Simulating Generalized Linear Models with INLA

Arnau Escapa Farrés

January 31, 2018

Abstract

INLA stands for Integrated Nested Laplace Approximations. It is used for fitting Latent Gaussian models (LGM) which include a wide range of commonly used regression models. Unlike MCMC which uses simulation methods, INLA uses approximation methods for Bayesian model fitting. Within the class of LGMs, INLA can fit models much faster than MCMC-based methods.

In this project we will explain the INLA algorithm and use the `R-INLA` package to perform experiments on a particular family of the LGM models, the generalized linear models (GLM). In particular, we will give examples of a linear regression and Poisson regression models.

Contents

1 Background

This is a small project of a course of Bayessian Statistics about a topic not seen in class. The concepts related to the project that we have already seen in class are assumed to be known and we will not discuss them (i.e. Laplace Approximation, JAGS). In constrast, before working on the INLA algorithm, we first introduce some concepts that have not been seen during the course in order to contextualize the algorithm.

1.1 Bayesian hierarchical models

Bayesian hierarchical modelling is a statistical model written in multiple levels (hierarchical form) that estimates the parameters of the posterior distribution using the Bayesian method.

In this kind of models the parameters of the prior distribution may also be descrived as a random variables. We use the follow nomenclature: *hyperparameters* for the parameters of the piror distribution and *hyperprior* for the distribution of a hyperparameter.

Definition. Let y_j be an observation and θ_j a parameter that descrives the data generation process of y_j. Assume that the parameters $\theta_1, \ldots, \theta_n$ are generated exchangeably from a common population, with distribution governed by a hyperparameter ϕ . The Bayesian hierarchical model is defined by the following stages:

$$\text{Stage I: } y_j \mid \theta_j, \phi \sim P(y_j \mid \theta_j, \phi)$$
$$\text{Stage II: } \theta_j \mid \phi \sim P(\theta_j \mid \phi)$$
$$\text{Stage III: } \phi \sim P(\phi)$$

The *likelihood* is $P(y_j \mid \theta_j, \phi)$, with $P(\theta_j, \phi)$ as its *prior distribution*. Note that the likelihood depends on ϕ only through θ_j.

Using the definition of conditional probability the prior distribution can be splited into

$$P(\theta_j, \phi) = P(\theta_j \mid \phi)P(\phi)$$

with ϕ as its hyperparameter with *hyperprior distribution*, $P(\phi)$. Thus, using Bayes theorem, the posterior distribution is proportional to:

$$P(\phi, \theta_j \mid y) \propto P(y_j \mid \theta_j, \phi)P(\theta_j \mid \phi)$$

$$P(\phi, \theta_j \mid y) \propto P(y_j \mid \theta_j)P(\theta_j, \phi)$$

1.2 Latent Gaussian Models

Definition. Following [4], the general form for a LFM is giben by:

$$\begin{aligned} &\mathbf{y} \mid \theta, \psi_2 \sim \prod p(y_i \mid \eta_i, \psi_2) && \text{Likelihood} \\ &\theta \mid \psi_1 \sim p(\theta|\psi_1) = N(0, \Sigma) && \text{Latent field} \\ &\psi = [\_1, \psi_2] \sim p(\psi) && \text{Hyperpiors} \end{aligned}$$

where $\mathbf{y}$ is an observed dataset, θ are not covariates but rather is the joint distribution of all parameters in the linear predictor (including itself), and ψ are the hyperparameters of the latent field that are not Gaussian.

This is a very general and complex model that is strongly related to Gaussian Markov Random fields. The effort required to understand properly wide range of models that this definition embrace would exceed the aspirations of this project. Hence we will just state that it can be seen that the following models are include in the LGM:

1. Stardard regression
2. Hierarchical models
3. Spatial and spatio-temporal models
4. Spline smoothing

We will now speak about the Generalized Linear models which are a relatively simple kind of hierarchical models. We focus in this particulat kind of models because to give a reader a better understanding of the last part of the project where we will simulate some of them.

