An Introduction to Bayesian Statistics
Day 2

David Carlson

March 18, 2019

A Motivating Example for Generalized Linear Models

- A sample from of a population of 52 female song sparrows was studied over the course of a summer and their reproductive activities were recorded

A Motivating Example for Generalized Linear Models

- A sample from of a population of 52 female song sparrows was studied over the course of a summer and their reproductive activities were recorded
  - The age and number of new offspring were recorded for each sparrow

A Motivating Example for Generalized Linear Models

- A sample from of a population of 52 female song sparrows was studied over the course of a summer and their reproductive activities were recorded
  - The age and number of new offspring were recorded for each sparrow
- Goal

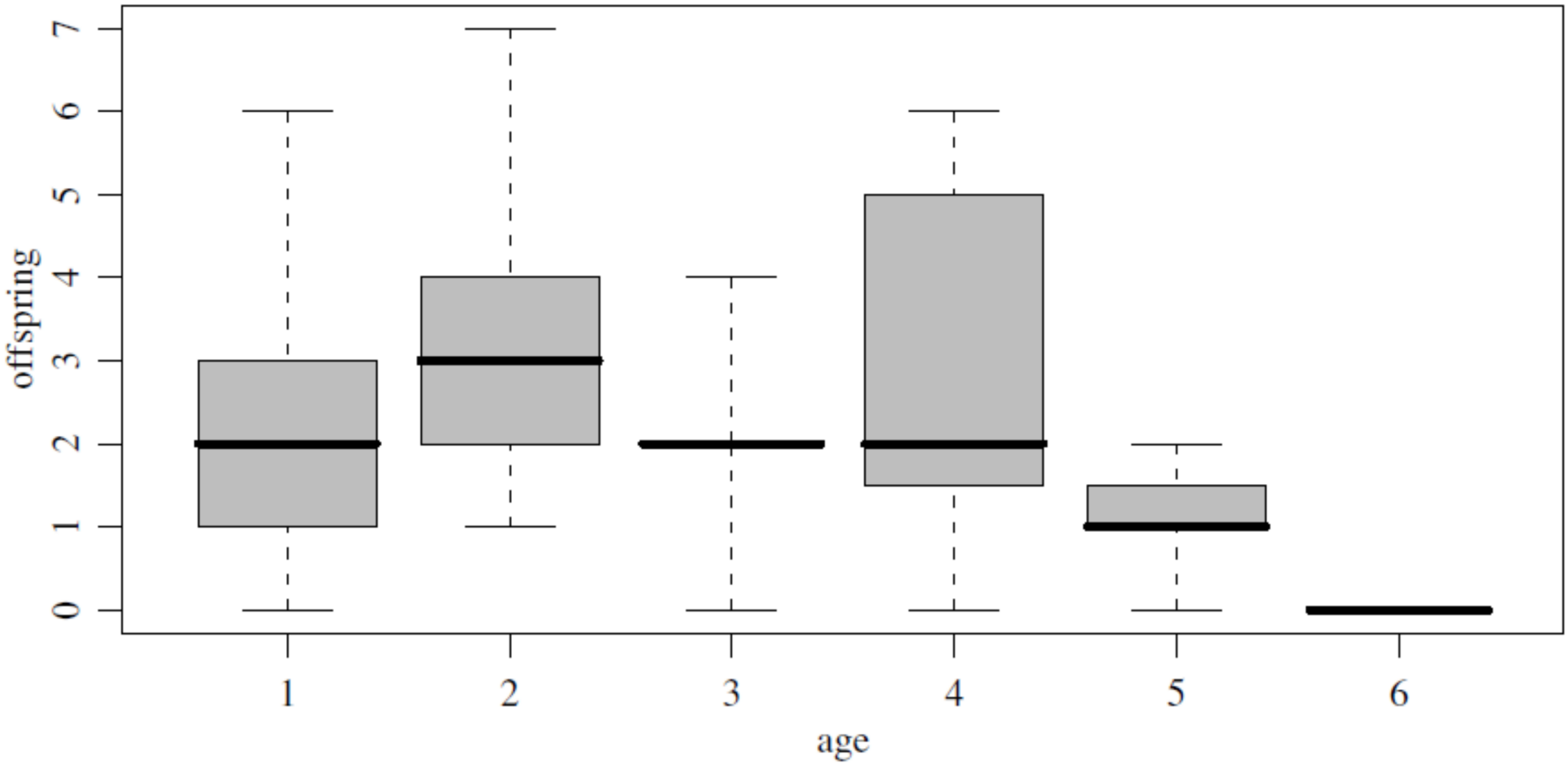
A Motivating Example for Generalized Linear Models

- A sample from of a population of 52 female song sparrows was studied over the course of a summer and their reproductive activities were recorded
  - The age and number of new offspring were recorded for each sparrow
- Goal
  - Understand the relationship between age and reproductive success

A Motivating Example for Generalized Linear Models

- A sample from of a population of 52 female song sparrows was studied over the course of a summer and their reproductive activities were recorded
  - The age and number of new offspring were recorded for each sparrow
- Goal
  - Understand the relationship between age and reproductive success
  - Make population forecasts for this group of birds

The Data



Poisson Regression

- Let y = number of offspring, x = age

Poisson Regression

- Let y = number of offspring, x = age
- $y|x \sim Poisson(\theta_x), \theta_x = E(y|x)$

Poisson Regression

- Let y = number of offspring, x = age
- $y|x \sim Poisson(\theta_x), \theta_x = E(y|x)$
- Separate estimate of θ_x for each age group may be imprecise when the number of birds of the same age is small

Poisson Regression

- Let y = number of offspring, x = age
- $y|x \sim Poisson(\theta_x), \theta_x = E(y|x)$
- Separate estimate of θ_x for each age group may be imprecise when the number of birds of the same age is small
- We may assume $\theta_x = \beta_1 + \beta_2 x + \beta_3 x^2$

Poisson Regression

- Let y = number of offspring, x = age
- $y|x \sim Poisson(\theta_x), \theta_x = E(y|x)$
- Separate estimate of θ_x for each age group may be imprecise when the number of birds of the same age is small
- We may assume $\theta_x = \beta_1 + \beta_2 x + \beta_3 x^2$
  - But this may produce negative values of θ_x

Poisson Regression

- Let y = number of offspring, x = age
- $y|x \sim Poisson(\theta_x), \theta_x = E(y|x)$
- Separate estimate of θ_x for each age group may be imprecise when the number of birds of the same age is small
- We may assume $\theta_x = \beta_1 + \beta_2 x + \beta_3 x^2$
  - But this may produce negative values of θ_x
- A better model: $\log \theta_x = \beta_1 + \beta_2 x + \beta_3 x^2$

Poisson Regression

- Let y = number of offspring, x = age
- $y|x \sim Poisson(\theta_x), \theta_x = E(y|x)$
- Separate estimate of θ_x for each age group may be imprecise when the number of birds of the same age is small
- We may assume $\theta_x = \beta_1 + \beta_2 x + \beta_3 x^2$
  - But this may produce negative values of θ_x
- A better model: $\log \theta_x = \beta_1 + \beta_2 x + \beta_3 x^2$
  - i.e. $\theta_x = \exp(\beta_1 + \beta_2 x + \beta_3 x^2)$

Poisson Regression

- Let y = number of offspring, x = age
- $y|x \sim Poisson(\theta_x), \theta_x = E(y|x)$
- Separate estimate of θ_x for each age group may be imprecise when the number of birds of the same age is small
- We may assume $\theta_x = \beta_1 + \beta_2 x + \beta_3 x^2$
  - But this may produce negative values of θ_x
- A better model: $\log \theta_x = \beta_1 + \beta_2 x + \beta_3 x^2$
  - i.e. $\theta_x = \exp(\beta_1 + \beta_2 x + \beta_3 x^2)$
- Poisson regression: $y|x \sim Poisson(\exp(\boldsymbol{\beta}^T \boldsymbol{x}))$

Generalized Linear Model

- Logistic regression

Generalized Linear Model

- Logistic regression
  - $y|x \sim Bernoulli(\theta_x), \theta_x = P(y = 1|x) = E(y|x)$

Generalized Linear Model

- Logistic regression
  - $y|x \sim Bernoulli(\theta_x), \theta_x = P(y = 1|x) = E(y|x)$
  - $\text{logit}(\theta_x) = \log \frac{\theta_x}{1-\theta_x} = \boldsymbol{\beta}^T \boldsymbol{x}$

Generalized Linear Model

- Logistic regression
  - $y|x \sim Bernoulli(\theta_x), \theta_x = P(y = 1|x) = E(y|x)$
  - $\text{logit}(\theta_x) = \log \frac{\theta_x}{1-\theta_x} = \boldsymbol{\beta}^T \boldsymbol{x}$
  - Link function: logit

Generalized Linear Model

- Logistic regression
  - $y|x \sim Bernoulli(\theta_x), \theta_x = P(y = 1|x) = E(y|x)$
  - $\text{logit}(\theta_x) = \log \frac{\theta_x}{1-\theta_x} = \boldsymbol{\beta}^T \boldsymbol{x}$
  - Link function: logit
- In general, $g(E(y|x)) = \boldsymbol{\beta}^T \boldsymbol{x}$

Generalized Linear Model

- Logistic regression
  - $y|x \sim Bernoulli(\theta_x), \theta_x = P(y = 1|x) = E(y|x)$
  - $\text{logit}(\theta_x) = \log \frac{\theta_x}{1-\theta_x} = \boldsymbol{\beta}^T \boldsymbol{x}$
  - Link function: logit
- In general, $g(E(y|x)) = \boldsymbol{\beta}^T \boldsymbol{x}$
  - $g()$: Link function

