# Introduction to Markov Chain Monte Carlo

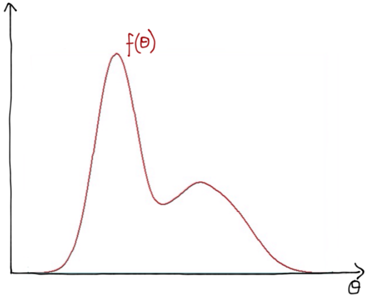
# Recap. on Bayesian inference

Last time we saw that the posterior distribution of $\theta$, given observed data is

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

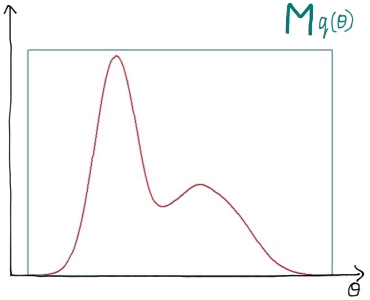
Our aim is to draw samples from this distribution.

# Rejection sampling




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

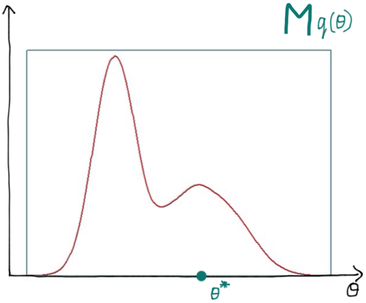
# Rejection sampling




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

- is simple to evaluate
- is easy to sample from
- one can find $M > 1$ such that $f(\theta) < Mq(\theta)$ for all $\theta$

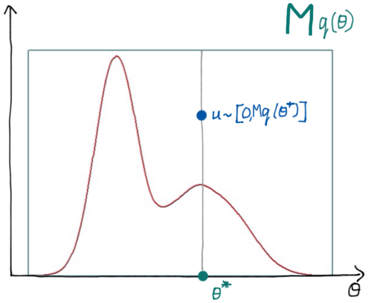
# Rejection sampling




The algorithm proceeds as follows:

1. Sample $\theta^*$ from $q(\theta)$

# Rejection sampling

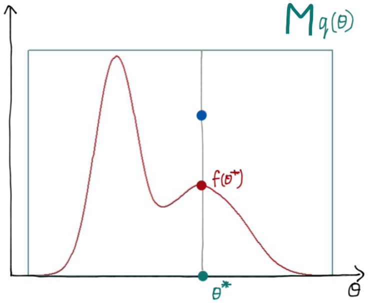



The algorithm proceeds as follows:

1. Sample $\theta^*$ from $q(\theta)$

2. Draw $u \sim Uniform[0, Mq(\theta^*)]$

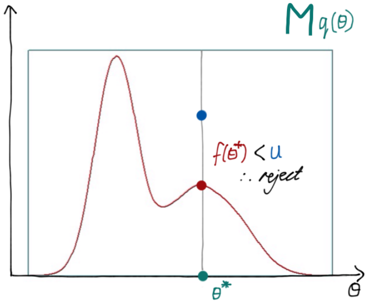
# Rejection sampling




The algorithm proceeds as follows:

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

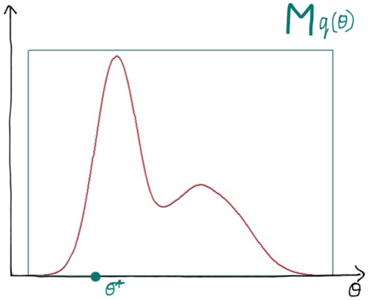
# Rejection sampling




The algorithm proceeds as follows:

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

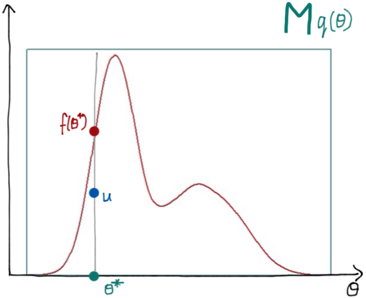
# Rejection sampling




The algorithm proceeds as follows:

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

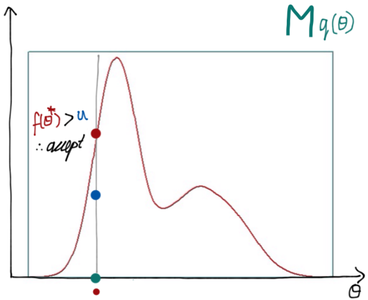
# Rejection sampling




The algorithm proceeds as follows:

