Lecture 10A
MTH6102: Bayesian Statistical Methods

Eftychia Solea

Queen Mary University of London

2023

Today's agenda

Today's lecture

- Review of the symmetric Metropolis-Hastings (MH)
- Understand implementation issues with MH.

Symmetric MH algorithm

Goal: Generate a Markov chain $\theta_1, \theta_2, \dots$ from the posterior $p(\theta \mid y)$.

Define $g(\theta) = p(\theta)\, p(y \mid \theta)$, the non-normalized posterior density/Bayes numerator.

- Start with θ_1, randomly such that $g(\theta_1) > 0$. For each $i > 1$:
  1. Generate $\psi \sim N(\theta_{i-1}, b^2)$, for some $b > 0$.
  2. Compute the probability of acceptance

$$r = \min\left(1, \frac{g(\psi)}{g(\theta_{i-1})}\right) = \min\left(1, \frac{p(\psi)p(y \mid \psi)}{p(\theta_{i-1})p(y \mid \theta_{i-1})}\right).$$

  3. Generate $U \sim U[0,1]$. Set

$$\theta_i = \begin{cases} \psi, & \text{if } U < r \\ \theta_{i-1}, & \text{otherwise} \end{cases}$$

$$r = \min\left(1, \frac{p(\psi|y)\, q(\theta_i|\cancel{\psi})}{p(\theta_i|y)\, q(\cancel{\psi|\theta_i})}\right) \qquad q(\theta_i|\psi) = q(\psi|\theta_i)$$

$$= \min\left(1, \frac{\frac{g(\psi)}{\cancel{T}}}{\frac{g(\theta_i)}{\cancel{T}}}\right) = \min\left(1, \frac{g(\psi)}{g(\theta_i)}\right)$$

$$g(\psi) = \underbrace{p(\psi)}_{\text{prior}} \times \underbrace{p(y|\psi)}_{\text{likelihood}}$$

Working on the log scale

- Let $y = (y_1, \ldots, y_n)$ be the observed data. The likelihood $p(y\theta)$ is typically a product of $p(y_i \mid \theta)$

$$p(y \mid \theta) = \prod_{i=1}^{n} p(y_i \mid \theta).$$

- For numerical stability, we usually do the computations using the log of the posterior density to work with sums instead of products.

- Define

$$\mathscr{L}(\theta) = \underbrace{\log\left(p(\theta)\, p(y \mid \theta)\right)}_{\log g(\theta)} = \log\left(p(\theta)\right) + \log\left(p(y \mid \theta)\right),$$

the log of the posterior density (up to a constant).

Working on the log scale

- So, the log of the likelihood is

$$\log\left(p(y \mid \theta)\right) = \sum_{i=1}^{n} \log\left(p(y_i \mid \theta)\right).$$

- The acceptance probability is

$$\delta = \min\left(0, \mathscr{L}(\psi) - \mathscr{L}(\theta_{i-1})\right).$$

Symmetric MH on the log scale

Define $\mathscr{L}(\theta) = \log\left(p(\theta)\, p(y \mid \theta)\right) = \log\left(p(\theta)\right) + \log\left(p(y \mid \theta)\right)$, the log of the posterior density (up to a constant).

Start with θ_1 randomly. For each $i > 1$:

1. Generate $\psi \sim N(\theta_{i-1}, b^2)$, for some $b > 0$.
2. Compute the probability of acceptance

$$\delta = \min\left(0, \mathscr{L}(\psi) - \mathscr{L}(\theta_{i-1})\right).$$

3. Generate $U \sim U[0,1]$. Set

$$\theta_i = \begin{cases} \psi, & \text{if } \log U < \delta \\ \theta_{i-1}, & \text{otherwise} \end{cases}$$

Board example: Exponential data/Gamma prior

See also exercise sheet 9

- The time until failure for a type of light bulb is exponentially distributed with parameter $\theta > 0$, where θ is unknown.
- We observe n bulbs, with failure times $t_1, \ldots, t_n$.
- We assume a Gamma(α, β) prior distribution for θ, where $\alpha > 0$ and $\beta > 0$ are known.

1. What is the posterior pdf for θ given the data $t = (t_1, \ldots, t_n)$?
2. Write down the steps of the Metropolis-Hastings algorithm to simulate realisations from the posterior distribution by using a normal proposal distribution with standard deviation b.

(a) Data: $t = (t_1, .., t_n)$, where each $t_i \sim Exp(\theta)$ with pdf

$$p(t_i|\theta) = \theta e^{-\theta t_i}, \qquad t_i > 0.$$

The likelihood is

$$\underline{p(t|\theta)} = \prod_{i=1}^{n} p(t_i|\theta) = \theta^n \exp\left(-\theta \sum_{i=1}^{n} t_i\right)$$

$$= \theta^n \exp(-\theta S), \text{ where}$$

$$S = \sum_{i=1}^{n} t_i$$

Prior: $p(\theta) \sim \text{Gamma}(a, b)$ with pdf

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

The posterior, $p(\theta|t)$ is

$$p(\theta|t) \propto p(\theta) \times p(t|\theta)$$

$$\propto \theta^{a-1} \exp\{-b\theta\} \theta^n \exp(-\theta S)$$

$$= \theta^{a+n-1} \exp\{-\theta(b+S)\}$$

Thus, $p(\theta|t) \sim \text{Gamma}(\alpha+n, \beta+S)$

(2) Goal: Construct a symmetric MH algorithm (log-scale) to generate $\{\theta_i\}$ from $p(\theta|t)$.

Define

$$\mathcal{L}(\theta) = \log p(\theta) + \log p(t|\theta)$$
$$= \log p(\theta) + \sum_{i=1}^{n} \log p(t_i|\theta)$$

Start with θ_1 randomly. For each $i > 1$ do the following

(1) $\psi \sim N(\theta_{i-1}, b^2)$

(2) Compute the probability of acceptance

$$\delta = \min\{0, L(\psi) - L(\theta_i)\}$$

3) Generate $v \sim U[0,1]$

$$\theta_i = \begin{cases} \psi, & \text{if } \log v < \delta \\ \theta_{i-1}, & \text{otherwise.} \end{cases}$$

Accept ψ with probability γ

For a Uniform, $U \sim U[0,1]$, the cdf is $P(U \leq \gamma) = \gamma$

we say that if the event $U < \gamma$ happens, we accept ψ.

Board example: Exponential data/Gamma prior

Let $t = (t_1, \dots, t_n)$ be independent and identically distributed data from exponential(θ). We assume a Gamma(α, β) prior distribution for θ. In the following R code, the data t is denoted by `t`, θ by `theta`, α by `alpha` and β by `beta`. We want to simulate from the posterior of θ, $p(\theta \mid t)$.

```
log.post = function(theta)
{
log.likelihood = dexp(t, rate=theta,log=TRUE)
log.prior= dgamma(theta, shape=alpha, rate=beta,log=TRUE)
return(log.prior+sum(log.likelihood))
}
```

- Explain what this function log.post is calculating. In your answer, include a formula involving the prior and likelihood that the function is implementing. It returns $L(\theta) = \log p(\theta) + \log p(y|\theta)$.

