. Results (II).

The 95% credible intervals for the county effects are generally much narrower for the hierarchical model (right plot) than for the Independence model (left plot), with the same y-axis ranges.

County intercepts: Independent Mod



County intercepts: Hierarchical Model



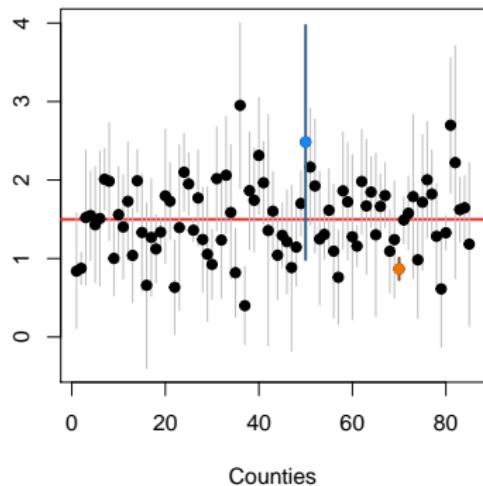
Hierarchical Models

Radon Levels: Hierarchical Model

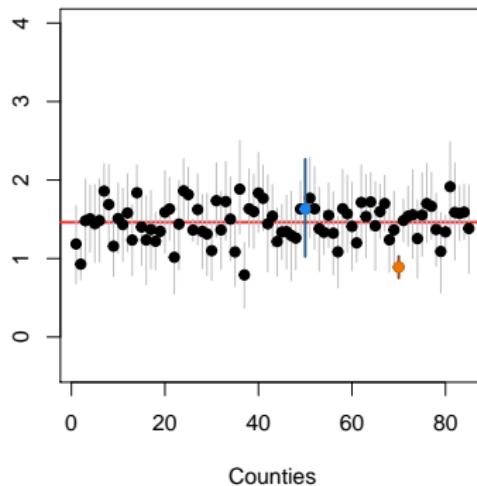
Hierarchical Model. Results (III).

The Shrinkage and Borrowing of Strength varies as a function of sample size as can be seen by two extreme cases below: a county with 1 measurement (Murray in blue) vs a county with 113 measures (St Louis in orange).

County intercepts: Independent Mod



County intercepts: Hierarchical Model



Section 10

Hierarchical Models: from JAGS to INLA

Hierarchical Models

From JAGS to INLA: Hierarchical Model

- We repeat the analysis of Radon data with the hierarchical model.
- Recall that in this model each county has a random intercept, but these share the same mean and variance, and we use some priors on these hyperparameters (i.e. the common mean and variance).
- Recall also that INLA stores log-transformed precision parameters hence we need to define σ_α and σ_{obs} , in terms of $\log(\tau)$.

As in INLA we need to specify the prior on $\theta = \log(\tau)$ for $\tau = \frac{1}{\sigma^2}$, a transformation of r.v. is needed. As explained in Section 5.3.2 of Bayesian Inference in INLA (<https://becarioprecario.bitbucket.io/inla-gitbook/ch-priors.html>), the corresponding log-density $\log(\pi(\theta))$ on $\theta = \log(\tau) = -2 \log(\sigma)$ can be written as

$$\begin{aligned}\log(\pi(\theta)) &= \log(\pi_\sigma(\sigma = f(\theta))|df(\theta)/d\theta|) \\ &= \log(\pi_\sigma(\exp(-\theta/2))) - \theta/2 - \log(2).\end{aligned}$$

In the specific case of $\sigma \sim U[0, b]$, we have $\pi_\sigma(x) = \frac{1}{b}$ for $0 \leq x \leq b$, and 0 elsewhere. So after rearrangement, it follows that

$$\log(\pi(\theta)) = \begin{cases} -\log(b) - \theta/2 - \log(2) & \text{if } \theta \geq -2 \log(b) \\ -\infty & \text{if } \theta < -2 \log(b) \end{cases}$$

Hierarchical Models

From JAGS to INLA: Hierarchical Model

- We can input this prior in INLA using the "expression:" string, which allows us to specify the log-density of the prior in a format that is similar to R

```
# sigma_obs:
sigma.unif.prior = "expression:
  b = 20;
  log_dens= (theta>=(-2*log(b)))*(-log(b)-theta/2-log(2))+
  (theta<(-2*log(b)))*(-Inf); return(log_dens);"
#
#sigma_alpha:
sigma.unif.prior.random.eff = "expression:
b = 20;
log_dens = (theta>=(-2*log(b)))*(-log(b)-theta/2-log(2)) +
(theta<(-2*log(b)))*(-Inf); return(log_dens);"
#
b=20;
prec.prior <- list(prec=list(prior = sigma.unif.prior,
initial = -2*log(b)+1, fixed = FALSE))
prec.prior.random.eff <- list(prec=list(
prior = sigma.unif.prior.random.eff, initial = -2*log(b)+1, fixed = FALSE))
```

Hierarchical Models

From JAGS to INLA: Hierarchical Model

- The random effects are added as `f(county,model="iid",hyper=prec.prior.random.eff)`, where "iid" defines a Gaussian random effect.

```

#
#prec.prior.random.eff encodes the prior for the random effect
#hyperparameter theta=log(tau.alpha)
#
#prec.prior encodes the prior on the likelihood precision tau.obs
#
#We set a Gaussian prior with mean 0 and precision 0.01 on the intercept
#(mu_alpha) and the regression coefficient (beta) using prior.beta
prior.beta <- list(mean.intercept = 0, prec.intercept = 0.01,
                     mean = 0, prec = 0.01)
#
m.radon.hierarchical=inla(logradon~1+floor+f(county,model="iid",hyper=
prec.prior.random.eff),data=radon.df, family="gaussian",
control.family=list(hyper=prec.prior),
control.fixed=prior.beta,control.compute=list(config=TRUE) )

```

Hierarchical Models

From JAGS to INLA: Hierarchical Model

→ Now look at the summary of the posteriors:

```
summary(m.radon.hierarchical)
```

```
##           mean      sd 0.025quant 0.5quant 0.975quant    mode
## (Intercept) 1.462 0.053       1.359     1.461      1.566 1.460
## floor       -0.693 0.071      -0.832     -0.693     -0.554 -0.693
## Prec.obs.    1.75  0.085       1.59      1.75      1.92   1.75
## Prec.alpha   9.46  2.699       5.31      9.08     15.82  8.35
```

→ The summary statistics are essentially identical to what we got from JAGS once the precision for the observations (Prec.obs.) and the county (Prec.alpha) are back-transformed into standard deviations.

Hierarchical Models

From JAGS to INLA: Hierarchical Model

```
# sigma obs
marginal.tau=m.radon.hierarchical$
  marginals.hyperpar$`Precision for the Gaussian observations`
#We could also obtain the same marginal by
#marginal.tau=m.radon.hierarchical$marginals.hyperpar[[1]]
marginal.sigma <- inla.tmarginal(function(tau) tau^(-1/2), marginal.tau)
inla.zmarginal(marginal.sigma)
#Sigma.obs Mean 0.756436, 95%CI [0.721341,0.793192]
#
#sigma.alpha (county)
marginal.tau2=m.radon.hierarchical$
  marginals.hyperpar$`Precision for county`
#We could also obtain the same marginal by
#marginal.tau2=m.radon.hierarchical$marginals.hyperpar[[2]]
marginal.sigma.alpha<-inla.tmarginal(function(tau) tau^(-1/2), marginal.tau
inla.zmarginal(marginal.sigma.alpha)
#Sigma.alpha Mean 0.334687, 95%CI [0.251692,0.432703]
```

The R code containing the analyses in Lecture 5 using JAGS and INLA is available on Learn.

Hierarchical Models

Further readings on Hierarchical Models

- ↪ Kruschke JK. Doing Bayesian Data Analysis, second edition. Academic Press, 2015. Chapter 9 (the introduction and sections 9.2, 9.2.2, 9.2.3, 9.2.4, 9.3 and 9.5)
- ↪ Reich BJ, Ghosh SK. Bayesian Statistical Methods. CRC press, 2019. Chapter 6, Section 6.1 and examples 6.3 and 6.4

Bayesian Data Analysis

Daniel Paulin & Nicolò Margaritella

University of Edinburgh



Semester 2, 2020/2021

With thanks to Jonathan Gair, Rubén Amorós-Salvador, Ken Newman, Vanda Inácio and
Natalia Bochkina for much of the material

Outline

- 1 Advice on Performing Analyses and Writing Reports
- 2 Hierarchical GLMs
- 3 Posterior Predictive Distributions
- 4 Model Selection
- 5 Spatial and Temporal Modelling in R-INLA

Section 1

Advice on Performing Analyses and Writing Reports

Hierarchical GLMs

Advice on Bayesian Analysis

Here are some general guidelines on carrying out Bayesian analysis (that don't always have to be followed!)

