

Introduction to Markov Chain Monte Carlo

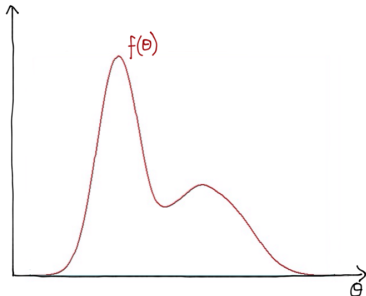
Recap. on Bayesian inference

Last time we saw that the **posterior distribution** of θ , given observed data is

$$p(\theta|\text{data}) \propto p(\text{data}|\theta)p(\theta)$$

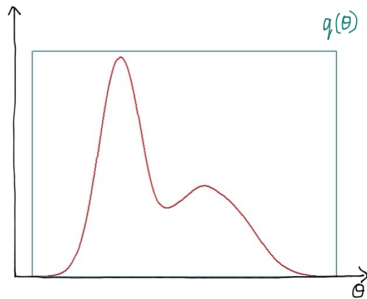
Our aim is to draw samples from this distribution.

Rejection sampling



- Consider a distribution $f(\theta)$, which we can evaluate for any θ
- How do we draw samples?

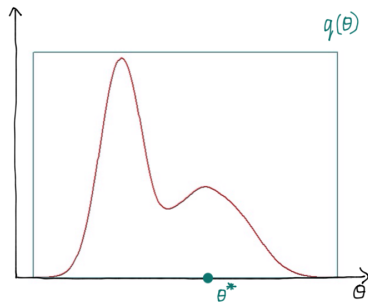
Rejection sampling



Rejection sampling uses a **proposal distribution** $q(\theta)$ which:

- is simple to evaluate
- is easy to sample from
- has a greater density than $f(\theta)$ for all θ

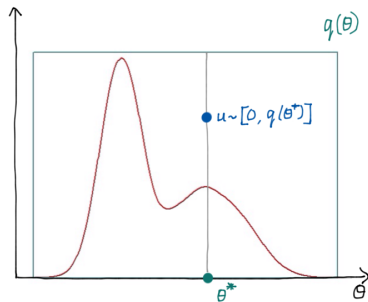
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$

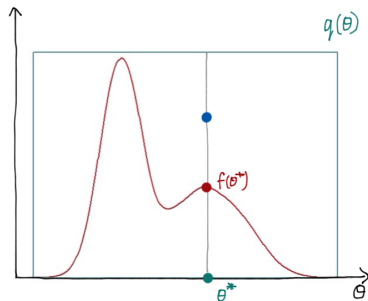
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$

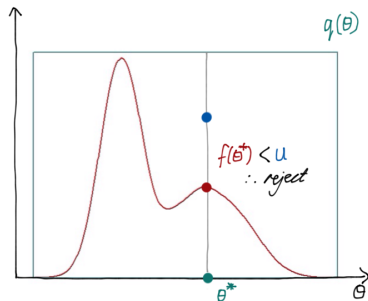
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$

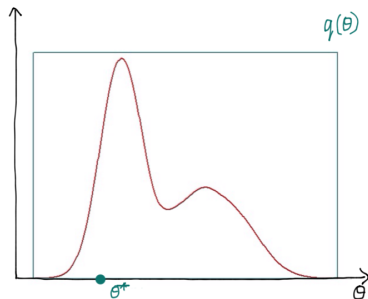
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject

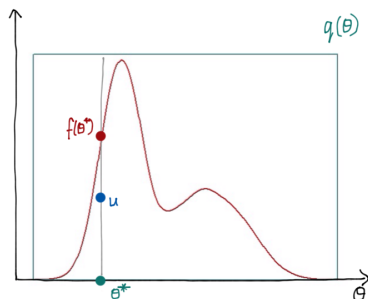
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

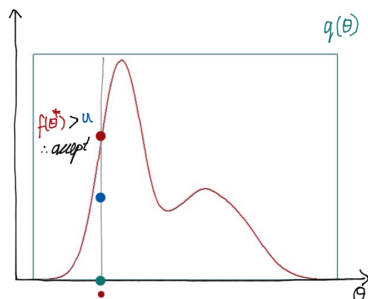
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