Board example: Exponential data/Gamma prior

```
M = 5000
THETA=NULL
theta0=1

for (m in 1:M){
psi=rnorm(1,theta0,0.2)
log.r <- log.post(psi)-log.post(theta0)
if (log(runif(1))<min(0,log.r))
{
theta0 <- psi
}
THETA=c(THETA,theta0)
}
```

- Explain what the command `psi=rnorm(1,theta0,0.2)` is doing in the context of the algorithm. $\psi \sim N(\theta_{i-1}, 0.2^2)$
- Explain what the command `if (log(runif(1))<min(0,log.r))` is doing in the context of the algorithm. In your answer, include a formula involving $p(\theta \mid y)$ that the code is implementing.

- The command generates ψ from the normal proposal distribution centered at theta0 and standard deviation ψ as a proposal for the next value of theta.

- This command is deciding whether to accept the value psi as the next value in the sequence

Board example: Exponential data/Gamma prior

- Although the chain starts nowhere near the posterior mean of 0.11, it arrives there after a few iterations.
- The chain moves up and down many times though the parameter space.

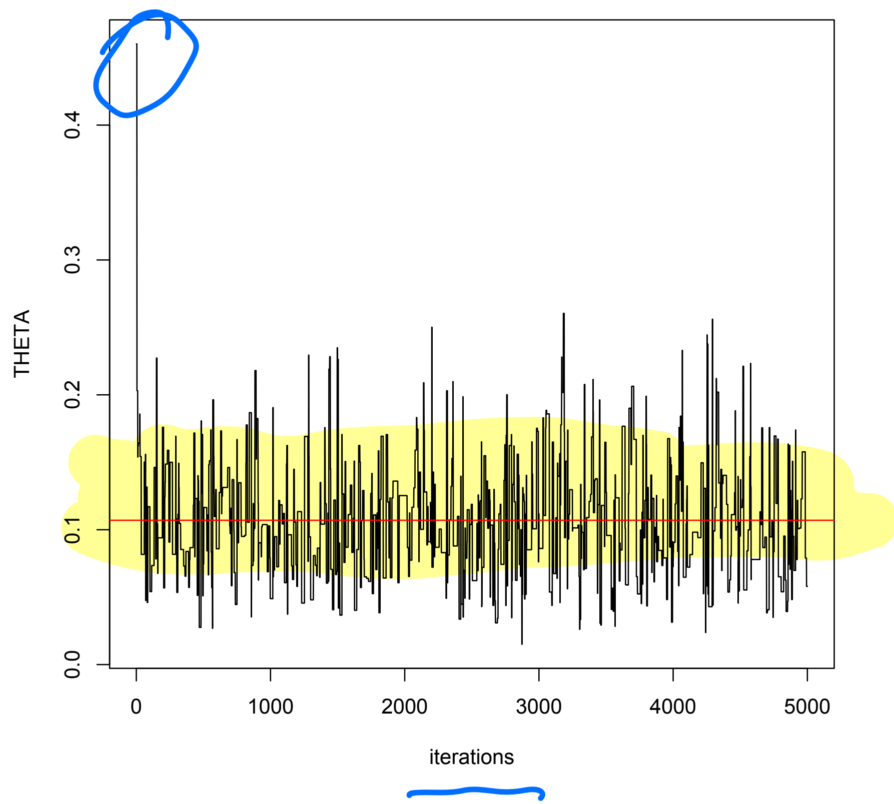


Figure: Plot of the 5000 MCMC observations against iterations. Red line is the posterior mean.

Board example: Exponential data/Gamma prior

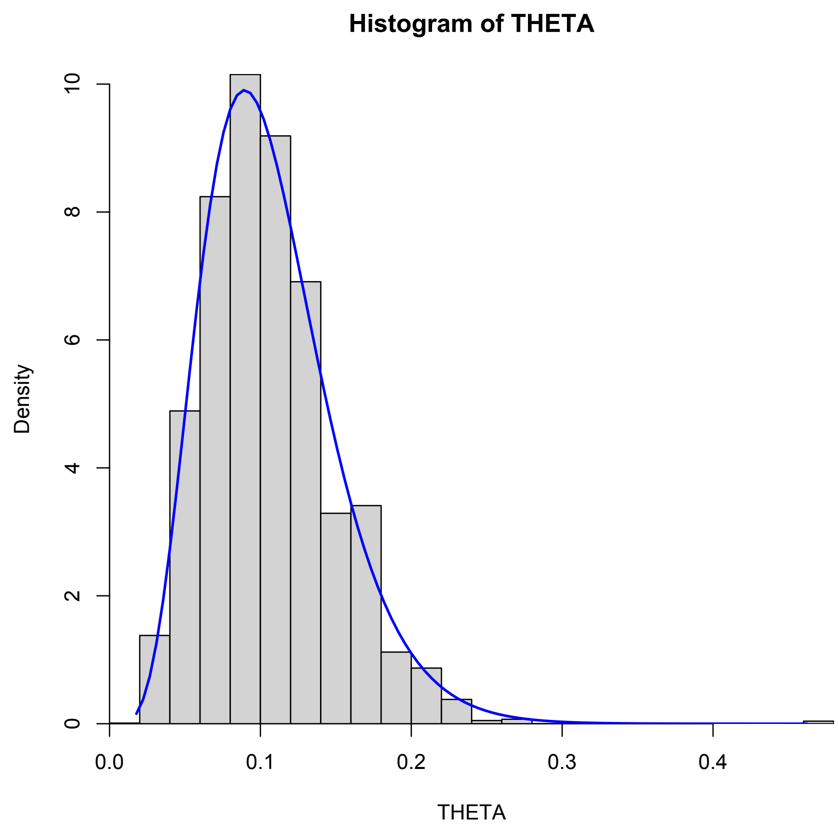


Figure: Histogram of the sample vs the true posterior density in blue

Board example: Exponential data/Gamma prior

Then arrives after few iterations at the region where the posterior density is high.