- ① Be able to clearly and succinctly **state your objective**: What is it that you are trying to do? E.g., I want to see if water temperature or fish size affects survival rates.
- ② Carry out **exploratory data analysis before** any model fitting. Informative plots (histograms, boxplots, and scatterplots) and numerical summaries.
- ③ Fit simple models, examine model **performance** (goodness of fit, predictive performance) and how **sensible parameter estimates** are (E.g., I expected survival to go down as the water got warmer but the model predicts that survival is going up—that does not make sense.). Then fit more complex models as needed or desired.
- ④ When using MCMC: always carry out **convergence checks**.
- ⑤ Examine the effect of your **priors**: try different ones.
- ⑥ Examine **predictive performance** of your final model.
- ⑦ **Honestly report** uncertainty, assumptions, concerns.

Hierarchical GLMs

Advice on Writing Reports of a Data Analysis

When you have analyzed a data set and are producing a written report of your analysis, some of the following remarks may be helpful. They are not absolute rules, but are often useful.

- 1 If a Table or Figure is included in the write-up, **label** it with a number, include a **caption**, and make **reference** to it in the writeup.

Label & caption: Figure 1. Scatterplot of the estimated number of gray whales against the sea surface temperature (years 1990-2020).

In the text: As shown in Figure 1, there is an apparent slightly positive relationship between sea surface temperature and the estimated number of gray whales.

Hierarchical GLMs

Advice on Writing Reports of a Data Analysis

- 2 When referring to a coefficient in a model, try to put its **numerical meaning into the context** of the data.
 - In the model predicting estimates of whale abundance as a function of temperature, the posterior distribution for the temperature coefficient is well above zero suggesting a relatively strong positive relationship between temperature and abundance. In particular, the model estimates that, given an increase of 10°C, the mean abundance of whales is multiplied by 1.3 (increase of 30%).

Hierarchical GLMs

Advice on Writing Reports of a Data Analysis

- 3 When writing a report do not expect the reader to have to figure out **what is important** and what is not. Tell them what you've learned.
 - Of the five environmental covariates that we examined, only water temperature at time of release appeared to have any relationship with salmon return rates. While others have found size at release important (reference ...), we did not find evidence for its importance in this study.

Hierarchical GLMs

Advice on Writing Reports of a Data Analysis

- ④ Briefly report **the technical details of the inference**, including type of algorithm, software and libraries, number of iterations and convergency checks. If computational time is to be discussed, include CPU, operative system, etc. This usually can be reported once for all the analyses.
- Inference for all the models was carried out using Markov Chain Monte Carlo (MCMC) methods using the software packages JAGS and R and the library `rjags` package.
 - Three MCMC chains of 10 000 simulations each were obtained for each of the models after a burn-in of 5 000 iterations except for Model 2 which required 15 000 burn-in iterations.
 - Convergency of the MCMC simulation methods was assessed by direct inspection of the mixing of the chains and by the Gelman-Rubin-Brooks diagnostic (lower than 1.1 for all the traced parameters). The number of effective iterations was over 5 000 for all the parameters and the ratio between their standard deviation and the MCMC error was always greater than 20.

Hierarchical GLMs

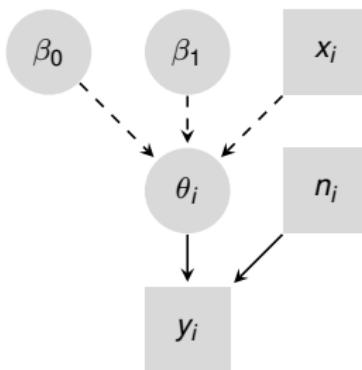
Advice on Writing Reports of a Data Analysis

- 5 Write the **mathematical expression of the models** (usually in terms of conditional distributions). Optionally, you can include a directed graph for complex models¹.
- The Binomial full model for the Beetles and its directed acyclic graph (DAG) representation:

$$y_i \mid \theta_i, \mathbf{x}_i \sim \text{Bin}(\theta_i, n_i), \quad i \in \{1, \dots, I\}$$

$$\text{logit}(\theta_i) = \beta_0 + \beta_1 x_i$$

$$\beta_j \sim N(0, \sigma_{\beta_j}^2), \quad j \in \{0, 1\}$$



We have chosen $\sigma_{\beta_j} = 100$ to reflect our lack of knowledge on the parameters β_j .

¹Check file `DAG.tex` on Learn for a `LATEX` example.

Section 2

Hierarchical GLMs

Hierarchical GLMs

Review

Recall the General Structure of a Hierarchical Model: two key ideas are

- ↪ Multiple levels in the model, a hierarchy, defined in terms of conditional probability distributions.
- ↪ Some parameter(s) different for each group of individuals of a certain partition, but with a common distribution (random effect).

A usual hierarchical model has:

- ↪ **A distribution for the observations** $y_{j,i}$, conditional on random parameters θ_j and covariates $x_{j,i}$:

$$y_{j,i} | \theta_j, x_{j,i} \sim \text{Distribution } (\theta_j, x_{j,i}) \quad j = 1, \dots, J, \quad i = 1, \dots, n_j \quad (1)$$

- ↪ **The distribution for the random parameters** θ_j with unknown parameter(s) ψ :

$$\theta_j | \psi \sim \text{Distribution } (\psi), \quad j = 1, \dots, J \quad (2)$$

- ↪ Then a **(hyper-)prior distribution for the priors** with known hyperparameter(s) η :

$$\psi | \eta \sim \text{Distribution } (\eta) \quad (3)$$

Hierarchical GLMs

Introduction

Random effects can be included in generalised linear models, in what can be called hierarchical GLMs.

- ↪ Allows the consideration of both fixed and random effects for non-Gaussian response variables.
- ↪ The parameters of the random effects can be more interpretable as they are modeling the expectation of the data.
- ↪ We can include extra variance for those distributions where the variance is determined by the expected value (Binomial, Poisson, Exponential...).

Hierarchical GLMs

Hatcheries example

y Number of recovered salmons on each hatchery

n Number of previously released salmons on each hatchery

alength Average length of the released salmons on each hatchery

```
load("salmons.Rdata")
```

```
salmons
```

```
##      y      n   alength
## 1    8    86    14.1
## 2   12    83    15.9
## 3   10    71    15.0
## 4    8    60    16.2
## 5   16    96    17.3
## 6   14    85    21.6
## 7   10    58    15.1
## 8   24    95    16.8
## 9   12    63    17.4
## 10  17   100    22.0
```

Hierarchical GLMs

No covariates (Hatcheries example)

We can model more interpretable random effects (and avoid colinearity problems).

$$Y_j \sim \text{Bin}(\eta_j, p_j)$$

From Lecture 5:

$$p_j \sim \text{Beta}(\alpha, \beta)$$

$$\alpha \sim \text{Lognorm}(\ln(8.1), 0.7)$$

$$\beta \sim \text{Lognorm}(\ln(21.3), 0.7)$$

New!

$$\text{logit}(p_j) = \nu_j$$

$$\nu_j \sim N(\mu_\nu, \sigma_\nu^2)$$

$$\mu_\nu \sim N(0, 10^2)$$

$$\sigma_\nu \sim \text{Unif}(0, 10)$$

Hierarchical GLMs

No covariates (Hatcheries example)

↪ Model Block

```
salmonB.model <- "model{
  for(j in 1:J) {
    y[j] ~ dbin(p[j],n[j])
    logit(p[j]) <- nu[j]
    nu[j] ~ dnorm(mu.nu,tau.nu)
  }

  # Priors
  mu.nu ~ dnorm(0, tau.mu.nu)
  tau.nu <- pow(sigma.nu, -2)
  sigma.nu ~ dunif(0, UB.sigma.nu)

  # Tracing the expected probabilities
  E.surv <- ilogit(mu.nu)
} "
```

Hierarchical GLMs

No covariates (Hatcheries example)

↪ Data and Inits

```
salmonB.data <- list(J=10, n=salmons$n, y=salmons$y,
                      tau.mu.nu=0.01, UB.sigma.nu=10)
salmonB.inits <- function() { list(mu.nu=rnorm(1, 0, 5),
                                     sigma.nu=runif(0,5)) }
```

↪ Call to JAGS:

```
salmonB.res.A <- jags.model(file=textConnection(salmonB.model),
                               data=salmonB.data, inits=salmonB.inits,
                               n.chains=3, quiet = TRUE)
update(salmonB.res.A, n.iter=5000)
salmonB.res.B <- coda.samples(salmonB.res.A,
                               variable.names=c("mu.nu", "sigma.nu", "E.surv", "p"),
                               n.iter=50000)
mcmcplots::mcmcplot(salmonB.res.B,
                      parms = c("mu.nu", "sigma.nu", "E.surv"))
```

Hierarchical GLMs

No covariates (Hatcheries example)

