GPH-GU2372/3372
Applied Bayesian Analysis in Public Health

Lesson 9: Metropolis-Hastings algorithms

Hai Shu, PhD

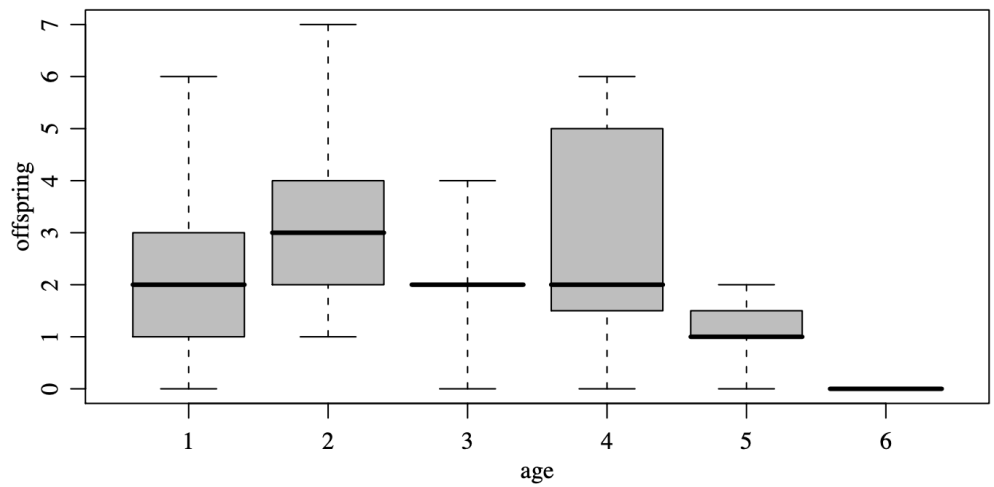
11/21/2022

Topics

- Generalized linear models
- The Metropolis algorithm
- The Metropolis-Hastings algorithm
- Combining the Metropolis and Gibbs algorithms

Generalized linear models

- We have seen previously the case of normal linear regression models. Not all the response variables fall within that framework.
- Example: 52 female song sparrows were studied over the course of a summer and for each sparrow, the age and the number of new offsprings were recorded.
- A boxplot that shows the distribution of the number of offsprings versus age of the female sparrow is given below



Two-year old birds had the highest median with the median number of offsprings declining as age increases.

Generalized linear models

- Suppose we want to fit a model to these data.
- The data here will be $y_1, \ldots, y_{52}$, the number of offspring for each of the 52 female sparrows.
- Each of the y_i is a non-negative integer in the set $\{0, 1, 2, \ldots\}$. Therefore, a good choice to model this data would be a Poisson model.
- Specifically, we would model each Y_i as a Poisson random variable with parameter θ_i.
- Given the boxplot, we can imagine that the parameter θ_i depends on the age x_i of bird i.
- However, probably the relationship between θ_i and x_i is not linear, but rather quadratic, that is:

$$\theta_i = \beta_1 + \beta_2 x_i + \beta_3 x_i^2$$

- Since θ_i needs to be positive, rather than working with θ_i directly, we model instead $\log(\theta_i)$.

Poisson linear regression

- Therefore the model that we consider is:

$$\log E(Y_i|x_i) = \log(\theta_i) = \beta_1 + \beta_2 x_i + \beta_3 x_i^2$$

or, equivalently

$$E(Y_i|x_i) = \theta_i = \exp\left(\beta_1 + \beta_2 x_i + \beta_3 x_i^2\right)$$

where Y_i is a Poisson random variable, for $i = 1, \ldots, n$.

- This is an example of a Poisson regression model. A Poisson regression model is an example of a generalized linear model.
- In this case the conditional expectation of Y_i is related to the linear regressor, or linear predictor $\beta_1 + \beta_2 x_i + \beta_3 x_i^2$ via the log function.
- For this reason, we say that this is a generalized linear model with a log link.

Poisson linear regression

- Now, if we want to make inference on the regression parameters $\beta_1, \beta_2, \beta_3$ from a Bayesian perspective, we would place a prior on $\beta = (\beta_1\ \beta_2\ \beta_3)'$ and derive the posterior distribution $p(\beta|\mathbf{y} = (y_1, \ldots, y_n)', \mathbf{X})$.
- A common choice for a prior on β is a multivariate normal, but this does not lead to a closed form posterior distribution.
- So, what do we do?
- We have seen previously some ways to handle non-closed form posterior distribution, now, we will look at a different algorithm: the Metropolis algorithm (and some of its variants).

Metropolis algorithm

- Note that the problem of non-closed form posterior distribution does not occur only for Poisson regression models: it occurs for any generalized linear model with a link function that is not the identity.
- Therefore, also for a logistic regression model with a logit link, we have a similar problem.
- To describe the Metropolis algorithm let's consider the general problem of wanting to approximate the posterior distribution $p(\theta|\mathbf{y})$ of a parameter θ given some data $\mathbf{y}$.
- The Metropolis algorithm is an algorithm that generates a Markov chain $\theta^{(1)}, \theta^{(2)}, \ldots$ with a specified target distribution $p(\theta)$ by using an acceptance/rejection rule that ensures convergence to the specified target distribution.
- In our case, the target distribution is the posterior distribution $p(\theta|\mathbf{y})$.