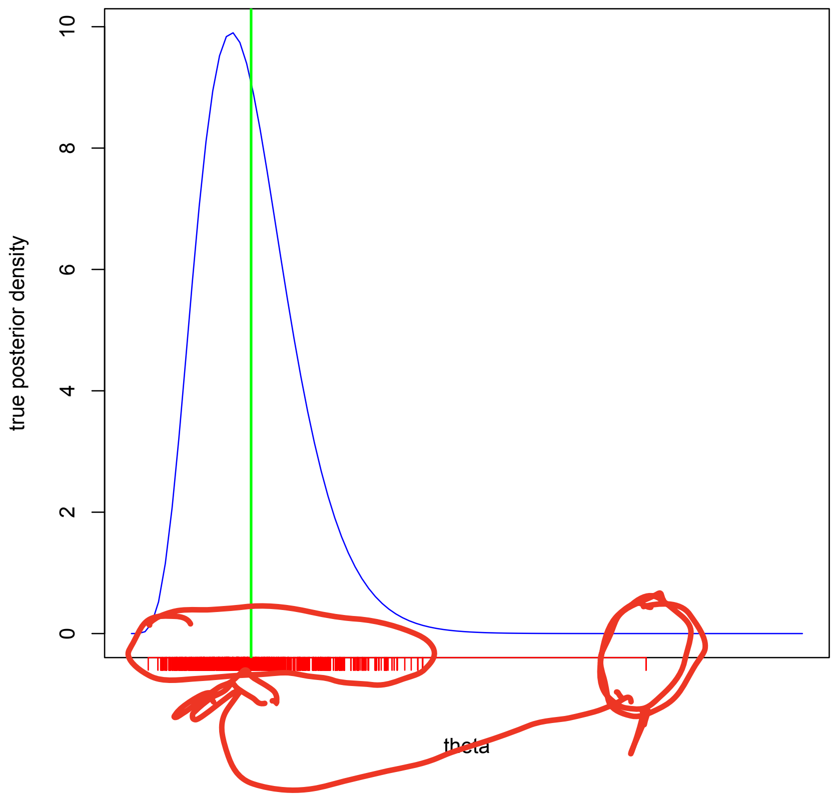


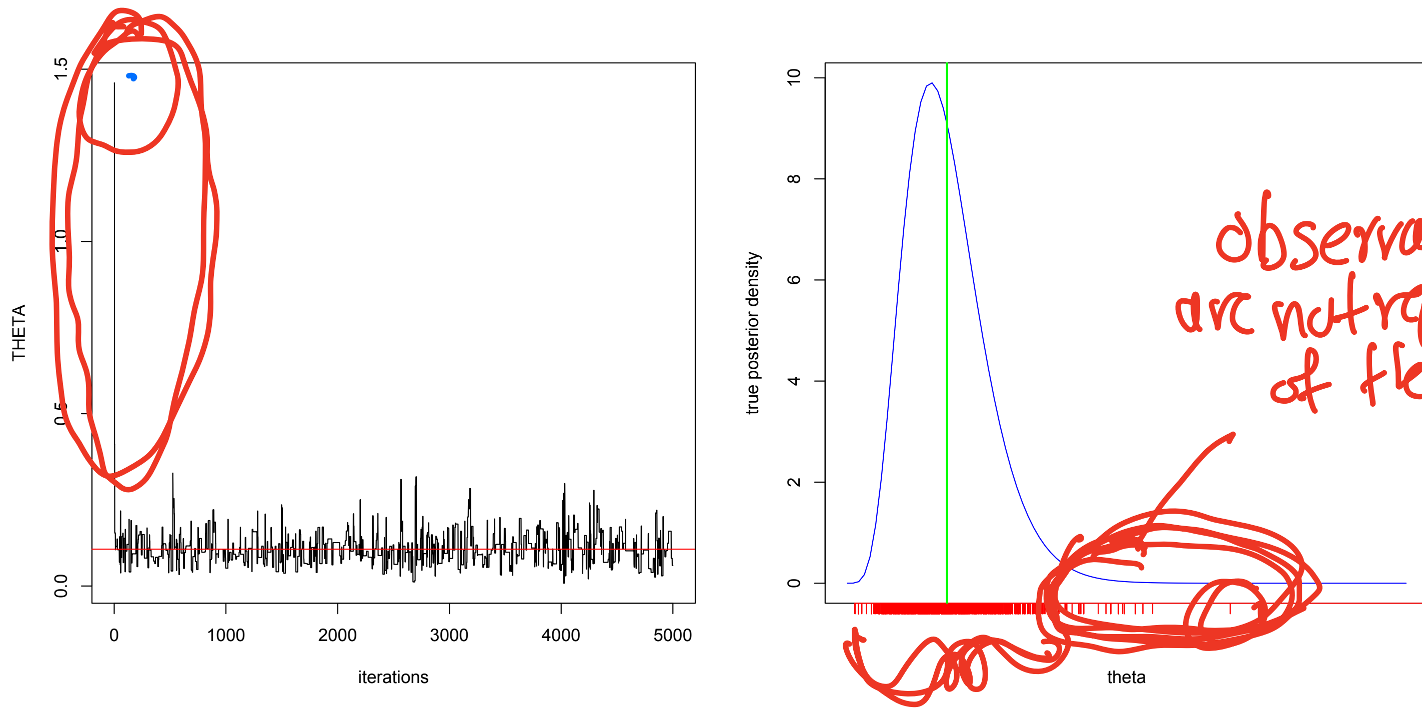
Figure: Blue: true posterior density. Green: true posterior mean. Red: MCMC observations

Choosing an MCMC starting value

- The algorithm eventually produces dependent points $\theta_1, \theta_2, \ldots$ distributed with pdf $p(\boldsymbol{\theta} \mid y)$.
- But we have to start from some θ_1, we can't choose it from $p(\boldsymbol{\theta} \mid y)$.
- **QUESTION:** How do we choose the starting value θ_1?

Exponential data/Gamma: Choosing an MCMC starting value

Plot shows that there are observations at low-probability region and are not representative of the posterior density.



observations are not representative of the posterior

Figure: Left: Plot of the 5000 MCMC observations against iterations with $\theta_1 = 2$. Red line is the posterior mean. Right plot: true density with MCMC observations in red

- The ideal is to start the chain at a region of the parameter space that has high posterior probability.
- However, with a complicated problem you might not know where a high probability region is.

Discarding early iterations: “burn-in”

- To diminish the influence of the starting values, we can generally discard the first 100 or the first 1000 iterations of the sample that are in a low probability region, and focus attention on the remaining observations.
- The practice of discarding early iterations of an MCMC run is known as “burn-in".

Discarding early iterations: “burn-in”

- A standard practice in MCMC approximation is as follows:
  1. Start the chain at some point chosen for convenience.
  2. Run algorithm until some iteration B.
  3. Run the algorithm N more times generating, $\{\theta^{(B+1)}, \ldots, \theta^{(B+N)}\}$
  4. Discard $\{\theta^{(1)}, \ldots, \theta^{(B)}\}$ and use the empirical distribution of $\{\theta^{(B+1)}, \ldots, \theta^{(B+N)}\}$ to approximate $p(\theta \mid y)$.

- The iterations up to and including B are called the “burn-in" period, in which the chain moves from its initial value to a region of the parameter space that has high posterior probability.
- When we say the chain has burned-in, we mean that it has entered a high-probability region.