→ Summary statistics

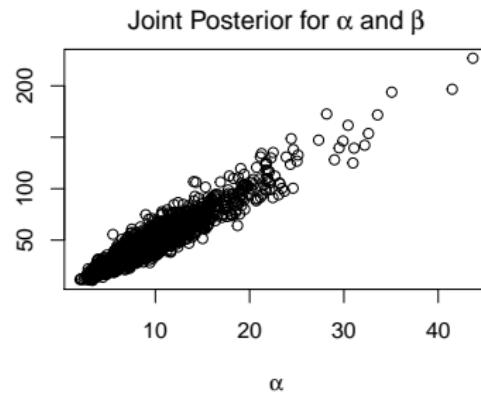
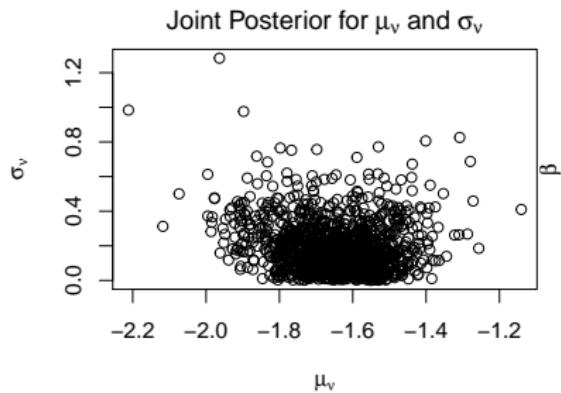
```
summary(salmonB.res.B)
# ...
#           Mean        SD   Naive SE Time-series SE
# mu.nu    -1.6446  0.12524 3.234e-04      0.0011650
# sigma.nu  0.2029  0.14277 3.686e-04      0.0022143
# E.surv    0.1626  0.01695 4.377e-05      0.0001584
# ...
```

```
summary(salmon.res.B)
# ...
#           Mean        SD   Naive SE Time-series SE
# alpha     9.5578  4.38067 8.761e-03      6.834e-02
# beta      47.8301 22.34306 4.469e-02      3.489e-01
# expect.survival 0.1682  0.02141 4.281e-05      7.209e-05
# ...
```

Hierarchical GLMs

No covariates (Hatcheries example)

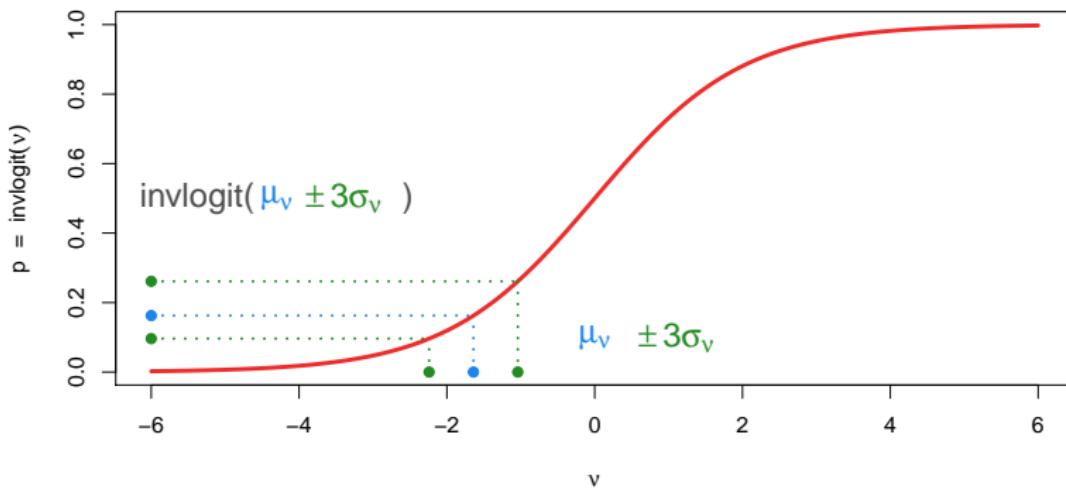
↪ The parameters μ_p and σ_p are much less correlated than α and β .



Hierarchical GLMs

No covariates (Hatcheries example)

↪ The parameters are more interpretable



Hierarchical GLMs

No covariates (Hatcheries example)

- ↪ The model can be rewritten so that one of the terms expresses an extra variance ²,
- ↪ by expressing the random effect ν_j as the sum of an overall mean $\mu = \mu_\nu$ plus a white noise term ϵ_j which follows a Gaussian distribution with zero mean and variance $\sigma_\varepsilon^2 = \sigma_\nu^2$.

$$Y_j \sim \text{Bin}(n_j, p_j)$$

$$\text{logit}(p_j) = \nu_j$$

$$\nu_j \sim N(\mu_\nu, \sigma_\nu^2)$$

$$\mu_\nu \sim N(0, 10^2)$$

$$\sigma_\nu \sim \text{Unif}(0, 10)$$

$$\text{logit}(p_j) = \mu + \varepsilon_j$$

$$\varepsilon_j \sim N(0, \sigma_\varepsilon^2)$$

$$\mu \sim N(0, 10^2)$$

$$\sigma_\varepsilon \sim \text{Unif}(0, 10)$$

²If there were no extra variance term, the variance would be determined by the mean

$$Y_j \sim \text{Bin}(p_j, n_j) \rightarrow E(Y_j) = n_j p_j, V(p_j) = n_j p_j (1 - p_j)$$

Hierarchical GLMs

No covariates (Hatcheries example)

↪ The code (equivalent to the previous one but for names)

```
salmonB2.model <- "model{  
  for(j in 1:J) {  
    y[j] ~ dbin(p[j],n[j])  
    logit(p[j])  <-  mu + epsilon[j]  
    epsilon[j] ~ dnorm(0 ,tau.epsilon)  
  }  
  
  # Priors  
  mu ~ dnorm(0, tau.mu)  
  tau.epsilon  <- pow(sigma.epsilon, -2)  
  sigma.epsilon ~ dunif(0, UB.sigma.epsilon)  
  
  # Tracing the expected probabilities  
  E.surv <- ilogit(mu)  
}  
"
```

Hierarchical GLMs

Including covariates (Hatcheries example)

Let us include a covariate in the model.

$$\begin{aligned}
 Y_j &\sim \text{Bin}(n_j, p_j) \\
 \text{logit}(p_j) &= \mu + \beta_1(X_j - \bar{X}) + \varepsilon_j \\
 \mu &\sim N(0, 10^2) \\
 \beta_1 &\sim N(0, 10^2) \\
 \varepsilon_j &\sim N(0, \sigma_\varepsilon^2) \\
 \sigma_\varepsilon &\sim \text{Unif}(0, 10)
 \end{aligned}$$

- ↪ Traditionally the intercept of a regressor is called β_0 .
- ↪ In this case, the notation $\mu = \beta_0$ is adequate because we have centered our covariates and therefore μ is the expected value of the regressor for the expected value of X_j .

Hierarchical GLMs

Including covariates (Hatcheries example)

↪ The code - including centred covariates

```
salmonC.model <- "model{
  for(j in 1:J) {
    y[j] ~ dbin(p[j],n[j])
    logit(p[j]) <- mu + betal*(X[j] - mean(X[])) + epsilon[j]
    epsilon[j] ~ dnorm(0 ,tau.epsilon)
  }

  # Priors
  mu ~ dnorm(0, tau.mu)
  betal ~ dnorm(0, tau.betal)
  tau.epsilon <- pow(sigma.epsilon, -2)
  sigma.epsilon ~ dunif(0, UB.sigma.epsilon)

  # Tracing the expected probabilities
  E.surv <- ilogit(mu)
}
```

Hierarchical GLMs

Including covariates (Hatcheries example)

↪ Data and Inits

```
salmonC.data <- list(J=10, n=salmons$n, y=salmons$y,
                      X=salmons$alength, tau.mu=0.01,
                      tau.betal=0.01, UB.sigma.epsilon=10)
salmonC.inits <- function() { list(mu=rnorm(1, 0, 2),
                                      betal=rnorm(1, 0, 2),
                                      sigma.epsilon=runif(0,2)) }
```

↪ Call to JAGS:

```
salmonC.res.A <- jags.model(file=textConnection(salmonC.model),
                               data=salmonC.data, inits=salmonC.inits,
                               n.chains=3, quiet = TRUE)
update(salmonC.res.A, n.iter=5000)
salmonC.res.B <- coda.samples(salmonC.res.A,
                               variable.names=c("mu", "betal", "sigma.epsilon", "E.surv", "p")
                               n.iter=50000)
mcmcplots::mcmcplot(salmonC.res.B,
                     parms = c("mu", "betal", "sigma.epsilon", "E.surv"))
```



Hierarchical GLMs

No covariates (Hatcheries example)

↪ Summary statistics

```
summary(salmonB.res.B)
# ...
#           Mean        SD   Naive SE Time-series SE
# mu.nu    -1.6446  0.12524 3.234e-04      0.0011650
# sigma.nu  0.2029  0.14277 3.686e-04      0.0022143
# E.surv    0.1626  0.01695 4.377e-05      0.0001584
# ...
```

```
summary(salmonC.res.B)
# ...
#           Mean        SD   Naive SE Time-series SE
# mu       -1.65582  0.12985 3.353e-04      8.279e-04
# betal     0.03521  0.04989 1.288e-04      3.356e-04
# sigma.epsilon 0.008831  0.094939 0.19137  0.30305  0.6047
# E.surv    0.16109  0.01745 4.504e-05      1.101e-04
# ...
```