Metropolis algorithm

- The algorithm works as follows:
  - Choose a number S of iterations.
  - Sample a starting value $\theta^{(0)}$ for which $p(\theta^{(0)}|\mathbf{y}) > 0$ from some starting distribution $p_0(\theta)$. This starting distribution might be an approximation to the posterior distribution $p(\theta|\mathbf{y})$, or $\theta^{(0)}$ might be an initial value close to the mean of $p(\theta|\mathbf{y})$.
  - For iterations $k = 1, \ldots, S$, repeat the following steps
    1. sample a proposal or candidate value $\theta^\star$ from a jumping or proposal distribution $J_k(\theta^\star|\theta^{(k-1)})$.
       For the Metropolis algorithm, the jumping distribution must be symmetric, that is: $J_k(\theta_a|\theta_b) = J_k(\theta_b|\theta_a)$ for **all** θ_b, θ_a and k.
    2. calculate the ratio of densities $r = \frac{p(\theta^\star|\mathbf{y})}{p(\theta^{(k-1)}|\mathbf{y})}$
    3. set $\theta^{(k)}$ equal to $\theta^\star$ with probability equal to $\min(r, 1)$. Otherwise, set $\theta^{(k)}$ equal to $\theta^{(k-1)}$.

Metropolis algorithm: some intuition

- To decide whether to accept or reject a proposed value $\theta^\star$, we need to compare the posterior distribution $p(\theta|\mathbf{y})$ at the proposed value $\theta^\star$ and at the current value $\theta^{(k-1)}$. Here the posterior distribution can be known up to the proportionality constant $p(\mathbf{y})$.
- Usually, since r is small, we compute the ratio on the log scale and then exponentiate it:

$$\log(r) = \log\left(\frac{p(\theta^\star|\mathbf{y})}{p(\theta^{(k-1)}|\mathbf{y})}\right) = \log(p(\theta^\star|\mathbf{y})) - \log(p(\theta^{(k-1)}|\mathbf{y}))$$

- Does the acceptance/rejection rule make sense?
- If $r > 1$, it means that $p(\theta^\star|\mathbf{y}) > p(\theta^{(k-1)}|\mathbf{y})$: therefore our Markov chain should include $\theta^{(\star)}$ rather than $\theta^{(k-1)}$. Hence, we set $\theta^{(k)} = \theta^\star$
- If $r < 1$, then $p(\theta^\star|\mathbf{y}) < p(\theta^{(k-1)}|\mathbf{y})$. The proportion of $\theta^{(\star)}$ to $\theta^{(k-1)}$ in the Markov chain should be equal to $\frac{p(\theta^\star|\mathbf{y})}{p(\theta^{(k-1)}|\mathbf{y})} = r$.

  Therefore, we are going to set $\theta^{(k)}$ equal to $\theta^\star$ or $\theta^{(k-1)}$ with probabilities r and $1-r$, respectively.

Metropolis algorithm: the proposal distribution

- How do we choose the proposal or jumping distribution $J_k(\theta^\star|\theta^{(k-1)})$?
- A good jumping distribution has the following properties:
  - for any θ, it is easy to sample from $J(\theta^\star|\theta)$.
  - each jump goes a reasonable distance in the parameter space, otherwise the Markov chain moves too slowly.
  - the jumps are not rejected too frequently, otherwise the Markov chain spends too much time standing still.
- Usual choices for the jumping distribution $J(\theta^\star|\theta)$ are:
  - A uniform distribution: $J(\theta^\star|\theta) = \text{Uniform}(\theta - \delta, \theta + \delta)$
  - A normal distribution: $J(\theta^\star|\theta) = \text{N}(\theta; \delta^2)$
- In both those choices, the parameter δ is chosen so that the algorithm runs efficiently.

Metropolis algorithm in practice

- In practice, the last step of the Metropolis algorithm, the accept or reject step, is performed in the following way:
  - sample a value $u \sim \text{Uniform}(0,1)$.
  - if $u < r = \frac{p(\theta^\star|\mathbf{y})}{p(\theta^{(k-1)}|\mathbf{y})}$, then $\theta^{(k)} = \theta^\star$, otherwise $\theta^{(k)} = \theta^{(k-1)}$.
- Let's actually check that the Metropolis algorithm works by applying it to a toy example for which we know the form of the posterior distribution.
- Let's consider the case where we have data $\mathbf{y} = (9.37, 10.18, 9.16, 11.60, 10.33)$ from a normal distribution $N(\theta, \sigma^2)$ where we assume σ^2 known and equal to 1.
- Thus, we want to infer upon the mean, θ. We place a $N(\mu, \tau^2)$ prior on θ with $\mu = 5$ and $\tau^2 = 10$.
- We know then that the posterior distribution $p(\theta|\mathbf{y})$ is $N(\mu_n, \tau_n^2)$ where

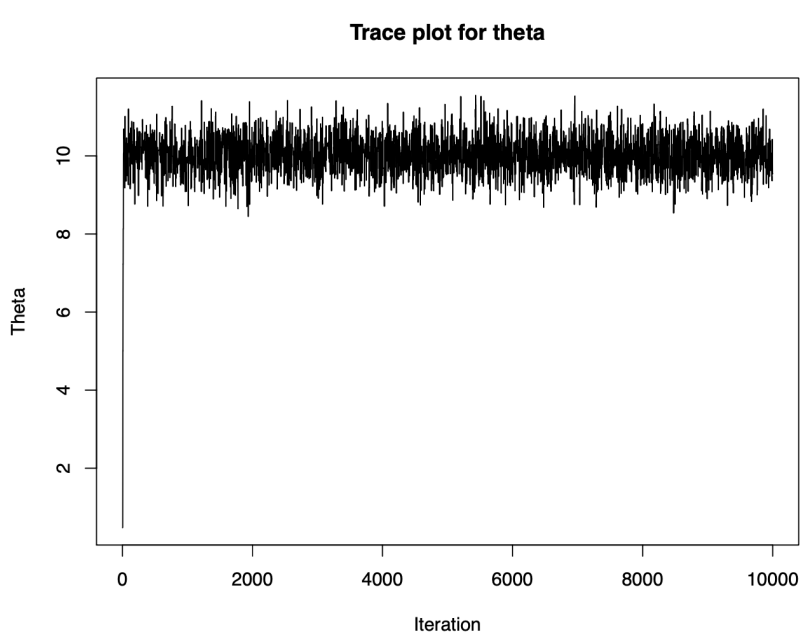
$$\tau_n^2 = \frac{1}{\frac{n}{\sigma^2} + \frac{1}{\tau^2}} = 0.20 \qquad \mu_n = \frac{\frac{n}{\sigma^2}\bar{y} + \frac{\mu}{\tau^2}}{\frac{n}{\sigma^2} + \frac{1}{\tau^2}} = 10.03$$