Generalized Linear Model

- Logistic regression
  - $y|x \sim Bernoulli(\theta_x), \theta_x = P(y = 1|x) = E(y|x)$
  - $\text{logit}(\theta_x) = \log \frac{\theta_x}{1-\theta_x} = \boldsymbol{\beta}^T \boldsymbol{x}$
  - Link function: logit
- In general, $g(E(y|x)) = \boldsymbol{\beta}^T \boldsymbol{x}$
  - $g()$: Link function
- No conjugate prior for GLM except identity link $\rightarrow$ we need a more general MCMC algorithm than Gibbs sampling

The Metropolis Algorithm

- When using simulation to describe the posterior $p(\theta|y)$, the goal is to construct a large collection of θ-values, $\{\theta^{(1)}, \ldots, \theta^{S}\}$, whose empirical distribution approximates $p(\theta|y)$

The Metropolis Algorithm

- When using simulation to describe the posterior $p(\theta|y)$, the goal is to construct a large collection of θ-values, $\{\theta^{(1)}, \ldots, \theta^{S}\}$, whose empirical distribution approximates $p(\theta|y)$
  - We need

$$\frac{\#\{\theta^{(s)}\text{'s in the collection } = \theta_a\}}{\#\{\theta^{(s)}\text{'s in the collection } = \theta_b\}} \approx \frac{p(\theta_a|y)}{p(\theta_b|y)}$$

The Metropolis Algorithm

- When using simulation to describe the posterior $p(\theta|y)$, the goal is to construct a large collection of θ-values, $\{\theta^{(1)}, \ldots, \theta^{S}\}$, whose empirical distribution approximates $p(\theta|y)$
  - We need

$$\frac{\#\{\theta^{(s)}\text{'s in the collection } = \theta_a\}}{\#\{\theta^{(s)}\text{'s in the collection } = \theta_b\}} \approx \frac{p(\theta_a|y)}{p(\theta_b|y)}$$

- How to construct $\{\theta^{(1)}, \ldots, \theta^{S}\}$

The Metropolis Algorithm

- When using simulation to describe the posterior $p(\theta|y)$, the goal is to construct a large collection of θ-values, $\{\theta^{(1)}, \ldots, \theta^{S}\}$, whose empirical distribution approximates $p(\theta|y)$
  - We need

$$\frac{\#\{\theta^{(s)}\text{'s in the collection } = \theta_a\}}{\#\{\theta^{(s)}\text{'s in the collection } = \theta_b\}} \approx \frac{p(\theta_a|y)}{p(\theta_b|y)}$$

- How to construct $\{\theta^{(1)}, \ldots, \theta^{S}\}$
  - Suppose we have $\{\theta^{(1)}, \ldots, \theta^{S}\}$ and want to add new $\theta^{(S+1)}$

The Metropolis Algorithm

- When using simulation to describe the posterior $p(\theta|y)$, the goal is to construct a large collection of θ-values, $\{\theta^{(1)}, \ldots, \theta^{S}\}$, whose empirical distribution approximates $p(\theta|y)$
  - We need

$$\frac{\#\{\theta^{(s)}\text{'s in the collection } = \theta_a\}}{\#\{\theta^{(s)}\text{'s in the collection } = \theta_b\}} \approx \frac{p(\theta_a|y)}{p(\theta_b|y)}$$

- How to construct $\{\theta^{(1)}, \ldots, \theta^{S}\}$
  - Suppose we have $\{\theta^{(1)}, \ldots, \theta^{S}\}$ and want to add new $\theta^{(S+1)}$
  - We may choose a candidate θ^* near $\theta^{(S)}$

The Metropolis Algorithm

- When using simulation to describe the posterior $p(\theta|y)$, the goal is to construct a large collection of θ-values, $\{\theta^{(1)}, \ldots, \theta^{S}\}$, whose empirical distribution approximates $p(\theta|y)$
  - We need

$$\frac{\#\{\theta^{(s)}\text{'s in the collection } = \theta_a\}}{\#\{\theta^{(s)}\text{'s in the collection } = \theta_b\}} \approx \frac{p(\theta_a|y)}{p(\theta_b|y)}$$

- How to construct $\{\theta^{(1)}, \ldots, \theta^{S}\}$
  - Suppose we have $\{\theta^{(1)}, \ldots, \theta^{S}\}$ and want to add new $\theta^{(S+1)}$
  - We may choose a candidate θ^* near $\theta^{(S)}$
  - Should we include θ^* into the collection?

The Metropolis Algorithm (cont.)

- If $p(\theta^*|y) > p(\theta^{(S)}|y)$ we clearly want θ^* in the collection

The Metropolis Algorithm (cont.)

- If $p(\theta^*|y) > p(\theta^{(S)}|y)$ we clearly want θ^* in the collection
- If $p(\theta^*|y) < p(\theta^{(S)}|y)$ we need to decide to keep it or not

The Metropolis Algorithm (cont.)

- If $p(\theta^*|y) > p(\theta^{(S)}|y)$ we clearly want θ^* in the collection
- If $p(\theta^*|y) < p(\theta^{(S)}|y)$ we need to decide to keep it or not
- Decision depends on ratio $r = \frac{p(\theta^*|y)}{p(\theta^{(S)}|y)}$

The Metropolis Algorithm (cont.)

- If $p(\theta^*|y) > p(\theta^{(S)}|y)$ we clearly want θ^* in the collection
- If $p(\theta^*|y) < p(\theta^{(S)}|y)$ we need to decide to keep it or not
- Decision depends on ratio $r = \frac{p(\theta^*|y)}{p(\theta^{(S)}|y)}$
- We can always calculate this ratio easily

$$r = \frac{p(\theta^*|y)}{p(\theta^{(S)}|y)} = \frac{p(y|\theta^*)p(\theta^*)}{p(y)} \frac{p(y)}{p(y|\theta^{(S)})p(\theta^{(S)})} = \frac{p(y|\theta^*)p(\theta^*)}{p(y|\theta^{(S)})p(\theta^{(S)})}$$

The Metropolis Algorithm (cont.)

- How to choose θ^*?

The Metropolis Algorithm (cont.)

- How to choose θ^*?
  - Sample θ^* from a proposal distribution $J(\theta^* | \theta^{(S)})$

The Metropolis Algorithm (cont.)

- How to choose θ^*?
  - Sample θ^* from a proposal distribution $J(\theta^*|\theta^{(S)})$
  - The Metropolis algorithm uses a symmetric proposal distribution $(J(\theta_a|\theta_b) = J(\theta_b|\theta_a))$

The Metropolis Algorithm (cont.)

- How to choose θ^*?
  - Sample θ^* from a proposal distribution $J(\theta^*|\theta^{(S)})$
  - The Metropolis algorithm uses a symmetric proposal distribution $(J(\theta_a|\theta_b) = J(\theta_b|\theta_a))$
  - Example: $\theta^*|\theta^{(S)} \sim U(\theta^{(S)} - \delta, \theta^{(S)} + \delta)$ or $\theta^*|\theta^{(S)} \sim \mathcal{N}(\theta^{(S)}, \delta^2)$

The Metropolis Algorithm (cont.)

- 1) Sample $\theta^* \sim J(\theta|\theta^{(s)})$

The Metropolis Algorithm (cont.)

- 1) Sample $\theta^* \sim J(\theta|\theta^{(s)})$
- 2) Compute the acceptance ratio

$$r = \frac{p(\theta^*|y)}{p(\theta^{(s)}|y)} = \frac{p(y|\theta^*)p(\theta^*)}{p(y|\theta^{(s)}p(\theta^{(s)})}.$$

The Metropolis Algorithm (cont.)

- 1) Sample $\theta^* \sim J(\theta|\theta^{(s)})$
- 2) Compute the acceptance ratio

$$r = \frac{p(\theta^*|y)}{p(\theta^{(s)}|y)} = \frac{p(y|\theta^*)p(\theta^*)}{p(y|\theta^{(s)}p(\theta^{(s)})}.$$

- 3) Let

$$\theta^{(s+1)} = \begin{cases} \theta^* & \text{with probability } \min(r,1) \\ \theta^{(s)} & \text{with probability } 1-\min(r,1) \end{cases}$$

The Metropolis Algorithm (cont.)

- 1) Sample $\theta^* \sim J(\theta|\theta^{(s)})$
- 2) Compute the acceptance ratio

$$r = \frac{p(\theta^*|y)}{p(\theta^{(s)}|y)} = \frac{p(y|\theta^*)p(\theta^*)}{p(y|\theta^{(s)}p(\theta^{(s)})}.$$

- 3) Let

$$\theta^{(s+1)} = \begin{cases} \theta^* & \text{with probability } \min(r,1) \\ \theta^{(s)} & \text{with probability } 1-\min(r,1) \end{cases}$$

- Step 3 can be accomplished by sampling $u \sim \text{uniform}(0,1)$ and setting $\theta^{(s+1)} = \theta^*$ if $u < r$ and setting $\theta^{(s+1)} = \theta^{(s)}$ otherwise

The Metropolis Algorithm (cont.)