Section 3

Posterior Predictive Distributions

Hierarchical GLMs

Posterior Predictive Distributions

- ↪ The posterior predictive distribution can be obtained by the law of total probability:

$$\begin{aligned} p(z|\mathbf{Y}) &= \int_{\Theta} p(z|\theta, \mathbf{Y})p(\theta|\mathbf{Y})d\theta \\ &= \int_{\Theta} p(z|\theta)p(\theta|\mathbf{Y})d\theta \end{aligned}$$

- ↪ And in a hierarchical model, also by the same law:

$$\begin{aligned} p(\theta_{J+1}|\mathbf{Y}) &= \int_{\Psi} p(\theta_{J+1}|\psi, \mathbf{Y})p(\psi|\mathbf{Y})d\psi \\ &= \int_{\Psi} p(\theta_{J+1}|\psi)p(\psi|\mathbf{Y})d\psi \end{aligned}$$

- ↪ For a variable (parameter) ϕ the samples obtained by MCMC $\phi^{(s)}$ are samples of $p(\phi|\mathbf{Y})$.
- ↪ Samples of a distribution obtained by an integral of the type

$$p(\gamma|\mathbf{Y}) = \int_{\Phi} p(\gamma|\phi)p(\phi|\mathbf{Y})d\phi$$

can be obtained by simply substituting each $\phi^{(s)}$ in $p(\gamma|\phi)$ and sampling.

- ↪ Therefore, $\gamma^{(s)} = \text{sample}(p(\gamma|\phi^{(s)}))$ is a sample of $p(\gamma|\mathbf{Y})$.

Hierarchical GLMs

Posterior Predictive Distributions

- ↪ The posterior predictive distribution for an observation of group j , considering the random effect for that group, is

$$p(z_j | \mathbf{Y}) = \int_{\Theta_j} p(z | \theta_j) p(\theta_j | \mathbf{Y}) d\theta_j$$

- ↪ As we have MCMC samples $\theta_j^{(s)}$ of $p(\theta_j | \mathbf{Y})$, we only have to draw a sample from $p(z | \theta_j^{(s)})$ for each $s \in \{1, \dots, S\}$.
- ↪ But the posterior predictive distribution for an observation from a new group (or if we want to ignore the group) is

$$p(z_{J+1} | \mathbf{Y}) = \int_{\Theta_{J+1}} p(z_{J+1} | \theta_{J+1}) p(\theta_{J+1} | \mathbf{Y}) d\theta_{J+1}$$

- ↪ We do not have MCMC samples of $p(\theta_{J+1} | \mathbf{Y})$, so we obtain them from

$$p(\theta_{J+1} | \mathbf{Y}) = \int_{\Psi} p(\theta_{J+1} | \psi) p(\psi | \mathbf{Y}) d\psi$$

by drawing a sample $\theta_{J+1}^{(s)}$ from $p(\theta_{J+1} | \psi^{(s)})$ and then a sample $z_{J+1}^{(s)}$ from $p(z_{J+1} | \theta_{J+1}^{(s)})$



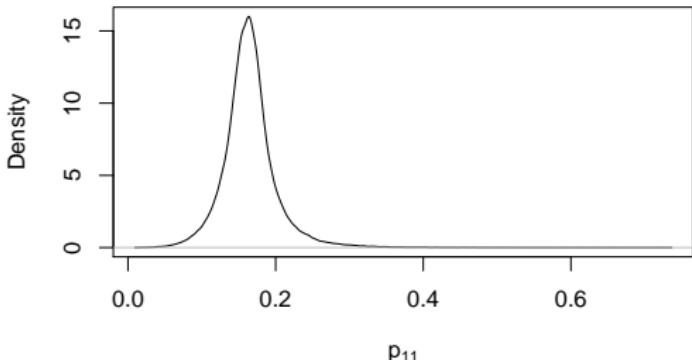
Hierarchical GLMs

Posterior Predictive Distributions

- To obtain samples from the posterior of a random effect for a new group $J + 1$, we have to draw a sample from $p(\theta|\psi^{(s)})$ for each $s \in \{1, \dots, S\}$. Draws from $p(p_{11}|\mathbf{y})$:

```
# Samples in data.frame form
salB.out <- do.call(rbind.data.frame, salmonB.res.B)
S <- dim(salB.out)[1] # Num. MCMC samples

null <- rnorm(n = S, mean = salB.out$mu.nu, sd = salB.out$sigma.nu)
p11 <- 1 / (1 + exp(-null)) # inverse-logit
```



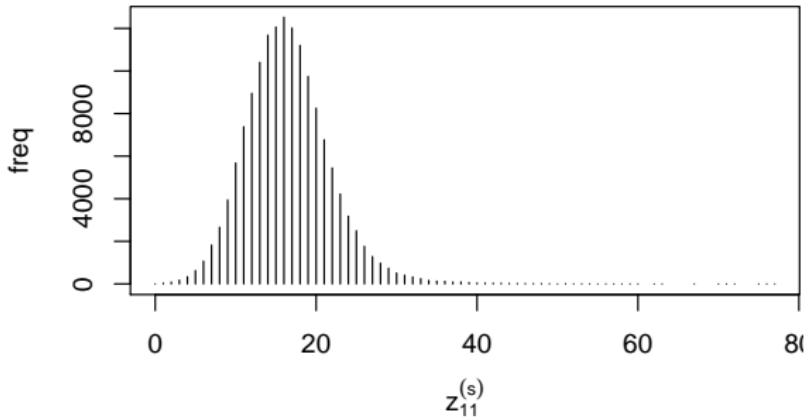
$$\begin{aligned}
 Y_j &\sim \text{Bin}(n_j, p_j) \\
 \text{logit}(p_j) &= \nu_j \\
 \nu_j &\sim N(\mu_\nu, \sigma_\nu^2) \\
 \mu_\nu &\sim N(0, 10^2) \\
 \sigma_\nu &\sim \text{Unif}(0, 10)
 \end{aligned}$$

Hierarchical GLMs

Posterior Predictive Distributions

- Draws from $p(z_{11} | \mathbf{Y})$ (posterior predictive number of recoveries for a new hatchery if it releases 100 juvenile salmons)

```
null <- rnorm(n = S, mean = salB.out$mu.nu ,sd = salB.out$sigma.nu)
p11 <- 1/(1+ exp(-nu11) ) # inverse-logit
z11 <- rbinom(n = S, size = 100, prob = p11)
```



$$Y_j \sim \text{Bin}(n_j, p_j)$$

$$\text{logit}(p_j) = \nu_j$$

$$\nu_j \sim N(\mu_\nu, \sigma_\nu^2)$$

$$\mu_\nu \sim N(0, 10^2)$$

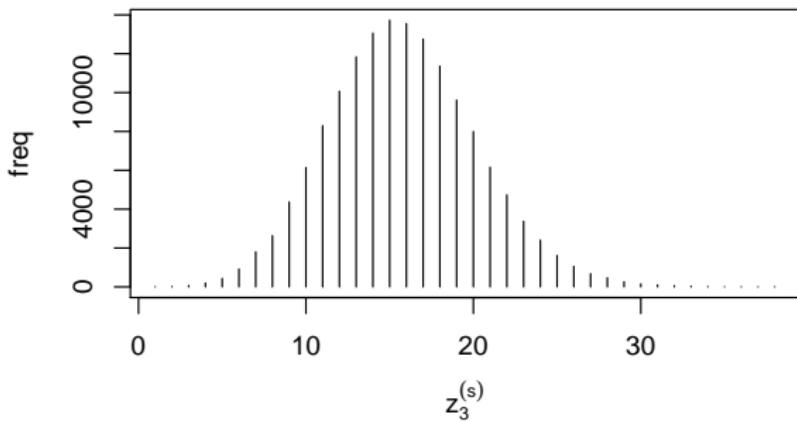
$$\sigma_\nu \sim \text{Unif}(0, 10)$$

Hierarchical GLMs

Posterior Predictive Distributions

- Draws from $p(z_3 | \mathbf{Y})$ (posterior predictive number of recoveries for the specific hatchery 3 if we releases 100 juvenile salmons, considering the particular value of the random effect)

```
z3 <- rbinom(n = S, size = 100, prob = salB.out$p[3])
```



$$Y_j \sim \text{Bin}(n_j, p_j)$$

$$\text{logit}(p_j) = \nu_j$$

$$\nu_j \sim N(\mu_\nu, \sigma_\nu^2)$$

$$\mu_\nu \sim N(0, 10^2)$$

$$\sigma_\nu \sim \text{Unif}(0, 10)$$

Hierarchical GLMs

Posterior Predictive Distributions

Some further remarks:

- ↪ If there are **covariates**, they **should be taken into account**. Are we predicting for a certain value of X ? For the mean value?
- ↪ Posterior and posterior predictive distributions can also be done **in JAGS** (as we have seen before). Be cautious when you predict for $p(z_j | \mathbf{Y})$. The prediction is different if you consider the particular random effect for j or not.

Section 4

Model Selection

Hierarchical GLMs

Model Selection

Bayesian Model Selection.