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

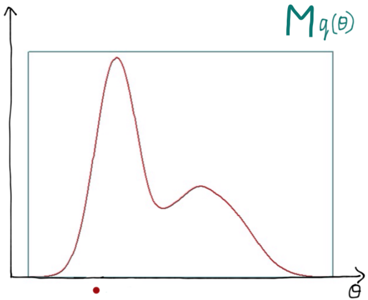
# Rejection sampling




The algorithm proceeds as follows:

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

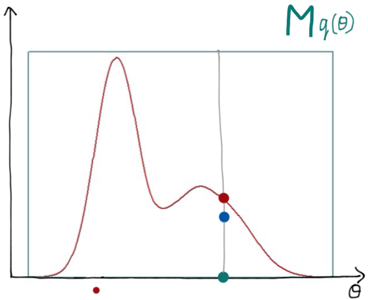
# Rejection sampling




The algorithm proceeds as follows:

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

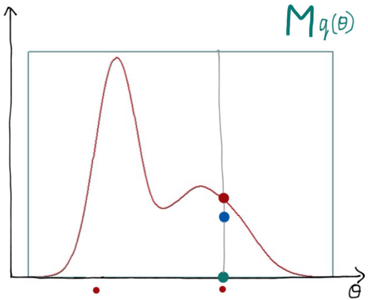
# Rejection sampling




The algorithm proceeds as follows:

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

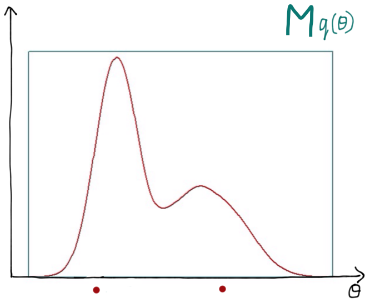
# Rejection sampling




The algorithm proceeds as follows:

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

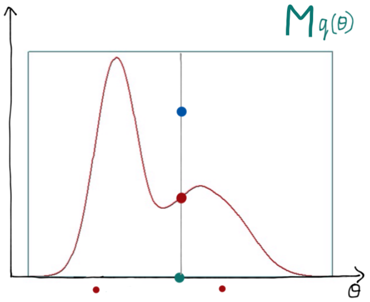
# Rejection sampling




The algorithm proceeds as follows:

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

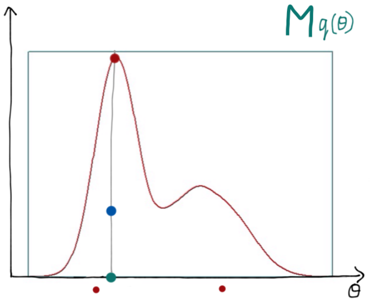
# Rejection sampling




The algorithm proceeds as follows:

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

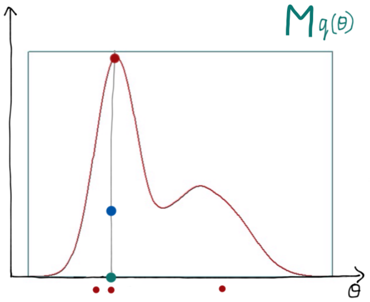
# Rejection sampling




The algorithm proceeds as follows:

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

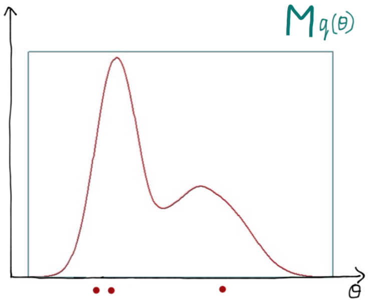
# Rejection sampling




The algorithm proceeds as follows:

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

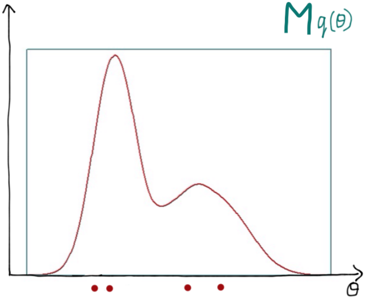
# Rejection sampling




The algorithm proceeds as follows:

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

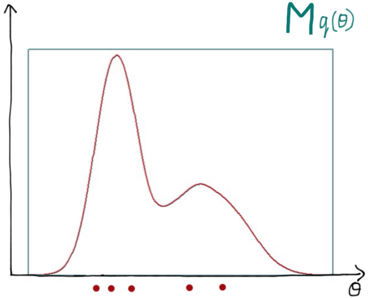
# Rejection sampling




The algorithm proceeds as follows:

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

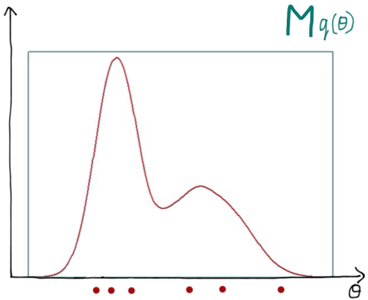
# Rejection sampling




The algorithm proceeds as follows:

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

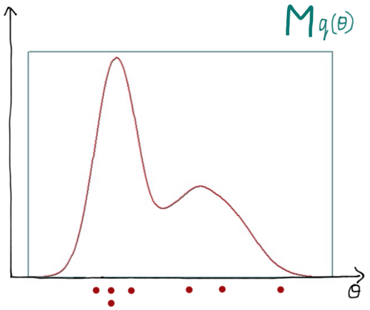
# Rejection sampling




The algorithm proceeds as follows:

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

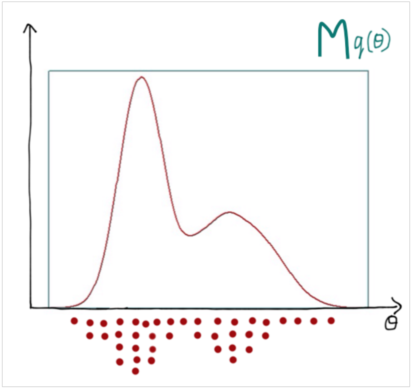
2. Draw $u \sim Uniform[0, Mq(\theta^*)]$

3. Evaluate $f(\theta^*)$

4. If $f(\theta^*) > u$ accept, else reject

5. Repeat steps 1-4

# Rejection sampling

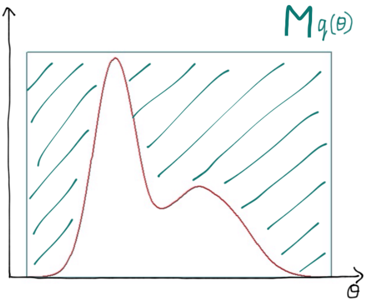



The algorithm proceeds as follows:

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

# Rejection sampling

- Rejection sampling works best if $q(\theta) \approx f(\theta)$ ($M \gtrapprox 1$)
- Acceptance rate of rejection sampler is $\frac{1}{M}$
- Requiring $f(\theta) < Mq(\theta)$ for all $\theta$ 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 $q(\theta)$ such that $f(\theta) < Mq(\theta)$.

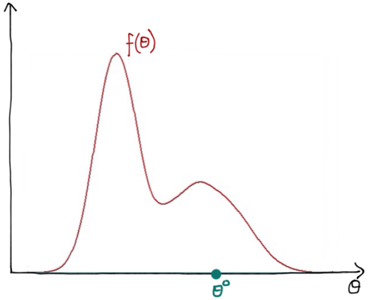
- Rather we build up a chain of samples where each proposed $\theta^*$ depends on the previous one

  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.

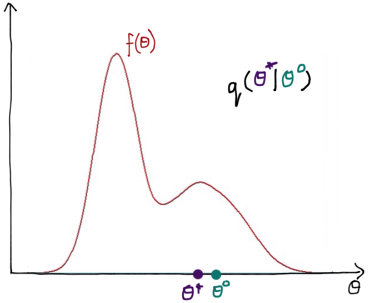
# Metropolis-Hastings




The algorithm proceeds as follows:

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

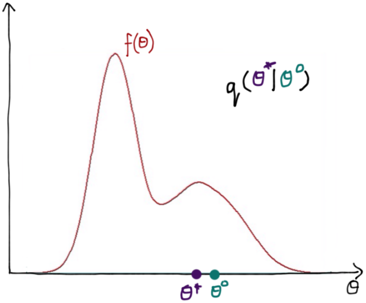
# Metropolis-Hastings




The algorithm proceeds as follows:

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

# Metropolis-Hastings




The algorithm proceeds as follows:

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

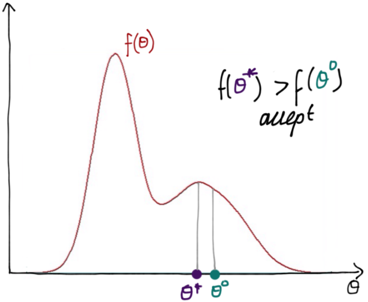
# Metropolis-Hastings

## Acceptance

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

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

- Definitely move to $\theta^*$ if more probable than $\theta$



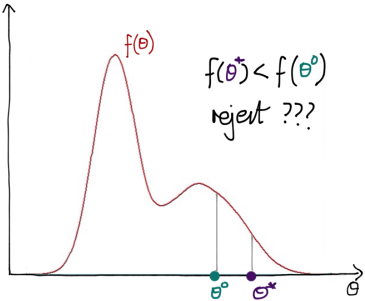
# Metropolis-Hastings

## Acceptance

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

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

- Definitely move to $\theta^*$ if more probable than $\theta$
- May move if $\theta^*$ less probable



# Metropolis-Hastings

## Acceptance

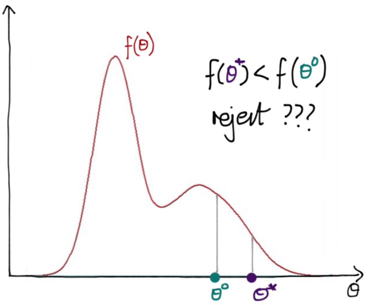
- If $q(\theta^*|\theta)$ symmetric, then

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

- Definitely move to $\theta^*$ if more probable than $\theta$
- May move if $\theta^*$ 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 $\theta^*$ if more probable than $\theta$
- May move if $\theta^*$ less probable

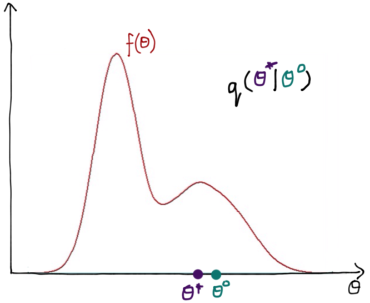
- If $q(\theta^*|\theta)$ asymmetric, then

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

The algorithm proceeds as follows:

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

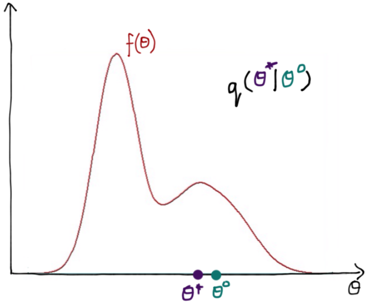
# Metropolis-Hastings




The algorithm proceeds as follows:

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

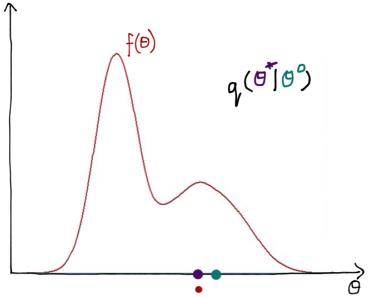
# Metropolis-Hastings




The algorithm proceeds as follows:

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

# Metropolis-Hastings

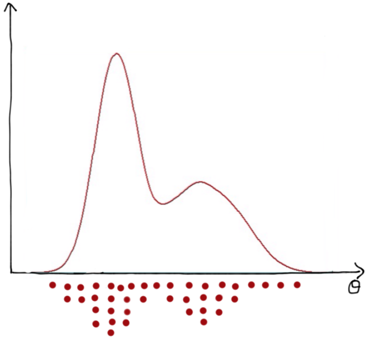



The algorithm proceeds as follows:

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

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

# Metropolis-Hastings




The algorithm proceeds as follows:

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

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

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:

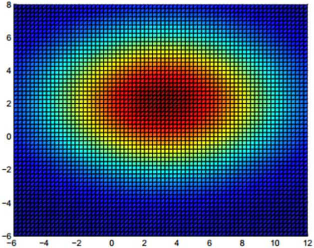
- extend to multivariate inference
- learn about MCMC diagnostics
- think about accuracy and efficiency