1.3 Generalized linear models

The generalized linear models (GLM) are a flexible generalization of ordinary linear regression that allows for response variables that have error distribution models other than a normal distribution. The GLM generalizes linear regression by allowing the linear model to be related to the response variable via a link function and by allowing the magnitude of the variance of each measurement to be a function of its predicted value. Let us give a formal definition the model by following [5].

Definition. *A generalized linear model (or GLM) stablishes a relationship between a response variable Y and the explanatory variables X. It consists of three components:*

1. *A random component, specifying the conditional distribution of the response variable, Y_i (for the i-th of n independently sampled observations), given the values of the explanatory variables in the model. In the initial formulation of GLMs, the distribution of Y_i was a member of an exponential family, such as the Gaussian, binomial, Poisson, gamma, or inverse-Gaussian families of distributions. Subsequent work, however, has extended GLMs to multivariate exponential families (such as the multinomial distribution), to certain non-exponential families (such as the two-parameter negative-binomial distribution), and to some situations in which the distribution of Yi is not specified completely.*

2. *A linear predictor - that is a linear function of regressors,*

$$\eta_i = \alpha + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_k X_{ik}$$

3. *A smooth and invertible linearizing link function $g(\cdot)$, which transforms the expectation of the response variable, $\mu_i = E(Y_i)$, to the linear predictor:*

$$g(\mu_i) = \eta_i = \alpha + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_k X_{ik}$$

This model is a generalization of various other statistical well known models, including linear regression, logistic regression or Poisson regression.

2 INLA algorithm

Let us know give the core ideas of how INLA work (again following [4]). Even though we treat the algorithm for the general LGM case ine imagine that we are working on fitting a hierarchical instead of a LGM in order to understand easily the following reasonaments.

Given the observations $\mathbf{y}$ what we want from our LGMs are the marginals for the elements of the latent field (i.e. regression parameters)

$$p(\theta_j \mid \mathbf{y}) = \int p(\theta_j, \psi \mid \mathbf{y}) d\psi \tag{1}$$

and the elements from hyperprior distributions (i.e., the correlation parameter in an autoregressive model, or the variance on a random effect).

$$p(\psi_j \mid \mathbf{y}) = \int p(\psi \mid \mathbf{y}) d\hat{\psi}_k \tag{2}$$

where $\hat{\psi}_k := (\psi_1, \ldots, \psi_{k-1}, \psi_{k+1}, \ldots, \psi_n)$. Observe that what we are doing is just fitting a hierarchical model.

Since the marginals (2) can be obtained from $p(\psi \mid \mathbf{y})$ for all k our goal is to approximate $p(\theta_j \mid \mathbf{y})$ and $p(\psi \mid \mathbf{y})$, from which we will compute the marginal postirior for the parameters. To do so INLA simply uses Bayes formula and Laplace approximation. Recall that from Bayes formula, assuming $p(b) > 0$

$$p(b) = \frac{p(a \cap b)}{p(a \mid b)} \Rightarrow p(b \mid c) = \frac{p(a \cap b \mid c)}{p(a \mid b \cap c)}$$

We can use this property to rewrite (2), using comas to denote intersection of events:

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

Now let $\bar{p}(\theta \mid \psi, \mathbf{y})$ be the Laplace approximation of $p(\theta \mid \psi, \mathbf{y})$. We approximate the last expression by

$$\begin{aligned} p(\psi \mid \mathbf{y}) &\propto \frac{p(\mathbf{y} \mid \theta) p(\theta \mid \psi) p(\psi)}{p(\theta \mid \psi, \mathbf{y})} \\ &\approx \left. \frac{p(\mathbf{y} \mid \theta) p(\theta \mid \psi) p(\psi)}{\bar{p}(\theta \mid \psi, \mathbf{y})} \right|_{\theta = \theta^*(\psi)} =: \bar{p}(\psi \mid \mathbf{y}) \end{aligned}$$

where $\theta^*(\psi)$ is its mode.