- ↪ Given two or more competing models how should one decide which model is better?
- ↪ Which model predicts future observations better is one approach.
- ↪ **Cross-validation** is often a good approach: leave out a portion of the data (test set), and fit competing models to the rest of the data (training set). Then see which model predicts the test set best.
- ↪ Which model has more sensible coefficients is another.
- ↪ Special priors for the parameters of the regression can also be used for variable selection (e.g. Laplace priors –**Bayesian Lasso**– or Spike-and-slab priors).
- ↪ Other approaches are based on setting a prior distribution to each model and compare the Bayes Factor

$$\frac{Pr(\mathbf{Y}|M_1)}{Pr(\mathbf{Y}|M_2)} = \frac{Pr(M_1|\mathbf{Y})Pr(M_1)}{Pr(M_2|\mathbf{Y})Pr(M_2)}$$

Hierarchical GLMs

Model Selection

- ↪ Large datasets and complex models hinder the use of cross validation and Bayes factor. Model-fit criteria represent a good alternative.

Deviance Information Criterion (DIC).

- ↪ DIC provides a (1) measure of the error in the fit with (2) a penalty for the complexity of the model.
- ↪ The lower DIC, the better the model.
- ↪ (1) The measure of error is called the “Deviance” and it is 2 times the negative log likelihood:

$$\text{Deviance: } D(\mathbf{Y}|\boldsymbol{\theta}) = -2 \ln f(\mathbf{Y}|\boldsymbol{\theta}),$$

The smaller the deviance, “generally speaking”, the better the model fits the data.

- ↪ The deviance $D(\mathbf{Y}|\boldsymbol{\theta})$ is a random variable in the Bayesian setting (as $\boldsymbol{\theta}$ is random) with its own posterior distribution

Hierarchical GLMs

Model Selection

Deviance Information Criterion (DIC).

- ↪ (2) Model complexity is summarized by the *Effective Number of Parameters*, pD .
- ↪ pD is similar to a count of the unknown parameters in a model where highly dependent parameters and those with strongly informative priors count for less than one. Particularly in hierarchical models, the effective number of parameters does not equal the number of parameters.

$$pD = \widehat{D(\mathbf{Y}, \theta)} - D(\mathbf{Y}, \bar{\theta})$$

where $\widehat{D(\mathbf{Y}, \theta)}$ is the estimate of the posterior mean of the deviance which can be obtained from

$$\widehat{D(\mathbf{Y}, \theta)} \approx \frac{1}{S} \sum_{s=1}^S D(\mathbf{y}, \theta^{(s)})$$

where $\theta^{(s)}$ is the s th sample from the MCMC chain (after burn-in).

- ↪ $D(\mathbf{Y}, \bar{\theta})$ is the deviance evaluated at the posterior mean of θ .

Hierarchical GLMs

Model Selection

↪ DIC is then the sum of (1) and (2):

$$DIC = \widehat{D(\mathbf{y}, \theta)} + p_D$$

- ↪ The actual value of the DIC is hard to interpret; however, it can be employed to rank models: models with small DIC are simple (small p_D) and fit well the data (small Deviance).
- ↪ Loose rule of thumb: Difference of 5= substantial; 10 = definitive.
- ↪ Note that p_D can be outside the range $[0; p]$ in some pathological cases, typically where the posterior mean of θ is not a good summary of the posterior (e.g. mixture models with multimodal priors and posteriors).

Hierarchical GLMs

Model Selection

Getting DIC from JAGS (our Salmon example: model with vs. without covariate):

```
salmonB.DIC <- dic.samples(model=salmonB.res.A, n.iter=10000, type="pD")
salmonC.DIC <- dic.samples(model=salmonB.res.A, n.iter=10000, type="pD")
```

```
salmonB.DIC # Model without average length
```

```
## Mean deviance: 50.73
## penalty 4.113
## Penalized deviance: 54.84
```

```
salmonC.DIC # Model with average length
```

```
## Mean deviance: 50.69
## penalty 4.036
## Penalized deviance: 54.73
```

Section 5

Spatial and Temporal Modelling in R-INLA

Spatial and Temporal Modelling

Introduction to the dataset

- In this section, we are going to show you how random effects can be used for spatial and temporal modelling.
- R-INLA will be used to perform the calculations.
- The dataset considered is the apartment price data from Taipei that we used previously in Assignment 1. This dataset contains 414 rows. The response variable is

$$y = \log(\text{house price per unit area}).$$

The following covariates are available:

- **transaction date** (this is stored as a real number between 2012 and 2014, i.e. 2013.5 corresponds to 1 July, 2013)
- **house age** (in years)
- **distance from nearest MRT (metro) station** (in meters)
- **number of convenience stores within walking distance**
- **latitude** (in degrees)
- **longitude** (in degrees)

Spatial and Temporal Modelling

Introduction to the dataset

```
house <- read.csv(file = 'Real_estate.csv')
head(house)

##   No X1.transaction.date X2.house.age X3.distance.to.the.nearest.MRT.station
## 1  1      2012.917        32.0                      84.87882
## 2  2      2012.917        19.5                     306.59470
## 3  3      2013.583        13.3                     561.98450
## 4  4      2013.500        13.3                     561.98450
## 5  5      2012.833        5.0                      390.56840
## 6  6      2012.667        7.1                    2175.03000
##   X4.number.of.convenience.stores X5.latitude X6.longitude
## 1                               10    24.98298    121.5402
## 2                                9    24.98034    121.5395
## 3                                5    24.98746    121.5439
## 4                                5    24.98746    121.5439
## 5                                5    24.97937    121.5425
## 6                                3    24.96305    121.5125
##   Y.house.price.of.unit.area
## 1                          37.9
## 2                          42.2
## 3                          47.3
## 4                          54.8
## 5                          43.1
## 6                          32.1
```

Spatial and Temporal Modelling

Overview of Taipei



Spatial and Temporal Modelling I

INLA model 1: linear model with only fixed effects

- ↪ We start by creating a baseline Bayesian linear regression model with only fixed effects.
- ↪ The first step is to standardize all of the covariates, and create the response variable y as the logarithm of the house price per unit area.
- ↪ After this, we fit a model using the following code,

```
m.I <- inla(y ~ 1 + transaction + age + distance+ stores  
+longitude+latitude,  
family = "Gaussian",  
data = house,  
control.compute = list(cpo=T,dic = T))
```

Spatial and Temporal Modelling I

INLA model 1: linear model with only fixed effects

```
summary(m.I)
```

Call:

```
c("inla(formula = y ~ 1 + transaction + age + distance + stores + ",
   longitude + latitude, family = \"Gaussian\", data = house, ",",
   control.compute = list(cpo = T, dic = T))")
```

Time used:

Pre = 3.33, Running = 0.376, Post = 0.294, Total = 3.99

Fixed effects:

	mean	sd	0.025quant	0.5quant	0.975quant	mode	kld
(Intercept)	3.567	0.011	3.545	3.567	3.588	3.567	0
transaction	0.038	0.011	0.017	0.038	0.060	0.038	0
age	-0.079	0.011	-0.101	-0.079	-0.058	-0.079	0
distance	-0.184	0.023	-0.228	-0.184	-0.139	-0.184	0
stores	0.082	0.014	0.054	0.082	0.109	0.082	0
longitude	0.006	0.019	-0.031	0.006	0.042	0.006	0
latitude	0.098	0.014	0.071	0.098	0.126	0.098	0

Spatial and Temporal Modelling II

INLA model 1: linear model with only fixed effects

Model hyperparameters:

	mean	sd	0.025quant	0.5quant
Precision for the Gaussian observations	20.46	1.43	17.76	20.43
			0.975quant	mode

Precision for the Gaussian observations	23.36	20.36
---	-------	-------

Expected number of effective parameters (stdev) : 7.05(0.004)

Number of equivalent replicates : 58.71

Deviance Information Criterion (DIC): -63.57

Deviance Information Criterion (DIC, saturated): 425.59

Effective number of parameters: 8.14

Marginal log-Likelihood: -19.21

CPO and PIT are computed

Posterior marginals for the linear predictor and
the fitted values are computed

Spatial and Temporal Modelling

INLA model 1: linear model with only fixed effects

- We also compute NLSCPO, DIC, and the standard deviation of mean residuals for INLA model 1.

```
## NLSCPO of INLA model 1: -28.70251  
## DIC of INLA model 1: -63.5663  
## Standard deviation of mean residuals for INLA model 1: 0.2199845
```