Exponential data/Gamma: Burn in

A chain that has burned in

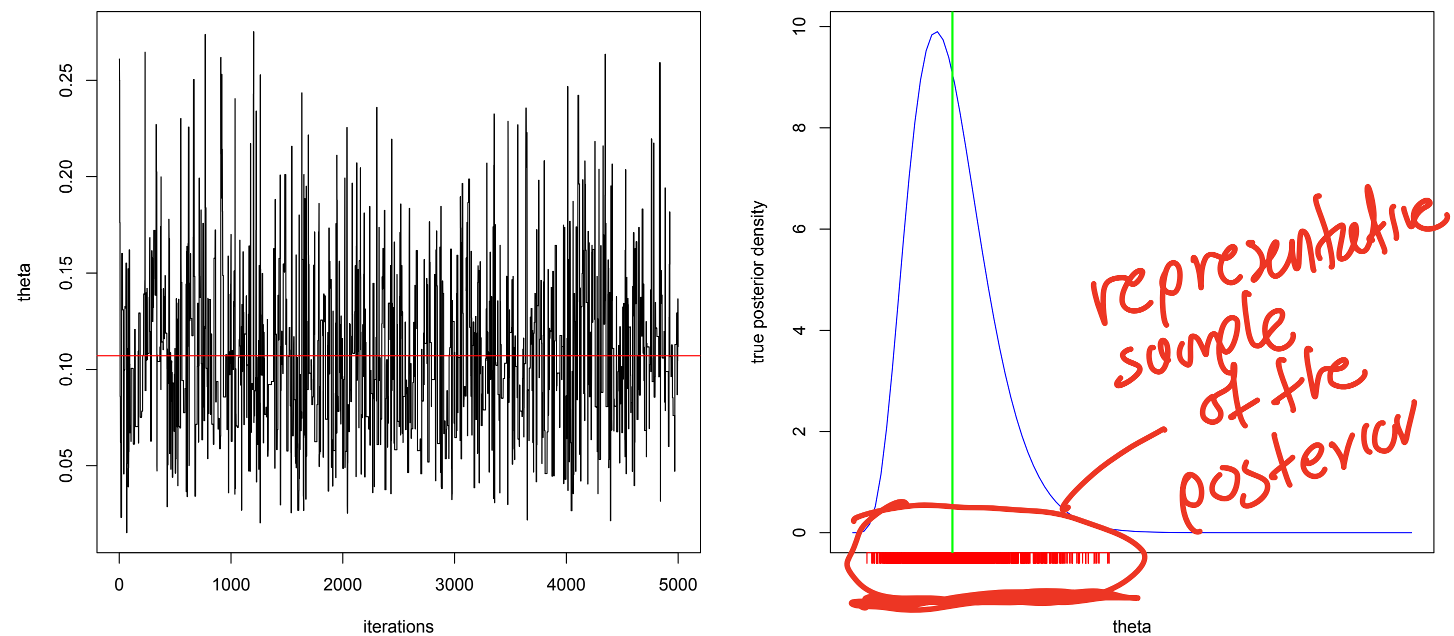
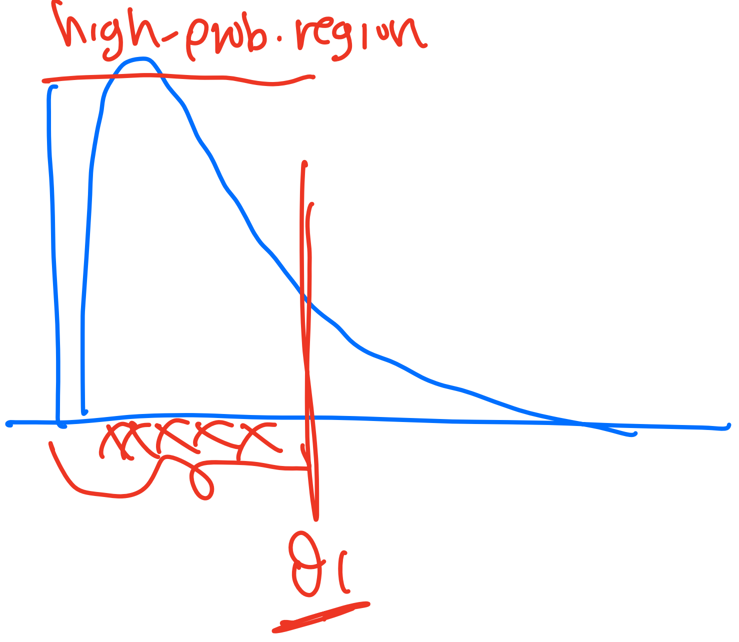


Figure: Left: Plot of the ~~5000~~ 2500 MCMC observations against iterations with $\theta_1 = 2$ after throwing out the first half iterations. Red line is the posterior mean. Right plot: true density with MCMC observations in red

Discarding early iterations

- In theory, longer burnin periods will cause the chain to “forget" its starting value so that the influence of this value will be lessened.
- If we have a good idea of where the high posterior probability region is, we can reduce the burn-in period by starting the chain there.
- In general, any value at where the posterior density is high will suffice, (e.g the MLE of the data or the posterior mode), and burn-in may not be necessary. The chain is burned in immediately.


high-prob. region
θ_1

Metropolis algorithm proposal distribution

- In the symmetric Metropolis-Hastings algorithm, the proposal distribution q is most often taken as a normal distribution centred on the current point
$$\psi \sim N(\theta_{i-1}, b^2).$$
- The efficiency of the Metropolis-Hastings sample depends on the choice of the standard deviation b.
- **QUESTION:** But, what value of b should we choose?

Metropolis algorithm proposal distribution

- Recall, the algorithm produces dependent points $\theta_1, \theta_2, \ldots$ distributed with pdf $p(\theta \mid y)$.

- An ideal choice of b would lead to a small correlation of subsequent realisations θ_{i-1} and θ_i.

Dependence of the iterations in each sequence

- The θ_{i-1} and θ_i simulated values from an MCMC algorithm are correlated:
    - There exists correlation between the θ_{i-1} and θ_i, since $\psi \sim q(\cdot \mid \theta_{i-1})$ and $\theta_i = \psi$ if ψ is accepted.
    - There exists correlation between θ_i and θ_{i-1} if ψ is rejected and $\theta_i = \theta_{i-1}$.

- $\psi \sim N(\theta_{i-1}, b^2)$

so ψ is correlated with θ_{i-1}.

If ψ is accepted, then $\theta_i = \psi$

$\Big\} \Rightarrow \theta_i$ and θ_{i-1} are correlated

If ψ is rejected, $\underbrace{\theta_i = \theta_{i-1}}_{\text{very correlated}}$

Metropolis algorithm proposal distribution

- The choice of b will affect the acceptance probability,

$$r = \min\left(1, \frac{g(\psi)}{g(\theta_{i-1})}\right),$$

and hence the correlation in the Markov chain.

Metropolis algorithm proposal distribution

- For example, if b is very small, then ψ is close to θ_{i-1}. So $g(\psi)$ is close to $g(\theta_{i-1})$.
- Hence there is a high probability of accepting the proposal.
- But the chain will move very slowly around the space, and the Markov chain will be highly correlated.