# Interlude: Multivariate Gaussian distribution

To infer more multiple parameters we can use multivariate Gaussian

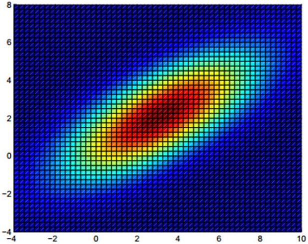
mean $\mu = \begin{bmatrix} 3 & 2 \end{bmatrix}$

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




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

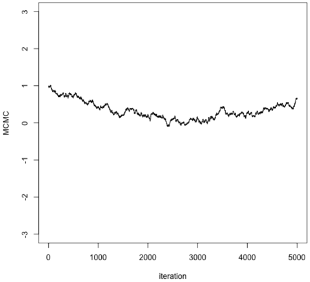
covariance $\Sigma = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix}$

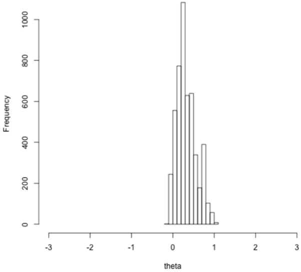



For accurate and efficient MCMC we tune the variance and covariance of the proposal distribution.

# Choosing a proposal distribution

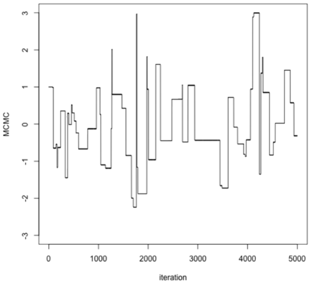
If variance is too small, the chain will be slow to reach the target distribution.

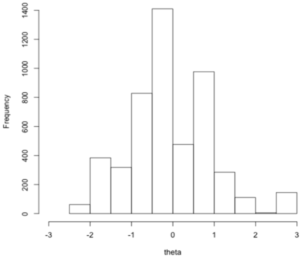


# Choosing a proposal distribution

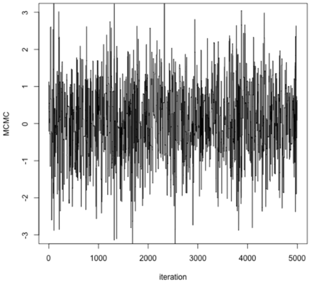
If variance is too high, many proposed values will be rejected and the chain will *stick* in one place for many steps.

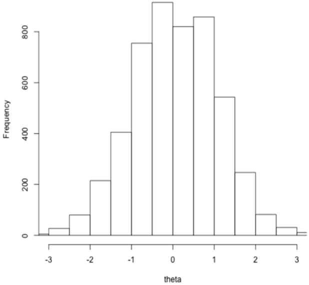


# Choosing a proposal distribution

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



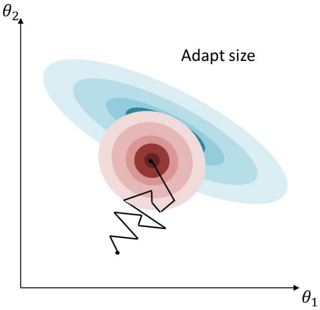

Try several different proposal distributions (pilot runs), aiming for an acceptance rate between 24% and 40%.

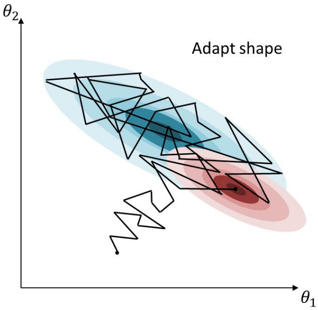
# Adaptive MCMC

- 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

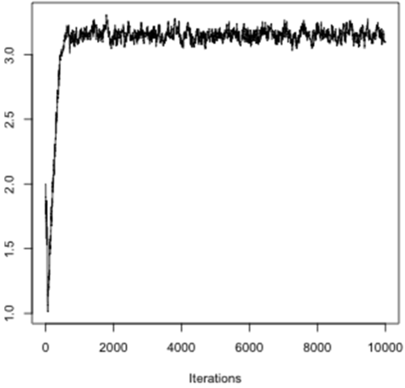
Two-stage adaptation



# 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*.