Let us now work on giving an approximation of (1). We write θ as $\{\theta_j, \hat{\theta}_j\}$ and use the same trick again.

$$\begin{aligned} p(\theta_j \mid \psi, \mathbf{y}) &= \frac{p(\{\theta_j, \hat{\theta}_j\} \mid \psi, \mathbf{y})}{p(\hat{\theta}_j \mid \theta_j, \psi, \mathbf{y})} \\ &= \frac{p(\{\theta_j, \hat{\theta}_j\}, \psi \mid \mathbf{y})}{p(\psi \mid \mathbf{y}) p(\hat{\theta}_j \mid \theta_j, \psi, \mathbf{y})} \\ &\propto \frac{p(\theta, \psi|\mathbf{y})}{p(\hat{\theta}_j \mid \theta_j, \psi, \mathbf{y})} \propto \frac{p(\psi)p(\theta \mid \psi)p(\mathbf{y} \mid \theta)}{p(\hat{\theta}_j \mid \theta_j, \psi, \mathbf{y})} \\ &\approx \left. \frac{p(\psi)p(\theta \mid \psi)p(\mathbf{y} \mid \theta)}{\bar{p}(\hat{\theta}_j \mid \theta_j, \psi, \mathbf{y})} \right|_{\hat{\theta}_j = \hat{\theta}_j^*(\theta_j, \psi)} =: \bar{p}(\theta_j \mid \psi, \mathbf{y}) \end{aligned}$$

where again $\bar{p}$ denotes the respective Laplace approximation and $\hat{\theta}^*$ the mode.

The approximation works generally well. However, this strategy can be computationally expensive for large dimensionaliy of the parameters because Laplace appoximation requires computing the inverse of a Hassian matrix. As an alternative one may use the **Simplified Laplace Approximation**, a most efficient algorithm than the Laplace Approximation but less precise. As well as Laplace Approximation, SLA is based on the Taylor series up to the third orther of our target function. In this case of the both numerator and denominator of $\bar{p}(\theta_j \mid \psi, \mathbf{y})$. `R-INLA` uses by defualt SLA, but if more precision is required it is possible to set the stardard Laplace Approximation - at the expense of a higher running time.

The last key point of INLA is using Newton-like methods to explore the joint posterior distribution for the hyperparameters $\bar{p}(\psi \mid \mathbf{y})$ to find suitable points for the numerical integration. This is

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

Observe that in addition to LA we used numerical integration and $p(\psi \mid \mathbf{y})$ to compute $p(\theta_j \mid \mathbf{y})$. This is the reason why the algoithm is called Integrated Nested Laplace Approximation.

3 Modelling with `R-INLA`

In this seccion we will perform a trial of the INLA algorithm using the package `R-INLA`. For further information about the package check its home page [6].

`R-INLA` works in a similar way than JAGS in the sense that we just have to define the model in its own language. Toghether with this report you can find a pair of R scripts that implement INLA for the following experimets. For running the code make sure to download the required packages, `INLA` and `R2JAGS` (we also use JAGS for comparissons and ggplots2 for nice plots).

3.1 A basic example: Linear Regression

We will begin our trial with a well-known model, linear regression. In this case the data (x_i, y_i) follow the model given by the parameters (a, b) and the hyperparameter σ^2.