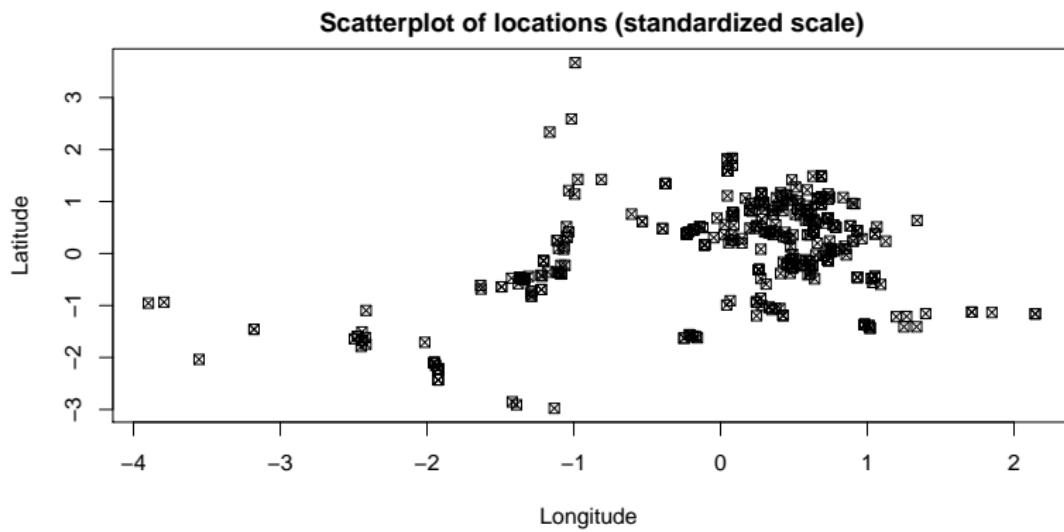
- As the response variable y is in log-scale, 0.22 standard error corresponds to approximately 22% relative error in the estimate of the house price (per unit area).
- This is quite large, and it raises the possibility that some important aspects that influence the house price are not included in the data, or not captured by our simple model.

Spatial and Temporal Modelling

INLA model 2: SPDE spatial random effects

- One obvious candidate for improved modelling is the location (latitude + longitude).

```
plot(house$longitude, house$latitude,  
      main="Scatterplot of locations (standardized scale)",  
      xlab="Longitude ", ylab="Latitude", pch=7)
```



Spatial and Temporal Modelling

INLA model 2: SPDE spatial random effects

- ↪ One way to use the location information is to model it with a smooth Gaussian field random effect.
- ↪ This means that there would a location specific effect on the response (log house price per unit area), but this effect is assumed to be changing relatively smoothly in the location (to make sure that we do not overfit on the noise).
- ↪ Stochastic Partial Differential Equation (SPDE) random effects were introduced by Finn Lindgren et al., "An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach." JRSSB 73.4 (2011): 423-498.
- ↪ They allow the creation of efficient spatial random effect models where the latent variables are chosen according to appropriately chosen meshes.

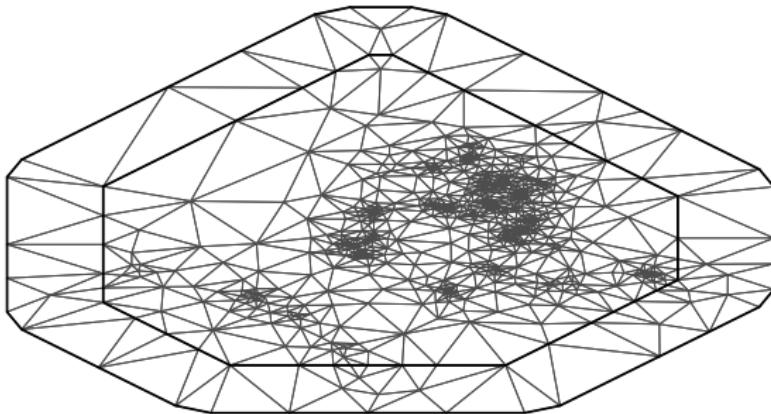
Spatial and Temporal Modelling

INLA model 2: SPDE spatial random effects

- When using spde spatial models in INLA, the first step is to create a mesh.

```
Locations = cbind(house$longitude, house$latitude)
loc.mesh <- inla.mesh.2d(Locations, max.edge = c(10, 20))
plot(loc.mesh)
```

Constrained refined Delaunay triangulation



Spatial and Temporal Modelling

INLA model 2: SPDE spatial random effects

- Once the mesh is ready, we need to create the A, spde and w objects that together define the SPDE random effects model. These can be created using the `inla.spde.make.A`, `inla.spde2.pcmatern` and `inla.spde.make.index` functions.

```
# Creating A matrix (Observation/prediction matrix)
loc.A <- inla.spde.make.A(loc.mesh, loc = Locations)

#Creating Matern SPDE model object with PC prior
loc.spde = inla.spde2.pcmatern(mesh = loc.mesh,
                                prior.range = c(1, 0.5),
                                prior.sigma = c(1, 0.5))

#Generating the SPDE model index vector
loc.w <- inla.spde.make.index('w', n.spde = loc.spde$n.spde)
```

- In the case of SPDE models, the dataframe that we pass along to INLA as the `data` argument changes (i.e. we cannot just use the `house` variable).
- We need to create the `data` argument using the `inla.stack` and `inla.stack.data` functions.

Spatial and Temporal Modelling

INLA model 2: SPDE spatial random effects

```
#First we make the model matrix using the model formula,
#but without response and intercept.
X0 <- model.matrix(~ 0 +transaction+age+distance+ stores, data = house)
X <- as.data.frame(X0) # convert to a data frame.
# Making the stack
N <- nrow(house) #Saving the number of rows in the data
StackHouse <- inla.stack(
  # specify the response variable
  data = list(y = house$y),
  # Vector of Multiplication factors for fixed effects
  A = list(1, 1, loc.A),
  #Specify the fixed and random effects
  effects = list(
    # specify the manual intercept!
    Intercept = rep(1, N),
    # attach the model matrix
    X = X,
    # attach the w
    w = loc.w) )
```

Spatial and Temporal Modelling

INLA model 2: SPDE spatial random effects

- The INLA call is described below.

```
m.I2 <- inla(y ~ 0 + Intercept + transaction + age + distance+ stores  
+f(w, model = loc.spde),  
family = "Gaussian",  
data = inla.stack.data(StackHouse),  
control.compute = list(cpo=T,dic = T),  
control.predictor = list(A = inla.stack.A(StackHouse) ))
```

- `inla.stack.data(StackHouse)` is used to pass along the `data` argument.
- The SPDE random effect is denoted as `f(w, model = loc.spde)`.

```
summary(m.I2)
```

Call:

```
c("inla(formula = y ~ 0 + Intercept + transaction + age + distance + "
  " stores + f(w, model = loc.spde), family = \"Gaussian\", data =
  inla.stack.data(StackHouse), ", " control.compute = list(cpo = T, dic
  T), control.predictor = list(A = inla.stack.A(StackHouse))))" )
```

Time used:

Pre = 3.41, Running = 1.53, Post = 0.34, Total = 5.28

Fixed effects:

	mean	sd	0.025quant	0.5quant	0.975quant	mode	kld
Intercept	3.537	0.085	3.369	3.536	3.711	3.535	0
transaction	0.044	0.008	0.027	0.044	0.060	0.044	0
age	-0.058	0.012	-0.081	-0.058	-0.035	-0.058	0
distance	-0.175	0.045	-0.267	-0.175	-0.086	-0.174	0
stores	0.068	0.035	-0.001	0.068	0.139	0.068	0

Random effects:

Name	Model
w	SPDE2 model

Model hyperparameters:

	mean	sd	0.025quant	0.5quant
Precision for the Gaussian observations	44.992	5.486	34.792	44.824
Range for w	0.878	0.255	0.508	0.833
Stdev for w	0.274	0.036	0.210	0.272
	0.975quant	mode		
Precision for the Gaussian observations	56.337	44.650		
Range for w	1.499	0.748		
Stdev for w	0.351	0.268		

Expected number of effective parameters (stdev) : 99.50 (16.18)
 Number of equivalent replicates : 4.16

Deviance Information Criterion (DIC): -296.27
 Deviance Information Criterion (DIC, saturated): 522.91
 Effective number of parameters: 102.01

Marginal log-Likelihood: 50.27
 CPO and PIT are computed

Posterior marginals for the linear predictor and
 the fitted values are computed

Spatial and Temporal Modelling

INLA model 2: SPDE spatial random effects

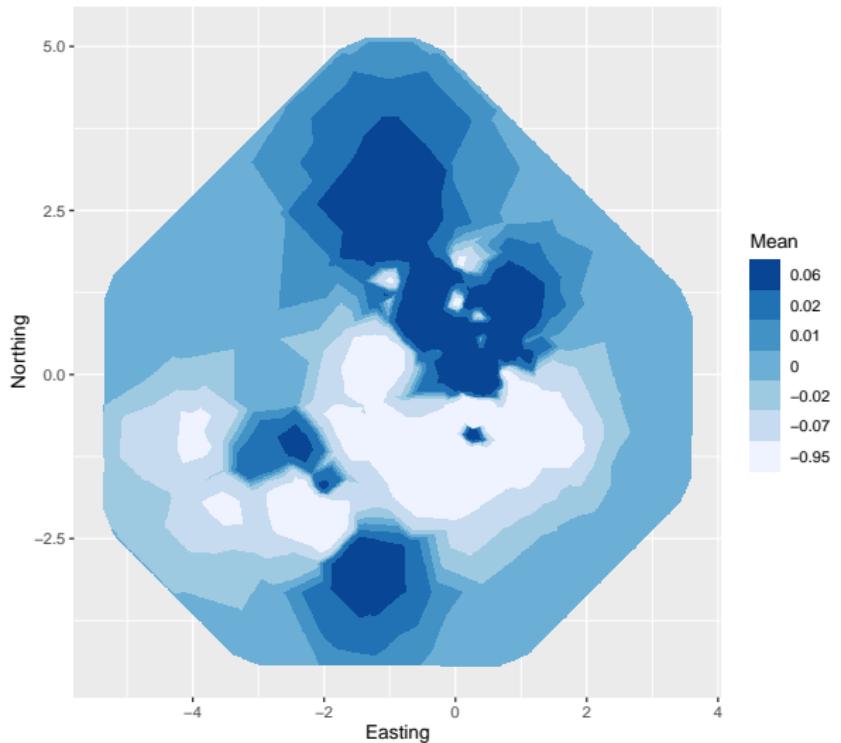
- We also compute NLSCPO, DIC, and the standard deviation of mean residuals for INLA model 2.

```
## NLSCPO of INLA model 2: -117.01  
## DIC of INLA model 2: -307.2324  
## Standard deviation of mean residuals for INLA model 2: 0.1247535
```