- 1) Sample $\theta^* \sim J(\theta|\theta^{(s)})$
- 2) Compute the acceptance ratio

$$r = \frac{p(\theta^*|y)}{p(\theta^{(s)}|y)} = \frac{p(y|\theta^*)p(\theta^*)}{p(y|\theta^{(s)}p(\theta^{(s)})}.$$

- 3) Let

$$\theta^{(s+1)} = \begin{cases} \theta^* & \text{with probability } \min(r,1) \\ \theta^{(s)} & \text{with probability } 1-\min(r,1) \end{cases}$$

- Step 3 can be accomplished by sampling $u \sim \text{uniform}(0,1)$ and setting $\theta^{(s+1)} = \theta^*$ if $u < r$ and setting $\theta^{(s+1)} = \theta^{(s)}$ otherwise
- Output: A Markov chain $\{\theta^{(1)}, \ldots, \theta^{(S)}\}$

Example: Normal Model with Known Variance

- Data: $y_1, \ldots, y_n | \theta \sim \mathcal{N}(\theta, \sigma^2)$

Example: Normal Model with Known Variance

- Data: $y_1, \ldots, y_n | \theta \sim \mathcal{N}(\theta, \sigma^2)$
- Prior: $\theta \sim \mathcal{N}(\mu, \tau^2)$

Example: Normal Model with Known Variance

- Data: $y_1, \ldots, y_n | \theta \sim \mathcal{N}(\theta, \sigma^2)$
- Prior: $\theta \sim \mathcal{N}(\mu, \tau^2)$
- Suppose that
  $\sigma^2 = 1, \tau^2 = 10, \mu = 5, n = 5, y = (9.37, 10.18, 9.16, 11.60, 10.33)$

Example: Normal Model with Known Variance

- Data: $y_1, \ldots, y_n | \theta \sim \mathcal{N}(\theta, \sigma^2)$
- Prior: $\theta \sim \mathcal{N}(\mu, \tau^2)$
- Suppose that
  $\sigma^2 = 1, \tau^2 = 10, \mu = 5, n = 5, y = (9.37, 10.18, 9.16, 11.60, 10.33)$
- From what we learned before, we can show that $\theta | y \sim \mathcal{N}(10.03, 0.44)$

Example: Normal Model with Known Variance

- Data: $y_1, \ldots, y_n | \theta \sim \mathcal{N}(\theta, \sigma^2)$
- Prior: $\theta \sim \mathcal{N}(\mu, \tau^2)$
- Suppose that
  $\sigma^2 = 1, \tau^2 = 10, \mu = 5, n = 5, y = (9.37, 10.18, 9.16, 11.60, 10.33)$
- From what we learned before, we can show that $\theta | y \sim \mathcal{N}(10.03, 0.44)$
- Let's sample from the posterior distribution using Metropolis algorithm and compare to exact solution

Example: Normal Model with Known Variance (cont.)

- The acceptance ratio is

$$r = \frac{p(\theta^*|\boldsymbol{y})}{p(\theta^{(s)}|\boldsymbol{y}} = \left(\frac{\prod_{i=1}^n \Phi(y_i, \theta^*, \sigma)}{\prod_{i=1}^n \Phi(y_i, \theta^{(s)}, \sigma)}\right) \times \left(\frac{\Phi(\theta^*, \mu, \tau)}{\Phi(\theta^{(s)}, \mu, \tau)}\right)$$

Example: Normal Model with Known Variance (cont.)

- The acceptance ratio is

$$r = \frac{p(\theta^*|\boldsymbol{y})}{p(\theta^{(s)}|\boldsymbol{y}} = \left(\frac{\prod_{i=1}^n \Phi(y_i, \theta^*, \sigma)}{\prod_{i=1}^n \Phi(y_i, \theta^{(s)}, \sigma)}\right) \times \left(\frac{\Phi(\theta^*, \mu, \tau)}{\Phi(\theta^{(s)}, \mu, \tau)}\right)$$

- Often, the above calculation is numerically unstable and may compute logarithm instead

$$\log r = \sum_{i=1}^n [\log \Phi(y_i, \theta^*, \sigma) - \log \Phi(y_i, \theta^{(s)}, \sigma)] + [\log \Phi(\theta^*, \mu, \tau) - \log \Phi(\theta^{(s)}, \mu, \tau)].$$

Example: Normal Model with Known Variance (cont.)

- The acceptance ratio is

$$r = \frac{p(\theta^*|\boldsymbol{y})}{p(\theta^{(s)}|\boldsymbol{y}} = \left(\frac{\prod_{i=1}^n \Phi(y_i, \theta^*, \sigma)}{\prod_{i=1}^n \Phi(y_i, \theta^{(s)}, \sigma)}\right) \times \left(\frac{\Phi(\theta^*, \mu, \tau)}{\Phi(\theta^{(s)}, \mu, \tau)}\right)$$

- Often, the above calculation is numerically unstable and may compute logarithm instead

$$\log r = \sum_{i=1}^n [\log \Phi(y_i, \theta^*, \sigma) - \log \Phi(y_i, \theta^{(s)}, \sigma)] + [\log \Phi(\theta^*, \mu, \tau) - \log \Phi(\theta^{(s)}, \mu, \tau)].$$

- On the log scale, proposal is accepted if $\log U(0,1) < \log r$

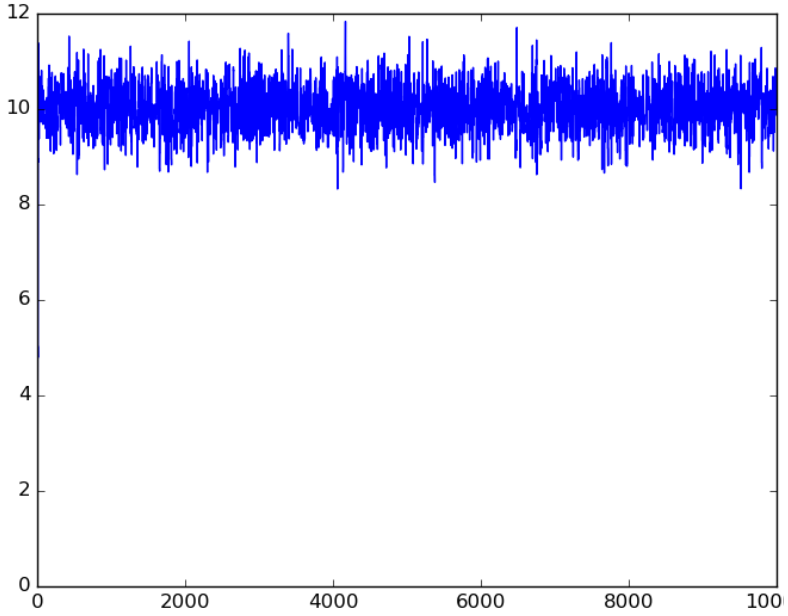
Python Code

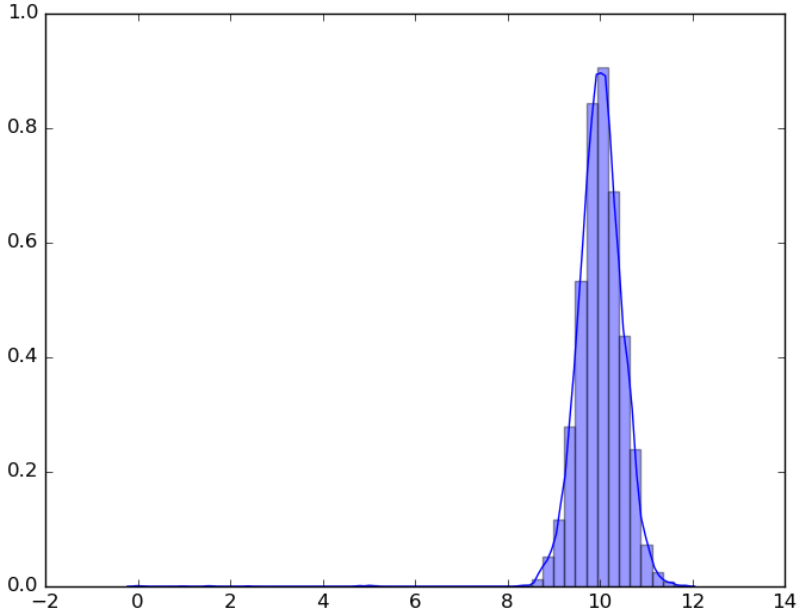
```
import numpy as np
import scipy.stats as stats
s2 = 1
t2 = 10
mu = 5
y = np.array([9.37, 10.18, 9.16, 11.60, 10.33])
theta = 0
delta2 = 2
S = 10000
THETA = []
```

Python Code (cont.)

```
for s in range(S):
    theta_star = np.random.normal(theta, np.sqrt(delta2), 1)
    log_r =
     (np.sum(np.log(stats.norm(theta_star, np.sqrt(s2)).pdf(y))) -
      np.sum(np.log(stats.norm(theta, np.sqrt(s2)).pdf(y)))) +
      (np.log(stats.norm(mu, np.sqrt(t2)).pdf(theta_star)) -
       np.log(stats.norm(mu, np.sqrt(t2)).pdf(theta)))
    if np.log(np.random.random(1)) < log_r:
        theta = theta_star

    THETA.append(theta)

import matplotlib as mpl
import matplotlib.pyplot as plt
plt.style.use('classic')
plt.plot(range(S), THETA)

import seaborn as sns
sns.distplot(THETA, hist = True)
```

Results


12
10
8
6
4
2
0
0
2000
4000
6000
8000
100


1.0
0.8
0.6
0.4
0.2
0.0
−2
0
2
4
6
8
10
12
14

General Property of the Metropolis Algorithm

- Under some mild conditions, the marginal distribution of $\theta^{(S)}$ approximates the posterior distribution $p(\theta|y)$ for large S.