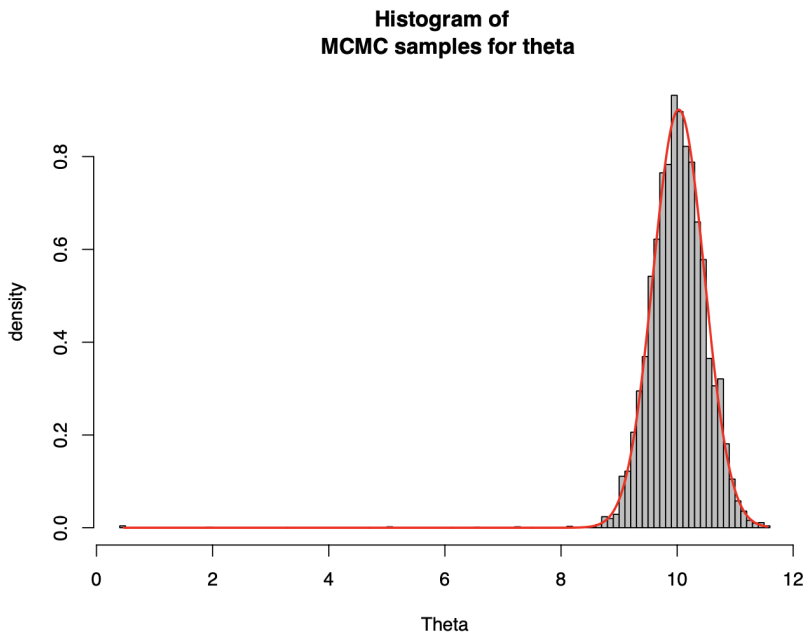
Metropolis algorithm in practice

- We implement a Metropolis algorithm where the proposal distribution is a normal distribution centered at the current value of θ with variance equal to δ^2, that is: $J_k(\theta^\star|\theta^{(k-1)})$
  $J_k(\theta^\star|\theta^{(k-1)}) = N(\theta^\star; \theta^{(k-1)}, \delta^2)$
- We ran the Metropolis algorithm for $S = 10,000$ iterations using $\theta^{(0)} = 0$ as starting value for θ and proposal variance $\delta^2 = 2$.

Metropolis algorithm in practice

- Trace plot for θ and histogram of the $S = 10,000$ sampled values for θ with overimposed the posterior density $N(\mu_n = 10.3, \tau_n^2 = 0.20)$ are provided below.