- Our second model is significantly better than the first one based on all 3 criteria. Using spatial effects have significantly increased the accuracy of the model.
- The figure on the next slide shows the mean of the spatial effects (plotted using `ggField`, see the R code Lecture 6 on Learn). We can see that some areas have higher house prices than others.

Spatial and Temporal Modelling

INLA model 2: SPDE spatial random effects



Spatial and Temporal Modelling

INLA model 3: SPDE spatial and temporal random effects

- ↪ In model 2, we have considered spatial random effects for the location parameter.
- ↪ In model 3, we consider random effects for the date and house age (temporal parameters) and distance to nearest MRT station (spatial parameter).
- ↪ All of these are one dimensional. We are going to create one dimensional meshes for them, and then use 3 additional SPDE random effects.
- ↪ The specification of these in R-INLA is quite similar to the previous spatial random effect.

Spatial and Temporal Modelling

INLA model 3: SPDE spatial and temporal random effects

```
Locations = cbind(house$longitude, house$latitude)
loc.mesh <- inla.mesh.2d(Locations, max.edge = c(10, 20))

loc.A <- inla.spde.make.A(loc.mesh, loc = Locations)
loc.spde = inla.spde2.pcmatern(mesh = loc.mesh,
                                prior.range = c(1, 0.5),
                                prior.sigma = c(1, 0.5))
loc.w <- inla.spde.make.index('w', n.spde = loc.spde$n.spde)

d.mesh <- inla.mesh.1d(house$distance) #Create a 1D mesh
d.A <- inla.spde.make.A(d.mesh, loc = house$distance)
d.spde = inla.spde2.pcmatern(mesh = d.mesh,
                                prior.range = c(1, 0.5),
                                prior.sigma = c(1, 0.5))
d.w <- inla.spde.make.index('d.w', n.spde = d.spde$n.spde)
```

Spatial and Temporal Modelling

INLA model 3: SPDE spatial and temporal random effects

```
a.mesh <- inla.mesh.1d(house$age)
a.A <- inla.spde.make.A(a.mesh, loc = house$age)
a.spde = inla.spde2.pcmatern(mesh = a.mesh,
    prior.range = c(1, 0.5),
    prior.sigma = c(1, 0.5))
a.w <- inla.spde.make.index('a.w', n.spde = a.spde$n.spde)

t.mesh <- inla.mesh.1d(seq(min(house$transaction),
    max(house$transaction), length.out=100))
t.A <- inla.spde.make.A(t.mesh, loc = house$transaction)
t.spde = inla.spde2.pcmatern(mesh = t.mesh,
    prior.range = c(1, 0.5),
    prior.sigma = c(1, 0.5))
t.w <- inla.spde.make.index('t.w', n.spde = t.spde$n.spde)
```

Spatial and Temporal Modelling

INLA model 3: SPDE spatial and temporal random effects

```
# Make the model matrix using the model formula without response and intercept.
X0 <- model.matrix(~ 0 +transaction+age+distance+ stores, data = house)
X <- as.data.frame(X0) # convert to a data frame.
# Making the stack #####
N <- nrow(house)
house$ID=1:N
StackHouse <- inla.stack(
  data = list(y = house$y), # specify the response variable
  # Vector of Multiplication factors for fixed and random effects
  A = list(1, 1, a.A, t.A, d.A, loc.A),
  effects = list(
    Intercept = rep(1, N), # specify the manual intercept!
    X = X, # attach the model matrix
    a.w = a.w,
    t.w = t.w,
    d.w = d.w,
    w = loc.w # attach the w
  )
)
```

Spatial and Temporal Modelling

INLA model 3: SPDE spatial and temporal random effects

- ↪ The call to INLA for model 3:

```
m.I3 <- inla(y ~ 0 + Intercept + transaction + age + distance+ stores  
+f(a.w,model=a.spde)+f(t.w,model=t.spde)  
+f(d.w,model=d.spde)+f(w, model = loc.spde),  
family = "Gaussian",  
data = inla.stack.data(StackHouse),  
control.compute = list(cpo=T,dic = T),  
control.predictor = list(A = inla.stack.A(StackHouse)))
```

- ↪ The data parameter is still passed along as `inla.stack.data(StackHouse)`.
- ↪ The 4 SPDE random effects are specified as `f(a.w,model=a.spde)`,
`f(t.w,model=t.spde)`, `f(d.w,model=d.spde)`, `f(w, model = loc.spde)`.

Spatial and Temporal Modelling I

INLA model 3: SPDE spatial and temporal random effects

```
summary(m.I3)
```

Call:

```
c("inla(formula = y ~ 0 + Intercept + transaction + age + distance + "
  " stores + f(a.w, model = a.spde) + f(t.w, model = t.spde) + ", "
  " f(d.w, model = d.spde) + f(w, model = loc.spde), family = \"Gaussian\""
  ", " data = inla.stack.data(StackHouse), control.compute = list(cpo =
T, ", " dic = T), control.predictor = list(A =
  inla.stack.A(StackHouse)))" )
```

Time used:

Pre = 5.85, Running = 16.3, Post = 0.695, Total = 22.8

Fixed effects:

	mean	sd	0.025quant	0.5quant	0.975quant	mode	kld
Intercept	3.630	1.807	-0.643	3.644	7.819	3.654	0.000
transaction	0.051	0.037	-0.022	0.049	0.127	0.046	0.002
age	-0.019	0.095	-0.203	-0.020	0.168	-0.025	0.000
distance	-0.230	0.122	-0.471	-0.230	0.011	-0.230	0.000
stores	0.064	0.038	-0.010	0.063	0.137	0.063	0.000

Spatial and Temporal Modelling II

INLA model 3: SPDE spatial and temporal random effects

Random effects:

Name	Model
a.w	SPDE2 model
t.w	SPDE2 model
d.w	SPDE2 model
w	SPDE2 model

Model hyperparameters:

		mean	sd	0.025quant	0.5quant	0.975quant
Precision for the Gaussian observations	51.341	4.674	42.583	51.212	51.212	51.212
Range for a.w	6.843	3.762	2.390	5.930	5.930	5.930
Stdev for a.w	0.400	0.321	0.106	0.307	0.307	0.307
Range for t.w	28.315	35.514	4.967	17.683	17.683	17.683
Stdev for t.w	0.456	0.486	0.062	0.311	0.311	0.311
Range for d.w	4.754	4.024	1.018	3.597	3.597	3.597
Stdev for d.w	0.083	0.083	0.001	0.051	0.051	0.051

Spatial and Temporal Modelling III

INLA model 3: SPDE spatial and temporal random effects

Range for w	0.624	0.190	0.375	0.582
Stdev for w	0.142	0.052	0.056	0.140
	0.975quant	mode		
Precision for the Gaussian observations	60.913	51.053		
Range for a.w	16.574	4.573		
Stdev for a.w	1.249	0.202		
Range for t.w	116.079	9.516		
Stdev for t.w	1.731	0.156		
Range for d.w	15.421	2.244		
Stdev for d.w	0.284	0.000		
Range for w	1.101	0.503		
Stdev for w	0.233	0.116		

Expected number of effective parameters (stdev) : 124.37 (9.43)
 Number of equivalent replicates : 3.33

Deviance Information Criterion (DIC) -339.23

Spatial and Temporal Modelling IV

INLA model 3: SPDE spatial and temporal random effects

```
Deviance Information Criterion (DIC, saturated) ....: 531.90
Effective number of parameters .....: 122.70
```