$$y_i = a + bx_i + \varepsilon_i \qquad \varepsilon_i \sim N(0, \sigma^2)$$

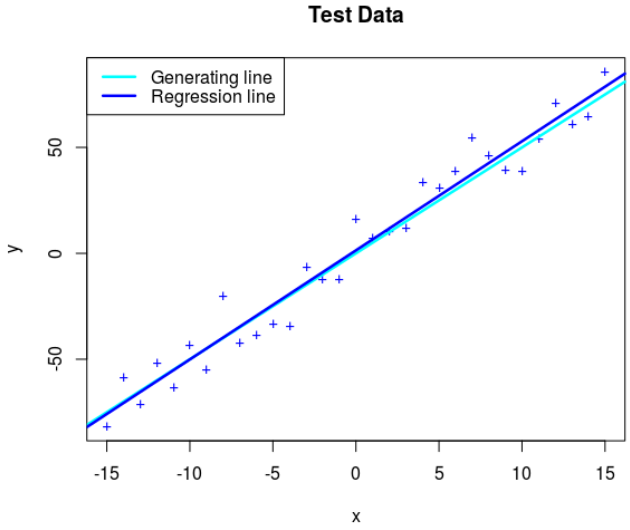
It is easy to see that this is a very paricullar case of the LGM, the simpliest in fact. When coding we may also use the precision parameter $\tau := \sigma^{-2}$ intead of the tradicional σ^2 . This is because some models use τ for computacional efficiency reasons.

In order to evaluate the performance of INLA we also fit the parameters using the frequentist approach and JAGS.

We first generate $n = 31$ samples of data setting $a = 0, b = 5$ and $\sigma = 11$.

Frequentist Approach

The advantage of working in such a simple and well-known model is that we can take advantage of the existing R functions and saving lots of efforts coding classical statistics. We can estimate b just fitting our data with the linear model routine `lm`. If we do so we obtain the maximum likelihodd estimators of the paramaters are giben by of beta given by $a_F = 1.40, b_F := 5.15$ and $\sigma_F := 9.702$. The 95% confidence interval of b is $I_F = (4.948, 5.223)$.



The results are congruent with the generated data.

3.1.1 MCMC - Using JAGS

We procedure now to use JAGS algortihm to generate a sample of the posterior. Since we have already seen this model during the course we do not give further explanations and we just present the following results. Using the following priors:

- Normal prior for the intercept a, with mean=0 and sd=5.
- Uniform(0,10) for the slope b.
- Uniform(0,30) for σ.

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

    Mean     SD  Naive SE Time-series SE
a  0.150 1.4482 0.0051200      0.0051201
b  4.928 0.1703 0.0006022      0.0007954
sd 8.367 1.1604 0.0041026      0.0060749