- $\psi \sim N(\theta_{i-1}, b^2)$

If b is small, then $\psi \approx \theta_{i-1}$
and so $g(\psi) \approx g(\theta_{i-1})$. The probability of acceptance is

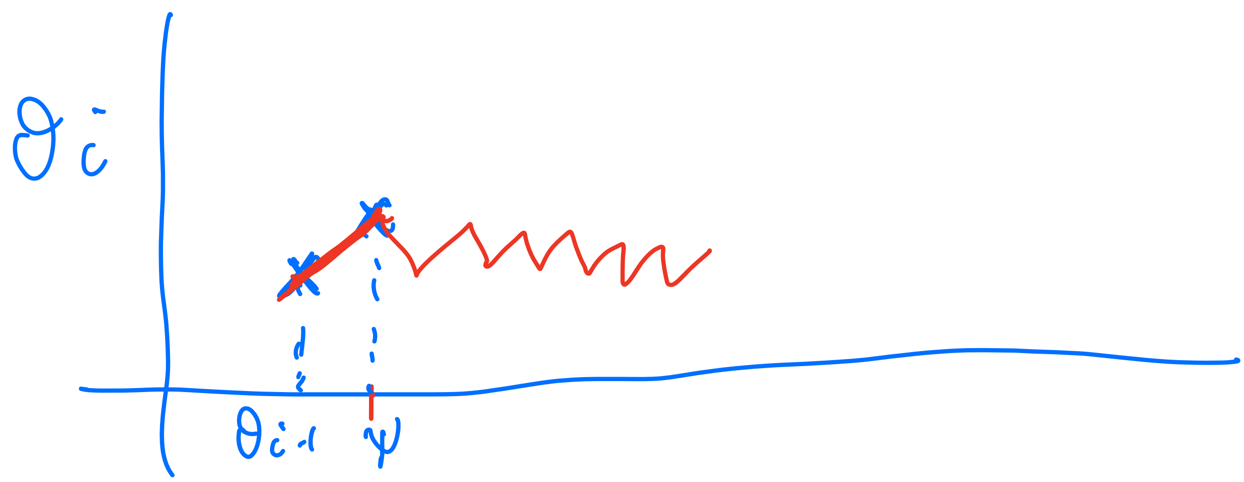
$$r = \min\left\{1, \frac{g(\psi)}{g(\theta_{i-1})}\right\}$$

When b is small, r is a large value.

Also, if b is small and ψ is accepted

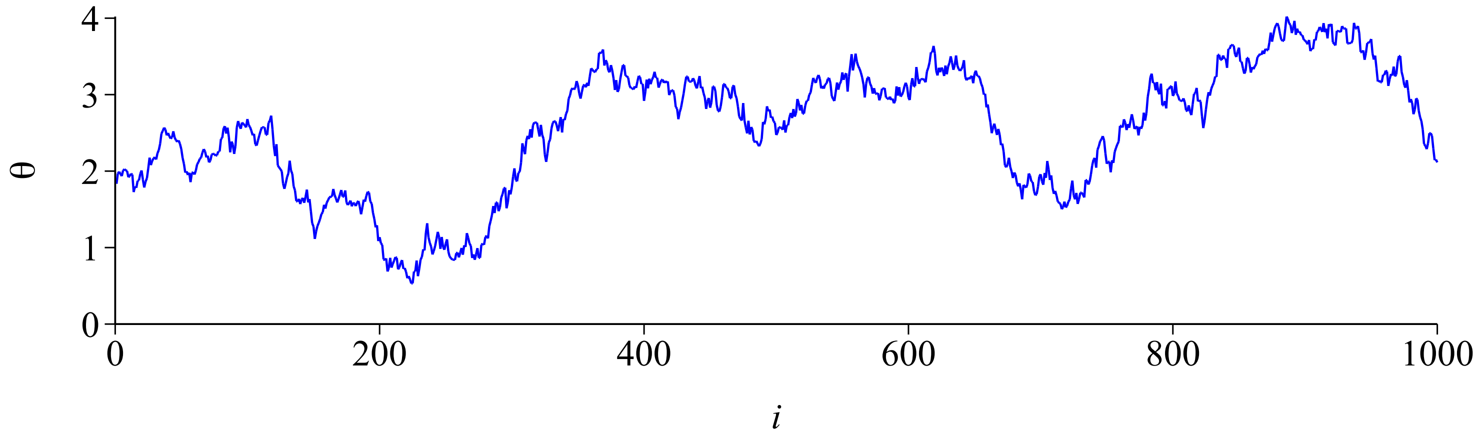
then $\theta_i = \psi$
but $\psi \approx \theta_{i-1}$
$\Rightarrow$ $\theta_i \approx \theta_{i-1}$

chain moves slowly
makes small steps



Example: Sample paths with small b

- Figure: θ against iteration number i.
- Not good, proposal standard deviation b is too small. The acceptance probability is high but the chain is hardly moving.



Metropolis algorithm proposal distribution

- On the other hand, if b is large, then ψ may be far from θ_{i-1}.
- And $g(\psi)$ may be much lower than $g(\theta_{i-1})$.
- Now there is a lower probability of accepting the proposal ψ.
- The chain makes large jumps (so moves fast) and remains at the same place quite often, and hence Markov chain will be highly correlated

$\psi \sim N(\theta_{i-1}, b^2)$

- b is large, so ψ is far from the mean θ_{i-1}

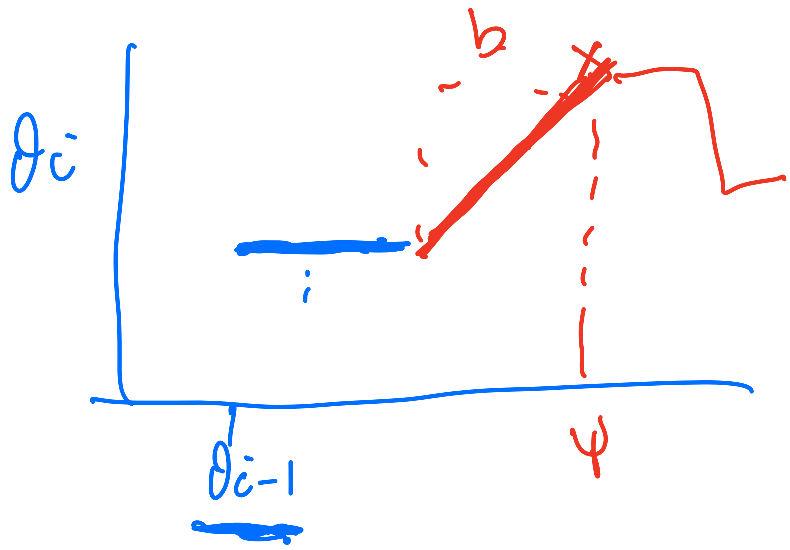
so $g(\psi)$ may be much lower than $g(\theta_{i-1})$

so $r = \min\left\{1, \dfrac{g(\psi)}{g(\theta_{i-1})}\right\}$ will be small

So this means most of the time ψ will be rejected. The chain remains at θ_{i-1} for a long time