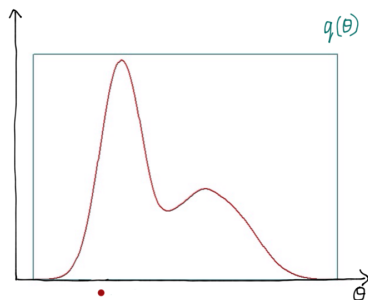
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

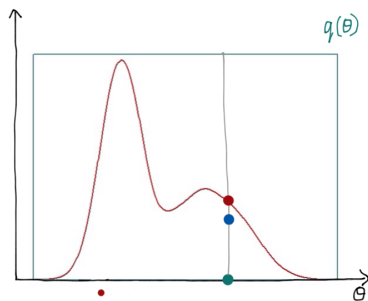
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

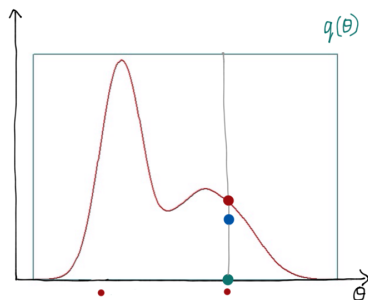
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

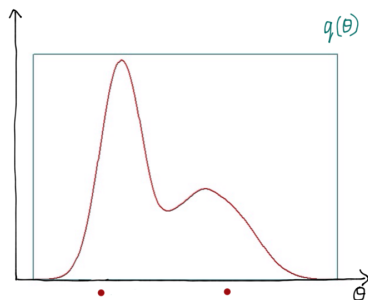
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

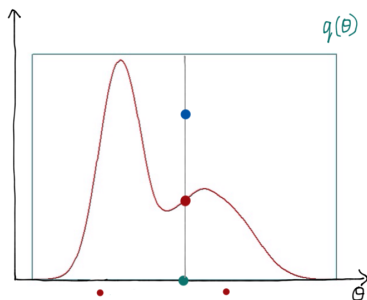
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

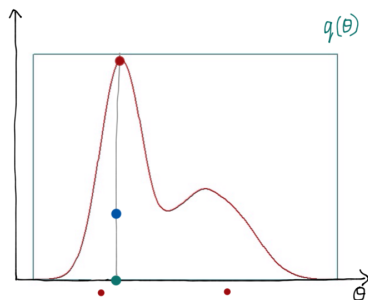
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

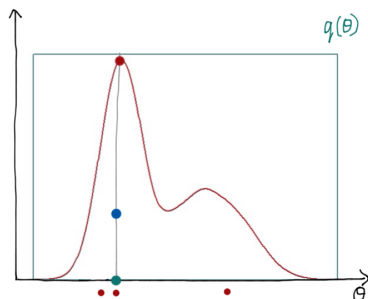
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

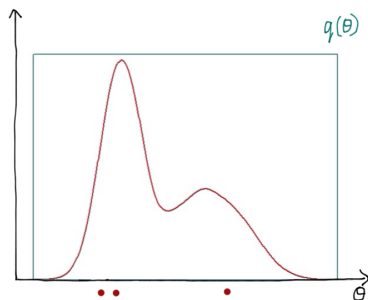
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

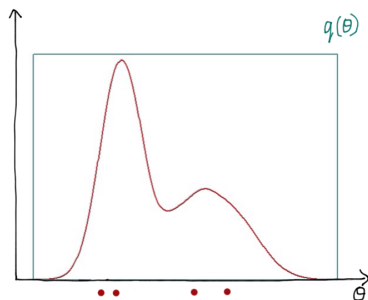
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

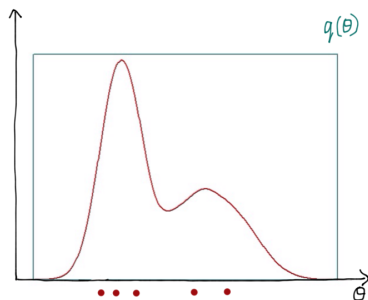
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