General Property of the Metropolis Algorithm

- Under some mild conditions, the marginal distribution of $\theta^{(S)}$ approximates the posterior distribution $p(\theta|y)$ for large S.
  - For any given value of θ_a,

$$\lim_{S\to\infty} \frac{\#\{\theta\text{'s in the sequence } < \theta_a\}}{S} = p(\theta < \theta_a|y)$$

Tune the Proposal Distribution

- By choosing a proper proposal variance δ^2, we can decrease the correlation in the Markov chain

Tune the Proposal Distribution

- By choosing a proper proposal variance δ^2, we can decrease the correlation in the Markov chain
  - Faster convergence

Tune the Proposal Distribution

- By choosing a proper proposal variance δ^2, we can decrease the correlation in the Markov chain
  - Faster convergence
  - Better mixing

Tune the Proposal Distribution

- By choosing a proper proposal variance δ^2, we can decrease the correlation in the Markov chain
  - Faster convergence
  - Better mixing
  - An increase in the effective sample size

Tune the Proposal Distribution

- By choosing a proper proposal variance δ^2, we can decrease the correlation in the Markov chain
  - Faster convergence
  - Better mixing
  - An increase in the effective sample size
- Optimal proposal variance is neither too large nor too small

Tune the Proposal Distribution

- By choosing a proper proposal variance δ^2, we can decrease the correlation in the Markov chain
  - Faster convergence
  - Better mixing
  - An increase in the effective sample size
- Optimal proposal variance is neither too large nor too small
  - Small δ^2 : θ^* is too close to current value $\theta^{(s)}$. No matter accepted or not, $\theta^{(s+1)}$ is similar to $\theta^{(s)}$. Takes a long time to move away from $\theta^{(s)}$

Tune the Proposal Distribution

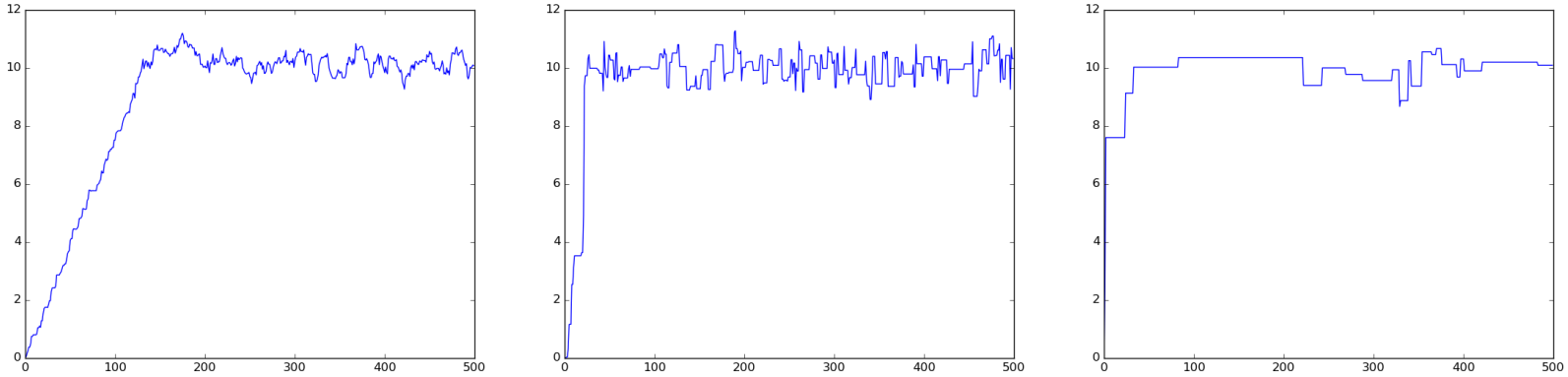
- By choosing a proper proposal variance δ^2, we can decrease the correlation in the Markov chain
  - Faster convergence
  - Better mixing
  - An increase in the effective sample size
- Optimal proposal variance is neither too large nor too small
  - Small δ^2 : θ^* is too close to current value $\theta^{(s)}$. No matter accepted or not, $\theta^{(s+1)}$ is similar to $\theta^{(s)}$. Takes a long time to move away from $\theta^{(s)}$
  - Large δ^2: If $\theta^{(s)}$ is close to posterior mode, θ^* is often very far from $\theta^{(s)}$ and leads to rejection. Then the chain gets stuck, because in most iterations $\theta^{(s+1)} = \theta^{(s)}$

Different Proposal Variances

- $\delta^2 = \left\{ \frac{1}{32}, \frac{1}{2}, 2, 32, 64 \right\}$

Different Proposal Variances

- $\delta^2 = \left\{\frac{1}{32}, \frac{1}{2}, 2, 32, 64\right\}$
- Corresponding autocorrelation: $(0.98, 0.77, \mathbf{0.69}, 0.84, 0.86)$



Metropolis Algorithm for Poisson Regression

- Model: $\log E(y_i|x_i) - \beta_1 + \beta_2 x_i + \beta_3 x_i^2$

Metropolis Algorithm for Poisson Regression

- Model: $\log E(y_i|x_i) - \beta_1 + \beta_2 x_i + \beta_3 x_i^2$
- Let $\boldsymbol{x}_i = (1, x_i, x_i^2)$, then $\log E(y_i|x_i) = \boldsymbol{\beta}'\boldsymbol{x}_i$

Metropolis Algorithm for Poisson Regression

- Model: $\log E(y_i|x_i) - \beta_1 + \beta_2 x_i + \beta_3 x_i^2$
- Let $\boldsymbol{x}_i = (1, x_i, x_i^2)$, then $\log E(y_i|x_i) = \boldsymbol{\beta}' \boldsymbol{x}_i$
- Prior distribution: $\boldsymbol{\beta} \sim \mathcal{N}(0, 100\boldsymbol{I})$

Metropolis Algorithm for Poisson Regression

- Model: $\log E(y_i|x_i) - \beta_1 + \beta_2 x_i + \beta_3 x_i^2$
- Let $\boldsymbol{x}_i = (1, x_i, x_i^2)$, then $\log E(y_i|x_i) = \boldsymbol{\beta}'\boldsymbol{x}_i$
- Prior distribution: $\boldsymbol{\beta} \sim \mathcal{N}(0, 100\boldsymbol{I})$
- Acceptance rate:

$$
\begin{aligned}
r &= \frac{p(\boldsymbol{\beta}^*|\boldsymbol{X}, \boldsymbol{y})}{p(\boldsymbol{\beta}^{(s)}|\boldsymbol{X}, \boldsymbol{y})} \\
&= \frac{\prod_{i=1}^n \mathcal{P}(y_i, \boldsymbol{x}_i'\boldsymbol{\beta}^*)}{\prod_{i=1}^n \mathcal{P}(y_i, \boldsymbol{x}_i'\boldsymbol{\beta}^{(s)})} \times \frac{\prod_{j=1}^3 \Phi(\beta_j^*, 0, 10)}{\prod_{j=1}^3 \Phi(\beta_j^{(s)}, 0, 10)}
\end{aligned}
$$

Metropolis Algorithm for Poisson Regression

- Model: $\log E(y_i|x_i) - \beta_1 + \beta_2 x_i + \beta_3 x_i^2$
- Let $\boldsymbol{x}_i = (1, x_i, x_i^2)$, then $\log E(y_i|x_i) = \boldsymbol{\beta}'\boldsymbol{x}_i$
- Prior distribution: $\boldsymbol{\beta} \sim \mathcal{N}(0, 100\boldsymbol{I})$
- Acceptance rate:

$$
\begin{aligned}
r &= \frac{p(\boldsymbol{\beta}^*|\boldsymbol{X}, \boldsymbol{y})}{p(\boldsymbol{\beta}^{(s)}|\boldsymbol{X}, \boldsymbol{y})} \\
&= \frac{\prod_{i=1}^n \mathcal{P}(y_i, \boldsymbol{x}_i'\boldsymbol{\beta}^*)}{\prod_{i=1}^n \mathcal{P}(y_i, \boldsymbol{x}_i'\boldsymbol{\beta}^{(s)})} \times \frac{\prod_{j=1}^3 \Phi(\beta_j^*, 0, 10)}{\prod_{j=1}^3 \Phi(\beta_j^{(s)}, 0, 10)}
\end{aligned}
$$

- Proposal distribution: Multivariate normal

Metropolis Algorithm for Poisson Regression

- Model: $\log E(y_i|x_i) - \beta_1 + \beta_2 x_i + \beta_3 x_i^2$
- Let $\boldsymbol{x}_i = (1, x_i, x_i^2)$, then $\log E(y_i|x_i) = \boldsymbol{\beta}'\boldsymbol{x}_i$
- Prior distribution: $\boldsymbol{\beta} \sim \mathcal{N}(0, 100\boldsymbol{I})$
- Acceptance rate:

$$
\begin{aligned}
r &= \frac{p(\boldsymbol{\beta}^*|\boldsymbol{X}, \boldsymbol{y})}{p(\boldsymbol{\beta}^{(s)}|\boldsymbol{X}, \boldsymbol{y})} \\
&= \frac{\prod_{i=1}^n \mathcal{P}(y_i, \boldsymbol{x}_i'\boldsymbol{\beta}^*)}{\prod_{i=1}^n \mathcal{P}(y_i, \boldsymbol{x}_i'\boldsymbol{\beta}^{(s)})} \times \frac{\prod_{j=1}^3 \Phi(\beta_j^*, 0, 10)}{\prod_{j=1}^3 \Phi(\beta_j^{(s)}, 0, 10)}
\end{aligned}
$$

- Proposal distribution: Multivariate normal
  - Choose the proposal variance similar to posterior variance

Metropolis Algorithm for Poisson Regression

- Model: $\log E(y_i|x_i) - \beta_1 + \beta_2 x_i + \beta_3 x_i^2$
- Let $\boldsymbol{x}_i = (1, x_i, x_i^2)$, then $\log E(y_i|x_i) = \boldsymbol{\beta}'\boldsymbol{x}_i$
- Prior distribution: $\boldsymbol{\beta} \sim \mathcal{N}(0, 100\boldsymbol{I})$
- Acceptance rate:

$$
\begin{aligned}
r &= \frac{p(\boldsymbol{\beta}^*|\boldsymbol{X}, \boldsymbol{y})}{p(\boldsymbol{\beta}^{(s)}|\boldsymbol{X}, \boldsymbol{y})} \\
&= \frac{\prod_{i=1}^n \mathcal{P}(y_i, \boldsymbol{x}_i'\boldsymbol{\beta}^*)}{\prod_{i=1}^n \mathcal{P}(y_i, \boldsymbol{x}_i'\boldsymbol{\beta}^{(s)})} \times \frac{\prod_{j=1}^3 \Phi(\beta_j^*, 0, 10)}{\prod_{j=1}^3 \Phi(\beta_j^{(s)}, 0, 10)}
\end{aligned}
$$

- Proposal distribution: Multivariate normal
  - Choose the proposal variance similar to posterior variance
  - From OLS, we know that $var(\hat{\beta}) = \sigma^2(X'X)^{-1}$