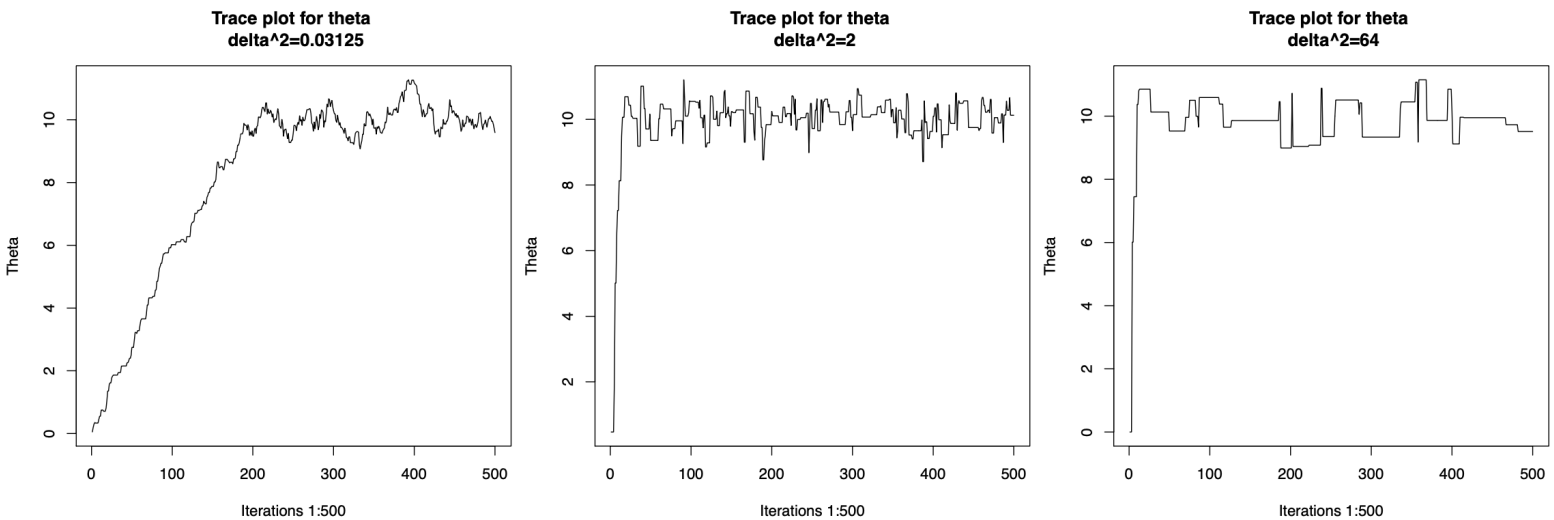


Metropolis algorithm in practice

- The Metropolis algorithm produces a Markov chain that has as stationary distribution the posterior distribution $p(\theta|\mathbf{y})$ of interest.
- As a consequence, before performing posterior inference, we need to run all the MCMC diagnostics that we have seen previously, that is, we need to verify that stationarity has been reached, and we make posterior inference after discarding the samples in the burnin period.
- The Metropolis algorithm generates a sequence of dependent samples for θ. The extent of the dependence among successive sampled values of θ depends on the jumping distribution $J(\theta^\star|\theta)$.
- In the previous example, the jumping distribution $J_k(\theta^\star|\theta^{(k-1)})$ was $N(\theta^\star; \theta^{(k-1)}, \delta^2)$. By varying δ^2 we can increase or decrease the correlation in the sample.
  A decrease in the autocorrelation will lead to: (i) an increase in the rate of convergence, (ii) an increase in the effective sample size and (iii) an improvement in the Markov chain approximation to the posterior distribution.

Metropolis algorithm in practice

- As an example, we re-ran the Metropolis algorithm for the toy example varying the value of δ^2, taking it to be in the set $\left\{\frac{1}{32}, \frac{1}{2}, 2, 32, 64\right\}$. These choices will lead to lag-1 autocorrelation of, respectively, $0.98, 0.77, 0.69, 0.85, 0.89$.
- Trace plots for the first 500 iterations corresponding to $\delta^2 = \frac{1}{32}$, $\delta^2 = 2$, $\delta^2 = 64$ are provided below:



Metropolis algorithm in practice

- In the left-most plot, $\delta^2 = \frac{1}{32}$: the algorithm has a high acceptance rate (about 87%), so the chain is moving, but the moves are never really large and thus the lag-1 autocorrelation is very high (about 0.98). As a consequence, it takes a lot of iterations for the Markov chain to move away from the starting value.
- In the right-most plot, $\delta^2 = 64$: the algorithm has a low acceptance rate (about 7%). The chain moves quickly to the posterior mode but once there it gets stuck. Because of that, the Markov chain has a very high autocorrelation (about 0.89).
- In the middle panel, $\delta^2 = 2$: the algorithm has an acceptance rate of 36%. The chain moves around the parameter space, but it does not get into regions of the parameter space where the proposed value is always rejected. The chain has a lag-1 autocorrelation of approximately 0.7.

Metropolis algorithm in practice

- In practice the proposal variance δ^2 is chosen by doing the following:
    - run the Metropolis algorithm several times for a small number of iterations, say 1000 under different choices of δ
    - compute the acceptance rate for each of these runs and select the value of δ that gives an acceptance rate within the range $20-50\%$.
    - Fix the value δ and run the Metropolis algorithm for the whole S iterations.
- Note that the proposal variance should **NOT** be changing in the Metropolis algorithm after the burnin period.

Poisson linear regression

- We have looked at the toy example to show how the Metropolis algorithm actually works. Now, we look at an actual real example where we don't know the posterior distributions of the parameters.
- Again consider the data for the 52 female sparrow birds: the data here is the number y_i of offspring for bird i of age x_i.
- The Poisson regression model that we considered is:

$$\log E(Y_i|x_i) = \log \theta_i = \beta_1 + \beta_2 x_i + \beta_3 x_i^2$$

- Let's denote with $\mathbf{x}_i$ the vector $\begin{pmatrix} 1 \\ x_i \\ x_i^2 \end{pmatrix}$, and with $\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix}$, then $\log E(Y_i|\mathbf{x}_i) = \log(\theta_i) = \beta' \mathbf{x}_i$ or, equivalently, $E(Y_i|\mathbf{x}_i) = \theta_i = \exp(\beta' \cdot \mathbf{x}_i)$.
- As prior distribution on β we use a multivariate normal $N_3(\mathbf{0}, 100 \cdot \mathbf{I}_3)$ or equivalently, the β_l's are independent a priori, each with a univariate normal prior $N(0, 100)$ for $l = 1, 2, 3$.

Poisson linear regression

- The posterior distribution $p(\beta|\mathbf{y})$ is given by:

$$\begin{aligned} p(\beta|\mathbf{y},\mathbf{x}) &\propto p(\mathbf{y}|\beta,\mathbf{x}) \cdot p(\theta) \\ &= \left[\prod_{i=1}^{n} \text{Poisson}\left(y_i; \exp\left(\beta' \cdot \mathbf{x}_i\right)\right)\right] \cdot \prod_{l=1}^{3} N(\beta_l; 0, 100) \end{aligned}$$

- This is clearly not a distribution that we know, so we will use a Metropolis algorithm to generate a Markov chain that has as stationary distribution $p(\beta|\mathbf{y},\mathbf{x})$.
- At iteration k, we use as jumping distribution $J_k(\beta^\star|\beta^{(k-1)})$ a multivariate normal distribution $N_3(\beta^{(k-1)}, \mathbf{V})$.
- We need to make a choice for the (co)variance matrix of the jumping distribution $J_k(\beta^\star|\beta^{(k-1)})$.
- We want the Metropolis algorithm to be as efficient possible. A choice of the (co)variance matrix $\mathbf{V}$ that yields an efficient algorithm is if $\mathbf{V}$ was the posterior variance of β or a rough approximation of it.