2. Quantiles for each variable:

     2.5%     25%   50%   75%  97.5%
a  -2.714 -0.8093 0.143 1.110  3.023
b   4.592  4.8145 4.928 5.040  5.265
sd  6.459  7.5503 8.242 9.044 11.011
```

And we can use the generated sample to approximate the postirior density functions (JAGS does all this work for us):

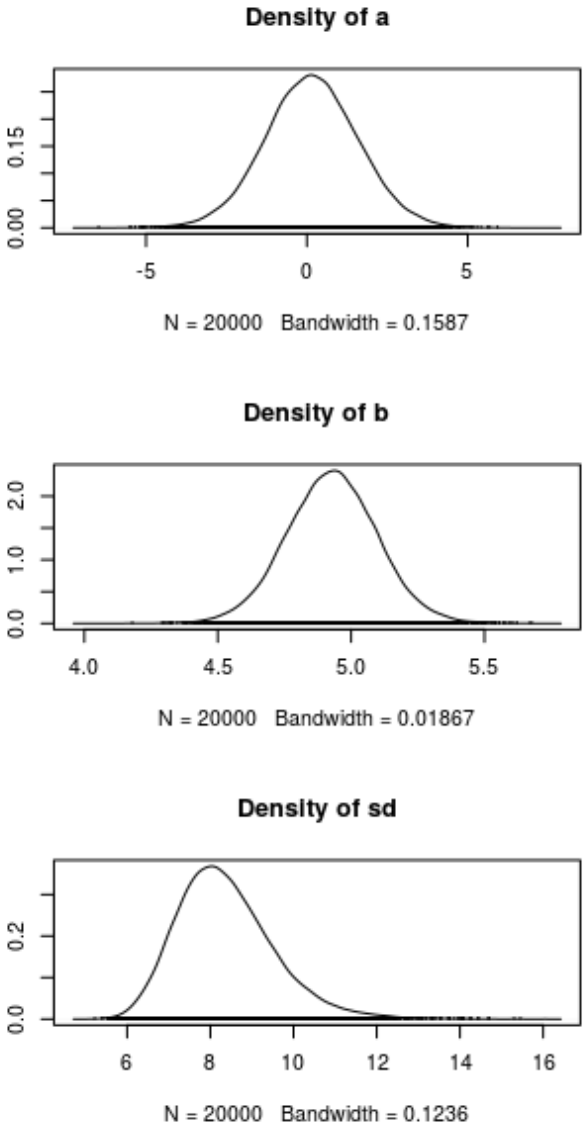


Figure 1: Pdf of the postirors of the parameters if the model using JAGS

Observe that we get pretty similar results compared to the frequentist model as expected.

3.1.2 Using INLA

INLA generates a posterior density rather than a simulated empirical distribution like MCMC.

The `inla()` function returns an object of class `inla` which is really similar to a `jags` object. The `inla` objects contanis the information of the fitted model, to get this information it has two methods available: `summary` and `plot`. For an inla fitted model, the summary method returns many information such that the the Fixed and Random effects as well as a number of information criterion and limited fit diagnostics. It can be a bit overwelming and hard to find the interesting information. For this reason we prefer to use `summary.fixed` and `summary.hyperpar` which give the mode and confidence intervals of the fitted parameters and hyperparameters of the model respectively.

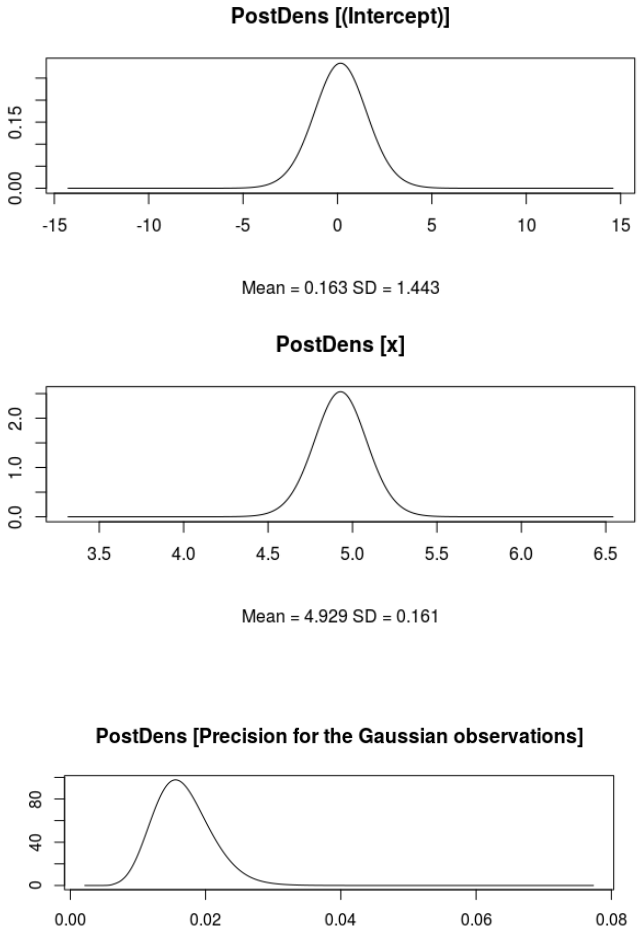
To use `inla()` we just have to give the data in a similar way than JAGS and chose a model. In our case, this is:

```
> imod <- inla(y ~ x , family="gaussian", data=data)
> imod$summary.fixed
         mean   sd   0.025qu  0.5qu   0.975qu  mode    kld
(Int)  0.1631 1.4432  -2.689  0.1631   3.011 0.1631 3.025e-13
x      4.9291 0.1613   4.610  4.9291   5.247 4.9291 3.752e-13