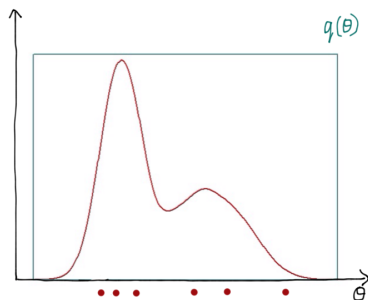
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

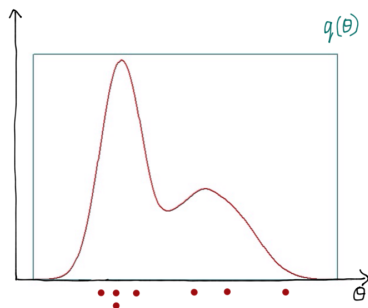
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

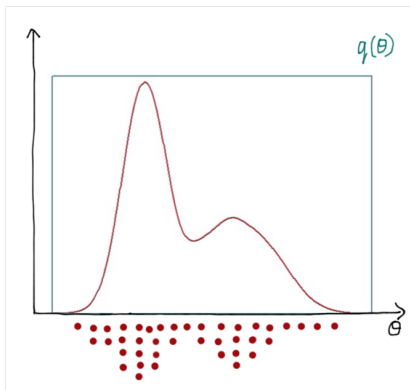
Rejection sampling



The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

Rejection sampling

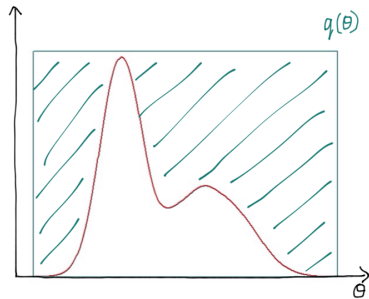


The algorithm proceeds as follows:

1. Sample θ^* from $q(\theta)$
2. Draw $u \sim \text{Uniform}[0, q(\theta^*)]$
3. Evaluate $f(\theta^*)$
4. If $f(\theta^*) > u$ accept, else reject
5. Repeat steps 1-4

Rejection sampling

- Rejection sampling works best if $q(\theta) \approx f(\theta)$
- Requiring $q(\theta) \geq f(\theta)$ for all θ can make rejection rate v. high
- Even more limited in high dimensions



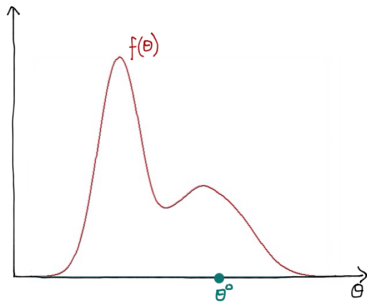
Markov Chain Monte Carlo

- In Markov Chain Monte Carlo (MCMC) we do not define one proposal density $q(\theta) \geq f(\theta)$.
- Rather we build up a **chain** of samples where each proposed θ^* depends on the previous one

i.e the proposal density takes the form $q(\theta^*|\theta)$

- A commonly used MCMC algorithm is **Metropolis-Hastings** (M-H).
- The acceptance rate of M-H is carefully derived to ensure **unbiased samples**.

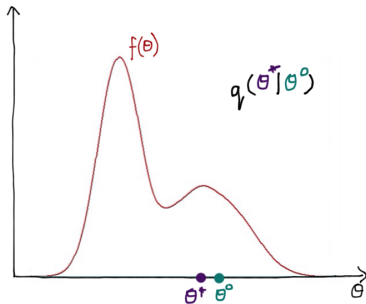
Metropolis-Hastings



The algorithm proceeds as follows:

1. Initialise θ^0 , set $\theta = \theta^0$

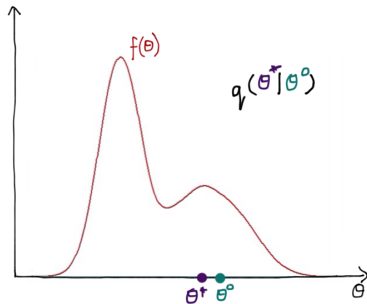
Metropolis-Hastings



The algorithm proceeds as follows:

1. Initialise θ^0 , set $\theta = \theta^0$
2. Sample $\theta^* \sim q(\theta^*|\theta)$

Metropolis-Hastings



The algorithm proceeds as follows:

1. Initialise θ^0 , set $\theta = \theta^0$
2. Sample $\theta^* \sim q(\theta^*|\theta)$
3. Compute acceptance probability, r

Metropolis-Hastings

Acceptance

- If $q(\theta^*|\theta)$ symmetric, then

$$r = \min \left(1, \frac{f(\theta^*)}{f(\theta)} \right)$$

The algorithm proceeds as follows:

1. Initialise θ^0 , set $\theta = \theta^0$
2. Sample $\theta^* \sim q(\theta^*|\theta)$
3. Compute acceptance probability, r

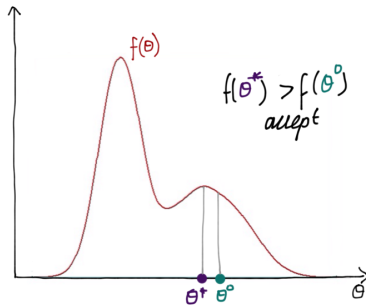
Metropolis-Hastings

Acceptance

- If $q(\theta^*|\theta)$ symmetric, then

$$r = \min \left(1, \frac{f(\theta^*)}{f(\theta)} \right)$$

- Definitely move to θ^* if more probable than θ



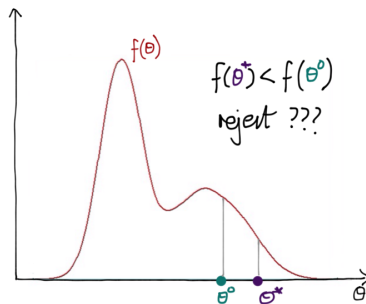
Metropolis-Hastings

Acceptance

- If $q(\theta^*|\theta)$ symmetric, then

$$r = \min \left(1, \frac{f(\theta^*)}{f(\theta)} \right)$$

- Definitely move to θ^* if more probable than θ
- May move if θ^* less probable



Metropolis-Hastings

Acceptance

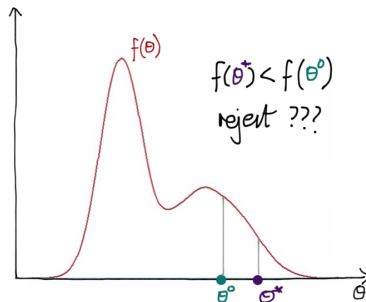
- If $q(\theta^*|\theta)$ symmetric, then

$$r = \min \left(1, \frac{f(\theta^*)}{f(\theta)} \right)$$

- Definitely move to θ^* if more probable than θ
- May move if θ^* less probable

- If $q(\theta^*|\theta)$ asymmetric, then

$$r = \min \left(1, \frac{f(\theta^*)q(\theta|\theta^*)}{f(\theta)q(\theta^*|\theta)} \right)$$



Metropolis-Hastings

Acceptance

- If $q(\theta^*|\theta)$ symmetric, then

$$r = \min \left(1, \frac{f(\theta^*)}{f(\theta)} \right)$$

- Definitely move to θ^* if more probable than θ
- May move if θ^* less probable

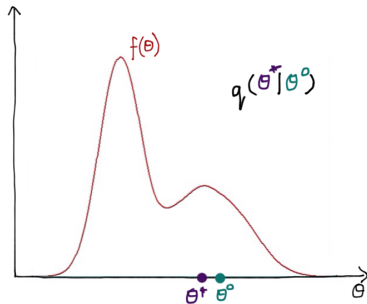
- If $q(\theta^*|\theta)$ asymmetric, then

$$r = \min \left(1, \frac{f(\theta^*)q(\theta|\theta^*)}{f(\theta)q(\theta^*|\theta)} \right)$$

The algorithm proceeds as follows:

1. Initialise θ^0 , set $\theta = \theta^0$
2. Sample $\theta^* \sim q(\theta^*|\theta)$
3. Compute acceptance probability, r

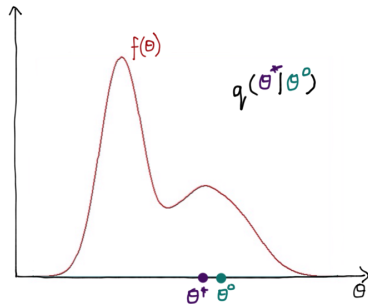
Metropolis-Hastings



The algorithm proceeds as follows:

1. Initialise θ^0 , set $\theta = \theta^0$
2. Sample $\theta^* \sim q(\theta^*|\theta)$
3. Compute acceptance probability, r

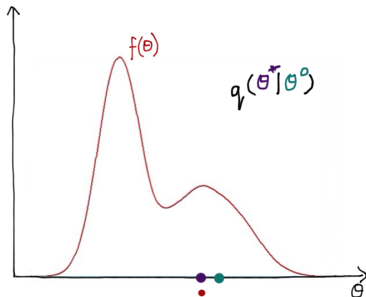
Metropolis-Hastings



The algorithm proceeds as follows:

1. Initialise θ^0 , set $\theta = \theta^0$
2. Sample $\theta^* \sim q(\theta^* | \theta)$
3. Compute acceptance probability, r
4. Draw $u \sim \text{Uniform}[0, 1]$

Metropolis-Hastings

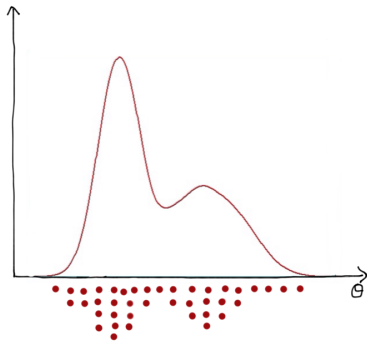


The algorithm proceeds as follows:

1. Initialise θ^0 , set $\theta = \theta^0$
2. Sample $\theta^* \sim q(\theta^*|\theta)$
3. Compute acceptance probability, r
4. Draw $u \sim \text{Uniform}[0, 1]$
5. Set new sample to

$$\theta^{(s+1)} = \begin{cases} \theta^*, & \text{if } u < r \\ \theta^{(s)}, & \text{if } u \geq r \end{cases}$$

Metropolis-Hastings



The algorithm proceeds as follows:

1. Initialise θ^0 , set $\theta = \theta^0$
2. Sample $\theta^* \sim q(\theta^*|\theta)$
3. Compute acceptance probability, r
4. Draw $u \sim \text{Uniform}[0, 1]$
5. Set new sample to

$$\theta^{(s+1)} = \begin{cases} \theta^*, & \text{if } u < r \\ \theta^{(s)}, & \text{if } u \geq r \end{cases}$$

6. Repeat steps 2-5

Review

In the practical you used *Metropolis-Hastings* with a *Gaussian* proposal distribution to infer *one* parameter, R_0

In this session we will:

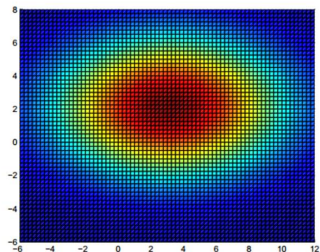
- extend to multivariate inference
- learn about MCMC diagnostics
- think about accuracy and efficiency

Interlude: Multivariate Gaussian distribution

To infer more multiple parameters we can use multivariate Gaussian

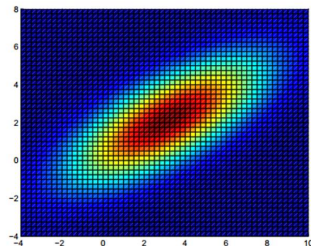
$$\text{mean } \mu = \begin{bmatrix} 3 & 2 \end{bmatrix}$$

$$\text{covariance } \Sigma = \begin{bmatrix} 25 & 0 \\ 0 & 9 \end{bmatrix}$$



$$\text{mean } \mu = \begin{bmatrix} 3 & 2 \end{bmatrix}$$

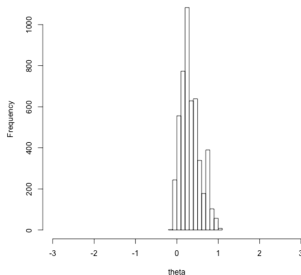
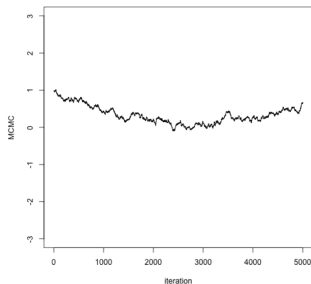
$$\text{covariance } \Sigma = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix}$$



For accurate and efficient MCMC we tune the variance and covariance of the proposal distribution.