Poisson linear regression

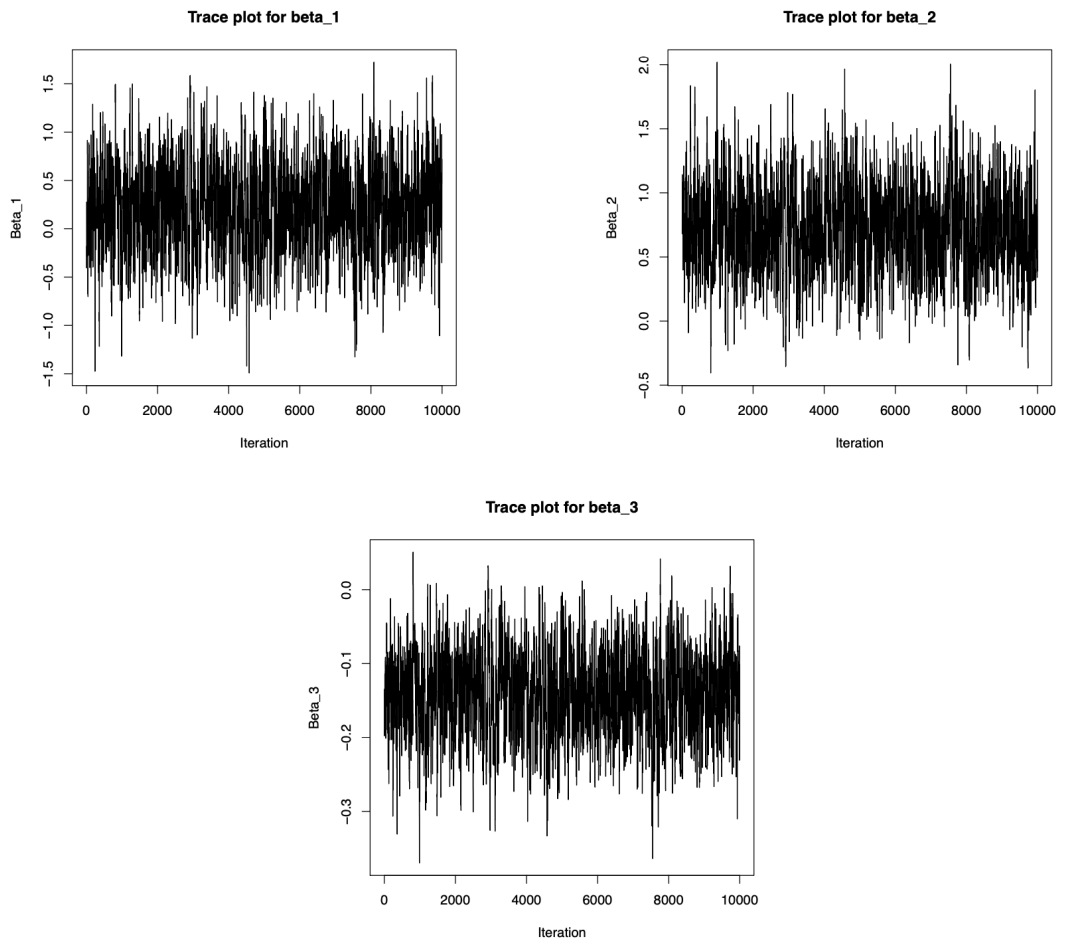
- We don't know the form of the posterior variance, but we can try to approximate it.
- In a normal linear regression model $\mathbf{y} \sim N_n(\mathbf{X}\beta, \sigma^2\mathbf{I}_n)$, we have seen that if $\beta \sim N_p(\beta_0, \mathbf{\Sigma_0})$, then, **given** σ^2, the posterior variance of β is a combination of the prior precision $\mathbf{\Sigma}_0^{-1}$ and of $\sigma^2(\mathbf{X}'\mathbf{X})^{-1}$ where σ^2 is the variance in the data, $y_1, \ldots, y_n$.
- We can use this result to derive a rough approximation to the posterior variance of β.
- Therefore we will set $\mathbf{V}$ equal to $\hat{\sigma}^2(\mathbf{X}'\mathbf{X})^{-1}$.
  Here our model is for $\log E(Y_i|\mathbf{x}_i)$, $i = 1, \ldots, n$, and not for $E(Y_i|\mathbf{x}_i)$, $i = 1, \ldots, n$, as in the normal linear regression model. Thus, we take $\hat{\sigma}^2$ to be an estimate of the variance in $\log(y_1), \ldots, \log(y_n)$.
- To avoid problems with zeroes, we actually set $\hat{\sigma}^2$ to be equal to the variance of $(\log(y_1 + \frac{1}{2}), \log(y_2 + \frac{1}{2}), \ldots, \log(y_n + \frac{1}{2}))$.

Poisson linear regression

- Hence, $J_k(\beta^\star|\beta^{(k-1)}) = N_3(\beta^{(k-1)}, \hat{\sigma}^2 \cdot (\mathbf{X}'\mathbf{X})^{-1})$ with $\hat{\sigma}^2$ as above.
- If this choice of the proposal variance does not lead to an acceptance rate in the range $20-50\%$, we can modify the (co)variance matrix of the proposal to $\delta\hat{\sigma}^2(\mathbf{X}'\mathbf{X})^{-1}$ with δ determined so to achieve the "right" acceptance rate.
- The R software code to fit this model is posted.
- As initial values for $\beta_1, \beta_2, \beta_3$ we use the frequentist estimates of the regression coefficients in a Poisson regression model.
- We ran the Metropolis algorithm for $S = 10,000$ iterations and we performed the usual MCMC diagnostics to make sure that the chain had reached its stationary distribution.
- Using the jumping distribution described above leads to an acceptance rate of about 41.4%.