> imod$summary.hyperpar
Precision for the Gaussian observations
  mean    sd      0.025qu  0.5qu   0.975qu  mode
0.01656  0.00408  0.00944  0.0162  0.0254  0.015
```

where `(Int)`, `x` and `precision` yields for our paramater a, τ and b respectively. The mode $\tau_I = 0.015$ is equivalent to $\sigma_I = 8.164$ with 95% acceptance confidence interval: $(6.274, 10.292)$.

We also used the `plot` method to plot the distributions of the parameters obtaining:



We did the same than using JAGS (possibly computationally faster) and just with a few lines of code!

However, as we have seen along the course, the choise of the prior of our model may have a big impact on our final result. When fitting the INLA model above, we did not specify the prior. Consequently, the default priors were employed. To know which are these priors, we can use the `inla.show.hyperspec()` function - which returns a list containing information about all the hyper-priors used in a model.

The regression parameters of an LGM have, by definition, a Gaussian prior. The default mean and precision are $\mu = 0$ and $\tau = 0.001$. Let us know repeat the modelling using the same priors that we used with JAGS.

To set the priors we just use

```
imod <- inla(y ~ x , family="gaussian",
             control.fixed=list(prec=1/(5*5)), data=data)
```

In this case the obtained results are pretty similar than using the default prior. Since the default paramater and our imposed paramater are not that different makes sense.

3.1.3 Prediction

Another very cool tool of INLA is that we can actually get predictions of y given x in a very confortable way. We just have to add the y component of the data we want to predict as a `NA` (NaN) and we pass it toghether with the known data. This can be made by

```
newdata <- data.frame(x=seq(min(data$x, na.rm=TRUE),
                      max(data$x, na.rm=TRUE), len=100), y=NA)
data.pred <- rbind(data, newdata)
```

Observe that we just added 100 $(X, \texttt{NA})$ points along the hole X range. Now we just run the model again with the new data set.

```
imod.pred <- inla(y~x, data=data.pred, control.fixed=list(prec=1/25))
```

the method not only gives the most likely value for each point that has NA as y coordinate but also gives a level of uncertintany. We can use this data

```
newdata <- cbind(newdata, imod.pred$summary.linear.predictor
                          [(nrow(data)+1):nrow(data.pred),])
```

to make a super cool plot that let us visualise the confidence interval regions for each x value.

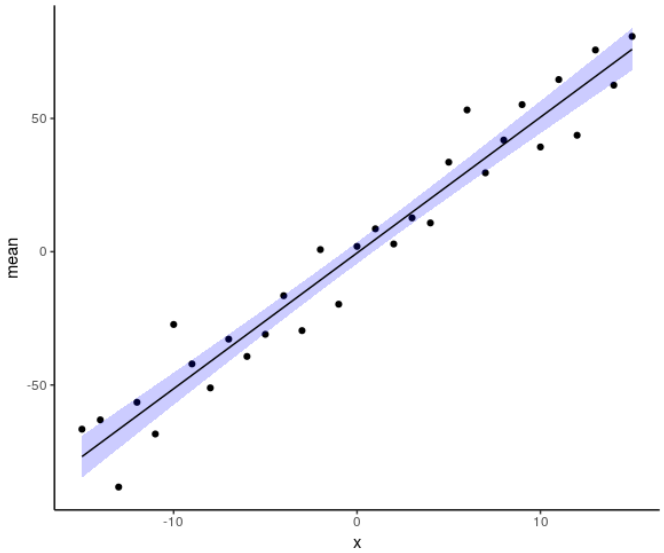


Figure 2: The blue region gives the confidence interval of the predicted value y given a sample x.

This is just a improved line regression. The good thing about INLA is that we can do the same for much more complex regression models in a very efficient way.

3.2 Just another example

Let us just assume that we want to model the amount that a phenomena is produced y against a continuous predictor x. Counting events is a Poisson model. Hence our model consist in fitting

$$y_i \sim Pois(\lambda_i)$$
$$\lambda_i = \beta_0 + \beta_1 x_i$$

We generated some data setting $\beta_0 = \beta_1 = 0.5$ and repeated the procedure of the last section. The file `PoissonRegrtion.R` constains the R script that contains the code which gave us the following results.