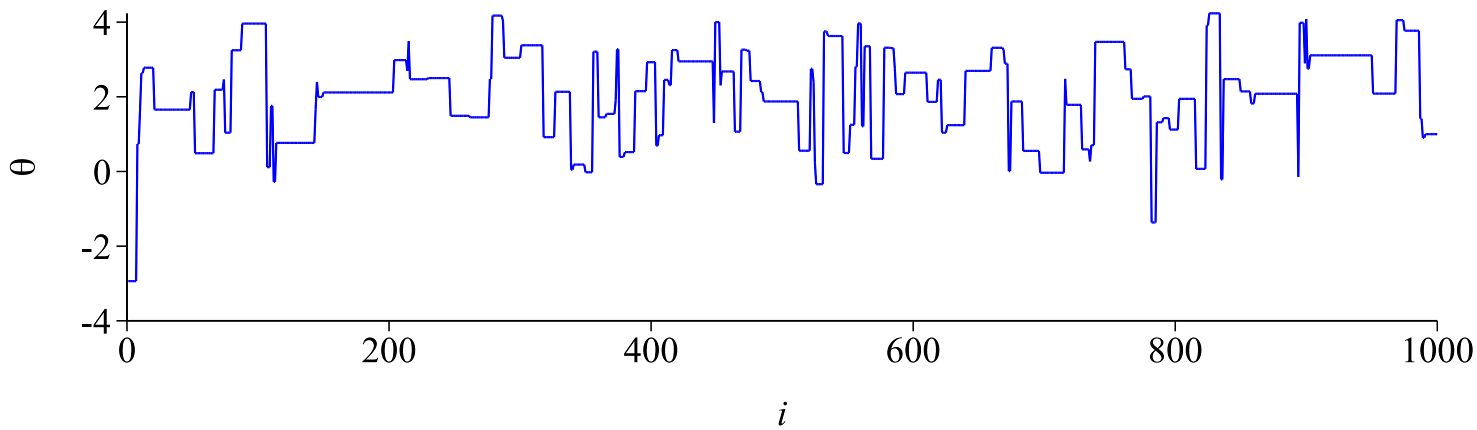
- If ψ is rejected
  $\theta_i = \theta_{i-1}$

- If ψ is accepted
  then $\theta_i = \psi$
  and ψ is far from θ_{i-1}



Example: Sample paths with too large b

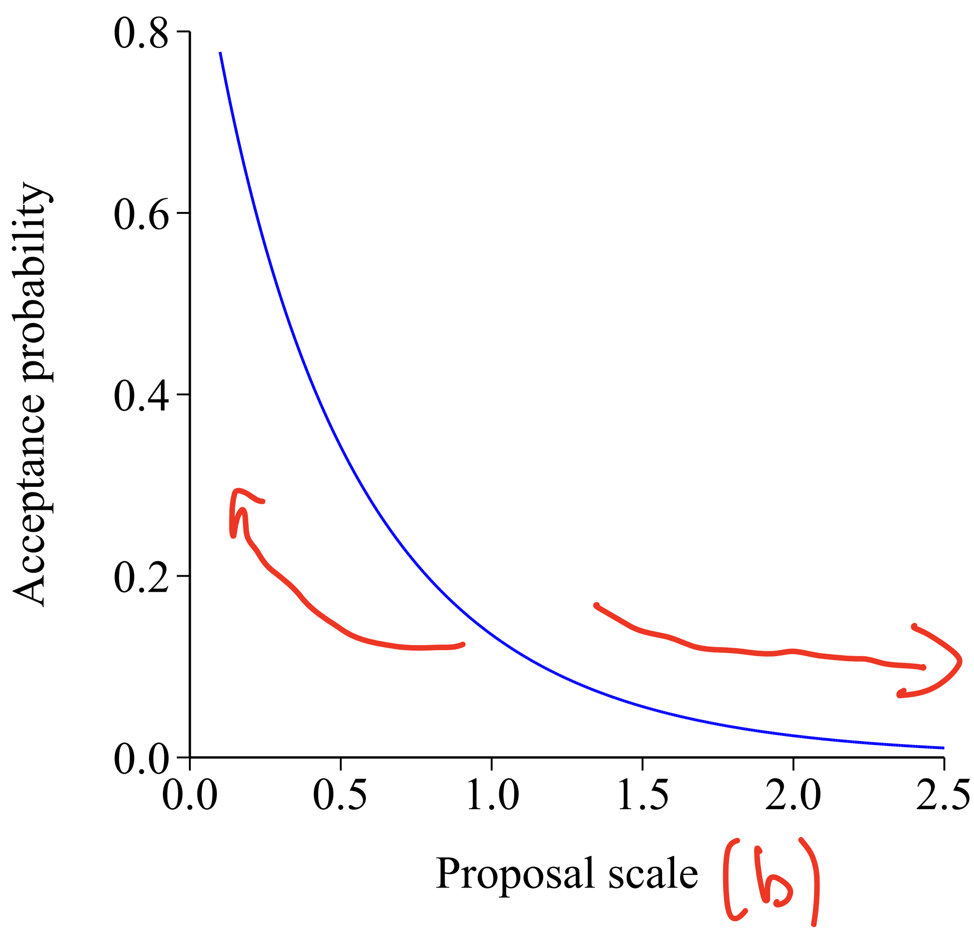
- Choosing too large b
- The chain moves fast but too many proposals are rejected (small acceptance probability), and hence remains for a long time at each accepted value.



Metropolis algorithm acceptance probability

b affects the acceptance probability.

- Probability that each proposal is accepted tends to decrease as the proposal scale b is increased.