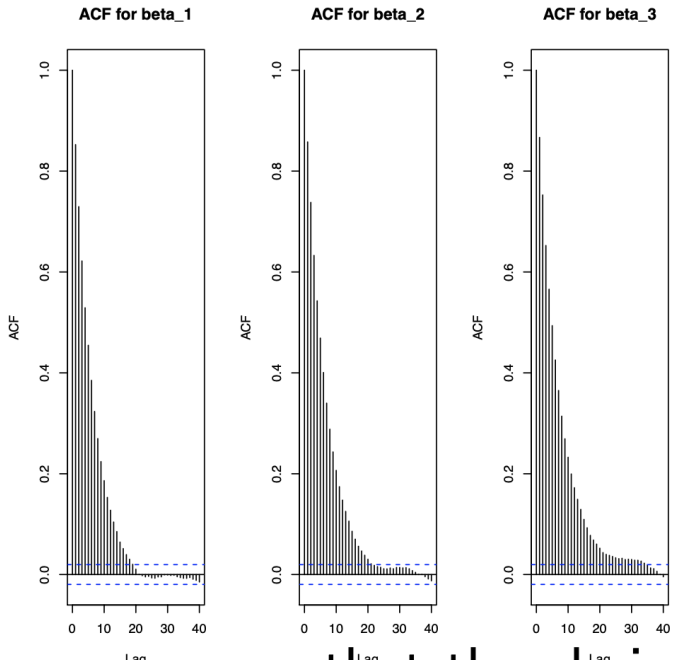
Poisson linear regression: an example

- Trace plots for β_1, β_2 and β_3.



Poisson linear regression: an example

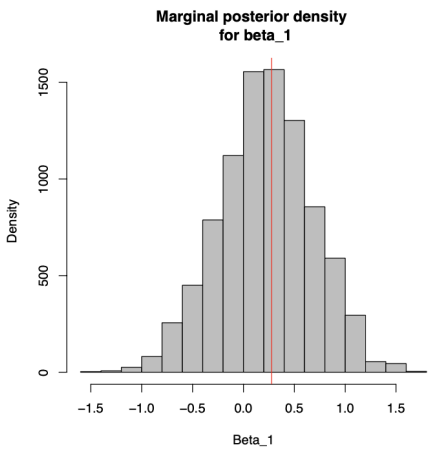
- Autocorrelation function for β_1, β_2 and β_3.

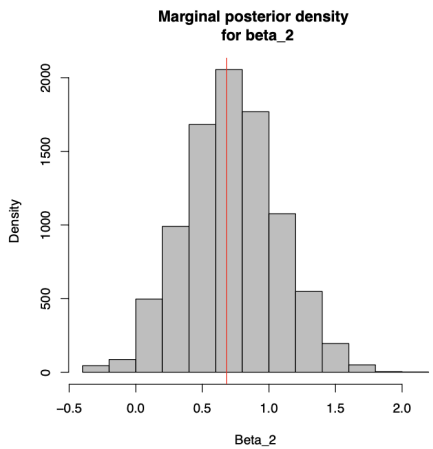


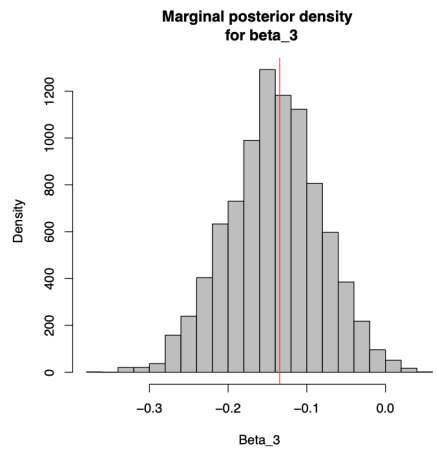
- From the ACF plots, we can see that the chains have an autocorrelation that is not significantly different from 0 at lag 30. Using a thinning of 10, we obtain almost independent chains.
- We could discard the first 1000 iterations for burnin and thin the MCMC chains using a thinning of 10, then we would we base our posterior inference on approximately 900 samples.

Poisson linear regression: an example

- Marginal posterior densities for all the coefficients are shown below. In each plot the frequentist GLM estimates of the regression coefficients are presented in red.

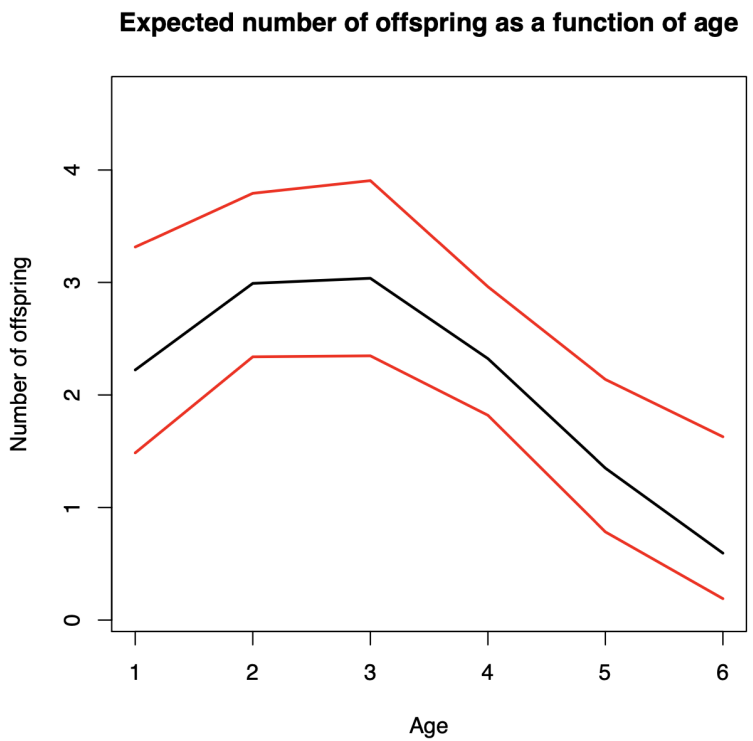






Poisson linear regression: an example

- We can also predict the number of offsprings for any female sparrows of ages between 1 to 6. Precisely we can generate posterior quantiles for the quantity $E[Y|\mathbf{x}]$ as the age of the sparrow varies between 1 and 6.