Metropolis Algorithm for Poisson Regression

- Model: $\log E(y_i|x_i) - \beta_1 + \beta_2 x_i + \beta_3 x_i^2$
- Let $\boldsymbol{x}_i = (1, x_i, x_i^2)$, then $\log E(y_i|x_i) = \boldsymbol{\beta}'\boldsymbol{x}_i$
- Prior distribution: $\boldsymbol{\beta} \sim \mathcal{N}(0, 100\boldsymbol{I})$
- Acceptance rate:

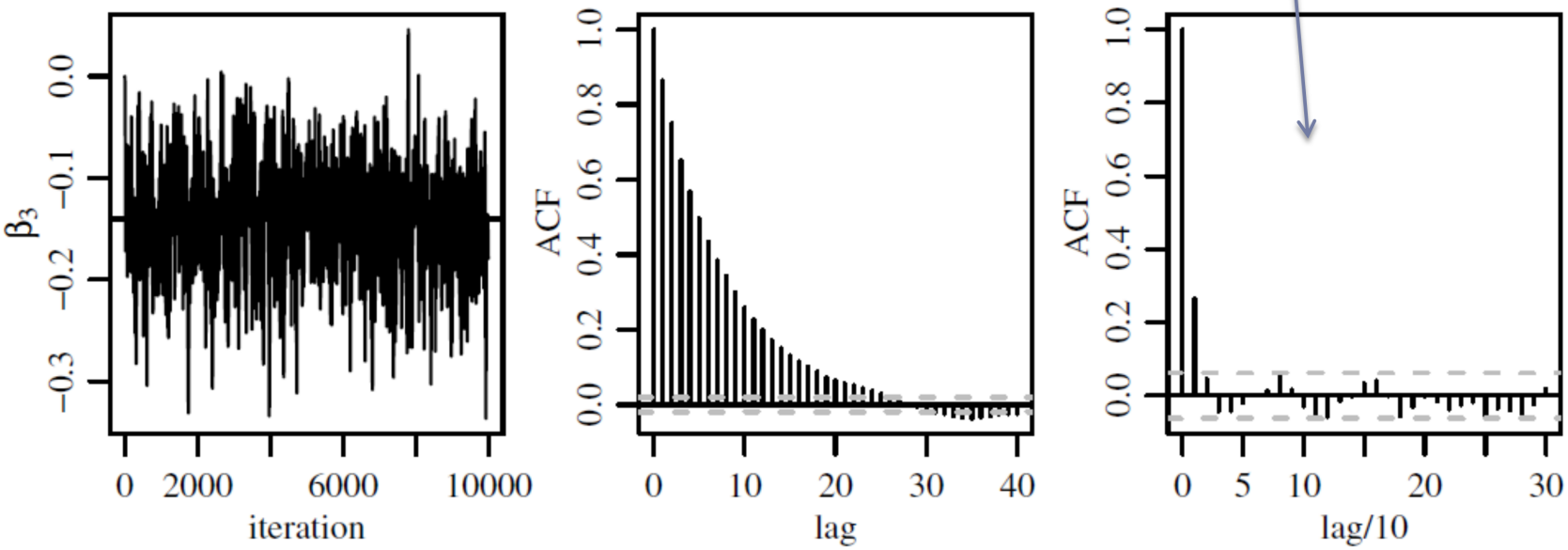
$$
\begin{aligned}
r &= \frac{p(\boldsymbol{\beta}^*|\boldsymbol{X}, \boldsymbol{y})}{p(\boldsymbol{\beta}^{(s)}|\boldsymbol{X}, \boldsymbol{y})} \\
&= \frac{\prod_{i=1}^n \mathcal{P}(y_i, \boldsymbol{x}_i'\boldsymbol{\beta}^*)}{\prod_{i=1}^n \mathcal{P}(y_i, \boldsymbol{x}_i'\boldsymbol{\beta}^{(s)})} \times \frac{\prod_{j=1}^3 \Phi(\beta_j^*, 0, 10)}{\prod_{j=1}^3 \Phi(\beta_j^{(s)}, 0, 10)}
\end{aligned}
$$

- Proposal distribution: Multivariate normal
  - Choose the proposal variance similar to posterior variance
  - From OLS, we know that $var(\hat{\beta}) = \sigma^2(X'X)^{-1}$
  - Construct $\hat{\sigma}^2$ using the sample variance of $\{\log(y_1 + \frac{1}{2}, \ldots, \log(y_n + \frac{1}{2})\}$

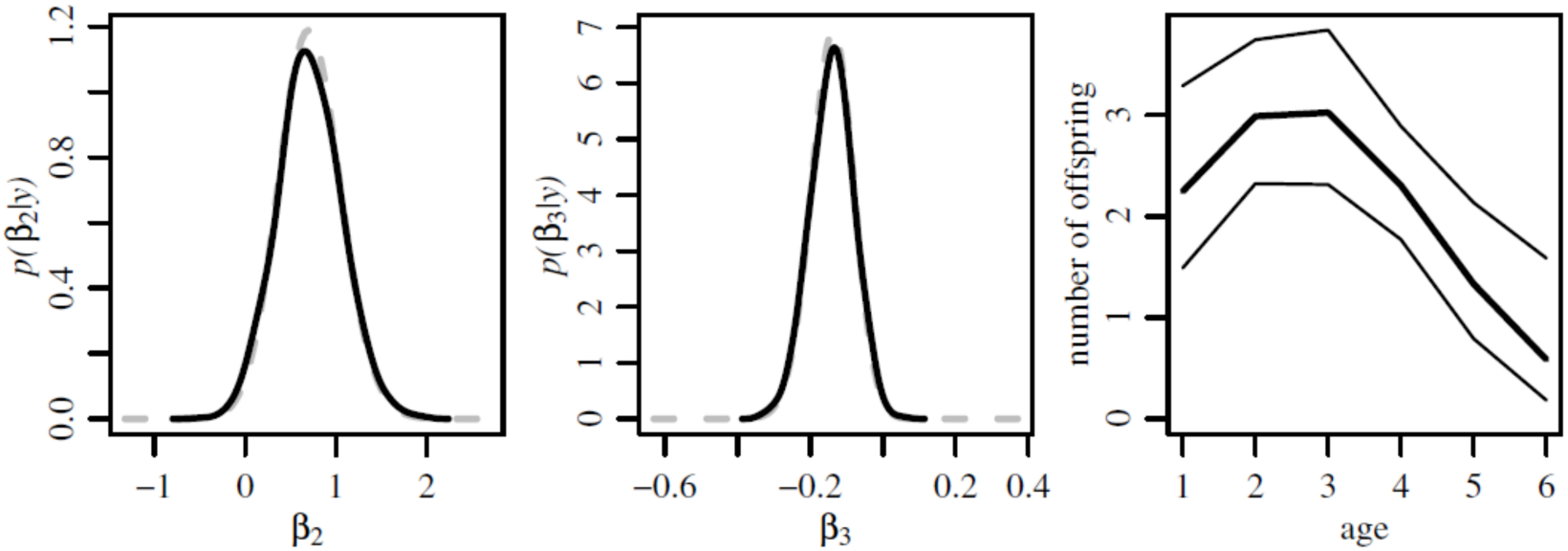
Result

- Acceptance rate: 43%

After thinning (every 10th observation)



Result


p(β2|y)
0.0
0.4
0.8
1.2
−1
0
1
2
β2
p(β3|y)
0
1
2
3
4
5
6
7
−0.6
−0.2
0.2
0.4
β3
number of offspring
0
1
2
3
1
2
3
4
5
6
age

The Metropolis-Hastings (M-H) Algorithm

- Consider to sample from a bivariate distribution $p_0(u, v)$
- Gibbs sampler

1. update U: sample $u^{(s+1)} \sim p_0(u|v^{(s)})$;
2. update V: sample $v^{(s+1)} \sim p_0(v|u^{(s+1)})$.

1. Proposal: full conditional distribution
2. Always accept

- The Metropolis algorithm

1. update U:
   a) sample $u^* \sim J_u(u|u^{(s)})$;
   b) compute $r = p_0(u^*, v^{(s)})/p_0(u^{(s)}, v^{(s)})$;
   c) set $u^{(s+1)}$ to u^* or $u^{(s)}$ with probability $\min(1, r)$ and $\max(0, 1 - r)$.
2. update V:
   a) sample $v^* \sim J_v(v|v^{(s)})$;
   b) compute $r = p_0(u^{(s+1)}, v^*)/p_0(u^{(s+1)}, v^{(s)})$;
   c) set $v^{(s+1)}$ to v^* or $v^{(s)}$ with probability $\min(1, r)$ and $\max(0, 1 - r)$.

Symmetric proposal distribution

The Metropolis-Hastings (M-H) Algorithm (cont.)

- Arbitrary proposal

1. update U:
   a) sample $u^* \sim J_u(u|u^{(s)}, v^{(s)})$;
   b) compute the acceptance ratio

$$r = \frac{p_0(u^*, v^{(s)})}{p_0(u^{(s)}, v^{(s)})} \times \frac{J_u(u^{(s)}|u^*, v^{(s)})}{J_u(u^*|u^{(s)}, v^{(s)})};$$

   c) set $u^{(s+1)}$ to u^* or $u^{(s)}$ with probability $\min(1, r)$ and $\max(0, 1-r)$.
2. update V:
   a) sample $v^* \sim J_v(v|u^{(s+1)}, v^{(s)})$;
   b) compute the acceptance ratio

$$r = \frac{p_0(u^{(s+1)}, v^*)}{p_0(u^{(s+1)}, v^{(s)})} \times \frac{J_v(v^{(s)}|u^{(s+1)}, v^*)}{J_v(v^*|u^{(s+1)}, v^{(s)})};$$

   c) set $v^{(s+1)}$ to v^* or $v^{(s)}$ with probability $\min(1, r)$ and $\max(0, 1-r)$.