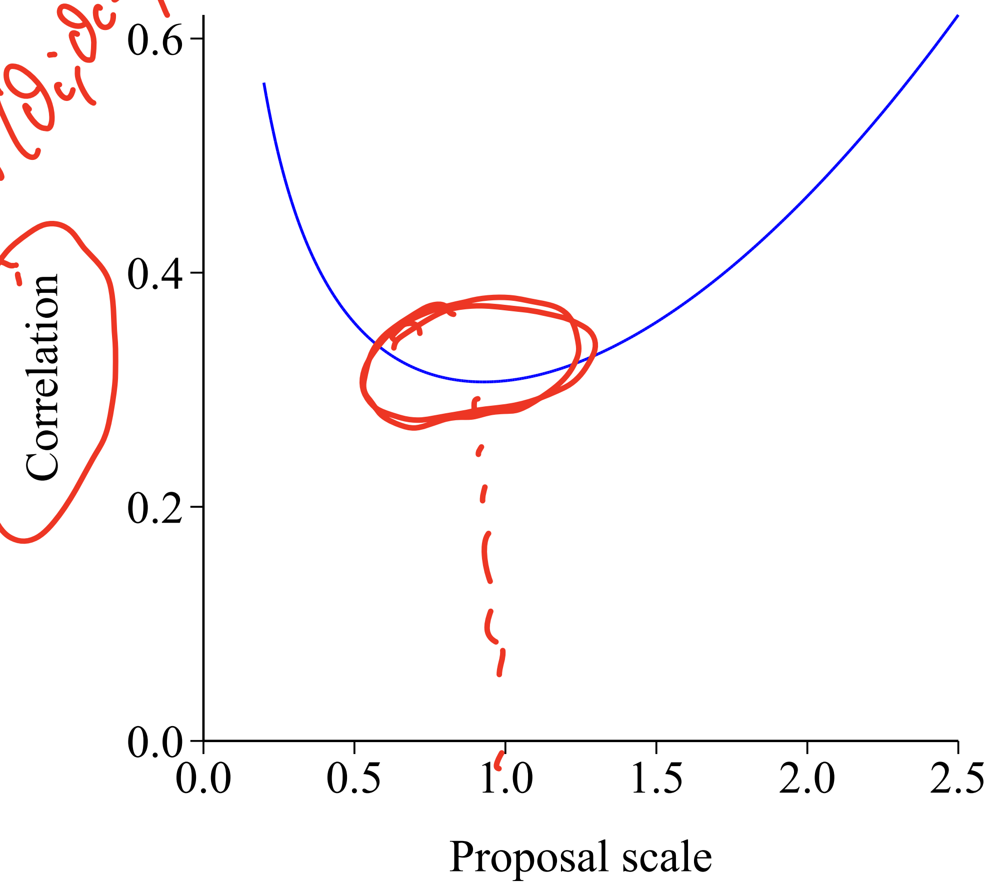


Proposal scale and acceptance probability

b affects the correlation in the Markov chain

- Some intermediate value for b tends to be best for reducing the correlation.
- Its value depends on the model and the data.

cov(θ_i, θ_{i-1})



Choosing b

- **Goal:** We want to choose b such that the chain moves fast and yields a high probability of acceptance, to reduce the correlation between θ_i and θ_{i+1} values.
- Theoretically, it has been shown that the optimal acceptance rate is around 0.234 (an asymptotic result).
- But experience suggests that an acceptance rate of around 20%-30%.
- Thus, the standard deviation b should be tuned to get an acceptance rate of around this level.

Choosing b

Recommendations

- It is common practice to implement several short runs of the Metropolis-Hastings algorithm under different values of b.
- Choose b that gives an acceptance rate r roughly between 20%-30%.
- Once a reasonable value b is selected a longer more efficient Markov chain can be run.

Exponential data/Gamma: Choosing b

- We examine the choices $b = 0.001$, $b = 0.02$, $b = 0.2$ and $b = 5$ for the Exponential data/Gamma prior example.
- Table 1 shows the acceptance probability for the different choices of the proposal standard deviation b

	Probability of acceptance
$b = 0.001$	0.98
$b = 0.02$	0.84
$b = 0.2$	0.25
$b = 5$	0.088

Exponential data/Gamma: $b = 0.001$

- Choosing b too small, $b = 0.001$, the acceptance probability is very high.
- However, the chain is in a low posterior probability region, and moves very slowly toward a higher probability region.

$\theta_1 = 2$ (starting value)

The chain is in a low-probability region

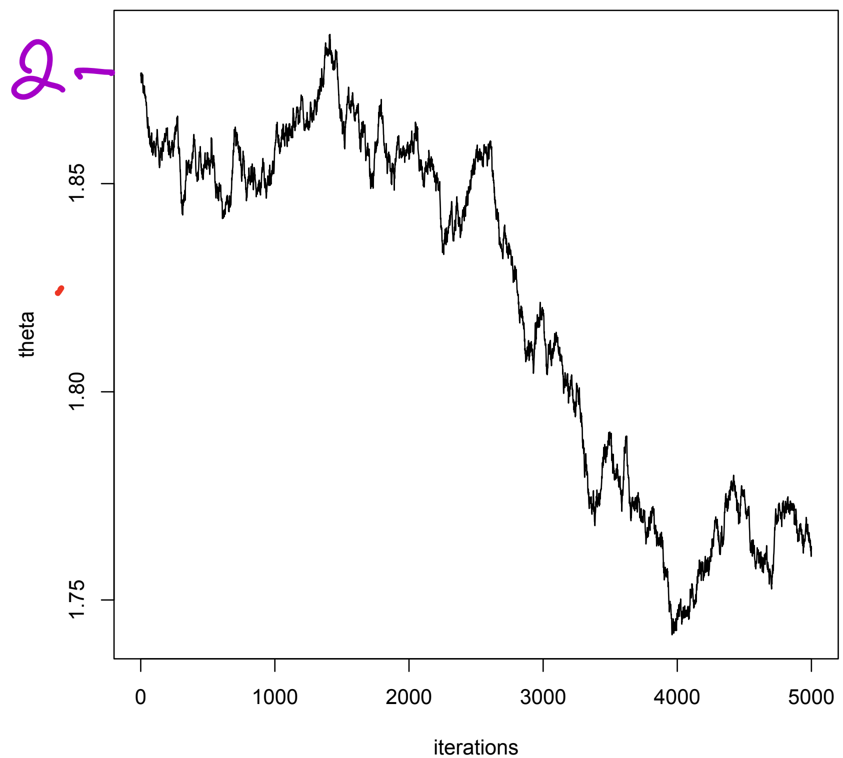


Figure: Sample paths when $b = 0.001$ $M = 5000$

Exponential data/Gamma: $b = 0.02$

- Choosing $b = 0.02$ yields again a high probability of acceptance of 0.84, but the chain changes only very slowly.

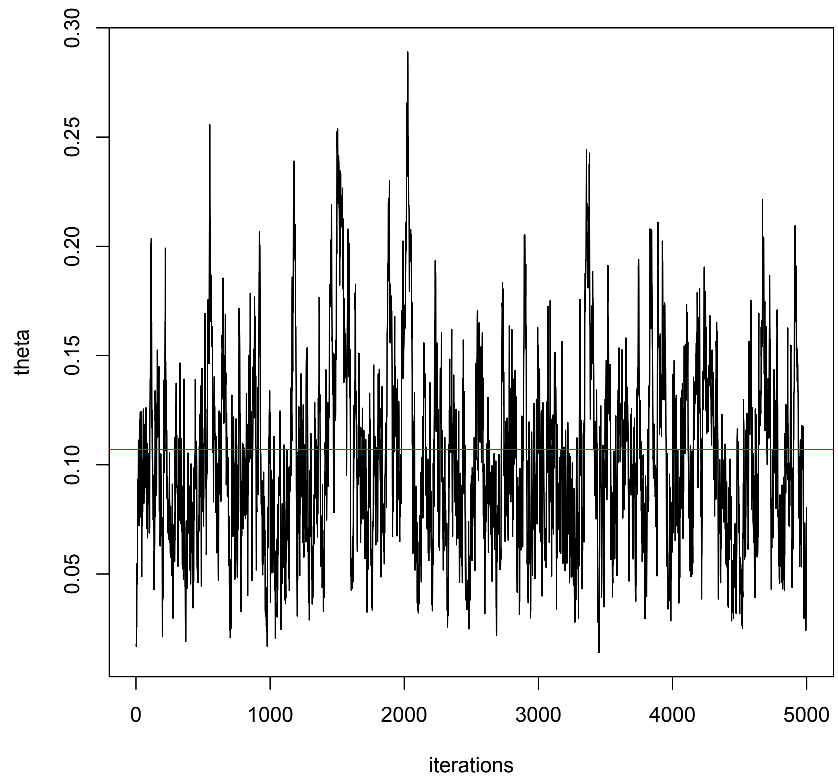


Figure: Sample paths when $b = 0.02$

Exponential data/Gamma: $b = 5$

- Choosing $b = 5$ too large allows the chain to make large jumps, however the acceptance probability is small
- So the chain remains for a long time at each accepted value.