```
            mean       sd     0.025quant  0.5quant 0.975quant    mode
(Int)    0.5668569 0.22696098 0.10595990 0.5722504  0.9979420 0.5831673
x        0.1098766 0.01757997 0.07600562 0.1096544  0.1450242 0.1092072
```

where `(Int)` holds for our β_0 and `x` for β_1. We failed fitting β_1, even though our plots seems okey.

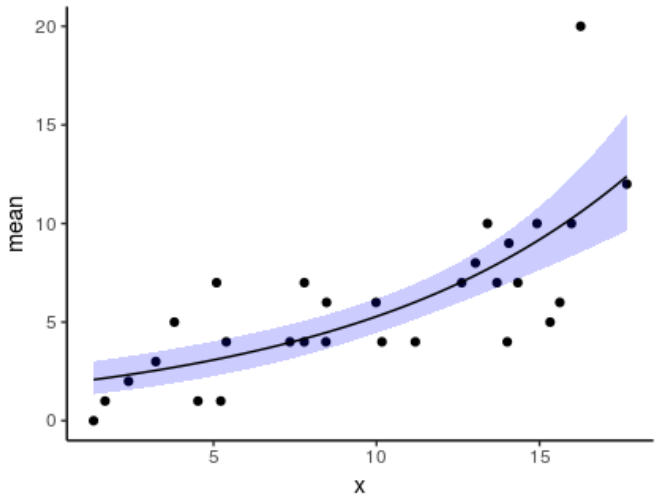


Figure 3: N=30.

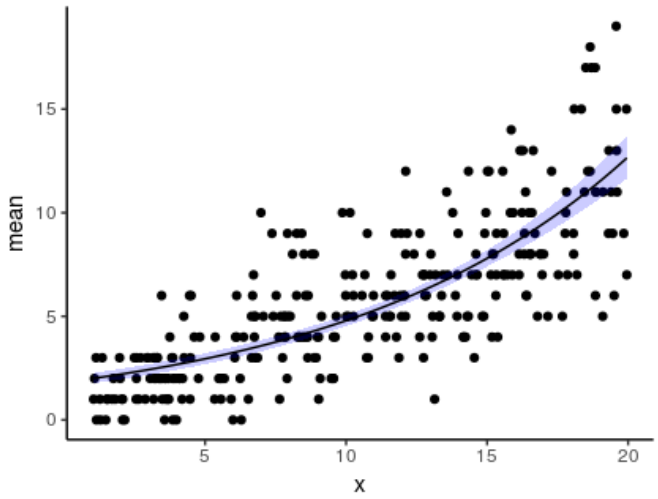


Figure 4: N=300.

We observed a better performance using logarithmic scale. This is considering the model:

$$y_i \sim Pois(\lambda_i)$$
$$log(\lambda_i) = \beta_0 + \beta_1 x_i$$

Setting $\beta_0 = 0.15$ and $\beta_1 = 0.075$ and generating $n = 50$ samples we obtained.

	mean
(Intercept)	0.156
x	0.070

References

[1] Rue H., Martino S. and Chopin N., *Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations* Journal of the Royal Statistical Society, 2009

[2] Fong Y., Rue H. and Wakefield J. *Bayesian inference for Generalized Linear Mixed Models* 2010.

[3] Gianluca Baio. *An introduction to INLA with a comparison to JAGS* University College London, 2013.

[4] A gentle INLA tutorial https://www.precision-analytics.ca/blog-1/inla

[5] Generalized Linear Model Theory. http://data.princeton.edu/wws509/notes/a2.pdf

[6] Package R-INLA documentation http://www.r-inla.org/