Gibbs Sampler as the M-H Algorithm

- If the proposal distribution is $J_u(u^*|u^{(s)}, v^{(s)}) = p_0(u^*|v^{(s)})$ in the M-H algorithm, the acceptance ratio is

$$
\begin{aligned}
r &= \frac{p_0(u^*, v^{(s)})}{p_0 u^{(s)}, v^{(s)}} \times \frac{J_u(u^{(s)}|u^*, v^{(s)})}{J_u(u^*|u^{(s)}, v^{(s)})} \\
&= \frac{p_0(u^*, v^{(s)})}{p_0 u^{(s)}, v^{(s)}} \frac{p_0(u^{(s)}|v^{(s)})}{p_0(u^*|v^{(s)})} \\
&= \frac{p_0(u^*|v^{(s)}) p_0(v^{(s)})}{p_0(u^{(s)}|v^{(s)}) p_0(v^{(s)})} \frac{p_0(u^{(s)}|v^{(s)})}{p_0(u^*|v^{(s)})} \\
&= \frac{p_0(v^{(s)})}{p_0(v^{(s)})} = 1
\end{aligned}
$$

Requirement for the Proposal Distribution

- The proposal distribution does not depend on values in the sequence previous to $x^{(s)} \rightarrow$ Markov property

Requirement for the Proposal Distribution

- The proposal distribution does not depend on values in the sequence previous to $x^{(s)} \rightarrow$ Markov property
- Regardelss where the chain started, every value x with $p_0(x) > 0$ will eventually be proposed $\rightarrow$ irreducible

Requirement for the Proposal Distribution

- The proposal distribution does not depend on values in the sequence previous to $x^{(s)} \rightarrow$ Markov property
- Regardelss where the chain started, every value x with $p_0(x) > 0$ will eventually be proposed $\rightarrow$ irreducible
- Aperiodic: A Markov chain lacking any periodic states is called aperiodic

Requirement for the Proposal Distribution

- The proposal distribution does not depend on values in the sequence previous to $x^{(s)} \rightarrow$ Markov property
- Regardelss where the chain started, every value x with $p_0(x) > 0$ will eventually be proposed $\rightarrow$ irreducible
- Aperiodic: A Markov chain lacking any periodic states is called aperiodic
  - A value x is periodic with period $k > 1$ in a Markov chain if it can only be visited every kth iteration

Requirement for the Proposal Distribution

- The proposal distribution does not depend on values in the sequence previous to $x^{(s)} \rightarrow$ Markov property
- Regardelss where the chain started, every value x with $p_0(x) > 0$ will eventually be proposed $\rightarrow$ irreducible
- Aperiodic: A Markov chain lacking any periodic states is called aperiodic
  - A value x is periodic with period $k > 1$ in a Markov chain if it can only be visited every kth iteration
- Recurrent: A value x is said to be recurrent if, when we continue to run the Markov chain from x, we are guaranteed to eventually return to x. And we want all of the possible values of x to be recurrent in our Markov chain

Ergodic Theorem

- If $\{x^{(1)}, x^{(2)}, \ldots\}$ is an irreducible, aperiodic, and recurrent Markov chain, then there is a unique probability distribution π such that as $s \to \infty$,

$$Pr(x^{(s)} \in A) \to \pi(A) \text{ for any set } A;$$
$$\frac{1}{S}\sum g(x^{(s)}) \to \int g(x)\pi(x)dx.$$

Ergodic Theorem

- If $\{x^{(1)}, x^{(2)}, \ldots\}$ is an irreducible, aperiodic, and recurrent Markov chain, then there is a unique probability distribution π such that as $s \to \infty$,

$$Pr(x^{(s)} \in A) \to \pi(A) \text{ for any set } A;$$
$$\frac{1}{S}\sum g(x^{(s)}) \to \int g(x)\pi(x)dx.$$

- The distribution π is called the stationary distribution because it has the following property: If $x^{(s)} \sim \pi$, and $x^{(s+1)}$ is generated from the Markov chain starting at $x^{(s)}$, then $Pr(x^{(s+1)} \in A) = \pi(A)$.

Ergodic Theorem

- If $\{x^{(1)}, x^{(2)}, \ldots\}$ is an irreducible, aperiodic, and recurrent Markov chain, then there is a unique probability distribution π such that as $s \to \infty$,

$$Pr(x^{(s)} \in A) \to \pi(A) \text{ for any set } A;$$
$$\frac{1}{S}\sum g(x^{(s)}) \to \int g(x)\pi(x)dx.$$

- The distribution π is called the stationary distribution because it has the following property: If $x^{(s)} \sim \pi$, and $x^{(s+1)}$ is generated from the Markov chain starting at $x^{(s)}$, then $Pr(x^{(s+1)} \in A) = \pi(A)$.
- $\to$ Once you are sampling from the stationary distribution, you are always sampling from the stationary distribution

Combining the Metropolis and Gibbs Algorithms

- In complex models it is often the case that conditional distributions are available for some parameters but not others

Combining the Metropolis and Gibbs Algorithms

- In complex models it is often the case that conditional distributions are available for some parameters but not others
- In these situations we can combine Gibbs and Metropolis-type proposal distributions to generate a Markov chain to approximate the joint posterior distribution of all the parameters

A Regression Model with Correlated Errors

- Ordinary linear regression

$$\boldsymbol{Y} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix} \sim \text{multivariate normal}(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}).$$

- Introducing AR(1) structure

$$\Sigma = \sigma^2\mathbf{C}_\rho = \sigma^2 \begin{pmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{n-1} \\ \rho & 1 & \rho & \cdots & \rho^{n-2} \\ \rho^2 & \rho & 1 & & \\ \vdots & \vdots & & \ddots & \\ \rho^{n-1} & \rho^{n-2} & & & 1 \end{pmatrix}$$

MCMC

- Prior: $\beta \sim N(\beta_0, \Sigma_0)$ and $\sigma^{-2} \sim gamma(\frac{\nu_0}{2}, \frac{\nu_0 \sigma_0^2}{2})$
- If ρ is known, a Gibbs sampler is available

$$\{\boldsymbol{\beta}|\mathbf{X}, \boldsymbol{y}, \sigma^2, \rho\} \sim \text{multivariate normal}(\boldsymbol{\beta}_n, \Sigma_n) \ , \ \text{where}$$
$$\Sigma_n = (\mathbf{X}^T \mathbf{C}_\rho^{-1} \mathbf{X}/\sigma^2 + \Sigma_0^{-1})^{-1}$$
$$\boldsymbol{\beta}_n = \Sigma_n (\mathbf{X}^T \mathbf{C}_\rho^{-1} \boldsymbol{y}/\sigma^2 + \Sigma_0^{-1} \boldsymbol{\beta}_0) \ , \ \text{and}$$

$$\{\sigma^2|\mathbf{X}, \boldsymbol{y}, \boldsymbol{\beta}, \rho\} \sim \text{inverse-gamma}([\nu_0 + n]/2, [\nu_0 \sigma_0^2 + \text{SSR}_\rho]/2) \ , \ \text{where}$$
$$\text{SSR}_\rho = (\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^T \mathbf{C}_\rho^{-1} (\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}).$$

- But ρ is unknown and the full conditional distribution of ρ is not so simple

MCMC (cont.)

1. Update $\boldsymbol{\beta}$: Sample $\boldsymbol{\beta}^{(s+1)} \sim$ multivariate normal$(\boldsymbol{\beta}_n, \Sigma_n)$, where $\boldsymbol{\beta}_n$ and Σ_n depend on $\sigma^{2(s)}$ and $\rho^{(s)}$.
2. Update σ^2: Sample $\sigma^{2(s+1)} \sim$ inverse-gamma$([\nu_0+n]/2, [\nu_0\sigma_0^2+\mathrm{SSR}_\rho]/2)$, where SSR_ρ depends on $\boldsymbol{\beta}^{(s+1)}$ and $\rho^{(s)}$.
3. Update ρ:
   a) Propose $\rho^* \sim$ uniform$(\rho^{(s)} - \delta, \rho^{(s)} + \delta)$. If $\rho^* < 0$ then reassign it to be $|\rho^*|$. If $\rho^* > 1$ reassign it to be $2 - \rho^*$.
   b) Compute the acceptance ratio

$$r = \frac{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{\beta}^{(s+1)}, \sigma^{2(s+1)}, \rho^*)p(\rho^*)}{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{\beta}^{(s+1)}, \sigma^{2(s+1)}, \rho^{(s)})p(\rho^{(s)})}$$

   and sample $u \sim$ uniform(0,1). If $u < r$ set $\rho^{(s+1)} = \rho^*$, otherwise set $\rho^{(s+1)} = \rho^{(s)}$.