Metropolis algorithm: why it works?

- We have seen previously the intuition behind the Metropolis algorithm and we have also seen in the toy example that it does approximate the posterior distribution.
- However, we haven't seen why even though within each iteration we are actually sampling from the proposal distribution $J_k(\theta^\star|\theta^{(k-1)})$, the resulting Markov chain has as stationary distribution the right distribution, the posterior distribution $p(\theta|\mathbf{y})$.
- To prove that the Metropolis algorithm really works we need to prove two things:
  - that the Markov chain $\{\theta^{(1)}, \theta^{(2)}, \ldots,\}$ generated by the Metropolis algorithm has a unique stationary distribution
  - that the stationary distribution is the posterior distribution $p(\theta|\mathbf{y})$

Metropolis algorithm: why it works?

- We have seen that if a Markov chain is irreducible, positive recurrent and aperiodic, or equivalently, if it is irreducible and ergodic, then it admits a unique stationary distribution , where
  - irreducible means that the Markov chain has a positive probability to eventually reach any state from any other state
  - positive recurrent means that the Markov chain is expected to eventually return to any state in a finite number of steps
  - aperiodic means that there is no state to which the Markov chain returns regularly only every k steps
- It is clear that whether the Markov chain satisfies the three properties above depends on the jumping distribution.
  If the jumping distribution is such that the there is a positive probability that the Markov chain can jump to any state, then the Markov chain is irreducible, positive recurrent and aperiodic.
- Hence, if we choose the jumping distribution well, the Markov chain admits a unique stationary distribution.

Metropolis algorithm: why it works?

- Now we are left to show that the unique stationary distribution of the Markov chain generated by the Metropolis algorithm is the posterior distribution $p(\theta|\mathbf{y})$.
- Remember that a distribution π is a stationary distribution for the Markov chain $\{\theta^{(1)}, \theta^{(2)}, \ldots\}$ if the following happens:
  - if we sample a value $\theta^{(k)}$ from π and then we generate a value $\theta^{(k+1)}$ conditional on $\theta^{(k)}$ according to the one-step transition probabilities (in the case of a Metropolis algorithm, according to the jumping distribution $J_{k+1}(\theta^{\star}|\theta^{(k)})$), the marginal distribution of $\theta^{(k+1)}$ is the same as that of $\theta^{(k)}$, i.e. $\theta^{(k+1)} \sim \pi$.
- Hence, we need to show that if $\theta^{(k)} \sim p(\theta|\mathbf{y})$ and we generate $\theta^{(k+1)}$ using the Metropolis algorithm, then $\theta^{(k+1)} \sim p(\theta|\mathbf{y})$.

Metropolis algorithm: why it works?

- Now, let θ_a and θ_b be two points in the parameters space and suppose that $p(\theta_b|\mathbf{y}) > p(\theta_a|\mathbf{y})$.
  Then, we have:

$$\begin{aligned} P(\theta^{(k)} = \theta_a, \theta^{(k+1)} = \theta_b) &= P(\theta^{(k)} = \theta_a) \cdot P(\theta^{(k+1)} = \theta_b | \theta^{(k)} = \theta_a) \\ &= p(\theta_a|\mathbf{y}) \cdot J_{k+1}(\theta_b|\theta_a) \end{aligned}$$

  since the probability of acceptance here is 1.
  On the other hand:

$$\begin{aligned} P(\theta^{(k)} = \theta_b, \theta^{(k+1)} = \theta_a) &= P(\theta^{(k)} = \theta_b) \cdot P(\theta^{(k+1)} = \theta_a | \theta^{(k)} = \theta_b) \\ &= p(\theta_b|\mathbf{y}) \cdot J_{k+1}(\theta_a|\theta_b) \cdot \frac{p(\theta_a|\mathbf{y})}{p(\theta_b|\mathbf{y})} \\ &= J_{k+1}(\theta_a|\theta_b) p(\theta_a|\mathbf{y}) \\ &= J_{k+1}(\theta_b|\theta_a) p(\theta_a|\mathbf{y}) \\ &= P(\theta^{(k)} = \theta_a, \theta^{(k+1)} = \theta_b) \end{aligned}$$

Metropolis algorithm: why it works?

- Suppose now that $\theta^{(k)} \sim p(\theta|\mathbf{y})$.
- Since $P(\theta^{(k)} = \theta_a, \theta^{(k+1)} = \theta_b) = P(\theta^{(k)} = \theta_b, \theta^{(k+1)} = \theta_a)$, it follows that:

$$\begin{aligned} P(\theta^{(k+1)} = \theta_b) &= \sum_{\theta_a} P(\theta^{(k)} = \theta_a, \theta^{(k+1)} = \theta_b) \\ &= \sum_{\theta_a} P(\theta^{(k+1)} = \theta_a, \theta^{(k)} = \theta_b) \\ &= P(\theta^{(k)} = \theta_b) \\ &= p(\theta_b|\mathbf{y}) \end{aligned}$$

that is, the Markov chain generated by the Metropolis algorithm has the posterior distribution $p(\theta|\mathbf{y})$ as its stationary distribution.