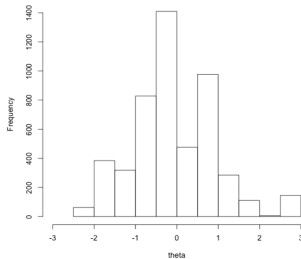
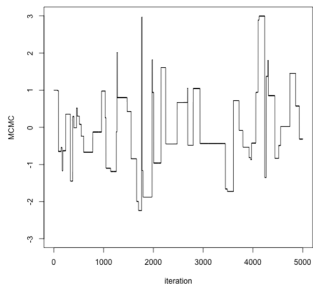
Choosing a proposal distribution

If **variance is too small**, the chain will be slow to reach the target distribution.



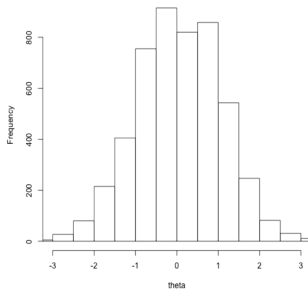
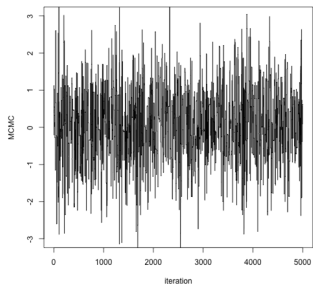
Choosing a proposal distribution

If **variance is too high**, many proposed values will be rejected and the chain will *stick* in one place for many steps.



Choosing a proposal distribution

If **variance is just right**, the chain will efficiently explore the full shape of the target distribution.



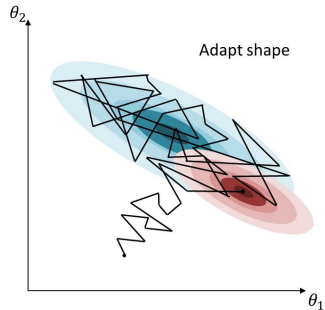
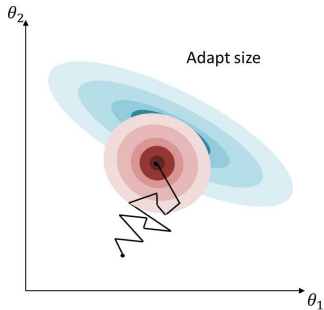
Try several different proposal distributions (**pilot runs**), aiming for an acceptance rate between 24% and 40%.

Adaptive MCMC

- **Adaptive MCMC** alters proposal distribution while chain is running.
- Start with large symmetric variance, scan around to find a mode.
- Then alter shape of proposal distribution to match covariance matrix of accepted values.
- Eventually proposal density should match the shape of target density.

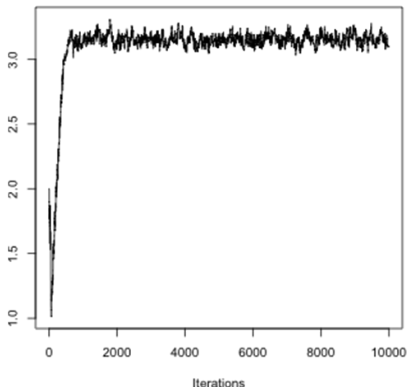
Adaptive MCMC

Two-stage adaptation



Burn-in

- We can start our MCMC chain anywhere.
- It can take a while to reach and explore the target density $f(\theta)$.
- Throw away early samples: **burn-in** phase.
- How much to discard?



MCMC sample size

- In MCMC, each sample depends on the one before - **auto-correlation**
- Reduce degree of auto-correlation by **thinning**, only retain every n^{th} sample.
- Information content of MCMC samples is given by the **effective sample size (ESS)**.
- We use the R package *coda*.