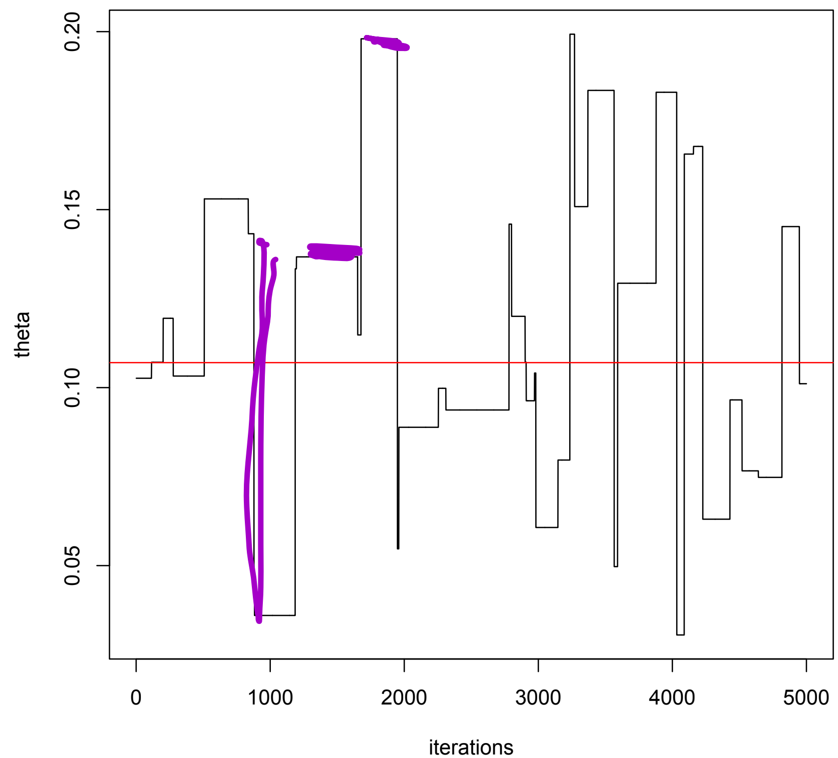
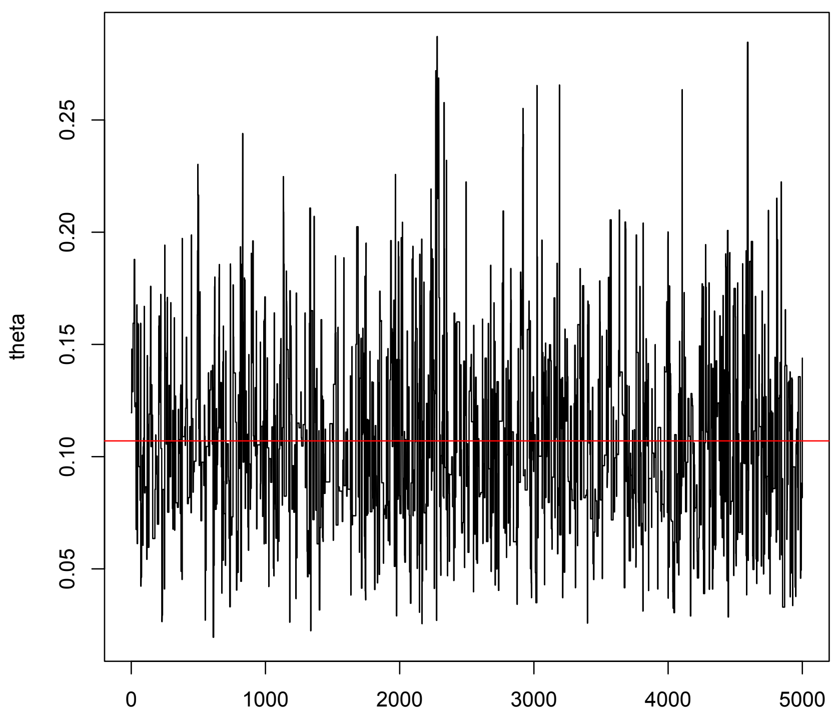
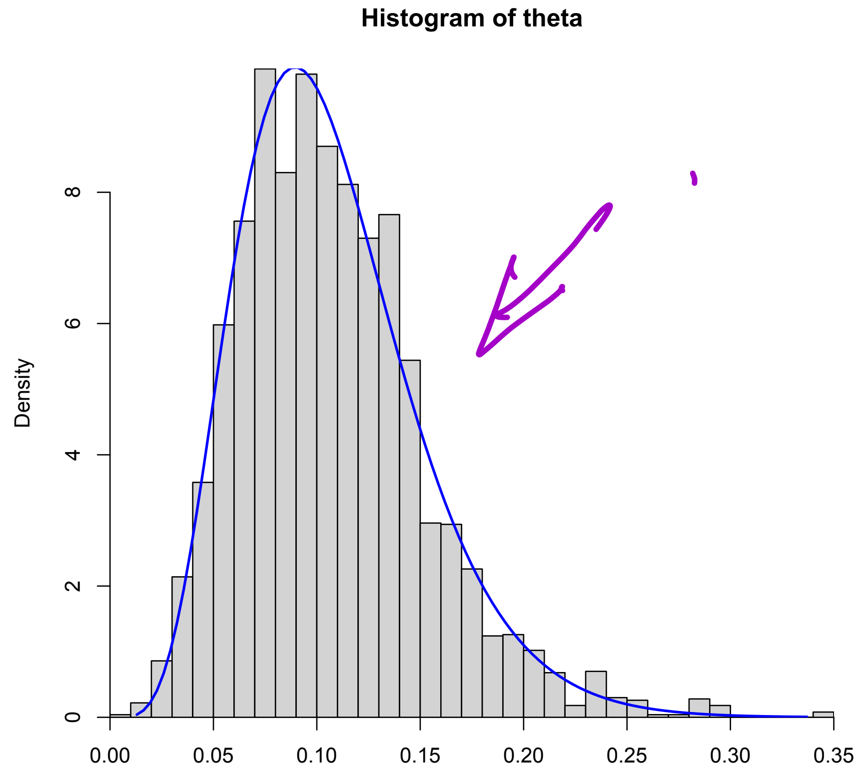


Figure: Sample paths when $b = 5$

Exponential data/Gamma: $b = 0.2$

- Choosing $b = 0.2$ yields an acceptance probability of 0.24. This is the optimal choice.
- Sequence should move up and down through the parameter space many times.
- By selecting b carefully, we can decrease the correlation in the chain, leading to an improvement in the approximation to the posterior distribution.





Checking that sampling worked

- Finally, we need to check if the method has sampled the posterior distribution well enough.
- Check that summaries such as posterior median, 95% credible intervals are similar for each sequence.