Metropolis-Hastings algorithm

- The Metropolis-Hastings algorithm generalizes the basic Metropolis algorithm in two ways:
  1. there is no requirement that the jumping distribution $J_k(\theta^\star|\theta^{(k-1)})$ is symmetric. That is, we no longer require
  $$J_k(\theta_b|\theta_a) = J_k(\theta_a|\theta_b)$$
  2. to correct for asymmetry in the jumping distribution, the ratio r that is used to determine whether to accept or reject the proposed value $\theta^\star$ is replaced by the ratio
  $$r = \frac{\frac{p(\theta^\star|\mathbf{y})}{J_k(\theta^\star|\theta^{(k-1)})}}{\frac{p(\theta^{(k-1)}|\mathbf{y})}{J_k(\theta^{(k-1)}|\theta^\star)}} = \frac{p(\theta^\star|\mathbf{y}) \cdot J_k(\theta^{(k-1)}|\theta^\star)}{p(\theta^{(k-1)}|\mathbf{y}) \cdot J_k(\theta^\star|\theta^{(k-1)})}$$

  Note that the ratio r is always defined because a jump from $\theta^{(k-1)}$ to $\theta^\star$ can only occur if both $p(\theta^{(k-1)}|\mathbf{y})$ and $J_k(\theta^\star|\theta^{(k-1)})$ are nonzero.

Metropolis-Hastings algorithm

- A Metropolis-Hastings algorithm (and thus, an asymmetric jumping proposal) can be useful to increase the speed at which the Markov chain converges to the target distribution, the posterior distribution $p(\theta|\mathbf{y})$.
- The fact that the Metropolis-Hastings algorithm does indeed generate an irreducible and ergodic Markov chain with the posterior distribution $p(\theta|\mathbf{y})$ as its unique stationary distribution can be proved exactly as we did for the Metropolis algorithm (see book for a proof).
- As for the Metropolis algorithm, the jumping/proposal distribution $J_k(\theta^\star|\theta^{(k-1)})$ is very important for the efficiency of the algorithm.
- Ideally, the proposal distribution would be the posterior distribution $p(\theta^\star|\mathbf{y})$. In that case, the ratio r is always 1 and the Markov chain is a sequence of values sampled independently from the posterior distribution.

Jumping distribution for Metropolis-Hastings

- It is possible to choose the jumping distribution $J_k(\theta^\star|\theta^{(k-1)})$ to be independent of $\theta^{(k-1)}$. That is:

$$J_k(\theta^\star|\theta^{(k-1)}) = J_k(\theta^\star)$$

  In this case, the Metropolis-Hastings algorithm is called an independent Metropolis-Hastings algorithm
- The most common choice for the jumping/proposal distribution $J_k(\theta^\star|\theta^{(k-1)})$ is to have

$$\theta^\star = \theta^{(k-1)} + \varepsilon \qquad \varepsilon \sim q_k$$

  where q_k is some distribution.
- If $\theta^\star = \theta^{(k-1)} + \varepsilon$ with $\varepsilon \sim q_k$, then the Metropolis-Hastings algorithm is called a random walk Metropolis-Hastings algorithm.
- If q_k is a symmetric distribution (for example a Normal distribution $q_k = N(0, \delta^2)$ or $q_k = \text{Uniform}(-\delta, \delta)$), then we would have again a Metropolis algorithm. Otherwise, we have a Metropolis-Hastings algorithm.

Jumping distribution for Metropolis-Hastings

- Note that the Gibbs sampling algorithm can be interpreted as a special case of a Metroplis-Hastings algorithm where the proposal distribution is the full conditional distribution. For that choice of the jumping distribution, the ratio r is always equal to 1 and thus every proposed value is always accepted.
- In general, the same guidelines that are used to choose a good jumping distribution for the Metropolis algorithm apply for the Metropolis-Hastings algorithm.

Gibbs sampler + Metropolis algorithm

- The Gibbs sampler and the Metropolis algorithm can be used as building blocks for approximating joint posterior distributions and performing Bayesian inference.
- The Gibbs sampler is the simplest of the Markov chain simulation algorithms and should be the first choice for conditionally conjugate models, where we can sample directly from each full conditional distribution.
- The Metropolis algorithm can be used for models that are not conditionally conjugate, for example the Poisson regression model.

Gibbs sampler + Metropolis algorithm

- The Gibbs sampler and the Metropolis algorithm can be combined together. If the parameter on which we are making inference is a multidimensional vector $\theta = (\theta_1, \ldots, \theta_p)$ and some of the full conditionals have closed forms while some have not, then we can proceed as follows:
  - sample the parameters for which we can sample directly from the full conditional using a Gibbs sampler
  - update the parameters for the full conditional is not available in closed form using a Metropolis or Metropolis-Hastings algorithm
- More generally, the parameters can be updated in blocks using the Gibbs sampler (sometimes this is called block Gibbs-sampling) or using the Metropolis/Metropolis-Hastings algorithm.
- A general problem with Markov chain algorithms is that they can be slow to converge to the posterior distribution especially when the parameters are highly correlated.

Strategies for posterior simulation

- Good initial values: Start the MCMC algorithm with crude estimates for the parameters.
- Sequential order of updates: If possible, update parameters sequentially, starting with the hyperparameters (or parameters in the higher stages of the hierarchy) and then moving to the main parameters.
- Gibbs sampling: For parameters whose full conditional distributions have standard forms, use the Gibbs sampling algorithm.
- Metropolis/Metropolis-Hastings algorithm: For parameters whose full conditional distributions do not have standard forms, use the Metropolis/Metropolis-Hastings algorithm tuning the jumping distribution to have an acceptance rate within a $20-50\%$ range.
- Transformation: if possible, re-parametrize the model so to have independent parameters. That speeds up convergence.

Strategies for posterior simulation

- Multiple chains: run the MCMC algorithm multiple times using different initial values to assess convergence to the stationary distribution.
- Check for stationarity: check that the Markov chain has reached its stationary distribution.
  If multiple chains have been run, compute Gelman-Rubin's R and check that it is close to 1.
  If only one chain has been run, you can check that the distribution in the samples is the same within batches of MCMC samples (after discarding samples in the burnin period).
- Compare with frequentist results: compare the posterior inference from MCMC algorithms with estimates obtained using classical statistical methods. If the location of the marginal posterior distributions are not close to the frequentist estimates, check your code for errors before believing the result from the MCMC.