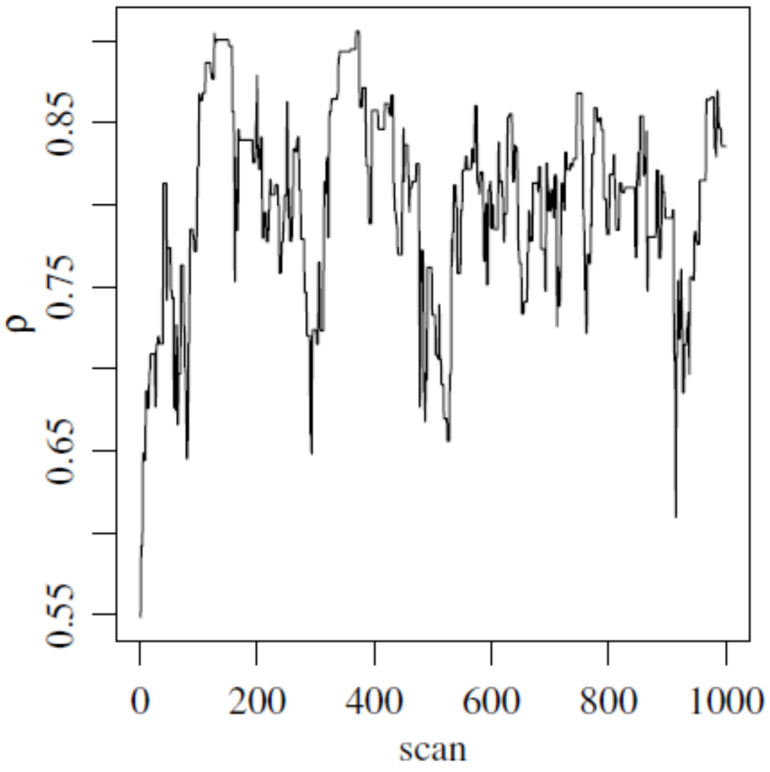
This proposal distribution is called reflecting random walk, which ensures $0 < \rho < 1$

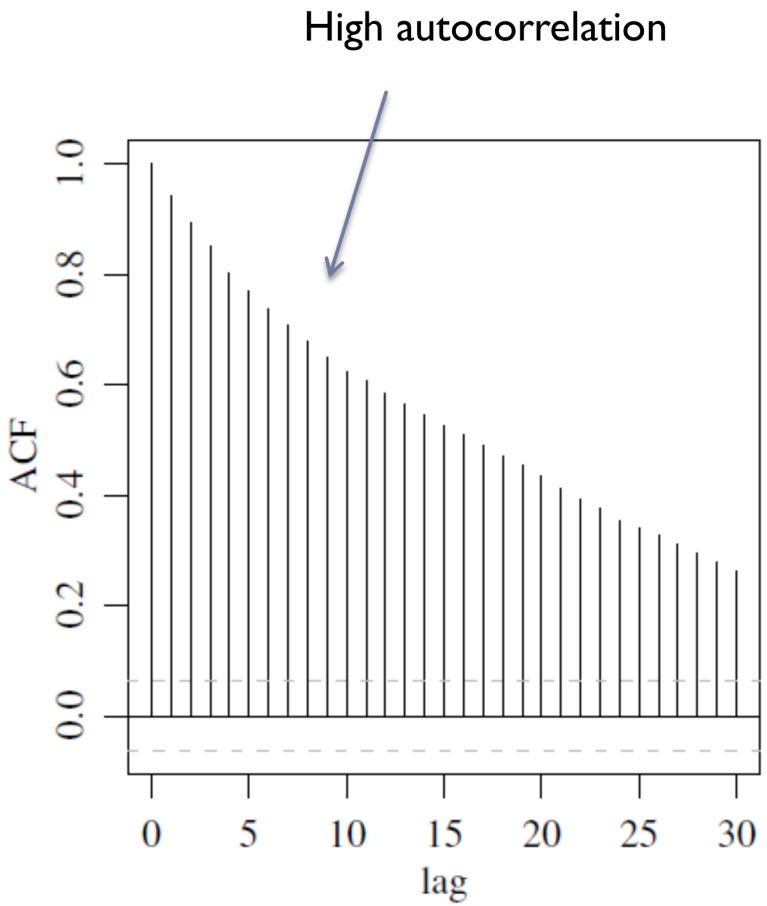
- It is symmetric

Example: Historical CO2 and Temperature Data

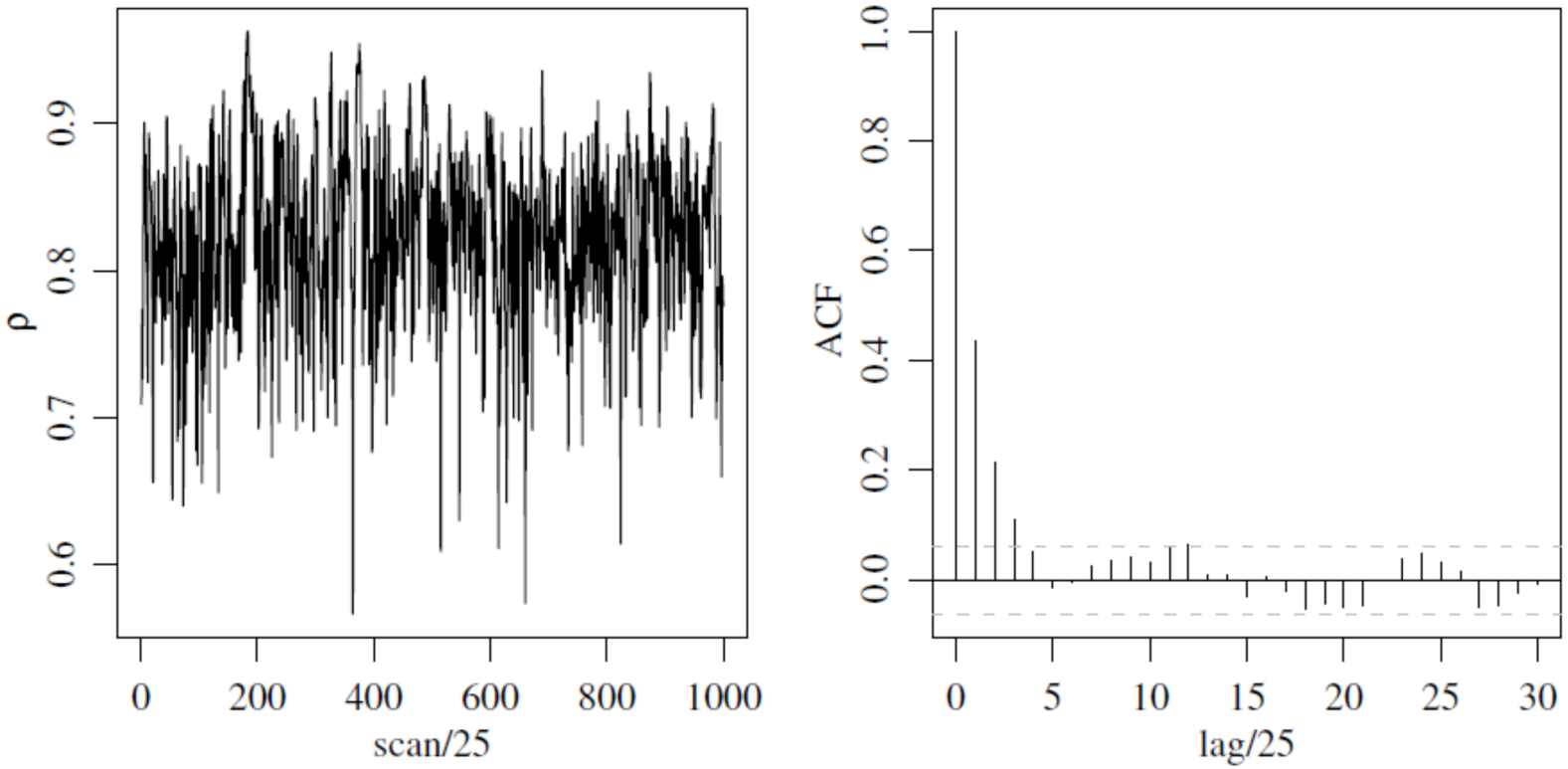
- "diffuse" prior:
  - $\beta \sim N(\beta_0, \Sigma_0)$ and $\sigma^{-2} \sim gamma(\frac{\nu_0}{2}, \frac{\nu_0 \sigma_0^2}{2})$ with $\beta_0 = 0, \Sigma_0 = diag(1000), \nu_0 = 1, \sigma_0^2 = 1$
  - $\rho \sim U(0,1)$

Before Thinning

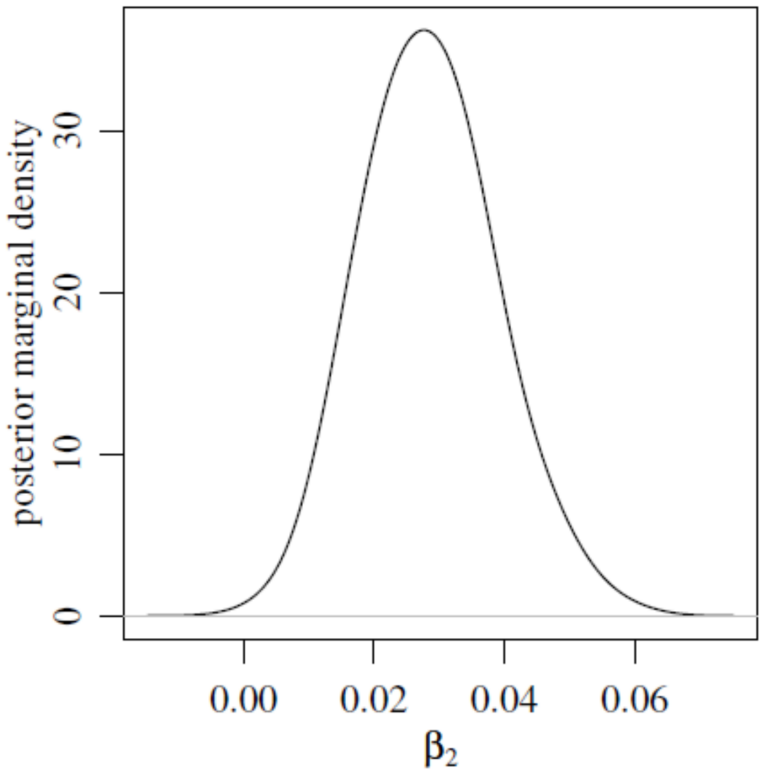


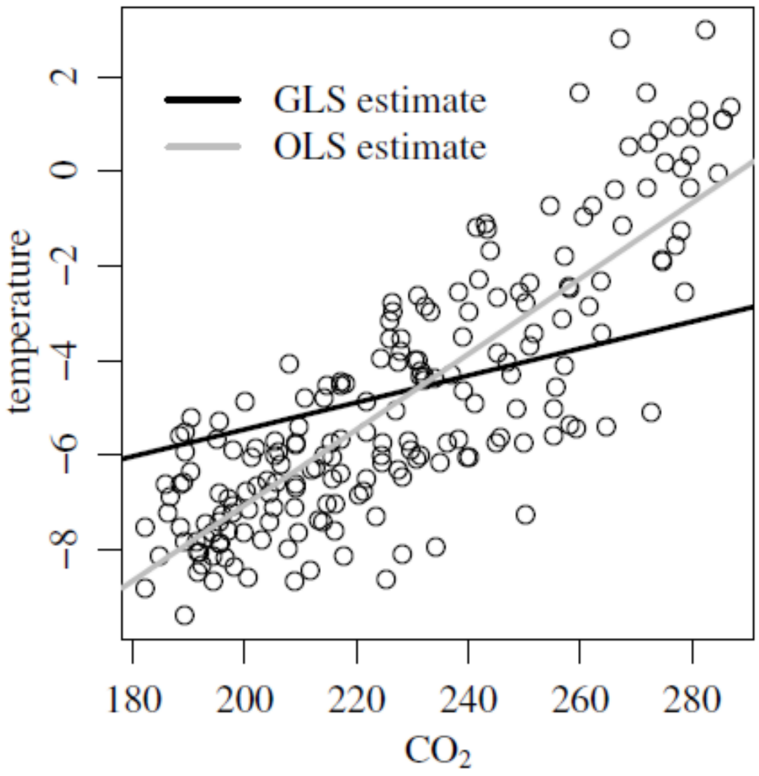


After Thinning (every 25th value)