Marginal log-Likelihood: 47.33

CPO and PIT are computed

Posterior marginals for the linear predictor and
the fitted values are computed

Spatial and Temporal Modelling

INLA model 3: SPDE spatial and temporal random effects

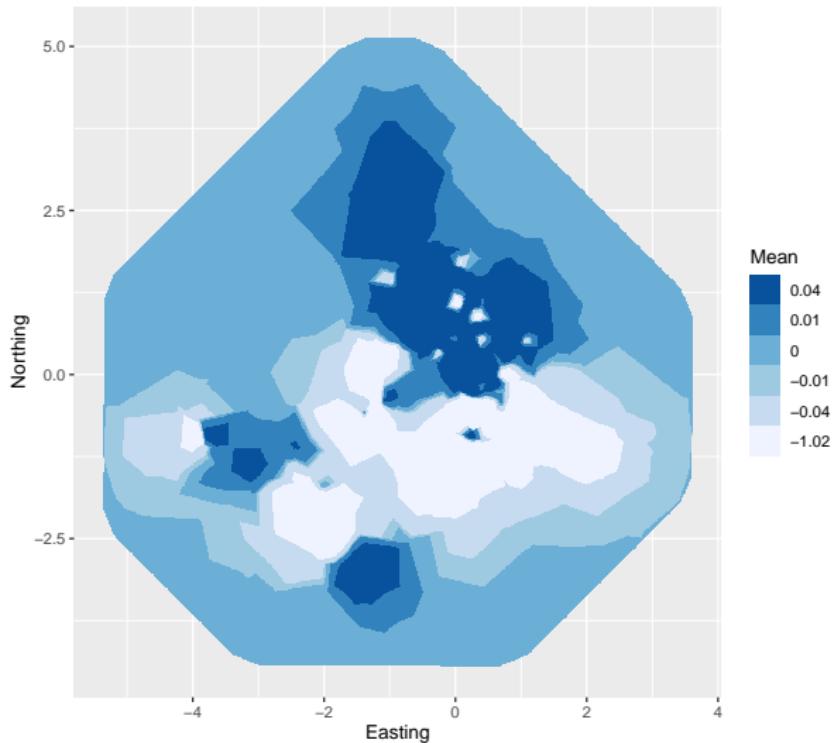
- We also compute NLSCPO, DIC, and the standard deviation of mean residuals for INLA model 3.

```
## NLSCPO of INLA model 3: -122.6155  
## DIC of INLA model 3: -339.2312  
## Standard deviation of mean residuals for INLA model 3: 0.1160327
```

- We can see that this is the best model according to all 3 criteria. By using both spatial and temporal random effects, we have been able to decrease the standard deviation of the mean residuals by half!
- The figure on the next slide shows the mean of the spatial effects for INLA model 3.
- This is quite similar to what we had for model 2.

Spatial and Temporal Modelling

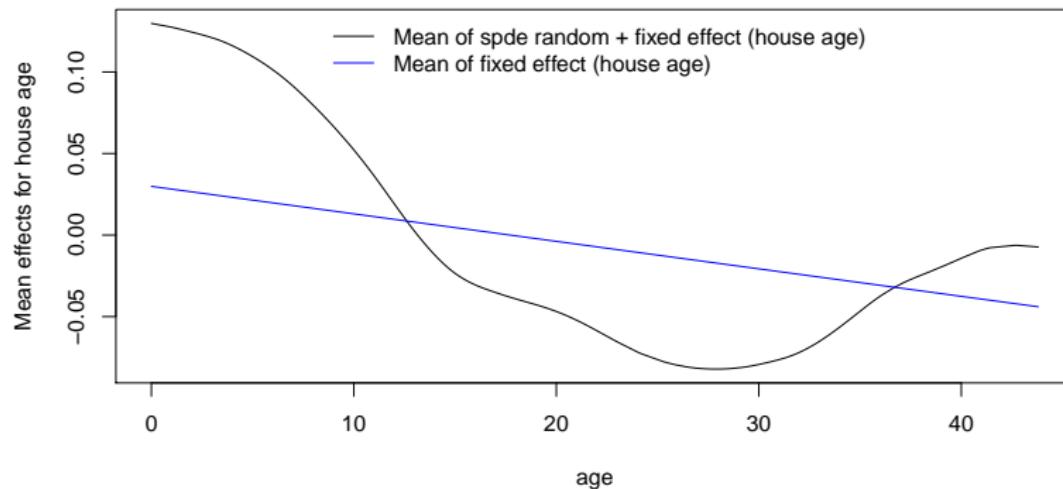
INLA model 3: SPDE spatial and temporal random effects



Spatial and Temporal Modelling

INLA model 3: SPDE spatial and temporal random effects

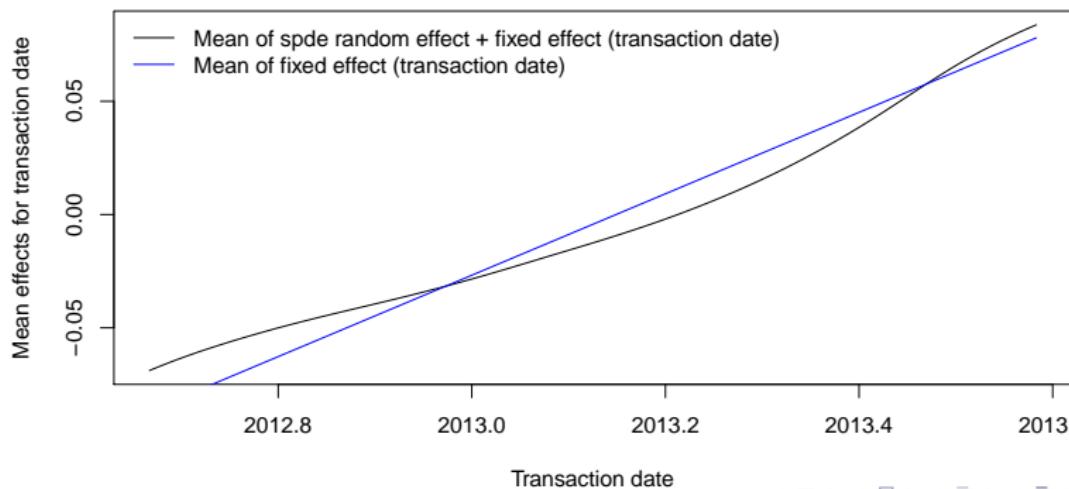
- As we can see on the plot, the dependence on the house age of the mean random effects is close to linear between 0 to 20 years, but becomes highly non-linear above 20 years.



Spatial and Temporal Modelling

INLA model 3: SPDE spatial and temporal random effects

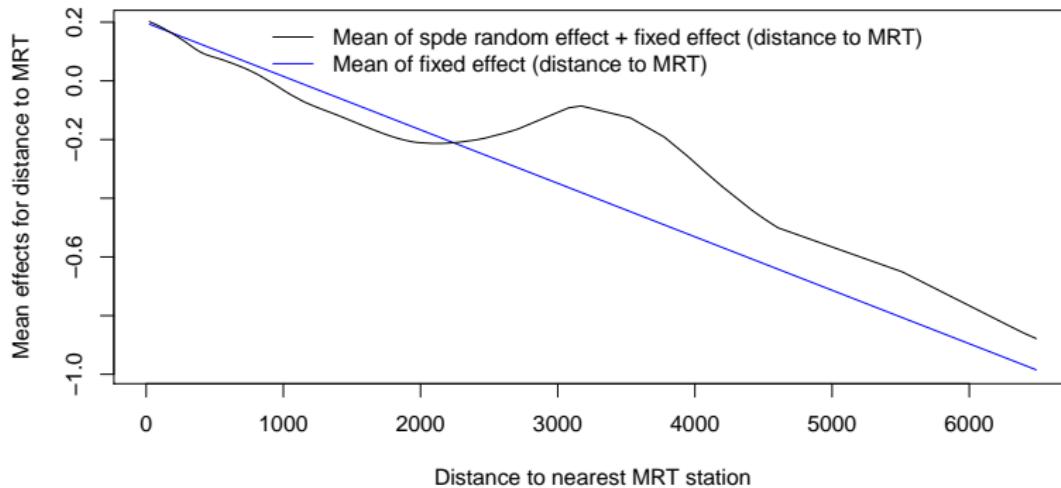
- As we can see on the plot, the dependence of the mean effect for transaction date on the transaction time is not very far from linear, but there is clearly faster growth starting from 2013.4, which corresponds to spring time. Indeed, spring and summer seems to be the best time to sell a house as the market is most active at that time, see <https://www.theadvisory.co.uk/house-selling/best-time-to-sell-house/>.



Spatial and Temporal Modelling

INLA model 3: SPDE spatial and temporal random effects

- The dependence of the mean effect for distance to MRT on the distance to MRT is not very far from linear, but there is a flat area between 2500-3500 meters..



Spatial and Temporal Modelling

Conclusion

- ↪ Overall, INLA Models 2 and 3 have much better performance than the simple Bayesian linear regression model, INLA Model 1.
- ↪ The advantage will further increase with more data, as these spde random effect models are able to flexibly increase their complexity (effective number of parameters) to match the amount of available data.
- ↪ The effective number of parameters is controlled by the number of cells in the mesh, and the priors of the spde models.
- ↪ The number of parameters of the linear regression model is fixed at 7 so it is unable to express complex spatial and temporal relationships.