Posterior


posterior marginal density
0
10
20
30
0.00
0.02
0.04
0.06
β_2


GLS estimate
OLS estimate
temperature
−8
−6
−4
−2
0
2
180
200
220
240
260
280
CO_2

Process the MCMC Output

- Burn-in

Process the MCMC Output

- Burn-in
- Thinning

Process the MCMC Output

- Burn-in
- Thinning
- Monitoring convergence

Process the MCMC Output

- Burn-in
- Thinning
- Monitoring convergence
- Tools

Process the MCMC Output

- Burn-in
- Thinning
- Monitoring convergence
- Tools
  - Traceplot

Process the MCMC Output

- Burn-in
- Thinning
- Monitoring convergence
- Tools
  - Traceplot
  - ACF plot

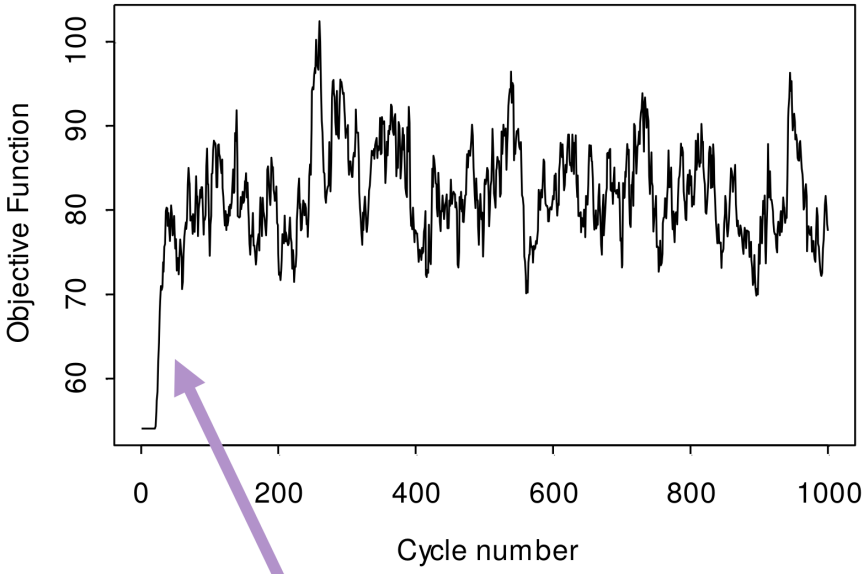
Process the MCMC Output

- Burn-in
- Thinning
- Monitoring convergence
- Tools
  - Traceplot
  - ACF plot
  - Acceptance (or rejection) rate for M-H algorithm

Process the MCMC Output

- Burn-in
- Thinning
- Monitoring convergence
- Tools
  - Traceplot
  - ACF plot
  - Acceptance (or rejection) rate for M-H algorithm
  - Quantitative procedures

Burn-In

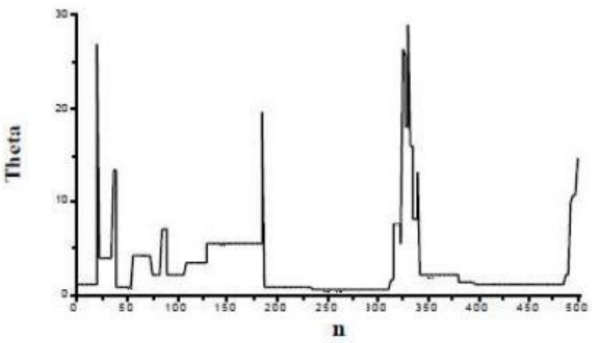


Need for a burn-in (the "pigtail")

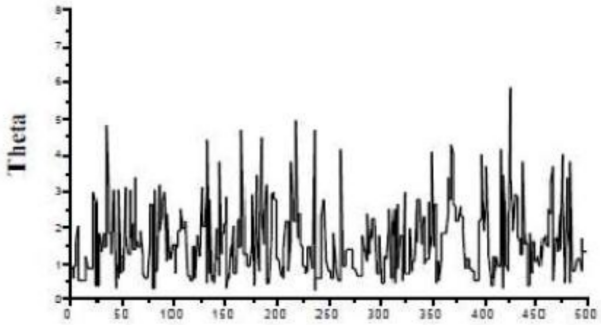
Mixing

- MCMC may have poor mixing when
  - autocorrelation is high
  - it takes a long time between jumps to different parts of the parameter space
  - Looking at the traceplot

- Poor mixing



- Well mixing



Gelman and Rubin (1992): Potential Scale Reduction Factor

- Based on normal theory approximation to exact Bayesian posterior inference

Gelman and Rubin (1992): Potential Scale Reduction Factor

- Based on normal theory approximation to exact Bayesian posterior inference
- Create an overdispersed estimate of the target distribution and use it to start several independent sequences

Gelman and Rubin (1992): Potential Scale Reduction Factor

- Based on normal theory approximation to exact Bayesian posterior inference
- Create an overdispersed estimate of the target distribution and use it to start several independent sequences
- For each scalar quantity of interest, after running the sampler chains estimate the factor by which the scale parameter might shrink if sampling were continued indefinitely

Gelman and Rubin (1992): Potential Scale Reduction Factor

- Based on normal theory approximation to exact Bayesian posterior inference
- Create an overdispersed estimate of the target distribution and use it to start several independent sequences
- For each scalar quantity of interest, after running the sampler chains estimate the factor by which the scale parameter might shrink if sampling were continued indefinitely
- If close to one, within-chain variance dominates between-chain variance

Gelman and Rubin (1992): Potential Scale Reduction Factor

- Based on normal theory approximation to exact Bayesian posterior inference
- Create an overdispersed estimate of the target distribution and use it to start several independent sequences
- For each scalar quantity of interest, after running the sampler chains estimate the factor by which the scale parameter might shrink if sampling were continued indefinitely
- If close to one, within-chain variance dominates between-chain variance
- This is called $\hat{R}$

About Convergence Diagnostics

- There are a lot of them

About Convergence Diagnostics

- There are a lot of them
- Can not prove the chain converges

About Convergence Diagnostics

- There are a lot of them
- Can not prove the chain converges
- Be cautious when using these diagnostics

About Convergence Diagnostics

- There are a lot of them
- Can not prove the chain converges
- Be cautious when using these diagnostics
- Use a variety of diagnosis tools rather than any single one

About Convergence Diagnostics

- There are a lot of them
- Can not prove the chain converges
- Be cautious when using these diagnostics
- Use a variety of diagnosis tools rather than any single one
- Learn as much as possible about the target density before applying the MCMC algorithm

Hamiltonian MC

- Physical analogy to Hamiltonian MC: Imagine a hockey pluck sliding over a surface without friction, being stopped at some point in time and then kicked again in a random direction

Hamiltonian MC

- Physical analogy to Hamiltonian MC: Imagine a hockey pluck sliding over a surface without friction, being stopped at some point in time and then kicked again in a random direction
- Hamiltonian MC employs the trick developed by nature. Velocity v is added to the parameters describing the system. Energy of the system consists of potential and kinetic parts

Hamiltonian MC

- Physical analogy to Hamiltonian MC: Imagine a hockey pluck sliding over a surface without friction, being stopped at some point in time and then kicked again in a random direction
- Hamiltonian MC employs the trick developed by nature. Velocity v is added to the parameters describing the system. Energy of the system consists of potential and kinetic parts
- velocities v and positions x have independent canonical distributions

Hamiltonian MC

- Physical analogy to Hamiltonian MC: Imagine a hockey pluck sliding over a surface without friction, being stopped at some point in time and then kicked again in a random direction
- Hamiltonian MC employs the trick developed by nature. Velocity v is added to the parameters describing the system. Energy of the system consists of potential and kinetic parts
- velocities v and positions x have independent canonical distributions
- So once we can sample from joint distribution $p(x, v)$, we also can sample x by ignoring computed velocities v

Pros and Cons of Hamiltonian MC

- Distances between successive generated points are typically large, so we need less iterations to get representative sampling

Pros and Cons of Hamiltonian MC

- Distances between successive generated points are typically large, so we need less iterations to get representative sampling
- price of a single iteration is higher, but HMC is still significantly more efficient

Pros and Cons of Hamiltonian MC

- Distances between successive generated points are typically large, so we need less iterations to get representative sampling
- price of a single iteration is higher, but HMC is still significantly more efficient
- Hamiltonian MC in most cases accepts new states

Pros and Cons of Hamiltonian MC

- Distances between successive generated points are typically large, so we need less iterations to get representative sampling
- price of a single iteration is higher, but HMC is still significantly more efficient
- Hamiltonian MC in most cases accepts new states
- HMC has problems with sampling from distributions with isolated local minimums puck doesn't have enough energy to jump from the first minimum to the second over the energy barrier